

Kezdő lépések a Lazarus környezetben

Az alábbiakban erősen szájbarágós stílusban (bocsánat!) szedjük pontokba a Lazarus környezetben történő programfejlesztés lépéseit.

Mivel az egyes komponensek property-i abc-sorrendben találhatók az Object Inspector-ban, ezért ebben a sorrendben szerepel ezek beállítása.

1. Lazarus megnyitás (File, New..., Application)
2. A Form elnevezése a feladathoz:
 - 2.1. Caption = "feladattcím"
 - 2.2. Name = a form-azonosító = feladattcim ékezetlenül!!!
 - 2.3. Position = poDesktopCenter (a képernyő közepén)
3. A mentés (először)
 - 3.1. Save/Save as/Save all (bármelyike)
 - 3.2. A Save párbeszédablakában beállítjuk a könyvtárat
 - 3.3. A Unit mentéséhez elfogadhatjuk a felajánlott: Unit1.pas nevet
 - 3.4. A projekthez --célszerű-- a feladathoz illeszkedő nevet választani (de el kell térjen a form-azonosítótól!)
4. Egy címke (label) a formra helyezése a tetején
 - 4.1. A Label kiválasztása Standard komponensek közül
 - 4.2. A formon teljes szélességben a tetejéhez közel elhelyezni
 - 4.3. Align = alTop (a címke helye: a form tetején)
 - 4.4. Alignment = taCenter (a szöveg igazítása a címkében: középre)
 - 4.5. Caption = a címke szövege
 - 4.6. Font = a megfelelő font, méret és stílus kiválasztása
 - 4.7. Height = az a magasság érték, amely mellett látszódik a szöveg (általában az egérrel jól be lett állítva)
 - 4.8. Left = a form bal szélétől vett távolság; ha teljes szélességben kitölti, akkor 0
 - 4.9. Name = a címke azonosító = lbCim (ékezetmentes, funkcióra utaló címke-azonosító)
 - 4.10. Top és Width = 0 és a form.Top-pal azonos érték (ha teljes szélességben kitölti)
5. Egy kilépésre szolgáló (Vége-) gomb elhelyezése a formon
 - 5.1. A Button kiválasztása Standard komponensek közül
 - 5.2. Caption = "&Vége" (az "&" a egyfajta forróbillentyűk kiválasztására való: ALT+V)
 - 5.3. Font = ha kell, értelemszerűen
 - 5.4. Height = ha kell, értelemszerűen (célszerű könnyen megjegyezhető egész értéket választani)
 - 5.5. Hint = ha kell a megjelenő, sárga súgószöveg, lehet
 - 5.6. Left = a bal szélének a helye (célszerű könnyen megjegyezhető egész értéket választani)
 - 5.7. Name = btVege (ékezetmentes, funkcióra utaló gomb-azonosító)
 - 5.8. ShowHint = True (látszódjék a súgószöveg)
 - 5.9. Top = a tetejének a helye (célszerű könnyen megjegyezhető egész értéket választani)
 - 5.10. Width = célszerű könnyen megjegyezhető egész értéket választani
 - 5.11. A gomblenyomás (OnClick) esemény programozása:
 - 5.11.1. Az Object Inspector-ban az Events lap kiválasztása

Első lépés Lazarus-ban

5.11.2. Az OnClick esemény szövegére kattintani, majd a megjelenő "..." gombra

5.11.3. (A "kinyílt" Unit1.pas-ban) a kurzor jelezte helyen az eseménykezelő eljárás törzsét kitöltjük:

"Close" (az ablak bezárása)

6. Minden mentése (File, Save all)

7. Fordítás, futtatások (Run, Run = F9); próbálkozások;

8. Majd az alkalmazás megállása után a tovább lépéshez: a form előcsalogatása (F12)

... innentől a folytatás igen sokféle ...

Apróságok

- a komponenseket lehet (általában értelmes is) másolni; de természetesen a másolat-komponens jellemzőin (property-in) érdemes egyenként végigmenni, és ha kell, igazítani (az azonosítóját majdnem biztos, hogy kell)
- a komponenseket egymásmellé és -alá illesztését segítik az illeszkedéskor megjelenő színes vonalak
- a formon levő, (Tab-bal) kiválasztható komponensek „bejárási” sorrendjét beállíthatjuk rákattintva egyre, jobb egérgomb: „Tab order...”
- ha egy komponenst nem akarunk Tab-bal kiválaszthatóvá tenni, akkor a TabStop-ja legyen False

- A függvényekről:

- Általában:

- Összetett értéktípus megengedett

Például:

```
Type TKar=record fo:Char; al:Char end;
Function CharToKar(const c:Char):TKar;
Function IntToKar(const k:longint):TKar;
```

○

- Konverziós függvények:

Val	Str	
IntToStr/StrToInt	Int64ToStr/StrToInt64	FloatToStr/StrToFloat

- String-függvények/operátorok

.[.]	+	copy
length	pos	

- A tömbökről:

Létezik dinamikusan létrehozható tömb is. Az alábbi példa sokmindent elárul róla:

```
Var t:array of string;
...
SetLength(t,N); //létrejön: 'array [0..N-1] of string' tömb
For i:=0 to Length(t)-1 do
Begin
    t[i]:=...
End;
...
```

- beírhatatlannak tűnő jelek (pl. „[”, „{”) beilleszthetők: Edit, Insert from Character Map
- a string típusra a Turbo Pascal hossz-korlátozása (<256) nem érvényes!

A polimorfiairól

Hasonlóan a Borland Pascal-hoz –sajnos– a saját függvények és eljárások azonos névvel elnevezése problematikus lehet, ui. nem igyekszik a fordító a paraméterek típusából meg-

határozni, hogy vajon melyikről lehet szó. Amikor az egyes típusokhoz külön-külön unit tartozik, akkor a unit neve használható minősítőként.

Például:

```
unit Szoveg;
...
type
  TSzoveg = record
    hossz:Integer;
    jelek:array [1..MaxHossz] of TKar;
  end;
  const   Ures:TSzoveg=(hossz:0 {jelek: tetszőleges});
  function Vegere(const s:TSzoveg; const k:TKar):TSzoveg;
  function Elejere(const s:TSzoveg; const k:TKar):TSzoveg;
  function Hossz (const s:TSzoveg):Integer;
...
unit Bekezd;
...
Uses
  ... Szoveg;
type
  TBekezdes = TSzoveg;
  var{const} Ures:TBekezdes;
  procedure Vegere(var b:TBekezdes; const sz,ej:TSzoveg);
  procedure ElsoSzo(var b:TBekezdes; var sz,ej:TSzoveg);
  function Hossz(const b:TBekezdes):Integer;
...
... a
  Vegere(bek,elSzo,elvalaszt)
hivatkozást hibásnak véli, pedig a paraméterei miatt egyértelműen kiderül, hogy melyikről van szó; javítási
lehetőség:
  Bekezd.Vegere(bek,elSzo,elvalaszt) ...
```

A unit-okról

A Borland Pascal-hoz hasonlóan használható. Itt is igaz, hogy a unit-nak van (lehet) inicializáló része, amely az alkalmazás elindulásakor egyszer végrehajtódik. Tehát alkalmas bizonyos (a unit által megvalósított típussal kapcsolatos) inicializáló tevékenység megszervezésére. Bár, amennyiben a unit egy form unit-ja, akkor célszerűbb az ablak születésének vagy megjelenésének (onCreate vagy onActivate) eseményéhez kapcsolni az inicializáló lépéseket.

Néhány jellegzetes komponensről

StringGrid-hez

- sg.1. Additional komponensek között van
- sg.2. ColCount/RowCount = oszlopok/sorok száma (2/9)
- sg.3. jobb egérgomb a formon, a griden: StringGrid Editor
 - sg.3.1. az oszlopok méretét beállíthatjuk
 - sg.3.2. beleírhatunk a megjelenő cellákba (pl. a fejsorokba)
- sg.4. cellahivatkozás: sg.cells[col,row], ahol col/row=0..ColCount-1/RowCount-1
- sg.5. a cellák szerkeszthetőségéhez: options.goEditing = True

Megjegyzések:

- A StringGrid alkalmas sorozat be- és kiviteli szerkezet, bár ...
- Üres (0 sorból álló) StringGrid már létezik (0.9.16-tól), a 0.9.8-ban üres sorozatot nem lehetett vele ábrázolni
- sg.Clear = teljesen törli a StringGrid-t (csak a külső keret marad látszó)
- sg.Clean = a cellák tartalmát törli (sorok/oszlopok száma nem változik)

UpDownButton-hoz

- udB.1. Common Controls komponensek között található
- udB.2. Increment a gombok lenyomásakor változás „egysége”
- udB.3. Min/Max az értékváltozás két határa
- udB.4. Position az éppen aktuális értéke (érdemes szinkronizálni egy edit mezővel, ha mutatni v. állítani is akarjuk az értéket, l. az Associate property-t)

Megjegyzések:

- a formon való húzogatóskor ügyelni kell arra, hogy „szét ne essen elemeire”; lehet külön állítani az egyik és a másik gombot

Menükhöz

- mn.1. Standard komponensek között van (egy ablakon egyet lehet használni, bár többet is ráhúzhatók a formra...)
- mn.2. célszerű a form bal felső sarkához húzni (bár akárhova is tesszük a „szokásos” helyen jelenik meg)
- mn.3. Name = mnFomenu
- mn.4. Items kiválasztásakor elindul a menüszerkesztő:
 - mn.4.1. A „New items”-re jobb egérgombbal kattintva választhatok a helyi menüből; pl. „Create submenu” = almenüket; „... after/before” = mögé/eléilleszteni újabb (fő)menüket
 - mn.4.2. a megfelelő menüt kijelölve az Object Inspector-ban hangolhatom:
 - mn.4.2.1. Caption = ... a megjelenő menüsöveg ...
 - mn.4.2.2. Name = ... a menüazonosító ...
 - mn.4.2.3. Enabled = ... alapállapotban kiválasztható legye-e ...
 - mn.4.2.4. OnClick = a kiválasztásakor bekövetkező esemény

Megjegyzések:

- Az Enabled property-t kell menet közben állítanunk, a szerint, hogy van-e értelme az adott pillanatban az egyes menük kiválasztásának
- Gyakori, hogy fájlmegnyitás, ill. –mentés funkciókat rendelünk az OnClick eseményekhez (lásd az Open file dialógus-nál...)

Open/Save file dialógus-hoz

- osd.1. Dialogs komponensek között található; és lehetővé teszi, hogy a megfelelő céllal megnyissunk egy fájlt
- osd.2. hasonlóan járunk el velük, mint a menüvel, mivel nem az ablakon jelennek meg, hanem külön kérésre, önálló ablakokként
- osd.3. Name = odBe (ha Open dialog), vagy sdKi (ha Save as dialog)
- osd.4. Title = „Fájl megnyitása”, vagy „Fájl mentése mint...”
- osd.5. egy tipikus eseménykezelő, amely a fájlmegetnyitást kezdeményezi:

```

procedure TForm1.mnBeClick(Sender: TObject);
var
    beFN:String;
begin
    if odBe.execute then
    begin
        beFN:=odBe.FileName;
        ShowMessage('A '+beFn+' fájl megnyitása sikerült.');
        (* ... *)
        mnKi.Enabled:=True;
    end
    else
    begin
        ShowMessage('A fájlmegetnyitás sikertelen.');
    end; //if odBe
end; //mnBeClick
    
```

Memo-hoz

- mm.1. Standard komponensek egyike
- mm.2. Font = értelemszerűen beállítható betűtípus/-jellemzők; ilyen módon jelenik meg benne a szöveg
- mm.3. Lines = a létrehozásakor ezt a szöveget fogja tartalmazni; szerkesztésekor a String Editor aktivizálódik
- mm.4. Name = memo-azonosító
- mm.5. ReadOnly = True, ha a felhasználó nem módosíthatja
- mm.6. ScrollBars = gördítő sávok megjelenjenek-e

Megjegyzések:

- TextFile-okkal való közvetlen kapcsolatot biztosítják a következő metódusok:
 - mmMemo.Lines.LoadFromFile(beFN);
 - mmMemo.Lines.SaveToFile(kiFN)
- Sorok kezelése:
 - mmMemo.Lines.Count = a sorainak a száma
 - mmMemo.Lines[0..mmMemo.LinesCount-1] = a memo sorai (CrLf nélkül)
 - mmMemo.Text = a teljes, memo-beli szöveg, a sorok végén CrLf-fel¹ (ez **nem property**!)
 - OnKeyPress eseményre reagálva megfigyelhető, hogy billentyű lenyomásakor éled az esemény, mégpedig akkor, amikor még a jel nem került a „helyére”, az eseménykezelés után kerül bele a jel; további érdekesség (hiba), hogy amikor végén

¹ Ebben a sorrendben: #13#10

kerül bele az CrLf, akkor létrejön egy újabb sor (ami ugyan üres), de a Lines.Count nem változik!

PageControl (fülesLapvezérlő² és lapok)-hoz

- pc.1. A Common Controls gyűjtemény eleme
- pc.2. A Form-on a füzetre kattintva a jobb egérgombbal megfelelő számú lapot adunk hozzá (Add page)
- pc.3. Name = fülesLapvezérlő azonosítója, pl. pcFulesLapVezerlo
- pc.4. Az egyes lapokra végrehajtjuk (legalább) a következőket:
 - tab.4.1. A szerkesztendő lap kiválasztása: a Form füzetére jobb egérgombbal kattintva a Show page ... menüponttal
 - tab.4.2. Caption = a fülön megjelenő címke
 - tab.4.3. Name = lapazonosító (pl. tabBekezd)
- Megjegyzés: a form-on nem lehet komponenseket másolás-beillesztéssel létrehozni, mert a másolat a form-ra, s nem a lapra kerül!
- pc.5. ActivePage = az elinduláskor elől lévő lap azonosítója

PopUp menühöz

- pu.1. A Standard komponens-gyűjteményben található
- pu.2. Kiválasztva a form-ra húzzuk egy alkalmas helyre (az ablakban nem lesz látható, csak a megfelelő komponensre jobb egérgombbal kattintva jelenik meg...) a menühöz hasonlóan készíthető el a PopUp menü is, így csak visszautalunk
- pu.3. Lásd mn.3. (pl. Name = puBetolt)
- pu.4. Lásd mn.4.
- pu.5. A komponens –amelyre kattintva szeretnék, hogy előbukkanjon– PopupMenu = popup menü azonosítója, amit pu.3.-ban állítottunk be

² A PageControl-t jobb híján: „füles lapvezérlő”-nek nevezem, mivel a fülekkel ellátott lapokat vezérli.

Kódolási „technikák”

Tájékoztató külön ablakban

- ta.0. a tájékoztató menüjét vagy gombját létrehozzuk
- ta.1. új form létrehozás: File+New Form
- ta.2. property-ket beállítjuk; Name=fmTajek; ... Visible=False
- ta.3. elhelyezzük rajta a megjelenítést végző komponenst (pl. memo-t)
- ta.4. fordítás előtt kimentjük pl. Tajekozato.pas néven a hozzátartozó unit-ot
- ta.5. összekapcsoljuk a fő form-ot és a tájékoztató form-ot
 - ta.5.1. átlépünk a fő form-ra (pl. Windows+xxxx, xxxx=a fő form neve)
 - ta.5.2. a fő form-on felvesszük e unit-ot is (a Uses sorban)
 - ta.5.3. a tájékoztatót aktivizáló komponens adott eseményét lekezelő eljárásába:
fmTajek.Visible:=True
 - ta.5.4. megszervezzük a tájékoztató fájl belekerülését az ablakba
 - ta.5.5. megszervezzük a visszalépést a fő ablakba

Textfájl-kezelés-hez

- tf.1. nagyban emlékeztet a Turbo Pascal textfájl kezelésére
- tf.2. Deklaráció: f:TextFile
- tf.3. Fájl-megnyitások: AssignFile(f,fN); Reset(f)/Rewrite(f);
- tf.4. Olvasás/írás: Read(f,...)/Readln(f,...)/Write(f,...)/Writeln(f,...);
- tf.5. Fájl-lezárás: CloseFile(f);

Az absztrakt felsorolási típus I/O-jának a megvalósításához

- lx.1. ötlet: ListBox, amely a Standard komponensek egyike
- lx.2. Items = itt írandók be a felsorolás konstansai, soronként egy-egy
- lx.3. Name = a felsorolásdoboz azonosítója (pl. lxNap)
- lx.4. futás közbeni kijelölés: ItemIndex:=melyik (melyik = a felsorolási értékek egyikének indexe [0..Items.Count-1])
- lx.5. futás közbeni kijelölés lekérdezése: Selected[i] = Igaz , ha az i. ki van jelölve

Például:

A felsorolási típus: **Type** TNap=(Hetfo,Kedd,...) ;
 Az Object Inspectorban beállítva: Items = HétfőKedd..., ahol CrLf

```

procedure TForm.lxNapClick(Sender: TObject);
(*
    A (TNap felsoroláshoz tartozó) lxNap egy elemére kattintáskori
    esemény; n:TNap
    Uf: n = amelyik elemre lett kattintva
*)
    var i:Integer;
begin
    i:=0;
    while (i<lxNap.Items.Count) and not lxNap.Selected[i] do Inc(i);
    n:=TNap(i);
end;//lxNapClick
    
```


Első lépés Lazarus-ban

```
procedure beallit(Const n:TNap);  
(*  
    Uf: n értékének megfelelő listaelem fókuszbba állítása (kijelölése)  
*)  
begin  
    form.lxNap.ItemIndex:=Ord(n)  
end;//beallit
```