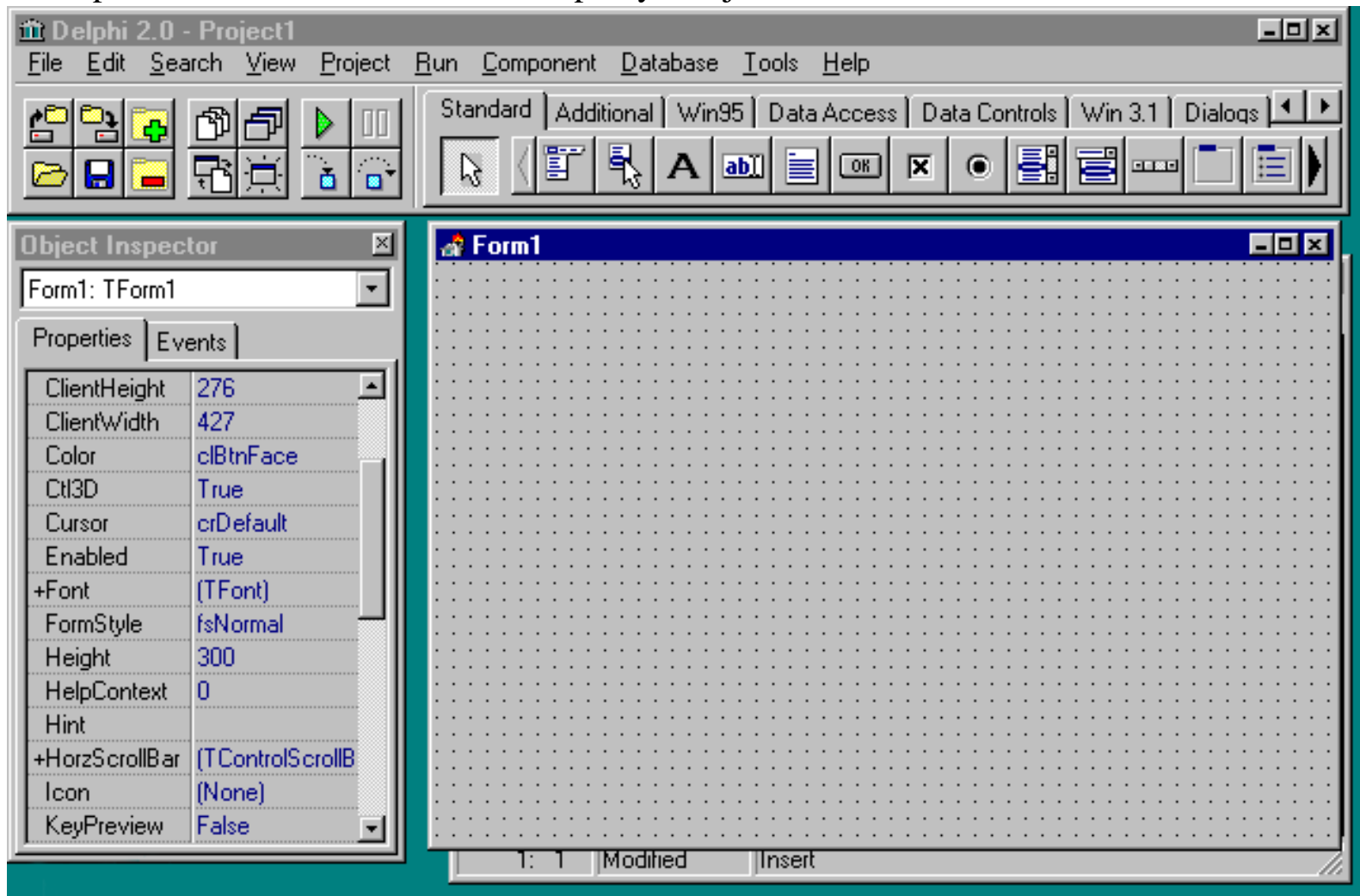
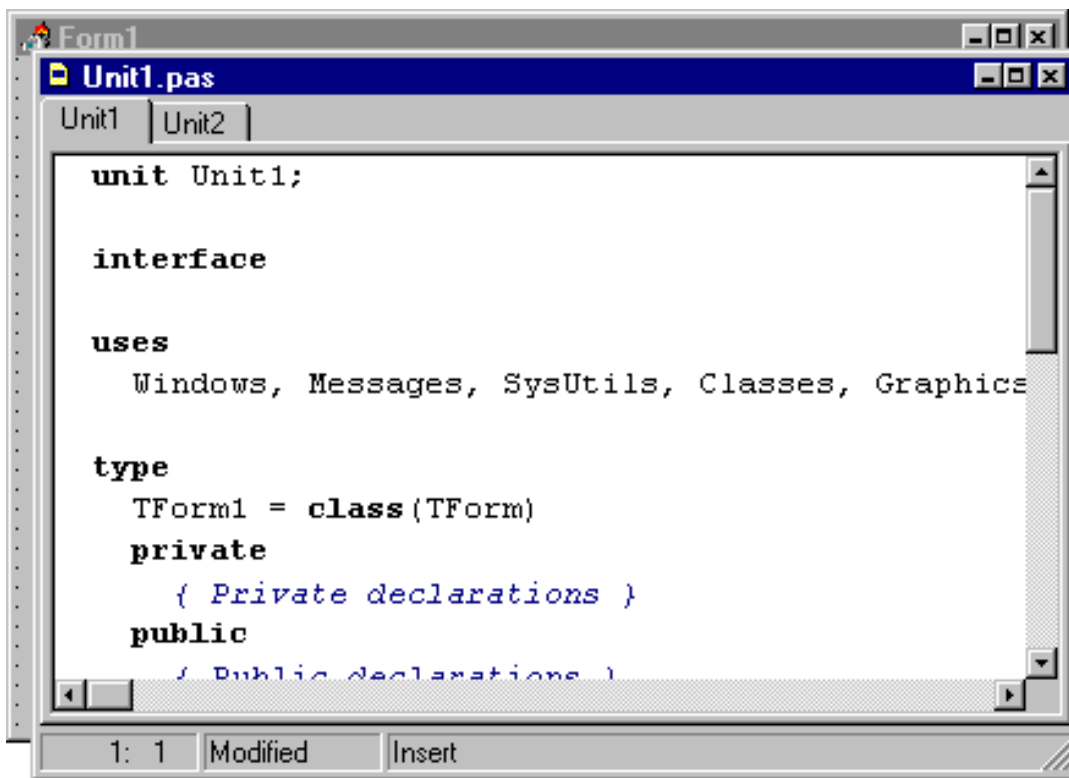


1. A Delphi megjelenése

A Delphi elindítása után a következő képernyővel jelentkezhet be:



A Delphi fejlesztőkörnyezet négy fő területből áll: a *fejrészből*, ami a menüt, az eszközöket és a komponenseket tartalmazza, a bal oldalt látható *objektumfelügyelőből* (*Object Inspector*), a *formszerkesztő* ablakból és a formszerkesztő által részben eltakart *kódszerkesztőből*. Utóbbi úgy válik láthatóvá, ha a formszerkesztő alól "kilógó" részére rákattintunk.

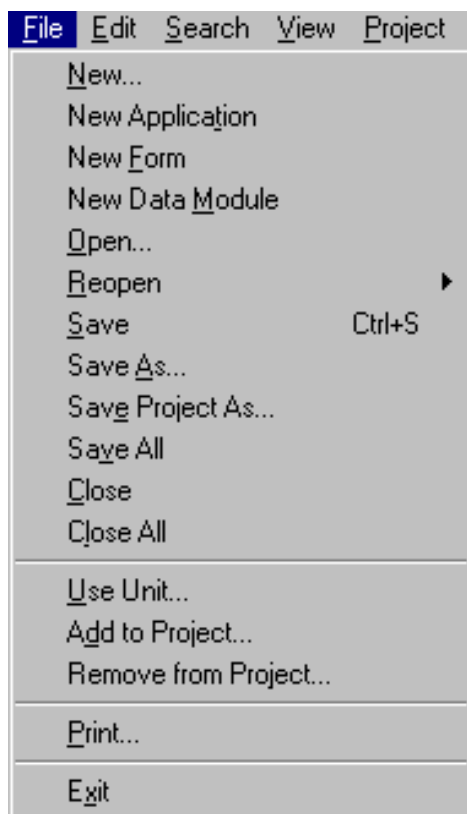


1.1. A Delphi munkaterület fejrésze

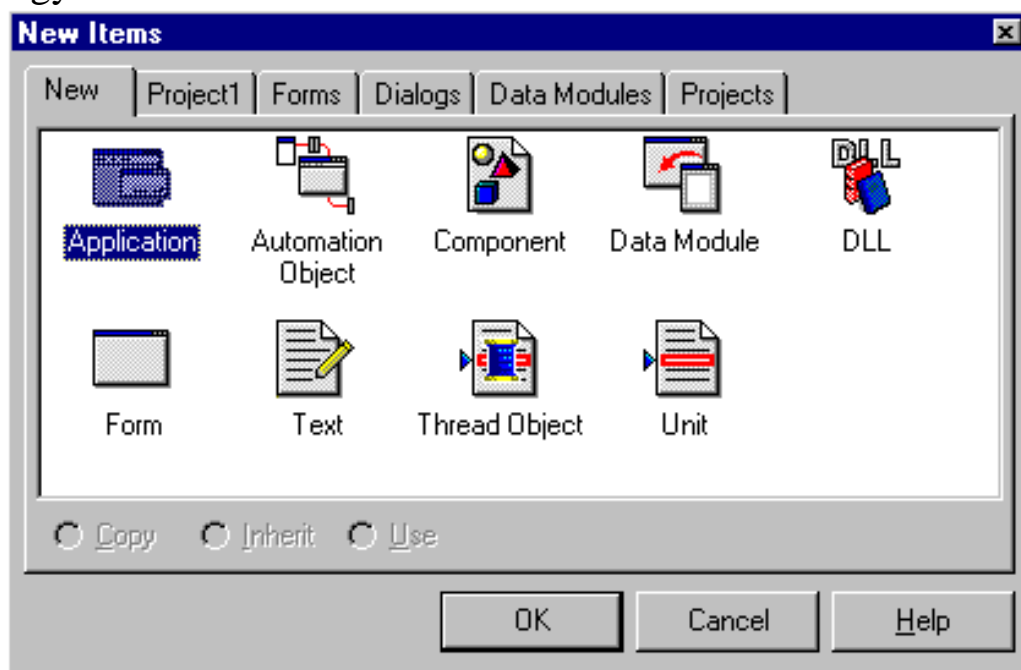
1.1.1. A menü

Bármilyen száraznak is tûnhet a következõ rész, a menük többé-kevésbé részletes ismertetése kikerülhetetlen. Elõzetesen csak annyit kell megemlíteni, hogy a Delphi menüje is helyzetérzékeny, azaz esetenként a legördülõ menülista tételei attól is függnnek, hogy éppen milyen mûveletet végezünk.

A File menü



- **New...:** A menüpont kiválasztásakor egy összefoglaló jellegű, többlapos párbeszéd-ablak jelenik meg (**New Items**), ahonnan új projektet, vagy más elemeket választhatunk. Általában van egyszerűbb módszer is...

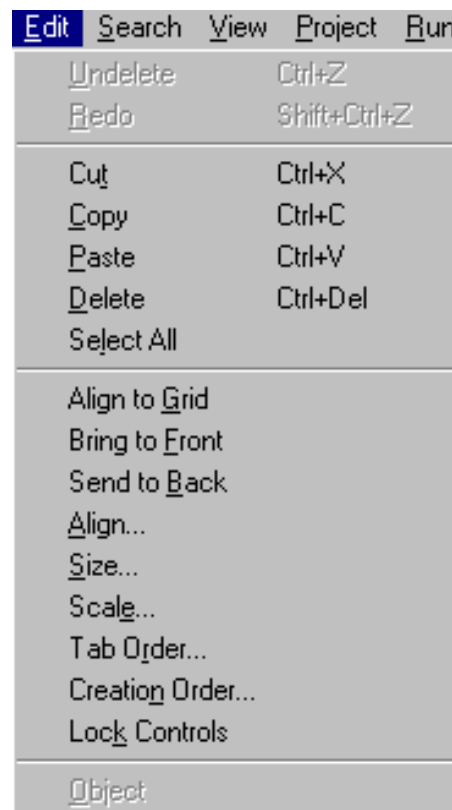


A *New* lapon új projektet nyithatunk meg, vagy pl. formot vagy unitot adhatunk a projekthez. A *Project1* lap a már kész formok átalakítását, belőlük új formok előállítását tesz lehetővé. A *Dialogs* lapról néhány típus párbeszéd-ablak készítéséhez nyújt segítséget. A *Data Modules* lapon adatkezelő formok készítéséhez kapunk segítséget, végül a *Projects* oldal a több dokumentumból (MDI) és az egy dokumentumból (SDI)

álló felületet tartalmazó projekthez nyújt segítséget.

- **New Application:** Innen egyszerûbben nyithatunk új projektet.
- **New Form:** Új form hozzáadása.
- **New Data Module:** Adatbázis jellegû formot kezdhethünk építeni.
- **Open...:** Létező unitot vagy formot nyit meg, amivel egyúttal a projektünkhöz hozzá is adhatjuk.
- **Reopen:** A legutóbb használt néhány projekt ill. fájl nyitható meg innen.
- **Save:** Elmenti az aktuális fájlt.
- **Save As...:** Másik név alatt menthetjük el a fájlt.
- **Save Project As...:** A projektet új néven menti el.
- **Save All:** A projekthez tartozó összes fájlt elmenti.
- **Close:** Bezárja az aktuális fájlt. Ha még nem volt elmentve, rákérdez a mentésre.
- **Close All:** Az összes megnyitott fájlt bezárja. A mentetlenekre rákérdez.
- **Use Unit:** A unitok kezelését könnyíti meg (ha több unitból áll a projekt).
- **Add To Project:** Formot vagy unitot adhatunk a projektünkhöz, ...
- **Remove From Project:** ... ezzel meg eltávolíthatjuk.
- **Print:** Nyomtatás.
- **Exit:** A Delphiből való kilépés egyik lehetősége.

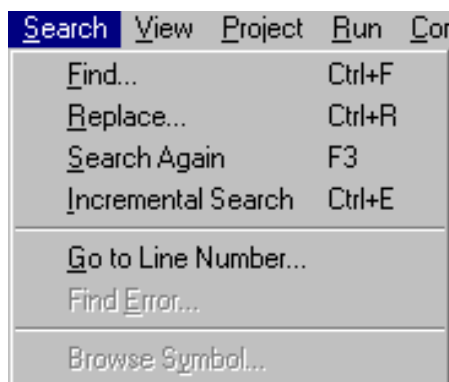
Az Edit menü



- **Undelete vagy Undo és Redo:** A szokásos visszavonás-visszaállítás pár.

- **Cut:** Kivágás.
- **Copy:** Másolás.
- **Paste:** Beillesztés.
- **Delete:** Törlés.
- **Select All:** A szövegszerkesztő ablak minden sorát kijelöli (formszerkesztőben minden elemet).
- **Align to Grid:** Rács, amelynek segítségével pontosan helyezhetők el a komponensek a formokon.
- **Bring to Front:** A kijelölt objektumot a többi fölé helyezi, ...
- **Send to Back:** ... ez pedig alá.
- **Align:** Párbeszédablak segíti a precíz komponens-elhelyezést.
- **Size:** A kijelölt objektum pontos méreteit állíthatjuk be.
- **Scale:** A komponens nagyíthatóságáról lehet intézkedni.
- **Tab Order:** Egy formon azt lehet beállítani, hogy a **Tab** billentyűvel milyen sorrendben lépkedjük végig a komponenseken.
- **Creation Order:** A nem vizuális komponensek létrehozása.
- **Lock Controls:** A form elemeinek rögzítésére való.
- **Object:** A formra helyezett OLE objektum szerkesztése.

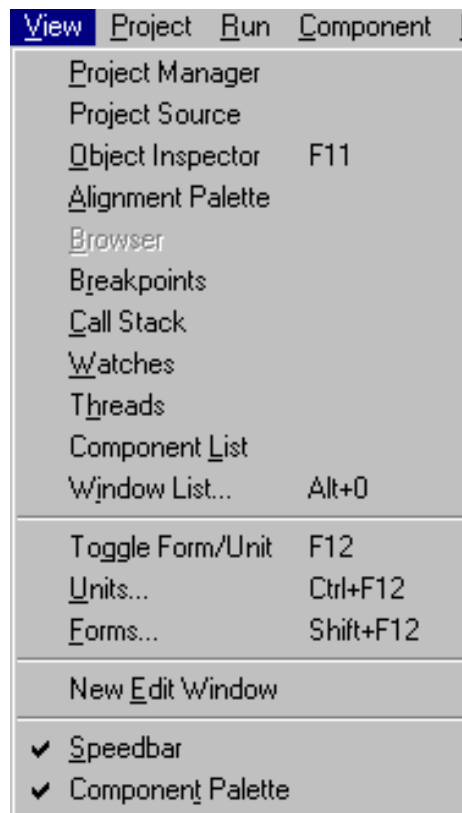
A Search menü



- **Find...:** Szöveget kereshetünk vele. A megnyíló párbeszédablakban megadható a szöveg, megadható, hogy megkülönböztesse-e a kis- és nagybetűket (*Case sensitive*), vagy hogy csak teljes szavakat kell-e keresni (*Whole words only*). Megadható a keresés tartománya, azaz hogy a teljes szövegben, vagy csak a kijelölt blokkban kell-e keresni (*Global, Selected*), a keresés iránya előre, vagy vissza (*Forward, Backward*) történjen stb.
- **Replace...:** Szöveget cserélhetünk, a *Text to find* sorba a keresendő, a *Replace with* sorba a csereszöveget kell beírni. A keresési funkciónál megismert beállítási lehetőségek itt is megvannak. A *Replace all* bekapcsolásával valamennyi előfordulást automatikusan lecseréli.
- **Search Again:** A keresési, illetve csere művelet ismételhető.
- **Incremental Search:** Ez is keresés, de begépelés közben a keresett szöveghez ugrik.

- **Go to Line Number...:** A megadott számú sorra ugrik a kurzor.
- **Find Error:** A fordítás vagy futtatás utáni hiba helyére ugrik.
- **Browse Symbol:** Szimbólumot kereshetünk (csak fordítás után használható).

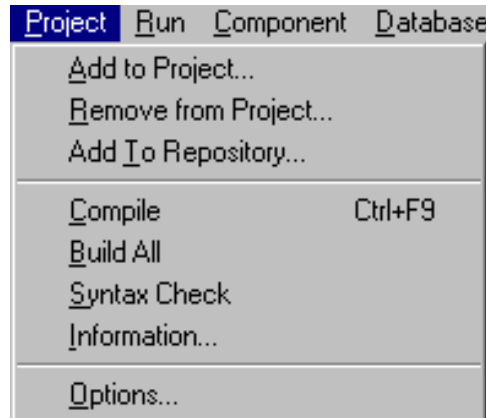
A View menü



- **Project Manager:** A Project Manager ablak megjelenítésére való.
- **Project Source:** A kódszerkesztő ablakot hozza be, ahol a projekt forráskódú szövegét nézhetjük meg.
- **Object Inspector:** Az objektumfelügyelő ablakot jeleníti meg.
- **Alignment Palette:** A megnyitott ablakban a kijelölt elemeket egyszerűen rendezhetjük el.
- **Browser:** Csak lefordított program esetén él. Osztályok, metódusok hivatkozásait, hatáskörét tekinthetjük meg.
- **Breakpoints:** Töréspontokat helyezhetünk el a programban, illetve ezeket tekinthetjük meg. A programfutás ellenőrzésére használjuk.
- **Call Stack:** Hibakeresésnél használjuk, az eljáráshívások ellenőrzésére.
- **Watches:** A programfutás ellenőrzésekor bizonyos változók értékét kísérhetjük figyelemmel.
- **Threads:** Másodlagos folyamat megjelenítése.
- **Component List:** Komponenseket helyezhetünk el a formon. A komponens palettáról egyszerűbb...
- **Window List:** A Delphiben megnyitott ablakok megjelenítésére való.

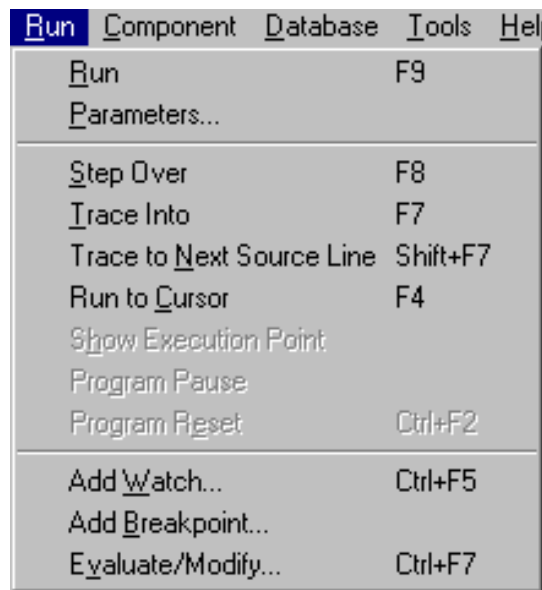
- **Toggle Form/Unit:** A form és a unit között kapcsolgathatunk.
- **Units:** A projektben lévő unitok jelennek meg.
- **Forms:** A projektben lévő formok jelennek meg.
- **New Edit Window:** Egy új szövegszerkesztő (kódszerkesztő) ablak nyílik meg, aminek felhasználásával a forráskódú szöveg különböző részeit szerkeszthetjük.
- **Speedbar:** Az eszközpalettát tehetjük láthatóvá () vagy tüntethetjük el.
- **Component Palette:** A komponenspalettát tehetjük láthatóvá vagy tüntethetjük el.

A Project menü



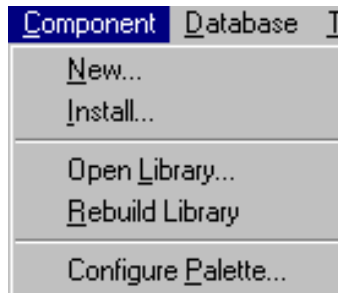
- **Add to Project...:** A File menüben is megtalálható, ...
- **Remove from Project...:** ... miként ez is.
- **Add to Repository...:** Ha olyan formot készítettünk, amit máskor is fel akarunk használni, felvehetjük a kelléktárba.
- **Compile:** A módosított forráskód lefordítására való.
- **Build All:** A projekt összes unitját lefordítja.
- **Syntax Check:** Ellenőrzi a kód szintaktikáját. Futtatás, de fordítás előtt is érdemes elvégezni.
- **Information...:** A lefordított fájlról szolgáltat adatokat.
- **Options...:** Többoldalas párbeszéd-ablakban lehet különböző beállításokat tenni.

A Run menü



- **Run:** Futtatja a programot. Ha a program szövege megváltozott, akkor előbb automatikus fordításra és szerkesztésre kerül sor (jegyezzük meg az **F9** billentyűt!).
- **Parameters:** Futási paraméterek beállítására való.
- **Step Over:** A programfutás követésére való. Szubrutinokba nem megy be (**F8**).
- **Trace Into:** A programfutás soronkénti követésére való. Szubrutinok hívásakor a szubrutinok utasításain is végigmegy a program. Itt is fontos az **F7** funkcióbillentyű.
- **Trace to Next Source Line:** Lépés a következő programsorra.
- **Run to Cursor:** Nem fut végig a program, hanem csak az aktuális kurzorpozícióig (**F4**).
- **Show Execution Point:** Megmutatja a következő programsort.
- **Program Pause:** Felfüggesztés.
- **Program Reset:** Befejezi a lépésenkénti futtatást.
- **Add Watch...:** Egy figyelendő változót lehet beírni, a programfutás során figyelemmel kísérhető az érték alakulása.
- **Add Breakpoint...:** Töréspontokat lehet beszúrni a programba.
- **Evaluate/Modify...:** A programfutás során keletkezett változó értékekre lehet rákérdezni az *Expression* mezőben, az értékek a *Result* mezőben jelennek meg.

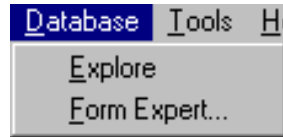
A Component menü



- **New...:** Új komponenseket készíthetünk (már ha szükséges).

- **Install...:** Kész, vásárolt komponenseket telepíthetünk a Delphi komponenskönyvtárába.
- **Open Library...:** Új komponenskönyvtárat hozhatunk létre.
- **Rebuild Library:** A komponenseken végrehajtott módosításokat rögzíthetjük.
- **Configure Palette...:** A komponenspaletta testreszabása.

A Database menü



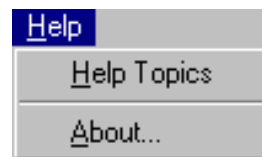
- **Explore:** Betölti a Database Explorert, amely lehetővé teszi az adatbázis-struktúrákban való keresést.
- **Form expert...:** Adatbázis beviteli képernyők készítését segíti.

A Tools menü



- **Options...:** Többlapos párbeszéd-ablakban a Delphi különböző beállításait lehet elvégezni.
- **Repository...:** Az *Add to Repository* paranccsal létrehozott kelléktárhoz való hozzáférés.
- **Tools...:** A Delphi-hez adott eszközök elérése.
- **Image Editor:** Az egyik ilyen eszköz lehet pl az *Image Editor* képszerkesztő-rajzoló, ...
- **Database Desktop:** ... a másik pedig a *Database Desktop*.

A Help menü



- **Help Topics:** A *Tartalom*, *Tárgymutató* és *Keresés* nevű lapokból álló segítség.
- **About...:** A minden programhoz járó *Névjegy* Delphi változata.








Egyáltalán nem fontos, de érdekes: ha az *About* ablakot megnyitottuk és az **Alt** gomb lenyomott állapotában beírjuk: *team*, akkor megtekinthetjük a Delphi létrehozásában közreműködő csapat névsorát. Ha hasonló módon a *developers* szót írjuk be, akkor a fejlesztők neveit olvashatjuk.

1.1.2. Az eszközpalletta

Látható, hogy a Delphi 2.0 ablakai a Windows95-nél megszokott jellemzőkkel bírnak: címsor, ablakkezelő gombok, vezérlőmenü stb. A Delphi címsora a **Delphi 2.0 - Project1** feliratot viseli (ez a felirat azonban a különböző állapotoknak megfelelően - pl. mentés után - változhat!). A menü 11 tételt tartalmaz, amelyekből szabványos legördülő menük jelennek meg. Az itt található parancsok gyakrabban használható része a menüsor alatt, baloldalon található eszközpallettán is megtalálható ikonként. Ezek a gombsorok természetesen testreszabhatók, azaz olyan gombokkal egészíthetők ki, amiket munkánk során gyakran használunk, a kevésbé használatosak pedig eltávolíthatók. A gombok funkciójáról súgócímké ad tájékoztatást, azaz ha valamelyikre ráállunk az egérkurzorral, egy keretben lévő felirat mondja meg, mire is való. Az alapbeállítású nyomógombokat mutatjuk be a következőkben:

	Open project	Megnyit egy létező projektet.
	Save all	Elmenti a projektet.
	Add file to project	Hozzácsatol a projekthez egy már meglévő formot vagy unitot.
	Open file	Megnyit egy formot vagy unitot.
	Save file	Elment egy kiválasztott form vagy unit fájlt.
	Remove file from project	Eltávolítja a kiválasztott formot vagy unitot a projektből.
	Select unit from list	Megnyitja a unit-megjelenítő párbeszéd-ablakot.
	Select form from list	Megnyitja a form-megjelenítő párbeszéd-ablakot.
	Toggle form/unit	A fókuszt a formról a unitra helyezi át.

	New form	Új formot ad a projekthez.
	Run	Futtatja a projektet.
	Pause	Megállítja a projekt futtatását.
	Trace into	Soronkénti futtás.
	Step over	Ez is soronkénti futtatás, de az eljárásokba nem megy bele.

1.1.3. A komponenspaletta

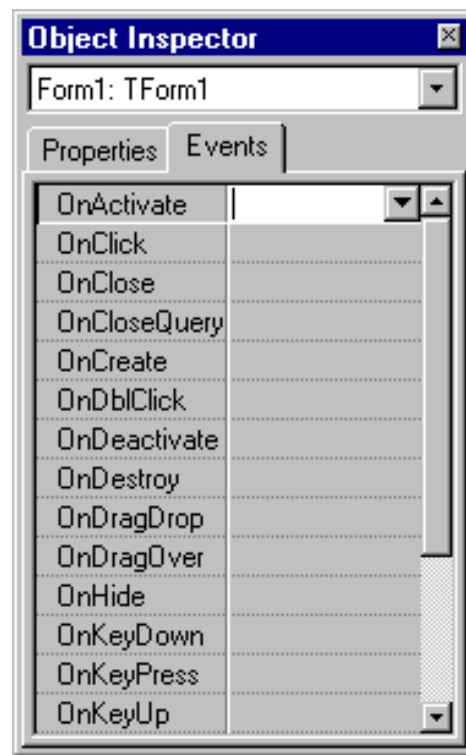
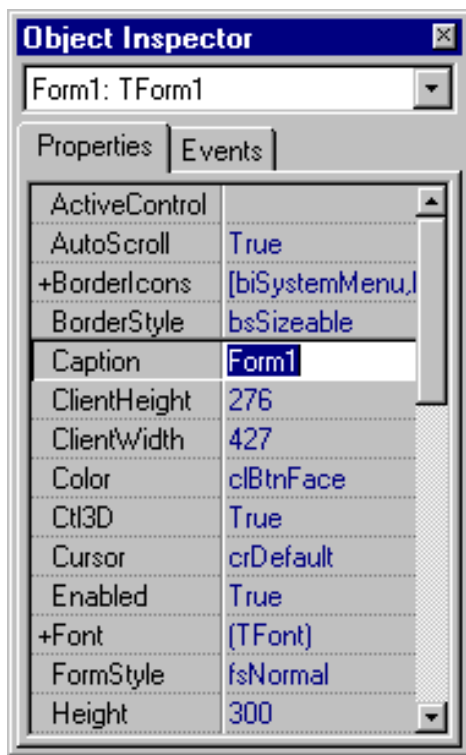
A komponenspalettán 11 lapon, tematikus csoportokba gyűjtve találhatók a Delphi komponensei (összesen mintegy 100 darab).



Már a *Standard* lapon található elemekkel felépíthető egy átlagos form, azaz Windows ablak (menü, gombok, jelölő négyzetek, rádiógombok stb. felhasználásával), az esetleg még szükséges komponensek a további lapokon találhatók. A form szerkesztése úgy történik, hogy a komponenseket az adott gombra kattintva elhelyezzük a formfelületen. Az egyes elemek tulajdonságait az Object Inspector-ban (objektumfelügyelő) állíthatjuk be.

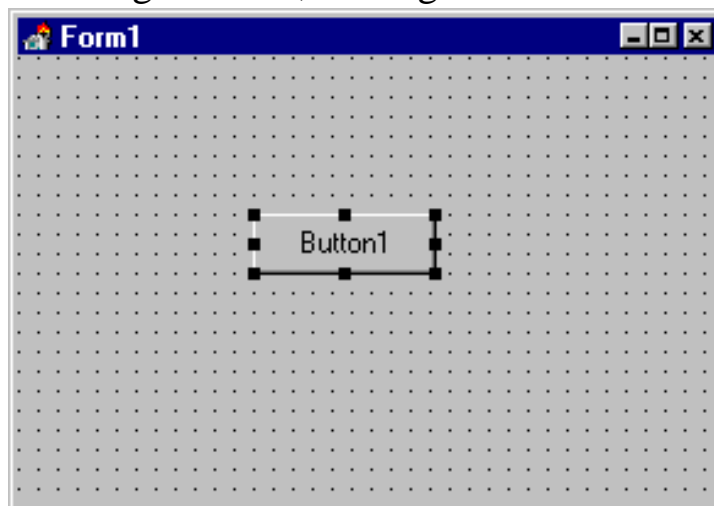
1.2. Az objektumfelügyelő

Az **Object Inspector** voltaképpen egy kétlapos párbeszéd-ablak, ahol beállíthatjuk a form és a komponensek tulajdonságait (név, méret, szín, betűk stb.). Ez történik a *Properties* nevű lapon. Az *Events* (események) lapon pedig az egyes elemekhez kapcsolódó eseményekről (pl. egérekattintás, billentyűütés stb.) intézkedhetünk. Az objektumfelügyelőben mindig a formszerkesztőn aktív (kijelölt) elem tulajdonságai állíthatók, módosíthatók.



1.3. A formszerkesztő

A formszerkesztő az a felület, amelyen a tervezett Windows ablakot kialakítjuk. A formon (űrlapon) kell elhelyezni a menüt, gombokat stb., az objektumfelügyelőben pedig megadhatjuk ezek tulajdonságait. Már maga a form is már egy teljes, ámbár üres Windows ablakot takar. Ha lefuttatnánk, egy olyan üres Windows ablakot kapnánk, amit méretezhetünk, áthelyezhetünk, használhatnánk a jobb felső ablakgombokat, de még a rendszermenüt is.



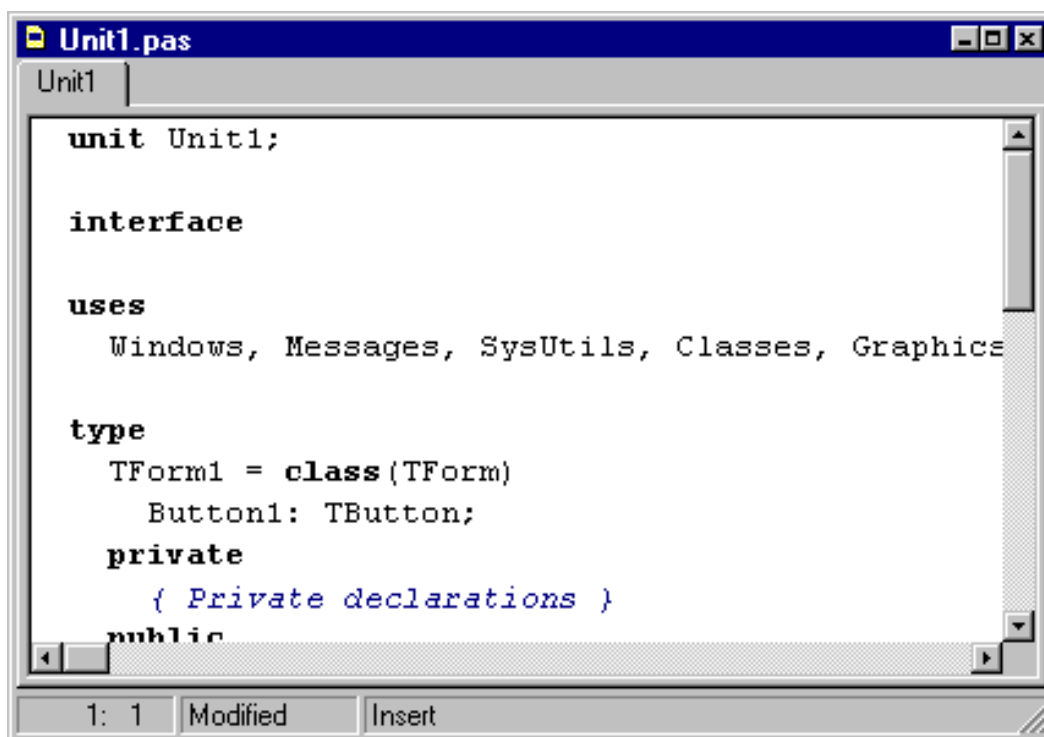
Az ábrán egy nyomógomb komponenst tettünk az űrlapra. A form felületén a komponensek elhelyezését pontozott rács segíti.

Az ablakmodellek leírását a Delphi **.DFM** kiterjesztésű állományokban tárolja.

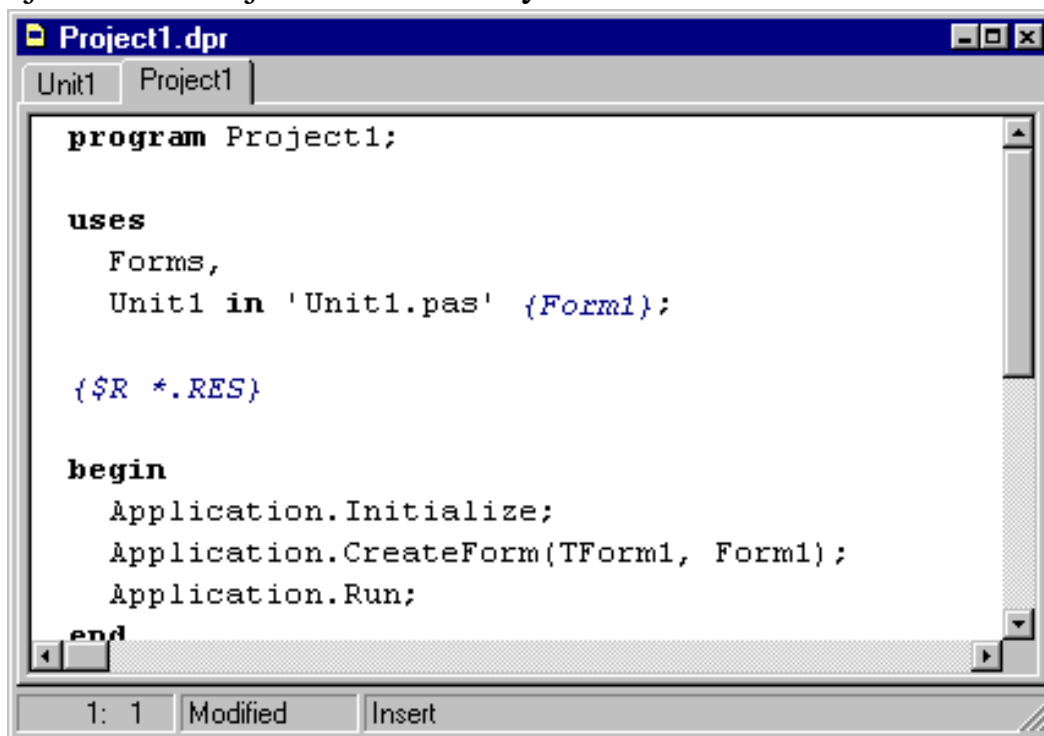
1.4. A kódszerkesztő ablak

A kód- (szöveg-) szerkesztő ablak alapesetben a formszerkesztő alatt alig látható. Akkor tűnik

elő, ha a kilógó alsó részére kattintunk, vagy megnyomjuk az **F12** funkcióbillentyűt. A következő ábra szerinti ablak látható:



Az ablakban az eddig "megírt" Object Pascal nyelvű kódszöveget látjuk. A Delphi ugyanis az általunk vizuális módon szerkesztett form, illetve a komponensek Pascal-kódját rögtön beilleszti az aktuális unit megfelelő helyére. A unitok fájljai **.PAS** kiterjesztésűek. Minden egyes formhoz tartozik egy unit (de létezik form nélküli unit is!). Egy program alá természetesen általában több form és több unit is tartozik. Maga a program (azaz "delphiül" projekt) kódja is megtekinthető, ha a *View* menü *Project Source* parancsát választjuk. A projekt forráskódja **.DPR** kiterjesztésű állományban tárolódik.



Bár az ábrák szerinti kódszerkesztő ablak a kódszövegnek csak kis részét láttatja (valójában ezek is méretezhető ablakok!), annyi máris kitűnik, hogy az Object Pascal szövegszerkesztője - a "régi" Turbo (Borland) Pascal 7.0 haladó hagyományait követve - automatikusan kiemeli a programnyelv kulcsszavait, a megjegyzéseket pedig dőlt kék betűkkel írja.

3. A komponensekről

Az előző fejezetekben láthattuk, hogy a programfejlesztés a Delphi-ben valóban egyszerű, de a felhasználható komponenseket meg kell ismernünk. Ugyancsak szükséges, hogy a komponensekhez tartozó tulajdonságokat is ismerjük. A következőkben ezeket tekintjük át - terjedelmi korlátok miatt korántsem a teljesség igényével.










3.1. A Standard lap komponensei

A komponenspaletta első, *Standard* lapján a leggyakrabban használt komponenseket találjuk meg.

	TMainMenu	Menütervezéshez használjuk.
	TPopupMenu	Gyorsmenü tervezéséhez használjuk.
	TLabel	Szövegmegjelenítésre használjuk.
	TEdit	Egysoros szövegmező.
	TMemo	Többsoros szövegmező.
	TButton	Nyomógomb.
	TCheckBox	Jelölőnégyzet.
	TRadioButton	Rádiógomb.
	TListBox	Lista létrehozása.
	TComboBox	Bővíthető lista létrehozása.
	TScrollBar	Gördítősáv.
	TGroupBox	Feliratos csoportablak.
	TRadioGroup	Rádiógomb csoportablak.
	TPanel	Felirat nélküli csoportablak.










3.2. Az Additional lap komponensei

Egyéb, különlegesebb, de még mindig kissé általános jellegű komponensek gyűjtőhelye.

	TBitBtn	Szöveggel, ábrával ellátható nyomógomb.
	TSpeedButton	Grafikával ellátható nyomógomb.
	TMaskEdit	A TEdit-hez hasonló, de itt megadható, hogy milyen karaktereket fogadjon el.
	TStringGrid	Szövegtáblázat.
	TDrawGrid	Képek táblázatos megjelenítése.
	TImage	Képmegjelenítő.
	TShape	Grafikai alakzatok megjelenítése.
	TBevel	3D-s vonal vagy négyszög.
	TScrollBar	Gördíthető felület.









3.3. A Win95 lap komponensei

Néhány Windows95-stílusú kiegészítő elem található itt.

	TTabControl	Többlapos párbeszédablak létrehozása.
	TPageControl	Többlapos párbeszédablak oldala.
	TTreeView	Tételek hierarchikus listája.
	TListView	Különböző módokon megjeleníthető tételista.
	TImageList	Grafikus képek csoportlistája.
	THeaderControl	Fejléc-szerkesztő (táblázatokhoz).
	TRichEdit	Rich text formátumú szövegdoz.
	TStatusBar	Állapotsor.
	TTrackBar	Tolópotméter (csúszka).
	TProgressBar	Egy művelet időbeli lefolyását szemléltető sáv.
	TUpDown	Le-fel nyíl adatkereséshez.
	THotKey	Billentyűkombináció megadása futási időben.













3.4. A Dialogs lap komponensei

A *Dialogs* lap komponensei a párbeszéd-ablakok néhány típusát segítik elkészíteni.

	TOpenDialog	A megszokott fájlnyitó párbeszéd-ablak.
	TSaveDialog	Mentés párbeszéd-ablak.
	TFontDialog	A betűbeállítások párbeszéd-ablaka.
	TColorDialog	A színbeállítás párbeszéd-ablaka.
	TPrintDialog	Nyomtatás párbeszéd-ablak.
	TPrinterSetupDialog	A nyomtató-beállítás párbeszéd-ablaka.
	TFindDialog	Keresés párbeszéd-ablak.
	TReplaceDialog	Csere párbeszéd-ablak.

3.5. A System lap komponensei

A *System* párbeszéd-lapon rendszer- és egyéb alkalmazáskomponensek találhatók.

	TTimer	Időkapcsoló, amivel beállítható időközönként alprogram-hívásokat lehet végrehajtani.
	TPaintBox	Rajzfelület kialakítása.
	TFileListBox	Az aktuális meghajtó tartalmát jeleníti meg.
	TDirectoryListBox	A könyvtári struktúrát mutatja meg.
	TDriveComboBox	Meghajtó-választás.
	TFilterComboBox	A TFileListBox szűrése.
	TMediaPlayer	Multimédia ablak.
	TOLEContainer	OLE kliens.
	TDDEClientConv	OLE kliens létrehozása.
	TDDEClientItem	A szerver-kliens irányú kapcsolat beállítása.
	TDDEServerConv	DDE szerver létrehozása.
	TDDEServerItem1	A kliens-szerver irányú kapcsolat beállítása.

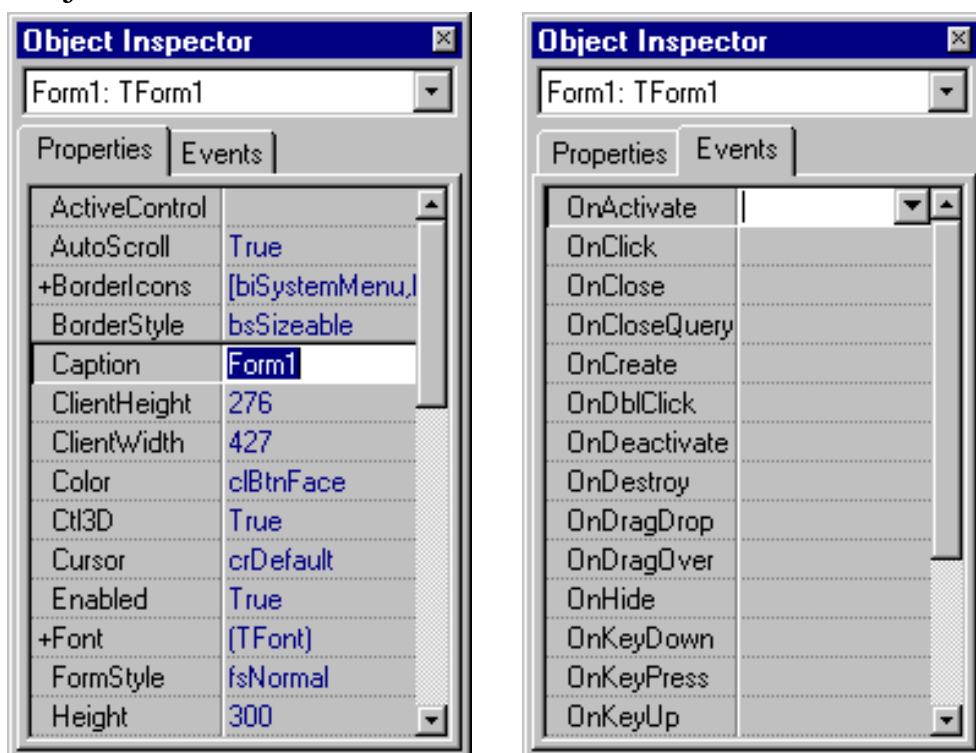
4. A formokról

A Delphi alapeleme a form vagy űrlap. Az űrlap komponens neve **TForm**. A Delphi indításakor kapott üres ablak egy **TForm** komponens, amelyen pontozott rács helyezkedik el.

A rács segíti a szükséges komponenseket elhelyezni az űrlapon. Az alkalmazás készítésének első szakasza nem más, mint a **TForm** objektum megszerkesztése (a komponensek elhelyezése), a tulajdonságok és eseménykezelések megadása.

4.1. A formok tulajdonságai

A Delphi objektumainak tulajdonságait általában az *Object Inspector*-ban, magyarul *objektumfelügyelőben* állíthatjuk be. Természetes, hogy mindig az az elem módosítható, amely aktív, amely ki van jelölve. A tulajdonságok nevei azonban ugyanazok minden komponens esetében, és ez igaz az eseményekre is. Korábban már láttuk, hogy az objektumfelügyelő kétlapos ablak. Az egyik, *Properties* nevű lapon a tulajdonságokat, az *Events* lapon pedig az eseményeket állíthatjuk be.



A tulajdonságokat a formok tulajdonságainak sorravételével tárgyaljuk. Egyéb esetekben használhatjuk a Delphi súgóját: jelöljük ki a tulajdonságot, és nyomjuk meg az **F1** billentyűt.

ActiveControl

Azt határozhatjuk meg, hogy melyik komponensen legyen a fókusz, azaz pl. gomb komponense esetén melyik nyomógomb legyen az alapértelmezett (körülötte sötétebb keret). A tulajdonság neve melletti, egyelőre üres részre kattintsunk rá, s a legördülő listából - amely a formon található komponenseket tartalmazza - kiválaszthatjuk a kívánt elemet.

AutoScroll

A tulajdonságnak két állása van: **True** és **False**. A **True** beállítása azt jelenti, hogy az ablak - amennyiben futási időben olyan kicsire módosítódik, hogy tartalma nem fér el benne - automatikusan kiegészül gördítősávval.

BorderIcons

A tulajdonságnevek egy része előtt + jel látható (hasonló a Windows Intézőhöz). Ez azt jelenti, hogy ha kétszer rákattintunk, akkor további altulajdonságok jelennek meg, amelyeket egyenként lehet beállítani (ha - jel van előtte, akkor viszont összezárható). A **BorderIcons** tulajdonsággal egyébként a Windowsos rendszermenü és ablakméretező gombok megjelenéséről rendelkezhetünk.

- BorderIcons	.biMaximize
biSystemMenu	True
biMinimize	True
biMaximize	True
biHelp	False

- **biSystemMenu**: a rendszermenü (vezérlőmenü) engedélyezése a **True** vagy a **False** választásával.
- **biMinimize**: az ablakminimalizáló gomb engedélyezése.
- **biMaximize**: az ablakmaximalizáló gomb engedélyezése.
- **biHelp**: a Help gomb engedélyezése.

BorderStyle

Az ablakszegély stílusát és méretezhetőségét határozza meg. A legördülő listából a következők közül választhatunk:

- **bsDialog**: szabványos párbeszéd-ablak keret, nem méretezhető.
- **bsSingle**: nem méretezhető szimpla vonalas keret.
- **bsNone**: nem látható, nem méretezhető keret, ablakmaximalizáló, minimalizáló ikonja, vagy rendszermenüje lehet.
- **bsSizeable**: normál méretezhető keret.
- **bsSizeToolWin**: olyan mint a bsSizeable, de kisebb címsora van.
- **bsToolWindow**: olyan mint a bsSingle, de kisebb címsora van.

Caption

Magyarul: képszöveg, felirat. Itt az ablak címsorában szereplő szöveg, az ablak neve. Közvetlenül kell beírni a **Caption** szövegmezőbe.

ClientHeight, ClientWidth

A form hasznos, kereten belüli méretét lehet megadni pixelben.

Color

A form háttérszínét adhatjuk meg. Vagy a legördülő listából választunk, vagy a jobb oldali cellára kétszer kattintunk, s a megjelenő párbeszéd-ablakból választunk.

Ctl3D

Három- vagy kétdimenziós megjelenítésről intézkedhetünk.

Cursor

Rengeteg kurzorforma közül válogathatunk, amely a form fölötti helyzetre értendő. A ságóban meg lehet tekinteni.

Enabled

A form csak **True** állásban reagál külső eseményekre, **False** állapotban nem!

Font

A betűk megjelenítését határozhatjuk meg egy többszintű beállítási listán:

- Font	[TFont] ...
Color	clWindowText
Height	-11
Name	MS Sans Serif
Pitch	fpDefault
Size	8
- Style	[]
fsBold	False
fsItalic	False
fsUnderline	False
fsStrikeOut	False

A három pontra kattintva összetett párbeszéd-ablakban tehetők meg a beállítások.

FormStyle

Négy beállítási lehetőség közül választhatunk. Az **fsNormal** az alapértelmezett, az **fsStayOnTop** esetén az ablak mindig a többi fölött fog elhelyezkedni, az MDI-s tulajdonságokat pedig MDI formok esetén kell választani.

Height

A form magasságát állíthatjuk be pixel egységben.

HelpContext

Súgó alkalmazása esetén használatos.

Hint

Ez is a súgóval kapcsolatos tulajdonság (a **ShowHint** tulajdonság **True**-ra állításánál aktív).

HorzScrollBar

A függőleges gördítősáv beállításának összetett beállítását tehetjük meg (a láthatóságot a **Visible** tulajdonság **True**-ra állításával biztosíthatjuk). A vízszintes gördítősáv tulajdonságait a **VertScrollBar** határozhatjuk meg.

Icon

A kívánt ikont a három pontra kattintva kereshetjük meg. Az ikon a rendszermenü jeleként fog szerepelni, s ezt látjuk a Tálcán összezárt alkalmazásikonon is. A Delphi 2.0-val adott "előre gyártott" ikonokat az Images/Icons mappában találjuk. Természetesen máshonnan származó, akár saját gyártású ikonok is felhasználhatók.

KeyPreview

A billentyűleütéseket a **False** beállításkor a komponens, **True** választásakor pedig a form dolgozza fel.

Left

A form helyzetét állíthatjuk be a képernyő bal oldalától (pixeleken mérve). A felső résztől való távolságot a **Top** tulajdonság írja le.

Menu

Menütervezésnél használjuk. Külön lesz róla szó.

Name

A formnak (de általában az objektumoknak) adhatunk nevet, hogy olvasható programszövegünk legyen. Legfeljebb egyszerű programoknál hagyjuk meg az eredetit. A név- (azonosító-) adás szabályai már a Pascal-ból ismertek.

ObjectMenuItem

OLE objektum menücímként való kezelésekor használjuk.

PixelsPerInch

Azt állíthatjuk be, hogy a form egy inch-nyi részén hány képernyőpont (pixel) legyen (a **Scaled** tulajdonság **True**-ra állásánál). A képernyőfelbontástól független programírást teszi lehetővé.

PopupMenu

A helyi vagy gyorsmenü nevét adhatjuk meg.

Position

A form elhelyezését leíró tulajdonság. Öt beállítási lehetőség van:

- **poDefault**: teljes mértékben a Delphi dönt a form helyéről és méretéről.
- **poDefaultPosOnly**: a Delphi dönt a form helyéről, méretét viszont nem változtatja.
- **poDefaultSizeOnly**: a form ott lesz, ahol a tervezéskor hagytuk, de a méretet a Delphi állítja be.
- **poDesigned**: ott és olyan lesz a form, amilyen a tervezéskor volt.
- **poScreenCenter**: a képernyő közepére helyezi a formot.

PrintScale

A form kinyomtatási módját leíró tulajdonság.

- **poNone**: nem lesz léptékváltoztatás, így a nyomat különbözni fog a képernyőtől.
- **poPrintToFit**: a nyomtatási méretek úgy módosulnak, hogy ráférjen a papírra.
- **poProportional**: a képernyőn és a papíron lévő form méretei megegyeznek.

Scaled

A **PixelsPerInch** tulajdonságnál volt róla szó.

ShowHint

A **Hint** tulajdonságnál volt róla szó.

Tag

Egész számot adhatunk meg, amivel információt tárolhatunk az adott objektumról.

Top

A form helyzetét állíthatjuk be a képernyő tetejétől (pixelekből mérve). Lásd **Left** tulajdonság is!

VertScrollBar

Lásd a **HorzScrollBar**-nál leírtakat.

Visible

A form látható vagy sem (futási időben).

Width

A form szélességi méretét lehet beállítani pixelekből.

WindowMenu

MDI alkalmazásoknál ablakmenüt lehet a formhoz rendelni.

WindowState

A form alapértelmezett méretarányát leíró tulajdonság.

- **wsMaximized**: Maximalizált méretű form.
- **wsMinimized**: Minimalizált méretű form.
- **wsNormal**: Sem nem maximalizált, sem nem minimalizált méretű form.

4.2. A formok eseményei

A Delphi programok eseményvezéreltek, ami azt jelenti, hogy a program valamilyen esemény bekövetkezésére (pl. egy nyomógomb megnyomására, billentyűleütésre stb. reagálva) fut tovább. Az események kezelését segíti az objektumfelügyelő második, *Events* nevű lapja. Az eseménykezelést lényegében nekünk kell megírunk, amihez a Delphi forráskódú sablont szolgáltat. A sablont az eseménynév jobb oldali cellájára való kétszeri kattintással, vagy az adott elemre való kettős kattintással hívhatjuk elő.

A következőkben a legfontosabb eseményeket ismertetjük (itt is a form események alapján).

OnActivate

Akkor használandó, amikor az inputfókusz áttevődik a formra.

OnClick

A formra való kattintás váltja ki. Első Delphi programunkban már használtuk.

OnClose, OnCloseQuery

Az **OnCloseQuery** eseményt az ablak bezárása, pontosabban bezárási kísérlete váltja ki. és az **OnCloseQuery** esemény után hajtódik végre. A formot bezárhatjuk az Alt+F4 billentyűpárossal, a bezáró ikonnal stb. Használhatjuk az eseményt a form bezárásának letiltására, memória-felszabadításra. Az **OnClose** esemény határozza meg az ablakbezárás módjait (az esemény *Action* paraméterei: *caNone*, *caHide*, *caFree*, *caMinimize*).

OnCreate

A form és komponensei kezdeti tulajdonságainak beállítására használjuk. Az esemény a form első végrehajtásakor következik be.

OnDblClick

Kétszeres egérekattintás váltja ki.

OnDeactivate

Más alkalmazásra való váltáskor következik be (pl. a programot a Tálcára tesszük

alkalmazásikon formájában, s egy másik alkalmazással, pl. szövegszerkesztővel kezdünk dolgozni).

OnDestroy

A memória-felszabadítás eszköze ablakbezáráskor.

OnDragDrop, OnDragOver

Az egér mozgását figyelő események (l. még OnMouse...).

OnHide

Az ablak elrejtésekor következik be.

OnKeyDown, OnKeyPress, OnKeyUp

A billentyűleütéseket figyelő események.

OnMouseDown, OnMouseMove, OnMouseUp

Az egér mozgását figyelő események (l. még OnDrag...).

OnPaint

Ha az ablak egy részét ideiglenesen eltakarta valami (pl. egy másik ablak), akkor az újrarajzolásról itt kell gondoskodni.

OnResize

Az ablak átméretezésekor az elemek áthelyezését és átméretezését itt kell megoldani.

OnShow

A form megjelenése előtt bekövetkező esemény.

5. A legfontosabb komponensekről

A komponenseket már áttekintettük korábban, most a leggyakrabban használt komponenseket ismerjük meg részletesebben, tulajdonságaikon keresztül. Itt természetesen csak azokat a komponens tulajdonságokat ismertetjük, amelyek a formtulajdonságoknál még nem fordultak elő (a tulajdonságnevek egyébként minden esetben azonosak).

5.1 A TLabel komponens

Első Delphi programunkban már használtuk ezt a komponenst, dolgoztunk a **Caption** tulajdonsággal és megismertük az **onClick** eseményt is. A **TLabel** komponens bármilyen szöveg elhelyezésére alkalmas: üzenet kiírására, egy másik komponens megnevezésére stb.

- **Align:** A Label komponenst a form, illetve a tartalmazó komponens széléhez viszonyítva igazítja.
 - **alTop:** A felső részre igazít.
 - **alBottm:** Az alsó részre igazít.
 - **alLeft:** Balra igazít.
 - **alRight:** Jobbra igazít.
 - **alNone:** Nem igazít sehová.
 - **alClient:** Elfoglalja az egész formot.

- **Alignment:** A szövegmezőn belül a szöveg elrendezését oldhatjuk meg.
 - **taCenter:** Középre rendez.
 - **taLeftJustify:** Balra rendez.
 - **taRightJustify:** Jobbra rendez.
- **AutoSize:** **True** állásnál a címke felveszi a szöveg méretét, **False** állásnál nem.
- **Transparent:** Ennek is két állása van: **True** és **False**. Ha a **True**-t állítjuk be, akkor a címke átlátszó lesz, s ami alatta van (pl. egy ábra), látszani fog.
- **Wordwrap:** **True** esetén sortörést állítunk be.

5.2. A TEdit komponens

Egysoros szövegmező, ahova a felhasználó beírhat valamit, de néha üzeneteket is megjelenítenek benne.

- **AutoSelect:** **True** állásnál ha a fókusz a mezőre kerül, akkor a benne lévő szöveg kijelölődik.
- **AutoSize:** **True** állásnál a szerkesztési terület magassága a választott betűmérethez igazodik, szélessége azonban nem változik.
- **CharCase:** A beírt szöveg karaktereit konvertálja (vagy nem konvertálja).
 - **ecLowerCase:** A beírt szöveget kisbetűssé alakítja.
 - **ecUpperCase:** Nagybetűssé konvertál.
 - **ecNormal:** Kis- és nagybetűk is megjeleníthetők, azaz a szöveg olyan lesz, ahogy a felhasználó begépelte.
- **HideSelection:** **True** állásnál ha a fókusz a mezőről elkerül, akkor a benne lévő szöveg kijelölése megszűnik, ellenkező esetben megmarad.
- **MaxLength:** A begépelhető karakterek számát adhatjuk meg.
- **PasswordChar:** Jelszó esetén használjuk. Az alapértelmezett érték **#0**, de ha nem akarjuk láttatni a begépelte jelszót, más karaktert kell beállítani.
- **ReadOnly:** **True** állásnál csak olvasható a szöveg, nem szerkeszthető.
- **Text:** A mezőben megjelenő kezdeti szöveg.

5.3. A TMemo komponens

Többsoros szövegmező, ahol majdhogynem korlátlan hosszúságú szöveget jeleníthetünk meg. Tulajdonságai - minthogy a **TEdit**-tel rokon komponens - sok tekintetben azonosak.

- **Lines:** A szövegmezőbe írhatunk be szöveget. Vagy a három pontra kattintunk, vagy a tulajdonság mezőre kétszer, s a megjelenő ablakban szerkeszthetjük szövegünket.
- **WantsTabs:** Ha azt akarjuk, hogy a felhasználó a tabulátor billentyűt is használhassa a szövegszerkesztés során, **True**-ra kell állítani. **False** esetén a fókuszot mozgatjuk a Tab billentyűvel a formon.

- **WantReturn:** **True** esetén a felhasználó az Enter-rel sort emelhet, **False** esetén az Enter-t a form kapja meg feldolgozásra.

5.4. A TButton komponens

Egyszerű nyomógomb, ami a Windowsos alkalmazások gyakorlatilag kikerülhetetlen eleme.

- **Cancel:** **True** esetén a nyomógomb az Esc billentyűre is reagál.
- **Default:** **True** esetén a nyomógomb lesz az alapértelmezett, így az Enter billentyűvel is működtethetjük.

5.5. A TBitBtn komponens

Ez is nyomógomb, ami az *Additional* lapon található. Abban különbözik a **TButton** komponenstől, hogy a szövegen kívül bittérképes grafika is elhelyezhető rajta. Tulajdonságai természetesen hasonlítanak a **TButton**-ra, itt csak a **TBitButton** tulajdonságait vesszük sorra.

- **Glyph:** A nyomógombon elhelyezendő ábrát kereshetjük meg és jelölhetjük ki, ha a tulajdonság jobb oldali cellájára duplán kattintunk, vagy a három pontos ikonra egyszer. A képszerkesztő ablakba (*Picture Editor*) tölthetjük a kívánt bittérképes ábrát. A Delphi által ajánlott ábrák az *Images/Buttons* könyvtárban találhatók.
- **Kind:** Tizenegy különböző Delphi-szabvány szerinti gomb közül választhatunk.
- **Layout:** Az ábra helyzetét határozhatjuk meg (balra, jobbra, fel, le).
- **Margin:** A gomb és kerete közötti távolság megadása.
- **NumGlyphs:** Egy gombon több, de maximum négy azonos méretű kép is elhelyezhető. Ezek számát állíthatjuk be itt.
- **Spacing:** A felirat és a keret közötti távolság megadása.

5.6. A TCheckBox és a TRadioButton komponens

A **TCheckBox** a jól ismert jelölőnégyzet, amiből több is elhelyezhető egy csoportban, s rákattintással több is kijelölhető. Ha egymást kizáró lehetőségek közül kell választani, a **TRadioButton** (rádiógomb) lehetőséget kell választani.

- **Checked:** **True** állásban az adott opció alapértelmezetten kijelölt (a négyzetben pipa, a körben pedig pont látszik).

5.7. A TListBox és a TComboBox komponens

A **TListBox** egy görgethető listaablak, a lista elemei közül választható ki a kívánt elem. A **TComboBox** is egy listaablak, de itt van lehetőség a listától eltérő elem beírására is.

- **Column:** A **TListBox** listaablakban a látható oszlopok számát lehet beállítani.

- **ExtendedSelect, MultiSelect**: A **TListBox** tulajdonságai ezek is. Amennyiben a **MultiSelect True**-ra van állítva, akkor lehetőség van egynél több elem kiválasztására. Ha a **MultiSelect** mellett az **ExtendedSelect** is **True** állásban van, akkor - Windowsos módra - a Shift-es és a Ctrl-os kijelölési mód is alkalmazható.
- **Sorted: True** állásban a lista új elem beírásakor sorbarendeződik.
- **Style**: A lista megjelenési formáját lehet beállítani.

5.8. A TStringGrid és a TDrawGrid komponens

Mindkét komponens az *Additional* lapon található. A **TStringGrid** komponens szövegtáblázat létrehozását segíti, míg a **TDrawGrid** komponensben a szövegen kívül ábrát is elhelyezhetünk. A formra letett komponens emlékeztet az Excel táblázatra.

- **ColCount**: A táblázat oszlopainak száma állítható be.
- **DefaultColWidth, DefaultRowHeight**: Az oszlopok és a sorok méretét lehet megadni.
- **FixedCols, FixedRows**: A fejléc, vagy az első oszlop celláinak módosíthatatlanságát biztosítja.
- **FixedColors**: A rögzített cellák színe állítható be.
- **GridLineWidth**: A cellahatárokat alkotó vonalak vastagsága adható meg (0 esetén nincs vonal).
- **Height, Width**: A táblázat méretei adhatók meg.
- **Options**: Rengeteg - elsősorban a táblázat kinézetéért felelős - tulajdonság állítható itt be. Például:
 - **goFixedVertLine: False** választásakor az első (rögzített) sor cellák nélküli összefüggő sáv lesz.
 - **goFixedHorzLine: False** állásban az első (rögzített) oszlop cellák nélküli összefüggő sáv lesz.
 - **goVertLine**: A **False** érték beállításakor a táblázat függőleges vonalai eltűnnek.
 - **goHorzLine: False** állásban a táblázat vízszintes vonalai eltűnnek.

6. Az Object Pascal-ról

Jó, ha a Delphi felhasználók rendelkeznek bizonyos Pascal ismeretekkel, hiszen a Delphi alapját képező Object Pascal a Turbo Pascal egy továbbfejlesztett változata. Noha a Pascal korábbi változatáról már készült egy hasonló jegyzet ("*Szabó László: A Pascal programnyelv*"), amit a kezdőknek mindenképpen ajánlatos áttanulmányozni, a következőkben vázlatosan áttekintjük az Object Pascal legfontosabb sajátosságait.

6.1. A Pascal program szerkezete

A Delphiben a programírás általában egy form és a komponensek tulajdonságainak leírásával kezdődik. A Delphi elindítása után megjelent Form1 űrlap mögött voltaképp egy, már megírt

program van. Ahogy korábban már láttuk, a formot egy unit (egység) írja le. Egy Delphi program tehát a főprogramon kívül legalább egy unitot is tartalmaz.

A unitok zárt, önálló modulok, adott céllal. Írhatunk ilyet mi is, de a Delphineknél - mint a Turbo Pascalnak is - vannak olyan saját belső egységei, amelyekben a Delphi eljárásait, függvényeit, objektumait stb. helyezték el. Az általunk írt unitok vagy formhoz kötöttek, vagy nem.

Elsősorban nagyobb projektek esetén célszerű a logikailag, működés szempontjából együvé tartozó programrészeket egy egységbe összevonni. Így - túl azon, hogy a programunk áttekinthetőbb - a fordítás is sokkal gyorsabb lesz.

A főprogram kódját a View menü Project Source parancsának kiadásával jeleníthetjük meg:

```

program Project1;

uses
    Forms,
    Unit1 in 'Unit1.pas' {Form1};

{$R *.RES}

begin
    Application.Initialize;
    Application.CreateForm(TForm1, Form1);
    Application.Run;
end.

```

Egy Delphi-program tehát a következő fő részekből áll:

- programfej,
- egységek felsorolása,
- főprogram.

Ettől izgalmasabb az egységek szerkezete. A Form1-hez kapcsolt unit forráskódja az F12 funkcióbillentyűvel hívható elő:

```

unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes,
  Graphics, Controls, Forms, Dialogs;

type
  TForm1 = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

end.

```

Egy Delphi unit a következő fő részekből áll:

- unitfej,
- az illesztő rész (*interface*),
- a deklarációs rész (type, var stb.),
- a kifejtő rész (*implementation*),
- az egységzárás (*end.*).

Az illesztő részben, az ***interface*** kulcsszó után definiálhatjuk azt, amit más programrészek is láthatnak. Az itt nem definiált részek csak az adott unit számára elérhetők. Itt kell megadni azt is, hogy melyik más unitok tartalmát akarjuk használni. Ezt kell felsorolni a ***uses*** kulcsszó után. A deklarációs részben a unit számára látható típus-, változó-, konstans- és címkedeklarációkat kell megadnunk. Az **<IMPLEMENTATION< i>** után kell a unitot alkotó eljárásokat, függvényeket leírnunk. Ezek közül kifelé csak azok láthatók, ha az *interface* után felsoroljuk őket (az eljárás- és függvényfejeket). Az egységet ***end.*** zárja le (a főprogram végét is pontos *end* jelzi!). A Delphi unitnak opcionálisan még lehetnek egyéb részei is, ezekről azonban itt nem ejtünk szót.

6.2. A Pascal építőelemei

Az *Object Pascal* nyelvû program alapvetõ építõelemei, azaz legkisebb értelmezhetõ egységei a következõk:

- szimbólumok,
- fenntartott szavak,
- azonosítók,
- címkék,
- konstansok,
- elválasztójelek.

a) Szimbólumok: A programban bizonyos szimbólumokat használunk. Gondoljunk csak a matematikai műveletek végzésére (műveleti jelek, relációs jelek, zárójelek stb.). A Pascal nyelvû programokban használatos szimbólumokat tartalmazza az alábbi táblázat.

+	-	*	/	=	;	:	,
<	>	<=	>=	<>	'	.	..
()	[]	{ }	(**)	^	\$	@	#

b) Foglalt szavak: A foglalt szavakat, vagy kulcsszavakat a programozónak nem szabad használni a programban változók, konstansok, eljárások, vagy bármi más elnevezésére. A foglalt szavakat a Delphi vastagon írja a forrásszövegben. A foglalt szavakat foglalja össze a következő táblázat (kiegészítve az ugyan nem tiltott, de nem ajánlott szavakkal):

and	array	as	asm	begin
case	class	const	constructor	destructor
div	do	downto	else	end
except	exports	file	finalization	finally
for	function	goto	if	implementation
in	inherited	initialization	inline	interface
is	label	library	mod	nil
not	object	of	on	or
packed	procedure	program	property	raise
record	repeat	set	shl	shr
string	then	threadvar	to	try
type	unit	until	uses	var
while	with	xor		
absolute	abstract	assembler	at	automated
cdecl	default	dynamic	export	external
far	forward	index	interrupt	message
name	near	nodefault	override	private
protected	public	published	read	register

resident	stdcall	stored	virtual	write
----------	---------	--------	---------	-------

c) Azonosítók: Az azonosítók a program bizonyos elemeinek elnevezésére szolgálnak. Pl. a programnak nevet kell adni, az egyes változókat, konstansokat, eljárásokat is el kell keresztelni. Az azonosító ASCII karakterekből állhat (kis- és nagybetűk, számjegyek). Vannak azonban bizonyos megszorítások:

- kis- és nagybetűket (ékezet nélkül!) tetszés szerint használhatunk;
- számmal nem kezdődhet, de lehet benne;
- kötőjel nem, de aláhúzásjel (_) lehet benne;
- szóköz, *, !, ? nem lehet;
- azonos változó- vagy konstansnév nem szerepelhet kétszer (eltérő típus esetén sem);
- fenntartott szó nem lehet;
- hossza nem meghatározott, de csak az első 63 karakter számít.

Az elmondottak értelmében jó nevek lehetnek a következők:

Hki, Hallgatók, Folyo_Ev, Bbe1, Bbe2, ALMA

Ezek azonban rosszak:

2Bbe, Hallgatók, Folyó Év, 222, Egy*Egy, Gyere!

Ahhoz, hogy a fordító tudja, hogy pl. a Hallgatók egy változónk azonosítója, a deklarációs részben közölni kell vele. Saját változóink nevét a típusmegadás és a változók felsorolása helyén adhatjuk meg (a type és a var kulcsszavak után). Az azonos típusú változókat vesszővel elválasztva lehet felsorolni, az utolsó után pedig pontosvesszőt kell tenni.

d) Címkék: A címkék azok az elemek, amik használatát általában kerülni kell. De mire is használjuk a címkéket? Arra, hogy a program bármely pontjáról elküldhetjük a vezérlést a program egy másik pontjára. A **goto 10;** utasítás segítségével például a program vezérlése az utasítás helyéről átugrik a **10:** címkével megjelölt helyre (a címke után kettőspontot kell tenni!). A címke elhelyezkedhet a goto előtti vagy utáni programrészen. A címke nemcsak szám lehet, hanem szó is (pl. Ide, vagy Gyere). Ahhoz, hogy a fordító felismerje a goto utáni címkét, a deklarációs részben a **label** kulcsszó után kell elhelyezni. Ha több címke is van, akkor vesszővel elválasztva kell felsorolni őket, a végén pontosvesszővel.

e) Konstansok: A konstans lehet szám- vagy szövegkonstans. Állandókat akkor adunk meg, ha egy bizonyos számértéket vagy szövegrészt a programban gyakran használunk. Ezeket az adatokat közvetlenül a programba (pl. képletbe, összefüggésbe) is be lehet írni, de ha esetleg a programfejlesztés során módosítani szükséges, akkor sokkal egyszerűbb állandóként meghatározni, s csak ezt az egy értéket javítani. Ilyen állandók lehetnek például: 1456, 21E7, -273.16, vagy 'Összesen:'. A konstansokat a **const** kulcsszó után egyenlőségjelet használva ismertetjük, pl. így:

```
sum = 'Összesen:';
```

vagy

```
T0abs = -273.16; .
```

f) Elválasztó elemek: A Pascal nyelvű programok elválasztó szimbólumai a szóköz és a

sorvég-jel. A szóköznek a programozás elemeinek felsorolásakor van jelentősége (a használatos szavak közé szóközt teszünk, ez természetes), a sorvég jel pedig általában pontosvessző (a program végén lévő end után pont). Ha egy programsor után nem teszünk pontosvesszőt, akkor a fordító a következő sort is az előző folytatásának fogja tekinteni. A program adott helyein megjegyzéseket helyezhetünk el, ez jelentős mértékben megkönnyíti a forrásszöveg olvashatóságát. A megjegyzéseket kapcsos zárójelek közé tesszük, de használhatjuk a (* és *) párokat is. A Delphi a megjegyzéseket sötétkék dőlt betűkkel jeleníti meg a megjegyzéseket. Például:

```
{Az adatbeolvasás kezdete}
```

vagy

```
(* Az n szerinti ciklus vége *)
```

6.3. Az Object Pascal típusai

A Delphi-változók típusai is bővültek a Turbo Pascalhoz képest. Ezért, igaz csak vázlatosan, de át kell tekintenünk ezeket is.

6.3.1. Egyszerű típusok

Az egyszerű típusokhoz tartozó változók típusazonosítóját nem kell a **type** kulcsszó után ismertetni. Egyes típusok esetén a 16 és a 32 bites Delphi-változatok között különbség lehet.

Egész típusok

ShortInt	1 bájt	előjeles	-128...127
SmallInt	2 bájt	előjeles	-32768...32767
Integer	4 bájt	előjeles	-2147483648...2147483647
LongInt	4 bájt	előjeles	-2147483648...2147483647
Byte	1 bájt	előjel nélküli	0...255
Word	2 bájt	előjel nélküli	0...65535
Cardinal	4 bájt	előjel nélküli	0...2147483647

Valós típusok

Real	6 bájt	11-12 j.	-2.9E-39...17E38 (kerülendő a használata!)
Single	4 bájt	7-8 j.	-1.5E-45...3.4E38
Double	8 bájt	15-16 j.	-5.0E-324...1.7E308
Extended	10 bájt	19-20 j.	-3.4E-4932...1.1E4932
Comp	8 bájt	19-20 j.	-2.0E-63...2.0E63-1

Karakter típusok

ANSIChar	1 bájt	ANSI	1 ANSI karakter
WideChar	2 bájt	Unicode	1 Unicode karakter
Char	1 bájt	Unicode	(0...255)

Karakterlánc (string) típusok

ShortString	255 bájt	ANSIChar	max. 255 karakter
String, AnsiString	~3 GB	ANSIChar	max. 4294967296
WideString	~1,5 GB	WideChar	Unicode karakterek

Logikai típusok

Boolean	1 bájt	0, 1	False, True	
	ByteBool	1 bájt		Bájt-méretű numerikus Boolean
	Bool, WordBool	2 bájt		Word-méretű numerikus Boolean
	LongBool	4 bájt		Kétszeres Word-méretű numerikus Boolean

6.3.2. Összetett típusok

Az összetett típusok egyszerű típusokból összetettek. Ezeket a **type** kulcsszó után ismertetni kell. Az összetett típusok tulajdonságait a felhasználó szabhatja meg, és saját névvel láthatja el.

Felsorolt típus:

A felsorolt típus esetén egyenként felsoroljuk az összes lehetséges értéket. A felsorolt értékekhez az Object Pascal sorszámokat rendel (0, 1, 2, 3, ...). Példák:

```
Madarak = (fecske, golya, sas, bagoly);      Lanyok = (Anna, Jolán, Éva,
Adrienn);      Napok = (Hétfő, Kedd, Szerda, Csütörtök, Péntek, Szombat, Vasárnap);
```

Intervallum típus:

Az intervallum (helyesebben részintervallum) típus egy nagyobb intervallumon belül egy kisebb intervallumot határoz meg. Például:

```
NagyBetuk : 'A'..'Z';      MunkaNapok : Hétfő..Péntek;
```

Az első példa a char típus, a második példa pedig az előző pontban ismertetett *Napok* sorszámozott típus intervalluma. Az intervallum megadásánál megfigyelhetjük, hogy az intervallumok jelölésére két pontot (..) kell használni.

Rekord típus:

A rekord típus igen rugalmas adattípus-fajta, amelyben minden elemnek, mezőnek saját típusa lehet (még rekord típus is!). A rekord deklarációja **record** és **end** kulcsszavak között felsorolva tartalmazza az elemek nevét és típusát. Például:

```
Datum = record      Ev : 1900..2000;      Ho :
1..12;      Nap : 1..31;      end;      Hallgato =
record      VezNev, KerNev : String[20];      Szulido :
```



```
Datum;                      TanKor : String[5];                      TanAtlag :
Single;                     end;
```

Egy ilyen rekordos változóra például a következő módon hivatkozhatunk: Hallgato.TanKor.

Tömb típus:

A tömb típust a kulcsszava után **array** típusnak is nevezik. A típus meghatározott számú és típusú elemből áll. Az elemek számát (pontosabban intervallumát) szögletes zárójelbe írt indexekkel adjuk meg, az elemek típusát pedig az **of** kulcsszó után jelöljük. Az elemekre a programban az index alapján hivatkozhatunk. Példák:

```
Oszlop = array[1..10] of Double;      Sor = array[1..20] of Double;
```

A fenti deklarációban szereplő adatokra például az Oszlop[3] vagy a Sor[15] formában hivatkozhatunk. A két deklaráció egyesítéséből kétdimenziós tömböt (táblázatot) hozhatunk létre:

```
Tablazat = array[1..10,1..20] of Double;
```

A hivatkozás módja ebben az esetben: Tablazat[3][15], vagy egyszerûbben: Tablazat[3,15]. Az array típus háromdimenziós (de természetesen akár nagyobb) tömb létrehozására is használható:

```
Tomb = array[1..10,1..20,1..4] of Integer;
```

Mutató típus:

A mutató típus, vagy pointer olyan változót határoz meg, amely egy másik változó memóriacímére mutat. A mutató deklarálásához a kalap karaktert használjuk:

```
PValtozo = ^Valtozo;
```

Állomány típus:

Az állomány típusok a fájl műveletek végzéséhez (beolvasás, mentés) szükségesek:

```
HallgFile = file of Hallgato;
```

6.4. A programozás elemei

6.4.1. Utasítások

A programok futását, azt hogy az adatokkal mit csináljon a program, utasításokkal lehet vezérelni. Az egyszerû vagy elemi utasítások - a feldolgozás legkisebb részei - közé az értékadás, az eljáráshívás és az ugrás tartozik. Az egyszerû utasításoktól eltérően a strukturált utasítások segítségével egymásba ágyazott feldolgozási szintek hozhatók létre. Létezik összekapcsoló, with, feltételes és ciklusutasítás.

6.4.1.1. Értékadás

Egy változónak értékadással adhatunk értéket. Az értékadásban szereplő kifejezés lehet érték, azonosító vagy függvényhívás. Az értékadás operátora a **:=** jel, aminek a jelentése: **legyen egyenlő**. Az értékadó operátor bal oldalán egy változó áll, jobb oldalán pedig egy kifejezés. Nézzünk néhány példát:

```
x:=50;      Hki:=H0-dh;      b5:=Sqr(B[i])+x1;      y:=Hatvany(a5,3);
```


6.4.1.2. Eljáráshívás

Egy eljárás (**procedure**) azonosítója megadásával hívható. Lehet paraméteres vagy paraméter nélküli eljárás. A paraméteres eljáráshíváskor a paramétereket zárójelben kell megadni. Az eljárásoknak a programban általában a hívás előtt kell elhelyezkedni. Példák:

```
AdatKiiras;      MessageDlg('Delphi 2.0',mtInformation,[mbOk],0);
```

6.4.1.3. A goto utasítás

A programvezérlést átadhatjuk egy - előzőleg a **label** kulcsszó után definiált - címkére, amelynek az utasítással azonos blokkban kell lennie. Például:

```
goto l;      goto Ide;
```

6.4.1.4. Összekapcsoló utasítás

Az összekapcsoló utasítással (a **begin** és az **end** kulcsszó-pár) az egy blokkba tartozó utasításokat fogjuk össze. Például:

```
procedure TFormElso.FormCreate(Sender: TObject);      begin      Tetel:=TElso.  
Create;      Darab:=TDarab.Create;      AktTetel:=Tetel;      end;
```

6.4.1.5. Feltételes utasítások

A feltételes utasításokkal a program feltétel(ek)től függő futása vezérelhető. Idetartozik az **if** és a **case** utasítás.

a) Az if utasítás

Az **if** (jelentése: **ha**) feltétele csak igaz vagy hamis eredményt adhat. Többirányú elágazást az **if ... then ... else** kombinációjával vagy a **case** utasítással lehet elérni. Példák:

```
if x<=10 then x:=10;  
if x < 10      then x:=x+1      else x:=10;
```

b) A case utasítás

A **case** (jelentése: **eset**) utasítás egy szelektorból és egy elágazási listából áll. A szelektornak sorszámozott típusúnak kell lennie. Az elágazások közé **if**-es ág is beépíthető. A **case** utasításban is lehet **else** ág, amelyre akkor kerül a vezérlés, ha nem létezik a kifejezésnek megfelelő elágazási út. A case utasítást minden esetben **end** zárja le. Példa:

```
case Honap of      1,3,5,7,8,10,12: NapokSzama:=31;      4,6,9,11:  
NapokSzama:=30;      2: if SzokoEv then NapokSzama:=29      else  
NapokSzama:=28;      end;
```

6.4.1.6. A ciklusok

A ciklusok segítségével megismételhető a program meghatározott része. Idesoroljuk a **for ... to (downto)**, a **while ... do**, a **repeat ... until** és a **with ... do** utasításokat.

a) A repeat ... until utasítás

A **repeat ... until** típusú ciklus felépítése: ismételd (**repeat**) a következő programrészt addig (**until**), amíg a következő feltétel megvalósul. Azaz a program az **until** kulcsszó után álló feltételt értékeli ki. Az utasítások legalább egyszer lefutnak, és a ciklus addig ismétlődik, amíg

a feltétel igaz lesz. Például:

```
repeat      i:=i+1;      j:=j+2;      until      (i>=10) or (j>=40);
```

b) A **while ... do** utasítás

Az előző ciklusutasítással szemben a **while ... do** utasítás a ciklus elején vizsgálja a feltételt, s az utasítások csak akkor ismétlődnek, ha a feltétel igaz. Példa:

```
while x<=10 do      osszeg:=osszeg+x;
```

c) A **for** utasítás

A **for** utasítással numerikus ciklus készíthető, amelyben a ciklusváltozó megadott értékig változik. A ciklusváltozó a **to** szóval növelhető, a **downto** szóval csökkenthető. Példa:

```
for i=1 to 10 do      osszeg:=osszeg+x[i];
```

d) A **with** utasítás

A **with** utasítás a rekord típusú változók mezőinek egyszerűbb kezelését teszi lehetővé. A típusismertetés során közölt példa rekordjainak például a következő módon adhatunk értéket: **with** nélkül:

```
begin      Hallgato.VezNev:='Kovács';      Hallgato.KerNev:='János';
Hallgato.TanKor:='K.102';      Hallgato.TanAtlag:=4,25;      end;
```

Ugyanez a **with** utasítás segítségével:

```
with Hallgato do begin      VezNev:='Kovács';      KerNev:='János';
TanKor:='K.102';      TanAtlag:=4,25;      end;
```

6.4.2. A Pascal rutinjai

A programban deklarált rutinok, azaz az eljárások és függvények arra teremtenek lehetőséget, hogy a programunkat logikailag és funkcionálisan meghatározott különálló egységekre bontsuk. Az eljárásokat és a függvényeket a főprogram (vagy egy másik eljárás) vezérli.

6.4.2.1. Eljárások

Az eljárás deklarálása a **procedure** kulcsszóval indul, amelyet az **azonosító** (név) követ. A név után zárójelben **paraméter(ek)** is megadható(k). Az eljárás szerkezete kicsiben követi a program szerkezetét. Az eljárás feje után a belső (**lokális**) **változók**, **típusok**, **konstansok** és **címkék** felsorolása következik (de nem biztos, hogy vannak). Az utasításokat **begin** és **end**; között kell felsorolni, minden utasítást logikailag külön sornak tekintve, azaz pontosvesszőt téve a végére (ettől még lehet egy sorba több utasítást írni). A záró **end** után is pontosvesszőt kell tenni.

```
procedure EljarasNev(parameter);      lokális címkék,
      konstansok,      változók,      típusok      begin
Utasítások;      end;
```

a) Példa paraméter nélküli eljárásra:

```
procedure Uzenet;      begin      ShowMessage('Kevés a memória!');      end;
```

b) Példa paraméteres eljárásra:

```
procedure Szorzas5(var Szam: Double);      begin      Szam:=Szam * 5;      end;
```

6.4.2.2. Függvények

A függvény abban különbözik az eljárástól, hogy értéket (de csak egy értéket) ad vissza. Deklarálása a **function** kulcsszóval indul, amelyet az **azonosító** (név), a paraméterek és az eredmény típusának deklarálása követ. Az utasításblokkban adjuk meg a végrehajtandó utasításokat. A függvényt az azonosítóval kell hívni, a paraméterek típusának és sorrendjének egyeznie kell.

A következő példa tetszőleges kitevőjű hatványozásra alkalmas függvényt mutat be (a Pascal nyelvben csak a négyzetre emelésre van beépített függvény, ez az **Sqr**).

```
function Hatvany(alap,kitevo: Double):Double;      begin      Hatvany:=Exp
(kitevo*ln(alap));      end;
```

Használata:

```
Eredmeny:=2.85*hatvany(szam,4);
```

6.4.2.3. Paraméterek

A paraméterekről sok szó esett már korábban. Nézzük meg most, mik is ezek. Paramétereket az eljárások és függvények deklarálásakor adhatunk meg az azonosító (név) mögött, kerek zárójelek között. A paraméterek tulajdonképpen a rutinon belüli lokális változók. A paraméterekre az azonosítókkal hivatkozhatunk. A Pascal három paramétertípust különböztet meg:

- értékparaméter,
- változó paraméter,
- típus nélküli paraméter.

Ha egy paramétert csak az azonosítójával adunk meg, akkor **értékparaméter**ről beszélünk. Ha a paraméter előtt a **var** kulcsszó áll, és a paramétert típusdeklaráció is követi, akkor **változó paraméter**ről van szó. Ha a **var** után nem áll típusdeklaráció, akkor **típus nélküli** a paraméter.

7. Az Object Pascal egységei

Az egységek (unit-ok) zárt, önálló modulok, adott céllal. Írhatunk ilyet mi is, de az Object Pascal-nak vannak olyan saját belső egységei is, amelyekben a programnyelv elemeit, eljárásait, függvényeit helyezték el. Ha valamely egységből akarunk használni eljárást vagy függvényt, akkor a programfejben a **uses** kulcsszó után meg kell nevezni az adott egységet (a System kivételével). Természetesen ezen bejegyzéseknek a jelentős részét a Delphi automatikusan elvégzi.

A következőkben a Delphi egységeit ismertetjük, természetesen nem teljes részletességgel. A Delphi képernyős Helpje segítségünkre lehet valamennyi megismerésében.

A Delphiben az egységek az **RTL**-ben (*Run-time Library*, azaz futási idejű könyvtár) és a **VCL**-ben (*Visual Components Library*, azaz a vizuális komponensek könyvtára) vannak. A Turbo Pascal egységeinek áttekintése még egyszerű volt, hiszen mindössze 6 (+2) volt belőlük. Az RTL-ben ugyan mindössze 3 unit van, ezzel szemben a VCL-ben 34 egység található. Ezeknek és tartalmuknak az ismertetése természetesen lehetetlen, csak a fontosabbaknak

ítéltekről lehet szó.

7.1. Az RTL standard unitjai

A *Run-time Library* elemei három egységben, a **System**, a **SysUtils** és a **Math** unitokban vannak elhelyezve.

7.1.1. A System unit

Ebben az egységben azon nyelvi elemek, konstansok, változók, eljárások és függvények vannak, amelyek a szabványos nyelvi környezethez tartoznak. Ezt az egységet valamennyi program igényli, ezért a `uses` után nem kell szerepeltetni.

A System egység elemeit két nagy csoportra oszthatjuk:

- eljárások és függvények,
- változók és tipizált konstansok.

Az ismertetésben nagyjából a Borland Delphi referenciakönyvének csoportosítását követjük. A következő terjedelmes táblázatban a Delphi System egységének eljárásait és függvényeit ismertetjük tematikailag csoportosítva.

A System egység eljárásai és függvényei	
Aritmetikai rutinok	
Abs	Az argumentum abszolút értékét adja vissza.
ArcTan	A függvény az átadott argumentum arkusz tangensét adja vissza. A számítások mértékegysége a radián.
Cos	Az átadott argumentum koszinuszát adja vissza.
Exp	A függvény az "e" szám valamely hatványát adja vissza.
Frac	Valamely numerikus kifejezés tizedes részét adja vissza. A számítás a $z := x - \text{Int}(x)$ formulának felel meg.
Int	A függvény valamely valós szám egész részét adja vissza.
Ln	Valamely numerikus kifejezés természetes alapú logaritmusát adja vissza.
Pi	A függvény a Pi matematikai állandó értékét adja vissza (3.1415926535897932385).
Round	A függvény - a kerekítési szabályoknak megfelelően felfelé vagy lefelé - a numerikus kifejezést a legközelebbi egészre kerekíti.
Sin	Az átadott argumentum szinuszt adja vissza.
Sqr	A függvény a megadott kifejezés négyzetét adja vissza.
Sqrt	A függvény a megadott kifejezés négyzetgyökét adja vissza.
Trunc	A függvény egy valós szám egész részét adja vissza, miközben levágja a tizedeseket.
Stringműveletek rutinjai	

Concat	Stringek összeadása.
Copy	Adott karakterlánc adott karakterétől kezdve kimásol adott számú karaktert.
Delete	Egy karakterláncból részstringet töröl ki.
Insert	Karakterláncba adott pozícióra stringet illeszt.
Length	A függvény visszaadja valamely karakterlánc karaktereinek számát.
Pos	Részstringet keres a karakterláncban.
SetLength	Stringváltozó dinamikus hosszát állítja be.
SetString	Az adott string tartalmát és hosszát állítja be.
StringOfCharS	Adott számú karakterből álló string.

I/O rutinok

AssignFile	Az eljárás tetszőleges állományváltozóhoz hozzárendeli a karakterlánccal megadott fizikai állományt.
CloseFile	Bezár egy megnyitott állományt.
Eof	A függvény azt vizsgálja, hogy elértük-e az állomány végét. Ha végigolvastunk egy állományt, akkor az eredmény True lesz.
Erase	Az eljárás törli az állományt, amelyet az AssignFile hívásakor valamely állományváltozóhoz rendeltünk.
FilePos	A függvény hívásával valamely típus nélküli vagy típusos állomány pozícióját kérdezhetjük le.
GetDir	Az eljárás - karakterláncként - az adott meghajtó aktuális katalógusának teljes útvonalát adja vissza.
IOResult	A függvény az ellenőrzött be- és kivittelek hibakódját adja vissza.
MkDir	Alkönyvtárat hoz létre a megadott néven.
Reaname	Az eljárás átnevezi valamelyik lemezes állományt.
Reset	Az eljárás megnyit egy külső állományt, amelyet előzőleg az AssignFile eljárással valamely fizikai állományhoz rendeltünk.
Rewrite	Az eljárás létrehoz és megnyit egy fizikai állományt, amelyet előzőleg az AssignFile eljárással valamely állományváltozóhoz kell rendelni.
Rmdir	Törli a karakterlánccal megadott üres katalógust.
Seek	Az eljárás egy megnyitott állomány paraméterrel megadott komponensére viszi az állománymutatót.
Truncate	Az eljárás az aktuális pozíción levágja az állományt.

Sorrenddel kapcsolatos rutinok

Dec	Az eljárás a paraméterként megadott változó értékét eggyel csökkenti ($x:=x-1$), vagy a másodikként megadott paraméterrel csökkenti ($x:=x-n$).
Inc	Az eljárás a változó értékét eggyel ($x:=x+1$), vagy a második paraméter értékével ($x:=x+n$) növeli.

Odd	Megvizsgálja, hogy a megadott numerikus érték páratlan szám-e (ha páratlan, True értéket ad vissza).
Pred	A függvény a megadott sorszámozott kifejezés sorszám szerinti előző értékét adja vissza.
Succ	Az átadott sorszámozott kifejezés sorszám szerint következő értékét adja vissza.

Text-fájl rutinok

Append	Az eljárással bővítésre nyithatunk meg egy szöveges állományt.
Eoln	Az eljárás True értéket ad vissza, ha az állománymutató az állomány valamely sorának végén van.
Flush	Az eljárás hívásával a megnyitott szöveges állomány pufferének tartalmát kiírathatjuk a lemezre.
Read	Az eljárás valamely megnyitott típusos állományból egy vagy több komponenst beolvas a megadott változóba. Az eljárás másik formája - az opcionális Állomány paraméter nélkül - a billentyűzetről olvas a megadott változóba.
Readln	Az eljárás a Read bővítése. Valamely megnyitott szöveges állományból olvas be egy vagy több komponenst a változóba, majd automatikusan a sor végére ugrik.
SeekEof	Az eljárás True értéket ad vissza, ha az állománymutató az állomány végén van.
SeekEoln	Ugyanaz, mint az Eoln, de a pozíció vizsgálata előtt átlépi valamennyi szóközt és tabulátor karaktert.
SetTextBuf	Az eljárás lefoglal a memóriában egy területet valamely szöveges állományváltozó számára.
Write	Az eljárás valamely megnyitott, típusos állományba írja egy vagy több változó tartalmát. Az állománymutató valamennyi kiírás esetén eggyel továbblép. Második formájában az eljárás egy vagy több változó tartalmát valamely megnyitott szöveges állományba írja.
Writeln	Az eljárás valamely megnyitott állományba beír egy vagy több változót, és a beírást soremeléssel zárja.

A dinamikus kiosztás rutinjai

Dispose	Az eljárás felszabadítja a lefoglalt memóriát.
FreeMem	Az eljárás felszabadít egy lefoglalt memóriaterületet.
GetMem	Az eljárás lefoglal egy dinamikus területet a memóriában, és ennek kezdőcímét a változóban adja vissza.
New	Az eljárással a memóriából foglalhatunk le területet valamely tetszőleges mutatóváltozó számára.

Flow-control rutinok

Break	Befejez egy for, while vagy repeat ciklust.
Continue	Folytat egy for, while vagy repeat ciklust.
Exit	Azonnal kilép az aktuális blokkból.
Halt	Megállítja a program futását, és visszatér az operációs rendszerbe.
RunError	Megállítja a program futását.
<i>Vegyes rutinok</i>	
AddExitProc	Egy eljárást ad a futásidejű könyvtár exit eljáráslistájához.
BeginThread	.
ChDir	Az eljárással az aktuális lemezt és/vagy könyvtárt változtathatjuk meg.
EndThread	.
Exclude	Kivesz egy elemet a halmazból.
FillChar	Az adott karakterrel és az adott hosszúságban feltölti az adott kezdőcímű memóriaterületet.
Finalize	Felszabadítja a dinamikus változók által lefoglalt memóriát.
Hi	A függvény egy numerikus kifejezés felső bájtját adja vissza.
Include	Bevesz egy elemet a halmazba.
Initialize	Inicializálja a változók számára a memóriát.
Lo	A függvény valamely numerikus kifejezés alsó bájtját adja vissza.
Move	Az eljárás a forrással meghatározott memóriacímről a célterületre másolja a megadott számú bájtot.
ParamCount	A függvény a programnak átadott parancssor paraméterek számát adja vissza.
ParamStr	Karakterláncként adja vissza a numerikus kifejezéssel meghatározott sorszámú parancssor paramétert.
Ptr	A függvény olyan mutatót ad vissza, amely a memória szegmens eltolás címére mutat.
Random	A függvény véletlen számot ad vissza.
Randomize	Az eljárás véletlen kiinduló helyzetbe állítja a véletlenszám-generátort.
SizeOf	A függvény a változó vagy az adattípus méretét adja vissza bájtokban.
Slice	A függvény lehetővé teszi egy tömbből egy megadott részt (szeletet) elkülöníteni.
Str	Egy numerikus változót stringgé konvertál.
Swap	A függvény felcseréli a kifejezés alsó és felső bájtját.
TypeInfo	Az adott változóról ad típus információt.
UniqueString	.
UpCase	Nagybetűssé konvertálja a karaktert.
Val	Egy stringváltozót numerikussá konvertál.

Pointer és címzés rutinok

Addr	A függvény eredménye egy mutató, amely az argumentum memóriabeli címét tartalmazza.
Assigned	Megvizsgálja, hogy egy mutató vagy egy eljárás-béli változó értéke nil-e.
<i>Átalakító rutinok</i>	
Chr	A függvény visszaadja a megadott számnak - mint ASCII kódnak - megfelelő karaktert.
High	Az argumentum legmagasabb értékét adja vissza.
Low	Az argumentum legalacsonyabb értékét adja vissza.
Ord	A függvény a megadott kifejezés sorszámát adja vissza.
<i>Típusnélküli fájlok rutinjai</i>	
BlockRead	Az eljárással valamely típus nélküli, megnyitott állományból meghatározott számú adatrekordot olvashatunk be egy változóhoz rendelt memóriaterületre.
BlockWrite	Az eljárással meghatározott számú adatrekordot írhatunk ki egy állományba a memória valamely fenntartott területéről.
<i>OLE string segédrutinok</i>	
OleStrToString	The OleStrToString function copies data received from an OLE function or method into a newly allocated native Delphi string. .
OleStrToStrVar	The OleStrToStrVar copies OLE string data into an existing Delphi string variable. .
StringToOleStr	The StringToOleStr function allocates memory and copies string data from a native Delphi string into a format that can be passed to OLE functions. .
<i>Karakter-típus rutinok</i>	
StringToWideChar	Egy ANSI stringet UNICODE stringgé konvertál.
WideCharLenToString	Az UNICODE karaktereket ANSI-ba konvertálja.
WideCharLenToStrVar	Egy WideChar (Unicode) stringet egy bájtos karakteres adattá konvertálja, és az eredményt stringváltozóban helyezi el.
WideCharToString	Egy UNICODE stringet ANSI stringgé konvertál.
WideCharToStrVar	Egy WideChar (Unicode) stringet egy bájtos karakteres adattá konvertálja, és az eredményt stringváltozóban helyezi el. A forrásnak 0-végű stringnek kell lenni.
<i>Egyéb segédrutinok</i>	
VarArrayCreate	Adott méretű és típusú tömbváltozót állít elő.
VarArrayDimCount	Az adott változó méretét adja meg.
VarArrayHighBound	Egy adott méretű tömbváltozó felső határával tér vissza.
VarArrayLock	Zárol egy tömbváltozót.
VarArrayLowBound	Egy adott méretű tömbváltozó alsó határával tér vissza.
VarArrayOf	Az adott tömböt specifikált elemekkel adja vissza.
VarArrayRedim	Az adott tömböt átméretezi a felső határnak megfelelően.

VarArrayUnLock	Megszünteti egy tömbváltozó zárolását.
VarAsType	Egy változót adott típusúra konvertál.
VarCast	Egy változót adott típusúra konvertál, és tárolja azt.
VarClear	Töröl egy adott változót.
VarCopy	Adott változót egy specifikált változóvá konvertál.
VarFromDateTime	Dátumot és időt tartalmazó változót ad vissza.
VarIsArray	Megmondja, hogy a változó tömb típusú-e.
VarIsEmpty	Megmondja, hogy a változó tömb megnyitott-e.
VarIsNull	Megmondja, hogy a változó értéke nulla-e.
VarToDateTime	Egy változót dátum és idő értékekké konvertál.
VarType	Egy változót adott típusúra konvertál, és tárolja azt.

Változók és tipizált konstansok a System egységben

Változók

Változó	Típus	Változó	Típus
DLLProc	Pointer	Input	Text
IsConsole	Boolean	IsLibrary	Boolean
Output	Text	Unassigned	Variant

Konstansok

Konstans	Típus	Konstans	Típus
CmdLine	PChar	CmdShow	Integer
ErrorAddr	Pointer	ErrorProc	Pointer
ExceptionClass	TClass	ExceptProc	Pointer
ExitCode	Integer	ExitProc	Pointer
FileMode	Byte	HInstance	LongInt
InOutRes	Integer	Null	Variant
RandSeed	LongInt		

7.1.2. A Math unit

A Math egységben olyan - a System-ből már a Turbo Pascal-ban is hiányolt - matematikai, statisztikai és pénzügyi elemek vannak, amelyek megkönnyítik a programozó munkáját. A jelentős számú rutin mellett itt található az **EInvalidArgument** objektum és a **TPaymentTime** típus is. A következő táblázat a Math egység legfontosabb rutinait ismerteti.

A Math egység eljárásai és függvényei

ArcCos	Az argumentum arkusz koszinuszát adja vissza.
ArcCosh	Az argumentum arkusz koszinusz hiperbolikusát adja vissza .

ArcSin	Az argumentum arkusz szinuszt adja vissza.
ArcSinh	Az argumentum arkusz szinusz hiperbolikuszt adja vissza.
ArcTan2	Az ArcTan(Y/X) értékét számítja ki.
Ceil	A függvény a numerikus kifejezést a pozitív végtelen irányában a legközelebbi egészre kerekíti.
Cosh	Az átadott argumentum koszinusz hiperbolikuszt adja vissza.
Cotan	Az argumentum kotangensét adja vissza.
CycleToRad	A függvény a körciklusokat radiánra konvertálja.
DegToRad	A fokban mért szöget radiánra számítja át.
Floor	A függvény a numerikus kifejezést a negatív végtelen irányában a legközelebbi egészre kerekíti.
Frexp	Az eljárás elkülöníti a mantisszát és a kitevőt.
GradToRad	Az úfokokban megadott szöget számítja át radiánra (a kör 400 újfokra van felosztva).
Hypot	A függvény a derékszögű háromszög átfogójával tér vissza.
LnXP1	Az X+1 érték természetes alapú logaritmusával tér vissza.
Log2	Valamely numerikus kifejezés kettes alapú logaritmusát adja vissza.
Log10	A numerikus kifejezés tízes alapú logaritmusát adja vissza.
LogN	A numerikus kifejezés N alapú logaritmusát adja vissza.
MaxValue	Az adatok legnagyobb értékével tér vissza.
Mean	Az adatok számtani átlagával tér vissza.
MeanAndStdDev	Kiszámítja egy menetben a számtani átlagot és a szórást.
MinValue	Az adatok legkisebb értékével tér vissza.
MomentSkewKurtosis	Az eloszlási görbe különböző paramétereivel tér vissza.
Norm	A négyzetes középértékkel tér vissza.
RadToCycle	A radiánban megadott értéket körciklussá számítja át.
RadToDeg	A radiánban megadott értéket fokokra számítja át.
RadToGrad	A radiánban megadott értéket újfokra számítja át.
SinCos	Egy menetben kiszámítja az argumentum szinuszt és koszinuszt.
Sinh	Az átadott argumentum szinusz hiperbolikuszt adja vissza.
StdDev	A szórás értékével tér vissza.
Sum	Kiszámítja az adattömb összegét.
SumOfSquares	Az adatok négyzetösszegével tér vissza.
SumsAndSquares	Az adatok összegével és négyzetösszegével tér vissza.
Tan	Az átadott argumentum tangensét adja vissza.
Tanh	Az átadott argumentum tangens hiperbolikuszt adja vissza.

TotalVariance	Kiszámítja a négyzetes középeltérést.
Variance	Kiszámítja az egyszerű eltérést.

7.1.3. A SysUtils unit

A SysUtils egység tartalmazza az Exception osztály (a kivétel-, azaz a hibakezelés) deklarációját, string, idő és dátum rutinokat, valamint különböző segédrutinokat. A következőkben a SysUtils unit fontosabb eljárásait és függvényeit tekintjük át.

A SysUtils egység eljárásai és függvényei	
<i>Dátum-idő rutinok</i>	
Date	Az aktuális dátummal tér vissza.
DateTimeToStr	Egy időértéket stringgé konvertál.
DateTimeToString	Egy időértéket stringgé konvertál.
DateToStr	Egy dátumértéket stringgé konvertál.
DayOfWeek	A hét aktuális napjával tér vissza.
DecodeDate	Az adott dátumot dekódolja.
DecodeTime	Az adott időt dekódolja.
EncodeDate	Az adott értéket visszakódolja dátum formátummá.
EncodeTime	Az adott értéket visszakódolja idő formátummá.
FormatDateTime	Egy dátumot és időt adott formátumra alakít.
Now	Az aktuális dátummal és idővel tér vissza.
StrToDate	Egy stringet dátum formátumra konvertál.
StrToDateTime	Egy stringet dátum/idő formátumra konvertál.
StrToTime	Egy stringet idő formátumra konvertál.
Time	Az aktuális időpont értékével tér vissza.
TimeToStr	Egy időformátumot stringgé konvertál.
<i>Stringformázó rutinok</i>	
FmtStr	Egy argumentum sorozatot (tömb) formáz, az eredmény a Result paraméterben.
Format	Egy argumentum sorozatot (tömb) formáz, az eredmény Pascal string.
FormatBuf	Megformáz egy argumentum sorozatot.
StrFmt	Megformáz egy argumentum sorozatot.
StrLFmt	Megformáz egy argumentum sorozatot, az eredmény tartalmaz egy mutatót a célpufferre.
<i>Fájlkezelő rutinok</i>	
ChangeFileExt	Megváltoztatja az állomány kiterjesztését.
DateTimeToFileDate	Egy Delphi dátum formátumot DOS-os dátumformátumra konvertál.
DeleteFile	Töröl egy fájlt.

DiskFree	Visszaadja a szóban forgó meghajtóban lévő lemez szabad területének méretét bájtokban.
DiskSize	A függvény a megadott meghajtóban lévő lemez teljes kapacitását adja vissza.
ExpandFileName	Egy stringgel tér vissza, amely a teljes útvonal- és fájlnevet tartalmazza.
ExpandUNCFileName	Egy stringgel tér vissza, amely a teljes útvonal- és fájlnevet tartalmazza.
ExtractFileDir	A meghajtó és az adott fájlt tartalmazó könyvtár nevével tér vissza.
ExtractFileDrive	A teljes hosszúságú útvonal-megnevezés meghajtó részével tér vissza.
ExtractFileExt	Az állomány kiterjesztésével tér vissza.
ExtractFileName	Az állomány nevével tér vissza.
ExtractFilePath	Az adott fájl útvonalának megnevezésével tér vissza.
FileAge	A fájl korával tér vissza.
FileClose	Bezárja az adott fájlt.
FileCreate	Adott néven fájlt hoz létre.
FileDateToDateTime	A DOS-os dátum formátumot Delphi formátumra konvertálja.
FileExists	Ha a fájl létezik, True értékkel tér vissza.
FileGetAttr	Az állomány attribútumaival tér vissza.
FileGetDate	Az állomány DOS-os dátum-idő adattal tér vissza.
FileOpen	Megnyit egy adott fájlt, adott módon.
FileRead	Olvas az adott fájlból.
FileSearch	Könyvtárakon keresztül keresi a megadott fájlt.
FileSeek	A fájl jelenlegi helyzetét módosítja.
FileSetAttr	Beállítja a fájl attribútumait.
FileSetDate	Beállítja a fájl DOS-os dátum-idő értékét.
FileWrite	Ír a megadott fájlba.
FindClose	Megszünteti a FindFirst/FindNext parancssorozatot.
FindFirst	Az eljárás a megadott katalógusban keres egy állományt.
FindNext	Megismétli a FindFirst eljárással elindított állománykeresést.
RenameFile	Átnevez egy fájlt.

Kivételkezelő rutinok

Abort	Előidéz egy szünet kivételt.
ExceptAddr	Az aktuálisan kiváltott kivétel címével tér vissza.
ExceptObject	Az aktuális kivételre utalással tér vissza.
OutOfMemoryError	Kiváltja az OutOfMemory kivételt.
ShowException	Egy kivételre utaló üzenetet jelenít meg.

Lebegőpontos konverziós rutinok

FloatToDecimal	A lebegőpontos értéket decimális megjelenésűre konvertálja.
-----------------------	---

FloatToStrF	A lebegőpontos értéket string megjelenésűre konvertálja.
FloatToStr	Egy lebegőpontos értéket string megjelenésűre konvertál.
FloatToText	Az adott lebegőpontos értéket decimális megjelenésűre konvertálja.
FloatToTextFmt	Az adott lebegőpontos értéket decimális megjelenésűre konvertálja.
FormatFloat	A lebegőpontos értéket a Format stringben tárolt formátum szerint formázza.
StrToFloat	Az adott stringet lebegőpontossá konvertál.
TextToFloat	A nulla végű stringet lebegőpontos értékke konvertál.
Egyéb rutinok	
AddExitProc	Az adott eljárást hozzáadja a futásidejű könyvtár exit eljáráslistájához.
AllocMem	A függvény kiosztja a heap adott méretű blokkját.
AnsiCompareStr	Nagy-kisbetűre érzékenyen összehasonlítja két karakterláncot.
AnsiCompareText	Összehasonlítja két karakterláncot.
AnsiLowerCase	Az adott string összes karakterét kisbetűssé konvertálja.
AnsiUpperCase	Az adott string összes karakterét nagybetűssé konvertálja.
Beep	Az eljárás a Windows API MessageBeep-jét hívja meg.
CompareStr	Nagy-kisbetűre érzékenyen összehasonlítja két karakterláncot.
CompareText	Összehasonlítja két karakterláncot.
CreateDir	A függvény létrehoz egy új könyvtárat.
GetCurrentDir	Visszatér egy karakterlánccal, ami az aktuális könyvtárat tartalmazza.
GetFormatSetting	Visszaállítja az összes adat- és számformátumú változót az alapértelmezett értékre.
IntToHex	A függvény egy stringgel tér vissza, amely az adott szám hexadecimális változatát tartalmazza.
IntToStr	Egy Integer típusú számot karakterlánccá konvertál.
IsValidIdent	True értékkel tér vissza, ha az adott karakterlánc érvényes Pascal azonosító.
LowerCase	Az adott karakterláncot kisbetűssé változtatja (csak 7 bites ASCII karakterek esetén).
NewStr	A függvény a lefelé kompatibilitásról gondoskodik.
RemoveDir	Töröl egy létező üres könyvtárt.
SetCurrentDir	Beállítja az aktuális könyvtárt.
StrToInt	Egy Integer típusú számot tartalmazó stringet számmá konvertál.
StrToIntDef	Stringet számmá konvertál.
SysErrorMessage	Egy olyan hibaüzenetet tartalmazó stringgel tér vissza, amely megegyezik az adott operációs rendszer hibakódjával.
Trim	A függvény levágja az adott string elejéről és végéről a szóközöket és a vezérlő karaktereket.
TrimLeft	Levágja az adott string elejéről a szóközöket és a vezérlő karaktereket.

TrimRight	Levágja az adott string végéről a szóközöket és a vezérlő karaktereket.
UpperCase	Az adott karakterláncot nagybetűssé változtatja.

7.2. A VCL unitjai

A *Video Component Library*-ben van a Delphi összes komponense és az űrlapok elődjei. Lehet mondani, ez a Delphi lényege. A komponensek a VCL egységeiben vannak deklarálva.

Amikor az űrlapon elhelyezünk egy komponenst, a Delphi automatikusan beilleszti a *uses* kulcsszó után a megfelelő unitot. Ezért itt most elegendő csupán felsorolni a létező VCL unitokat.

A Visual Component Library standard egységei			
1.	Buttons unit	2.	Classes unit
3.	ClipBrd unit	4.	ComCtrls unit
5.	Controls unit	6.	DB unit
7.	DBCtrls unit	8.	DBGrids unit
9.	DBLogDlg unit	10.	DBLookup unit
11.	DBTables unit	12.	DDEMan unit
13.	Dialogs unit	14.	DsgnIntf unit
15.	ExtCtrls unit	16.	FileCtrl unit
17.	Forms unit	18.	Graphics unit
19.	Grids unit	20.	IniFiles unit
21.	Mask unit	22.	Menus unit
23.	Messages unit	24.	MPlayer unit
25.	OutLine unit	26.	OleAuto unit
27.	OleCtrls unit	28.	OleCtrls unit
29.	Printers unit	30.	Registry unit
31.	Report unit	32.	StdCtrls unit
33.	TabNotBk unit	34.	Tabs unit

Dr. Szabó László: A Delphi 2.0-ról röviden
Miskolc, 2000 © Szabó László