

6.1. Dinamikusan szerkeszthető könyvtárak (DLL)

1. *Dll* egész számok összeadására [Dllmin](#)
2. Dátumbekérő párbeszédablak *Dll*-ben [DatumDLL](#)
3. Többnyelvű alkalmazás erőforrásai *DLL*-ben [NyelvEsz](#)



Első lépésként előállítjuk a dinamikusan szerkeszthető könyvtárat. Ehhez a **File/New/DLL** választás után keletkező projektet *DLLMIN.DPR* néven lemezen tároljuk. Ezt követően a létrejövő üres könyvtármodulban (**library**) elhelyezzük az összeadó függvény kódját. Ahhoz, hogy a függvényünk más alkalmazásból is elérhető legyen, a függvényt szabványos Windows alatti hívási konvenciók szerint kell deklarálnunk (**stdcall**), továbbá a függvény nevét el kell helyoznünk az **exports** részben:

```
library DLLMIN;
uses
  SysUtils,
  Classes;

function Osszead(A, B : integer):integer; stdcall;
begin
  result:=a+b;
end;

exports
  Osszead;

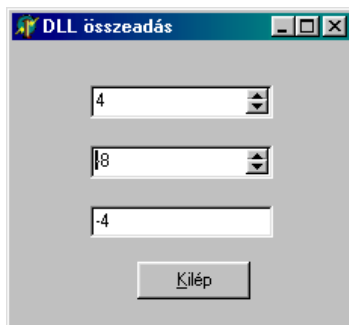
begin
  Beep;
end.
```

A DLL-modul önmagában nem futtatható, így csak fordítani tudjuk a **Project/Build ...** menüpont kiválasztásával. A fordítás eredménye a *DLLMIN.DLL* állomány.

A következőkben a *DLLMIN* könyvtárat használó alkalmazást készítjük el. A szokásos programfejlesztési lépéseken túlmenően egyetlen teendőnk marad, az *Osszead* függvény megfelelő deklarációjának elkészítése:

```
function Osszead(A, B: integer):integer; stdcall;
external 'DLLMIN.DLL';
```

A *Teszt* alkalmazás ablaka a szokásos vezérlőelemeket tartalmazza:



Ha *DLLMIN.DLL* nem található, akkor a program indításakor hibajelzést kapunk. Miután elkészült a tesztelő alkalmazás, azt a DLL projektjéhez rendelhetjük a **Run/Parameters/Host Application** menüválasztással:

Ha az összerendelés után a **Run/Run** menüpontot választjuk, akkor a Delphi először felépíti a DLL-modult, majd elindítja a megadott gazdaalkalmazást. Így a DLL-ben végzett módosítások eredményét azonnal tesztelhetjük.



A feladat megoldásával bemutatjuk, hogyan lehet Delphi formot *DLL*-ből megjeleníteni, és ezáltal más rendszerekbe átvinni. (A *Datum.DPR* a *DLL* projektje, a *DatumTst.DPR* pedig a *DATUM.DLL* könyvtárat tesztelő alkalmazás.)

A dinamikusan szerkeszthető könyvtár előállításánál az előző ([Dllmin](#)) feladat lépéseéhez hasonlóan járunk el, azzal a különbséggel, hogy most egy form-modult is csatolunk a projekthez (**File/New Form**).

A *DATUM.DPR* állomány most csak deklarációkat tartalmaz:

```
library Datum;
uses
  DatumDlg in 'DatumDlg.pas' {DatumFrm};

exports
  GetDatum;

{$R *.res}
begin
end.
```

A *DatumDlg* modulban tárolt formra helyezzük a már jól ismert építőelemeket és a *Samples* komponenslapon található **Calendar** (naptár) komponenst. A kiválasztott hónap napjainak megjelenítését a naptárvezérlő végzi, míg a hónapot és az évet kombinált ablakból választhatjuk ki. Az elmondottaknak megfelelően az ablakosztály és az ablakobjektum:

```
type
  TDatumFrm = class(TForm)
    Naptar: TCalendar;
    OKBtn: TBitBtn;
    CancelBtn: TBitBtn;
    Bevell: TBevel;
    Honapok: TComboBox;
    Evok: TComboBox;
    procedure FormCreate(Sender: TObject);
    procedure HonapokChange(Sender: TObject);
    procedure EvokChange(Sender: TObject);
  public
    function DatumString: ShortString;
  end;

var
  DatumFrm: TDatumFrm;
```

A *DLL*-könyvtárat használó alkalmazás a *GetDatum* eljárás hívásával veszi fel a kapcsolatot a modullal. A *GetDatum* eljárás létrehozza, megjeleníti majd megszünteti a párbeszédablakot, továbbá az *SDatum* paraméterben visszaadja a kiválasztott dátumot tartalmazó sztringet:

```
procedure GetDatum(HLeiro: THandle; var SDatum: ShortString);Stdcall;
begin
  Application.Handle := HLeiro;
  DatumFrm := TDatumFrm.Create(Application);
  if DatumFrm.ShowModal <> mrCancel then
    SDatum := DatumFrm.DatumString
  else
    SDatum := '0000.00.00.';
  DatumFrm.Free;
end;
```

A párbeszédablak létrehozásakor feltöltjük a hónapok nevét és az éveket tartalmazó szerkesztő-listaablakokat, majd kiválasztottá tesszük a mai dátumnak megfelelő hónap, illetve év elemeket. A hónapok nevét egy globális sztringtömbben adjuk meg:

```

var
  HonapNevek : Array[1..12] of ShortString =
    ('Január', 'Február', 'Március', 'Április',
     'Május', 'Június', 'Július', 'Augusztus',
     'Szeptember', 'Október', 'November', 'December');

procedure TDatumFrm.FormCreate(Sender: TObject);
var
  i: Integer;
begin
  for i := 1 to 12 do
    Honapok.Items.Add(HonapNevek[i]);
  Honapok.ItemIndex := Naptar.Month - 1;
  for i := 1900 to 2100 do
    Evek.Items.Add(IntToStr(i));
  Evek.ItemIndex := Evek.Items.IndexOf(IntToStr(Naptar.Year));
end;

```

A hónap és az év megváltoztatása esetén a naptárvezérlőt aktualizáljuk a változásnak megfelelően:

```

procedure TDatumFrm.HonapokChange(Sender: TObject);
begin
  Naptar.Month := Honapok.ItemIndex + 1;
end;

procedure TDatumFrm.EvekChange(Sender: TObject);
begin
  Naptar.Year := StrToInt(Evek.Items.Strings[Evek.ItemIndex]);
end;

```

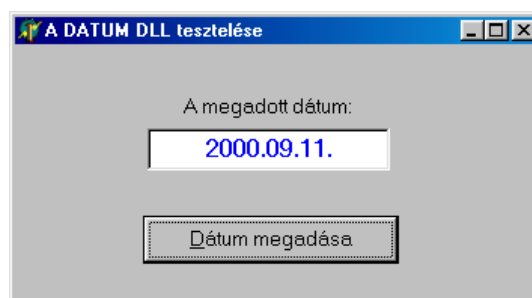
Végezetül nézzük azt a metódust, amit a dátum *év.hó.nap.* formátumú előállításához használunk!

```

function TDatumFrm.DatumString: ShortString;
var tar : ShortString;
begin
  with Naptar do
  begin
    Result := IntToStr(Year) + '.';
    Tar := '0'+IntToStr(Month);
    Result := result + Copy(Tar, Length(Tar)-1,2) + '.';
    Tar := '0'+IntToStr(Day);
    Result := result + Copy(Tar, Length(Tar)-1,2) + '.';
  end;
end;

```

Az elkészített *DLL* használatát a *DatumTst* alkalmazás egyszerű vezérlőelemek segítségével mutatja be:



A nyomógomb megnyomásakor megjelenik a *DATUM.DLL* könyvtárban definiált párbeszédablak:

V	H	K	Sze	Cs	P	Szo
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

A tesztelő alkalmazásból csak a *GetDatum* eljárás deklarációjára és hívására hívjuk fel a figyelmet:

```
procedure GetDatum(HLeiro: THandle; var SDatum: ShortString);stdcall;
external 'DATUM.DLL' name 'GetDatum';

procedure TForm1.DatumBtnClick(Sender: TObject);
var
  Datum: ShortString;
begin
  GetDatum(Handle, Datum);
  Edit1.Text:='          '+Datum;
end;
```



Alakítsuk át a fejezetben található többnyelvű alkalmazást oly módon, hogy a különböző nyelvek esetén eltérő háttérképeket töltsön be az erőforrás DLL modulokból! (*NyelvEsz*)

Első lépésben kiegészítjük az erőforrás állományokat a magyar és az angol zászló képével.

A *hun.rc* erőforrás-állomány:

```
// SzstringTabla

STRINGTABLE DISCARDABLE
BEGIN
    1            "Számolás"
    2            "Kilépés"
    3            "Adatbevitel"
    4            "Eredmény"
END

// Bitkep
5 Bitmap "HUN.BMP"
```

Az *eng.rc* erőforrásfájl:

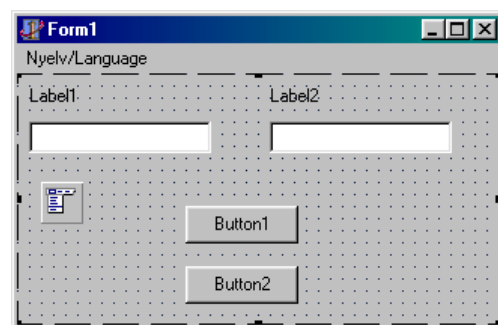
```
// Szstringtabla

STRINGTABLE DISCARDABLE
BEGIN
    1            "Calculation"
    2            "Exit"
    3            "Datainput"
    4            "Result"
END

// Bitkep
5 Bitmap "ENG.BMP"
```

Elhelyezzük a formon az *Image1* vezérlőt a zászlók megjelenítésére, továbbá a *Label1* és *Label2* vezérlőket áttetszőre állítjuk (*Transparent=true*), hogy ne zavarja a szürke háttér a látványt.

```
type
    TForm1 = class(TForm)
        Edit1: TEdit;
        Edit2: TEdit;
        Label1: TLabel;
        Button1: TButton;
        Label2: TLabel;
        Button2: TButton;
        MainMenu1: TMainMenu;
        NyelvLanguage1: TMenuItem;
        MagyarHungarian1: TMenuItem;
        EnglishAngol1: TMenuItem;
        Image1: TImage;
        procedure MagyarHungarian1Click(Sender: TObject);
        procedure EnglishAngol1Click(Sender: TObject);
        procedure Button1Click(Sender: TObject);
        procedure Button2Click(Sender: TObject);
        procedure FormCreate(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;
```



Beépítjük a programba a bitkép DLL-erőforrásból való feltöltésére szolgáló *LoadDLLBmp* eljárást az alábbiak szerint.

```
// Bitkép betöltése erőforrás DLL-ből
procedure LoadDLLBmp(Lib:THandle; BmpId:Integer; Dest:TBitmap);
resourcestring
  Err1 = 'Nem betölthető %s (%d)';
  Err2 = 'Az #%d erőforrás nem található (%s)';
  Err3 = 'A #%d bitkép hibás formátumú';
var
  rInfo,hMemory: THandle;
  rSize: Longint;
  pData: PByte;
  BmpHdr: TBitmapFileHeader;
  stream: TMemoryStream;
begin
  if Lib < HINSTANCE_ERROR then Exit;
  // A bitkép megkeresése a könyvtárban
  rInfo := FindResource(Lib, MakeIntResource(BmpId), rt_Bitmap);
  if rInfo = 0 then
    raise Exception.CreateFmt(Err2, [BmpId, 'FindResource']);
  // A bitkép betöltése globális memóriaterületre
  hMemory := LoadResource(Lib,rInfo);
  try
    if hMemory = 0 then
      raise Exception.CreateFmt(Err2, [BmpId, 'LoadResource']);
    // A globális memóriablokk rögzítése, a mutatóval történő
    // eléréshez
    pData := LockResource(hMemory);
    try
      // A bitkép mérete
      rSize := SizeofResource(Lib,rInfo);
      if rSize = 0 then
        raise Exception.CreateFmt(Err3, [BmpId]);
      // A bitképet memóriafolyamba másoljuk
      stream := TMemoryStream.Create;
      try
        // Bitkép fájl típusjel: BM
        BmpHdr.bfType := $4D42;
        stream.SetSize(sizeof(BmpHdr)+rSize);
        stream.Write(BmpHdr,sizeof(BmpHdr));
        stream.Write(pData^,rSize);
        stream.Seek(0,0);
        Dest.LoadFromStream(stream);
      finally
        stream.Free;
      end;
    finally
      UnlockResource(hMemory);
    end;
  finally
    FreeResource(hMemory);
  end;
end;
```

A *LoadResources* eljárást átalakítjuk úgy, hogy a bitképet is betöltse:

```
// A szövegek és a bitkép betöltése az erőforrásból
procedure LoadResources(lib:string);
var
  DLLleiro : THandle;
  Puffer : array[0..100] of char;
begin
  DLLleiro:=LoadLibrary(PChar(lib));
  if DLLleiro<32 then
    begin
      beep; exit;
    end;
  LoadDLLBmp(DLLleiro, 5, Form1.Image1.Picture.Bitmap);
  LoadString(DLLleiro, 1, Puffer, Sizeof(puffer));
  Form1.Button1.Caption:=Puffer;
  LoadString(DLLleiro, 2, Puffer, Sizeof(puffer));
  Form1.Button2.Caption:=Puffer;
```

```

LoadString(DLLLeiro, 3, Puffer, Sizeof(puffer));
Form1.Label1.Caption:=Puffer;
LoadString(DLLLeiro, 3, Puffer, Sizeof(puffer));
Form1.Label1.Caption:=Puffer;
LoadString(DLLLeiro, 4, Puffer, Sizeof(puffer));
Form1.Label2.Caption:=Puffer;
FreeLibrary(DllLeiro);
end;

```

A program magyarul indul:

```

procedure TForm1.FormCreate(Sender: TObject);
begin
    LoadResources('HUN.DLL');
end;

```

Az összes többi programrészt a fejezet példájából változatlanul használjuk.

```

// Magyar beállítások
procedure TForm1.MagyarHungarian1Click(Sender: TObject);
begin
    LoadResources('HUN.DLL');
end;

// Angol beállítások
procedure TForm1.EnglishAngol1Click(Sender: TObject);
begin
    LoadResources('ENG.DLL');
end;

// Számítás
procedure TForm1.Button1Click(Sender: TObject);
begin
    Edit2.text:=inttostr(sqr(strToInt(Edit1.text)));
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
    Close;
end;

```

