```
unit StackPointer;

interface
  Uses Tipos, stdctrls, SysUtils, Variants;

Const
  MIN = 1;    // No se utiliza en este caso pero se mantiene por compatibilidad
  MAX = 2000;   // Control de pila llena
  Nulo= NIL;  // Posicion NO valida del TOPE de la pila

Type
  PosicionPila = ^NodoPila;

  NodoPila = Object
    Datos: TipoElemento;
    Prox: PosicionPila;
  End;

  Pila = Object
    Private
      Inicio: PosicionPila;
      Q_Items: Integer;
      TDatoDeLaClave: TipoDatosClave;
      Size : LongInt;
      Function CantidadElementos(): LongInt;
    Public
      Function Crear(avTipoClave: TipoDatosClave; alSize: LongInt): Resultado;
      Function EsVacia(): Boolean;
      Function EsLlena(): Boolean;
      Function Apilar(X:TipoElemento): Resultado;
      Function DesApilar(): Resultado;
      Function Recuperar(): TipoElemento;
      Function RetornarClaves(): String;
      Function InterCambiar(Var PAux: Pila; bCrearVacia: Boolean): LongInt;
      Function LlenarClavesRandom(alSize: LongInt; RangoDesde, RangoHasta: LongInt): Resultado;
      Function Tope():PosicionPila;
      Function DatoDeLaClave: TipoDatosClave;
      Function SizeStack(): LongInt;
      Function MaxSizeStack(): LongInt;
  End;


implementation

// Crea la Pila Vacia
Function Pila.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;
    Q_Items:= 0;
    TDatoDeLaClave := avTipoClave;
    Size := alSize;
    Crear := OK;
  End;
End;

// control de pila vacia
Function Pila.EsVacia(): Boolean;
Begin
  EsVacia := (Inicio = Nulo);
End;

// control de pila llena
Function Pila.EsLLena(): Boolean;
Begin
  EsLLena := (Q_Items = Size);
End;

// Agrega un elemento a la Pila
Function Pila.Apilar(X:TipoElemento): Resultado;
```

```
Var Q:PosicionPila;
Begin
  Apilar := CError;
  // Controla que el Tipo de Dato de la Clave sea Homogeneo a la Lista
  if X.TipoDatoClave(X.Clave) <> TDatoDeLaClave then Begin
    Apilar := ClaveIncompatible;
    Exit;
  End;
  // Ahora lo apila
  If EsLlena() Then Apilar := Llena
  Else Begin
    New(Q);  // Pido memoria dinamica para almacenar el nodo
    Q^.Prox := Inicio;
    Inicio := Q;
    Q^.Datos := X;
    Inc(Q_Items);
    Apilar := OK;
  End;
End;

// Elimina un elemento de la Pila. Siempre del Tope
Function Pila.DesApilar(): Resultado;
Var Q:PosicionPila;
Begin
  DesApilar := CError;
  If EsVacia() Then DesApilar := Vacia
  Else Begin
    Q := Inicio;
    Inicio := Inicio^.Prox ;
    Dec(Q_Items);
    Dispose(Q);  // Libera el espacio de memoria del nodo
    DesApilar := OK;
  End;
End;

// retorna el elemento del tope de la pila.
Function Pila.Recuperar(): TipoElemento;
Var X: TipoElemento;
Begin
  Recuperar := X.TipoElementoVacio;
  If Not EsVacia() Then
  Begin
    Recuperar := Inicio^.Datos ;
  End;
End;

// Pasa los Items de una Pila Auxiliar a la Pila "Pila"
Function Pila.InterCambiar(Var PAux: Pila; bCrearVacia: Boolean): LongInt;
Var X: TipoElemento;
    I: LongInt;
Begin
  I := 0;
  If bCrearVacia = true Then Crear(TDatoDeLaClave, Size);
  While Not PAux.EsVacia() Do Begin
    X := PAux.Recuperar();
    If Apilar(X) = OK Then Inc(I);
    PAux.DesApilar;
  End;
  InterCambiar := I;
End;

// Retorna un string con todos los elementos de Pila
// Cada elemento separado por Retorno de Carro + Final de Linea
Function Pila.RetornarClaves():String;
Var X: TipoElemento;
    S, SS: String;
    PAux: Pila ;
Begin
  SS:= '';
  Paux.Crear(TDatoDeLaClave, Size);
  While Not EsVacia() Do Begin
    X := Recuperar();
```

```
    Paux.Apilar(X);
    S := X.ArmarString;
    SS := SS + S + cCRLF;
    DesApilar();
  End;
  InterCambiar(Paux, True);
  RetornarClaves := SS;
End;

// Llena la pila con valores random en el atributo DI
// desde <RangoDesde>  hasta <RangoHasta>
Function Pila.LlenarClavesRandom(alSize: LongInt; RangoDesde, RangoHasta: LongInt): Resultado;
Var X: TipoElemento;
Begin
  Randomize;
  TDatoDeLaClave := Numero;
  If Crear(TDatoDeLaClave, alSize) <> OK Then Begin
    LlenarClavesRandom := CError;
    Exit;
  End;
  // Ahora la lleno random
  X.Inicializar(TDatoDeLaClave,'');
  While Not EsLLena() Do Begin
    X.Clave := RangoDesde + Random(RangoHasta);
    Apilar(X);
  End;
  LlenarClavesRandom := OK;
End;

Function Pila.Tope():PosicionPila;
Begin
  Tope := Inicio;
End;

Function Pila.CantidadElementos(): LongInt;
Begin
  CantidadElementos := Q_Items;
End;

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

Function Pila.SizeStack(): LongInt;
Begin
  SizeStack := Size;
End;

Function Pila.MaxSizeStack(): LongInt;
Begin
  MaxSizeStack := MAX;
End;

end.
```