

## 7.1 Fájlkezelés

1. Numerikus adatok szövegfájlban [TextFile1](#)
2. Diákok adatai szöveges fájlban [Diak](#)
3. Numerikus adatok típusos fájlban [RecFile1](#)
4. Könyvadatok típusos fájlban [Konyvtar](#)
5. Állományok másolása típusnélküli fájlokkal [FMasolasBl](#)



Készítsünk programot, amely alkalmas számadatok megadására, az adatok szöveges fájlba történő mentésére, illetve állományból való olvasására! Készítsük fel a programot számadatok összegének, átlagának kiszámítására és a szélsőértékek megkeresésére is! (*TextFile1*)

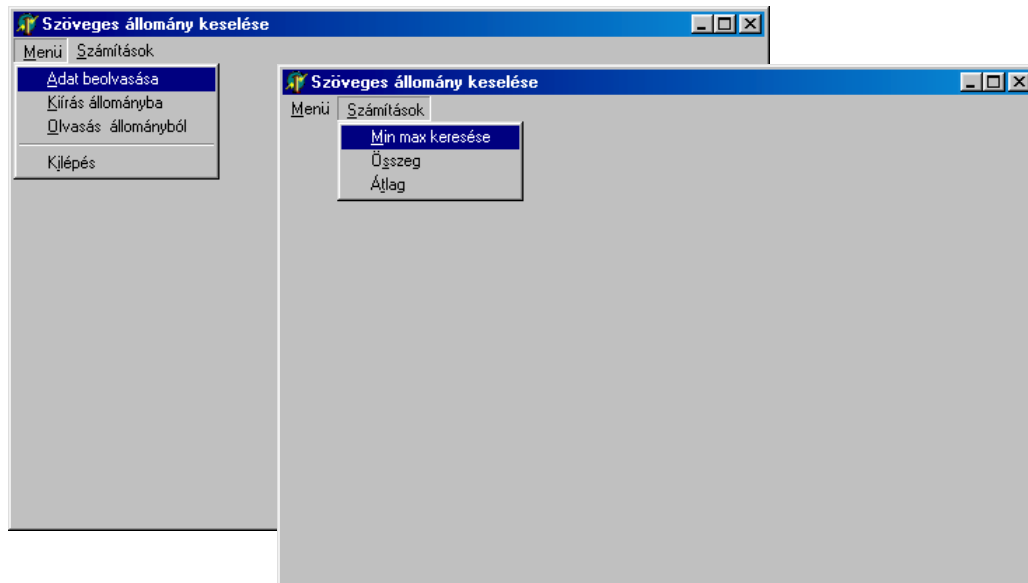
Tervezzük meg a program menürendszerét!

#### **Menü**

Adat beolvasása  
Kiírás állományba  
Olvasás állományból  
Kilépés

#### **Számítások**

Min max keresése  
Összeg  
Átlag



A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    Menu: TMenuItem;
    Adat: TMenuItem;
    Kiir: TMenuItem;
    Olvas: TMenuItem;
    Nl: TMenuItem;
    Kilepes: TMenuItem;
    Szamitasok: TMenuItem;
    MinMaxKeres: TMenuItem;
    OpenFileDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    Osszeg: TMenuItem;
    Atlag: TMenuItem;
    procedure AdatClick(Sender: TObject);
    procedure KiirClick(Sender: TObject);
    procedure OlvasClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure KilepesClick(Sender: TObject);
    procedure MinMaxKeresClick(Sender: TObject);
    procedure OsszegClick(Sender: TObject);
    procedure AtlagClick(Sender: TObject);
  end;
```

A "Adat beolvasása" menüpont kiválasztásakor az *AdatClick* eseménykezelő eljárás hívódik meg, amelyben megjelenítjük a *Form2* formot.

```
procedure TForm1.AdatClick(Sender: TObject);
begin
    Form2.ShowModal;
end;
```

A "Kiírás állományba" menüpont kijelölésekor a *KiirClick* eseménykezelő eljárás aktivizálódik, amelyben az adatokat .TXT kiterjesztésű szöveges állományba írjuk. A *SaveDialog* párbeszédablakban adjuk meg az állomány nevét.

```
procedure TForm1.KiirClick(Sender: TObject);
var
    f1 : TextFile;
    i:integer;
begin
    if x_db > 0 then
    begin
        SaveDialog1.Filter := 'Text files (*.txt)|*.txt|All files (*.*)|*.*';
        SaveDialog1.FilterIndex := 1;
        if SaveDialog1.Execute then
        begin
            AssignFile(f1,SaveDialog1.Filename);
            Rewrite(f1);
            for i:=1 to x_db do
            begin
                writeln(f1,x[i]);
            end;
            CloseFile(f1);
        end;
    end;
end;
```

A "Olvasás állományból" menüpont kijelölésekor az *OlvasClick* eseménykezelő eljárás hívódik meg, amelyben az adatokat .TXT kiterjesztésű állományból olvassuk be.

```
procedure TForm1.OlvasClick(Sender: TObject);
var
    i:integer;
    f2:TextFile;
begin
    OpenFileDialog1.Filter := 'Text files (*.txt)|*.txt|All files (*.*)|*.*';
    OpenFileDialog1.FilterIndex := 1;
    If OpenFileDialog1.Execute then
    begin
        AssignFile(f2,OpenDialog1.Filename);
        Reset(f2);
        i:=0;
        Form2.Adatok.Clear;
        While not Eof(f2) do
        begin
            i:=i+1;
            Readln(f2,x[i]);
            Form2.Adatok.Items[i-1]:=FloatToStr(x[i]);
        end;
        x_db := i;
        CloseFile(f2);
    end;
end;
```

A *Kilépés* menüpont kiválasztásával a program befejezi a működését.

```
procedure TForm1.KilepesClick(Sender: TObject);
begin
    Application.Terminate;
end;
```

Az "Min max keresése" menüpont kiválasztásakor a *MinMaxClick* eseménykezelő eljárás hívódik meg, amelyben a *MinMax* eljárással megkeressük a tömb minimális és maximális elemét, amiket a *Form4* form *Edit1*, illetve *Edit2* szövegmezőjébe írunk, majd pedig megjelenítjük formot.

```
procedure TForm1.MinMaxClick(Sender: TObject);
var
  n,k:real;
begin
  MinMax(x,x_db,k,n);
  Form4.Edit1.Text:=FloatToStr(k);
  Form4.Edit2.Text:=FloatToStr(n);
  Form4.ShowModal;
end;
```

Az "Összeg" menüpont kiválasztásakor az *OsszegClick* eseménykezelő eljárás aktivizálódik, amelyben *ShowMessage* hívásával megjelenítjük az *OsszegSzamitas* függvény segítségével meghatározott elemösszeget.

```
procedure TForm1.OsszegClick(Sender: TObject);
begin
  ShowMessage('Összege: ' + FloatToStr(OsszegSzamitas(x,x_db)));
end;
```

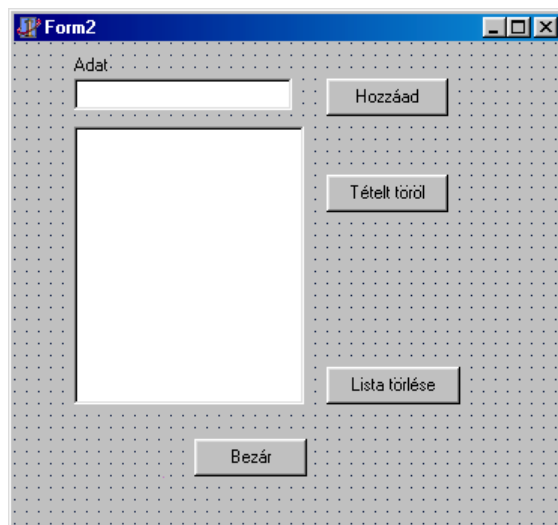
Az "Átlag" menüpont kiválasztásakor meghívódó *AtlagClick* eseménykezelő eljárásban a *ShowMessage* hívásával jelenítjük meg az elemek átlagát. A számítást az *AtlagSzamitas* függvény hívásával végezzük el.

```
procedure TForm1.AtlagClick(Sender: TObject);
begin
  ShowMessage('Átlaga: ' + FloatToStr(AtlagSzamitas(x,x_db)));
end;
```

A *Form2* űrlapon az adatok beolvasására egy szövegmezőt használunk, melynek a tartalmát a *Hozzáad* nyomógommbal a listához adjuk. A kijelölt tételt a listából a "Tétel töröl" nyomógommbal törölhetjük. A teljes listát pedig a "Lista törlése" nyomógommbal törölhetjük.

A *TForm2* osztály deklarációja:

```
type
  TForm2 = class(TForm)
    Adatok: TListBox;
    Adat: TEdit;
    Label1: TLabel;
    HozzaAd: TButton;
    Teteltorles: TButton;
    Listatorles: TButton;
    Bezar: TButton;
  procedure ListatorlesClick(Sender: TObject);
  procedure HozzaAdClick(Sender: TObject);
  procedure TeteltorlesClick(Sender: TObject);
  procedure BezarClick(Sender: TObject);
  end;
```



Az "Lista törlése" nyomógomb megnyomásakor a *ListaTorlésClick* eseménykezelő eljárásra kerül a vezérlés, amelyben az *Adatok* lista *Clear* metódusának hívásával a lista törlődik.

```
procedure TForm2.ListatorlesClick(Sender: TObject);
begin
  Adatok.Clear;
end;
```

Az "Hozzáad" nyomógomb megnyomásakor a *HozzaAdClick* eseménykezelő aktivizálódik, amelyben ha az *Adat* adatmező nem üres, és a benne lévő adat nincs a listában (*IndexOf*), akkor az *Items* tulajdonság *Add* metódusával az adatmező tartalmát a listához adjuk. Ezt követően töröljük az adatmező tartalmát, és a fókuszot a *SetFocus* metódussal a szövegmezőre állítjuk.

```
procedure TForm2. HozzaAdClick(Sender: TObject);
begin
    if Adat.text <> '' then
        begin
            if Adatok.Items.IndexOf(Adat.Text) = -1 then
                Adatok.Items.Add(Adat.Text);
            Adat.Clear;
            Adat.SetFocus;
        end;
    end;
end;
```

A "Tételt töröl" nyomógomb megnyomásakor a *TételTorlésClick* eseménykezelő eljárás hívódik meg, amelyben, ha van a listának kijelölt eleme (ha a lista *ItemIndex* tulajdonsága nem negatív), akkor töröljük azt az *Items* tulajdonság *Delete* metódusának hívásával. Ezután töröljük az *Adat* adatmező tartalmát, és a fókuszot a *SetFocus* metódussal a szövegmezőre állítjuk.

```
procedure TForm2.TeteltorlesClick(Sender: TObject);
begin
    If Adatok.ItemIndex > -1 then
        begin
            Adatok.Items.Delete(Adatok.ItemIndex);
            Adat.Text := '';
            Adat.Setfocus;
        end;
    end;
end;
```

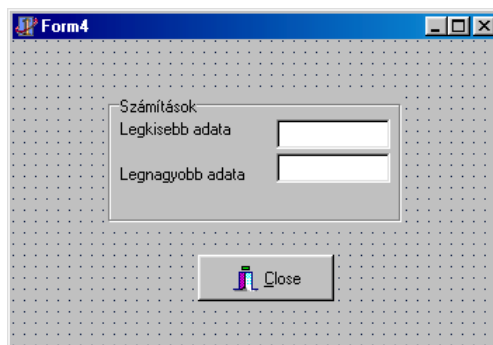
A "Bezár" nyomógomb megnyomásakor meghívódó *BezársClick* eseménykezelő eljárásban az *x* tömbbe másoljuk a lista tartalmát. Ha az *Adatok* lista *Items* tulajdonságának *Count* mezeje nagyobb nullánál, akkor a lista *Items* tömbjének az elemeit áttöltjük valóssá konvertálva (*StrToFloat*) az *x* tömbbe. Kivételkezelést alkalmazunk arra az esetre, ha a lista nem numerikus értéket tárolna.

```
procedure TForm2.BezarClick(Sender: TObject);
var
    i:integer;
begin
    try
        If Adatok.Items.Count <> 0 then
            begin
                for i:=0 to Adatok.Items.Count-1 do
                    begin
                        x[i+1] := StrToFloat(Adatok.Items[i]);
                    end;
                x_db := Adatok.Items.Count;
            end;
        except
            On EConvertError do
                ShowMessage('Konvetálási hiba');
            end;
        Close;
    end;
end;
```

A *Form4* űrlap jeleníti meg az *x* tömb legkisebb, illetve legnagyobb elemét:

A *TForm4* osztály deklarációja:

```
type
  TForm4 = class(TForm)
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    Edit2: TEdit;
    BitBtn1: TBitBtn;
    procedure FormCreate(Sender: TObject);
  end;
```



A *FormCreate* eseménykezelő eljárásban az *Edit1* és az *Edit2* adatmezők tartalmát töröljük.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.Clear;
  Edit2.Clear;
end;
```

A *Unit3* tartalmazza a tömbspecifikációt és a globális deklarációkat, az összeg számítására az *OsszegSzamitas* függvényt, az átlag számítására az *AtlagSzamitas* függvényt, valamint a *MinMax* eljárást a tömb minimális és maximális elemének a keresésére.

```
unit Unit3;
interface
type
  tomb = array[1..20] of real;
var
  x : tomb;
  x_db : integer;
  min_ertek,max_ertek:real;
  function OsszegSzamitas(y:tomb;n:integer):real;
  function AtlagSzamitas(y:tomb;n:integer):real;
  procedure MinMax(y:tomb;n:integer; var min,max:real);
implementation

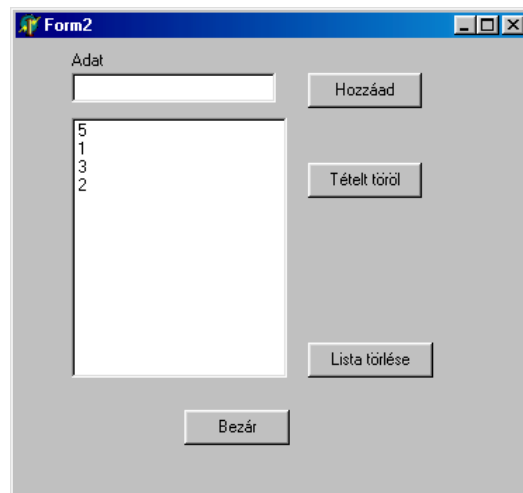
function OsszegSzamitas(y:tomb;n:integer):real;
var
  i:integer;
begin
  result:=0;
  for i:=1 to n do
    result:=result+y[i];
  end;

function AtlagSzamitas(y:tomb;n:integer):real;
begin
  result:=OsszegSzamitas(y,n)/n;
end;

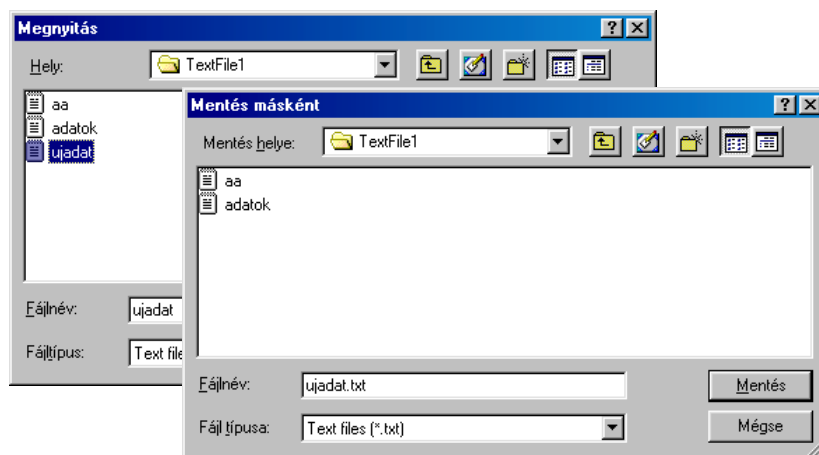
procedure MinMax(y:tomb;n:integer; var min,max:real);
var
  i : integer;
begin
  min := y[1];
  max := y[1];
  for i:=2 to n do
  begin
    if min>y[i] then min := y[i];
    if max<y[i] then max := y[i];
  end;
end;
end.
```

A program néhány futási képe:

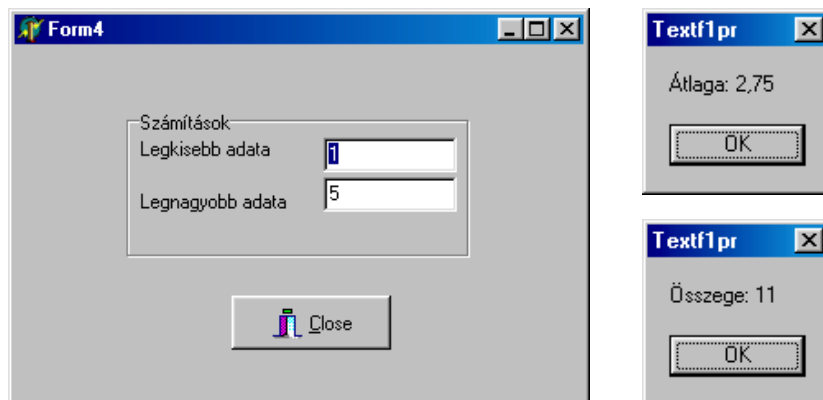
Az adatok megadása:



Az adatok beolvasása és kimentése:



A legkisebb és a legnagyobb adat megkeresése, az összeg és az átlag:





Készítsünk programot, amely alkalmas diákok adatainak (név, tanulókör, kreditpont) kezelésére! Tároljuk az adatokat szöveges fájlban! (*Diak*)

Először megtervezzük a program működéséhez szükséges osztályokat és metódusokat, melyet a *DiakAdat* form nélküli unitban tárolunk.

Két objektumtípust definiálunk:

1. A *TDiak* objektumtípus tartalmazza a diák nevét, tankörszámát és kreditpontját. Az *elozo* és *kovetkezo* adatmezői lehetővé teszik az objektum láncbafűzését. Az osztálynak két metódusa van, az objektumot *Dcreate* konstruktor hozza létre, és a *Dfree* metódus szünteti meg.

```
TDiak = class
  nev: string[20];
  tankor: integer;
  kreditpont: integer;
  eloza, kovetkezo: TDiak;
  constructor DCreate;
  procedure DFree;
end;
```

Az objektumot létrehozó *DCreate* konstruktor:

```
constructor TDiak.DCreate;
begin
  nev      := '';
  tankor   := 0;
  kreditpont := 0;
  eloza    := nil;
  kovetkezo := nil;
end;
```

Az objektumot megszüntető *DFree* metódus:

```
procedure TDiak.DFree;
begin
  Free;
end;
```

2. A *TEvf* objektumtípus *első* és *utolsó* adatmezeje tartja nyilván a *TDiak* objektumlánc első és utolsó elemét. Metódusai - a konstruktor és objektumot megszüntető metóduson kívül - lehetővé teszik a névszerinti keresést, a legkisebb és a legnagyobb kreditponttal rendelkező diák kiválasztását.

```
TEvf = class
  első, utolsó: TDiak;
  constructor ECreate;
  procedure EFree;
  function KeresNevSzerint(Neve:string):TDiak;
  function MinKreditpont:integer;
  function MaxKreditpont:integer;
end;
```

Az objektumot létrehozó *ECreate* konstruktor:

```
constructor TEvf.ECreate;
begin
  első := nil;
  utolsó := nil;
end;
```



Az objektumot megszüntető *EFree* metódus:

```
procedure TEvf.EFree;
var
  p : TDiak;
begin
  if Assigned(elso) then
  begin
    p := elso;
    while Assigned(p) do
    begin
      elso := elso.kovetkezo;
      p.DFree;
      p := elso;
    end;
  end;
end;
```

A *KeresNevSzerint* metódus a paraméterben megadott nevű diákot megkeresi a láncban, ha megtalálta, akkor az objektum referenciáját adja vissza:

```
function TEvf.KeresNevSzerint (Neve:string) :TDiak;
var
  p, van:TDiak;
begin
  p:= elso;
  van := nil;
  while p <> nil do
  begin
    if Neve = p.nev then
    begin
      van := p;
      break;
    end;
  end;
  KeresNevSzerint:=van;
end;
```

A *MinKreditpont* metódus a minimális kreditpontot adja vissza.

```
function TEvf.MinKreditpont:integer;
var
  p      : TDiak;
  minimum: integer;
begin
  p:= elso;
  minimum := elso.kreditpont;
  while p <> nil do
  begin
    if p.kreditpont< minimum
    then
    begin
      minimum:= p.kreditpont;
    end;
    p := p.kovetkezo;
  end;
  MinKreditpont:=minimum;
end;
```

A *MaxKreditpont* metódus a maximális kreditpontot adja vissza.

```
function TEvf.MaxKreditpont:integer;
var
  p : TDiak;
  maximum: integer;
begin
  p:= elso;
  maximum := elso.kreditpont;
  while p <> nil do
    begin
      if p.kreditpont> maximum
      then
        begin
          maximum:= p.kreditpont;
        end;
      p:=p.kovetkezo;
    end;
  MaxKreditpont:=maximum;
end;
end.
```

Globális deklarációk:

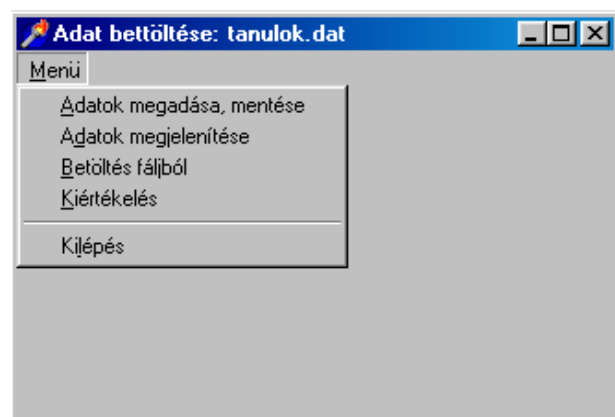
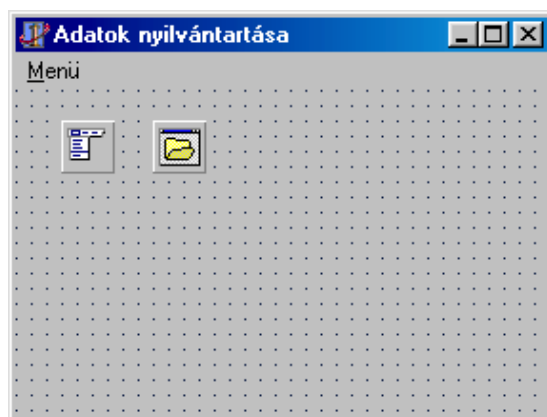
```
var
  diak : TDiak;
  evf : TEvf;
  FNev : string;
```

Ezek után tervezzük meg a program menüjét a *form1* űrlapon!

#### **Menü**

*Adatok megadása, mentése*  
*Adatok megjelenítése*  
*Betöltés fájlból*  
*Kiértékelés*  
*Kilépés*

A megvalósított program menürendszere:



### ***A TForm1 osztály deklarációja:***

```
type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    Menu: TMenuItem;
    Adatokmegadasa: TMenuItem;
    Betoltes: TMenuItem;
    Kiertekeles: TMenuItem;
    N1: TMenuItem;
    Kilepes: TMenuItem;
    OpenDialog1: TOpenDialog;
    Megjelenites: TMenuItem;
    procedure AdatokmegadasaClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure KilepesClick(Sender: TObject);
    procedure BetoltesClick(Sender: TObject);
    procedure MegjelenitesClick(Sender: TObject);
    procedure KiertekelesClick(Sender: TObject);
  end;
```

A *FormCreate* eseménykezelő eljárásban az *Evf* változóban létrehozuk a *TEvf* osztály egy példányát, valamint az *Evf* objektum *elso* mezejét **nil**-re állítjuk. Az "*Adatok megjelenítése*" és a "*Kiértékelés*" menüpontokat letiltjuk.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Evf := TEvf.ECreate;
  Evf.elso := nil;
  Megjelenites.Enabled := false;
  Kiertekeles.Enabled := false;
end;
```

A "*Kilépés*" nyomógomb megnyomásakor meghívódik a *KilépésClick* eseménykezelő eljárás, amelyben az *Evf* objektumot megszüntetjük, és a program befejezi a működését.

```
procedure TForm1.KilepesClick(Sender: TObject);
begin
  Evf.EFree;
  Application.Terminate;
end;
```

Az "*Adatok megadása, mentése*" menüpont kiválasztásakor a *KilépésClick* eseménykezelő eljárás hívódik meg, amelyben modálisan megjelenítjük a *Form3* űrlapot.

```
procedure TForm1.AdatokmegadasaClick(Sender: TObject);
begin
  Form3.ShowModal;
end;
```

Az "Betöltés fájlból" menüpont kiválasztásakor a *BetöltésClick* eseménykezelő eljárás lesz aktív, amelyben az *OpenDialog* párbeszédablakból kiválasztjuk a fájl nevét, majd ha volt korábban adat a memóriában, azt töröljük. Az állományból beolvasott adatokat a *Form2* és a *Form3* űrlap a listáiba töltjük, valamint egyidejűleg létrehozuk az objektumláncot.

```

procedure TForm1.BetoltesClick(Sender: TObject);
var
    d : TDiak;
    Inpf : Textfile;
    i, j : integer;
begin
    if Assigned(Evf.elseo) then
        begin
            Evf.EFree;
            Evf := TEvf.ECreate;
            Evf.elseo := nil;
        end;
    OpenFileDialog1.FileName := '*.dat';
    if OpenFileDialog1.Execute then
        begin
            Fnev := OpenFileDialog1.FileName;
            Form1.Caption := 'Adat bettöltése: ' +
                ExtractFileName(OpenDialog1.FileName);
            AssignFile(Inpf, Fnev);
            Reset(Inpf);
            if Assigned(Evf.elseo) then
                begin
                    Evf.EFree;
                    Evf := TEvf.ECreate;
                    Evf.elseo := nil;
                end;
            while not Eof(Inpf) do
                begin
                    d := TDiak.DCreate;
                    Readln(Inpf, d.nev, d.tankor, d.kreditpont);
                    d.elozo := nil;
                    d.kovetkezo := nil;
                    j := 0;
                    for i:=1 to length(d.nev) do
                        if d.nev[i] = ' ' then j:=j+1
                        else break;
                    Delete(d.nev, 1, j);
                    Form3.ListBox1.Items.Add(d.nev);
                    Form2.ListBox1.Items.Add(d.nev);
                    if Evf.elseo = nil
                        then Evf.elseo := d
                        else
                            begin
                                Evf.utolso.kovetkezo := d;
                                d.elozo := Evf.utolso;
                            end;
                    Evf.utolso := d;
                end;
            CloseFile(Inpf);
            Form3.Etankor.Text := IntToStr(Evf.elseo.tankor);
            Form3.Ekredit.Text := IntToStr(Evf.elseo.kreditpont);
            Form2.Etankor.Text := IntToStr(Evf.elseo.tankor);
            Form2.Ekredit.Text := IntToStr(Evf.elseo.kreditpont);
        end;
        Megjelenites.Enabled := true;
        Kiertekeles.Enabled := true;
    end;

```

Az "Adatok megjelenítése" menüpont kiválasztásakor a *MegjelenítésClick* eseménykezelő eljárás aktivizálódik, amelyben a *Form2* űrlap vezérlőit töröljük, majd megjelenítjük a *Form2* űrlapot.

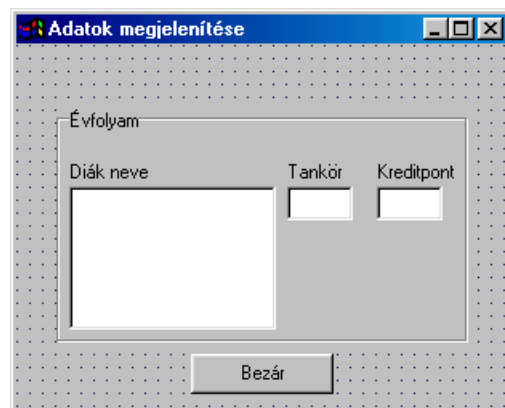
```
procedure TForm1.MegjelenitesClick(Sender: TObject);
begin
    Form2.ListBox1.Clear;
    Form2.Etankor.Clear;
    Form2.Ekredit.Clear;
    Form2.ShowModal;
end;
```

A "Kiértékelés" menüpont kiválasztásakor a *KiértékelésClick* eseménykezelő eljárás lesz aktív, amelyben a *Form4* űrlap vezérlőit töröljük, és megjelenítjük a *Form4* űrlapot.

```
procedure TForm1.KiertekelesClick(Sender: TObject);
begin
    Form4.ListBox1.Clear;
    Form4.nev.Clear;
    Form4.tankor.Clear;
    Form4.kreditpont.Clear;
    Form4.ShowModal;
end;
```

### A *TForm2* osztály deklarációja:

```
type
TForm2 = class(TForm)
    GroupBox1: TGroupBox;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    ListBox1: TListBox;
    Etankor: TEdit;
    Ekredit: TEdit;
    Bezar: TButton;
    procedure BezarClick(Sender: TObject);
    procedure ListBox1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
end;
```



A *Form2* űrlap használja a *DiakAdat* unitot.

```
uses DiakAdat;
```

A "Bezár" nyomógomb megnyomásakor a *BezárClick* eseménykezelő eljárás hívódik meg, amelyben bezárjuk az ablakot.

```
procedure TForm2.BezarClick(Sender: TObject);
begin
    Close;
end;
```

A *ListBox1* lista tételén való kattintáskor az *ListBox1Click* eseménykezelő eljárás lesz aktív, amelyben megjelenítjük a kiválasztott diák tankörszámát és kreditpontját.

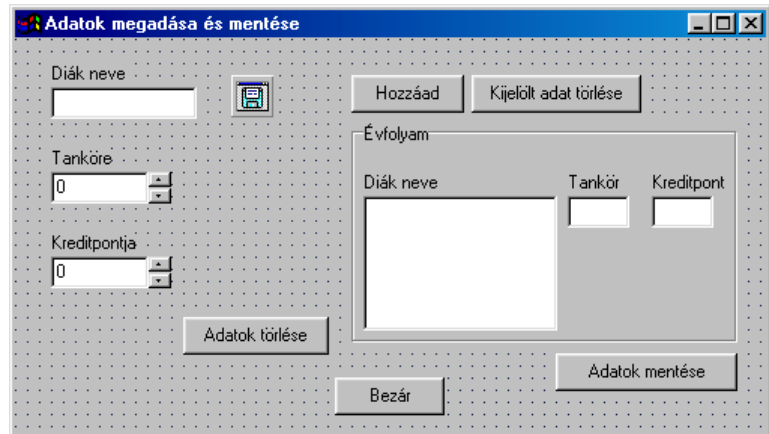
```
procedure TForm2.ListBox1Click(Sender: TObject);
var
  p:TDiak;
begin
  If ListBox1.ItemIndex>=0 then
    begin
      p:= Evf.elso;
      while p <> nil do
        begin
          if ListBox1.Items[ListBox1.ItemIndex]= p.nev then
            begin
              Etankor.Text := IntToStr(p.tankor);
              Ekredit.Text := IntToStr(p.kreditpont);
              break;
            end;
          p:= p.kovetkezo;
        end;
      end;
    end;
end;
```

A *FormCreate* eseménykezelő akkor hívódik meg, amikor a *Form2* űrlap megkapja a fókuszt, így lehetőség van az adatok betöltésére a listába, valamint a kijelző adatmezők törlésére, mielőtt az űrlap megjelenne.

```
procedure TForm2.FormActivate(Sender: TObject);
var
  p : TDiak;
begin
  if Assigned(Evf.elso)
  then
    begin
      p := Evf.elso;
      Etankor.Clear;
      Ekredit.Clear;
      while p <> nil do
        begin
          ListBox1.Items.Add(p.nev);
          p:= p.kovetkezo;
        end;
      end;
    end;
end;
```

### A TForm3 osztály deklarációja:

```
type
TForm3 = class(TForm)
    Label1: TLabel;
    Dnev: TEdit;
    Label2: TLabel;
    Dtankor: TEdit;
    UpDown1: TUpDown;
    Label3: TLabel;
    Dkredit: TEdit;
    UpDown2: TUpDown;
    GroupBox1: TGroupBox;
    ListBox1: TListBox;
    Etankor: TEdit;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Ekredit: TEdit;
    HozzaAd: TButton;
    Bezar: TButton;
    Torles: TButton;
    Adatmentes: TButton;
    SaveDialog1: TSaveDialog;
    AdatokTorlese: TButton;
    procedure HozzaAdClick(Sender: TObject);
    procedure BezarClick(Sender: TObject);
    procedure ListBox1Click(Sender: TObject);
    procedure TorlesClick(Sender: TObject);
    procedure AdatmentesClick(Sender: TObject);
    procedure AdatokTorleseClick(Sender: TObject);
end;
```



A TForm3 űrlap is használja a *DiakAdat* unitot.

```
uses DiakAdat, Diakm;
```

A "Hozzáad" nyomógomb megnyomásakor a *HozzaAdClick* eseménykezelő eljárás aktivizálódik, amelyben az új diák adatait hozzáadjuk a listához és az adatlánchoz.

```
procedure TForm3.HozzaAdClick (Sender: TObject);
var
    d: TDiak;
    t,k:integer;
begin
    t := StrToInt(Dtankor.Text);
    k := StrToInt(Dkredit.Text);
    if (Dnev.Text <> '') and (t > 0) and (k > 0) then
    begin
        ListBox1.Items.Add(Dnev.Text);
        Etankor.Text := Dtankor.Text;
        Ekredit.Text := Dkredit.Text;
        d:= TDiak.DCreate;
        d.nev := Dnev.Text;
        d.tankor := t;
        d.kreditpont:= k;
        Dnev.Clear;
        Dtankor.Clear;
        Dkredit.Clear;
        d.elozo := nil;
        d.kovetkezo := nil;
        if Evf.else = nil then Evf.else := d
        else
            begin
                Evf.utolso.kovetkezo := d;
                d.elozo := Evf.utolso;
            end;
        Evf.utolso := d;
    end;
end;
```

A "Bezár" nyomógomb megnyomásakor a *BezarClick* eseménykezelő eljárás hívódik meg, amelyben ha van már adat, akkor "Adatok megjelenítése" és a "Kiértékelés" menüpontok hozzáférhetővé válnak, majd lezárjuk az ablakot.

```
procedure TForm3.BezarClick(Sender: TObject);
begin
    if Assigned (Evf.else) then
    begin
        Form1.Megjelenites.Enabled := true;
        Form1.Kiertekes.Enabled := true;
    end;
    Close;
end;
```

A *ListBox1* lista tételén való kattintáskor az *ListBox1Click* eseménykezelő eljárás lesz aktív, amelyben megjelenítjük a kiválasztott diák tankörszámát és kreditpontját.

```
procedure TForm3.ListBox1Click(Sender: TObject);
var
    p:TDiak;
begin
    If ListBox1.ItemIndex>=0 then
    begin
        p:= Evf.else;
        while p <> nil do
        begin
            if ListBox1.Items[ListBox1.ItemIndex] = p.nev then
            begin
                Etankor.Text := IntToStr(p.tankor);
                Ekredit.Text := IntToStr(p.kreditpont);
                break;
            end;
            p:= p.kovetkezo;
        end;
    end;
end;
```

A "Kijelölt adat törlése" nyomógomb megnyomásakor a *TorlesClick* eseménykezelő aktivizálódik, amelyben a listából kiválasztott tételt töröljük a listából és az adatláncból is.

```
procedure TForm3.TorlesClick(Sender: TObject);
var p, pAct:TDiak;
begin
    If ListBox1.ItemIndex>-1 then
    begin
        p:= Evf.else;
        while p <> nil do
        begin
            pAct := p;
            Etankor.Text := IntToStr(pAct.tankor);
            Ekredit.Text := IntToStr(pAct.kreditpont);
            if ListBox1.Items[ListBox1.ItemIndex]= pAct.nev then
            begin
                ListBox1.Items.Delete(ListBox1.ItemIndex);
                Etankor.Clear; Ekredit.Clear;
                if pAct = Evf.else then
                begin
                    Evf.else := Evf.else.kovetkezo;
                    if Evf.else <> nil then Evf.else.elozo := nil else Evf.utolso := nil;
                end
            else
            end;
        end;
    end;
```



```

        if pAct = Evf.utolso then
        begin
            pAct := Evf.utolso.elozo;
            pAct.kovetkezo := nil;
            Evf.utolso := pAct;
        end
        else
        begin
            pAct.elozo.kovetkezo := pAct.kovetkezo;
            pAct.kovetkezo.elozo := pAct.elozo;
        end;
        p.Destroy;
        p:= nil;
        break;
    end;
    p:= p.kovetkezo;
end;
end;
end;

```

A "Adatok mentése" nyomógomb megnyomásakor a *AdatmentesClick* eseménykezelő eljárás hívódik meg, amelyben a *SaveDialog1* párbeszédablakban megadott néven szöveges állományba tároljuk az adatokat.

```

procedure TForm3.AdatmentesClick(Sender: TObject);
var Outf: TextFile;
    p : TDiak;
begin
    if Assigned (Evf.elso) then
    begin
        SaveDialog1.FileName := '*.dat';
        if SaveDialog1.Execute then
        begin
            FNev:=SaveDialog1.FileName;
            Form1.Caption := 'Adat mentése: ' +
            ExtractFileName(SaveDialog1.FileName);
            AssignFile(Outf,FNev); Rewrite(Outf);
            p := Evf.elso;
            while p <> nil do
            begin
                Writeln(Outf,p.nev:20,p.tankor:3,p.kreditpont:4);
                p := p.kovetkezo;
            end;
            CloseFile(Outf);
            Form3.Caption:= Form1.Caption;
        end;
    end;
end;
end;

```

A "Adatok törlése" nyomógomb megnyomásakor az *AdatokTorleseClick* eseménykezelő eljárás aktivizálódik, amelyben az összes adatot töröljük a listából és az adatláncból.

```

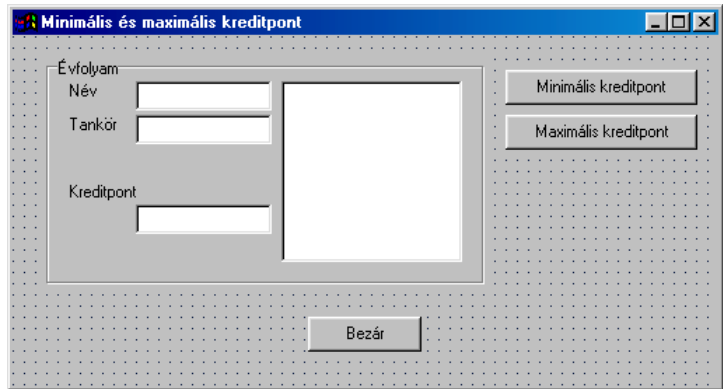
procedure TForm3.AdatokTorleseClick(Sender: TObject);
begin
    if Assigned (Evf.elso) then
    begin
        Evf.EFree;
        Evf:= TEvf.ECreate;
        Evf.elso := nil;
        Form1.Megjelenites.Enabled := false;
        Form1.Kiertekes.Enabled := false;
        Etankor.Clear;
        Ekredit.Clear;
        ListBox1.Clear;
    end;
end;

```

### A TForm4 osztály deklarációja:

type

```
TForm4 = class(TForm)
  GroupBox1: TGroupBox;
  Label1: TLabel;
  nev: TEdit;
  Label2: TLabel;
  tankor: TEdit;
  kreditpont: Tedit;
  ListBox1: TListBox;
  MinKredit: Tbutton;
  MaxKredit: Tbutton;
  Bezar: TButton;
  Label4: TLabel;
  procedure MinKreditClick(Sender: TObject);
  procedure ListBox1Click(Sender: TObject);
  procedure MaxKreditClick(Sender: TObject);
  procedure BezarClick(Sender: TObject);
end;
```



A *Form4* űrlap használja a *DiakAdat* unitot.

```
uses DiakAdat;
```

A "Minimális kreditpont" nyomógomb megnyomásakor a *MinKreditClick* eseménykezelő eljárás lesz aktív, amelyben a *MinKreditpont* metódus által visszaadott érték alapján megkeressük az összes minimumot és betöltjük a listába.

```
procedure TForm4.MinKreditClick(Sender: TObject);
var min: integer; d : TDiak;
begin
  ListBox1.Clear; nev.Clear;
  tankor.Clear; kreditpont.Clear;
  if Assigned(Evf.elseo) then
  begin
    min:=Evf.MinKreditpont;
    d := Evf.elseo;
    while d <> nil do
    begin
      if d.kreditpont = min then ListBox1.Items.Add(d.nev);
      d:= d.kovetkezo;
    end;
    kreditpont.Text := IntToStr(min);
    Label4.Caption := 'Minimális kreditpont';
  end;
end;
```

A listán való kattintáskor végigkeressük a listát, és ha találunk bejegyzést, feltöltjük a vezérlőket.

```
procedure TForm4.ListBox1Click(Sender: TObject);
var p:TDiak;
begin
  If ListBox1.ItemIndex>=0 then
  begin
    p:= Evf.elseo;
    while p <> nil do
    begin
      if ListBox1.Items[ListBox1.ItemIndex]= p.nev then
      begin
        nev.Text := ListBox1.Items[ListBox1.ItemIndex];
        tankor.Text := IntToStr(p.tankor);
        kreditpont.Text := IntToStr(p.kreditpont); break;
      end;
      p:= p.kovetkezo;
    end;
  end;
end;
```

A "Maximális kreditpont" nyomógomb megnyomásakor a *MaxKreditClick* eseménykezelő eljárás lesz aktív, amelyben a *MaxKreditpont* metódus által visszaadott érték alapján megkeressük az összes maximumot és betöltjük a listába.

```
procedure TForm4.MaxKreditClick(Sender: TObject);
var
  max: integer;
  d : TDiak;
begin
  ListBox1.Clear;
  nev.Clear;
  tankor.Clear;
  kreditpont.Clear;
  if Assigned(Evf.elso) then
  begin
    d := Evf.elso;
    max:=Evf.MaxKreditpont;
    while d <> nil do
    begin
      if d.kreditpont = max then
        ListBox1.Items.Add(d.nev);
      d:= d.kovetkezo;
    end;
    kreditpont.Text := IntToStr(max);
    Label4.Caption := 'Maximális kreditpont';
  end;
end;
```

A "Bezár" nyomógomb megnyomásakor a *BezarClick* eseménykezelő eljárás hívódik meg, amelyben bezárjuk az ablakot.

```
procedure TForm4.BezarClick(Sender: TObject);
begin
  Close;
end;
```

A betöltött adatok megjelenítése.

Diák neve	Tankör	Kreditpont
Kiss Anna	1	15
Nagy Bela		
Tot Edit		
Vigh Agi		
Veres Aniko		
Szekeres Viktor		
Bana Balint		

A "Kiértékelés" menüpont kiválasztásakor megjelenő párbeszédablakban megtekinthetjük a minimális és a maximális kreditpontot elért diákok listáját (a "Minimális kredipont" nyomógomb megnyomása után).

The dialog box titled "Minimális és maximális kreditpont" contains the following elements:

- Évfolyam** section:
  - Név**: Text box containing "Kiss Anna".
  - Tankör**: Text box containing "1".
  - Minimális kreditpont**: Text box containing "15".
- Student List**: A list box showing "Kiss Anna", "Tot Edit", and "Balogh Elvira". "Kiss Anna" is selected.
- Buttons**: "Minimális kreditpont", "Maximális kreditpont", and "Bezár".

Az "Adatok megadása és mentése" menüpont kiválasztásakor megjelenő párbeszédablakban új adatokat is megadhatunk, illetve a meglévő adatokat további adattal is kiegészíthetjük.

The dialog box titled "Adatok megadása és mentése" contains the following elements:

- Diák neve**: Text box containing "Komáromi Anna".
- Tanköre**: Spinner box set to "1".
- Kreditpontja**: Spinner box set to "75".
- Buttons**: "Hozzáad", "Kijelölt adat törlése", "Adatok törlése", "Bezár", and "Adatok mentése".
- Évfolyam** section:
  - Diák neve**: List box containing "Kiss Anna", "Nagy Bela", "Tot Edit", "Vigh Ági", "Veres Aniko", "Szekeres Viktor", and "Balogh Elvira". "Kiss Anna" is selected.
  - Tankör**: Text box containing "1".
  - Kreditpont**: Text box containing "15".



Készítsünk programot, amely alkalmas számadatok megadására, az adatok típusos fájlba történő mentésére, illetve állományból való olvasására! Készítsük fel a programot számadatok összegének-, átlagának kiszámítására, a szélsőértékek megkeresésére is! (*RecFile1*)

A program szerkezetében teljesen megfelel a [TextFile1](#) programnak, különbségek csak az fájlkezelésben vannak.

A "Kiírás állományba" menüpont kijelölésekor a *KiirClick* eseménykezelő eljárást hívódik meg, amelyben az adatokat .DAT kiterjesztésű típusos állományba írjuk. A *SaveDialog* párbeszédablakban adjuk meg az állomány nevét.

```
procedure TForm1.KiirClick(Sender: TObject);
var
  f1 : file of real;
  i:integer;
begin
  if x_db > 0 then
  begin
    SaveDialog1.Filter := 'Binary files (*.dat)|*.dat|All files (*.*)|*.*';
    SaveDialog1.FilterIndex := 1;
    if SaveDialog1.Execute then
    begin
      AssignFile(f1,SaveDialog1.FileName);
      Rewrite(f1);
      for i:=1 to x_db do
      begin
        write(f1,x[i]);
      end;
      CloseFile(f1);
    end;
  end;
end;
```

Az "Olvasás állományból" menüpont kijelölésekor az *OlvasClick* eseménykezelő eljárás lesz aktív, amelyben az adatokat .DAT kiterjesztésű típusos állományból olvassuk be.

```
procedure TForm1.OlvasClick(Sender: TObject);
var
  i:integer;
  f1 : file of real;
begin
  OpenFileDialog1.Filter := 'Binary files (*.dat)|*.dat|All files (*.*)|*.*';
  OpenFileDialog1.FilterIndex := 1;
  If OpenFileDialog1.Execute then
  begin
    AssignFile(f2,OpenDialog1.FileName);
    Reset(f2);
    i:=0;
    Form2.Adatok.Clear;
    While not Eof(f2) do
    begin
      i:=i+1;
      Read(f2,x[i]);
      Form2.Adatok.Items[i-1]:=FloatToStr(x[i]);
    end;
    x_db := i;
    CloseFile(f2);
  end;
end;
```



Készítsünk programot házi könyvtárunk katalogizálására! Tároljuk a könyvek szerzőit, címét, a vásárlás idejét és az árát! Az adatok tárolására használjunk típusos fájlt! (*Könyvtar*)

A programból az adatokat típusos állományokba mentjük. A program deklarációit a *GlobalU* modul tartalmazza. Az adatrekord:

```
type
  // A könyv adatait tároló rekord
  TKonyv=record
    sorszam      : integer;
    szerzo       : string[20];
    cim          : string[30];
    beszerezve   : Tdatetime;
    ar           : real;
  end;
```

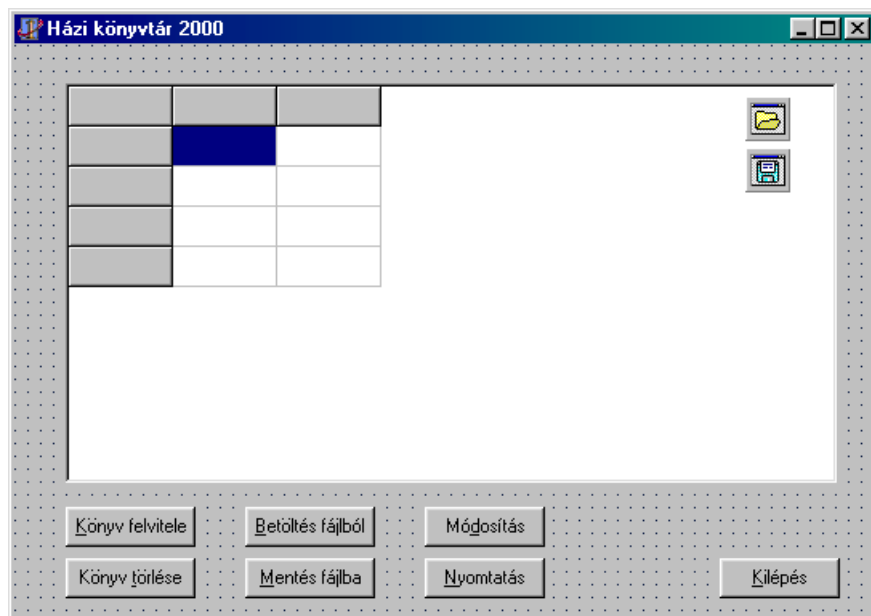
A katalóguskezelő műveletek azonosítására létrehozuk a *TMuvelet* típust.

```
TMuvelet=(Felvitel, Modositas);
```

Szükségünk lesz néhány globális változóra is:

```
var
  Konyv      : TKonyv;
  KonyvOK    : Boolean = false;
  Mentett    : Boolean = true;
  Muvelet    : TMuvelet;
```

A program főablaka az *frmKonyvtar* form.



A könyv felvitelére, törlésére fájlba mentésre, állományból történő olvasására és nyomtatására nyomógombokat használunk.

```
type
  TfrmKonyvtar = class(TForm)
    StringGrid1: TStringGrid;
    Konyvfelvitel: TButton;
    BetoltesFajlbol: TButton;
    Mentefajlba: TButton;
    Nyomtatás: TButton;
    Kilepes: TButton;
    OpenFileDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    Konyvtorles: TButton;
    btnModositas: TButton;
```

```

procedure FormCreate(Sender: TObject);
procedure KonyvfelvitelClick(Sender: TObject);
procedure BetoltesFajlbolClick(Sender: TObject);
procedure KilepesClick(Sender: TObject);
procedure MentefajlbaClick(Sender: TObject);
procedure StringGrid1SetEditText(Sender: TObject; ACol, ARow: Integer;
    const Value: String);
procedure KonyvtorlesClick(Sender: TObject);
procedure NyomtatasClick(Sender: TObject);
procedure btnModositasClick(Sender: TObject);
procedure StringGrid1DrawCell(Sender: TObject; ACol, ARow: Integer;
    Rect: TRect; State: TGridDrawState);
private
    procedure KonyvSorba(const Konyv:TKonyv; rw:integer);
    procedure SorKonyvbe(rw:integer; var Konyv:TKonyv);
public
    procedure UpdateTable;
end;

```

A megjelenítéshez a *StringGrid1* rácsot használjuk, melynek adatait a form létrehozásakor állítjuk be. Létrehozunk egy bejegyzést is, amelyet felviszünk a táblázatba.

```

// A form létrehozásakor kialakítjuk a táblát
procedure TfrmKonyvtar.FormCreate(Sender: TObject);
begin
    with stringgrid1 do
        begin
            fixedcols:=1;
            fixedrows:=1;
            rowcount:=1;
            colcount:=5;
            cells[0,0]:='No';
            cells[1,0]:='Szerző';
            cells[2,0]:='Könyvcím';
            cells[3,0]:='Vásárolva';
            cells[4,0]:='Ár';
            colwidths[0]:=25;
            colwidths[1]:=150;
            colwidths[2]:=150;
            colwidths[3]:=80;
            colwidths[4]:=50;
        end;

        with Konyv do
            begin
                sorszam:=1;
                szerzo:='Borland Int.';
                cim:='Delphi 5 Developers guide';
                beszerezve:=encodedate(2000,02,23);
                ar := 21900;
            end;
            Muvelet:=Felvitel;
            UpdateTable;
end;

```

A *StringGrid1* sorait a *TKonyv* típusú paraméterbe-, és viszont metódusokkal tölthetjük:

```

// A könyv beillesztése a táblába
procedure TfrmKonyvtar.KonyvSorba(const Konyv:TKonyv; rw : integer);
begin
    with Stringgrid1 do
        begin
            GlobalU.Konyv.sorszam:=Konyv.sorszam;
            cells[0,rw]:=inttostr(Konyv.sorszam);
            cells[1,rw]:=Konyv.szerzo;
            cells[2,rw]:=Konyv.cim;
            cells[3,rw]:=dateTostr(Konyv.beszerezve);
            cells[4,rw]:=format('%6.0f Ft', [Konyv.ar]);
        end;
end;

```

```

// A táblázat egy sorának adatait a globális
// konyv változóban tároljuk
procedure TfrmKonyvtar.SorKonyvbe(rw : integer; var Konyv:TKonyv);
var sv:string;
begin
    with Stringgrid1 do
        begin
            Konyv.sorszam:=strToInt(cells[0,rw]);
            Konyv.szerzo:=cells[1,rw];
            Konyv.cim:=cells[2,rw];
            Konyv.beszerezve:=strToDate(cells[3,rw]);
            sv:=cells[4,rw];
            delete(sv,length(sv)-1,2);
            Konyv.ar:=strToFloat(sv);
        end;
    end;

// Könyv hozzáadása a táblához (az utolsó sorba)
procedure TfrmKonyvtar.UpdateTable;
var
    aktsor:integer;
begin
    with stringgrid1 do
        case Muvelet of
            Felvitel:
                begin
                    aktsor:=rowcount;
                    rowcount:=rowcount+1;
                    fixedrows:=1;
                    KonyvSorba(Globalu.Konyv, aktsor);
                end;
            Modositas:
                begin
                    KonyvSorba(Globalu.Konyv, Stringgrid1.row);
                end;
        end;
    end;
end;

```

A *StringGrid1* kirajzolásakor gondoskodunk a kiválasztott cella átszínezéséről:

```

// A kiválasztott és a fókuszált cella átszínezése
procedure TfrmKonyvtar.StringGrid1DrawCell(Sender: TObject; ACol,
    ARow: Integer; Rect: TRect; State: TGridDrawState);
begin
    with Stringgrid1 do
        begin
            if (gdselected in state) or (gdfocused in state) then
                begin
                    canvas.pen.color:=clGreen;
                    canvas.brush.color:=clYellow;
                    canvas.font.color:=clred;
                    canvas.Rectangle(rect.left,rect.top,rect.Right, rect.bottom);
                    canvas.TextOut(Rect.left+2, Rect.top+2,Cells[ACol,ARow]);
                end;
            end;
        end;
    end;
end;

```



A könyvek állományból való betöltése a *BetoltesFajlbol* gomb megnyomásakor történik. A fájl nevét általános párbeszédablakkal választjuk ki, és a (*TKonyv* típusú) LIB állományokból a *KonyvSorba* metódussal töltjük. A *Mentett* változó igaz értéke azt jelzi, hogy táblázatunk és a fájl azonos információt hordoznak. A hibákat kivételekkel kezeljük.

```
// Könyvek betöltése LIB állományból
procedure TfrmKonyvtar.BetoltesFajlbolClick(Sender: TObject);
var
  f      : file of TKonyv;
  Konyv  : TKonyv;
  i      : integer;
begin
  if Opendialog1.Execute then
    begin
      try
        assignfile(f, Opendialog1.FileName);
        reset(f);
        Stringgrid1.RowCount:=Filesize(f)+1;
        Stringgrid1.FixedRows:=1;
        for i:=1 to Stringgrid1.RowCount-1 do
          begin
            read(f,Konyv);
            KonyvSorba(Konyv,i);
          end;
        Stringgrid1.Row:=1;
        Stringgrid1.Col:=1;
        closefile(f);
        Mentett:=true;
      except
        ShowMessage('Állományolvasási hiba!')
      end;
    end;
end;
```

A könyvek állományba való írása a *MentesFajlba* gomb megnyomásakor történik. A fájl nevét általános párbeszédablakkal választjuk ki, és a (*TKonyv* típusú) LIB állományba a *write* eljárás segítségével írjuk. A hibákat kivételekkel kezeljük.

```
// A táblázatban tárolt könyveket LIB fájlba mentjük
procedure TfrmKonyvtar.MentesfajlbaClick(Sender: TObject);
var
  f      : file of TKonyv;
  Konyv  : TKonyv;
  i      : integer;
begin
  if Savedialog1.Execute then
    begin
      try
        assignfile(f, Savedialog1.FileName);
        rewrite(f);
        try
          for i:=1 to Stringgrid1.RowCount-1 do
            begin
              SorKonyvbe(i,Konyv);
              write(f,Konyv);
            end;
          finally
            closefile(f);
          end;
        except
          ShowMessage('File write error!')
        end;
      end;
    end;
end;
```

Ha adatot változtatunk, akkor a *Mentett* változót hamisra állítjuk

```
// Ha valamit megváltoztatunk a táblázatban, szükséges lesz
// a mentés
procedure TfrmKonyvtar.StringGrid1SetEditText(Sender: TObject; ACol,
  ARow: Integer; const Value: String);
begin
  Mentett:=false;
end;
```

Kilépéskor is mentjük adatainkat, ha szükséges.

```
// Kilépés, ha szüksége felajánlja a mentést
procedure TfrmKonyvtar.KilepesClick(Sender: TObject);
begin
  if not Mentett then MentésFajlbaClick(Sender);
  frmKonyvtar.close;
end;
```

A könyvek törlését a táblázaton végezzük.

```
// Könyv törlése
procedure TfrmKonyvtar.KonyvtorlesClick(Sender: TObject);
var i:integer;
begin
  if stringgrid1.row=0 then
    begin
      beep;
      exit;
    end;
  with stringgrid1 do
    for i:=row to rowcount-1 do
      rows[i]:=rows[i+1];
  stringgrid1.rowcount:=stringgrid1.rowcount-1;
  if stringgrid1.row=0 then Konyv.sorszam:=0;
end;
```

A nyomtatás az *AssignPrn* metódussal történik a cellák tartalma alapján.

```
// A táblázat egyszerű eszközzel történő nyomtatása
procedure TfrmKonyvtar.NyomtatásClick(Sender: TObject);
var f : TextFile;
    i,j: integer;
begin
  AssignPrn(f);
  Rewrite(f);
  with Stringgrid1 do
    for i:=0 to RowCount-1 do
      begin
        writeln(f);
        writeln(f);
        for j:=0 to 4 do
          write(f,cells[j,i],#9);
        end;
      closefile(f);
end;
```

Az új könyvek felvitelét és a könyvbejegyzések módosítását egyaránt az *frmKonyv* párbeszédablakban végezzük.

```
// Új könyv felvitele
procedure TfrmKonyvtar.KonyvfelvitelClick(Sender: TObject);
begin
  Muvelet:=Felvitel;
  inc(Konyv.sorszam);
  Konyv.szerzo:='';
  Konyv.cim:='';
  Konyv.beszerezve:=date;
  Konyv.ar:=0;
  frmKonyv.showmodal;
end;
```

```
// A Könyvbejegyzés módosítása
procedure TfrmKonyvtar.btnModositasClick(Sender: TObject);
begin
    if Stringgrid1.row=0 then exit;
    Muvelet:=Modositas;
    SorKonyvbe(stringgrid1.row, Globalu.Konyv);
    frmKonyv.showmodal;
end;
```

A form aktiválásakor töltjük be az adatokat.

```
// Amikor az ablak aktívává válik, a szövegmezők tartalmát
// feltöltjük a globális Könyv rekordból
procedure TfrmKonyv.FormActivate(Sender: TObject);
begin
    edAuthor.setfocus;
    edIdent.text:=inttostr(Konyv.sorszam);
    edAuthor.text:=Konyv.szerzo;
    edTitle.text:=Konyv.cim;
    edObtained.text:=dateTostr(Konyv.beszerezve);
    edPrice.text:=floatTostr(Konyv.ar);
end;
```

Az OK és a Mégse gombokkal a beállított adatok mentése, illetve elvetése történik.

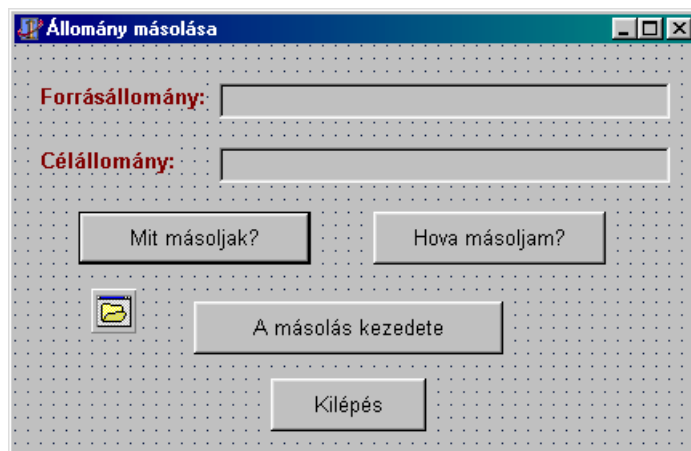
```
// Kilépése az OK megnyomásával
procedure TfrmKonyv.btnOKClick(Sender: TObject);
begin
    // Az adatok áttöltése a globális Könyv rekordba
    Konyv.szerzo:=edAuthor.text;
    Konyv.cim:=edTitle.text;
    Konyv.beszerezve:=strToDate(edObtained.text);
    Konyv.ar:=strToFloat(edPrice.text);
    KonyvOK:=true;
    // A könyv felvitele a táblázatba
    frmKonyvtar.UpdateTable;
    frmKonyv.close;
end;
```

```
// A bevitt adatok elvetése
procedure TfrmKonyv.btnMegseClick(Sender:
    TObject);
begin
    KonyvOK:=false;
    frmKonyv.close;
end;
```

No	Szerző	Könyvcím	Vásárolva	Ár
1	Borland Int.	Delphi 5 Developers guide	2000.02.23.	21900 Ft



Tervezzük meg az alkalmazás felhasználói felületét!



A „Mit másoljak?” gomb megnyomásakor az általános fájlnyitás párbeszédablakkal kiválasztjuk a másolandó állományt (az állomány neve az *Edit1* vezérlőbe kerül).

```
procedure TForm1.btnMitClick(Sender: TObject);
begin
    // Szűrő beállítása *.*
    OpenFileDialog1.Filter := 'Minden fájl (*.*)|*.*|';
    OpenFileDialog1.FilterIndex := 1;
    OpenFileDialog1.Title := 'A forrásállomány kiválasztása';
    // Az állomány nevének kiválasztása után a fájl nevével
    // tárolása a formon
    Edit1.Text := '';
    if OpenFileDialog1.Execute then
        Edit1.Text := OpenFileDialog1.FileName;
end;
```

A „Hova másoljam?” gombbal a másolás célját választhatjuk ki, ami az *Edit2* vezérlőbe is belekerül.

```
procedure TForm1.btnHovaClick(Sender: TObject);
begin
    // Szűrő beállítása *.*
    OpenFileDialog1.Filter := 'Minden fájl (*.*)|*.*|';
    OpenFileDialog1.FilterIndex := 1;
    OpenFileDialog1.Title := 'A célállomány kiválasztása';
    // Az állomány nevének kiválasztása után a fájl nevével
    // tárolása a formon
    Edit2.Text := '';
    if OpenFileDialog1.Execute then
        Edit2.Text := OpenFileDialog1.FileName;
end;
```

Az állományok nevét természetesen a szöveges vezérlőkben begépeléssel is meg lehet adni.

A másolás a „*Másolás kezdete*” gombbal indítható. Gondoskodunk arról is, hogy a DOS rendszerben megszokott módon célnak elegendő legyen csak egy könyvtárat választani, ilyenkor a másolt állomány neve nem változik. Az esetleges hibákat kivételeken keresztül kezeljük.

```

procedure TForm1.btnMasolasClick(Sender: TObject);
var

    Fnev : string;
    i     : integer;
begin
    if (Edit1.Text<>'' ) and (Edit2.Text<>'' ) then
        begin

            // Ha a cél útvonal, az állománynév hozzáillesztése
            for i:=length(Edit1.Text) downto 1 do
                if Edit1.Text[i]='\ ' then Break;
            if Edit1.Text[i]='\ ' then
                Fnev:=copy(Edit1.Text, i+1, length(Edit1.Text)-i);
            if Edit2.Text[length(Edit2.Text)]='\ ' then
                Edit2.Text:=Edit2.Text+Fnev;
            try
                CopyIt(Edit1.Text, Edit2.Text);
            except
                MessageDlg('Hiba lépett fel a másolás során!', mtError, [mbOK],0);
                Edit1.Setfocus;
                Exit;
            end;
            MessageDlg('A másolás megtörtént!', mtInformation, [mbOK],0);
            Edit1.Text:='';
            Edit2.Text:='';
            btnMit.Setfocus;
        end
    else
        begin
            MessageBeep(0);
            MessageDlg('Rossz, vagy hiányzó paraméterek!', mtError, [mbOK],0);
        end;
    end;

```

A másolás a *CopyIt* eljárással, 1 bájtos blokkokkal történik. Az adatok tárolására 100 kBájtos puffert használunk.

```

procedure CopyIt( const FromName, ToName: string );
const
    pmeret=102400; // 100 kilobájt
var
    ff, tf : file;
    puffer : array [1..pmeret] of byte;
    nOlvasott, nKiirt: integer;
begin
    {$I+}
    AssignFile(ff, FromName);
    FileMode:=0; // A forrásfájlt csak olvassuk
    Reset(ff,1);
    try
        AssignFile(tf, ToName);
        Rewrite(tf,1);
        try
            repeat
                BlockRead(ff,puffer,Sizeof(puffer),nOlvasott);
                BlockWrite(tf,puffer,nOlvasott,nKiirt);
            until (nOlvasott = 0) or (nKiirt <> nOlvasott);
        finally
            CloseFile(tf);
        end;
    finally
        CloseFile(ff);
    end;
end;

```