```
unit Conjuntos;

interface

Uses
  Tipos, ListPointer, stdctrls, SysUtils, Variants;

Type

  Conjunto = Object
    Private
      Items: Lista;
      TDatoDeLaClave: TipoDatosClave;
    Public
      // Comportamiento del objeto (Operaciones)
      Function Crear(avTipoClave: TipoDatosClave; alSize: LongInt): Resultado;
      Function EsVacio(): Boolean;
      Function EsLLeno(): Boolean;
      Function Agregar(X:TipoElemento): Resultado;
      Function Borrar(X:TipoElemento):Resultado;
      Function BorrarPosicion(PLogica: Integer):Resultado;
      Function Recuperar(PLogica: Integer): TipoElemento;
      Function RetornarClaves(): String;
      // operaciones tipicas de conjuntos
      Function Interseccion(Var C: Conjunto): Conjunto;
      Function Union(Var C: Conjunto): Conjunto;
      Function Diferencia(Var C: Conjunto): Conjunto;
      Function Pertenece(X:TipoElemento): Boolean;
      // operaciones generales
      Function LlenarClavesRandom(alSize: LongInt; RangoDesde, RangoHasta: LongInt): Resultado;
      Function CantidadElementos: LongInt;
      Function DatoDeLaClave: TipoDatosClave;
      Function SizeSet(): LongInt;
      Function MaxSizeSet(): LongInt;
      // Ordenar
      Procedure Sort();
  End;


// Escribir la implementación del Objeto CONJUNTO
implementation

// Crea el conjunto vacio
Function Conjunto.Crear(avTipoClave: TipoDatosClave; alSize: LongInt): Resultado;
Begin
  Crear := items.Crear(avTipoClave, alSize);
  TDatoDeLaClave := avTipoClave;
End;

// control de conjunto vacio
Function conjunto.EsVacio(): Boolean;
Begin
  EsVacio := items.EsVacia;
End;

// control de conjunto lleno
Function conjunto.EsLLeno() : Boolean;
Begin
  EsLLeno := items.EsLLena;
End;

// Agrega un elemento al conjunto
Function conjunto.Agregar(X:TipoElemento): Resultado;
Begin
  // controlo el tipo de dato de la clave
  if X.TipoDatoClave (X.Clave) <> TDatoDeLaClave  then Begin
    Agregar := ClaveIncompatible;
    Exit;
  End;
  // Controla que no exista
  if Items.Buscar(X) <> Nulo  then Begin
    Agregar := ClaveDuplicada;
```

```
    Exit;
  End;
  // Ahora lo agrego
  Agregar := items.Agregar(X);
End;

// Busca un elemento dentro del conjunto en función de un dato
// Borra el elemento si lo encuentra
Function conjunto.Borrar(X:TipoElemento): Resultado;
Var Q: PosicionLista;
Begin
  Borrar := Cerror;
  // controlo el tipo de dato de la clave
  if X.TipoDatoClave (X.Clave) <> TDatoDeLaClave  then Begin
    Borrar := ClaveIncompatible;
    Exit;
  End;
  // ahora lo borra
  Q := items.Buscar(X);
  If Q <> Nulo Then Borrar := items.Eliminar(Q);
End;

  // retorna por referencia en X el elemento del conjunto
Function conjunto.Recuperar(PLogica: Integer): TipoElemento;
Var Q: PosicionLista;
    X: TipoElemento;
Begin
  Recuperar := X.TipoElementoVacio;
  Q := items.Ordinal(PLogica);
  If Q <> Nulo Then Recuperar := items.Recuperar(Q);
End;

// Ordena el conjunto. Ordena los strings sencible a Mayusculas y Minusculas
Procedure conjunto.Sort();
Begin
  items.Sort(True);
End;

// Retorna un string con todos los elementos del conjunto
Function conjunto.RetornarClaves():String;
Begin
  RetornarClaves := items.RetornarClaves;
End;

// Cantidad de elementos del conjunto
Function conjunto.CantidadElementos: LongInt;
Begin
  CantidadElementos := items.CantidadElementos;
End;

// Llena el conjunto de 0 a <RangoHasta> el atributo DI de la lista
// Llena el conjunto de 0 a <RangoHasta> el atributo DI de la lista
Function conjunto.LlenarClavesRandom(alSize: LongInt; RangoDesde, RangoHasta: LongInt): Resultado;
Var X: TipoElemento;
Begin
  Randomize;
  TDatoDeLaClave := Numero;
  If Items.Crear(TDatoDeLaClave, alSize) <> OK Then Begin
    LlenarClavesRandom := CError;
    Exit;
  End;
  // Ahora lo lleno random
  X.Inicializar(TDatoDeLaClave, '');
  while Not Items.EsLLena Do Begin
    X.Clave := RangoDesde + Random(RangoHasta);
    if Items.Buscar(X) = Nulo then Begin
      Items.Agregar(X);
    End;
  End;
  LlenarClavesRandom := OK;
End;
```

```
// Borrar por posición lógica dentro del conjunto
Function conjunto.BorrarPosicion(PLogica: Integer):Resultado;
Var Q: PosicionLista;
Begin
  BorrarPosicion := CError;
  Q := items.Ordinal(PLogica);
  If Q <> Nulo Then BorrarPosicion := items.Eliminar(Q);
End;

//-----------------------------------------------------------------
// Funciones típicas de un conjunto
//-----------------------------------------------------------------
// Retorna True si el elemento pertenece al menos 1 vez al conjunto
Function conjunto.Pertenece(X:TipoElemento): Boolean;
Begin
  Pertenece := False;
  If items.Buscar(X) <> Nulo Then Pertenece := True;
End;

// Retorna un conjunto INTERSECCION del mismo con <C>
Function conjunto.Interseccion(Var C: Conjunto): Conjunto;
Var Q : PosicionLista;
    CI: Conjunto;
    X : TipoElemento;
Begin
  CI.Crear(TDatoDeLaClave, SizeSet());
  // Controlo que si uno de los 2 conjuntos es vació no hay intersección
  If items.EsVacia Or C.EsVacio Then Interseccion := CI
  Else Begin
    Q := items.Comienzo;
    While Q <> Nulo Do Begin
      X := items.Recuperar(Q);
      // Lo Busco en C para Saber si esta
      If C.Pertenece(X) = True Then Begin
        CI.Agregar(X);
      End;
      Q := items.Siguiente(Q);
    End;
  End;
  // Retorno el conjunto intersección
  Interseccion := CI;
End;

// Retorna un conjunto UNION del mismo con <C>
Function conjunto.Union(Var C: Conjunto): Conjunto;
Var Q : PosicionLista;
    CU: Conjunto;
    X : TipoElemento;
Begin
  CU.Crear(TDatoDeLaClave, (items.SizeList + C.SizeSet));
  // Paso Todos los elementos de conjunto
  Q := items.Comienzo;
  While Q <> Nulo Do Begin
    X := items.Recuperar(Q);
    CU.Agregar(X);
    Q := items.Siguiente(Q);
  End;
  // Agrego todos los elementos del conjunto C incluso lo repetidos
  Q := C.Items.Comienzo;
  While Q <> Nulo Do Begin
    X := C.Items.Recuperar(Q);
    CU.Agregar(X);
    Q := C.Items.Siguiente(Q);
  End;
  // Retorno el conjunto Unión
  Union := CU;
End;

// Retorna un conjunto con la diferencia del mismo sobre <C>
Function conjunto.Diferencia(Var C: Conjunto): Conjunto;
Var Q : PosicionLista;
    CD: Conjunto;
```

```
    X : TipoElemento;
Begin
  CD.Crear(TDatoDeLaClave, SizeSet());
  // Controlo que si alguno de los conjuntos es vacio
  If items.EsVacia Then Diferencia := CD
  Else Begin
    Q := items.Comienzo;
    While Q <> Nulo Do Begin
      X := items.Recuperar(Q);
      // Lo Busco en C para Saber si esta
      If Not C.Pertenece(X) Then CD.Agregar(X);
      Q := items.Siguiente(Q);
    End;
  End;
  // Retorno el conjunto Diferencia
  Diferencia := CD;
End;

Function Conjunto.DatoDeLaClave: TipoDatosClave;
Begin
  DatoDeLaClave := TDatoDeLaClave;
End;

Function Conjunto.SizeSet(): LongInt;
Begin
  SizeSet := items.SizeList;
End;

Function Conjunto.MaxSizeSet(): LongInt;
Begin
  MaxSizeSet := MAX;
End;

End.
```