```
unit ListPointer;

interface
  Uses Tipos, stdctrls, SysUtils, Variants;

Const
  MIN = 1;  // No se utiliza en esta implementacion, se mantiene por compatibilidad
  MAX = 2000; // Corte de Lista Llena
  Nulo= Nil; // Indica posicion invalida de la lista

Type
  PosicionLista = ^NodoLista;  // Puntero a un nodo lista

  NodoLista = Object // Casillero en la Memoria de la lista
    Datos: TipoElemento;
    Ante,Prox: PosicionLista;
  End;

  Lista = Object
    Private
      Inicio, Final: PosicionLista;
      Q_Items: Integer;           // cantidad de elementos de la lista
      TDatoDeLaClave: TipoDatosClave;  // Tipo de dato de la clave
      Size: LongInt;              // Tamaña de la Lista
      // Comportamiento privado
      Procedure Intercambio (P,Q: PosicionLista);
    Public
      // Comportamiento del objeto (Operaciones del TAO)
      Function Crear(avTipoClave: TipoDatosClave; alSize: LongInt): Resultado;
      Function EsVacia(): Boolean;
      Function EsLLena(): Boolean;
      Function Agregar(X:TipoElemento): Resultado;
      Function Insertar(X:TipoElemento; P:PosicionLista): Resultado;
      Function Eliminar(P:PosicionLista): Resultado;
      Function Buscar(X:TipoElemento):PosicionLista;
      Function Siguiente(P:PosicionLista): PosicionLista;
      Function Anterior(P:PosicionLista): PosicionLista;
      Function Ordinal(PLogica: Integer): PosicionLista;
      Function Recuperar(P:PosicionLista): TipoElemento;
      Function Actualizar(X:TipoElemento; P:PosicionLista): Resultado;
      Function ValidarPosicion(P:PosicionLista): Boolean;
      Function RetornarClaves(): String;
      Function LlenarClavesRandom(alSize: LongInt; RangoDesde, RangoHasta: LongInt): Resultado;
      Function Comienzo: PosicionLista;
      Function Fin: PosicionLista;
      Function CantidadElementos: LongInt;
      Function DatoDeLaClave: TipoDatosClave;
      Function SizeList(): LongInt;
      Function MaxSizeList(): LongInt;
      Procedure Sort(Ascendente: Boolean);
  End;


// Escribir la implementacion del Objeto LISTA
implementation

// crea una lista vacia
Function Lista.Crear(avTipoClave: TipoDatosClave; alSize: LongInt): Resultado;
Begin
  if alSize < Min then Crear:= CError;
  if alSize > Max then Crear:= CError;
  if (alSize >= Min) And (alSize <= Max) then Begin
    Inicio := Nulo;
    Final := Nulo;
    Q_Items := 0;
    TDatoDeLaClave := avTipoClave;
    Size := alSize;
    Crear := OK;
  End;
End;

// Control de lista vacia
```

```
Function Lista.EsVacia(): Boolean;
Begin
  EsVacia := (Inicio = Nulo);
End;

// Control de lista llena
Function Lista.EsLLena(): Boolean;
Begin
  EsLLena := (Q_items = Size);
End;

// Agrega un Items al Final de Lista
Function Lista.Agregar(X:TipoElemento): Resultado;
Var Q: PosicionLista;
Begin
  Agregar := CError;
  // Controla que el Tipo de Dato de la Clave sea Homogeneo a la Lista
  if X.TipoDatoClave(X.Clave) <> TDatoDeLaClave then Begin
    Agregar := ClaveIncompatible;
    Exit;
  End;
  // Ahora lo Agrego
  If EsLlena Then Agregar := LLena
  Else Begin
    New(Q); // DM ---> Casillero de la lista (crea el puntero)
    Q^.Datos := X;
    Q^.Prox := Nulo;
    Q^.Ante := Final;
    If EsVacia Then Inicio := Q // Asigno como Primer elemento de la lista
    Else Final^.Prox := Q;      // Apuntamos al ultimo
    Final := Q;                 // Actualizamos el final
    Inc(Q_Items);
    Agregar := OK;
  End;
End;

// Inserta un Item entre el Inicio y el Final de Lista
Function Lista.Insertar(X:TipoElemento; P:PosicionLista): Resultado;
Var Q: PosicionLista;
Begin
  Insertar := CError;
  // Controla que el Tipo de Dato de la Clave sea Homogeneo a la Lista
  if X.TipoDatoClave(X.Clave) <> TDatoDeLaClave then Begin
    Insertar := ClaveIncompatible;
    Exit;
  End;
  // Ahora lo Inserto
  If EsLlena Then Insertar := Llena
  Else Begin
    If ValidarPosicion(P) Then Begin
      New(Q); // DM ---> Casillero de la lista
      Q^.Datos := X;
      Q^.Prox := P;
      Q^.Ante := P^.Ante;
      If P = Inicio Then Inicio := Q // Caso que cambia el Inicio
      Else P^.Ante^.Prox := Q;
      P^.Ante := Q;
      Inc(Q_Items);
      Insertar := OK;
    End
    Else
      Insertar := PosicionInvalida;
  End;
End;

// Retorno la posicion fisica que se corresponde con la logica
// Se debe recorrer desde el inicio de la lista e ir contando las posiciones
Function Lista.Ordinal(PLogica: Integer): PosicionLista;
Var Q: PosicionLista;
I: Integer;
Begin
  Ordinal := Nulo;
```

```
  If Not EsVacia Then Begin
    I := 1;
    Q := Inicio;
    While (I < PLogica) And (Q <> Nulo) Do Begin
      Inc(I);
      Q := Q^.Prox;
    End;
    If Q <> Nulo Then Ordinal := Q;
  End;
End;

// Elimina un elemento de cualquier posicion de la lista.
Function Lista.Eliminar (P:PosicionLista): Resultado;
Var Q: PosicionLista;
Begin
  Eliminar := CError;
  If EsVacia() Then Eliminar := Vacia
  Else Begin
    If ValidarPosicion(P) Then Begin
      Q := P;
      If (P = Inicio) And (P = Final) Then Crear(TDatoDeLaClave, Size) // Unico de la lista. Se cre
a vacia
      Else Begin
        If (P = Inicio) Then Begin // Se elimina el primer elemento. Cambia el inicio
          Inicio := Inicio^.Prox;
          Inicio^.Ante := NULO;
        End
        Else If (P = Final) Then Begin // Se elimina el ultimo elemento. Cambia el Final
              Final := Final^.Ante;
              Final^.Prox := NULO;
            End
            Else Begin // Se elimina en cualquier otro lugar que no es ni Inicio, ni Final
              P^.Ante^.Prox := P^.Prox;
              P^.Prox^.Ante := P^.Ante;
            End;
        Dispose(Q); // Elimino la posicion de memoria
      End;
      Dec(Q_Items);
      Eliminar := OK;
    End
    Else
      Eliminar := PosicionInvalida;
  End;
End;

// Busca un elemento dentro de la lista en función de un dato
// Retorna la posición de la primer ocurrencia o NULO según corresponda
Function Lista.Buscar(X:TipoElemento): PosicionLista;
Var Q: PosicionLista;
    Encontre: Boolean;
Begin
  Buscar := Nulo;
  // Controla que el Tipo de Dato de la Clave sea Homogeneo a la Lista
  if X.TipoDatoClave(X.Clave) <> TDatoDeLaClave then Begin
    Exit;
  End;
  // Arranca por el Inicio y con Q recorre la lista
  Encontre:= False;
  Q := Inicio;
  While (Q <> Nulo) And Not(Encontre) Do Begin
    If Q^.Datos.Clave = X.Clave Then Encontre := True
    Else Q := Q^.Prox;
  End;
  // Controla si lo encontro, si es asi retorno la posicion Q
  If Encontre Then Buscar := Q;
End;

// retorna el siguiente de una Posicion siempre que sea valida
// SINO retorna NULO
Function Lista.Siguiente(P:PosicionLista): PosicionLista;
Begin
  Siguiente := Nulo;
```

```
  If ValidarPosicion(P) Then Siguiente := P^.Prox;
End;

// retorna el anterior de una Posicion siempre que sea valida
// SINO retorna NULO
Function Lista.Anterior(P:PosicionLista): PosicionLista;
Begin
  Anterior := Nulo;
  If ValidarPosicion(P) Then Anterior := P^.Ante;
End;

// retorna un elemento de la lista de la posicion <P>
Function Lista.Recuperar(P:PosicionLista): TipoElemento;
Var X: TipoElemento;
Begin
  Recuperar := X.TipoElementoVacio;
  If ValidarPosicion(P) Then Begin
    Recuperar := P^.Datos;
  End;
End;

// Cambia un Item dentro de la lista
Function Lista.Actualizar(X:TipoElemento; P:PosicionLista): Resultado;
Begin
  Actualizar := CError;
  // Controla que el Tipo de Dato de la Clave sea Homogeneo a la Lista
  if X.TipoDatoClave(X.Clave) <> TDatoDeLaClave then Begin
    Actualizar := ClaveIncompatible;
    Exit;
  End;
  // ahora actualiza
  If ValidarPosicion(P) Then Begin
    P^.Datos := X;
    Actualizar := OK;
  End
  Else
    Actualizar := PosicionInvalida;
End;

// Valida una posicion de la lista
Function Lista.ValidarPosicion(P:PosicionLista): Boolean;
Var Q: PosicionLista;
    Encontre: Boolean;
Begin
  ValidarPosicion := False;
  If Not EsVacia() Then Begin
    Q := Inicio;
    Encontre := False;
    While (Q <> Nulo) And Not(Encontre) Do Begin
      If Q = P Then Encontre := True
      Else Q := Q^.Prox;
    End;
    // Si encontro la posicion retorna verdadero
    If Encontre Then ValidarPosicion := True;
  End;
End;

// Intercambia los datos entre 2 posiciones de la misma lista
Procedure Lista.Intercambio (P,Q: PosicionLista);
Var X1, X2: TipoElemento;
Begin
  X1 := P^.Datos;
  X2 := Q^.Datos;
  P^.Datos := X2;
  Q^.Datos := X1;
End;

// Ordena la lista. Ordena los strings sencible a Mayusculas y Minusculas
Procedure Lista.Sort(Ascendente: Boolean);
Var P, Q: PosicionLista;
    X1, X2: TipoElemento;
Begin
```

```
  // Ordeno por metodo de burbuja
  P := Inicio;
  While P <> Nulo Do Begin
    Q := Inicio;
    While Q <> Nulo Do begin
      X1 := Q^.Datos;
      If Q^.Prox <> Nulo Then Begin
        X2 := Q^.Prox^.Datos;
        if Ascendente then If X1.Clave > X2.Clave Then Intercambio(Q, Q^.Prox);
        if Not Ascendente then If X1.Clave < X2.Clave Then Intercambio(Q, Q^.Prox);
      End;
      Q := Q^.Prox;
    End;
    P := P^.Prox;
  End;
End;

// Retorna un string con todos los elementos de lista
// Cada campo de cada item separado por Tabuladores
// Cada Item separado por Retorno de Carro + Final de Linea
Function Lista.RetornarClaves(): String;
Var Q: PosicionLista;
    X: TipoElemento;
    S, SS: String;
Begin
  SS:= '';
  Q := Inicio;
  While Q <> Nulo Do Begin
    X := Recuperar(Q);
    S := X.ArmarString;
    SS := SS + S + cCRLF;
    Q := Siguiente(Q);
  End;
  RetornarClaves := SS;
End;

Function Lista.Comienzo: PosicionLista;
Begin
  Comienzo := Inicio;
End;

Function Lista.Fin: PosicionLista;
Begin
  Fin := Final;
End;

Function Lista.CantidadElementos: LongInt;
Begin
  CantidadElementos := Q_Items;
End;

// Llena la lista de 0 a <RangoHasta> el atributo DI de la lista
Function Lista.LlenarClavesRandom(alSize: LongInt; RangoDesde, RangoHasta: LongInt): Resultado;
Var X: TipoElemento;
Begin
  TDatoDeLaClave := Numero;
  // Siempre la Crea vacia como Numero
  If Crear(TDatoDeLaClave, alSize) <> OK Then Begin
    LlenarClavesRandom := CError;
    Exit;
  End;
  // Ahora la llana Random
  X.Inicializar(TDatoDeLaClave, '');
  Randomize;
  While Not EsLLena Do Begin
    X.Clave := RangoDesde + Random(RangoHasta);
    Agregar(X);
  End;
  LlenarClavesRandom := OK;
End;

Function Lista.DatoDeLaClave: TipoDatosClave;
```

```
Begin
  DatoDeLaClave := TDatoDeLaClave;
End;

Function Lista.SizeList(): LongInt;
Begin
  SizeList := Size;
End;

Function Lista.MaxSizeList(): LongInt;
Begin
  MaxSizeList := MAX;
End;

end.
```