

## 6.2. Delphi komponensek készítése

1. *TEdit* osztályból származtatott összeadó komponens [XYEdit](#)
2. Háromállapotú jelölőnégyzet [CheckBox3S](#)
3. Háromdimenziós címke [Cimke3D](#)
4. Szövegszerkesztő kapcsolt címke címmel [LabEdit](#)
5. A 6.2.4 *Ora* komponensének tesztelő programja [Ora](#)



Írjunk komponst, amely a **TEdit** osztályt kiegészíti két egész tulajdonsággal ( $x$ ,  $y$ ) és automatikusan megjeleníti az  $x+y$  összeget! (**XYEdit**)

A **TXYEdit** osztályt a **TEdit**-ből származtatjuk az **XYEdit** modulban. Az új osztály  $x$  és  $y$  tulajdonságának írására és olvasására a **Setx**, **Getx**, és a **Sety**, **Gety** metódusokat használjuk. Az adatokat az  $fx$  és  $fy$  privát adatmezőkben tároljuk. A komponst a **Samples** vezérlőpaletta lapra regisztráljuk a **RegisterComponent** eljárással.

```
unit XYEdit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

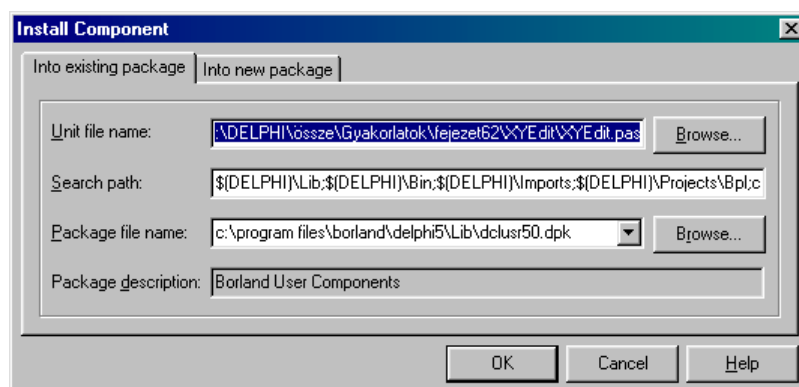
type
  TXYEdit = class(TEdit)
  private
    { Privát deklarációk }
    //Az adatmezők
    fx,fy : integer;
  public
    { Publikus deklarációk }
    //Az író, olvasó metódusok
    function Getx: integer;
    procedure Setx(ertek: integer);
    function Gety: integer;
    procedure Sety(ertek: integer);
  published
    { Publikált deklarációk }
    //Az tulajdonságok
    property x : integer read getx write setx;
    property y : integer read gety write sety;
  end;

procedure Register;

implementation

function TXYEdit.Getx: integer;
begin
  //Az x tulajdonság értéke az fx mezőből
  result:=fx;
end;
procedure TXYEdit.Setx(ertek: integer);
begin
  //Az x tulajdonság beállítása és tárolása az fx mezőbe
  fx:=ertek;
  //x+y megjelenítése
  text:=inttostr(fx+fy);
end;
function TXYEdit.Gety: integer;
begin
  //Az y tulajdonság értéke az fy mezőből
  result:=fy;
end;
procedure TXYEdit.Sety(ertek: integer);
begin
  //Az y tulajdonság beállítása és tárolása az fy mezőbe
  fy:=ertek;
  //x+y megjelenítése
  text:=inttostr(fx+fy);
end;
procedure Register;
begin
  RegisterComponents('Samples', [TXYEdit]);
end;
end.
```

Amennyiben használni szeretnénk az új komponenst, akkor először telepítenünk kell azt (**Component / Install Component...**).



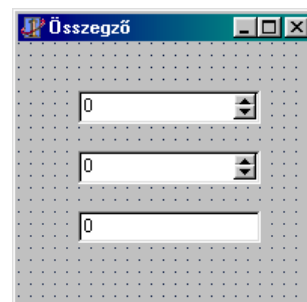
A telepített építőelemet már használhatjuk például a *TestP* projektben. A formot úgy tervezzük, hogy két **SpinEdit** komponens által beállított számot az *XYEdit* *X* és *Y* tulajdonságába írva végezzük el az összeadást. Ha a telepített komponenst a formra helyezzük, akkor a *TestU* modul **uses** listájába automatikusan bekerül az *XYEdit* modul.

#### uses

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, XYEdit, Spin;
```

#### type

```
TForm1 = class(TForm)
  SpinEdit1: TSpinEdit;
  SpinEdit2: TSpinEdit;
  edtSum: TXYEdit;
  procedure SpinEdit1Change(Sender: TObject);
  procedure SpinEdit2Change(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
```



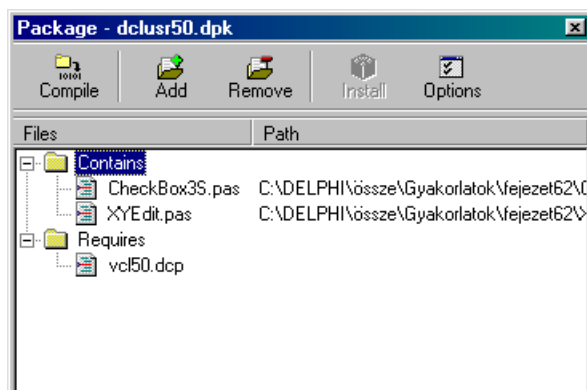
A **SpinEdit** vezérlők tartalmának megváltozásakor a *TXYEdit* típusú *edtSum* vezérlő *x* és *y* tulajdonságainak értékét aktualizáljuk, melynek hatására az összeadás automatikusan megtörténik.

```
procedure TForm1.SpinEdit1Change(Sender: TObject);
begin
  edtSum.x:=strtoint(SpinEdit1.Text);
end;

procedure TForm1.SpinEdit2Change(Sender: TObject);
begin
  edtSum.y:=strtoint(SpinEdit2.Text);
end;
```



Ha le szeretnénk törölni a telepített komponenst a komponenspalettáról, akkor először meg kell nyitnunk (**File/Open**) azt a csomagot (esetünkben a *C:\Program Files\Borland\Delphi5\Lib\dclusr50.dpk*) amely tárolja a komponenst. Ezt követően a csomagkezelőbe jutunk, ahonnan a **Remove** funkcióval törölhetjük a komponenst.





Készítsünk a **TCheckBox** osztályból származtatott komponenst, amely újradefiniálja a jelölőnégyzet állapotváltozásakor meghívódó **Toggle** virtuális metódust! (*CheckBox3S*)

A magyarul beszélő háromállapotú jelölőnégyzetet a **TCheckBox** osztályból származtatjuk. Bevezetünk egy *FAllapotszam* nevű belső adatmezőt. Az *Allapotszam* tulajdonság adata a belső adatmezőben tárolható. A jellemző írásáról a privát *SetAllapotszam* eljárás gondoskodik. Annak érdekében, hogy állapotváltozáskor a komponens szövege automatikusan megváltozzon, bevezetjük a *szoveg* tömböt, amit a konstruktorban inicializálunk és átdefiniáljuk a *Toggle* metódust a kiíráshoz. A **RegisterComponent** hívásakor intézkedünk arról, hogy a komponens a *Sample* palettalapra kerüljön. A komponenst tartalmazó modul a *CheckBox3S*.

```
unit CheckBox3S;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TCheckBox3S = class(TCheckBox)
  private
    FAllapotszam : integer;
    procedure SetAllapotszam(v:integer);
  public
    szoveg: array[cbUnchecked..cbGrayed] of ShortString;
    procedure Toggle; override;
    constructor Create(AOwner:TComponent); override;
  published
    property Allapotszam : integer read FAllapotszam write SetAllapotszam;
  end;

procedure Register;

implementation

{ Az új jelölőnégyzet konstruktora}
constructor TCheckBox3S.Create(AOwner:TComponent);
begin
  inherited Create(AOwner);
  Allapotszam:=3;
  Szoveg[cbUnchecked] := 'Nem kiválasztott';
  Szoveg[cbChecked] := 'Kiválasztott';
  Szoveg[cbGrayed] := 'Szürkített';
end;

procedure TCheckBox3S.SetAllapotszam(v:integer);
begin
  if v in [1..3] then FAllapotszam:=v
    else FAllapotszam:=3;
end;

procedure TCheckBox3S.Toggle;
begin
  beep;
  State := TCheckBoxState((Ord(state)+1) mod Allapotszam);
  Caption:=Szoveg[State];
end;

procedure Register;
begin
  RegisterComponents('Samples', [TCheckBox3S]);
end;

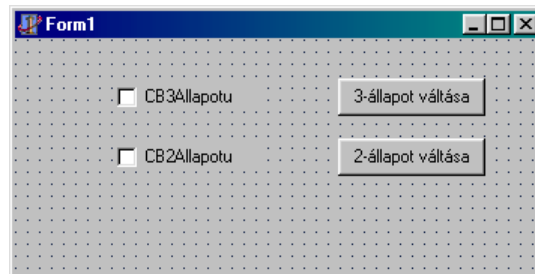
end.
```

Ha tesztelni is szeretnénk a programot, akkor a [XYEdit](#) komponenshez hasonlóan a *CheckBox3S* komponenst is telepítenünk kell.

A *TestP* projekt formjára felteszünk két *TCheckBox3S* komponenst és két gombot. A *CheckBox3S* modul felkerül a **uses** listára. A *CB2Allapotu* vezérlő *Allapotszam* tulajdonságát az objektumszerkesztőben 2-re állítjuk.

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, CheckBox3S;

type
  TForm1 = class(TForm)
    CB3Allapotu: TCheckBox3S;
    CB2Allapotu: TCheckBox3S;
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```



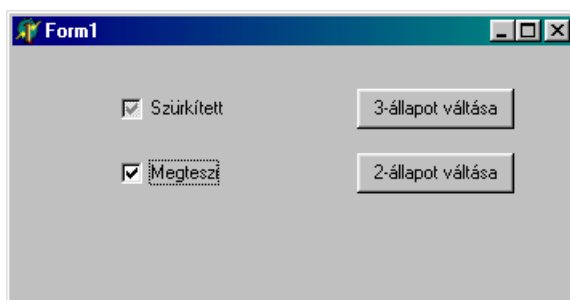
A form töltésekor kétállapotú jelölőnégyzet esetében megadjuk a szövegtömb elemeit.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  CB2Allapotu.szoveg[cbUnchecked] := 'Nem teszi meg';
  CB2Allapotu.szoveg[cbChecked] := 'Megteszi';
end;
```

A két gomb is a megfelelő vezérlők ***Toggle*** módszerát hívja.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  CB3Allapotu.Toggle;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  CB2Allapotu.Toggle;
end;
```



A komponens az [XYEdit](#) példában leírtaknak megfelelően törölhető a komponenespalettáról.



Származtassunk a *TLabel* komponensből 3D-s szövegmegjelenítést végző *TCimke3D* grafikus alkotóelemet! A komponens a 3D-s megjelenítést a szöveg 1 képponttal való eltolásával valósítsa meg! (*Cimke3D*)

A *TLabel*-ből származtatott *TCimke3D* komponens legyen lesüllyeszthető, vagy kiemelt! Ehhez definiáljuk a *TCimke3DStilus* felsorolt típust. Az osztálydefiníció során a *Stilus* írható, olvasható tulajdonság mellett a *DoDrawText* metódust használjuk a címke kirajzolásához.

```
type
  TCimke3DStilus = (lsSullyesztett, lsKiemelt);

  TCimke3D = class(TLabel)
  private
    FStilus: TCimke3DStilus;
    procedure SetStilus(AStilus: TCimke3DStilus);
  protected
    procedure Paint; override;
  public
    constructor Create(AOwner: TComponent); override;
    procedure DoDrawText(var Rect: TRect;Flags: Longint); override;
  published
    property Stilus: TCimke3DStilus read FStilus write SetStilus
      default lsSullyesztett;
  end;

procedure Register;
```

A címke létrehozásakor beállítjuk az alapértelmezett stílust.

```
constructor TCimke3D.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FStilus := lsSullyesztett;
end;
```

A *Stilus* tulajdonság beállításakor újrarajzoljuk a vezérlőt.

```
procedure TCimke3D.SetStilus(AStilus: TCimke3DStilus);
begin
  FStilus := AStilus;
  Invalidate;
end;
```

A *DoDrawText* metódust használjuk újrajzolásakor.

```
procedure TCimke3D.DoDrawText(var Rect: TRect;Flags: Longint);
var
  SzinMentes: TColor;
begin
  if not ShowAccelChar then
    // A '&' karakter megjelenítése
    Flags := Flags or DT_NOPREFIX;
  Canvas.Font := Font; // a címke betűtípusa
  if Enabled then
  begin
    // A szöveg kiírása a sötét és a világos rész
    // megjelenítésével
    SzinMentes := Canvas.Font.Color;

    if FStilus = lsSullyesztett then
      // Először sötét
      Canvas.Font.Color := clBtnShadow
    else
      // Először világos
      Canvas.Font.Color := clBtnHighlight;
    // A szöveg kiírása 1 pozícióval feljebb és balra
    OffsetRect(Rect, -1, -1);
    DrawText(Canvas.Handle, PChar(Caption), Length(Caption), Rect, Flags);
```

```

    if FStilus = lsKiemelt then
        // Másodszor sötét
        Canvas.Font.Color := clBtnShadow
    else
        // Másodszor világos
        Canvas.Font.Color := clBtnHighlight;
        // A szöveg kiírása 2 pozícióval lejjebb és jobbra
        OffsetRect(Rect, 2, 2);
        DrawText(Canvas.Handle, PChar(Caption), Length(Caption), Rect, Flags);
        // A szöveg eredeti színének és pozíciójának visszaállítása
        OffsetRect(Rect, -1, -1);
        Canvas.Font.Color := SzinMentes;
    end
else
    // A címke vezérlő le van tiltva
    Canvas.Font.Color := clGrayText;
    // A szöveg megjelenítése az eredeti pozíciójában
    DrawText(Canvas.Handle, PChar(Caption), Length(Caption), Rect, Flags);
end;

```

A **Paint** eseményben használjuk a *DoDrawText* metódust. Először azonban a háttérrel gondoskodunk.

```

// A vezérlő újrarajzolása
procedure TCimke3D.Paint;
const
    // A DrawText kiigazítási konstansainak tömbje
    Alignments: array[TAlignment] of Word = (DT_LEFT, DT_RIGHT, DT_CENTER);
var Rect: TRect;
    Flags: Longint;
begin
    with Canvas do
    begin
        if not Transparent then
        begin
            // Nem áttetsző háttér esetén
            // a szöveg alatti terület kifestése háttérszínnel
            Brush.Color := Self.Color;
            Brush.Style := bsSolid;
            FillRect(ClientRect);
        end;
        // Áttetsző háttér
        Brush.Style := bsClear;
        // A címkét befoglaló téglalap
        Rect:=ClientRect;
        // A DrawText működését vezérlő konstansok
        Flags := DT_EXPANDTABS or DT_WORDBREAK or Alignments[Alignment];
        // A szöveg megjelenítése
        DoDrawText(Rect,Flags);
    end;
end;

```

A regisztrálás a *Samples* palettalapra.

```

procedure Register;
begin
    RegisterComponents('Samples', [TCimke3D]);
end;

```

A komponens telepítése után a *TestP* projektben csak a betűméreteket, színeket és az újonnan létrehozott *Stilus* tulajdonságot állítottuk be az objektumszerkesztőben.







A *LabEdit* modulban a *TWinControl* leszármazottjaként hozzuk létre a *TLabEdit* komponenst. A komponens átméretezhető (kezeljük a *WM\_SIZE* üzenetet), az editor szövege (*ESzoveg*), a cím szövege (*ECimke*), a betűtípus, a szín tulajdonságként elérhetők. Hozzáférést engedünk a két építőelem komponenshez is (*cEdit*, *cLabel*).

```
type
  TLabEdit = class(TWinControl)
  private
    procedure WMSize(var Message:TMessage); message WM_SIZE;
    procedure CimkeBe(const s : String);
    procedure SzovegBe(const s : String);
    function CimkeKi : String;
    function SzovegKi : String;
  public
    constructor Create(AOwner: TComponent); override;
  published
    cEdit : TEdit;
    cLabel : TLabel;
    property ESzoveg : String read SzovegKi write SzovegBe ;
    property ECimke : String read CimkeKi write CimkeBe ;
    property Font;
    property Color;
  end;

procedure Register;
```

Létrehozáskor beállíthatjuk az alapértékeket.

```
constructor TLabEdit.Create(AOwner: TComponent);
begin
  inherited;
  cLabel:=TLabel.Create(Self);
  cLabel.Parent:=Self;
  cEdit:=TEdit.Create(Self);
  cEdit.Parent:=Self;
  Self.Width:=cEdit.Width+cLabel.Width+5;
  Self.Height:=cEdit.Height;
  ECimke:='címke';
  ESzoveg:='szöveg';
end;
```

Átméretezéskor megadjuk az építőelemek helyét és szövegét.

```
procedure TLabEdit.WMSize(var Message:TMessage);
begin
  Inherited;
  with cLabel do
  begin
    Autosize:=true;
    Caption:=ECimke;
    Height:=self.Height;
    Left:=0;
    Top:=3;
    Refresh;
  end;
  with cEdit do
  begin
    Width:=self.Width-5-cLabel.Width;
    Left:=cLabel.Width+5;
    Height:=self.Height;
    Top:=0;
    Text:=ESzoveg;
  end;
end;
```

A tulajdonságok írása és olvasása, valamint a regisztrálás semmi újdonstágot nem tartalmaz, az építőelemek elérésére a *cLabel*, *cEdit* mezőket használjuk..

```
procedure TLabEdit.CimkeBe(const s : String);
begin
    cLabel.Caption:=s;
end;

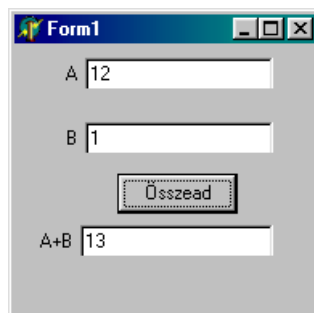
procedure TLabEdit.SzovegBe(const s : String);
begin
    cEdit.Text:=s;
end;

function TLabEdit.CimkeKi : String;
begin
    result:=CLabel.caption;
end;

function TLabEdit.SzovegKi : String;
begin
    result:=CEdit.text;
end;

procedure Register;
begin
    RegisterComponents('Samples', [TLabEdit]);
end;
```

A telepített komponenst a *TestP* projektben használjuk. Két *TLabEdit* típusú komponens szövegszerkesztőjének tartalmát összeadjuk, és az eredményt a harmadik *TLabEdit* típusú szövegmezőbe írjuk. A címkéket ( $A$ ,  $B$ ,  $A+B$ ) az objektum-felügyelőben állítottuk be.





Hozzuk létre a 6.2.4. fejezetben ismertetett *TOra* vezérlőt, illetve a komponens tesztelő alkalmazást!  
(Ora)

A 6.2.4 fejezet *Ora* komponensét és tesztprogramot egy könyvtárba másolva telepíthetjük és tesztelhetjük komponens és a programot. Az alábbiakban csak a programlistákat közöljük egyben. A komponens kódja:

```
unit Ora;

interface

uses
  SysUtils, Windows, Messages, Classes, Graphics,
  Controls, DsgnIntf, Dialogs, Math ;

type
  TEbresztesEvent = procedure (Sender: TObject) of object;

TOra = class(TCustomControl)
private
  FEOra : Byte;
  FEperc : Byte;
  FIdozito : Integer;
  FMukodik : Boolean;
  FEbreszteni : Boolean;
  FOnEbresztes : TEbresztesEvent;
  FColor: TColor;
  procedure SetColor(Color: TColor);
  procedure SetMukodik(Run: Boolean);
  procedure SetEOra(Ora24 : byte);
  procedure SetEPerc(Perc : byte);
protected
  procedure Paint; override;
  procedure WMIdozito(var Message: TMessage); message WM_Timer;
  procedure WMDestroy(var Message: TMessage); message wm_Destroy;
  procedure Atmeretezes(Sender : TObject);
public
  constructor Create(AOwner: TComponent); override;
published
  property Ebreszteni : Boolean read FEbreszteni write FEbreszteni;
  property Mukodik: Boolean read FMukodik write SetMukodik;
  property Szin : TColor read FColor write SetColor;
  property EbresztOra : Byte read FEOra write SetEOra;
  property EbresztPerc : Byte read FEperc write SetEPerc;
  property OnEbresztes: TEbresztesEvent read FOnEbresztes write FOnEbresztes;
end;

TOraEditor = class(TComponentEditor)
  procedure Edit; override;
end;

TSzinProperty = class(TColorProperty)
public
  function GetAttributes: TPropertyAttributes; override;
  procedure Edit; override;
end;

procedure Register;

implementation
{$R Ora.res}

procedure Register;
begin
  RegisterComponents('Samples', [TOra]);
  RegisterComponentEditor(TOra, TOraEditor);
  RegisterPropertyEditor(TypeInfo(TColor), TOra, 'Szin',
    TSzinProperty);
end;
```

```

{ --- TOra --- }

constructor TOra.Create(AOwner: TComponent);
begin
    inherited Create(AOwner);
    Width := 100;
    height := 100;
    FIdozito := 1;
    FColor := clAqua;
    onResize:=Atmeretezes;
end;

procedure TOra.WmDestroy(var Message: TMessage);
begin
    KillTimer(Handle, FIdozito);
    FIdozito := 0;
    inherited;
end;

procedure TOra.Paint;
begin
    Canvas.Brush.Color := FColor;
    Canvas.Pen.Color := clBlack;
    Canvas.RoundRect(0, 0, width, height,width div 3,height div 3);
    inherited Paint;
end;

procedure TOra.SetMukodik(Run: Boolean);
begin
    if Run then begin
        SetTimer(Handle, FIdozito, 50, nil);
        FMukodik := True;
    end else begin
        KillTimer(Handle, FIdozito);
        FMukodik := False;
    end;
end;

procedure TOra.SetColor(Color: TColor);
begin
    FColor := Color;
    InvalidateRect(Handle, nil, True);
end;

procedure TOra.SetEOra(Ora24: Byte);
begin
    if Ora24 in [0..23] then FEOra:=Ora24;
end;

procedure TOra.SetEPerc(Perc: Byte);
begin
    if Perc in [0..59] then FEperc:=Perc;
end;

procedure TOra.WMIdozito(var Message: TMessage);
var
    S: string;
    Hs,Ms,SS,MSs : word;
begin
    S := FormatDateTime('hh:mm:ss',Time);
    DecodeTime(Time, Hs, Ms, SS ,MSs);
    Canvas.Font.Size:=min(width, Height) div 5;
    Canvas.TextOut((Width div 2) - (Canvas.TextWidth(S) div 2),
        (Height div 2) - (Canvas.TextHeight(S) div 2),S);

    if Ebreszteni and (Hs=FEora) and (Ms=FEperc) then
    begin
        FEbreszteni:=False;
        if Assigned(FOnEbresztes) then FOnEbresztes(Self);
    end;
end;

```

```

procedure Tora.Atmeretezes(Sender:TObject);
begin
    SysUtils.beep;
    repaint;
end;

{ --- TOraEditor --- }

procedure TOraEditor.Edit;
begin
    MessageDlg('Óra építőelem', mtInformation, [mbOK],0);
end;

{ --- TSzinProperty --- }

function TSzinProperty.GetAttributes;
begin
    Result := [paMultiSelect, paValueList, paDialog];
end;

procedure TSzinProperty.Edit;
var
    S: String;
begin
    S := '';
    InputQuery('Színmegadás', 'Kérem a színt', S);
    SetValue(S);
end;

end.

```

A tesztelő program formájához tartozó unit:

```

unit OraTu;

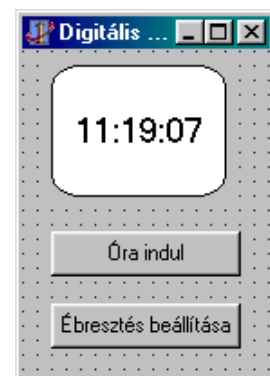
interface

uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
    StdCtrls, Ora;

type
    TOraFrm = class(TForm)
        EbresztBtn: TButton;
        Button1: TButton;
        Ora1: Tora;
        procedure EbresztBtnClick(Sender: TObject);
        procedure Ora1Ebresztes(Sender: TObject);
        procedure Button1Click(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    OraFrm: TOraFrm;

```



## implementation

```
{ $R *.DFM }  
  
procedure TOraFrm.EbresztBtnClick(Sender: TObject);  
var Hs, Ms, Ss, MSS : word;  
begin  
    DecodeTime(Time, Hs, Ms, Ss, MSS);  
    Ora1.EbresztOra:=Hs;  
    Ora1.EbresztPerc:=(Hs*60+MS+1) mod 60;  
    Ora1.Ebreszteni:=True;  
end;  
  
procedure TOraFrm.Ora1Ebresztes(Sender: TObject);  
begin  
    Beep;  
    ShowMessage('Hétalvók keljete fel, ' + #13#10 +  
                ' Frédi jött hozzátok el!');  
end;  
  
procedure TOraFrm.Button1Click(Sender: TObject);  
begin  
    Ora1.Mukodik:=true;  
end;  
  
end.
```

A tesztelő alkalmazás futás közbeni ablakai ébresztéskor:

