```
Unit QueuesCursor;

Interface
Uses
  Tipos, stdctrls, SysUtils, Variants;

Const
  MIN = 1;  // Inicio del cursor
  MAX = 2000; // fin del cursor
  NULO= 0;  // Indica posicion invalida de la cola

Type
  PosicionCola = LongInt;

  Nodo_Cola = Object
    Datos: Tipoelemento;
    Prox: PosicionCola;
  End;

  Cola = Object
    Private
      Cursor: Array Of Nodo_Cola;
      Inicio, Fin, Libre: PosicionCola;
      Q_Items: LongInt;
      TDatoDeLaClave: TipoDatosClave;
      Size : LongInt;
      Function CantidadItems(): LongInt;
    Public
      Function Crear(avTipoClave: TipoDatosClave; alSize: LongInt): Resultado;
      Function EsVacia(): Boolean;
      Function EsLlena(): Boolean;
      Function Encolar(X:TipoElemento): Resultado;
      Function DesEncolar(): Resultado;
      Function Recuperar(): TipoElemento;
      Function RetornarClaves(): String;
      Function InterCambiar(Var CAux: Cola; bCrearVacia: Boolean): LongInt;
      Function LLenarClavesRandom(alSize: LongInt; RangoDesde, RangoHasta:LongInt): Resultado;
      Function Frente(): PosicionCola;
      Function Final(): PosicionCola;
      Function DatoDeLaClave: TipoDatosClave;
      Function SizeQueue(): LongInt;
      Function MaxSizeQueue(): LongInt;
  End;


Implementation

// crea una cola vacia
Function Cola.Crear(avTipoClave: TipoDatosClave; alSize: LongInt): Resultado;
Var Q: PosicionCola;
Begin
  if alSize < Min then Crear:= CError;
  if alSize > Max then Crear:= CError;
  if (alSize >= Min) And (alSize <= Max) then Begin
    SetLength(Cursor, (alSize+1));
    For Q := MIN To (alSize - 1) Do // Encadenamientos de Libres
      Cursor[Q].Prox := Q + 1;
    Cursor[alSize].Prox := NULO;
    Libre := MIN;
    Inicio := NULO;
    Fin := NULO;
    Q_items := 0;
    TDatoDeLaClave := avTipoClave;
    Size := alSize;
    Crear := OK;
  End;
End;

// control de la cola vacia
Function Cola.EsVacia(): Boolean;
Begin
  EsVacia := (Inicio = NULO);
```

```
End;

// control de la cola llena
Function Cola.EsLlena(): Boolean;
Begin
  EsLlena := (Libre = NULO);
End;

// Agrega elementos al final de la Cola. Despues del ultimo.
Function Cola.Encolar(X:TipoElemento): Resultado;
Var Q: PosicionCola;
Begin
  Encolar := CError;
  // Controla que el Tipo de Dato de la Clave sea Homogeneo a la Lista
  if X.TipoDatoClave(X.Clave) <> TDatoDeLaClave then Begin
    Encolar := ClaveIncompatible;
    Exit;
  End;
  // Ahora Encolo
  If Not(Esllena()) Then Begin
    Q := Libre; // Tomo el primer libre disponible
    Libre := Cursor[Libre].Prox; // Encadeno el resto de los libres
    Cursor[Q].Datos := X;
    Cursor[Q].Prox := NULO;
    If EsVacia() Then Inicio := Q // Controlo si es el primer elemento de la lista
    Else Cursor[Fin].Prox := Q;
    Fin := Q;
    Inc(Q_items);
    Encolar := OK;
  End
  Else
    Encolar := Llena;
End;

// Elimina un elemento de la cola
Function Cola.DesEncolar(): Resultado;
Var Q: PosicionCola;
Begin
  DesEncolar := CError;
  If EsVacia() Then DesEncolar := Vacia
  Else Begin
    If (Inicio = Fin) Then Crear(TDatoDeLaClave, Size) // Unico de la cola. Se crea vacia
    Else Begin
      Q := Inicio;
      Inicio := Cursor[Inicio].Prox;
      Cursor[Q].Prox := Libre; // retorno a Libres el Nodo Eliminado
      Libre := Q;
      Dec(Q_Items);
    End;
    DesEncolar := OK;
  End;
End;

// retorna el elemento del frente de la cola
Function Cola.Recuperar(): TipoElemento;
Var X: TipoElemento;
Begin
  Recuperar := X.TipoElementoVacio;
  If Not EsVacia() Then
  Begin
    Recuperar := Cursor[Inicio].Datos;
  End;
End;

// Pasa los elementos de una Cola Auxiliar a la Cola "Cola"
Function Cola.InterCambiar(Var CAux: Cola; bCrearVacia: Boolean): LongInt;
Var X: TipoElemento;
    I: Integer;
Begin
  I:= 0;
  If bCrearVacia = true Then Crear(TDatoDeLaClave, CAux.Size);
  While Not CAux.EsVacia() Do Begin
```

```
    X := CAux.Recuperar();
    If Encolar(X) = OK Then Inc(I);
    CAux.DesEncolar;
  End;
  InterCambiar := I;
End;

// Retorna un string con todos los elementos de Cola
// Cada Item separado por Retorno de Carro + Final de Linea
Function Cola.RetornarClaves():String;
Var X: TipoElemento;
  S, SS: String;
  CAux: Cola;
Begin
  SS:= '';
  CAux.Crear(TDatoDeLaClave, Size);
  While Not EsVacia() Do Begin
    X := Recuperar();
    CAux.Encolar(X);
    S := X.ArmarString;
    SS := SS + S + cCRLF;
    DesEncolar();
  End;
  InterCambiar(Caux, True);
  RetornarClaves := SS;
End;

// Llena la cola con valores aletarios en el atributo DI
// desde <RangoDesde> hasta <RangoHasta>
Function Cola.LLenarClavesRandom(alSize: LongInt; RangoDesde, RangoHasta:LongInt): Resultado;
Var X: TipoElemento;
Begin
  TDatoDeLaClave := Numero;
  If Crear(TDatoDeLaClave, alSize) <> OK Then Begin
    LLenarClavesRandom := CError;
    Exit;
  End;
  X.Inicializar(TDatoDeLaClave,'');
  Randomize;
  While Not EsLlena Do Begin
    X.Clave := RangoDesde + Random(RangoHasta);
    Encolar(X);
  End;
  LLenarClavesRandom := OK;
End;

// retorno posicion del frente
Function Cola.Frente(): PosicionCola;
Begin
  Frente := Inicio;
End;

// retorno posicion del final
Function Cola.Final(): PosicionCola;
Begin
  Final := Fin;
End;

// Retorno la cantidad de elementos
Function Cola.CantidadItems(): LongInt;
Begin
  CantidadItems := Q_Items;
End;

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

Function Cola.SizeQueue(): LongInt;
Begin
  SizeQueue := Size;
```

```
End;

Function Cola.MaxSizeQueue(): LongInt;
Begin
  MaxSizeQueue := MAX;
End;

End.
```