```
unit Tipos;

interface
  Uses Variants, SysUtils;

Const
  cTab = Char(9); // Tabulador
  cCR = Char(13); // Retorno de carro
  cCRLF= Char(13) + Char(10); // Retorno de Carro + Fin de Linea
  cLF = Char(10); // Fin de Linea solamente


Type
  // Retorno de las funciones de manejo de las estructuras
  Resultado = (OK, CError, LLena, Vacia, PosicionInvalida, Otro, ClaveIncompatible, ClaveDuplicada)
;

  // Tipos de Datos Soportados x la clave <variant>
  TipoDatosClave = (Numero, Cadena, Fecha, Otros, Desconocido);

  // Tipos de Funcion Hash a Aplicar
  TipoFuncionesHash = (Modulo, Plegamiento, MitadDelCuadrado);

  // Datos a Guardar dentro de las estructuras
  TipoElemento = Object
    Clave: Variant;   // Cualquier valor soportado x <TipoDatosClave>
    Valor1: Variant;  // Cualquier valor soportado x un Variant
    Valor2: Pointer;  // Puntero Generico a Cualquier cosa (dato primitivo, otra estructura, un obj
eto, etc.)
    // Comportamiento basico del <TipoElemento>
    Function ArmarString(): String; // Retorno la Clave como un String
    Function ArmarStringConSeparador(var Separador: String): String; // Retorno la Clave como un St
ring
    Function TipoDatoClave(avClave: Variant): TipoDatosClave;  // Valida el tipo de dato de la clav
e
    Function TipoElementoVacio(): TipoElemento;  // Creo un <TipoElemento> Vacio
    Function EsTEVacio(): Boolean; // Retorno TRUE cuando TE es Vacio
    Procedure Inicializar(aTDC: TipoDatosClave; aValor1Inicial: Variant);  // Para inicilizar segun
 el dato clave
  End;


  // Operaciones del TipoElemento
implementation

  // Arma un String de la clave. Convierte el variant a un string
  Function TipoElemento.ArmarString: String;
  Var  SV: String;
  Begin
    Try
      SV := VarToStr(Clave); // se convierte a string el campo variant sin importar lo que tenga
      SV := SV + cTab + VarToStr(Valor1);  // Convierto en String el Valor 1
      ArmarString := SV;
    except
      ArmarString := '';
    End
  End;

  Function TipoElemento.ArmarStringConSeparador(var Separador: String): String;
  Var  SV: String;
  Begin
    Try
      SV := VarToStr(Clave); // se convierte a string el campo variant sin importar lo que tenga
      SV := SV + Separador + VarToStr(Valor1);  // Convierto en String el Valor 1
      ArmarStringConSeparador := SV;
    except
      ArmarStringConSeparador := '';
    End
  End;

  // Evalua el valor de la clave y retorna el Tipo de Datos del Variant
  Function TipoElemento.TipoDatoClave(avClave: Variant): TipoDatosClave;
```

```
  Var iTipo: Integer;
  Begin
    iTipo := VarType(avClave);

    Case iTipo of
      varEmpty     : TipoDatoClave := Otros;
      varNull      : TipoDatoClave := Otros;
      varAny       : TipoDatoClave := Otros;
      varSmallInt  : TipoDatoClave := Numero;
      varInteger   : TipoDatoClave := Numero;
      varSingle    : TipoDatoClave := Numero;
      varDouble    : TipoDatoClave := Numero;
      varCurrency  : TipoDatoClave := Numero;
      varDate      : TipoDatoClave := Fecha;
      varOleStr    : TipoDatoClave := Otros;
      varDispatch  : TipoDatoClave := Otros;
      varError     : TipoDatoClave := Otros;
      varBoolean   : TipoDatoClave := Otros;
      varVariant   : TipoDatoClave := Otros;
      varUnknown   : TipoDatoClave := Desconocido;
      varShortint  : TipoDatoClave := Numero;
      varByte      : TipoDatoClave := Numero;
      varWord      : TipoDatoClave := Numero;
      varLongWord  : TipoDatoClave := Numero;
      varInt64     : TipoDatoClave := Numero;
      varStrArg    : TipoDatoClave := Cadena;
      varString    : TipoDatoClave := Cadena;
      varArray     : TipoDatoClave := Otros;
      varByRef     : TipoDatoClave := Otros;
      varUString   : TipoDatoClave := Cadena;
      varTypeMask  : TipoDatoClave := Otros;
      else
        TipoDatoClave := Otros;
    end;
  end;

  // Asigna vacio a la clave y valor1,  NIL al puntero generico
  Function TipoElemento.TipoElementoVacio(): TipoElemento;
  Var X: TipoElemento;
  Begin
    X.Clave := '';
    X.Valor1 := '';
    X.Valor2 := NIL;
    TipoElementoVacio := X;
  End;

  // Retorno Verdadero si el TE esta vacio
  Function TipoElemento.EsTEVacio(): Boolean;
  Begin
    EsTEVacio := False;
    Try
      If (Clave = '') AND (Valor1 = '') AND (Valor2 = NIL) Then EsTEVacio := True;
    except
    End;
  End;

  // Inicializa el TE segun el Tipo de Dato Clave.  Solo para numero, string y fechas
  Procedure TipoElemento.Inicializar(aTDC: TipoDatosClave; aValor1Inicial: Variant);
  Begin
    Case aTDC of
      Numero: clave := 0;
      Cadena: clave := '';
      Fecha:  clave := Date;
    End;
    // Seteo el Valor Inicial del Valor 1 y NIL en el puntero
    Valor1:= aValor1Inicial;
    Valor2:= NIL;
  End;

end.
```