


#### 4. Többablakos alkalmazások kialakítása

1. Párbeszédablak tervezésének menete: [dialog1](#)
2. Egyszerű párbeszédablakok: [Dialog\\_minta](#)
3. Búvós négyzet megjelenítése jelszómegadással: [Buvos](#)
4. Színes téglalap rajzolása: [Dialog\\_color](#)
5. Szöveg betűkészletének változtatása: [Dialog\\_font](#)
6. Bitkép állományból való betöltése, megjelenítése és mentése: [Dialog\\_picture](#)
7. Bitkép nyomtatása: [Dialog\\_print](#)
8. Képek váltogatása **TabControl** vezérlővel [Regiszterful](#)
9. Képek váltogatása **PageControl** vezérlővel [Regiszter](#)
10. SDI alkalmazás **Memo** komponenssel: [Sdi\\_1](#)
11. SDI alkalmazás **RichEdit** komponenssel, keresés és helyettesítés menüpontokkal: [Dialog\\_find](#)
12. Többlapos MDI alkalmazás készítése [MDIApp](#)
13. MDI-alkalmazás az „Application Wizard” varázslóval [MDI\\_Auto](#)

---

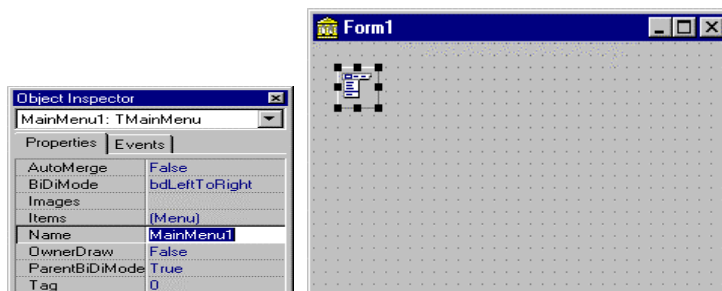
 Tervezzünk olyan menüvezérelt programot, amely bemutatja a modális és a nem modális megjelenítés közötti különbséget! (*dialog!*)

---

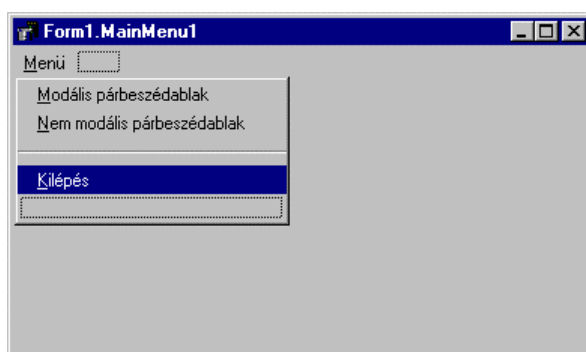
A program menüje legyen:

Menü  
Modális párbeszédablak  
Nem modális párbeszédablak  
Kilépés

Tegyük fel a *Form1* űrlapra a **MainMenu** komponenst, miáltal létrejön a **TMainMenu** típusú *MainMenu1* objektum.



Ezt követően megtervezzük a menüpontokat:

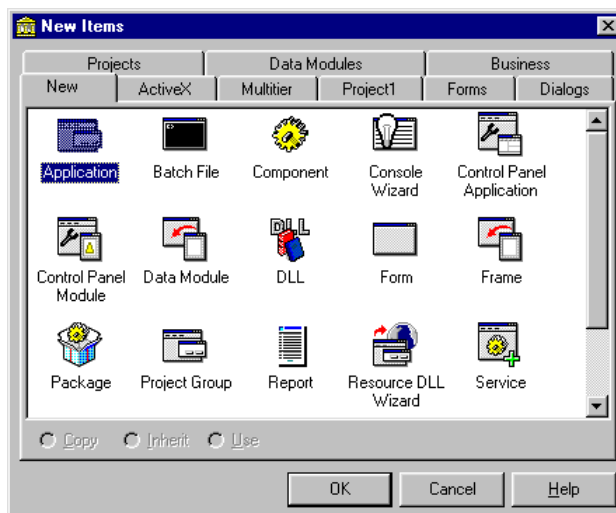


Legyen az ablak fejléce a "*Párbeszédablak típusai*", melyet a *Form1* **Caption** tulajdonságnál adunk meg.

Tároljuk el a főablak modulját *DIALOGIM* néven!


További két formra van szükségünk, hogy a modális és a nem modális párbeszédablakokat megjelenítsük a megfelelő menüpont kiválasztásakor. Az újabb formot kétféleképpen is létrehozhatjuk:

1. A **File|New...** menüpont kiválasztásával megjelenő **New Items** párbeszédablakban a *Form* ikonon kattintva.



2. Kiválaszthatjuk még a **File|New Form** menüpontot, melynek hatására a form a következő sorszámmal jön létre. Esetünkben a *Form2* jelenik meg a kódszerkesztő ablakban

Helyezzünk a formra egy nyomógomb komponenst! A „**View|Alignment palette**” menüpont kiválasztásakor megjelenik az **Align** ablak, amely megkönnyíti a komponensek rendezett elhelyezését az űrlapon. Ha az **Align** ablak "Center horizontally in window" nyomógombján kattintunk, a kijelölt *Button1* gomb vízszintesen, az ablak közepre kerül. A *Button1* komponens **Caption** tulajdonságát *Bezár*-ra állítjuk, a **Name** pedig *Bezar* lesz. Tegyük a formra egy **Memo** vezérlőt, amely többsoros szöveg tárolására használható!

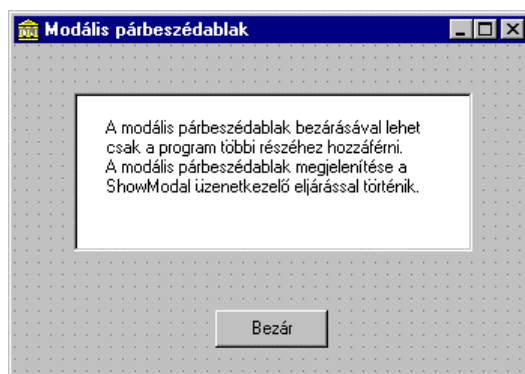
Állítsuk be a form **BorderStyle** tulajdonságát *bsDialog* értékre, amely a párbeszédablak fejlécében csak az ablaklezáró  gombot jeleníti meg, és nem engedi meg az ablak átméretezését, valamint az ikonná való lekicsinyítést sem. A **Caption** tulajdonság mellett megadjuk a párbeszédablak fejlécét (*Modális párbeszédablak*).

Jelöljük ki a *Memo1* vezérlőt és kattintsuk a **Lines** tulajdonságán. A megjelenő „**String list editor**” ablakban megadjuk a modális párbeszédablakra vonatkozó információkat, majd az **OK** gombon kattintva kilépünk a szerkesztőből.

Ezt követően beállítjuk a *Form2* űrlap a bal (**Left**) felső (**Top**) sarkának koordinátáit, valamint az ablak szélességét (**Width**) és magasságát (**Height**) az alábbiak szerint:

<b>Left</b>	250
<b>Top</b>	160
<b>Height</b>	246
<b>Width</b>	350

A megtervezett párbeszédablak:

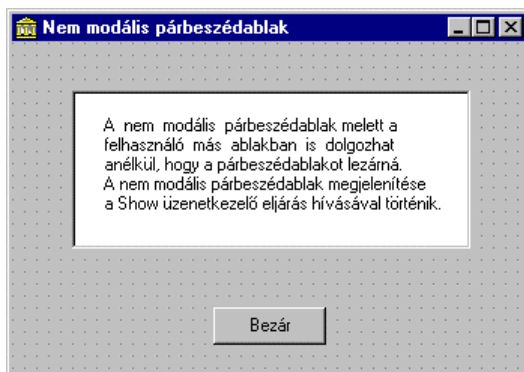


Ezután a **File|Save As...** menüpont kiválasztásával tároljuk a *Unit2* forrásállományt *DIALOG1.DI* néven.

Hozzunk létre egy új formot a második párbeszédablak számára a fent ismertetett módok egyikével. Az új form automatikusan a *Form3* néven jön létre.

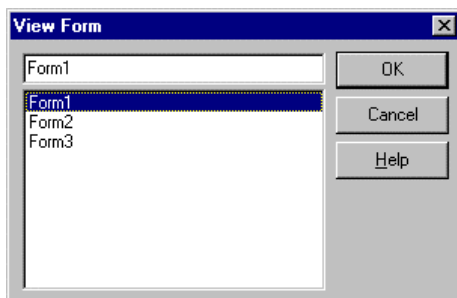
Állítsuk a *Form3* űrlap **FormStyle** tulajdonságát *fsStayTop*-ra, amely a párbeszédablakot mindig a főablak előtt tartja. Ennek a párbeszédablaknak a tervezése hasonló előzőekben ismertetetthez, csak a fejléc és a *Memo1* vezérlő szövege változik, a különbség csak az ablak megjelenítési módjában lesz.

Mindkét párbeszédablakban a *Memo1* vezérlőt csak olvashatóra állítjuk (**ReadOnly** ← *true*).



Tároljuk el a "Nem modális párbeszédablak"-ot *DIALOG1D2* néven!

A **View|Forms** menüpont kiválasztásával megjelenő párbeszédablakban kijelöljük a *Form1* a főablakot.



Az ablakok lezárása után megadhatjuk a főablak helyét és méretét:

A *Form1* ablakon kattintva a *FormCreate* eseménykezelő eljárásban beállítjuk a bal felső sarok koordinátáit és az ablak méreteit.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    // az ablak bal sarkának koordinátái
    Left  := 190;
    Top   := 110;
    // az ablak magassága, szélessége
    Height := 190;
    Width  := 280;
end;
```

A *Form2* ablak helyét és méreteit szintén adott értékekkel definiáljuk, és a *Memol* vezérlőt csak olvashatóra állítjuk, mivel csak kijelzésre használjuk:

```
procedure TForm2.FormCreate(Sender: TObject);
begin
    // a Memol vezérlő csak olvasható
    Memo1.ReadOnly := true;
    // az ablak bal sarkának koordinátái
    Left := 250;
    Top := 160;
    // az ablak magassága, szélessége
    Height := 246;
    Width := 350;
end;
```

A *Form2* űrlap "Bezár" gombján való kattintás hatására megjelenő *BezarClick* eseménykezelő függvényben a *Close* tagfüggvény hívásával bezárjuk a "Modális párbeszédablak" ablakot.

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

A fentiekhez hasonlóan megadjuk a *Form3* ablak pozícióját és méreteit is. A *Memol* vezérlő itt is csak olvasható:

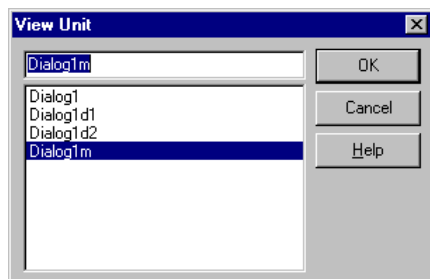
```
procedure TForm3.FormCreate(Sender: TObject);
begin
    // a Memol vezérlő csak olvasható
    Memo1.ReadOnly := true;
    // az ablak bal sarkának koordinátái
    Left := 250;
    Top := 160;
    // az ablak magassága, szélessége
    Height := 246;
    Width := 350;
end;
```

Kattintva az egérrel a "Bezár" gombon, a *BezarClick* eseménykezelő függvény hívódik meg, amelyben bezárjuk a nem modális párbeszédablakot:

```
procedure TForm3.BeazarClick(Sender: TObject);
begin
    Close;
end;
```

A projektet *DIALOG1*-ként tárolva, a módosítások elvégzése után válasszuk ki „*File|Save All*” menüpontot a módosítások tárolására!

Az eseményvezérelt kód írásához betöltjük a programmodulokat, és a *View|Units* menüpont kiválasztásával megjelenő párbeszédablakból kijelöljük a főprogramot:




A főprogram **uses** részében láthatjuk, hogy melyik formhoz melyik .PAS állomány tartozik.

Először a *Form1* "Kilépés" menüpontjához tartozó kódot írjuk meg.

Betöltjük a *Form1* űrlapot, kattintunk a **MainMenu** komponensen, majd a "Kilépés" menüponton. A *KilepesClick* eseménykezelő eljárásba egyetlen utasítást kell írunk: Az *Application* objektum **Terminate** metódusának hívásával a program befejezi a működését:

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

Ezután kattintunk a (**Toggle Form/Unit**)  ikonon, mert ezzel a legkönnyebben tudjuk a formot a kóddal váltani. Ezt követően a "Menü" főmenü "Modális párbeszédablak " menüpontján kattintunk. Az *ModalisClick* eseménykezelő eljárásban a *Form2* ablakot modális párbeszédablakként jelenítjük meg a **ShowModal** metódus hívásával.

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

Ezt követően újra megjelenítjük a *Form1* űrlapot és kattintunk a "Nem modális párbeszédablak" menüponton. A *NemModalisClick* eseménykezelő eljárásban a *Form3* ablakot nem-modális párbeszédablakként jelenítjük meg.

```
procedure TForm1.NemModalisClick(Sender: TObject);
begin
    Form3.Show;
end;
```

Az eddigi munkánkat ellenőrizhetjük, ha először elmentjük a változtatásokat (**File|Save All**), majd megkísérlünk egy futtatást (**Run|Run**).

Két figyelmeztetést kapunk, hogy a *Form2* és a *Form3* modulokat nem ismeri fel a *Form1*. A *Yes* gomb megnyomásának hatására a modulok elérésére a **uses** mellett automatikusan megtörténik a *Dialog1d1* és a *Dialog1d2* modulokra való hivatkozás:

```
implementation
uses Dialog1d1, Dialog1d2;
```

A *TForm1* osztály deklarációja:

```
type
    TForm1 = class(TForm)
        MainMenu: TMainMenu;
        Menu: TMenuItem;
        Modalis: TMenuItem;
        NemModalis: TMenuItem;
        N1: TMenuItem;
        Kilepes: TMenuItem;
        procedure FormCreate(Sender: TObject);
        procedure KilepesClick(Sender: TObject);
        procedure ModalisClick(Sender: TObject);
        procedure NemModalisClick(Sender: TObject);
    end;
```

*A TForm2 osztály deklarációja:*

```
type
  TForm2 = class(TForm)
    Bezar: TButton;
    Memo1: TMemo;
    procedure FormCreate(Sender: TObject);
    procedure BezarClick(Sender: TObject);
  end;
```

*A TForm3 osztály deklarációja:*

```
type
  TForm3 = class(TForm)
    Memo1: TMemo;
    Bezar: TButton;
    procedure FormCreate(Sender: TObject);
    procedure BezarClick(Sender: TObject);
  end;
```

A Dialog1.DPR az alkalmazás főprogramja:

```
program Dialog1pr;

uses
  Forms,
  Dialog1m in 'Dialog1m.pas' {Form1},
  Dialog1d1 in 'Dialog1d1.pas' {Form2},
  Dialog1d2 in 'Dialog1d2.pas' {Form3};

{$R *.RES}

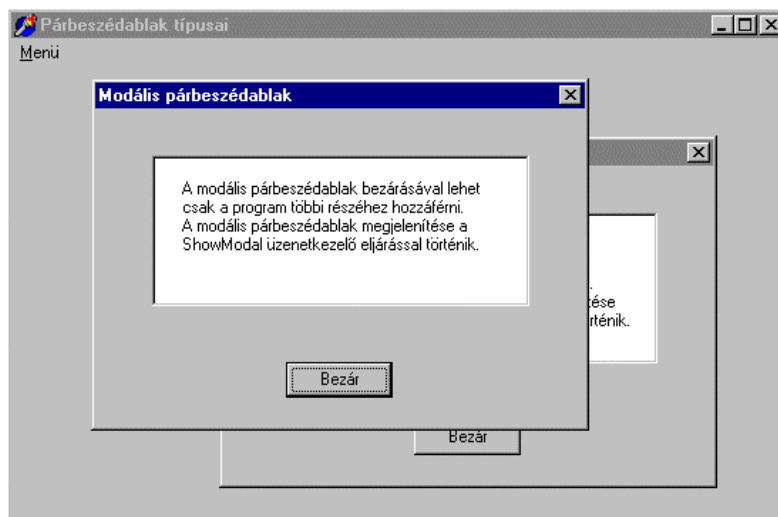
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.Run;
end.
```

A sikeres indítás után megjelenik a program főmenüje.

A "Menü|Modális párbeszédablak" menüpont kiválasztásakor, illetve az <Alt+M M> billentyűk leütésekor a "Modális párbeszédablak" jelenik meg.


A "Menü|Nem modális párbeszédablak" menüpont kiválasztásakor, illetve az <Alt+M N> billentyűk leütésekor a "Nem modális párbeszédablak" jelenik meg.

A nem modális párbeszédablak mellett kiválaszthatjuk újra a "Modális párbeszédablak" menüpontot. Ez követően csak a "Modális párbeszédablak" bezárása után lehet a "Nem modális párbeszédablak" bezárni.

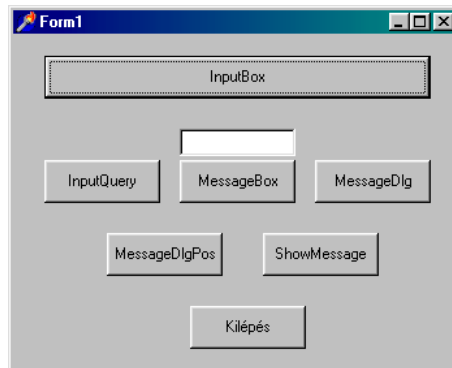


A "Menü| Kilépés" menüpont kiválasztásával fejezzük be a program futását és a fejlesztői környezet „File|Close All” menüpontjának kiválasztásával lezárjuk a DIALOG1 alkalmazás állományait.



 Tervezzünk olyan programot, amelyben a **Dialogs** unit eljárásait használjuk egyszerű párbeszédablakok megjelenítéséhez! (*Dialog\_minta*)

A *Form1* űrlapon nyomógombok segítségével jelenítsük meg a Delphi **Dialogs** unitjában definiált párbeszédablakokat!

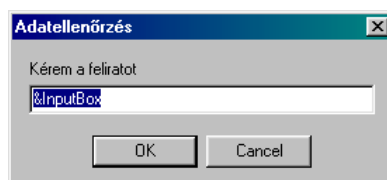


A táblázat megmutatja, hogy a nyomógombok fejlécéhez milyen párbeszédablak jelenik meg.

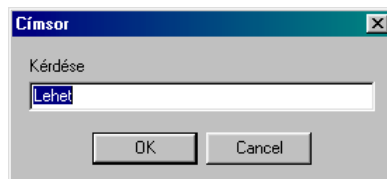
#### *A nyomógomb fejléce*

*InputBox*

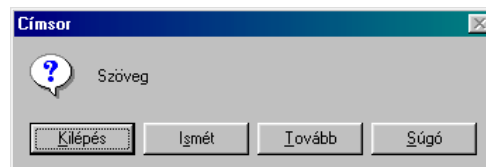
#### *A megjelenő párbeszédablak*



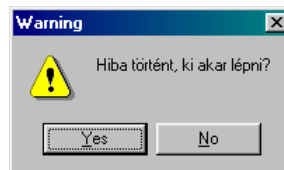
*InputQuery*



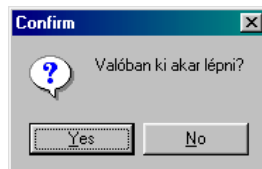
*MessageBox*



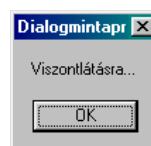
*MessageDlg*



*MessageDlgPos*



*ShowMessage*



*A TForm1 osztály deklarációja:*

```
type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Edit1: TEdit;
    Button6: TButton;
    Kilepes: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure KilepesClick(Sender: TObject);
  end;
```

Az *InputBox* nyomógomb megnyomásakor meghívódó *Button1Click* eseménykezelő eljárás az *InputBox* előre definiált párbeszédablakot aktiválja.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  nev : string;
begin
  {function InputBox(const ACaption, APrompt, ADefault:string):string;}
  nev :=InputBox('Adatellenőrzés','Kérem a feliratot','&InputBox');
  Button1.Caption := nev;
end;
```

Az *InputQuery* nyomógomb megnyomásakor meghívódó *Button2Click* eseménykezelő eljárás az *InputQuery* előre definiált párbeszédablakot aktiválja.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  Clok : boolean;
  nev : string;
begin
  {
    function InputQuery(const ACaption,Aprompt:string;
      var Value: string):string;
  }
  nev := 'Lehet';
  Clok:= InputQuery('Címsor','Kérdése', nev);
  if Clok then Button2.Caption := '&'+nev;
end;
```

A *MessageBox* nyomógomb megnyomásakor meghívódó *Button3Click* eseménykezelő eljárás az *MessageBox* előre definiált párbeszédablakot aktiválja.

```

procedure TForm1.Button3Click(Sender: TObject);
begin
{
  MB_ABORTRETRYIGNORE Megszakítás/Ismét/Tovább,
  MB_OK OK,
  MB_OKCANCEL OK/Mégse,
  MB_RETRYCANCEL Ismét/Mégse,
  MB_YESNO Igen/Nem,
  MB_YESNOCANCEL Igen/Nem/Mégse,
  or MB_DEFBUTTON1, 2, 3, 4 -melyik gomb az aktuális,
  or MB_HELP Súly,
  or MB_ICONINFORMATION I ikon,
  or MB_ICONERROR X ikon,
  or MB_ICONEXCLAMATION ! ikon,
  or MB_ICONQUESTION ? ikon
}
  case Application.MessageBox('Szöveg','Címsor',
    MB_ABORTRETRYIGNORE or MB_ICONQUESTION or MB_HELP) of
    IDOK      : edit1.Text := '1 - OK ';
    IDCANCEL  : edit1.Text := '2 - Mégse ';
    IDABORT   : edit1.Text := '3 - Megszakítás ';
    IDRETRY   : edit1.Text := '4 - Ismét ';
    IDIGNORE  : edit1.Text := '5 - Tovább ';
    IDYES     : edit1.Text := '6 - Igen ';
    IDNO      : edit1.Text := '7 - Nem ';
  end;
end;

```

A *MessageDlg* nyomógomb megnyomásakor meghívódó *Button4Click* eseménykezelő eljárás az *MessageDlg* előre definiált párbeszédablakot aktiválja.

```

procedure TForm1.Button4Click(Sender: TObject);
begin
{
  TMsgBlgBtn = (mbYes, mbNo, mbOk, mbCancel, mcAbort, mbRetry,
    mbIgnore, mbAll, mbHelp);
  TMsgDlgButtons = set of TMsgDlgBtn;

  TMsgDlgType = (mtWarning, mtError, mtInformation,
    mtConfiguration, mtCustom);
  function MessageDlg(const Msg: string; Atype: TMsgDlgType;
    AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
    visszatérési értékek: mrNone, mrAbort, mrYes, mrOk, mrRetry, mrNo,
    mrCancel, mrIgnore, mrAll
  }
  if MessageDlg('Hiba történt, ki akar lépni?', mtWarning,
    [mbYes, mbNo], 0) = mrYes then
  begin
    MessageDlg('Kilépés most!', mtInformation, [mbOk], 0);
    Close;
  end;
end;

```

A *MessageDlgPos* nyomógomb megnyomásakor meghívódó *Button5Click* eseménykezelő eljárás az *MessageDlgPos* előre definiált párbeszédablakot aktiválja.

```
procedure TForm1.Button5Click(Sender: TObject);
begin
{
    function MessageDlgPos(const Msg: string; AType: TMsgDlgType;
        AButtons: TMsgDlgButtons; HelpCtx: Longint;
        X, Y: Integer): Word;

}
    if MessageDlgPos('Valóban ki akar lépni?',
        mtConfirmation, [mbYes, mbNo], 0, 100, 100)
        = mrYes then
    begin
        MessageDlgPos('Kilépés most!', mtInformation, [mbOk], 0, 400, 200);
        Close;
    end;
end;
```

A *ShowMessage* nyomógomb megnyomásakor meghívódó *Button6Click* eseménykezelő eljárás az *ShowMessage* előre definiált párbeszédablakot aktiválja.

```
procedure TForm1.Button6Click(Sender: TObject);
begin
{
    procedure ShowMessage(const Msg: string);
}
    ShowMessage('Visszontlátásra...');
end;
```

A *Kilepes* nyomógomb megnyomásakor meghívódó *KilepesClick* eseménykezelő eljárás befejezi a program működését.

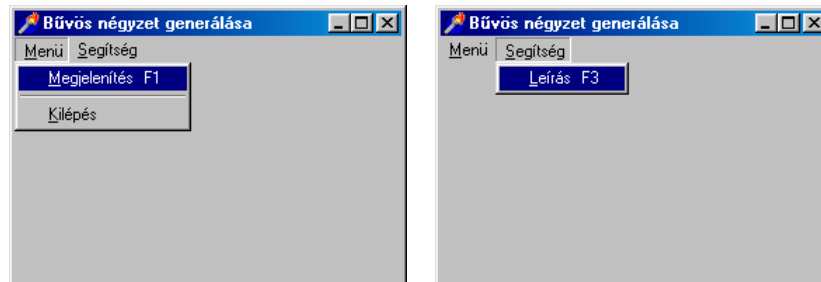
```
procedure TForm1.KilepesClick(Sender: TObject);
begin
    Application.Terminate;
end;
```



Titkos kódzó beolvasása után jelenítsünk meg egy 3x3-as bűvös négyzetet! A kódzó beolvasására használjuk az előre definiált „*Password Dialog*” párbeszédablakot! (*Buvos*)

A program 3x3-as bűvös négyzetet jelenít meg, ha megfelelő a jelszót kapta. Azt a négyzetet nevezzük bűvös négyzetnek, mely sorainak, oszlopainak és átlóinak összege azonos. A bűvös négyzetet az *Agrippa* módszer segítségével generálhatunk.

A program menürendszer:



A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    Menu: TMenuItem;
    Megjelenites: TMenuItem;
    N1: TMenuItem;
    Kilepes: TMenuItem;
    Segitsag: TMenuItem;
    Leiras: TMenuItem;
    procedure FormCreate(Sender: TObject);
    procedure KilepesClick(Sender: TObject);
    procedure MegjelenitesClick(Sender: TObject);
    procedure LeirasClick(Sender: TObject);
  end;
```

A *FormCreate* eseménykezelő eljárásban a *jelszo\_van* változót *false* értékre állítjuk.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Jelszo_van := false;
end;
```

A "Menü/Kilépés" menüpont kiválasztásakor meghívódó *KilepesClick* eseménykezelő eljárás befejezi a program futását.

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

A "Menü/Megjelenítés" menüpont kiválasztásakor, a *Jelszo\_van false* értéke esetén, meghívódik a *PassWordDlg* párbeszédablak. A helyes jelszó megadása után az *Agrippa* módszer kerül meghívásra a *meret* paraméterrel, majd a mátrix értékei beírásra kerülnek az *OKBottomDlg1* párbeszédablak *Panel1..Panel9* vezérlőinek *Caption* tulajdonságába. Ezek után megjelenítjük az *OKBottomDlg1* párbeszédablakot.

```

procedure TForm1.MegjelenitesClick(Sender: TObject);
var i,j,k: integer;
    vissza : boolean;
begin
    meret := 3;
    if not Jelszo_van then
        PasswordDlg.ShowModal;
    if Jelszo_van then
        begin
            repeat
                vissza:= Agrippa(meret);
            until vissza;
            OKBottomDlg1.Caption := '3x3-as méretű bűvös négyzet';
            k:= 0;
            for i:=1 to meret do
                for j:=1 to meret do
                    begin
                        k:= k+1;
                        case k of
                            1: OKBottomDlg1.Panel1.Caption := IntToStr(matrix[i+j*meret]);
                            2: OKBottomDlg1.Panel2.Caption := IntToStr(matrix[i+j*meret]);
                            3: OKBottomDlg1.Panel3.Caption := IntToStr(matrix[i+j*meret]);
                            4: OKBottomDlg1.Panel4.Caption := IntToStr(matrix[i+j*meret]);
                            5: OKBottomDlg1.Panel5.Caption := IntToStr(matrix[i+j*meret]);
                            6: OKBottomDlg1.Panel6.Caption := IntToStr(matrix[i+j*meret]);
                            7: OKBottomDlg1.Panel7.Caption := IntToStr(matrix[i+j*meret]);
                            8: OKBottomDlg1.Panel8.Caption := IntToStr(matrix[i+j*meret]);
                            9: OKBottomDlg1.Panel9.Caption := IntToStr(matrix[i+j*meret]);
                        end;
                    end;
                OKBottomDlg1.ShowModal;
            end;
        end;

```

A *Segítség/Leírás* menüpont kiválasztásakor a *LeirasClick* eseménykezelő eljárás hívódik meg, amely az *OKHelpBottomDlg* párbeszédablakot jeleníti meg.

```

procedure TForm1.LeirasClick(Sender: TObject);
begin
    OKHelpBottomDlg.ShowModal;
end;

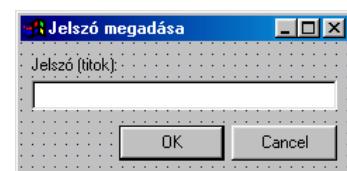
```

*TPasswordDlg* párbeszédablak és deklarációja:

```

type
    TPasswordDlg = class (TForm)
        Label1: TLabel;
        Password: TEdit;
        OKBtn: TButton;
        CancelBtn: TButton;
        procedure OKBtnClick(Sender: TObject);
        procedure FormCreate(Sender: TObject);
        procedure CancelBtnClick(Sender: TObject);
    private
        jelszo: string;
    end;

```



Az OKBottomDlg1 párbeszédablak és deklarációja:

```
type
  TOKBottomDlg1 = class(TForm)
    OKBtn: TButton;
    CancelBtn: TButton;
    Panel1: TPanel;
    Panel3: TPanel;
    Panel4: TPanel;
    Panel5: TPanel;
    Panel6: TPanel;
    Panel7: TPanel;
    Panel8: TPanel;
    Panel9: TPanel;
    Bevel1: TBevel;
    Panel2: TPanel;
  end;
```



Pascal modulban tároljuk a globális változókat és az *Agrippa* függvényt:

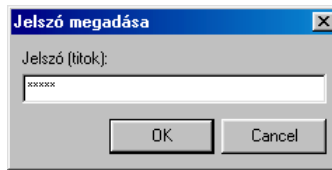
```
unit Modul;
interface
var Jelszo_van: boolean;
    meret: integer;
    matrix : array[0..120] of integer;
    function Agrippa(m: integer):boolean;

implementation

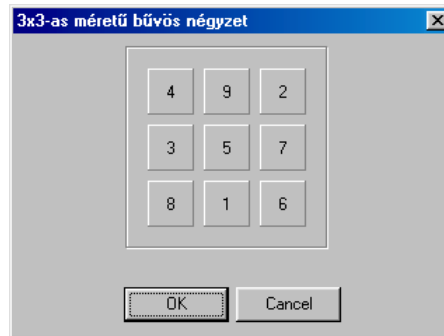
function Agrippa(m: integer):boolean;
var i,j, szam, mk: integer; kilep : boolean;
begin
  mk := (m+1)*(m+1);
  for i:=0 to mk do
    matrix[i] := 0;
  j := (m+1) div 2;
  i := j+1;
  szam := 1;
  kilep :=false;
  while (matrix[i+m*j] >= 0) do
  begin
    if matrix[i+m*j]= 0 then
    begin
      matrix[i+m*j]:= szam;
      if szam = m*m then
      begin
        kilep := true;
        break;
      end;
      szam:= szam+1; i:=i+1; j:=j+1;
      if i > m then
      begin
        if j> m then
        begin
          i:=2; j:= m;
        end
        else i:=1;
      end
      else
      begin
        if j > m then j:=1;
      end
      end
      else
      begin
        i:= i+1;
        j:= j-1;
        if i>m then i:=1;
      end;
    end;
    Agrippa:= kilep;
  end;
end.
```

A program néhány futási képe:

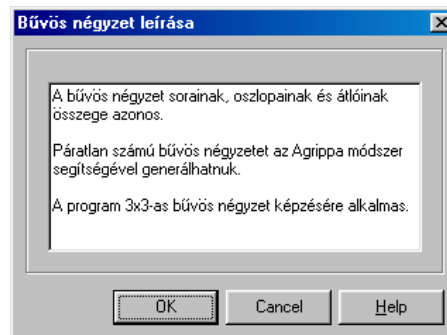
A Menü/Megjelenítés kiválasztásakor először a Jelszó megadása párbeszédablak jelenik meg:




Ha a *Jelszó* helyesen került beírásra, akkor jelenik meg a párbeszédablak a 3x3-as bűvös négyzettel:



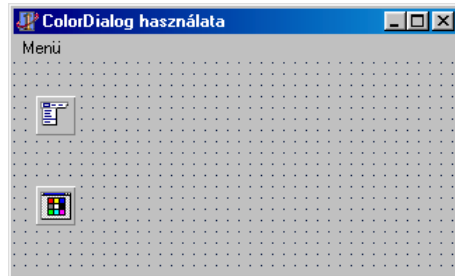
A Segítség/Leírás menüpont kiválasztásakor jelenik meg:



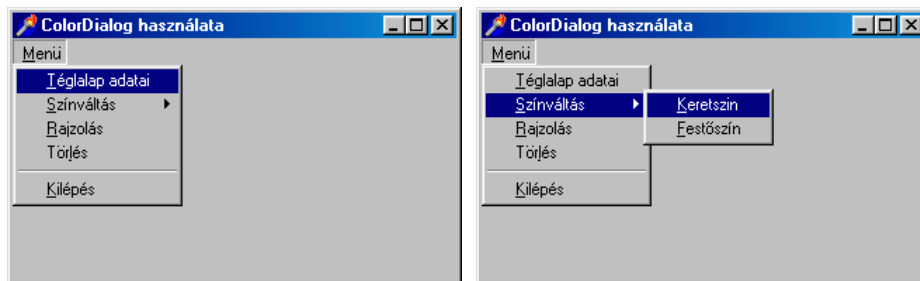


 Rajzoljunk színes téglalapot menüvezérelt programmal! A téglalap adatait billentyűzetről, vagy állományból olvassuk be! A téglalap keretszíne és belsejének festőszíne legyen megváltoztatható! (*Dialog\_Color*)

A *Form1* űrlapon helyezzük el a *ColorDialog* komponenst, amely lehetővé teszi a *ColorDialog* párbeszédablak megjelenítését színválasztás céljából!



A program menürendszere:



A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    Menu: TMenuItem;
    TAdatok: TMenuItem;
    Szin: TMenuItem;
    ColorDialog1: TColorDialog;
    N1: TMenuItem;
    Kilepes: TMenuItem;
    Rajzol: TMenuItem;
    Keretszin: TMenuItem;
    Festoszin: TMenuItem;
    Torles: TMenuItem;
    procedure KilepesClick(Sender: TObject);
    procedure TAdatokClick(Sender: TObject);
    procedure RajzolClick(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure KeretszinClick(Sender: TObject);
    procedure FestoszinClick(Sender: TObject);
    procedure TorlesClick(Sender: TObject);
  end;
```

Egyéb deklarációk:

```
var
  Fnev : string;
  X,Y,DX,DY : integer;
  fx : text;
  rajz : boolean;
  KeretSzine,FestesSzine : TColor;
```

A "Téglalap adatai" menüpont kiválasztásakor meghívódó *TadatokClick* eseménykezelő eljárás jeleníti meg az *OKBottomDlg* párbeszédablakot, amely a téglalap adatait olvassa be. A *rajz* változó **false**-ra állításával és az **Invalidate** metódus meghívásával töröljük az űrlap rajzfelületét.

```
procedure TForm1.TadatokClick(Sender: TObject);
begin
    OkBottomDlg.ShowModal;
    rajz := false;
    Invalidate;
end;
```

A "Rajzolás" menüpont kiválasztásakor meghívódó *RajzolClick* eseménykezelő eljárással először töröljük a korábbi rajzot, majd a *rajz* változó **true** értékre állításával és az **Invalidate** metódus meghívásával a *FormPaint* eseménykezelő megrajzolja az új téglalapot.

```
procedure TForm1.RajzolClick(Sender: TObject);
begin
    // előző téglalap törlése
    rajz := false;
    Invalidate;
    rajz := true;
    // a háttér frissítése
    Invalidate;
end;
```

A *FormPaint* eseménykezelő eljárás az **Invalidate** metódus hívásának hatására szólal meg. A *rajz* változó **true** értéke esetén rajzol, egyébként törli a *Form1* űrlap felületét.

```
procedure TForm1.FormPaint(Sender: TObject);
begin
    if rajz then
    begin
        // rajzol
        Canvas.Brush.Color := KeretSzine;
        Canvas.Pen.Color := KeretSzine;
        Canvas.Rectangle(X,Y, X+t.a_old,Y+t.b_old);
        Canvas.Brush.Color := FestesSzine;
        Canvas.Rectangle(X+DX,Y+DY,X+DX+t.a_old-2*DX,Y+DY+t.b_old-2*DY);
    end;
    // különben töröl
end;
```

A *FormCreate* eseménykezelő eljárásban a *rajz* változó **false** értéket kap és a téglalap megjelenítéshez szükséges adatok kezdőértéket kapnak.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    rajz := false;
    X := 80; Y := 80;
    DX := 20; DY := 20;
    KeretSzine := clpurple;
    FestesSzine := clyellow;
end;
```

A "Színváltás/Keretszin" menüpont kiválasztásakor meghívódó *KeretszinClick* eseménykezelő eljárás megjeleníti a **ColorDialog** párbeszédablakot, és a *KeretSzine* változó felveszi a kijelölt színt, amely az **Invalidate** meghívásával azonnal aktualizálódik.

```
procedure TForm1.KeretszinClick(Sender: TObject);
begin
    ColorDialog1.Execute;
    KeretSzine := ColorDialog1.Color;
    Invalidate;
end;
```

A "Szinváltság/Festőszin" menüpont kiválasztásakor meghívódó *FestoszinClick* eseménykezelő eljárás megjeleníti a **ColorDialog** párbeszédablakot, és a *FestésSzine* változó felveszi a kijelölt színt, amely az *Invalidate* meghívásával azonnal aktualizálódik.

```
procedure TForm1.FestoszinClick(Sender: TObject);
begin
    ColorDialog1.Execute;
    FestesSzine := ColorDialog1.Color;
    Invalidate;
end;
```

A "Törlés" menüpont kiválasztásakor meghívódó *TorlésClick* eseménykezelő eljárás törli a rajzterületet.

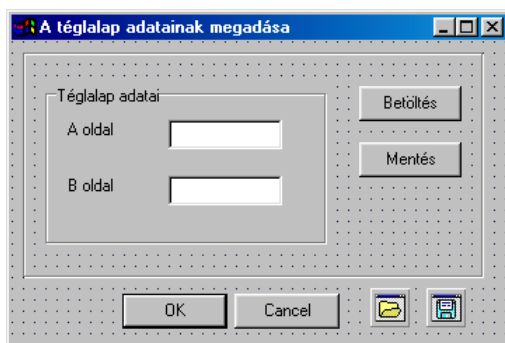
```
procedure TForm1.TorlesClick(Sender: TObject);
begin
    rajz := false;
    Invalidate;
end;
```

A "Kilépés" menüpont kiválasztásakor meghívódó *KilepesClick* eseménykezelő eljárás megszakítja program futását.

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

## Form2 űrlap

Form2 űrlapon tervezzük meg a párbeszédablakot a téglalap adatainak megadására. Az adatokat megadhatjuk billentyűzetről, vagy állományból is beolvashatjuk. A téglalap adatait kimenthetjük állományba.



A *TForm1* osztály deklarációja:

```
type
    TOKBottomDlg = class(TForm)
        OKBtn: TButton;
        CancelBtn: TButton;
        Bevel1: TBevel;
        GroupBox1: TGroupBox;
        Label1: TLabel;
        Aoldal: TEdit;
        Label2: TLabel;
        Boldal: TEdit;
        OpenFileDialog1: TOpenDialog;
        SaveDialog1: TSaveDialog;
        Betolt: TButton;
        Ment: TButton;
        procedure OKBtnClick(Sender: TObject);
        procedure BetoltClick(Sender: TObject);
        procedure MentClick(Sender: TObject);
    end;
```

*További deklaráció:*

```
var
  OKBottomDlg: TOKBottomDlg;
```

Az *OKBtbClick* eseménykezelő eljárás olvassa le a téglalap adatait és tárolja a *t* globális változóba (Lásd a *Modul-t*).

```
procedure TOKBottomDlg.OKBtnClick(Sender: TObject);
begin
  try
    t.a_old := StrToInt(Aoldal.Text);
  except
    t.a_old := 100;
  end;
  try
    t.b_old := StrToInt(Boldal.Text);
  except
    t.b_old := 100;
  end;
end;
```

A *BetoltClick* eseménykezelő eljárás nyitja meg a \*.dat szűrővel a *Megnyitás* párbeszédablakot, amellyel a korábban tárolt adatokból olvashatunk.

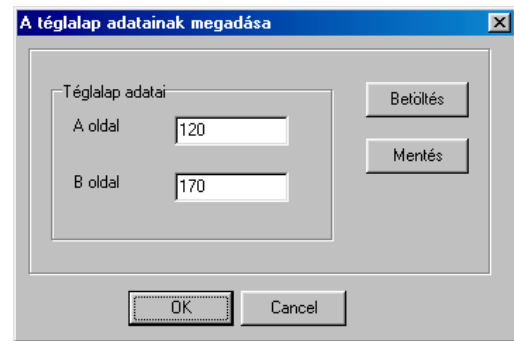
```
procedure TOKBottomDlg.BetoltClick(Sender: TObject);
begin
  OpenFileDialog1.FileName := '*.dat';
  if OpenFileDialog1.Execute then
  begin
    Fnev := OpenFileDialog1.FileName;
    AssignFile(fx, Fnev);
    Reset(fx);
    while not Eof(fx) do
    begin
      Readln(fx, t.a_old, t.b_old);
      Aoldal.Text := IntToStr(t.a_old);
      Boldal.Text := IntToStr(t.b_old);
    end;
    CloseFile(fx);
  end;
end;
```

A *MentésClick* eseménykezelő eljárás nyitja meg a \*.dat szűrővel a *Mentés másként* párbeszédablakot, amellyel a téglalap adatait állományba menthetjük.

```
procedure TOKBottomDlg.MentClick(Sender: TObject);
var
  Outf: TextFile;
begin
  SaveDialog1.FileName := '*.dat';
  if SaveDialog1.Execute then
  begin
    Fnev := SaveDialog1.FileName;
    AssignFile(Outf, Fnev);
    Rewrite(Outf);
    Writeln(Outf, t.a_old:6, t.b_old:6);
    CloseFile(Outf);
    Form1.Caption := 'Mentés: ' + Fnev;
  end;
end;
```

A *Modul* tartalmazza a *Teglalap* osztály deklarációját és a *t* globális változó deklarációját. A konstruktor kezdőértéket ad a két adatmezőnek.

```
unit Modul;  
  
interface  
type  
  Teglalap = class  
    a_old, b_old : integer;  
    constructor létrehoz;  
  end;  
var  
  t: Teglalap;  
implementation  
  // konstruktor  
  constructor Teglalap.Letrehoz;  
  begin  
    a_old := 100;  
    b_old := 100;  
  end;  
initialization  
  // a t objektum létrehozása  
  t := Teglalap.Letrehoz;  
finalization  
  // a t objektum megszüntetése  
  t.free;  
end.
```

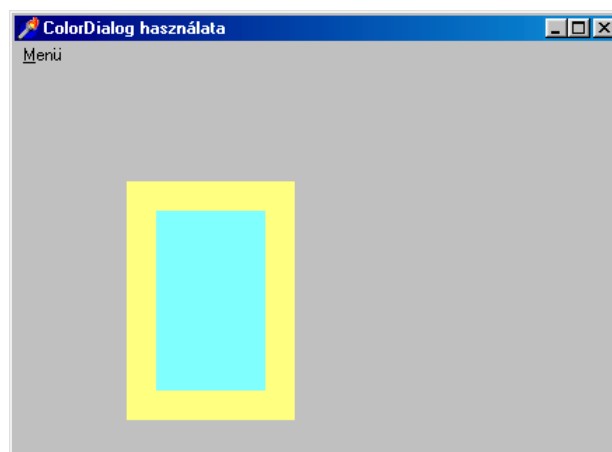


*Futási eredmény:*

A téglalap adatait állítsuk be az alábbiakra:

A *Színváltás* menüpont *Keretszín* és a *Festőszín* menüpontjának kiválasztásával állítsuk be a rajzoláshoz a színeket. Legyen a téglalap keretszíne sárga, a festés színe türkiz!

A *Rajzolás* menüpont kiválasztásakor megjelenik a kívánt téglalap.



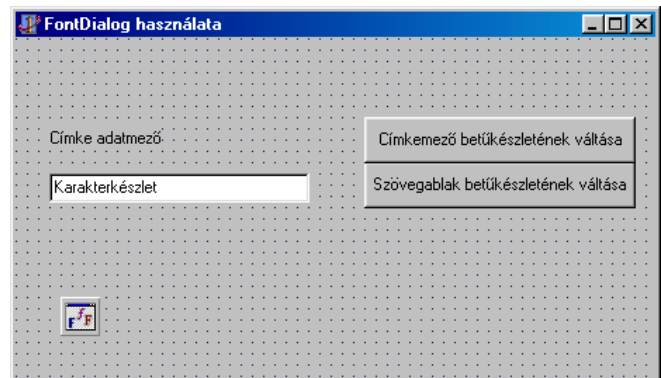


Készítsünk olyan programot, amelyben az **Edit** és **Label** komponensek szövegének illetve fejlécének betűtípusát, betűstílusát és méretét változtatni lehet! (*Dialog\_font*)

A *Form1* űrlapon helyezzük el a **FontDialog** komponenst, amely lehetővé teszi a *FontDialog* párbeszédablak megjelenítését betűkészlet választása céljából.

A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    FontDialog1: TFontDialog;
    Button1: TButton;
    Label1: TLabel;
    Edit1: TEdit;
    Cime: TButton;
  procedure SzovegClick(Sender: TObject);
  procedure CimeClick(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  end;
```



A *FormCreate* eseménykezelő eljárásban a *FontDialog1* objektumpéldány néhány tulajdonságának adunk kezdőértéket.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  // Méret beállítása
  FontDialog1.Options := [fdLimitSize];
  FontDialog1.MaxFontSize := 14;
  FontDialog1.MinFontSize := 8;
end;
```

A "Szövegablak betűkészletének váltása" menüpont kiválasztásakor meghívódó *SzovegClick* eseménykezelő eljárás megjeleníti a *FontDialog* párbeszédablakot, és a *Edit1* szövegmező **Font** tulajdonsága átveszi a kiválasztott betűtípust.

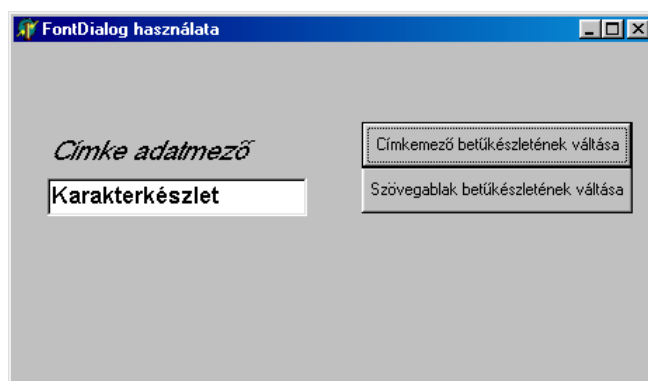
```
procedure SzovegClick(Sender: TObject);
begin
  if FontDialog1.Execute then
    Edit1.Font := FontDialog1.Font;
end;
```

A "Címkemező betűkészletének váltása" menüpont kiválasztásakor meghívódó *CimeClick* eseménykezelő eljárás megjeleníti a *FontDialog* párbeszédablakot, és a *Label1* címke *Font* tulajdonsága átveszi a kiválasztott betűtípust.

```
procedure TForm1.CimeClick(Sender: TObject);
begin
  if FontDialog1.Execute then
    Label1.Font := FontDialog1.Font;
end;
```

A program egyik futási ablaka:

A *Label1* **Caption** tulajdonsága *MS Sans Serif-Dőlt-14* betűtípussal, míg az *Edit1* **Text** tulajdonsága *Ariel-Félkövér-12* betűtípussal jelenik meg.





Készítsünk olyan programot, amelyik bitkép beolvasására és kimentésére az **OpenPictureDialog** és a **SavePictureDialog** párbeszédablakokat használja fel. (*Dialog\_picture*)

A Form1 űrlapon helyezzük el az **OpenPictureDialog** és a **SavePictureDialog** komponenseket, amelyek lehetővé teszik a bitképek könnyű betöltését és tárolását.

A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    OpenPictureDialog1: TOpenPictureDialog;
    SavePictureDialog1: TSavePictureDialog;
    Kilepes: TButton;
    Megjelenites: TButton;
    Mentos: TButton;
    Label1: TLabel;
    procedure KilepesClick(Sender: TObject);
    procedure MegjelenitesClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure MentosClick(Sender: TObject);
  end;
```

Egyéb változók deklarációja:

```
var
  Form1: TForm1;
  kep : TBitmap; // a bitkép tárolására
  ARect: TRect; // tartomány a bitkép megjelenítéshez
```

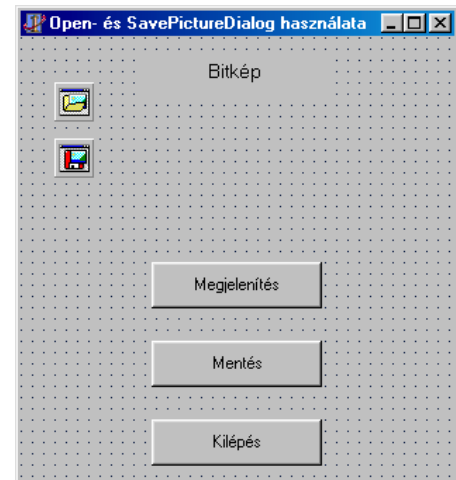
A *FormCreate* eseménykezelő eljárásban létrehozuk a **TBitmap** objektumpéldányt, értéket adunk az *ARect* tartománynak és letiltjuk a *Mentos* nyomógomb hozzáférését.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  kep := TBitmap.Create;
  ARect.Left := 105;
  ARect.Top := 50;
  ARect.Right := 205;
  ARect.Bottom := 150;
  Mentos.Enabled := false;
end;
```

A "Megjelenítés" nyomógomb megnyomásakor meghívódó *MegjelenitesClick* eseménykezelő eljárásban megjelenítjük az *OpenPictureDialog* párbeszédablakot a bitkép betöltésére. A kiválasztott állomány nevét a **FileName** tulajdonság tárolja. Ezt követően a **LoadFromFile** módszerrel betöltött bitképet a **StretchDraw** módszerrel a tartományhoz igazítva rajzolja ki, majd a *Mentos* gomb hozzáférhetővé válik.

```
procedure TForm1.MegjelenitesClick(Sender: TObject);
begin
  if OpenPictureDialog1.Execute then
  begin
    kep.LoadFromFile(OpenPictureDialog1.FileName);
    Canvas.StretchDraw(ARect,kep);
    Mentos.Enabled := true;
  end;
end;
```

A "Mentés" nyomógomb megnyomásakor meghívódó *MentosClick* eseménykezelő eljárásban a *SavePictureDialog* párbeszédablakot jelenítjük meg. Ha a megadott **FileName** nem tartalmazza a .bmp kiterjesztést akkor a *pos* nullát ad és ebben az esetben az állománynévhez a .bmp kiterjesztés hozzáadjuk. A *Mentos* nyomógombot újra hozzáférhetetlenné tesszük.





```

procedure TForm1.MentesClick(Sender: TObject);
begin
    if SavePictureDialog1.Execute then
        begin
            if pos('.bmp',SavePictureDialog1.FileName)>0 then
                kep.SaveToFile(SavePictureDialog1.FileName)
            else
                kep.SaveToFile(SavePictureDialog1.FileName+'.bmp');
            end;
            Mentes.Enabled := false;
end;

```

A "Kilépés" nyomógomb megnyomásakor meghívódó *KilepesClick* eseménykezelő eljárásban a *kep TBitmap* objektumpéldányt felszabadítjuk és a program befejezi a futását.


```

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

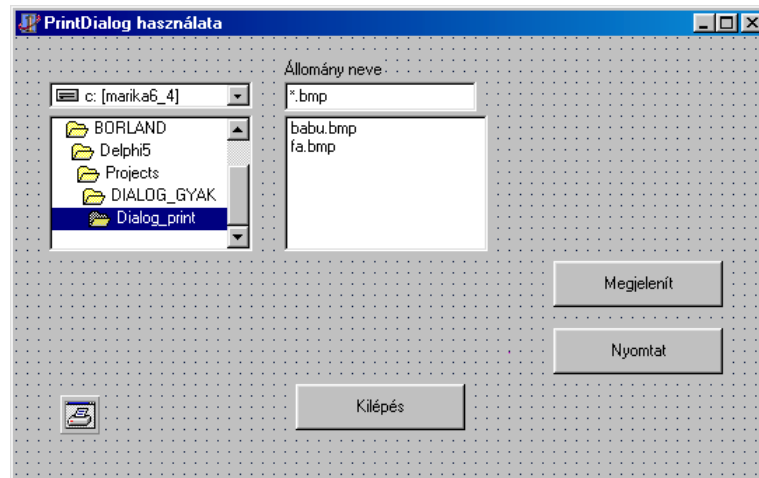
```

A program futási képe:



 Tervezzünk bitkép beolvasására és kimentésére programot, amelyben lehetőség van a bitkép kinyomtatására is! (*Dialog\_print*)

A *Form1* űrlapon helyezzük el az **PrintDialog** komponenst, amely lehetővé teszi a bitképek nyomtatóra való küldését.



A *TForm1* osztály deklarációja:

```
type
  TForm1 = class(TForm)
    FileListBox1: TFileListBox;
    DriveComboBox1: TDriveComboBox;
    DirectoryListBox1: TDirectoryListBox;
    Label1: TLabel;
    Edit1: TEdit;
    Kilepes: TButton;
    Megjelenit: TButton;
    PrintDialog1: TPrintDialog;
    Nyomtat: TButton;
  procedure KilepesClick(Sender: TObject);
  procedure MegjelenitClick(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure NyomtatClick(Sender: TObject);
end;
```

Egyéb deklarációk:

```
var
  kep : TBitmap;           // a bitkép tárolására
  ARect: TRect;            // a bitkép megjelenítésének tartománya
  betoltve : boolean;      // a bitmap betöltésének megjelölésére szolgál
```

Ha nyomtatót használunk, akkor a programban be kell tölteni a *Printers* unitot.

```
implementation
uses Printers;
```

A *FormCreate* eseménykezelő eljárásban a *kep* változóban létrehozuk a **TBitmap** osztály objektumtípusát, kezdőértéket adunk a kép megjelenítéséhez szükséges tartománynak, majd a *betoltve* változót **false** értékre állítjuk.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    kep := TBitmap.Create;
    ARect.Left := 400;
    ARect.Top := 50;
    ARect.Right := 500;
    ARect.Bottom := 150;
    betoltve := false;
end;
```

A "Megjelenít" nyomógomb megnyomásakor meghívott *MegjelenitesClick* eseménykezelő eljárásban rajzoljuk ki meg a bitképet. A *DriveComboBox* vezérlőből kiválasztjuk a perifériát, a *DirectoryListBox* vezérlőből a könyvtárat, a *FileListBox* vezérlőből pedig a bitkép állomány nevét, amely megjelenik az *Állomány neve* adatmezőben. Ha az adatmező üres, vagy \*.bmp, akkor üzenetet küldünk, különben betöltjük a bitképet *LoadFromFile* metódussal, a *StretchDraw* metódussal pedig kirajzoljuk a *ARect* tartományba. A *betolt* változót **true** értékre állítjuk.

```
procedure TForm1.MegjelenitClick(Sender: TObject);
begin
    if (Edit1.Text = '*.bmp') or
       (Edit1.Text = '')
    then ShowMessage('Az állomány neve hibás')
    else
    begin
        kep.LoadFromFile(Edit1.Text);
        Canvas.StretchDraw(ARect, kep);
        betoltve := true;
    end;
end;
```

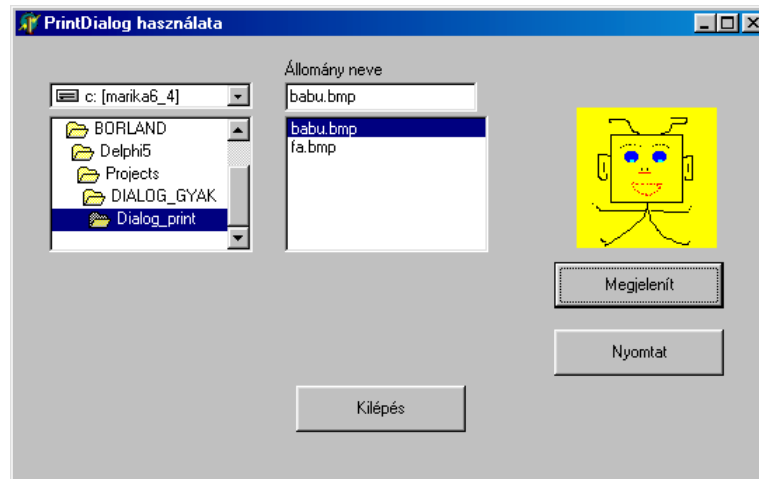
A "Nyomtat" nyomógomb megnyomásakor meghívott *NyomtatClick* eseménykezelő eljárásban a *PrintDialog* párbeszédablakot jelenítjük meg a nyomtatáshoz, ha a *betoltve* változó értéke **true** (a bitkép betöltését jelzi). A bitkép nagyítva kapjuk meg a papírlapon.

```
procedure TForm1.NyomtatClick(Sender: TObject);
begin
    if betoltve then
    begin
        if PrintDialog1.Execute then
        begin
            with Printer do
            begin
                BeginDoc;
                Canvas.StretchDraw(Canvas.ClipRect, kep);
                EndDoc;
            end
        end
    end
    else ShowMessage('A kép nincs betöltve');
end;
```

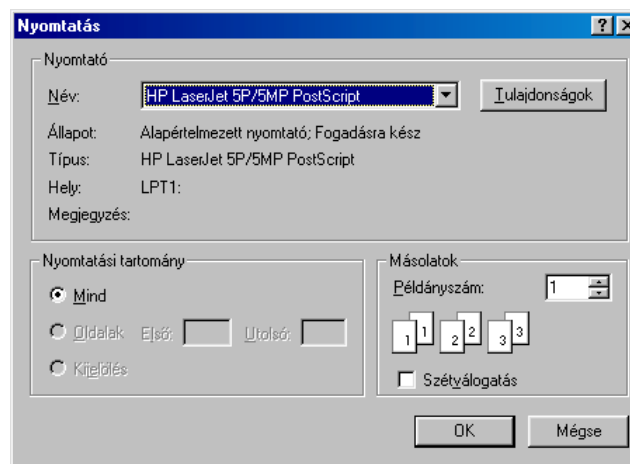
A "Kilépés" nyomógomb megnyomásakor hívott *KilepesClick* eseménykezelő eljárásban felszabadítjuk a *kep* változóban tárolt **TBitmap** objektumtípusú objektumot, a *betoltve* változót **false** értékre állítjuk és a program befejezi a működését.

```
procedure TForm1.KilepesClick(Sender: TObject);
begin
    kep.Free;
    betoltve:=false;
    Application.Terminate;
end;
```

A program futási képe:



A *Nyomtat* gomb megnyomásakor a *Nyomtatás* ablakban indíthatjuk el a nyomtatást.



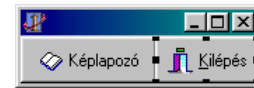


Készítsünk programot, melyben a párbeszédablak felületén **TabControl** vezérlővel váltogatjuk a képeket! (*Regisztratul*)

Készítsük el az alkalmazás főablakát (*Form1*)! Az ablak csak két gombot tartalmaz, egyet a képlapozó párbeszédablak indítására és egyet a kilépésre.

A *Form1* osztálya:

```
type
  TForm1 = class(TForm)
    SpeedButton1: TSpeedButton;
    BitBtn1: TBitBtn;
    procedure SpeedButton1Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```



Mivel a *Unit2* tartalmazza a párbeszédablakot szükséges a modulra való hivatkozás.

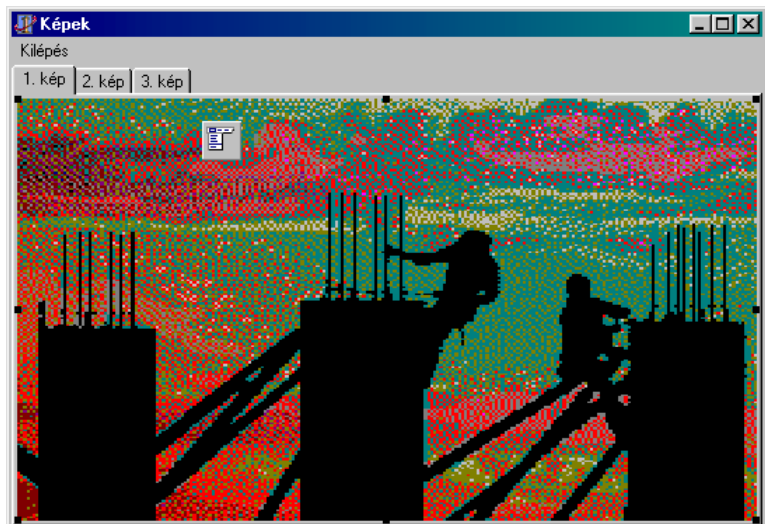
```
uses Unit2;
```

A gombnyomás események kezelői:

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  Kepek.ShowModal;
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  Application.Terminate;
end;
```

Készítsük el a *Kepek* formot tartalmazó párbeszédablakot! Helyezzünk rá egy menüt a *Kilépés* menüpontnak (*Kilps1*), egy regisztrál vezérlőt (*TabControl1*)! A *TabControl1 Tabs* tulajdonságába írjuk be a három fül nevét a „*String List editor*” segítségével! Állítsuk az *Align* tulajdonságot *alClient* értékre! Helyezzünk három *Image* vezérlőt a regisztrál vezérlőre, állítsuk ezek *Align* tulajdonságát is *alClient* értékre, majd mindegyikbe töltsünk képet a *Picture* tulajdonság segítségével! Ha az *Image* vezérlők *Stretch* jellemzőjét igazra állítjuk, akkor a képek kitöltik a vezérlőt.



Ezek után arról kell gondoskodni a programban, hogy mindig a megfelelő kép legyen látható és a többi láthatatlan.

A *Kepek* megjelenésekor alapállapotot állítunk be:

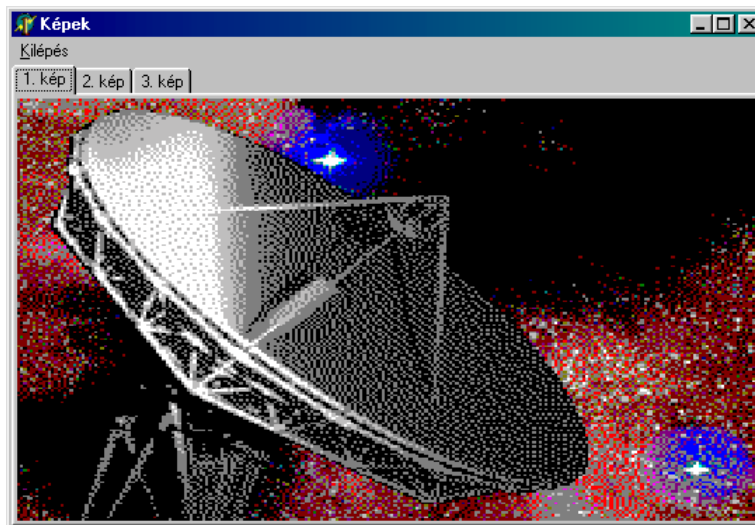
```
procedure TKepek.FormShow(Sender: TObject);  
begin  
    Image1.Visible:=true;  
    Image2.Visible:=false;  
    Image3.Visible:=false;  
end;
```

Az aktuális lap változásakor cseréljük a képet

```
procedure TKepek.TabControl1Change(Sender: TObject);  
begin  
    Image1.Visible:=false;  
    Image2.Visible:=false;  
    Image3.Visible:=false;  
    Case TabControl1.Tabindex of  
        0: Image1.Visible:=true;  
        1: Image2.Visible:=true;  
        2: Image3.Visible:=true;  
    end;  
end;
```

A *Kilépés* menü választásakor elrejtjük a párbeszédablakot

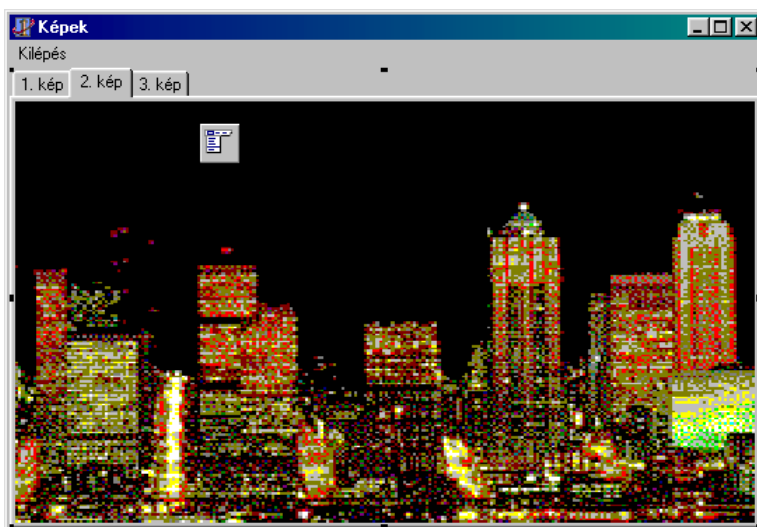
```
procedure TKepek.Kilps1Click(Sender: TObject);  
begin  
    Kepek.close;  
end;
```





Készítsünk programot, melyben a párbeszédablak felületén *PageControl* vezérlővel váltogatjuk a képeket! (*Regiszer*)

A programot ugyanúgy készítjük el, mint ahogy azt a [Regiszerful](#) programban tettük, azonban a párbeszédablak felületén most *PageControl* vezérlő helyezkedik el. A három kép külön-külön szerepel a három lapon, így azzal sem kell foglalkoznunk, hogy mikor melyik látható.



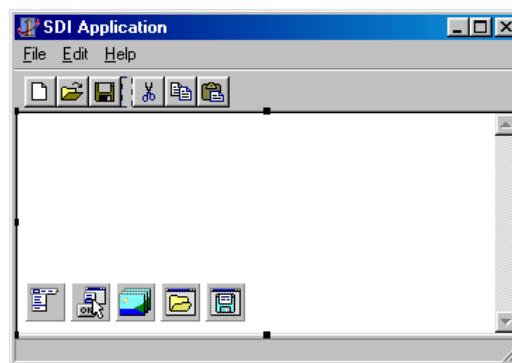


Készítsünk olyan programot, amelyben felhasználjuk a Delphi rendszer által előre definiált SDI alkalmazást! A szövegek szerkesztéséhez használjunk **Memo** komponenst! (*Sdi\_1*)

A **File/New** menüpont kiválasztásával megjelenő „New Items” párbeszédablakban először a *DataModules* fülön, majd az „SDI Application” ikonon kattintunk. A megjelenő „Select Directory” párbeszédablakban megadjuk azt a könyvtárat, ahol a minta SDI alkalmazást létre kívánjuk hozni. Ezt követően a formra egy **Memo** vezérlőt helyezünk, melynek **Align** tulajdonságát *alClient* értékre állítjuk.

A *TForm1* osztály deklarációja:

```
type
  TSDIAppForm = class(TForm)
    OpenDialog: TOpenDialog;
    SaveDialog: TSaveDialog;
    ToolBar1: TToolBar;
    ToolButton9: TToolButton;
    ToolButton1: TToolButton;
    ToolButton2: TToolButton;
    ToolButton3: TToolButton;
    ToolButton4: TToolButton;
    ToolButton5: TToolButton;
    ToolButton6: TToolButton;
    ActionList1: TActionList;
    FileNew1: TAction;
    FileOpen1: TAction;
    FileSave1: TAction;
    FileSaveAs1: TAction;
    FileExit1: TAction;
    EditCut1: TEditCut;
    EditCopy1: TEditCopy;
    EditPaste1: TEditPaste;
    HelpAbout1: TAction;
    StatusBar: TStatusBar;
    ImageList1: TImageList;
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    FileNewItem: TMenuItem;
    FileOpenItem: TMenuItem;
    FileSaveItem: TMenuItem;
    FileSaveAsItem: TMenuItem;
    N1: TMenuItem;
    FileExitItem: TMenuItem;
    Edit1: TMenuItem;
    CutItem: TMenuItem;
    CopyItem: TMenuItem;
    PasteItem: TMenuItem;
    Help1: TMenuItem;
    HelpAboutItem: TMenuItem;
    Memo1: TMemo;
    procedure FileNew1Execute(Sender: TObject);
    procedure FileOpen1Execute(Sender: TObject);
    procedure FileSave1Execute(Sender: TObject);
    procedure FileExit1Execute(Sender: TObject);
    procedure HelpAbout1Execute(Sender: TObject);
  end;
```



A **File/New** menüpont kiválasztásakor meghívódó *FileNew1Execute* eseménykezelő eljárásban a *Memo1* vezérlő tartalmát töröljük.

```
procedure TSDIAppForm.FileNew1Execute(Sender: TObject);
begin
  Memo1.Clear;
end;
```

A **File/Open** menüpont kiválasztásakor meghívódó *FileOpen1Execute* eseménykezelő eljárásban az állományszűrőt „\*.txt”-re állítjuk. Megjelenítjük az *OpenDialog* párbeszédablakot az állomány betöltésére. A *Memo1* vezérlő tartalmát töröljük és a **LoadFromFile** metódussal beolvassuk a szöveges állományt a *Memo1* vezérlő **Lines** tulajdonságába.



```

procedure TSDIAppForm.FileOpen1Execute(Sender: TObject);
begin
  OpenDialog.FileName:= '*.txt';
  if OpenDialog.Execute then
    begin
      Memo1.Clear;
      Memo1.Lines.LoadFromFile(OpenDialog.FileName);
    end;
end;

```

A *File/Save* ill. „*File/Save As...*” menüpont kiválasztásakor meghívódó *FileSave1Execute* eseménykezelő eljárásban az állományszűrőt „\*.txt”-re állítjuk. Megjelenítjük a *SaveDialog* párbeszédablakot az állomány elmentésére. A *Memo1* vezérlő **Lines** tulajdonság tartalmát **SaveToFile** metódussal a háttértárra mentjük.

```

procedure TSDIAppForm.FileSave1Execute(Sender: TObject);
begin
  SaveDialog.FileName:= '*.txt';
  if SaveDialog.Execute then
    Memo1.Lines.SaveToFile(SaveDialog.FileName);
end;

```

A *File/Exit* menüpont kiválasztásakor a *FileExit1Execute* eseménykezelő eljárásban az alkalmazás befejezi a működését.

```

procedure TSDIAppForm.FileExit1Execute(Sender: TObject);
begin
  Close;
end;

```

A „*Help/About...*” menüpont kiválasztásakor a *HelpAbout1Execute* eseménykezelő eljárásban megjelenítjük az *AboutBox* ablakot.

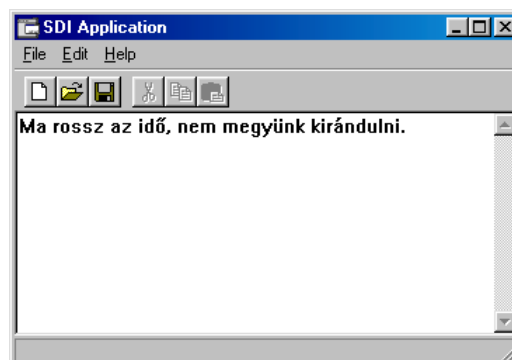
```

procedure TSDIAppForm.HelpAbout1Execute(Sender: TObject);
begin
  AboutBox.ShowModal;
end;

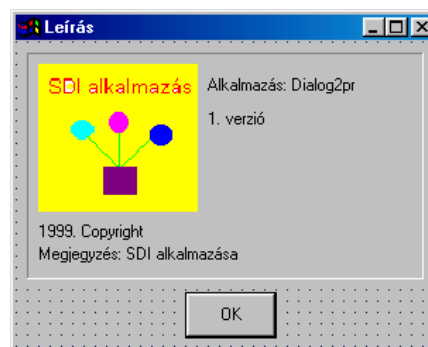
```

*A program futási képe:*

A kijelölt szöveget kivághatjuk az *Edit/Cut*, másolásra kijelölhetjük az *Edit/Copy* és másolhatjuk az *Edit/Paste* menüpontok kiválasztásával.



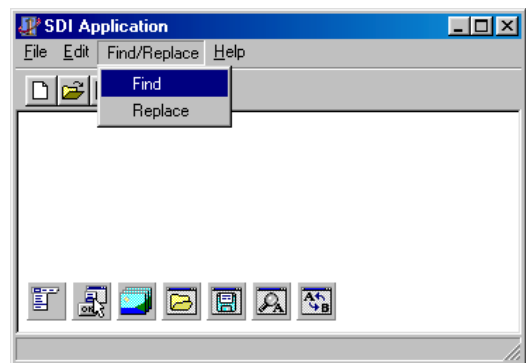
Az *Help/About* menüválasztás hatására megjelenő párbeszédablak:





Készítsünk olyan SDI alkalmazást, ahol a szerkesztőmező **RichEdit** komponens, és legyen lehetőség szöveg keresésére és helyettesítésére is! (*Dialog\_find*)

A **File/New** menüpont kiválasztásával megjelenő „New Items” párbeszédablakban a **DataModules** fület kiválasztva, majd az „SDI Application” ikonon kattintva, a megjelenő „Select Directory” párbeszédablakból kiválasztjuk a könyvtárat, ahová kérjük a minta SDI alkalmazás létrejöttét. Ezt követően a formra helyezünk egy **RichEdit** vezérlőt, melynek **Align** tulajdonságát **alClient** értékre állítjuk. A form menürendszerét a **Find/Replace** főmenüvel és a **Find**, illetve a **Replace** menüpontokkal bővítjük. (A műveletek elvégzése érdekében az űrlapra kell helyezni a **FindDialog** és a **ReplaceDialog** komponenseket is.) A **TForm1** osztály deklarációja:



**type**

```

TSDIAppForm = class(TForm)
    OpenDialog: TOpenDialog;
    SaveDialog: TSaveDialog;
    ToolBar1: TToolBar;
    ToolButton9: TToolButton;
    ToolButton1: TToolButton;
    ToolButton2: TToolButton;
    ToolButton3: TToolButton;
    ToolButton4: TToolButton;
    ToolButton5: TToolButton;
    ToolButton6: TToolButton;
    ActionList1: TActionList;
    FileNew1: TAction;
    FileOpen1: TAction;
    FileSave1: TAction;
    FileSaveAs1: TAction;
    FileExit1: TAction;
    EditCut1: TEditCut;
    EditCopy1: TEditCopy;
    EditPaste1: TEditPaste;
    HelpAbout1: TAction;
    StatusBar: TStatusBar;
    ImageList1: TImageList;
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    FileNewItem: TMenuItem;
    FileOpenItem: TMenuItem;
    FileSaveItem: TMenuItem;
    FileSaveAsItem: TMenuItem;
    N1: TMenuItem;
    FileExitItem: TMenuItem;
    Edit1: TMenuItem;
    CutItem: TMenuItem;
    CopyItem: TMenuItem;
    PasteItem: TMenuItem;
    Help1: TMenuItem;
    HelpAboutItem: TMenuItem;
    FindDialog1: TFindDialog;
    ReplaceDialog1: TReplaceDialog;
    FindReplace1: TMenuItem;
    Find1: TMenuItem;
    Replace1: TMenuItem;
    RichEdit1: TRichEdit;
    procedure FileNew1Execute(Sender: TObject);
    procedure FileOpen1Execute(Sender: TObject);
    procedure FileSave1Execute(Sender: TObject);
    procedure FileExit1Execute(Sender: TObject);
    procedure HelpAbout1Execute(Sender: TObject);
    procedure Replace1Click(Sender: TObject);
    procedure FindDialog1Find(Sender: TObject);
    procedure Find1Click(Sender: TObject);
    procedure ReplaceDialog1Replace(Sender: TObject);
end;

```

A *File/New* menüpont kiválasztásakor meghívódó *FileNew1Execute* eseménykezelő eljárásban a ***RichEdit*** vezérlő tartalmát töröljük.

```
procedure TSDIAppForm.FileNew1Execute(Sender: TObject);
begin
    RichEdit1.Clear;
end;
```

A *File/Open* menüpont kiválasztásakor meghívódó *FileOpen1Execute* eseménykezelő eljárásban az állományszűrőt „\*.txt”-re állítjuk. Megjelenítjük az ***OpenDialog*** párbeszédablakot az állomány betöltésére. A *RichEdit1* vezérlő tartalmát töröljük és a ***LoadFromFile*** metódussal beolvassuk a szöveges állományt a *RichEdit1* vezérlő ***Lines*** tulajdonságába.

```
procedure TSDIAppForm.FileOpen1Execute(Sender: TObject);
begin
    OpenFileDialog.FileName:= '*.txt';
    if OpenFileDialog.Execute then
    begin
        RichEdit1.Clear;
        RichEdit1.Lines.LoadFromFile(OpenDialog.FileName);
    end;
end;
```

A *File/Save* ill. „*File/Save As...*” menüpont kiválasztásakor meghívódó *FileSave1Execute* eseménykezelő eljárásban az állományszűrőt „\*.txt”-re állítjuk. Megjelenítjük a ***SaveDialog*** párbeszédablakot az állomány elmentésére. A *Richedit1* vezérlő ***Lines*** tulajdonság tartalmát ***SaveToFile*** metódussal lemezre mentjük.

```
procedure TSDIAppForm.FileSave1Execute(Sender: TObject);
begin
    SaveDialog.FileName:= '*.txt';
    if SaveDialog.Execute then
        RichEdit1.Lines.SaveToFile(SaveDialog.FileName);
end;
```

A *File/Exit* menüpont kiválasztásakor a *FileExit1Execute* eseménykezelő eljárásban az alkalmazás befejezi a működését.

```
procedure TSDIAppForm.FileExit1Execute(Sender: TObject);
begin
    Close;
end;
```

A „*Help/About...*” menüpont kiválasztásakor a *HelpAbout1Execute* eseménykezelő eljárásban megjelenítjük az *AboutBox* ablakot.

```
procedure TSDIAppForm.HelpAbout1Execute(Sender: TObject);
begin
    AboutBox.ShowModal;
end;
```

A *Find/Replace* menüpont *Find* menütételen kattintva a *Find1Click* eseménykezelő eljárásban beállítjuk a ***FindDialog*** párbeszédablak bal felső sarkának pozícióját, majd megjelenítjük a párbeszédablakot.

```
procedure TSDIAppForm.Find1Click(Sender: TObject);
begin
    FindDialog1.Position := Point(RichEdit1.Left + RichEdit1.Width, RichEdit1.Top);
    FindDialog1.Execute;
end;
```

A keresendő szöveg beírása után a *Következő* (*Find*) gomb megnyomásakor hívódik meg a *FindDialog1Find* eseménykezelő eljárás, amelyben a ***FindText*** metódussal keressük meg a szöveget. Ha talált, akkor a keresett szó első karakterének pozícióját adja vissza és a szöveget kijelöljük.

```
procedure TSDIAppForm.FindDialog1Find(Sender: TObject);
var
  Kezd, Veg : integer;
  Talalt : LongInt;
begin
  Talalt := -1;
  With RichEdit1 do
    begin
      RichEdit1.SelLength := 0;
      Kezd := 0;
      Veg := Length( Text) - Kezd;
      Talalt := FindText(FindDialog1.FindText, Kezd, Veg, [stMatchCase]);
      if Talalt <> -1 then
        begin
          SetFocus;
          SelStart := Talalt;
          SelLength := Length(FindDialog1.FindText);
        end;
      end;
    end;
end;
```

A *Find/Replace* menü *Replace* menüponját kiválasztja hívódik meg a *Replace1Click* eseménykezelő eljárás, melyben beállítjuk a ***ReplaceDialog*** párbeszédablak bal felső sarkának pozícióját és megjelenítjük a párbeszédablakot.

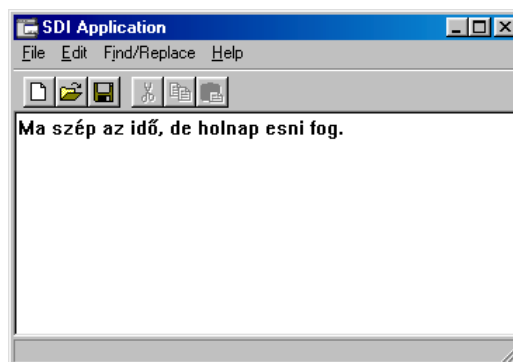
```
procedure TSDIAppForm.Replace1Click(Sender: TObject);
begin
  ReplaceDialog1.Position := Point(RichEdit1.Left + RichEdit1.Width, RichEdit1.Top);
  ReplaceDialog1.Execute;
end;
```

"Mit Keres" és "Mire cseréli" mezőket kitöltve megnyomjuk a *Következő*, majd a *Cserél* gombot, ennek hatására hívódik meg a *ReplaceDialog1Replace* eseménykezelő eljárás, amelyben a ***Pos*** függvénnyel megkeressük a szöveget. Ha a függvény megtalálta a szöveget, akkor a visszaadott érték nem nulla, ez esetben történik meg a csere a ***ReplaceText*** beírásával a ***SelText*** tulajdonságba.

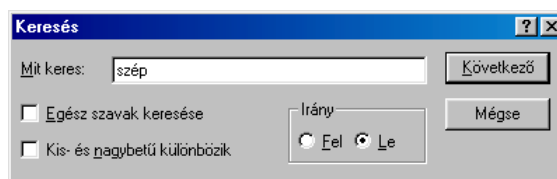
```
procedure TSDIAppForm.ReplaceDialog1Replace(Sender: TObject);
var
  Talal : integer;
begin
  with TReplaceDialog(Sender) do
    begin
      Talal := Pos(FindText, RichEdit1.Lines.Text);
      if Talal > 0 then
        begin
          RichEdit1.SelStart := Talal - 1;
          RichEdit1.SelLength := Length(FindText);
          RichEdit1.SelText := ReplaceText;
        end
      else
        ShowMessage(FindText + ' nem talált már!');
      end;
    end;
end;
```

A program néhány futási képe:

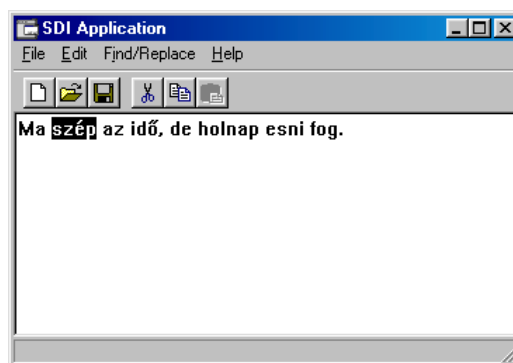
Szöveg a szerkesztőablakban



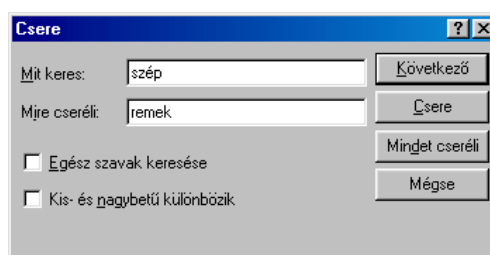
Keresés a *szép* szóra:



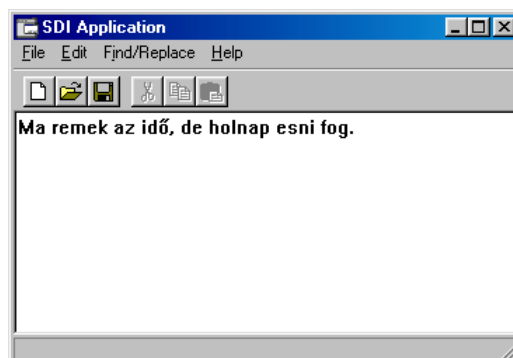
A megtalált szó kijelölése:



A *szép* szó *remek* szóra való cseréje:



A módosított szöveg a szerkesztőmezőben:





Készítsünk MDI-alkalmazást, amely kétféle grafikus gyermekablak megjelenítésére és kezelésére használható (*MDIApp*)

A megoldásban, amit az *MDIAPP* alkalmazásban valósítottunk meg, az egyik ablakot *firkapapírnak*, másik ablakot pedig *rajzlaponak* hívjuk. A firkapapíron az egeret lenyomott bal gombbal mozgatva vonalakat rajzolhatunk. A rajzlapon grafikus objektumokat – ellipszisszeletet, ellipsziscikket, ellipszist, poligont, téglalapot, lekerekített téglalapot és vonalat – jeleníthetünk meg előre beállított pozíciókban.

A grafikus ablakokat tartalmazó MDI-keretben az aktuális gyermekablaknak megfelelő menüt használjuk. A menüsor alatt eszközsor található, amelyen a gyermekablakok létrehozására vonatkozó gyorsítógombokat helyeztük el. A keretablak alsó részén állapotsor látható, amely magyarázatokat, illetve a firkapapír esetében egér-koordinátákat jelenít meg.

Ha a „*File/New/Projects/MDI Application*” választással készítjük el az MDI-alkalmazásunk vázát, akkor a keretablak (*MainForm*) modelljének típusa (***FormStyle***) *fsMDIForm* lesz. Ugyancsak létrejön egy gyermekablak-osztály (*TMDIChild*), melynek típusa *fsMDIChild*.

Ha megnézzük a „*Project/Options/Forms*” információkat, azt találjuk, hogy fenti két ablakosztály példányait nem hozza létre az alkalmazás automatikusan.

Ha további gyermekablak-osztályra van szükségünk, akkor azt a szokásos módon előállíthatjuk. A tervezés végén beállítjuk a form típusát *fsMDIChild*-ra és átvisszük a formot az automatikusan létrehozott ablakok közül (*Auto-create Forms*) a program során létrehozható ablakok közé (*Available Forms*) a *Project/Options* menüválasztáskor megjelenő párbeszédablak *Forms* lapján. A *Névjegyzék* (*AboutBox*) ablak automatikusan létrejön.

Az *MDIAPP.DPR* projekt-állomány **uses** sorában szereplő bejegyzések az ablakokra utalnak:

```
program Mdiapp;
uses
  Forms,
  Main in 'MAIN.PAS' {MainForm},
  Childwin in 'CHILDWIN.PAS' {MDIChild},
  About in 'ABOUT.PAS' {AboutBox},
  Childwin1 in 'Childwin1.pas' {MDIChild1};

{$R *.RES}

begin
  Application.CreateForm(TMainForm, MainForm);
  Application.CreateForm(TAboutBox, AboutBox);
  Application.Run;
end.
```

A *MAIN.PAS* állományban tárolt főablak-osztálya az alábbi szerkezetű:

```

type
  TMainForm = class(TForm)
    MainMenu1: TMainMenu;
    MainMenu2: TMainMenu;
    File1: TMenuItem;
    FileNewItem: TMenuItem;
    FileNewItem1: TMenuItem;
    FileCloseItem: TMenuItem;
    Window1: TMenuItem;
    Help1: TMenuItem;
    N1: TMenuItem;
    FileExitItem: TMenuItem;
    WindowCascadeItem: TMenuItem;
    WindowTileItem: TMenuItem;
    WindowArrangeItem: TMenuItem;
    HelpAboutItem: TMenuItem;
    WindowMinimizeItem: TMenuItem;
    SpeedPanel: TPanel;
    OpenBtn: TSpeedButton;
    ExitBtn: TSpeedButton;
    StatusBar: TStatusBar;
    SpeedButton1: TSpeedButton;
    MenuItem1: TMenuItem;
    MenuItem2: TMenuItem;
    MenuItem3: TMenuItem;
    MenuItem4: TMenuItem;
    MenuItem5: TMenuItem;
    MenuItem6: TMenuItem;
    window2: TMenuItem;
    MenuItem8: TMenuItem;
    MenuItem9: TMenuItem;
    MenuItem10: TMenuItem;
    MenuItem11: TMenuItem;
    MenuItem12: TMenuItem;
    MenuItem13: TMenuItem;
    Rajzok1: TMenuItem;
    Ellipsziisszelet1: TMenuItem;
    Ellipszis1: TMenuItem;
    Ellipsziscikk1: TMenuItem;
    Poligon1: TMenuItem;
    Tglalap1: TMenuItem;
    KerTglal: TMenuItem;
    Vonal1: TMenuItem;
    sszes1: TMenuItem;
    Semelyik1: TMenuItem;
    procedure FormCreate(Sender: TObject);
    procedure FileNewItemClick(Sender: TObject);
    procedure WindowCascadeItemClick(Sender: TObject);
    procedure UpdateMenuItems(Sender: TObject);
    procedure WindowTileItemClick(Sender: TObject);
    procedure WindowArrangeItemClick(Sender: TObject);
    procedure FileCloseItemClick(Sender: TObject);
    procedure FileExitItemClick(Sender: TObject);
    procedure WindowMinimizeItemClick(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure HelpAboutItemClick(Sender: TObject);
    procedure FormMouseMove(Sender: TObject; Shift: TShiftState;
      X, Y: Integer);
    procedure FileNewItem1Click(Sender: TObject);
    procedure Ellipsziisszelet1Click(Sender: TObject);
    procedure Ellipszis1Click(Sender: TObject);
    procedure Ellipsziscikk1Click(Sender: TObject);
    procedure Poligon1Click(Sender: TObject);
    procedure Tglalap1Click(Sender: TObject);
    procedure KerTglalClick(Sender: TObject);
    procedure Vonal1Click(Sender: TObject);
    procedure sszes1Click(Sender: TObject);
    procedure Semelyik1Click(Sender: TObject);
  private
    procedure CreateMDIChild(const Name: string);
    procedure CreateMDIChild1(const Name: string);
    procedure ShowHint(Sender: TObject);
end;

```

A főablak létrehozásakor intézkedünk a vezérlőkhöz kapcsolt buboréksúgó (*Hint*) megjelenítéséről, továbbá megteremtjük az ablakmenü megjelenésének lehetőségét.

```
procedure TMainForm.FormCreate(Sender: TObject);
begin
    //magyarázó szöveg engedélyezése
    Application.OnHint := ShowHint;
    //az ablakmenü
    Screen.OnActiveFormChange := UpdateMenuItems;
end;
```

A kapcsolódó eljárás a gyermekablakok számát használja a különböző ablakkezelő elemek engedélyezésére.

```
procedure TMainForm.UpdateMenuItems(Sender: TObject);
begin
    //zárás
    FileCloseItem.Enabled := MDIChildCount > 0;
    //átlapolás
    WindowCascadeItem.Enabled := MDIChildCount > 0;
    //egymás mellé rendezés
    WindowTileItem.Enabled := MDIChildCount > 0;
    //ikonok elrendezése
    WindowArrangeItem.Enabled := MDIChildCount > 0;
    //minimalizálás
    WindowMinimizeItem.Enabled := MDIChildCount > 0;
end;
```

#### *Az MDI menük funkciói*

Mindkét gyermekablakhoz készítünk menüt. A *MainMenu1* a firkapapírhoz, a *MainMenu2* pedig a rajzlaphoz tartozik. Mindkét menü "Fájl" menüpontja tartalmazza az "Új firkapapír", "Új rajzlap" létrehozására szolgáló menüpontokat ugyanúgy, mint az aktuális ablak bezárását és a programból való kilépést. Mindkét menünek van "Ablakok" főmenüpontja az ablakok elrendezésére (*Egymásra*, *Egymásmellé*, *Ikonok rendezése*, *Mind ikon*), valamint "Segít" fő-menüpontja a "Névjegy" megjelenítésére. A rajzlaphoz tartozó menü tartalmaz még egy "Rajzok" legördülő menüt a geometriai alakzatok (ellipszisszeletet, ellipsziscikket, ellipszist, poligont, téglalapot, lekerekített téglalapot és vonal) megjelenítésére és eltüntetésére.

Mindkét menü "Fájl/Új firkapapír" menüpontja, illetve az *OpenBtn* gyorsítógomb a *CreateMDIChild* metódust hívja egy névvel paraméterezve, ahol a név egy számlálóval egészül ki.

```
procedure TMainForm.FileNewItemClick(Sender: TObject);
begin
    CreateMDIChild('Új firkapapír' + IntToStr(MDIChildCount + 1));
end;
```

A hívott metódus:

```
procedure TMainForm.CreateMDIChild(const Name: string);
var
    Child: TMDIChild;
begin
    // új MDI firkapapír
    Child := TMDIChild.Create(Application);
    Child.Caption := Name;
end;
```

Ugyanez a helyzet a rajzlappal az "Új rajzlap" menüpont, illetve a *SpeedButton1* gyorsítógomb esetében.

```
procedure TMainForm.FileNewItem1Click(Sender: TObject);
begin
    CreateMDIChild1('Új rajzlap' + IntToStr(MDIChildCount + 1));
end;
```



A hívott metódus

```
procedure TMainForm.CreateMDIChild1(const Name: string);
var
    Child: TMDIChild1;
begin
    // új Rajzlap
    Child := TMDIChild1.Create(Application);
    Child.Caption := Name;
end;
```

A *Segít/Névjegy* menüpont aktivizálása mindkét menüben az *AboutBox* formot jeleníti meg modálisan:

```
procedure TMainForm.HelpAboutItemClick(Sender: TObject);
begin
    AboutBox.ShowModal;
end;
```

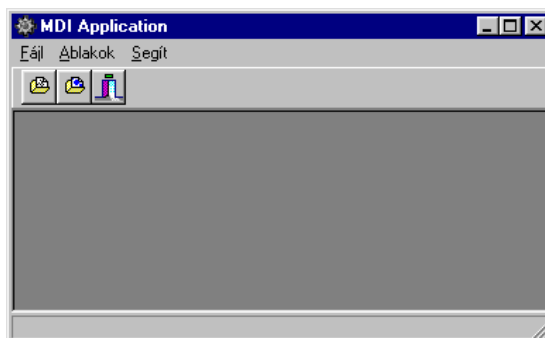
Mindkét menüpont "*Bezár*" funkciója az aktuális gyermekablakot zárja, ha van ilyen:

```
procedure TMainForm.FileCloseItemClick(Sender: TObject);
begin
    if ActiveMDIChild <> nil then ActiveMDIChild.Close;
end;
```

Mindkét menü "*Fájl/Kilép*" menüpontja, illetve az *ExitBtn* gyorsítógomb a főablakra aktivizálja a *Close* metódust, melynek hatásmechanizmusát a bevezetőben már ismertettük.

```
procedure TMainForm.FileExitItemClick(Sender: TObject);
begin
    Close;
end;
```

A program menürendszere:



Az *Ablakok* menü funkciói:

```
procedure TMainForm.WindowCascadeItemClick(Sender: TObject);
begin
    Cascade;
end;

procedure TMainForm.WindowTileItemClick(Sender: TObject);
begin
    Tile;
end;

procedure TMainForm.WindowArrangeItemClick(Sender: TObject);
begin
    ArrangeIcons;
end;

procedure TMainForm.WindowMinimizeItemClick(Sender: TObject);
var
    I: Integer;
begin
    //Visszafelé haladva a gyermekablakokon
    for I := MDIChildCount - 1 downto 0 do
        MDIChildren[I].WindowState := wsMinimized;
    end;
```

## A gyermekablakok működése

A firkapapír gyermekablak osztályát a *CHILDDWIN.PAS* modul tartalmazza:

```
type
  TMDIChild = class(TForm)
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormMouseMove(Sender: TObject; Shift: TShiftState;
      X, Y: Integer);
    procedure FormMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure FormActivate(Sender: TObject);
  end;
```

Az ablak megjelenésekor be kell állítanunk, hogy az aktuális menü a *MainMenu1* legyen, és az ablaklista a *Window1* menü alatt szerepeljen. Ennek megfelelően az aktivizáláskor meghívott eljárás

```
procedure TMDIChild.FormActivate(Sender: TObject);
begin
  Mainform.Menu:=Mainform.MainMenu1;
  Mainform.Windowmenu:=Mainform.Window1;
end;
```

Ha a firkapapíron mozgatjuk az egeret, akkor az állapotsorban az egér koordinátái jelennek meg. Ha lenyomjuk a bal oldali egérgombot, akkor vonalat kezdünk húzni, és húzzuk egészen addig, míg fel nem engedjük az egér gombját.

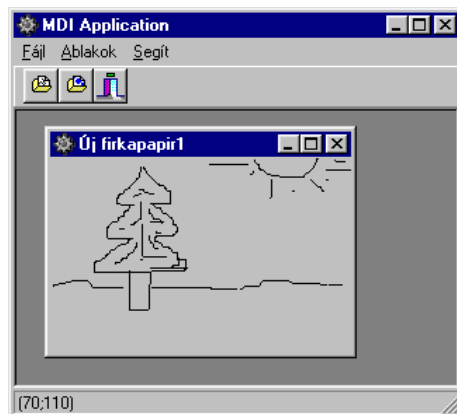
```
procedure TMDIChild.FormMouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
begin
  //koordináták megjelenítés
  Mainform.Statusbar.SimpleText:=
    '('+inttostr(x)+';'+inttostr(y)+' ');
  //vonalhúzás
  if ssLeft in Shift then TForm(Sender).Canvas.Lineto(x,y);
end;

procedure TMDIChild.FormMouseDown(Sender: TObject;
  Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  //vonalhúzás kezdete
  TForm(Sender).Canvas.Moveto(x,y);
end;
```

A firkapapír ablak bezárásakor csak az erőforrások felszabadításáról kell intézkedünk.

```
procedure TMDIChild.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  Action := caFree;
end;
```

Rajzolás a firkapapír ablakban:



Másképpen működik a rajzlap, melynek osztályát *CHILDWINI.PAS* állomány tartalmazza:

```
type
  TMDIChild1 = class(TMDIChild)
    procedure FormActivate(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  public
    chord:boolean;
    ellipse:boolean;
    pie:boolean;
    polygon:boolean;
    rectangle:boolean;
    roundrect:boolean;
    line:boolean;
  end;
```

A rajzlap aktivizáláskor szintén megváltoztatjuk a keretablak menüjét. (A rajzlapon menüből alakzatokat rajzolhatunk). Annak nyilvántartására, hogy melyik alakzat látszik, a *TMDIChild1* osztályban nyilvános elérésű adatmezőket definiáltunk, melyek alapállapotát (nincs kirajzolt alakzat) a form létrehozásakor állítjuk be:

```
procedure TMDIChild1.FormCreate(Sender: TObject);
begin
  chord:=false;
  ellipse:=false;
  pie:=false;
  polygon:=false;
  rectangle:=false;
  roundrect:=false;
  line:=false;
  inherited;
end;
```

A rajzlap aktivizálásakor nem csak a menüváltásról kell gondoskodnunk (a *MainMenu2* az aktuális menü és az ablaklista a *Window2* alatt található), hanem a nyilvántartásunkhoz kapcsolódó menüpontokat is ki kell pipálni.

```
procedure TMDIChild1.FormActivate(Sender: TObject);
begin
  // menü beállítás
  Mainform.Menu:=Mainform.MainMenu2;
  Mainform.Windowmenu:=Mainform.Window2;
  // menüpontok kipipálása
  Mainform.Ellipszis1.Checked:=TMDIChild1(Sender).Chord;
  Mainform.Ellipszis1.Checked:=TMDIChild1(Sender).Ellipse;
  Mainform.Ellipsziscikk1.Checked:=TMDIChild1(Sender).Pie;
  Mainform.Polygon1.Checked:=TMDIChild1(Sender).Polygon;
  Mainform.Tglalap1.Checked:=TMDIChild1(Sender).Rectangle;
  Mainform.KerTglal1.Checked:=TMDIChild1(Sender).Roundrect;
  Mainform.Vonal1.Checked:=TMDIChild1(Sender).Line;
end;
```

A nyilvántartásunknak megfelelően történik a rajzlap kifestése.

```
procedure TMDIChild1.FormPaint(Sender: TObject);
var points:array [0..5] of TPoint;
    ppoints:array [0..11] of TPoint;
begin
    if( chord) then
        begin
            // ellipszis és egyenes által határolt terület
            // paraméterek: az ellipszist befoglaló téglalap koordinátái
            // egyenes két pontja /2x2 db/
            Canvas.Chord( 10, 10, 180, 140, 12, 12, 128, 135);
        end;
    if(ellipse) then
        begin
            // ellipszis
            // paraméterek: az ellipszist befoglaló téglalap koordinátái /4 db/
            Canvas.Ellipse( 30, 20, 200, 400);
        end;
    if( pie) then
        begin
            // ellipsziscikk
            // paraméterek: az ellipszist befoglaló téglalap
            // koordinátái /4 db/
            // az ellipszis középpontjából induló két egyenes
            // egy - egy pontjának koordinátái /2x2 db/
            Canvas.Pie( 150, 100, 300, 140, 10, 40, 100, 10);
        end;
    if( polygon) then
        begin
            // sokszög
            // paraméterek: a csúcspontokat tartalmazó tömb
            // csúcspontok száma (tömb mérete)
            points[ 0].x:= 5;
            points[ 0].y:= 5;
            points[ 1].x:= 20;
            points[ 1].y:= 5;
            points[ 2].x:= 20;
            points[ 2].y:= 20;
            points[ 3].x:= 10;
            points[ 3].y:= 20;
            points[ 4].x:= 5;
            points[ 4].y:= 10;
            points[ 5].x:= 5;
            points[ 5].y:= 5;
            Canvas.Polygon( points);
        end;
    if ( rectangle) then
        begin
            // téglalap
            // paraméterek: koordinátái (bal felső sarok (x,y),
            // jobb alsó sarok)
            Canvas.Rectangle( 60, 160, 400, 180);
        end;
    if( roundrect) then
        begin
            // kerekített sarkú téglalap
            // paraméterek: befoglaló téglalap koordinátái /4 db/
            // sarkokat kerekítő ellipszisek hosszúsága
            // és magassága /2 db/
            Canvas.RoundRect( 200, 10, 300, 110, 80, 30);
        end;

    if( line) then
        begin
            // vonalhúzás
            // az aktuális pozíció kezdőpontba mozgatása
            Canvas.MoveTo( 75, 20);
            // megadott pontig (végpont) vonalhúzás
            Canvas.LineTo( 195, 56);
        end;
    inherited;
end;
```

Természetesen a főprogramban az egyes menüpontok kiválasztásakor az állapot-tulajdonságok állításáról és az újrafestésről is gondoskodnunk kell. Példaként tekintsük az ellipsziscikkhez tartozó programrészlet!

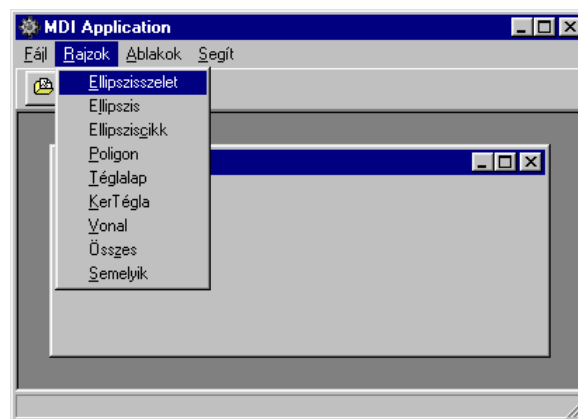
```
procedure TMainForm.Ellipszisszelet1Click(Sender: TObject);
begin
    TMDIChild1(Mainform.ActiveMdiChild).Chord:=
        not(TMDIChild1(Mainform.ActiveMdiChild).Chord);
    Mainform.Ellipszisszelet1.Checked:=
        TMDIChild1(Mainform.ActiveMdiChild).Chord;
    TMDIChild1(Mainform.ActiveMdiChild).Refresh;
end;
```

Rajzlap esetén a menüben még annak a lehetősége is szerepel, hogy az összes rajzelemet megjelenítsük, illetve eltüntessük. Természetes, hogy a fenti menüválasztásokat kezelő eljárások az állapottulajdonságokat megfelelő módon állítják.

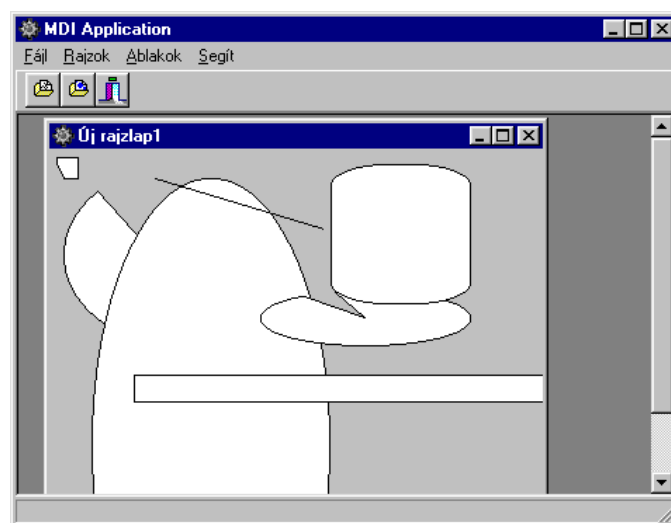
A *Rajzlap* bezárásakor arról is gondoskodunk, hogy a menü alapállapotba (*MainMenu1*) kerüljön.

```
procedure TMDIChild1.FormClose(Sender: TObject;
                                var Action: TCloseAction);
begin
    Mainform.Menu:=Mainform.MainMenu1;
    Mainform.Windowmenu:=Mainform.Window1;
    Action := caFree;
end;
```

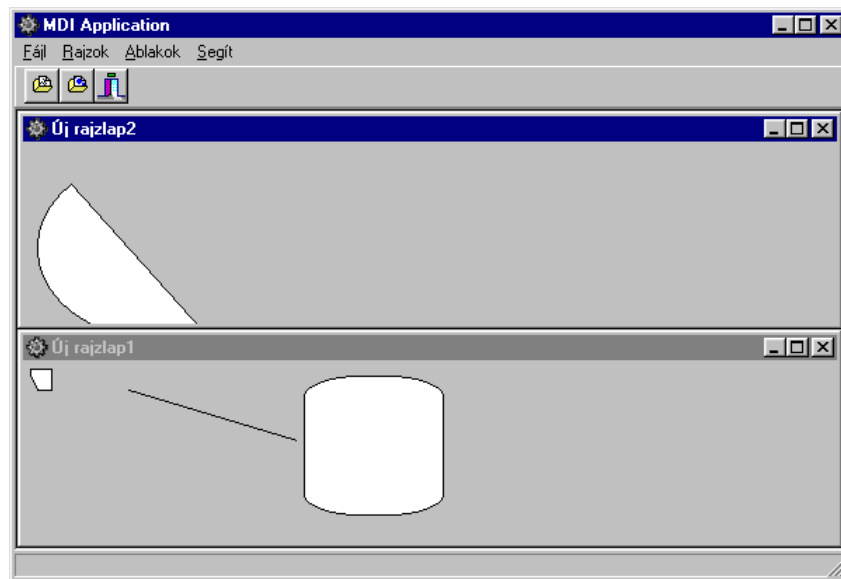
A *Rajzok* menü menüpontjai:



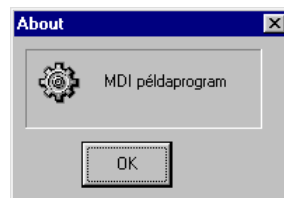
Az *Összes* menüpont kiválasztása az összes alakzatot kirajzolja:



Az ablakok egymásmellé rendezése:



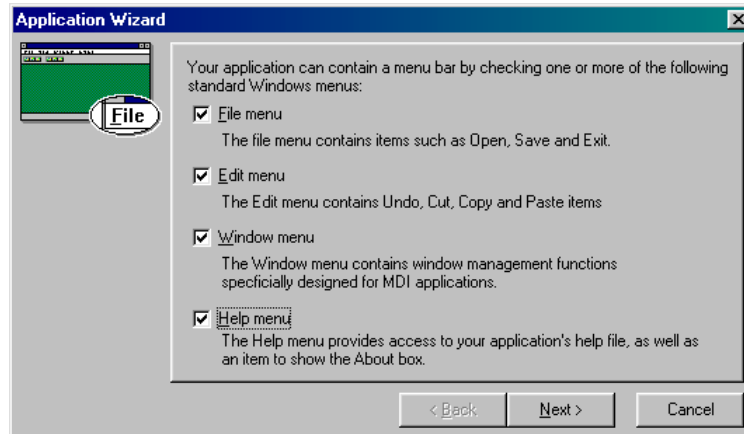
Az *About* ablak:



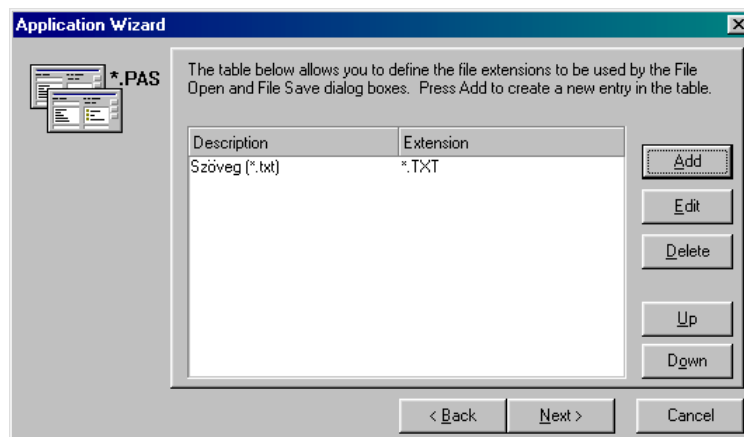


Az „Application Wizard” varázslót a „File/New” menüpont választása után a *Projects* lapon aktivizálhatjuk.

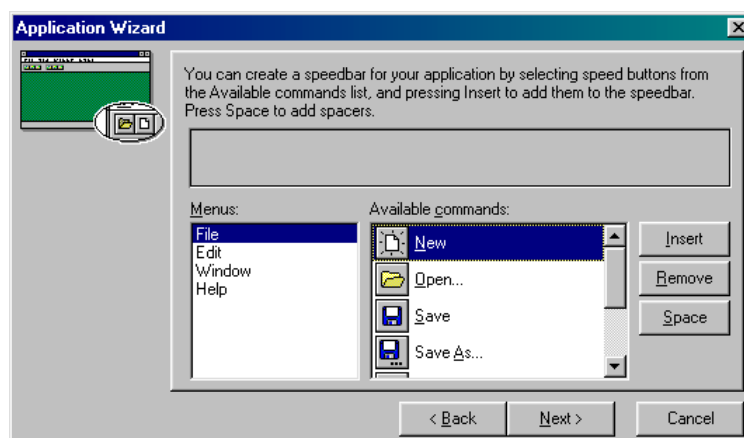
Első lépésben döntenünk kell arról milyen legyen a menüszerkezet:



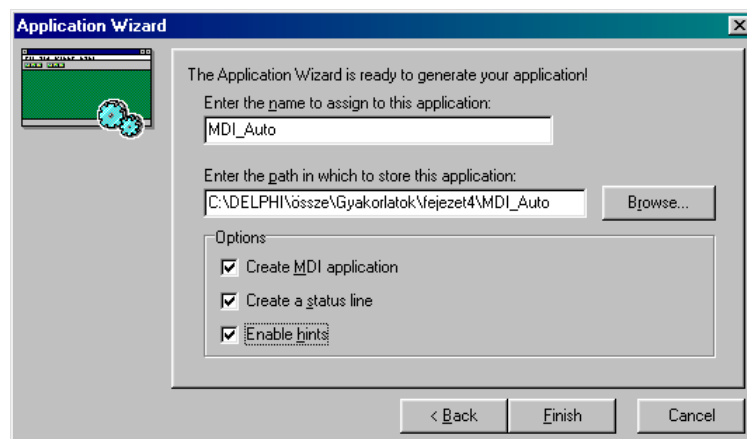
Az alkalmazás használja a közös fájlkezelési párbeszédablakokat, ezért meg kell adnunk állományaink kiterjesztését.



Megadhatjuk, hogy mely elemeket tesszük fel az eszközsorra:



Ezek után meg kell adnunk az alkalmazás nevét és a projekt könyvtárát. Intézkedhetünk arról is, hogy MDI alkalmazás készüljön, legyen állapotsor és buboréksúgó.



Az elkészült MDI-váz alkalmazás futás közben:

