```
unit U_TimeFunctions;

interface

uses
  U_TimeFunctionsConst;

type

  TDateAndTime = TDateTime;
  TSeasonsChange = (scSummer, scWinter, scWinterSummer, scSummerWinter);

  function _DateTimeToStr(DateAndTime: TDateAndTime): string;
  function _StrToDateTime(DateAndTime: string): TDateAndTime;
  function GetSeasonType(DateAndTime: TDateAndTime): TSeasonsChange;
  function GetSeason(DateAndTime: TDateAndTime): boolean;
  function GetSeasonChangeWinterSummer(BeginDate, EndDate: TDateAndTime): cardinal;
  function GetSeasonChangeSummerWinter(BeginDate, EndDate: TDateAndTime): cardinal;
  function EQ_DT(D1, D2: TDateAndTime): boolean;
  function NE_DT(D1, D2: TDateAndTime): boolean;
  function GT_DT(D1, D2: TDateAndTime): boolean;
  function GE_DT(D1, D2: TDateAndTime): boolean;
  function LT_DT(D1, D2: TDateAndTime): boolean;
  function LE_DT(D1, D2: TDateAndTime): boolean;
  function Max_DT(D1, D2: TDateAndTime): TDateAndTime;
  function Min_DT(D1, D2: TDateAndTime): TDateAndTime;
  function toTime(Hour, Min: cardinal): TDateAndTime; overload;
  function toTime(Min: cardinal): TDateAndTime; overload;
  function TimeFromDate(DT: TDateAndTime): TDateAndTime;
  function incTime(DT: TDateAndTime; Min: cardinal): TDateAndTime;
  function RealDT(DT: TDateAndTime): TDateAndTime;
  function DTTS(DateAndTime: TDateAndTime): string;
  function DT_IsCorrectRange(BeginDate, EndDate: TDateAndTime; DeltaTime: cardinal): boolean;
  function DT_GetCorrectRange(BeginDate, EndDate: TDateAndTime; DeltaTime: cardinal): TDateAndTime;


implementation

uses
  DateUtils, SysUtils,
  Classes;

type
  TGSTArrayRecord = record
    DT: TDateAndTime;
    SC: TSeasonsChange;
  end;

const
  GSTArrayMax = 10;

var
  GSTArray1: array[1..GSTArrayMax] of TGSTArrayRecord;
  GSTIndex: integer;
  i: integer;

//---------------------------------------------------------------------------//

function TimeFromDate(DT: TDateAndTime): TDateAndTime;
begin
  result := DT - trunc(DT);
end;

function EQ_DT(D1, D2: TDateAndTime): boolean;        //==
var
  c1, c2: cardinal;
begin
  c1 := cardinal(trunc(100000 * D1));
  c2 := cardinal(trunc(100000 * D2));
  if (c1 > c2)
    then result := c1 - c2 < 12
    else result := c2 - c1 < 12;
end;

function NE_DT(D1, D2: TDateAndTime): boolean;       //<>
begin
  result := not(EQ_DT(D1, D2));
end;
```

```
function GT_DT(D1, D2: TDateAndTime): boolean;      //>
begin
  result := (D1 > D2) and NE_DT(D1, D2);
end;

function GE_DT(D1, D2: TDateAndTime): boolean;      //>=
begin
  result := (D1 >= D2) or EQ_DT(D1, D2);
end;

function LT_DT(D1, D2: TDateAndTime): boolean;      //<
begin
  result := (D1 < D2) and NE_DT(D1, D2);
end;

function LE_DT(D1, D2: TDateAndTime): boolean;       //<=
begin
  result := (D1 <= D2) or EQ_DT(D1, D2);
end;

function Max_DT(D1, D2: TDateAndTime): TDateAndTime;
begin
  if (GT_DT(D1,D2)) then result := D1 else result := D2;
end;

function Min_DT(D1, D2: TDateAndTime): TDateAndTime;
begin
  if (LT_DT(D1,D2)) then result := D1 else result := D2;
end;

function _DateTimeToStr(DateAndTime: TDateAndTime): string;
var
  FS: TFormatSettings;
begin
  GetLocaleFormatSettings(0, FS);
  FS.ShortDateFormat := 'mm.dd.yyyy hh:mm:ss';
  FS.LongDateFormat := '';
  FS.ShortTimeFormat := '';
  FS.LongTimeFormat := '';
  FS.DecimalSeparator := ' ';
  FS.TimeSeparator := ':';
  FS.DateSeparator := '.';
  FS.ListSeparator := ' ';
  result := DateTimeToStr(DateAndTime, FS);
end;


function DTTS(DateAndTime: TDateAndTime): string;
var
  FS: TFormatSettings;
begin
  GetLocaleFormatSettings(0, FS);
  FS.ShortDateFormat := 'dd.mm.yyyy hh:mm:ss';
  FS.LongDateFormat := '';
  FS.ShortTimeFormat := '';
  FS.LongTimeFormat := '';
  FS.DecimalSeparator := ' ';
  FS.TimeSeparator := ':';
  FS.DateSeparator := '.';
  FS.ListSeparator := ' ';
  result := DateTimeToStr(DateAndTime, FS);
end;


function _StrToDateTime(DateAndTime: string): TDateAndTime;
var
  FS: TFormatSettings;
begin
  GetLocaleFormatSettings(0, FS);
  FS.ShortDateFormat := 'mm.dd.yyyy hh:mm:ss';
  FS.LongDateFormat := 'mm.dd.yyyy hh:mm:ss';
  FS.ShortTimeFormat := 'mm.dd.yyyy hh:mm:ss';
  FS.LongTimeFormat := 'mm.dd.yyyy hh:mm:ss';
  FS.DecimalSeparator := ' ';
  FS.TimeSeparator := ':';
  FS.DateSeparator := '.';
  FS.ListSeparator := ' ';
```

```
  result := StrToDateTime(DateAndTime, FS);
end;

function GetSeasonChangeWinterSummer(BeginDate, EndDate: TDateAndTime): cardinal;
var
  i: integer;
begin
  result := 0;
  for i := trunc(BeginDate) to trunc(EndDate) do
    if (GetSeasonType(i) = scWinterSummer) then result := result + 1;
end;

function GetSeasonChangeSummerWinter(BeginDate, EndDate: TDateAndTime): cardinal;
var
  i: integer;
begin
  result := 0;
  for i := trunc(BeginDate) to trunc(EndDate) do
    if (GetSeasonType(i) = scSummerWinter) then result := result + 1;
end;

function toTime(Hour, Min: cardinal): TDateAndTime; overload;
begin
  //result := (Hour * 3600 + Min * 60) / (3600 * 24);
  result := DTHourMinutesArray[Hour, Min];
end;

function toTime(Min: cardinal): TDateAndTime; overload;
begin
  //result := (Hour * 3600 + Min * 60) / (3600 * 24);
  result := DTMinutesArray[Min];
end;

function incTime(DT: TDateAndTime; Min: cardinal): TDateAndTime;
var
  H, M, S, MS: word;
  dDay, FMin: cardinal;
begin
  DecodeTime(DT, H, M, S, MS);
  FMin := (60 * H + M) + Min;
  dDay := FMin div 1440; //1440 = 24 * 60
  FMin := FMin - dDay * 1440;
  result := trunc(DT + DTSec) + dDay + toTime(FMin);
end;

function GetSeason(DateAndTime: TDateAndTime): boolean;
var
  ST: TSeasonsChange;
begin
  ST := GetSeasonType(DateAndTime);
  result := true;
  case ST of
    scWinter: result := true;
    scSummer: result := false;
    scWinterSummer: if (LE_DT(DateAndTime, toTime(02, 00))) then result := true else result := fals
    scSummerWinter: if (LE_DT(DateAndTime, toTime(02, 00))) then result := false else result := tru
  end;
end;

function RealDT(DT: TDateAndTime): TDateAndTime;
begin
  if (GetSeason(DT) = true)
    then result := DT
    else result := DT - toTime(01, 00);
end;

function DT_IsCorrectRange(BeginDate, EndDate: TDateAndTime; DeltaTime: cardinal): boolean;
var
  Days, Mins: integer;
  DDate, tDate: TDateAndTime;
begin
  result := false;
  if (EQ_DT(BeginDate, EndDate)) then exit;
  if (EndDate < BeginDate) then exit;
  DDate := EndDate - BeginDate;
  Days := trunc(DDate);
  Mins := trunc( (DDate - trunc(DDate)) * 24 * 60 );
  Mins := Days * 24 * 60 + Mins;
```

```
  result := (Mins mod DeltaTime) = 0;
end;

function DT_GetCorrectRange(BeginDate, EndDate: TDateAndTime; DeltaTime: cardinal): TDateAndTime;
var
  Days, Mins, delta: integer;
  DDate, tDate: TDateAndTime;
begin
  result := EndDate;
  if (EQ_DT(BeginDate, EndDate)) then exit;
  if (EndDate < BeginDate) then exit;
  DDate := EndDate - BeginDate;
  Days := trunc(DDate);
  Mins := trunc( (DDate - trunc(DDate)) * 24 * 60 );
  Mins := Days * 24 * 60 + Mins;
  delta := Mins - (Mins div DeltaTime) * DeltaTime;
  result := EndDate + toTime(delta);
end;


{$OPTIMIZATION ON}

function GetSeasonType(DateAndTime: TDateAndTime): TSeasonsChange;
Var
  year, i: integer;
  tmpDate, wtmpDate, stmpDate: TDateTime;
begin
  //
  DateAndTime := trunc(DateAndTime);
  //
  //Cash function BEGIN
  for i := 1 to GSTIndex do begin
    if (GSTArray1[i].DT = DateAndTime) then begin
      result := GSTArray1[i].SC;
      exit;
    end;
  end;
  //Cash function END
  //
  wtmpDate := 0;
  year := YearOf(DateAndTime);
  tmpDate := EncodeDate(year, 10, 31);
  for i := 0 to 6 do Begin
    //tmpDate := _StrToDateTime('10.' + IntToStr(31-i) + '.' + IntToStr(YearOf(DateAndTime)));
    //tmpDate := EncodeDate(year, 10, 31 - i);
    tmpDate := tmpDate - 1;
    if 7 = DayOfTheWeek(tmpDate) then wtmpDate := tmpDate;
  end;
  stmpDate := 0;
  tmpDate := EncodeDate(year, 03, 31);
  for i := 0 to 6 do Begin
    //tmpDate := _StrToDateTime('03.' + IntToStr(31-i) + '.' + IntToStr(YearOf(DateAndTime)));
    //tmpDate := EncodeDate(year, 03, 31 - i);
    tmpDate := tmpDate - 1;
    if 7 = DayOfTheWeek(tmpDate) then stmpDate := tmpDate;
  end;

  if DateAndTime = stmpDate then result := scWinterSummer else
    if DateAndTime = wtmpDate then result := scSummerWinter else
      if ((DateAndTime > wtmpDate) or (Trunc(DateAndTime) < stmpDate))
       then result := scWinter
       else result := scSummer;

  //result := scWinter;
  //
  //Cash function BEGIN
  GSTIndex := GSTIndex + 1;
  if (GSTIndex > GSTArrayMax) then GSTIndex := 1;
  GSTArray1[GSTIndex].DT := DateAndTime;
  GSTArray1[GSTIndex].SC := result;
  //Cash function END
  //
end;

//-----------------------------------------------------------------------------//


initialization
```

```
  GSTIndex := 1;

  GSTArray1[1].DT := _StrToDateTime('03.27.2005 04:00:00');
  GSTArray1[1].SC := scSummer;
  for i := 2 to GSTArrayMax do
    GSTArray1[i] := GSTArray1[1];

finalization

end.
```