

Gyakorlatok

1. Készítsük el a 3. fejezetben leírt „Első programunkat”.
2. Próbáljunk meg különféle komponenseket elhelyezni az ablakunkban, majd futtassuk le a programot és figyeljük, hogyan jelennek meg, ill. milyen értékeket tudunk megadni nekik.
3. Hozzunk létre egy alkalmazást, amelyen két gomb lesz (Kiírás, Kilépés) és egy címke. Az egyik megnyomásakor átírja a címke feliratát (ezt a programkódban: **Label1.Caption := 'Uj felirat'**; formában adhatjuk meg), a másik gomb megnyomására kilép a programból.
4. Készítsünk programot, amely egy címkét és egy nyomógombot tartalmaz (Sorsolás). A gomb megnyomásakor a számítógép a címke feliratába írjon ki 5 véletlenszerű lottószámot 1-től 90-ig (ilyen véletlenszámokat a **random(90)+1** függvénnyel tudunk generálni, majd a számot az **IntToStr()** függvénnyel tudjuk szöveggé alakítani). Ne felejtsük el előtte beállítani a véletlenszám generátort (**randomize;**), hogy minden indítás után ne kapjuk ugyanazokat a számokat. A program tervezésekor állítsuk be az Objektum felügyelőben, hogy a címke betűmérete nagyobb legyen (ezt a címke **Font.Size** tulajdonságával tehetjük meg).
5. Próbáljunk meg készíteni egy alkalmazást, amelyen három gomb (Páros, Páratlan, Fibonacci) és egy címke szerepel. Az első gomb megnyomásakor a címke feliratát átírja az első 10 páratlan számra (1, 3, 5, ...), a második megnyomásakor az első 10 páros számra (2, 4, 6, ...), a harmadik megnyomásakor kiírja az első 10 Fibonacci számot (1, 1, 2, 3, 5, 8, ... - mindegyik szám az előző kettő összege). A számokat ciklus segítségével próbáljuk meg generálni.
6. Jelenjen meg a képernyőn két nyomógomb Belevágok! és *Kilépés* felirattal. A belevágok gombra való kattintás után jelenjen meg az *Üdvözöllek a programozás világában!* üzenet. (tulajdonság: **Caption**, esemény: **OnClick**, metódus: **Form1.Close**)
7. Jelenjen meg a képernyőn egy gomb *Kilép* felirattal. Ha a felhasználó rákattint, jelenjen meg egy üzenet *Meggondolta?* kérdéssel. Majd ha „leokézza”, egy másik üzenet *Biztos benne?* kérdéssel, stb. Legalább ötször egymás után. (ismétlés Turbo Pascalból: egy **i** változó deklarációja a unit implementation részében, **case** elágazás használata)
8. Bővítsük ki az előző feladatot úgy, hogy az ablak helye minden gombnyomás után máshol legyen a képernyőn véletlenszerűen kiválasztva. (új tulajdonságok: **Left**, **Top**, **Width**, **Height**, **Screen.Width**, **Screen.Height**, események: **OnCreate**, ismétlés Turbo Pascalból: **random**, **randomize**)
9. Próbáljuk meg a programot úgy átírni, hogy ha a felhasználó máshogy (X-szel a jobb felső sarokban, ALT+F4-gyel, stb.) akarja bezárni az alkalmazást, akkor se tudja és jelenjen

meg neki ebben az esetben az *Így nem fog menni, csak a gombbal!* felirat. (új esemény: **Form1.OnCloseQuery**, ennek **CanClose** paramétere)

10. A képernyőn jelenjen meg egy adatlap (ábra). Ha az *Edit1* beviteli mezőbe beírjuk a nevünket, akkor a *Label3* címkébe kerüljön be a bevitt adat! (új tulajdonságok: **Edit1.Text**, **Font.Style** halmaz)

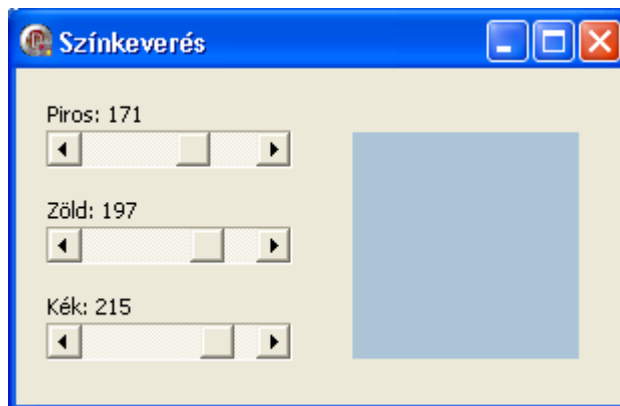


11. Bővítsük az előző feladatot egy újabb kérdéssel (*Életkora:*), ami csak akkor jelenjen meg, amikor a felhasználó válaszolt az előző kérdésre. (új tulajdonság: **Visible**)
12. Jelenjen meg a képernyőn két beviteli mező és egy *Csere* feliratú gomb. A gombra kattintáskor a két beviteli mező tartalma cserélődjön meg.
13. Zöldséges standunkon háromféle terméket árulunk: burgonyát, répát és káposztát. Egységárukat egy-egy címke jeleníti meg, a vásárolt mennyiséget egy-egy beviteli mezőbe írjuk. Egy gomb megnyomása után számítsuk ki és jelenítsük meg a fizetendő összeget! (új tulajdonság: **Font.Size**, függvények: **StrToFloat**, **FloatToStr**, **Round**)
14. A programablak bal felső sarkában jelenjen meg egy nyomógomb. Ha a felhasználó rákattint, menjen a gomb a jobb felső sarokba, majd a jobb alsó, bal alsó, végül újra a bal felső sarokba, stb. (új tulajdonságok: **Form1.ClientWidth**, **Form1.ClientHeight**)
15. Találjuk ki a gép által gondolt egész számot tippeléssel, ha a gép minden tipp után megmondja, hogy az kicsi vagy nagy! (új tulajdonságok: **Button1.Default**, **Button1.Cancel**, új metódus: **Edit1.SelectAll**)
16. Készítsünk programot elektronikus pizza rendeléshez! A kért összetevőket jelölőnégyzetekkel lehessen megadni. A program ezek alapján automatikusan a jelölés közben jelenítse meg a pizza árát! (új tulajdonságok: **CheckBox1.Checked**, saját eljárás létrehozása, az összes **CheckBox** **OnClick** eseményére ugyanakkor az eljárásnak a megadása, mint az **CheckBox1**-nek)
17. Készítsünk szoftvert kávé automatához! Rádiógombokkal lehessen megadni az italt (kávé, tea, kakaó), jelölőnégyzetekkel a hozzávalókat (citrom, cukor, tej, tejszín). A szoftver számolja ki és jelenítse meg a fizetendő összeget! Teához ne lehessen tejszínt, kávéhoz

citromot, kakaóhoz se citromot, se tejszínt kérni! (ábra) (új tulajdonságok: **Enabled**, **RadioButton1.Checked**)



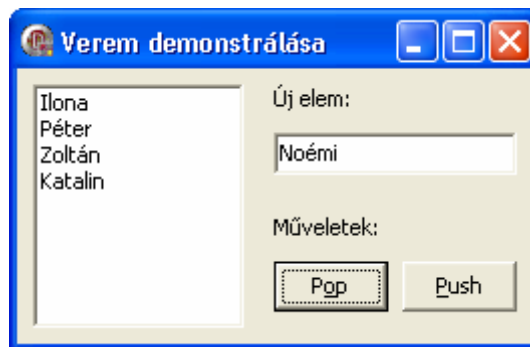
18. Színkeverés RGB színmodell alapján. A képernyőn jelenjen meg három görgetősáv, amely az RGB színmodell három alapszínét állítja be 0 és 255 között. A kikevert szín egy címke háttérében jelenjen meg! (ábra) (új tulajdonságok: **ScrollBar1.Min**, **ScrollBar1.Max**, **ScrollBar1.Position**, **Form1.DoubleBuffered**, új esemény: **OnChange**, új Windows API függvény: **RGB**)



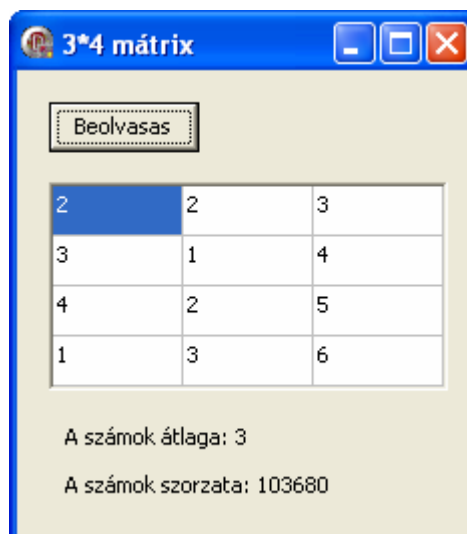
19. Készítsünk csúszkás számológépet! A kért számot egy-egy vízszintes görgetősáv tologatásával lehessen bevinni, majd a megfelelő nyomógombra (feliratuk: *Összeadás*, *Kivonás*, *Szorzás*, *Osztás*) való kattintáskor jelenjen meg egy címkében az eredmény!
20. Készítsünk programot, amely egy ListBox-ot tartalmaz. Ha rákattintunk a form-ra egérrel, duplán rákattintunk, vagy megnyomunk egy billentyűt, írassuk ki a ListBox-ba az **OnMouseDown**, **OnClick**, **OnMouseUp**, **OnDbiClick**, **OnKeyDown**, **OnKeyPress**, **OnKeyUp**

események neveit olyan sorrendben, ahogy következnek. (tulajdonság: **Form1.KeyPreview**, metódus: **ListBox1.Items.Add**)

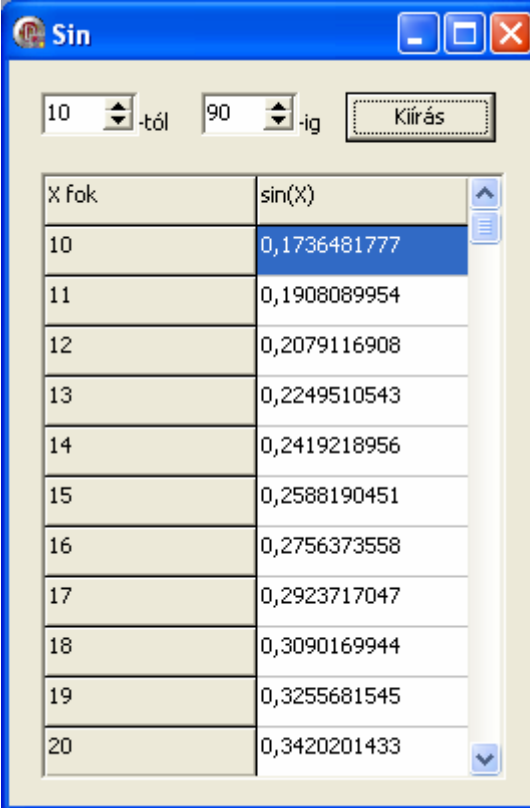
21. Verem demonstrálása: készítsünk egy alkalmazást, amely tartalmaz egy listát és egy beviteli mezőt. A beviteli mező adata a Push gomb hatására kerüljön a lista tetejére, míg a Pop gomb hatására a lista felső eleme kerüljön a beviteli mezőbe, és törlődjön a listáról (ábra). A lista legfeljebb 10 elemű lehet. Ha a lista tele van (Full) vagy üres (Empty), akkor a megfelelő gomb hatására kapjunk hibajelzést (üzenet ablak)! (új tulajdonság: **ListBox1.Items[0]**, új metódusok: **ListBox1.Items.Insert**, **ListBox1.Count**, **ListBox1.Items.Delete**)



22. Sor bemutatása: a képernyőn jelenjen meg egy lista és egy beviteli mező. A Push gomb hatására a beviteli mező tartalma kerüljön a lista tetejére, a Pop gomb hatására a lista alsó eleme kerüljön a beviteli mezőbe. A lista legfeljebb 10 elemű lehet. Ha a lista tele van vagy üres, akkor a megfelelő gomb generáljon hibajelzést!
23. Olvassunk be az InputBox függvény segítségével egy 3*4-es mátrixot, melyet egy StringGrid komponensbe jelenítsünk meg. Számoljuk ki az elemek átlagát és szorzatát.




24. Írjuk ki a Sin függvény értékeit táblázatosan egy StringGrid komponensbe előre megadott intervallumban fokenként. Ne engedjük, hogy az intervallum alsó értéke nagyobb legyen, mint a felső.



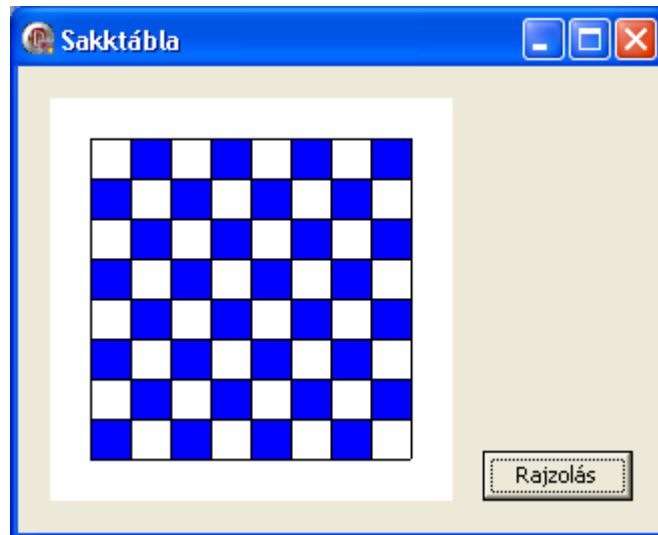
X fok	sin(X)
10	0,1736481777
11	0,1908089954
12	0,2079116908
13	0,2249510543
14	0,2419218956
15	0,2588190451
16	0,2756373558
17	0,2923717047
18	0,3090169944
19	0,3255681545
20	0,3420201433

25. Olvassunk be egy 3*3-as mátrixot, majd ellenőrizzük, hogy a mátrix bűvös négyzet-e, azaz sorainak, oszlopainak és átlóinak összege azonos-e (az eredményt egy MessageBox segítségével jelenítsük meg). Az alábbi példában szereplő mátrix bűvös négyzet.

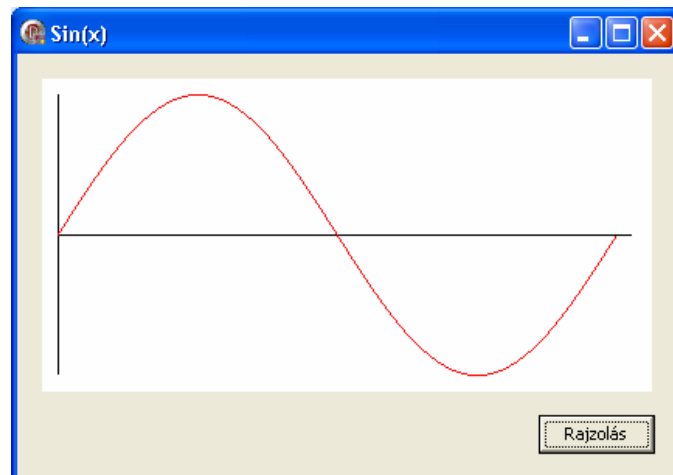


7	9	-1
-3	5	13
11	1	3

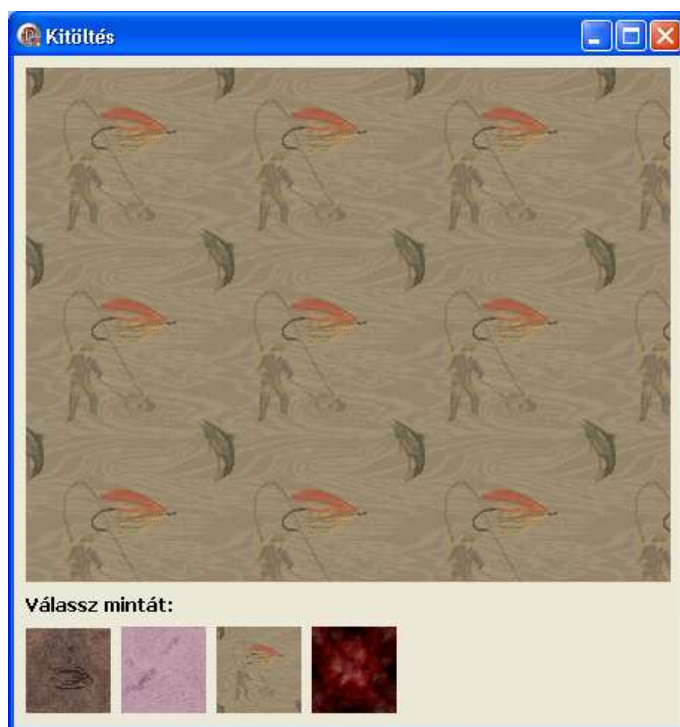
26. Programunk írja ki mely billentyűt kell lenyomni, és írja ki a megtalálás idejét. Folyamatosan értékelje sebességünket (átlagos sebesség egy billentyű lenyomására).
27. Készítsünk programot, amely egy nyomógomb megnyomásakor kirajzol egy sakktáblát egy image komponensre.



28. Készítsünk egy alkalmazást, amely egy nyomógomb megnyomásakor kirajzolja egy image komponensbe a $\sin(x)$ függvény grafikonját.



29. Készítsünk egy alkalmazást, amely tartalmaz egy nagyobb méretű üres Image komponenset és négy kisebb Image komponenset, melyekben különböző háttérmintákat jelenítünk meg. Ha valamelyik háttérmintára rákattintunk egérrel, a program töltsse ki a megadott mintával a nagyobb Image komponenset.



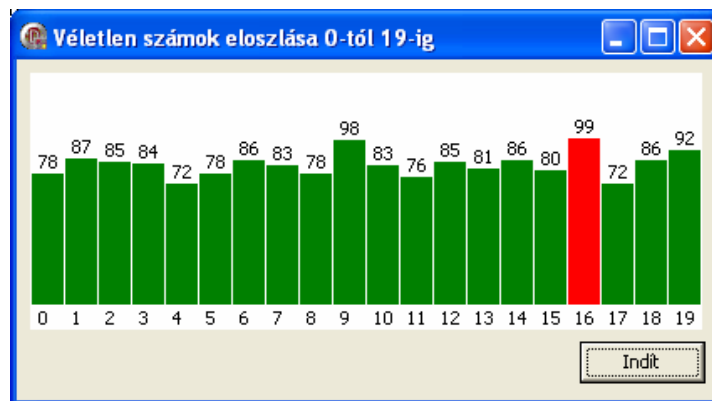
30. Készítsünk egy "pecsételő programot". A program tartalmazzon néhány kép kicsinyített változatát. Ha valamelyik képre rákattintunk egérrel, majd a rajzlapra kattintunk (nagyobb méretű Image komponens), akkor minden egyes kattintás helyére a program "pecsételje oda" a kiválasztott rajzot. A rajzot úgy rakjuk ki a rajzlapra, hogy a kattintás helye (koordinátái) a kirajzolandó kép közepén legyen. Az alkalmazásunk tartalmazzon még egy nyomógombot is, mellyel letörölhetjük a rajzlapot.



31. Készítsünk alkalmazást, amely szemlélteti a véletlen számok eloszlását. A számítógép 0 és 19 közötti véletlen számokat generáljon ki és számolja az egyes számok előfordulását, melyet oszlopokkal szemléltessen. Mindegyik oszlop fölé írja oda, hogy mennyiszer volt az adott szám kigenerálva. Amelyik szám(ok) az adott pillanatban a legtöbbször fordulnak elő, azokat zöld oszlop helyett mindig pirossal szemléltessük. A számok generálását egy nyomógomb segítségével lehessen elindítani. Ha újra megnyomjuk a nyomógombot, a számok generálása előlről kezdődjön. A program tehát a nyomógomb megnyomása után minden szám oszlopának magasságát beállítja nullára, majd:

- Kigenerál egy 0-19 közötti véletlen számot.
- Az adott szám oszlopának magasságát megnöveli egy pixellel és fölé kiír egyet nagyobb számot.
- Figyeli, melyik számok előfordulása a legnagyobb, ezeket piros oszloppal szemlélteti, a többit zölddel.
- Kigenerálja a következő véletlen számot...

A program a nyomógomb megnyomása után automatikusan működjön és növelje bizonyos időközönként (pl. 0,01 sec-ként) a kigenerált szám oszlopának magasságát mindaddig, amíg valamelyik nem éri el a 99-et. Ekkor a számok generálása álljon le.

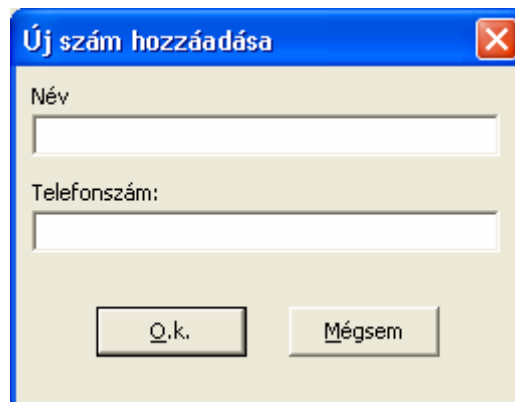
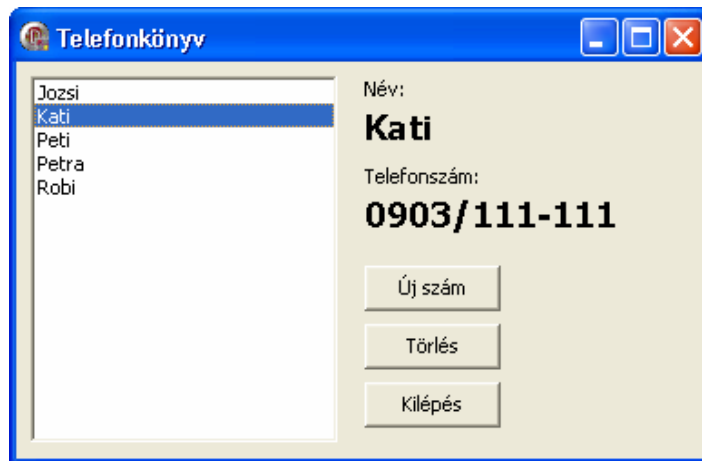


32. Készítsünk programot, amely tartalmazni fog egy Memo komponenst és három nyomógombot. Az első nyomógomb egy dialógusablak segítségével válasszon ki egy TXT fájlt, majd olvassa be a program a Memo komponensünkbe a fájl tartalmát. A második nyomógomb mentse el a fájlt (dialógusablakkal lehessen megadni a fájl nevét és helyét), a harmadik nyomógomb segítségével lehessen megváltoztatni a Memo komponens betűtípusát. Az alkalmazást bővítsük ki menüvel (MainMenu), ahonnan szintén elérhető legyen ez a három funkció.

33. Készítsünk telefonkönyvet. Az alkalmazás tartalmazzon egy ListBox-ot, melyben nevek találhatók ABC sorrendben. Ha valamelyik névre rákattintunk (kijelöljük), a jobb oldalon jelenjen meg a név és a hozzá tartozó telefonszám.

Az „**Új szám**” nyomógombra kattintáskor egy új (modális) ablakban kérjünk be egy nevet és egy telefonszámot, melyet helyezünk el a névsorban a megfelelő helyre (úgy, hogy a nevek ABC sorrendben maradjanak). A „**Törlés**” gombra kattintáskor a kijelölt nevet töröljük a névsorból. Ilyenkor a jobb oldalon a törölt elem után következő (ha nincs akkor az előtte levő) név jelenjen meg (ha nincs előtte levő sem, akkor a jobb oldalon ne jelenjen meg semmilyen név és telefonszám).

Az összes nevet és telefonszámot a programból való kilépéskor mentjük el egy külső állományba. A program indításakor olvassuk be ebből a fájlból a neveket.



34. Készítsünk alkalmazást, amely megjeleníti és folyamatosan mutatja (frissíti) az aktuális időt.



35. Készítsünk „csigák versenye” játékot. A csigák valamelyik gomb megnyomásával induljanak el. Mindegyik csiga véletlenszerű helyre menjen jobbra mindaddig, amíg valamelyik nem éri el az ablak jobb szélét. Ha az a csiga nyert, amelyre tippeltünk, akkor a pontszámunk növekedjen 3-mal, különben csökkenjen 1-gyel. A nyertes csiga színét egy MessageBox segítségével írjuk ki, majd a csigák álljanak újra a rajtvonalra, és újból lehessen tippelni valamelyik nyomógomb megnyomásával!

