

3.4. Speciális vezérlők

1. Gyorsítógombok meghatározása programból [Hotkey1](#)
2. Adatmegadás **TrackBar** vezérlők felhasználásával [trackb1](#)
3. **TrackBar** adatainak beállítása [trackb2](#)
4. Folyamat állapotának jelzése **ProgressBar** vezérlővel [Progb1](#)
5. Adatmegadás **UpDown** vezérlők felhasználásával [UpDown1](#)
6. Adatmegadás **MaskEdit** vezérlők felhasználásával [MaskEdit1](#)
7. A **CheckListBox** használata [CheckListBox1](#)
8. Háromállapotú (**AllowGrayed**) **CheckListBox** használata [CheckListBox2](#)
9. A **ListView** vezérlő programozása [ListView](#)
10. A **TreeView** vezérlő programozása [TreeView0](#)
11. A **TreeView** és **ListView** vezérlők használata [TreeView1](#)
12. A **StatusBar** vezérlő használata [StatusBar1](#)
13. A **ToolBar** vezérlő használata [ToolBar1](#)
14. A **ScrollBar** vezérlő használata [ScrollBar](#)
15. A **Splitter** használata [Splitter](#)
16. Pascal háromszög megjelenítése **StringGrid**-ben [Pascal_hszog](#)
17. Szorzótábla saját kifestésű **StringGrid**-ben [Szorzotabla](#)
18. Képes nyomógombok sztringtáblában [Kepgaleria](#)
19. Naptár és időpont (**DateTimePicker**, **MonthCalendar**) [Naptar1](#)

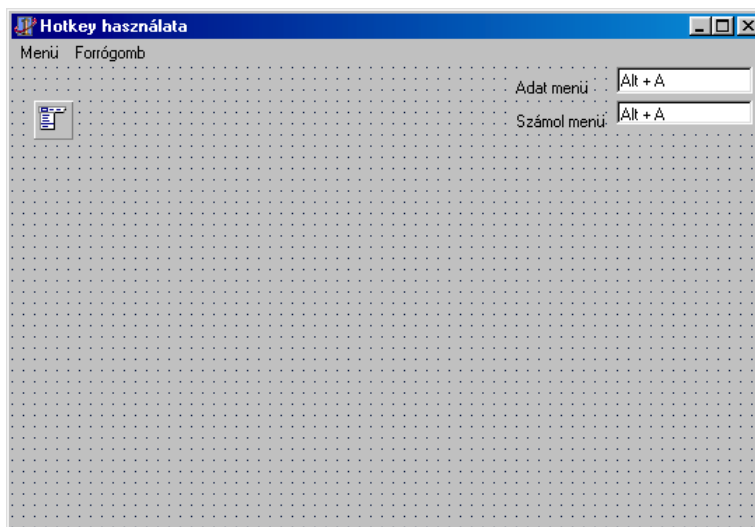


Készítsünk menüvezérelt programot, amely kiszámolja a beolvasott adat négyzetgyökét. Változtassuk dinamikusan az *Adat* és *Számol* menüpont gyorsítóbillentyűjét! (*HotKey1*)

Legyen az alábbi a program menürendszere!

Menü	Forrógomb
Adat	Beállítás
Számol	Letiltás
Kilépés	

Az *Adat* és a *Számol* menü forrógombjainak változtatását **Hotkey** vezérlőkkel lehet megoldani. A *Forrógomb* menü *Beállítás* almenüje engedélyezi a gyorsítógombot fogadó **Hotkey** vezérlők működését. A *Letiltás* almenü a **Hotkey** vezérlőket letiltja.



A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    menu: TMenuItem;
    Adat: TMenuItem;
    Szamol: TMenuItem;
    N1: TMenuItem;
    Kilepes: TMenuItem;
    HotKey1: THotKey;
    HotKey2: THotKey;
    Label1: TLabel;
    Label2: TLabel;
    Forrgomb1: TMenuItem;
    Beallitasa: TMenuItem;
    Letiltasa: TMenuItem;
    procedure KilepesClick(Sender: TObject);
    procedure AdatClick(Sender: TObject);
    procedure SzamolClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure LetiltasaClick(Sender: TObject);
    procedure BeallitasaClick(Sender: TObject);
  end;

var
  a, ered : real;
```

Az *Adat* menüpont kiválasztásakor meghívódó *AdatClick* eseménykezelő eljárásban a menüpont *ShortCut* tulajdonságába írjuk a *Hotkey1* vezérlő által fogadott gyorsítógomb-kombinációt. *InputBox* párbeszédablakkal olvassuk be az adatot és valós számmá alakítva írjuk be az *a* változóba.

```

procedure TForm1.AdatClick(Sender: TObject);
var
    s:string;
begin
    Adat.ShortCut:= Hotkey1.hotkey;
    s := InputBox('Adat megadása','Adat','1');
    a := StrToFloat(s);
end;

```

A *Számol* menüpont kiválasztásakor meghívódó *SzamolClick* eseménykezelő eljárásban a menüpont *ShortCut* tulajdonságába írjuk a *Hotkey1* vezérlő által fogadott gyorsítógomb-kombinációt. Kiszámítjuk az adat gyökét és *ShowMessage* párbeszédablakban jelenítjük meg.

```

procedure TForm1.SzamolClick(Sender: TObject);
var
    s:string;
begin
    Szamol.ShortCut:= Hotkey2.hotkey;
    ered := sqrt(a);
    s := FloatToStr(a);
    ShowMessage(s+' gyöke: '+ FloatToStr(ered));
end;

```

A *FormCreate* eseménykezelő eljárásban hozzáférhetővé tesszük a két gyorsítógombot fogadó vezérlőt, valamint beállítjuk a <Ctrl> billentyű módosítási lehetőségét. A *Hotkey1* vezérlőnek letiltjuk a <Shift> és <Alt>, a *Hotkey2* vezérlőnek pedig a <Shift> billentyűk használatát.

```

procedure TForm1.FormCreate(Sender: TObject);
begin
    Hotkey1.Enabled := True;
    Hotkey2.Enabled := True;
    Hotkey1.Modifiers := [hkCtrl];
    Hotkey1.InvalidKeys := [hcShift,hcAlt];
    Hotkey2.Modifiers := [hkCtrl];
    Hotkey2.InvalidKeys := [hcShift];
end;

```

A *Letiltása* menüpont kiválasztásakor meghívódó *LetiltasaClick* eseménykezelő eljárásban a *Hotkey1* és a *Hotkey2* vezérlő hozzáférését letiltjuk.

```

procedure TForm1.LetiltasaClick(Sender: TObject);
begin
    Hotkey1.Enabled := False;
    Hotkey2.Enabled := False;
end;

```

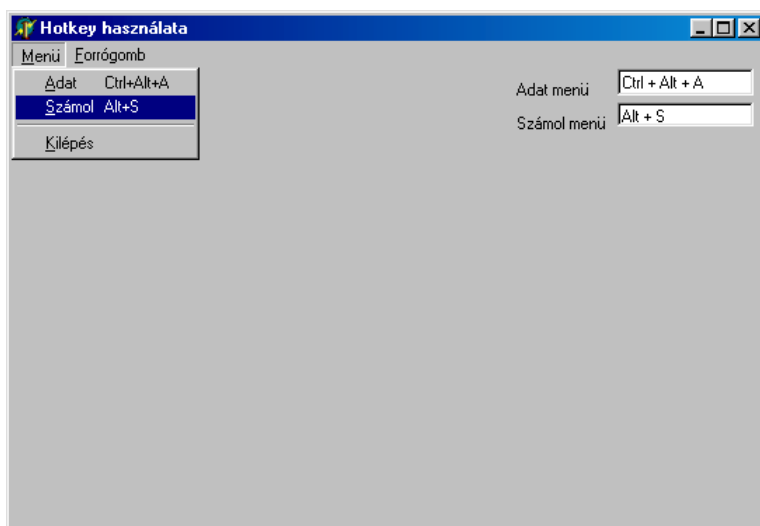
A *Beállítás* menüpont kiválasztásakor meghívódó *BeallitasaClick* eseménykezelő eljárásban a *Hotkey1* és a *Hotkey2* vezérlő hozzáférését engedélyezzük.

```

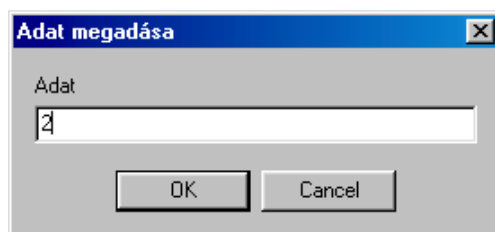
procedure TForm1.BeallitasaClick(Sender: TObject);
begin
    Hotkey1.Enabled := True;
    Hotkey2.Enabled := True;
end;

```

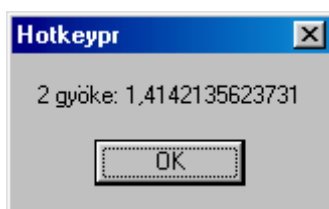
A program futási képe:




Adat beolvasása:

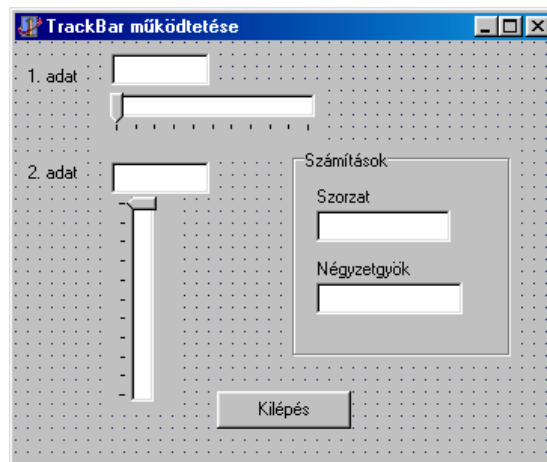


Eredmény megjelenítése a *Számol* menüpont kiválasztására:



 Készítsünk programot, amely meghatározza két egész szám szorzatát és négyzetgyökét, az adatok változtatása vízszintes és függőleges fekvésű **TrackBar** vezérlők felhasználásával történjen! (*trackb1*)

A feladat megoldásához használt vezérlők (Az adatokat megjelenítő *Edit1* és *Edit2* csak olvashatók):



A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    TrackBar1: TTrackBar;
    TrackBar2: TTrackBar;
    Kilepes: TButton;
    GroupBox1: TGroupBox;
    Edit1: TEdit;
    Edit2: TEdit;
    Label1: TLabel;
    Edit3: TEdit;
    Label2: TLabel;
    Edit4: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure TrackBar1Change(Sender: TObject);
    procedure TrackBar2Change(Sender: TObject);
    procedure KilepesClick(Sender: TObject);
  private
    szorzat : integer;
    negyzetgyok : real;
  end;
```

A *FormCreate* eseménykezelő eljárásban a vezérlők tulajdonságait kezdőértékre állítjuk.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TrackBar1.Min := 0;
  TrackBar1.Max := 10;
  TrackBar1.LineSize := 1;
  TrackBar1.PageSize := 2;
  TrackBar2.Min := 0;
  TrackBar2.Max := 10;
  TrackBar2.LineSize := 1;
  TrackBar2.PageSize := 2;
  szorzat := 0;
  negyzetgyok := 0;
  TrackBar1.Position := 0;
  TrackBar2.Position := 0;
  Edit1.Text := '0';
  Edit2.Text := '0';
  Edit3.Text := '0';
  Edit4.Text := '0';
end;
```

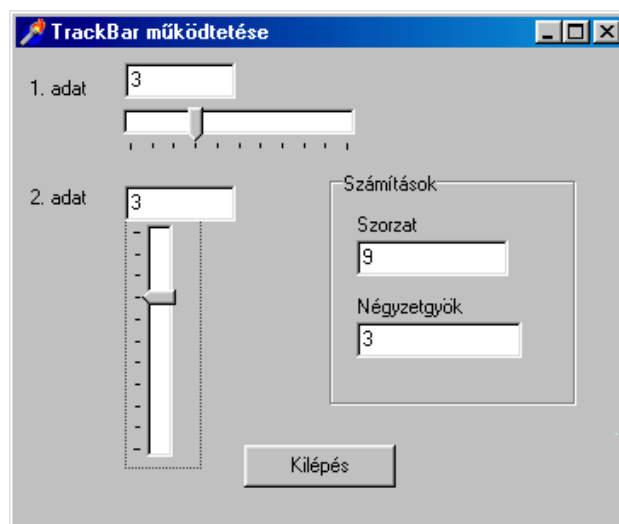
A "1. adat" értékét a *TrackBar1* vezérlő mozgatásával meghívódó *TrackBar1Change* eseménykezelő eljárással állítjuk be és az eredményt azonnal kiszámítjuk a másik adat leolvasásával.


```
procedure TForm1.TrackBar1Change(Sender: TObject);
var
    x1, x2 : integer;
begin
    Edit1.Text := IntToStr(TrackBar1.Position);
    x1 := TrackBar1.Position;
    x2 := TrackBar2.Position;
    szorzat := x1*x2;
    negyzetgyok := sqrt(szorzat);
    Edit3.Text := IntToStr(szorzat);
    Edit4.Text := FloatToStr(negyzetgyok);
end;
```

A "2. adat" értékét a *TrackBar2* vezérlő mozgatásával meghívódó *TrackBar2Change* eseménykezelő eljárással állítjuk be és az eredményt azonnal kiszámítjuk a másik adat leolvasásával.

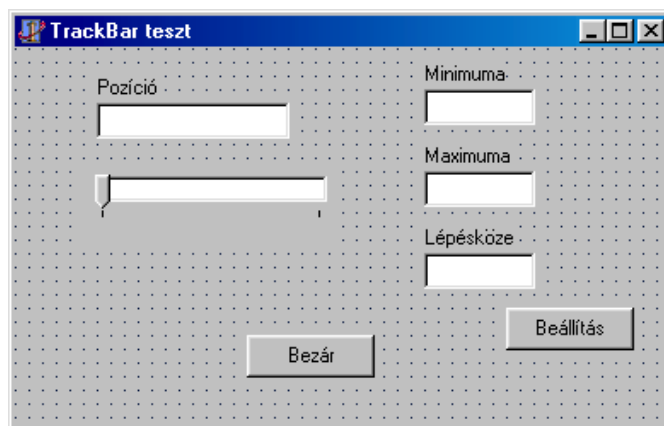
```
procedure TForm1.TrackBar2Change(Sender: TObject);
var
    x1, x2 : integer;
begin
    Edit2.Text := IntToStr(TrackBar2.Position);
    x1 := TrackBar1.Position;
    x2 := TrackBar2.Position;
    szorzat := x1*x2;
    negyzetgyok := sqrt(szorzat);
    Edit3.Text := IntToStr(szorzat);
    Edit4.Text := FloatToStr(negyzetgyok);
end;
```

A program futási képe:



 Készítsünk programot, amelyben a csúszka kezdő- és végpozícióját, illetve lépésközét a megadott értékekre lehet beállítani. A csúszka pozícióját jelezzük is ki! (*trackb2*)

A feladat megoldásához a Form1 vezérlői:



A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    TrackBar1: TTrackBar;
    Edit1: TEdit;
    Button1: TButton;
    Bezar: TButton;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure TrackBar1Change(Sender: TObject);
    procedure BezarClick(Sender: TObject);
  end;
```

A *FormCreate* eseménykezelő eljárásban a vezérlők tulajdonságait alapállapotra állítjuk:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  TrackBar1.TickStyle := tsManual;
  TrackBar1.Min := 5;
  TrackBar1.Max := 25;
  TrackBar1.LineSize := 5;
  TrackBar1.PageSize := 5;
  Button1.Enabled := true;
end;
```

A *Beállítás* nyomógomb megnyomásakor a *Button1Click* eseménykezelő eljárás hívódik meg. A maximumot, minimumot és a lépésközt bemenő adatként adjuk meg és ezeket az értékeket beírjuk a *TrackBar1* vezérlő *Max*, *Min*, *LineSize* és *PageSize* tulajdonságaiba. Kivételkezeléssel kerüljük el a hibás adatmegadást. Kiszámítjuk és megjelenítjük a lépésközt jelző vonalkákat, a csúszka pozícióját pedig a minimumra állítjuk. A *Beállítás* nyomógomb hozzáférhetőségét letiltjuk és a fókusz a *Trackbar1* vezérlőre helyezzük.

```

procedure TForm1.Button1Click(Sender: TObject);
var
    i,k,m :integer;
begin
    try
        TrackBar1.Min := StrToInt(Edit2.Text);
    except
        Edit2.Text := '5';
        TrackBar1.Min := 5;
    end;
    try
        TrackBar1.Max := StrToInt(Edit3.Text);
    except
        Edit3.Text := '5';
        TrackBar1.Max := 25;
    end;
    try
        TrackBar1.PageSize := StrToInt(Edit4.Text);
    except
        Edit4.Text := '5';
        TrackBar1.PageSize := 5;
    end;
    m := (TrackBar1.Max-TrackBar1.Min) div TrackBar1.LineSize;
    k:= TrackBar1.Min+TrackBar1.LineSize;
    for i:=1 to m do
    begin
        TrackBar1.SetTick(k);
        k:= k+TrackBar1.LineSize;
    end;
    Button1.Enabled := false;
    TrackBar1.Position := TrackBar1.Min;
    Edit1.Text := IntToStr(TrackBar1.Position);
    TrackBar1.SetFocus;
end;

```

A csúszka változtatásával a *TrackBar1Change* eseménykezelő eljárásban a csúszka pozícióját a *Pozíció* mezőben jelenítjük meg.

```

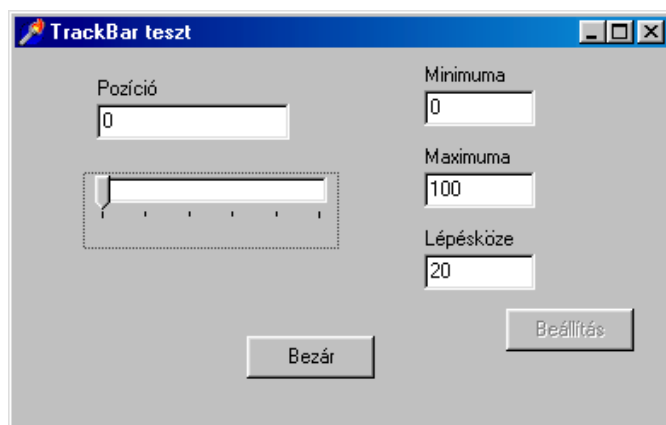
procedure TForm1.TrackBar1Change(Sender: TObject);
begin
    Edit1.Text := IntToStr(TrackBar1.Position);
end;

```

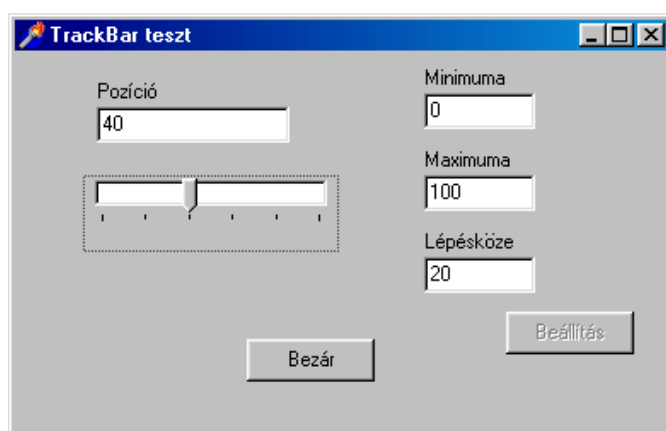
A program futási képei:

A beállításhoz szükséges adatok megadása:

A *TrackBar1* vezérlő beosztásának megjelenítése:

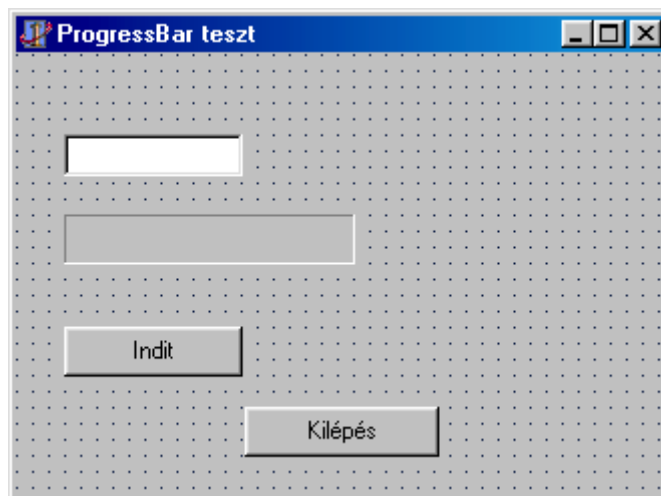


A csúszka mozgatása:





A feladat megoldásához szükséges vezérlők:



A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    ProgressBar1: TProgressBar;
    Indit: TButton;
    Kilepes: TButton;
  procedure FormCreate(Sender: TObject);
  procedure InditClick(Sender: TObject);
  procedure KilepesClick(Sender: TObject);
  private
    Szamlalo : integer;
  end;
```

A *FormCreate* eseménykezelő eljárásban a *Szamlalo* változót nullázzuk és a *Form1* bal felső sarkának koordinátáit beállítjuk a kívánt értékre.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Szamlalo := 0;
  Edit1.Text := '0';
  Left := 200;
  Top := 100;
end;
```

Az *Indit* nyomógomb megnyomásakor meghívódó *InditClick* eseménykezelő eljárásban egy 100x100-as ciklusban aritmetikai műveletet hajtunk végre és küldünk egy *PostMessage* üzenetet (értesítve ezzel az aktuális programszálat). A *ProgressBar1* vezérlő pozíciójába a külső ciklus (*i*) értékét írjuk be, melyet a szövegmezőben is megjelenítünk.

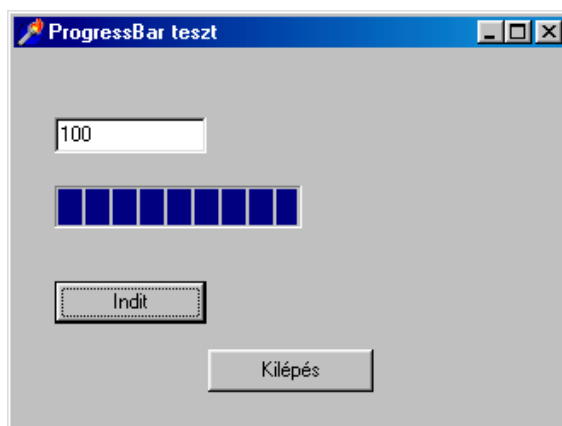
```

procedure TForm1.InditClick(Sender: TObject);
var
    i,j,k : integer;
begin

    for i:= 1 to 100 do
    begin
        k := 0;
        for j:= 1 to 100 do
        begin
            PostMessage(0,0,0,0);
            k := k + i * j; k := k + 1; // műveletvégzés
        end;
        ProgressBar1.Position := i;
        Edit1.Text := IntToStr(i);
        Edit1.Refresh;
    end;
end;

```

A program futási képe:





A feladat megoldásához az alábbi vezérlők szükségesek:

Ne felejtsük el az *UpDown* vezérlőket a megfelelő *Edit* vezérlőkhöz csatlakoztatni az *Associate* tulajdonsággal.

A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    Kilepes: TButton;
    GroupBox1: TGroupBox;
    Edit1: TEdit;
    Edit2: TEdit;
    Label1: TLabel;
    Edit3: TEdit;
    Label2: TLabel;
    Edit4: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    UpDown1: TUpDown;
    UpDown2: TUpDown;
  procedure FormCreate(Sender: TObject);
  procedure KilepesClick(Sender: TObject);
  procedure UpDown1Click(Sender: TObject; Button: TUDBtnType);
  procedure UpDown2Click(Sender: TObject; Button: TUDBtnType);
  private
    szorzat : integer;
    negyzetgyok : real;
  end;
```

A *FormCreate* eseménykezelő eljárásban beállítjuk az adatok megadásához szükséges *UpDown1* és *UpDown2* vezérlő *Min* és *Max* tulajdonságait, valamint a *szorzat*, *negyzetgyok* változókat és a vezérlők pozíciót a 0-ra állítjuk.

```

procedure TForm1.FormCreate(Sender: TObject);
begin
    UpDown1.Min := 0;
    UpDown1.Max := 10;

    UpDown2.Min := 0;
    UpDown2.Max := 10;

    szorzat := 0;
    negyzetgyok := 0;
    UpDown1.Position := 0;
    UpDown2.Position := 0;
    Edit1.Text := '0';
    Edit2.Text := '0';
    Edit3.Text := '0';
    Edit4.Text := '0';
end;

```

Az 1. adat beállítását a szövegmező jobb oldalán lévő *UpDown1* vezérlőn való kattintással végezhetjük. Ebben az esetben hívódik meg az *UpDown1Click* eseménykezelő eljárás, amelyben leolvassuk az 1. adat és a 2. adat vezérlőinek pozícióit, majd elvégezzük a műveleteket, az eredményeket beírjuk a megfelelő adatmezőkbe kijelzés céljából.

```

procedure TForm1.UpDown1Click(Sender: TObject; Button: TUDBtnType);
var
    x1, x2 : integer;
begin
    Edit1.Text := IntToStr(UpDown1.Position);
    x1 := UpDown1.Position;
    x2 := UpDown2.Position;
    szorzat := x1*x2;
    negyzetgyok := sqrt(szorzat);
    Edit3.Text := IntToStr(szorzat);
    Edit4.Text := FloatToStr(negyzetgyok);
end;

```

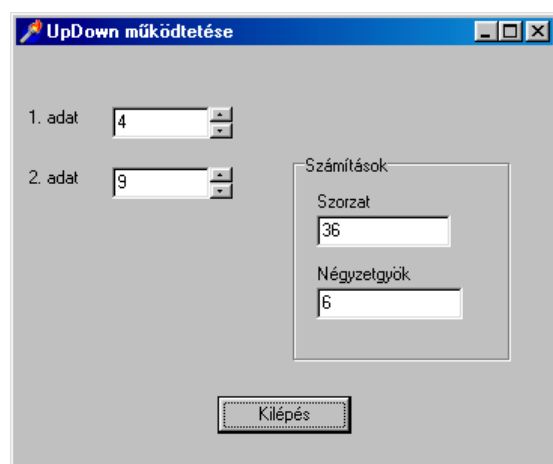
Az 2. adat beállítását a szövegmező jobb oldalán lévő *UpDown2* vezérlőn való kattintással végezhetjük. Ebben az esetben hívódik meg az *UpDown2Click* eseménykezelő eljárás, amelyben leolvassuk az 1. adat és a 2. adat vezérlőinek pozícióit, majd elvégezzük a műveleteket, az eredményeket beírjuk a megfelelő adatmezőkbe kijelzés céljából.

```

procedure TForm1.UpDown2Click(Sender: TObject; Button: TUDBtnType);
var
    x1, x2 : integer;
begin
    Edit2.Text := IntToStr(UpDown2.Position);
    x1 := UpDown1.Position;
    x2 := UpDown2.Position;
    szorzat := x1*x2;
    negyzetgyok := sqrt(szorzat);
    Edit3.Text := IntToStr(szorzat);
    Edit4.Text := FloatToStr(negyzetgyok);
end;

```

A program futási képe:





Tervezzünk különféle **MaskEdit** vezérlőket telefonszám, dátum, idő, kód és rendszám beolvasására.

A program futási képei:

A *TForm1* osztály deklarációja:

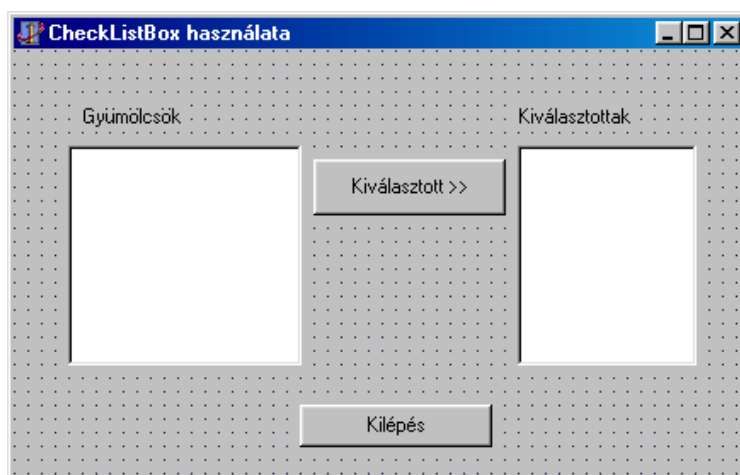
```
type
  TForm1 = class(TForm)
    MaskEdit1: TMaskEdit;
    Label1: TLabel;
    MaskEdit2: TMaskEdit;
    MaskEdit3: TMaskEdit;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    MaskEdit4: TMaskEdit;
    MaskEdit5: TMaskEdit;
    Label5: TLabel;
  procedure FormCreate(Sender: TObject);
  end;
```

A *FormCreate* eseménykezelő eljárásban a **MaskEdit** vezérlők *EditMask* tulajdonságában kell maszkot megadni.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  MaskEdit1.EditMask := '!\\(1\\)00-00-0-000-000;1';
  MaskEdit1.Text := '';
  MaskEdit1.AutoSelect := False;
  MaskEdit2.EditMask := '0000/00/00';
  MaskEdit2.Text := '';
  MaskEdit2.AutoSelect := False;
  MaskEdit3.EditMask := '##:##:##';
  MaskEdit3.Text := '';
  MaskEdit3.AutoSelect := False;
  MaskEdit4.EditMask := '>AAA#9999';
  MaskEdit4.Text := '';
  MaskEdit4.AutoSelect := False;
  MaskEdit5.EditMask := 'LLL-999';
  MaskEdit5.Text := '';
  MaskEdit5.AutoSelect := False;
end;
```



Tároljunk gyümölcsöket a *CheckBox1* vezérlőben, és a kiválasztott gyümölcsöket másoljuk át a *Kiválasztottak* listadobozba. A feladat megoldásához a *Form1* űrlapot az alábbiakban építettük fel:



A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    CheckBox1: TCheckBox;
    ListBox1: TListBox;
    Label1: TLabel;
    Button1: TButton;
    Kilepes: TButton;
  procedure FormCreate(Sender: TObject);
  procedure KivalasztottClick(Sender: TObject);
  procedure KilepesClick(Sender: TObject);
  end;
```

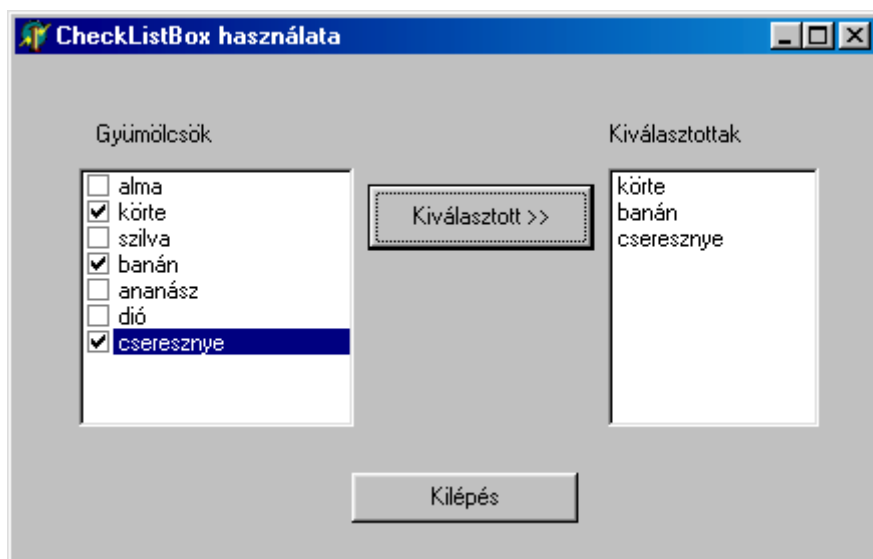
A *FormCreate* eseménykezelő eljárásban feltöltjük a **CheckBox1** vezérlőt az *Items* tulajdonság *Add* módszerével.


```
procedure TForm1.FormCreate(Sender: TObject);
begin
  CheckBox1.Items.Add('alma');
  CheckBox1.Items.Add('körte');
  CheckBox1.Items.Add('szilva');
  CheckBox1.Items.Add('banán');
  CheckBox1.Items.Add('ananász');
  CheckBox1.Items.Add('dió');
  CheckBox1.Items.Add('csersznye');
end;
```

A *Kiválasztott* nyomógomb megnyomásakor meghívódó *KivalasztottClick* eseménykezelő eljárásban töröljük a *ListBox1* listadobozt. Megvizsgálunk minden *CheckBox1* elemet, és a kijelölt elemeket átmásoljuk a *ListBox1* listadobozba.

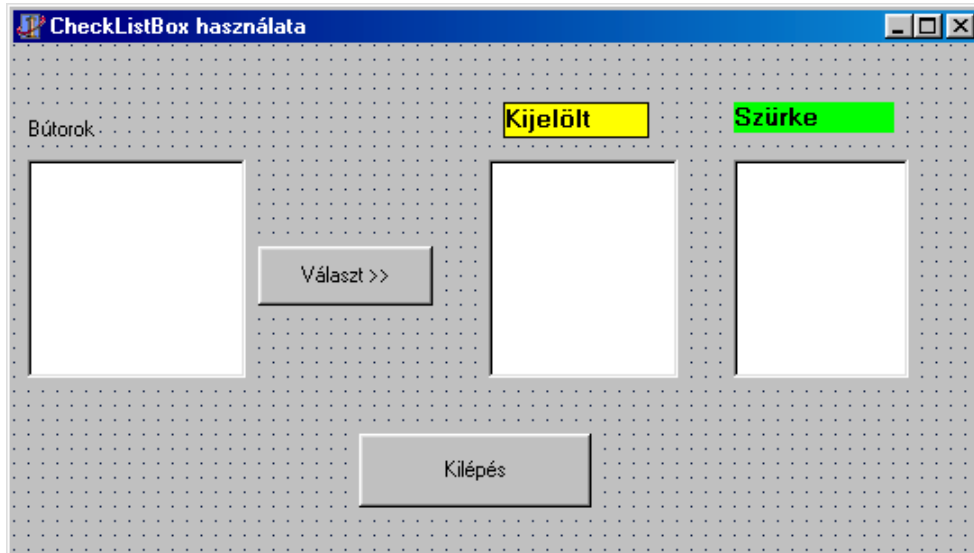
```
procedure TForm1.KivalasztottClick(Sender: TObject);
var
  i: integer;
begin
  ListBox1.Clear;
  for i:=0 to CheckBox1.Items.Count-1 do
    if CheckBox1.Checked[i] then
      ListBox1.Items.Add(CheckBox1.Items[i]);
  end;
```

A program futási képe:



 A *CheckBox* vezérlő *AllowGrayed* tulajdonságát állítsuk *True* értékre, hogy a vezérlő a szürke állapotot is felvegye! Másoljuk a kijelölt és a szürke állapotú tételeket külön – külön *ListBox* vezérlőkbe! (*CheckBox2*)

Tervezzük meg a feladatnak megfelelően a *Form1* űrlapot!



A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    CheckBox1: TCheckBox;
    StaticText1: TStaticText;
    Valaszt: TButton;
    ListBox1: TListBox;
    ListBox2: TListBox;
    StaticText2: TStaticText;
    Kilepes: TButton;
    Label1: TLabel;
  procedure FormCreate(Sender: TObject);
  procedure ValasztClick(Sender: TObject);
  procedure KilepesClick(Sender: TObject);
  procedure CheckBox1Click(Sender: TObject);
end;
```

A *FormCreate* eseménykezelő eljárásban beállítjuk a *CheckBox1* vezérlő *AllowGrayed* tulajdonságát *true* értékre, hogy szürke állapota is legyen, majd feltöltjük adatokkal.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  CheckBox1.AllowGrayed := true;
  CheckBox1.Items.Add('asztal');
  CheckBox1.Items.Add('szék');
  CheckBox1.Items.Add('lámpa');
  CheckBox1.Items.Add('szőnyeg');
end;
```

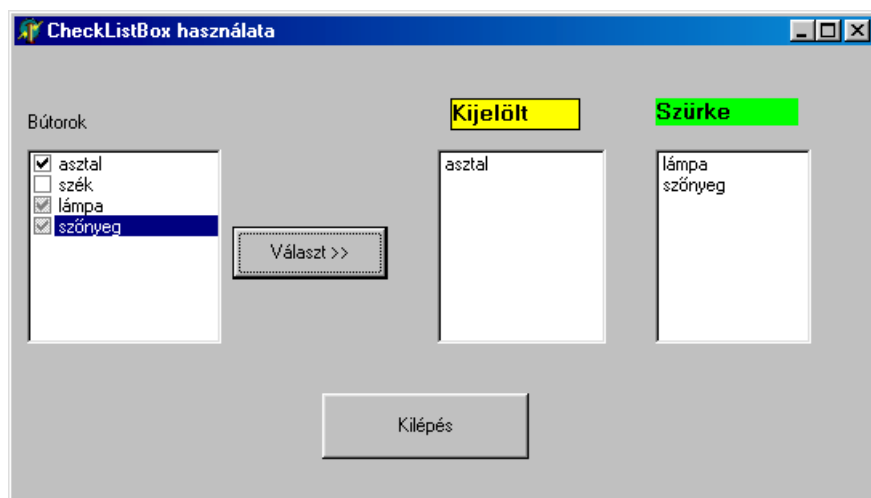
A "Választ >>" nyomógomb megnyomásakor meghívódó *ValasztClick* eseménykezelő eljárásban töröljük a *Kijelölt* és a *Szürke* fejlécű listadobozokat, és a *CheckListBox1* vezérlőből szétválogatjuk a kijelölt (*cbChecked*) és a szürke állapotú elemeket (*cbGrayed*).


```
procedure TForm1.ValasztClick(Sender: TObject);
var
  i : integer;
begin
  ListBox1.Clear;
  ListBox2.Clear;
  for i:=0 to CheckListBox1.Items.Count-1 do
  begin
    if CheckListBox1.State[i] = cbChecked then
      ListBox1.Items.Add(CheckListBox1.Items[i]);
    if CheckListBox1.State[i] = cbGrayed then
      ListBox2.Items.Add(CheckListBox1.Items[i]);
  end;
end;
```

A *CheckListBox1* vezérlőn való kattintás hatására meghívódó *CheckListBoxClick* eseménykezelő eljárásban töröljük a két eredményt tartalmazó listaablakokat.

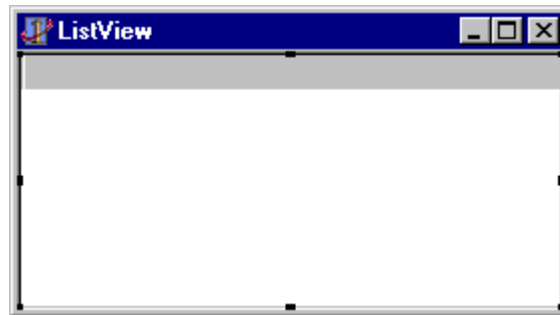
```
procedure TForm1.CheckListBox1Click(Sender: TObject);
begin
  ListBox1.Clear;
  ListBox2.Clear;
end;
```

A program futási képe:



 Készítsünk alkalmazást, mely programból feltölti a **ListView** vezérlőt elemekkel és „alelemekkel”, valamint részletes formában (*vsReport*) jeleníti meg az adatokat! (*ListView*)

Helyezzük a *ListView1* vezérlőt a szélekhez igazítva a *Form1* űrlapra! Legyen a *ViewStyle* jellemző *vsReport*!



A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    ListView1: TListView;
    procedure FormCreate(Sender: TObject);
    procedure ListView1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

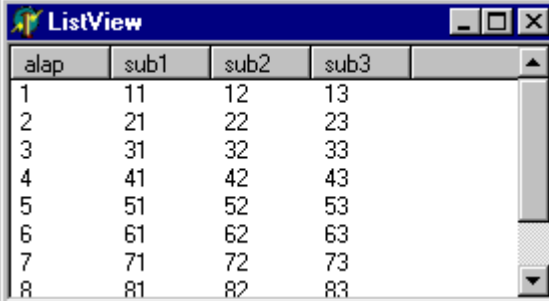
Az ablak létrehozásakor töltjük az elemeket az alábbiak szerint.

```
procedure TForm1.FormCreate(Sender: TObject);
var
  I,j: Integer;
  ListItem: TListItem;
  NewColumn: TListColumn;
begin
  with listview1 do
    begin
      // oszlopdefiníciók
      NewColumn := Columns.Add;
      NewColumn.Caption := 'alap';
      NewColumn := Columns.Add;
      NewColumn.Caption := 'sub1';
      NewColumn := Columns.Add;
      NewColumn.Caption := 'sub2';
      NewColumn := Columns.Add;
      NewColumn.Caption := 'sub3';
      // sorok kitöltése
      for I := 1 to 9 do
        begin
          ListItem := Items.Add;
          ListItem.Caption := inttostr(I);
          // az alelemek definíciója
          for j:=1 to 3 do
            ListItem.SubItems.Add(inttostr(I)+inttostr(J));
          end;
        end;
      end;
    end;
```

A listán való kattintáskor hangjelzést adunk.

```
procedure TForm1.ListView1Click(Sender: TObject);  
begin  
    beep;  
end;
```

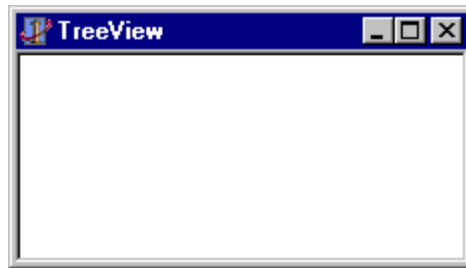
A futási képen jól látható a riport formátum



alap	sub1	sub2	sub3
1	11	12	13
2	21	22	23
3	31	32	33
4	41	42	43
5	51	52	53
6	61	62	63
7	71	72	73
8	81	82	83



Helyezzük a *TreeView1* vezérlőt a szélekhez igazítva a *Form1* űrlapra!



A *TForm1* osztály deklarációja:

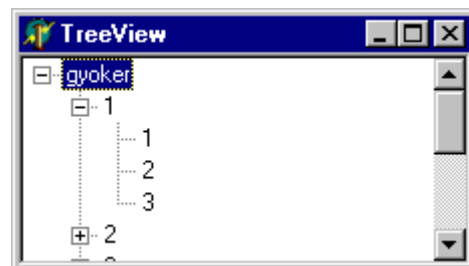
```
type
  TForm1 = class(TForm)
    TreeView1: TTreeView;
    procedure FormCreate(Sender: TObject);
    procedure TreeView1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

Az ablak létrehozásakor töltjük az elemeket:

```
procedure TForm1.FormCreate(Sender: TObject);
var
  I, j: Integer;
  gyoker: TTreeNode;
  gyerek: array[1..9] of TTreeNode;
begin
  with treeview1.Items do
  begin
    gyoker:=Add(nil, 'gyoker'); // a gyökér csomópont
    // első szintű gyermekek létrehozása
    for I := 1 to 9 do
    begin
      gyerek[i]:=AddChild(gyoker, inttostr(I));
      // második szintű gyermekek létrehozása
      for j:=1 to 3 do
        AddChild(gyerek[i], inttostr(j));
      end;
    end;
  end;
end;
```

A megjelenítőn való kattintáskor hangjelzést adunk.

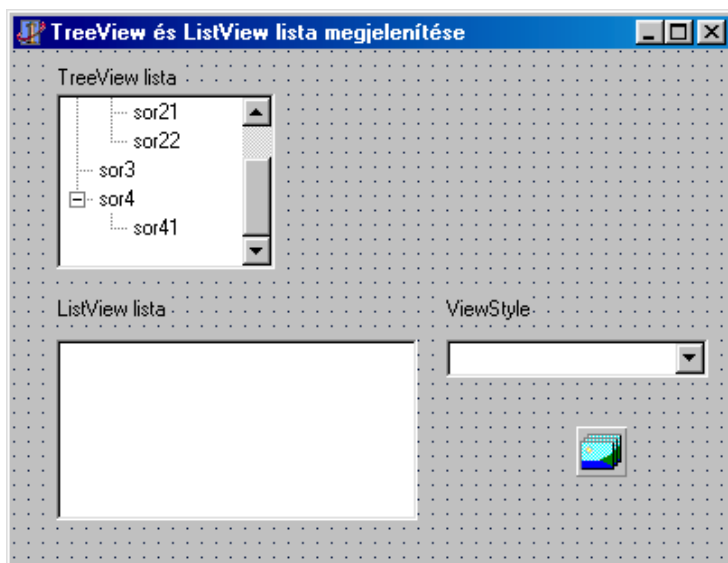
```
procedure TForm1.TreeView1Click(Sender: TObject);
begin
  beep;
end;
```



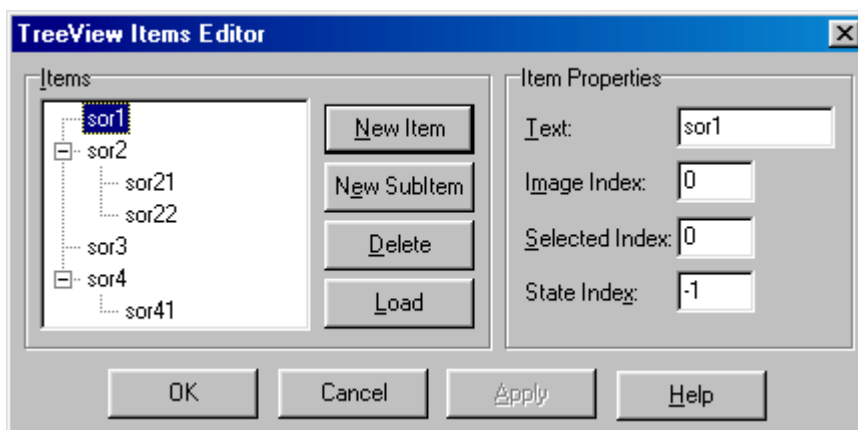


TreeView Editorral töltjük fel a ***TreeView*** listát, és egy ***Listview*** listába pedig tároljunk képeket! Jelenítsük meg az elemeket különböző ***ViewStyle*** módon! (***TreeView1***)

A *Form1* űrlap vezérlői:



A "TreeView Items Editorral" tervezhetjük meg a lista elemeit:



A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    TreeView1: TTreeView;
    ListView1: TListView;
    Combo1: TComboBox;
    ImageList1: TImageList;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure Combo1Click(Sender: TObject);
  end;
```

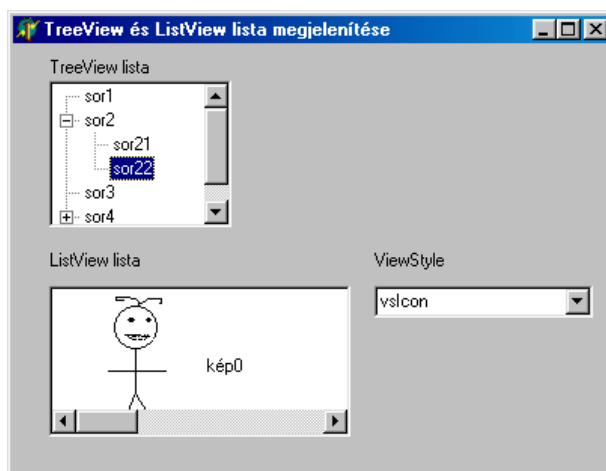
A *FormCreate* eseménykezelő eljárásban kezdőértéket adunk a vezérlők megfelelő tulajdonságainak.

```
procedure TForm1.FormCreate(Sender: TObject);
var
  I: Integer;
  ListItem: TListItem;
  NewColumn: TListColumn;
begin
  with ListView1 do
    begin
      SmallImages := ImageList1;
      LargeImages := ImageList1;
      // képekkel való feltöltés
      for I := 0 to ImageList1.Count - 1 do
        begin
          ListItem := Items.Add;
          ListItem.Caption := 'kép' + IntToStr(I);
          ListItem.ImageIndex := I;
        end;
      // oszlopok fejlécének megadása
      NewColumn := Columns.Add;
      NewColumn.Caption := '1. oszlop';
      NewColumn := Columns.Add;
      NewColumn.Caption := '2. oszlop';
      NewColumn := Columns.Add;
      NewColumn.Caption := '3. oszlop';
      // a kombinált lista feltöltése
      Combo1.Items.AddObject('vsIcon', TObject(vsIcon));
      Combo1.Items.AddObject('vsList', TObject(vsList));
      Combo1.Items.AddObject('vsReport', TObject(vsReport));
      Combo1.Items.AddObject('vsSmallIcon', TObject(vsSmallIcon));
      Combo1.ItemIndex := 0;
    end;
  end;
```

A *ViewStyle* kombinált lista elemén kattintva meghívódik a *Combo1Click* eseménykezelő eljárás, melyben a kiválasztott tulajdonságot a **ListView** vezérlő *ViewStyle* tulajdonságába írjuk.

```
procedure TForm1.Combo1Click(Sender: TObject);
begin
  with Combo1 do
    ListView1.ViewStyle := TViewStyle(Items.Objects[ItemIndex]);
  end;
```

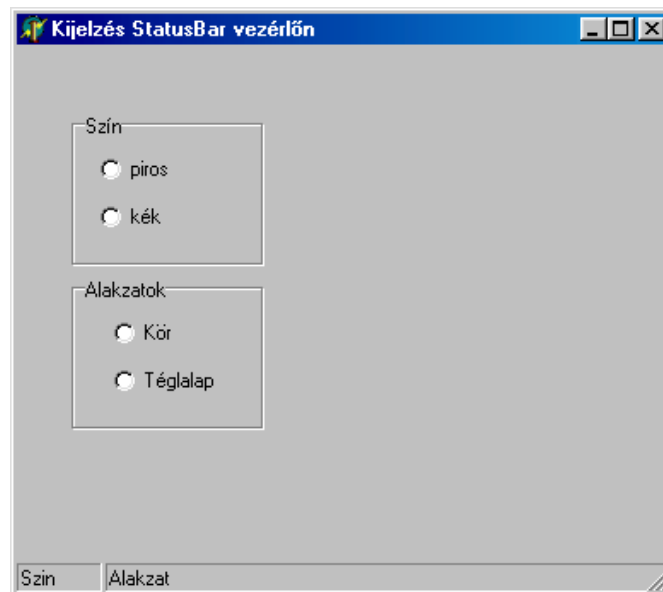
A program futási képe:





Rajzoljunk piros és kék színű kört, vagy téglalapot! A kiválasztott színt az állapotsor *Szín*, a kiválasztott alakzat nevét pedig az *Alakzat* helyén jelezzük ki! (*StatusBar1*)

A *Form1* vezérlői:



A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    StatusBar1: TStatusBar;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    GroupBox2: TGroupBox;
    RadioButton3: TRadioButton;
    RadioButton4: TRadioButton;
    procedure RadioButton1Click(Sender: TObject);
    procedure RadioButton3Click(Sender: TObject);
    procedure RadioButton4Click(Sender: TObject);
    procedure RadioButton2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormPaint(Sender: TObject);
  end;
```

Deklaráció:

A kiválasztott szín tárolására a *szin* változó, a kiválasztott alakzat tárolására pedig az *alakzat* változó szolgál.

```
var
  Form1: TForm1;
  szin : TColor;
  alakzat: integer;
```

A *FormCreate* eseménykezelő eljárásban a deklarált változóknak kezdőértéket adunk.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  szin := clYellow;
  alakzat := 0;
end;
```


A *Piros* választógombon kattintva meghívódik a *RadioButton1Click* eseménykezelő eljárás, amelyben az állapotsor első pozícióján a *piros* szöveget jelenítjük meg, valamint a *szin* változóba a *piros* szíkonstanst írjuk. Meghívjuk az *Invalidate* metódust, mely a *FormPaint* aktivizálásával elvégzi az ablak újrarajzolását.

```
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
    StatusBar1.Panels[0].Text := 'piros';
    szin := clRed;
    invalidate;
end;
```

A *Kék* választógombon kattintva meghívódik a *RadioButton2Click* eseménykezelő eljárás, amelyben az állapotsor első pozícióján a *kék* szöveget jelenítjük meg, valamint a *szin* változóba a *kék* szíkonstanst írjuk. Meghívjuk az *Invalidate* metódust, mely a *FormPaint* aktivizálásával elvégzi az ablak újrarajzolását.

```
procedure TForm1.RadioButton2Click(Sender: TObject);
begin
    StatusBar1.Panels[0].Text := 'kék';
    szin := clBlue;
    invalidate;
end;
```

A *Kör* választógombon kattintva meghívódik a *RadioButton3Click* eseménykezelő eljárás, amelyben az állapotsor második pozícióján a *Kör* szöveget jelenítjük meg, valamint a *alakzat* változóba az 1 értéket írjuk. Meghívjuk az *Invalidate* metódust, mely a *FormPaint* aktivizálásával elvégzi az ablak újrarajzolását.

```
procedure TForm1.RadioButton3Click(Sender: TObject);
begin
    StatusBar1.Panels[1].Text := 'Kör';
    alakzat := 1;
    invalidate;
end;
```

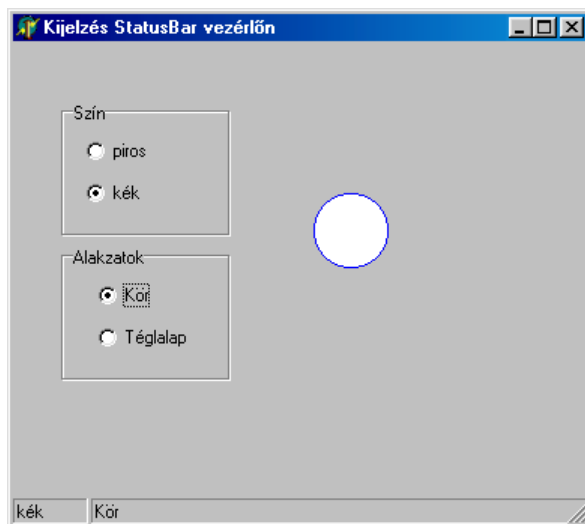
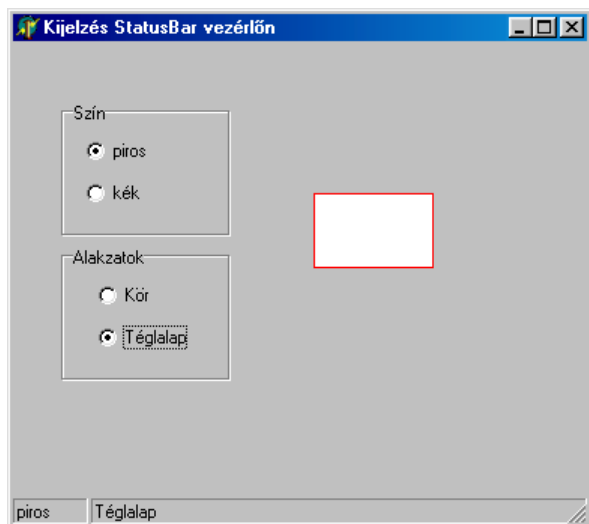
A *Téglalap* választógombon kattintva meghívódik a *RadioButton4Click* eseménykezelő eljárás, amelyben az állapotsor második pozícióján a *Téglalap* szöveget jelenítjük meg, valamint a *alakzat* változóba a 2 értéket írjuk. Meghívjuk az *Invalidate* metódust, mely a *FormPaint* aktivizálásával elvégzi az ablak újrarajzolását.

```
procedure TForm1.RadioButton4Click(Sender: TObject);
begin
    StatusBar1.Panels[1].Text := 'Téglalap';
    alakzat := 2;
    invalidate;
end;
```

A *FormPaint* megrajzolja a kiválasztott színnel a kijelölt alakzatot.

```
procedure TForm1.FormPaint(Sender: TObject);
begin
    if alakzat > 0 then
    begin
        Canvas.Pen.Color:=szin;
        case alakzat of
            1: Canvas.Ellipse(200,100,250,150);
            2: Canvas.Rectangle(200,100,280,150);
        end;
    end;
end;
```

A program futási képei:



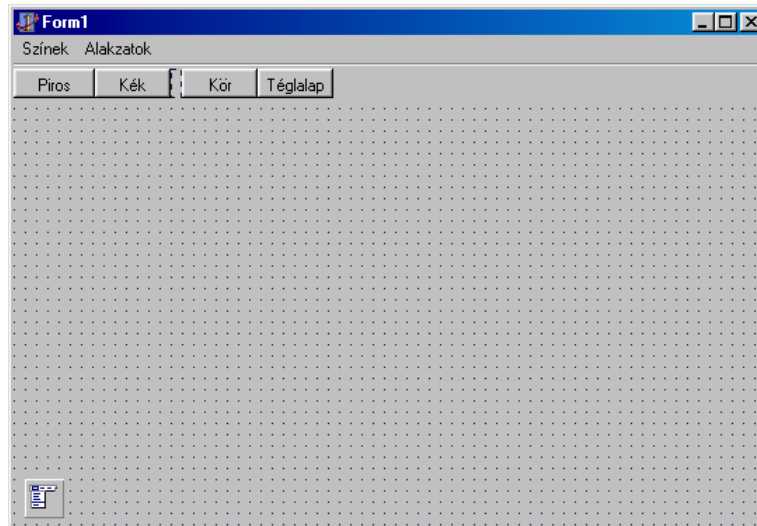


Készítsünk menüvezérelt programot, amellyel piros és kék színű kört, vagy téglalapot rajzolhatunk! A menüpontokhoz tervezzünk eszköztárban nyomógombokat is! (*ToolBar!*)

A program menürendszere:

Színek	Alakzatok
Piros	Kör
Kék	Téglalap

ugyanakkor a *ToolButton* vezérlőkkel is kiválhatjuk a menüpontokat.



A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    ToolBar1: TToolBar;
    ToolButton1: TToolButton;
    ToolButton2: TToolButton;
    ToolButton3: TToolButton;
    ToolButton4: TToolButton;
    ToolButton5: TToolButton;
    MainMenu1: TMainMenu;
    Men1: TMenuItem;
    Piros: TMenuItem;
    Kek: TMenuItem;
    Alakzatok1: TMenuItem;
    Kor: TMenuItem;
    Teglalap: TMenuItem;
    procedure FormCreate(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure PirosClick(Sender: TObject);
    procedure KekClick(Sender: TObject);
    procedure KorClick(Sender: TObject);
    procedure TeglalapClick(Sender: TObject);
  end;
```

A deklaráció:

A *szin* változó a kiválasztott színt, az *alakzat* változó a kiválasztott alakzatot tartalmazza:

Kör – 1, *Téglalap* – 2

```
var
  szin : TColor;
  alakzat: integer;
```

A *FormCreate* eseménykezelő eljárásban a deklarált változóknek kezdőértéket adunk.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    alakzat := 0;
    szin := clRed;
end;
```

A *FormPaint* eseménykezelő eljárás az *Invalidate* metódus hívásakor hívódik meg és az ablakot újrarajzolja.

```
procedure TForm1.FormPaint(Sender: TObject);
begin
    if alakzat > 0 then
    begin
        canvas.Pen.Color := szin;
        case alakzat of
            1: Canvas.Ellipse(100,100,250,250);
            2: Canvas.Rectangle(100,100,350,250);
        end;
    end;
end;
```

A *Piros* menüpont kiválasztásakor meghívódik a *PirosClick* eseménykezelő eljárás, amelyben a *szin* változóba írjuk a piros színek konstans, majd kérünk egy ablakújrifestést az *Invalidate* metódus hívásával. Ugyancsak a *PirosClick* eseménykezelő eljárás hívódik meg, ha a *Piros* eszközgombot nyomjuk meg.

```
procedure TForm1.PirosClick(Sender: TObject);
begin
    szin := clRed;
    Invalidate;
end;
```

A *Kék* menüpont kiválasztásakor meghívódik a *KekClick* eseménykezelő eljárás, amelyben a *szin* változóba írjuk a kék szín konstans, majd kérünk egy ablak újrifestést az *Invalidate* metódus hívásával. Ugyancsak a *KekClick* eseménykezelő eljárás hívódik meg, ha a *Kék* eszközgombot nyomjuk meg.

```
procedure TForm1.KekClick(Sender: TObject);
begin
    szin := clBlue;
    Invalidate;
end;
```

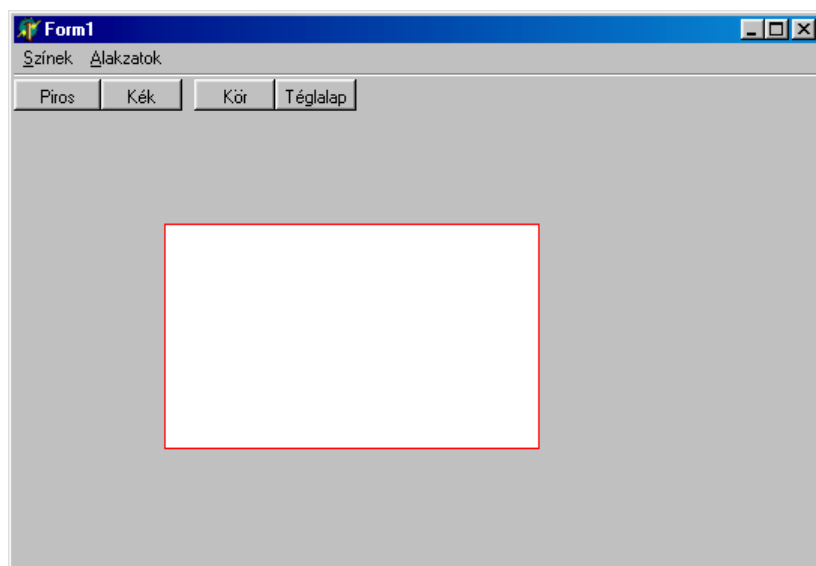
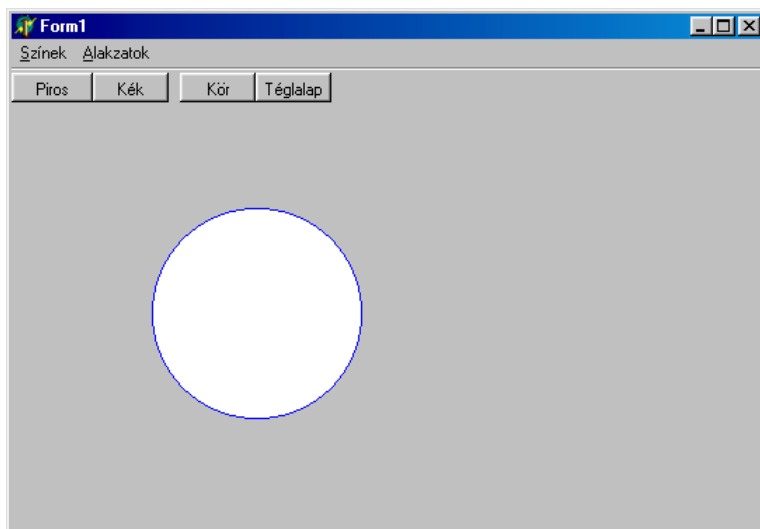
A *Kör* menüpont megnyomásakor meghívódik a *KorClick* eseménykezelő eljárásban az *alakzat* változóba írjuk a 1-et jelentő kört, majd kérünk egy ablak újrifestést az *Invalidate* metódus hívásával. Ugyancsak a *KorClick* eseménykezelő eljárás hívódik meg, ha a *Kör* eszközgombot nyomjuk meg.

```
procedure TForm1.KorClick(Sender: TObject);
begin
    alakzat :=1;
    Invalidate;
end;
```

A *Téglalap* menüpont megnyomásakor meghívódik a *TeglalapClick* eseménykezelő eljárásban az *alakzat* változóba írjuk a 2-et jelentő téglalapot, majd kérünk egy ablak újrifestést az *Invalidate* metódus hívásával. Ugyancsak a *TeglalapClick* eseménykezelő eljárás hívódik meg, ha a *Téglalap* eszközgombot nyomjuk meg.

```
procedure TForm1.TeglalapClick(Sender: TObject);
begin
    alakzat :=2;
    Invalidate;
end;
```

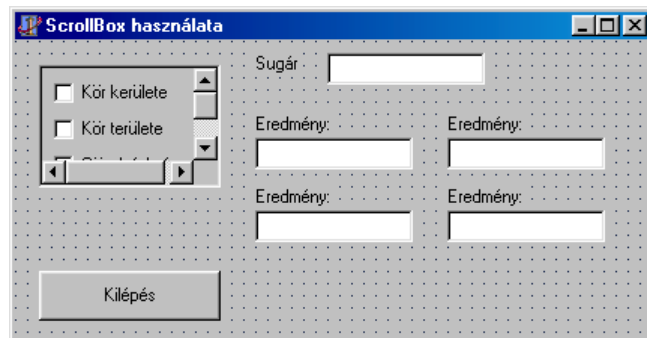
A program néhány futási képe:





Számítsuk ki a kör területét, kerületét a gömb felszínét és térfogatát! A számításhoz használjunk *ScrollBar* vezérlőben tárolt jelölőnégyzeteket, amelyeken való kattintáskor az eredménykijelzőben megjelenik az eredmény! (*ScrollBar*)

A feladatnak megfelelően a *Form1* űrlapot az alábbiakban építettük fel:



A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    ScrollBox3: TScrollBar;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    CheckBox3: TCheckBox;
    CheckBox4: TCheckBox;
    Kilepes: TButton;
    Label1: TLabel;
    Sugar: TEdit;
    Label2: TLabel;
    Ered1: TEdit;
    Label3: TLabel;
    Ered2: TEdit;
    Label4: TLabel;
    Ered3: TEdit;
    Label5: TLabel;
    Ered4: TEdit;
    procedure CheckBox1Click(Sender: TObject);
    procedure CheckBox2Click(Sender: TObject);
    procedure CheckBox3Click(Sender: TObject);
    procedure CheckBox4Click(Sender: TObject);
    procedure KilepesClick(Sender: TObject);
  end;
```

A "Kör kerülete" jelölőnégyzet kijelölésekor a *CheckBox1Click* eseménykezelő eljárás hívódik meg, melyben ha a sugár nem üres sztring, kiszámítjuk és megjelenítjük a kör kerületét. Kivételkezeléssel akadályozzuk meg a hibás adatmegadást.

```
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  if (CheckBox1.Checked) and (Sugar.Text <> '') then
  begin
    try
      r := StrToFloat(Sugar.Text);
    except
      r:=0;
    end;
    if r > 0 then
    begin
      Label2.Caption := 'Kör kerülete';
      Ered1.text := FloatToStr(2*r*pi);
    end
  end
  else
    Ered1.text := '';
end;
```

A "Kör területe" jelölőnégyzet kijelölésekor a *CheckBox2Click* eseménykezelő eljárás hívódik meg, amelyben ha a sugár nem üres string, kiszámítjuk és megjelenítjük a kör területét Kivételkezeléssel akadályozzuk meg a hibás adatmegadást.

```
procedure TForm1.CheckBox2Click(Sender: TObject);
begin
    if (CheckBox2.Checked) and (Sugar.Text <> '') then
    begin
        try
            r := StrToFloat(Sugar.Text);
        except
            r:=0;
        end;
        if r > 0 then
        begin
            Label3.Caption := 'Kör területe';
            Ered2.text := FloatToStr(r*r*pi);
        end
    end
    else
        Ered2.Text := '';
end;
```

A "Gömb felszíne" jelölőnégyzet kijelölésekor a *CheckBox3Click* eseménykezelő eljárás hívódik meg, melyben ha a sugár nem üres string, kiszámítjuk és megjelenítjük a gömb felszínét. Kivételkezeléssel akadályozzuk meg a hibás adatmegadást.

```
procedure TForm1.CheckBox3Click(Sender: TObject);
begin
    if (CheckBox3.Checked) and (Sugar.Text <> '') then
    begin
        try
            r := StrToFloat(Sugar.Text);
        except
            r:=0;
        end;
        if r > 0 then
        begin
            Label4.Caption := 'Gömb felszíne';
            Ered3.text := FloatToStr(4*r*r*pi);
        end
    end
    else
        Ered3.text := '';
end;
```

A "Gömb térfogata" jelölőnégyzet kijelölésekor a *CheckBox4Click* eseménykezelő eljárás hívódik meg, melyben ha a sugár nem üres string, kiszámítjuk és megjelenítjük a gömb térfogatát. Kivételkezeléssel akadályozzuk meg a hibás adatmegadást.

```
procedure TForm1.CheckBox4Click(Sender: TObject);
begin
    if (CheckBox4.Checked) and (Sugar.Text <> '') then
    begin
        try
            r := StrToFloat(Sugar.Text);
        except
            r:=0;
        end;
        if r > 0 then
        begin
            Label5.Caption := 'Gömb térfogata';
            Ered4.text := FloatToStr(4*r*r*r*pi/3);
        end
    end
    else
        Ered4.text := '';
end;
```

A program futási képe:

ScrollBar használata

☒ Kör területe
☒ Gömb felszíne
☒ Gömb térfogata

Sugár: 1

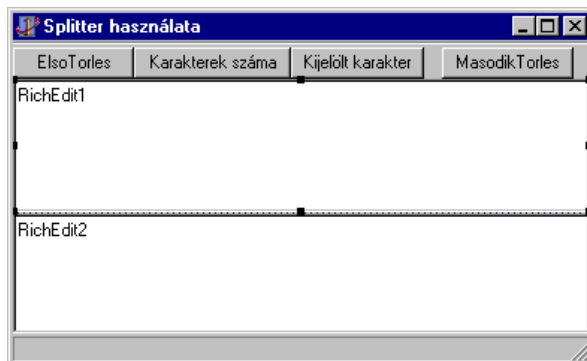
Kör kerülete	Gömb felszíne
6,28318530717959	12,5663706143592
Kör területe	Gömb térfogata
3,14159265358979	4,18879020478639

Kilépés



Két **Memo** vezérlőt válasszunk el egy **Splitter** vezérlővel, hogy könnyen változtatható legyen a méretük! Használjunk **ToolBar**-on elhelyezett nyomógombot a vezérlők törlésére, valamint az egyik vezérlő karaktereinek megszámlálására! (**Splitter**)

A feladatnak megfelelően a Form1 űrlap az alábbi vezérlőket tartalmazza:



A vezérlők néhány fontos tulajdonságának beállított értéke:

<i>RichEdit1</i>	<i>Align</i>	<i>alTop</i>
<i>RichEdit2</i>	<i>Align</i>	<i>alClient</i>
<i>Splitter1</i>	<i>Align</i>	<i>alTop</i>
	<i>Cursor</i>	<i>crVSplit</i>
	<i>ResizeStyle</i>	<i>rsPattern</i>

A *TForm1* osztály deklarációja:

```

type
  TForm1 = class(TForm)
    StatusBar1: TStatusBar;
    SpeedButton1: TSpeedButton;
    RichEdit1: TRichEdit;
    RichEdit2: TRichEdit;
    Splitter1: TSplitter;
    ToolBar1: TToolBar;
    ElsőTorles: TButton;
    MasodikTorles: TButton;
    Button1: TButton;
    Button2: TButton;
    Label1: TLabel;
    procedure ElsőTorlesClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure MasodikTorlesClick(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

```

A "Elsőablak törlése" nyomógomb megnyomásakor meghívódó *ElsőTorlesClick* eseménykezelő eljárásban a *RichEdit1* vezérlő tartalmát a *Clear* metódus hívásával töröljük.

```

procedure TForm1.ElsőTorlesClick(Sender: TObject);
begin
  RichEdit1.Clear;
end;

```

A "Karakterek száma" nyomógomb megnyomásakor meghívódó *Button1Click* eseménykezelő eljárásban a *RichEdit1* szövegmező karakterhosszát üzenetablakban jelenítjük meg.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  a: integer;
begin
  a:=RichEdit1.GetTextLen;
  ShowMessage('Karakterek száma: '+IntToStr(a));
end;
```

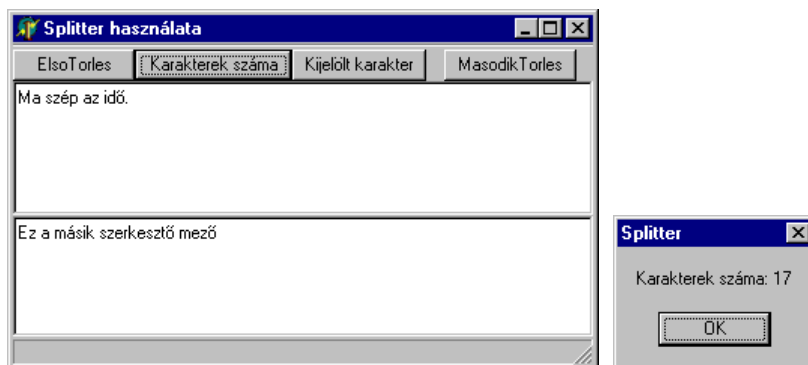
A "Második ablak törlése" nyomógomb megnyomásakor meghívódó *MasodikTorlesClick* eseménykezelő eljárásban a *RichEdit2* vezérlő tartalmát a *Clear* metódus hívásával töröljük.

```
procedure TForm1.MasodikTorlesClick(Sender: TObject);
begin
  RichEdit2.Clear;
end;
```

A "Kijelölt karakter törlése" nyomógomb megnyomásakor meghívódó *Button2Click* eseménykezelő eljárásban töröljük a kijelölést, ha van.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  if RichEdit1.SelLength <> 0 then
  begin
    RichEdit1.SelText:='';
  end
end;
```

A program futási képe:





A form létrehozásakor intézzük el az elemek típusos konstans által meghatározott méretű táblázat elemeinek feltöltését.

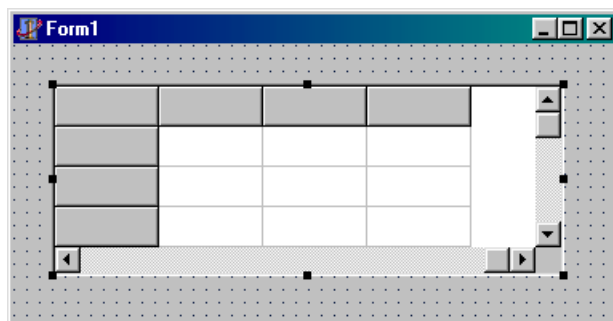
```
procedure TForm1.FormCreate(Sender: TObject);
var i,j:integer;
begin
  Caption:='Pascal háromszög';
  // A táblázat illeszkedik a formhoz
  StringGrid1.Align:=alClient;
  with StringGrid1 do
    begin
      ColCount:=meret;
      RowCount:=meret;
      FixedCols:=1;
      FixedRows:=1;
      Cells[0,0]:=Caption;
      for i:=1 to meret do
        begin
          // A címsorban és oszlopban a sorszám
          Cells[i,0]:=InttoStr(i);
          Cells[0,i]:=InttoStr(i);
          // Az első sor és oszlop csupa egyes
          Cells[i,1]:=InttoStr(1);
          Cells[1,i]:=InttoStr(1);
        end;
      for i:=1 to meret do
        for j:=1 to meret do
          // A bal oldali és a fölöttelévő elem összege adja a cella értékét
          Cells[i,j]:=InttoStr(strtoint(Cells[i,j-1])+strtoint(Cells[i-1,j]));
        end;
      end;
    end;
end;
```

Pascal három	1	2	3	4	5	6	7	8
1	2	4	7	11	16	22	29	37
2	4	8	15	26	42	64	93	130
3	7	15	30	56	98	162	255	385
4	11	26	56	112	210	372	627	1012
5	16	42	98	210	420	792	1419	2431
6	22	64	162	372	792	1584	3003	5434
7	29	93	255	627	1419	3003	6006	11440
8	37	130	385	1012	2431	5434	11440	22880
9	46	176	561	1573	4004	9438	20878	43758
10	56	232	793	2366	6370	15808	36686	80444
11	67	299	1092	3458	9828	25636	62322	142766



Az 5. fejezet áttanulmányozása után készítsünk 10x10-es szorzótábla programot, ahol a jobb olvashatóságot saját rajzolatú cellák biztosítják! (*Szorzotabla*)

Helyezzünk a formra egy *StringGrid* vezérlőt!



A kilépés gomb feliratát **resourcestring**-ben tároljuk. Az aktuális cella koordinátái sz *SRow* és *SCol* egészek.

```
resourcestring
    sExit='Exit';

var
    // A kiválasztott cella koordinátái
    SRow:integer=-1;
    SCol:integer=-1;
```

A form létrehozásakor elvégezzük az alapbeállításokat.

```
procedure TForm1.FormCreate(Sender: TObject);
var
    i,j : Integer;
begin
    with StringGrid1 do
        begin
            // Beállítások
            Options:=Options+[goTabs]-
                [goRangeSelect];

            // Méretek
            Height:=360;
            Width:=345;
            ColCount:=11;
            RowCount:=11;
            for i:=0 to 10 do
                begin
                    colwidths[i]:=30;
                    rowheights[i]:=30;
                end;
            //Színek
            Color:=clYellow;
            FixedColor:=clBlue;
            Font.Color:=clBlue;
            Font.size:=12;
            // A rögzített cellák feltöltése
            for i:=1 to 11 do
                begin
                    cells[i,0]:=format('%2d',[i]);
                    cells[0,i]:=format('%2d',[i]);
                end;
            // A táblázat feltöltése
            for i:=1 to 10 do
                for j:=1 to 10 do
                    cells[i,j]:=format('%d',[i*j]);
                end;
            end;
        end;
end;
```

A cellák kiválasztásakor átírjuk az *SRow* és *SCol* változókat, valamint újrafestjük a táblázatot.

```
procedure TForm1.StringGrid1SelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  SRow:=ARow;
  SCol:=ACol;
  StringGrid1.Invalidate;
end;
```

A kifestés az **OnDrawCell** eseményben történik. A *[0,0]* cellát, a rögzített cellákat - kiválasztott és nem kiválasztott állapotban -, valamint a fókuszált cellát másképpen jelenítjük meg. Az eltérő megjelenés úgy biztosítható, hogy az cellakoordinátáknak megfelelően változtatjuk a betűtípusokat, színeket és a kiírás módját.

```
procedure TForm1.StringGrid1DrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
begin
  // [0,0] cella megjelenítése
  if (ARow=0) and (ACol=0) then
  begin
    with Stringgrid1.Canvas do
    begin
      Font.Color:=clYellow;
      Font.size:=10;
      Font.style:=[fsBold];
      Brush.Color:=Clred;
      TextRect(Rect,Rect.Left+2,Rect.Top+6,sExit)
    end;
  end;
  // A rögzített cellák megjelenítése
  if (gdFixed in State) and (ARow<>ACol) then
  begin
    if (ARow=SRow) or (ACol=SCol) then
      // A kiválasztott cella sora, oszlopa
    begin
      with Stringgrid1, Canvas do
      begin
        Font.Color:=clWhite;
        Font.size:=12;
        Brush.Color:=rgb(255,0,128);
        if (ARow=0) then
          TextRect(Rect,Rect.Left,Rect.Top+10,cells[ARow,ACol])
        else
          TextRect(Rect,Rect.Left+10,Rect.Top,cells[ARow,ACol]);
        end;
      end;
    end;
  else
    // Nem a kiválasztott cella sora, oszlopa
  begin
    with Stringgrid1, Canvas do
    begin
      Font.Color:=clWhite;
      Font.size:=12;
      Brush.Color:=ClBlue;
      TextRect(Rect,Rect.Left,Rect.Top,cells[ARow,ACol]);
    end;
  end;
  // A fókuszált cella megjelenítése
  if (gdFocused in State) then
  begin
    with Stringgrid1, Canvas do
    begin
      Font.Color:=clWhite;
      Font.size:=16;
      Font.style:=[fsBold];
      Brush.Color:=rgb(255,0,128);
      TextRect(Rect,Rect.Left+2,Rect.Top+2,cells[ARow,ACol]);
    end;
  end;
end;
```

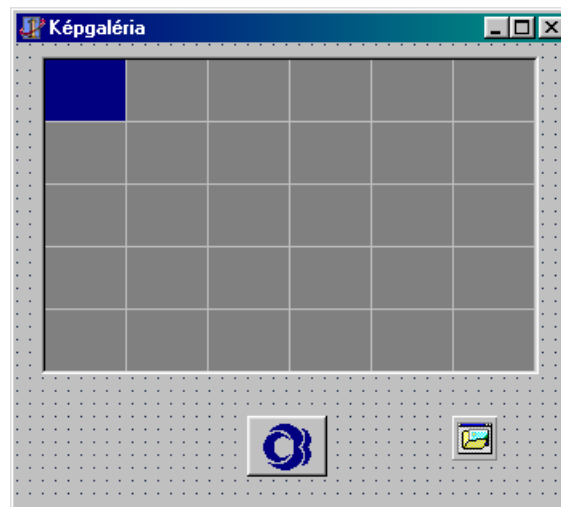
Gombnyomáskor csak azt figyeljük, hogy kilépünk-e a programból.

```
procedure TForm1.StringGrid1MouseDown(Sender: TObject;  
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);  
var  
  Arow, Acol : Integer;  
begin  
  StringGrid1.MouseToCell(x,y,arow,acol);  
  // Kilépés a programból  
  if (Arow=0) and (Acol=0) then form1.close;  
end;
```

Exit	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100



Helyezzünk egy **StringGrid** komponenst, egy **BitBtn** komponenst a formra! Szükségünk lesz az **OpenPictureDialog** általános párbeszédablakra is.



Megakadályozzuk az ablak átméretezését:

```
procedure TForm1.FormCanResize(Sender: TObject; var NewWidth,
    NewHeight: Integer; var Resize: Boolean);
begin
    Resize:=false;
end;
```

A nyomógomb eseménykezelőjében választhatunk a képek közül.

```
// A BitBtn1 gomb eseménykezelőjében feltöltjük az
// új bitképet
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
    if OpenPictureDialog1.Execute then
        try
            BitBtn1.Glyph.LoadFromFile(OpenPictureDialog1.FileName);
        except
            Beep;
        end;
end;
```

A cella választásakor, ha nincs gomb a cellában, akkor az **Objects** tulajdonságot használva betöltünk egy gombot, amelynek képe megegyezik a **BitBtn1** képével, egérgombnyomás eseményének kezelője pedig az **EgerGomblenyomasa** eljárás lesz.

```

// A cella kiválasztásának eseménykezelője
procedure TForm1.StringGrid1SelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  // Ha meg nincs gomb a cellában létrehozzuk azt
  with StringGrid1 do
    if not (Objects[ACol, ARow] is TBitBtn) then
      begin
        Objects[ACol, ARow]:=TBitBtn.Create(StringGrid1);
        with (Objects [ACol, ARow] As TBitBtn) do
          begin
            // A gomb tulajdonságainak beállítása
            Parent:=StringGrid1;
            Visible:=True;
            // Mérete a cellaméret
            Boundsrect:=CellRect(ACol, ARow);
            // A gomb bitképe megegyezik a BitBtn1 gombéval
            Glyph := TBitmap.Create;
            Glyph:=BitBtn1.Glyph;
            // A egérgomblenyomás esemény kezelése
            OnMouseDown:=EgorgombLenyomasa;
          end;
        end;
      end;
end;

```

Ha van gomb a cellában, akkor azon való kattintás azt eredményezi, hogy a *BitBtn1* képe lesz a gomb képe.

```

// A gombok közös eseménykezelője
procedure TForm1.EgorgombLenyomasa(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  // A gombon való egérekattintás alkalmával
  // a gomb bitképe frissül a BitBtn1 képével
  (Sender as TBitBtn).Glyph:=BitBtn1.Glyph;
end;

```

A form zárásakor takarítunk.

```

// A form lezárásakor felszabadítjuk a lefoglalt erőforrásokat
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
var i, j: integer;
begin
  with StringGrid1 do
    for i:=0 to ColCount-1 do
      for j:=0 to RowCount-1 do
        if (Objects[i, j] is TBitBtn) then
          Objects[i, j].Free;
end;

```

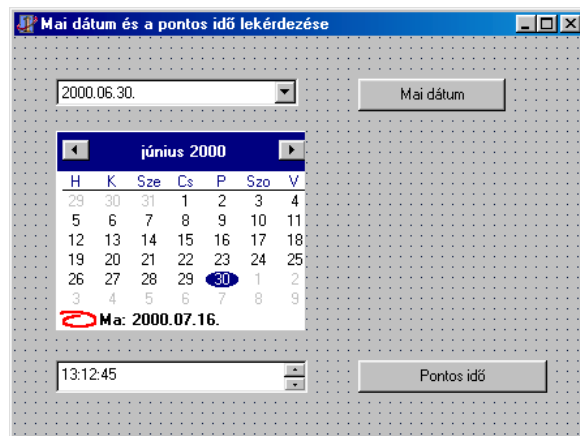




Két **DateTimePicker** és egy **MonthCalendar** vezérlőt, valamint két nyomógombot helyezünk az űrlapra

A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    DateTimePicker1: TDateTimePicker;
    MonthCalendar1: TMonthCalendar;
    Button1: TButton;
    DateTimePicker2: TDateTimePicker;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  end;
```



A "Mai dátum" nyomógomb megnyomásakor meghívódó *Button1Click* eseménykezelő eljárásban a **Date** függvénnyel lekérdezzük az aktuális dátumot és értékül adjuk a *MonthCalendar1* és a *DateTimePicker1* vezérlők **Date** tulajdonságának. A dátum megjelenítéshez a *DateTimePicker1* vezérlő **Kind** tulajdonságát *dtkDate* értékre kell állítanunk.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  MonthCalendar1.Date := Date;
  DateTimePicker1.Kind := dtkDate;
  DateTimePicker1.Date := Date;
end;
```

A "Pontos idő" nyomógomb megnyomásakor meghívódó *Button2Click* eseménykezelő eljárásban *DateTimePicker2* vezérlő **Kind** tulajdonságát *dtkTime* értékre kell állítanunk az idő kijelzésére. A **Time** függvény hívásával pedig beállítjuk a vezérlő **Time** tulajdonságát.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  DateTimePicker2.Kind := dtkTime;
  DateTimePicker2.Time := Time;
end;
```

A program futási képe:

