Sort TStrings objects with extra functionality ala UNIX style parameters

TStringList has a *Sort* method and a *Sorted* property. This feature is not available in *TStrings*.

This class allows sorting of *TString* objects with extra functionality ala UNIX style parameters. (Yes I know UNIX is a four letter word but they do have some neat features). The SORT algorythm utilizes the QUICK SORT method. The features I have implemented are

Options:

- Sort descending – *srtDescending*
- Treat sort field as numeric – *srtEvalNumeric*
- Ignore leading blanks in field – *srtIgnoreBlank*
- Ignore case of field – *srtIgnoreCase*

Switches:

- `-k` – Start,End position of substring for search
- `-f` – Field number of a delimited string (Zero column based)
- `-d` – Character delimiter for -f switch (Default = SPACE)

In it's simplest form the class just sorts the *TStrings* ascending e.g.

```
SuperSort.SortStrings(Memo1.Lines,[]);
```

Assume a semi-colon delimited list like:

```
'Mike;34;Green'
'harry;25;Red'
'Jackie;6;Black'
'Bazil;9,Pink'
'john;52;Blue'
```

To sort this list DESCENDING on AGE (Field 1) and ignore case:

```
SuperSort(MyStrings, ['-f 1','-d ;'],
  [srtDescending,srtEvalNumeric,srtIgnoreCase]);
```

Assume a string list of:

```
'1999 12 20 AA432 Comment 1'
'2002 10 12 SWA12 Some other words'
'1998 09 11 BDS65 And so on and so on'
```

To sort this list on ITEM CODE (Positions 12 to 17) with no options

```
SuperSort(MyStrings,['-k 12,17']);
```

Methods:

```
procedure SortStrings(StringList: TStrings; Switches: array of string;
  Options: TSuperSortOptionSet = []);
```

Switches is a string array of -k,-d and -f settings. If it is set to empty array [] then NO switches are active. Options is an OPTIONAL set of [srtDescending,srtIgnoreCase,srtIgnoreBlank,srtEvalNumeric] The default is empty set [].

Properties:

```
SortTime : TDateTime;
```

Returns the time taken for the sort for stats purposes.

Usage Example:

```
uses
  SuperSort;
```

```
procedure TForm1.Test;
var
  Srt: TSuperSort;
begin
  Srt := TSuperSort.Create;
  Srt.SortStrings(Memo1.Lines,[],[srtIgnoreBlank]);
  Label1.Caption := 'Time : ' + FormatDateTine('hh:nn:ss:zzz',Srt.SortTime);
  Srt.Free;
end;
```

The unit.

```
unit SuperSort;

interface

uses
  Classes,SysUtils;

// ============================================================================
// Class TSuperSort
// Mike Heydon Nov 2002
//
// Sort class that implements Unix style sorts including ..
//
// SWITCHES
// --------
// -k [StartPos,EndPos]  - Keyfield to sort on. Start and End pos in string
// -d [Field Delimiter]  - Delimter to use with -f switch. default = SPACE
// -f [FieldNumber]      - Zero based field number delimeted by -d
//
// OPTIONS SET
// ============
// srtDescending         - Sort descending
// srtIgnoreCase         - Ignore case when sorting
// srtIgnoreBlank        - Ignore leading blanks
// srtEvalNumeric        - Treat sort items as NUMERIC
//
// ============================================================================

type
  // Sort Options
  TSuperSortOptions = (
    srtDescending,srtIgnoreCase, srtIgnoreBlank,srtEvalNumeric
  );
  TSuperSortOptionSet = set of TSuperSortOptions;

  // ============
  // TSuperSort
  // ============
  TSuperSort = class(TObject)
  protected
    function GetKeyString(const Line : string) : string;
    procedure QuickSortStrA(SL : TStrings);
    procedure QuickSortStrD(SL : TStrings);
    procedure ResolveSwitches(Switches : array of string);
  private
    FSortTime : TDateTime;
    FIsSwitches,
    FIsPositional,
    FIsDelimited,
    FDescending,
    FIgnoreCase,
    FIgnoreBlank,
    FEvalDateTime,
    FEvalNumeric : boolean;
    FFieldNum,
    FStartPos,FEndPos : integer;
    FDelimiter : char;
  public
    procedure SortStrings(StringList : TStrings; Switches : array of string;
      Options : TSuperSortOptionSet = []);
```

```
    property SortTime : TDateTime read FSortTime;
  end;

// ---------------------------------------------------------------------------
implementation

const
  BLANK    = -1;
  EMPTYSTR = '';

// ================================================
// INTERNAL CALL
// Resolve switches and set internal variables
// ================================================

procedure TSuperSort.ResolveSwitches(Switches : array of string);
var
  i : integer;
  Sw,Data : string;
begin
  FStartPos := BLANK;
  FEndPos := BLANK;
  FFieldNum := BLANK;
  FDelimiter := ' ';
  FIsPositional := false;
  FIsDelimited := false;

  for i := Low(Switches) to High(Switches) do
  begin
    Sw := trim(Switches[i]);
    Data := trim(copy(Sw,3,1024));
    Sw := UpperCase(copy(Sw,1,2));

    // Delimiter
    if Sw = '-D' then
    begin
      if length(Data) > 0 then FDelimiter := Data[1];
    end;

    // Field Number
    if Sw = '-F' then
    begin
      FIsSwitches := true;
      FIsDelimited := true;
      FFieldNum := StrToIntDef(Data,BLANK);
      Assert(FFieldNum <> BLANK,'Invalid -f Switch');
    end;

    // Positional Key
    if Sw = '-K' then
    begin
      FIsSwitches := true;
      FIsPositional := true;
      FStartPos := StrToIntDef(trim(copy(Data,1,pos(',',Data) - 1)),BLANK);
      FEndPos := StrToIntDef(trim(copy(Data,pos(',',Data) + 1,1024)),BLANK);
      Assert((FStartPos <> BLANK) and (FEndPos <> Blank),'Invalid -k Switch');
    end;
  end;
end;

// ================================================
// INTERNAL CALL
// Resolve the Sort Key part of the string based on
// the Switches parameters
// ================================================

function TSuperSort.GetKeyString(const Line : string) : string;
var
  Key : string;
  Numvar : double;
  DCount, i, DPos : integer;
  Tmp : string;
begin
  // Default
```

```
  Key := Line;
  // Extract Key from switches -k takes precedence
  if FIsPositional then
    Key := copy(Key,FStartPos,FEndPos)
  else
    if FIsDelimited then
    begin
      DPos := 0;
      DCount := 0;
      for i := 1 to length(Key) do
      begin
        if Key[i] = FDelimiter then
         inc(DCount);
        if DCount = FFieldNum then
        begin
           if FFieldNum = 0 then
             DPos := 1
           else
             DPos := i + 1;
          break;
        end;
      end;

      if DCount < FFieldNum then
        // No such Field Number
        Key := EMPTYSTR
      else
      begin
        Tmp := copy(Key,DPos,4096);
        DPos := pos(FDelimiter,Tmp);
        if DPos = 0 then
          Key := Tmp
        else
          Key := copy(Tmp,1,DPos - 1);
       end;
    end;

  // Resolve Options
  if FEvalNumeric then
  begin
    Key := trim(Key);
    // Strip any commas
    for i := length(Key) downto 1 do
      if Key[i] = ',' then delete(Key,i,1);
    try
      Numvar := StrToFloat(Key);
    except
      Numvar := 0.0;
    end;
    Key := FormatFloat('############0.000000',Numvar);
    // Leftpad num string
    Key := StringOfChar('0',20 - length(Key)) + Key;
  end;

  // Ignores N/A for Numeric and DateTime
  if not FEvalNumeric and not FEvalDateTime then
  begin
    if FIgnoreBlank then Key := trim(Key);
    if FIgnoreCase then Key := UpperCase(Key);
  end;

  Result := Key;
end;

// ============================================
// INTERNAL CALL
// Recursive STRING quick sort routine ASCENDING.
// ============================================

procedure TSuperSort.QuickSortStrA(SL : TStrings);

  procedure Sort(l,r : integer);
  var
    i, j : integer;
```

```
    x, Tmp : string;
  begin
    i := l;
    j := r;
    x := GetKeyString(SL[(l + r) div 2]);

    repeat
      while GetKeyString(SL[i]) < x do
        inc(i);
      while x  < GetKeyString(SL[j]) do
        dec(j);
      if i <= j then
      begin
        Tmp := SL[j];
        SL[j] := SL[i];
        SL[i] := Tmp;
        inc(i);
        dec(j);
      end;
    until i > j;

    if l < j then
      Sort(l,j);
    if i < r then
      Sort(i,r);
  end;

begin
  if SL.Count > 0 then
  begin
    SL.BeginUpdate;
    Sort(0,SL.Count - 1);
    SL.EndUpdate;
  end;
end;

// ============================================
// INTERNAL CALL
// Recursive STRING quick sort routine DECENDING
// ============================================

procedure TSuperSort.QuickSortStrD(SL : TStrings);

  procedure Sort(l,r : integer);
  var
    i, j : integer;
    x, Tmp : string;
  begin
    i := l;
    j := r;
    x := GetKeyString(SL[(l + r) div 2]);

    repeat
      while GetKeyString(SL[i]) > x do
        inc(i);
      while x  > GetKeyString(SL[j]) do
        dec(j);
      if i <= j then
      begin
        Tmp := SL[j];
        SL[j] := SL[i];
        SL[i] := Tmp;
        inc(i);
        dec(j);
      end;
    until i > j;

    if l < j then
      Sort(l,j);
    if i < r then
      Sort(i,r);
  end;

begin
```

```
  if SL.Count > 0 then
  begin
    SL.BeginUpdate;
    Sort(0,SL.Count - 1);
    SL.EndUpdate;
  end;
end;

// ====================
// Sort a stringlist
// ====================

procedure TSuperSort.SortStrings(StringList : TStrings;
  Switches : array of string; Options : TSuperSortOptionSet = []);
var
  StartTime : TDateTime;
begin
  StartTime := Now;
  FDescending := (srtDescending in Options);
  FIgnoreCase := (srtIgnoreCase in Options);
  FIgnoreBlank := (srtIgnoreBlank in Options);
  FEvalNumeric := (srtEvalNumeric in Options);
  ResolveSwitches(Switches);

  if FDescending then
    QuickSortStrD(StringList)
  else
    QuickSortStrA(StringList);

  FSortTime := Now - StartTime;
end;

end.
```

Author:	Mike Heydon
Contributor:	Loris Luise
Added:	2010-12-17
Last updated:	2010-12-17