4 Függelék

Ez a **Függelék** olyan **Visual Basic for Application** ismereteket tartalmaz, melyek nem szükségesek a tárgy teljesítéséhez, de az egyetemi évek alatt kiadott feladatok programozással történő megoldása során hasznosak lehetnek a hallgatók számára.

Ebben a fejezetben bemutatott **Rekordok és használatuk**, valamint a **UserForm használata** a VBA programozásnak a „Haladó szinttű" témái közé tartoznak.

E témák tanulása már feltételezi a Jegyzet előbbi fejezetei – **VBA programozási ismeretek** – készségszintű ismeretét, a Jegyzet **Feladatok** fejezetrészben kidolgozott gyakorlófeladatok egyéni megoldásait is, mert csak erre építve lehet a további ismereteket gyarapítani.

Ennek a „Haladó szinttű" ismeretanyagnak megtanulása, viszont rendkívül ajánlott mindazoknak, akiket a logikus gondolkodás „nem fáraszt", és szeretnének „komoly, látványos" prezentációkat alkotni.

A Visual Basic for Application (VBA) programnyelv, a **Windows Office** felhasználói programok beépített háttérnyelve. Simonyi Károly, (a Windows op. rendszer és az Office és az Excel megteremtője; lásd előbb) azzal a céllal építette be a rendszereiben, hogy a Felhasználók, – az egyéni feladataik megoldásához –, egyéni Felhasználói megoldásokat és megjelenéseket programozhassanak. A VBA programozás, a leggyakrabban használt az Excel táblázat szerkesztő program keretében van bemutatva.

A Jegyzet „egyedi szellemiségű összeállítása", a számítástechnika, egy másik világhírű tudósának, Dr. Kemény János György, 1960-as évekbeli munkásságának, – szerényen mondva – a folytatása.

Kemény János munkássága adta meg a lehetőséget arra, hogy az Egyetemi hallgatók, egyéni tanulással is elsajátíthassák a Számítógép használatot és annak Programozását is

E Jegyzet is arra törekszik, hogy a hallgatók, a Számítástechnika tárgy alapozó tudása után, egyénileg is, akár magas szintű programozásig is el tudjanak jutni.

A Jegyzet eleje is úgy van szerkesztve, hogy minden „kezdő" hallgatónak megmutassa az „indulás" minden fázisát, és itt a Függelékben bemutatott **UserForm** használata fejezet is, a szokásosnál hosszabban foglalkozik az „indulás" lépéseivel. A téma további tárgyalása, egyenként bemutatja a VBA-ban használható Controlokat – működésüket és programozásukat – mintapéldán keresztül. Ez egy „hiánypótló" összeállítás is, mert igazi irodalma, csak a (nagy) Visual Basicnak van, a VBA-nak csak alig-alig van, és azok is régiek és hiányosak.

Ezért, e Jegyzet írói nagy gondot fordítottak arra, hogy egy „alapozó – de átfogó" tudásanyagot adjanak.

Lásd: a Jegyzet 2.2 Kemény János György pont fejezetből, részlet.

Kemény János György

(Budapest, 1926. május 31. – New Hampshire, USA, 1992. december 26.) matematikus, számítás technikus.

Kemény János, az 1970-es, 1980-as években az Egyesült Államokban Teller Edén kívül valószínűleg **a legismertebb magyar-amerikai tudós volt**.

Középiskolai tanulmányait New Yorkban fejezte be, majd a Princetoni Egyetemen végzett matematikusként, majd 1949-ben doktorált logikából. A Kenti Egyetem munkatársa lett. Jellemző volt rá, hogy autója rendszámára, a „LOGIC" (LOGIKA) szót íratta rá. 27 évesen meghívták a Dartmouthi Főiskolára matematikaprofesszornak, s két év múlva a Matematikai Intézet vezetője lett.

1962-ben ő javasolta az egyetemi számítógépközpont megépítését is, melyet 1966-ban adták át.

Kemény János azon gondolkozott, hogyan tegye a számítógépet hozzáférhetővé egyszerre több használó számára, ezért kidolgozta és bevezette az időbeosztás módszerét: minden használó saját terminálján dolgozik, a központi számítógép pedig beosztja processzorának munkaidejét a használók közt. A számítógép-időbeosztás rendszerét először a Dartmouth Kollégium vezette be (1963), a rendszer kidolgozásáért Kemény János kapta az IBM legelső Robinson-díját (1991).

A múlt század közepén a FORTRAN volt a kutatók közt a legelterjedtebb nyelv, de nem volt elég emberszabású, ezért Kemény elhatározta, hogy egy interaktív nyelvet fejleszt ki, amelyik rögtön reagál a használó utasítására, így lehetővé teszi, hogy minden diák vagy felnőtt próba-szerencse alapon lépésről-lépésre építse föl, tapasztalja ki saját programját. Megfogalmazta elvárásait:

- A programozási nyelv legyen alkalmas eltérő célok kielégítésére.
- A nyelv kezdők számára is könnyű legyen.
- A magasabb szintű utasításokat csak később kelljen megtanulni.
- A használó és a gép között a nyelv legyen interaktív.
- Könnyen érthető hibajelzéseket adjon.
- A speciális gép-architektúra ismerete nélkül is lehessen használni.
- Óvja meg a használót a gép operációs rendszerének problémáitól.

Így született meg Kemény János és Tom Kurtz kezében 1964-ben a BASIC nyelv, melyről később így nyilatkozott:

„Nem csak azért teremtettem meg a BASIC et, hogy eggyel több számítógépes nyelv legyen. Azért csináltam, hogy a számítógép minden egyetemi hallgató (és minden diák) számára hozzáférhetővé váljék".

Az első BASIC program 1964. május 1-jén reggel 5 órakor futott.

4.1 Rekordok és használatuk

Irodalom: Peter G. Aitken: Programozás VISUAL BASIC 6 nyelven, „Kék Könyv” / 372 old.

4.1.1 Fájlok írása és olvasása

Ha adatainkat el akarjuk menteni, hogy a gép kikapcsolása után is megmaradjanak, akkor a lemezen tároljuk azokat. Ha olyan adatokat akarunk felhasználni, amit egy másik program hozott létre, akkor általában fájlból kell azokat beolvasnunk. A Basic nyelv számtalan utasítást és függvényt tartalmaz, melyek rendkívül rugalmassá teszik a fájlok használatát. Mivel ezek az utasítások és függvények a Basic nyelv részét képezik, ezért nincsenek közvetlen kapcsolatban a Visual Basic felhasználói felületével, illetve annak objektumaival és eseményeivel.

A Visual Basicben egy fájl használata három lépésből tevődik össze:

- A fájl megnyitása.
- Adat olvasása a fájlból, adat írása fájlba.
- A fájl bezárása.

Ez így elég egyszerűen hangzik, de e három lépés során számos nehézség merülhet fel. Először vessünk egy pillantást a Visual Basicben használható három különböző fájlelérési módra.

4.1.2 A fájlelérés típusai

Minden fájlban lévő adat, akárhonnan is származzon, akármi is legyen a tartalma, bájtsorozatként tárolódik. Az adat típusától, illetve a program igényeitől függően három lehetséges fájlelérési mód közül választhatunk. A programunk által használt fájlelérési mód nagymértékben befolyásolja, hogy milyen technikákkal olvashatunk és írhatunk adatot, és egyáltalán azt, hogy mit kezelünk adatként

4.1.2.1 Szekvenciális elérésű fájlok

Egy szekvenciális elérésű fájl változó hosszúságú rekordok sorozatát tárolja. Itt a rekord szó egyszerűen egy adategységet jelent. Például, ha egy **Integer** típusú változó értékét tároljuk a fájlban, akkor az egy rekord. Hasonlóan egy **String** típusú változó értéke egy másik rekord lesz. A rekordok változó hosszúságúak, mivel minden rekord méretét a tartalma határozza meg, nem pedig a fájl általános típusa. Más szóval a szekvenciális fájlok változó hosszúságú rekordokat tartalmazhatnak.

A szekvenciális fájlokban lévő adatot szövegnek tekintjük. Ez annyit jelent, hogy a fájl minden bájtja egy karakter. A szekvenciális szó azt jelzi, hogy az ilyen típusú fájlt mindig az elejéről kell olvasni, ahhoz, hogy a fájl közepéből olvashassunk, először ki kell olvasni minden azt megelőző elemet. Nincs egyszerű módja annak, hogy egy meghatározott sorszámú adatra ugorjunk. A szekvenciális fájlokban a numerikus (számszerű) adatok a hozzájuk tartozó karaktersorozatokkal

kerülnek tárolásra, a String adatok pedig macskakörmök között, Például a 123 nem bináris adatként, egy bájton, hanem a „123” szövegként kerül tárolásra.

A szekvenciális elérésű fájlokban minden sort egy rekordnak tekintünk, melyet egy. „kocsi-vissza” + „soremelés” karakterpár (CR+LF) zár. Egy rekord tartalmazhat több, de akár nulla karaktert is, és a rekordok hossza nem függ a többi rekordtól. Ez a megközelítés akkor lehet hasznos, ha eltérő hosszúságú adatokkal kell dolgozni, és csak szekvenciális feldolgozást akarunk megvalósítani.

4.1.2.2 Véletlen elérésű fájlok

Egy véletlen elérésű (*random Access*) fájl állandó hosszúságú rekordok sorozataként tárolja az adatokat (a rekordok mérete állandó). A véletlen elérésű fájlban a rekordok sorszámokkal rendelkeznek. A sorszámozás egytől kezdődik. E fájlokban egy megadott sorszámú rekordot azonnal elérhetünk, anélkül, hogy az előtte lévő adatokon végig kellene, fussunk.

Például egy 2000 rekordból álló fájlból kiolvashatjuk az 1199-dik rekordot anélkül, hogy előzőleg kiolvasnánk az összes többit 1-től 1198-ig. A véletlen elérésű fájlokban szöveget és számokat egyaránt tárolhatunk. A szöveg karakteresen, a számok pedig speciális bináris formában tárolódnak.

A véletlen elérésű fájlokban minden rekord azonos számú bájtból, illetve karakterből áll. Minden rekord tartalmaz egy vagy több mezőt, melyek közül mindegyiknek rögzített a hossza. Ebből következik, hogy egy rekord mérete megegyezik a mezők hosszainak összegével. A fájl rekordjainak, illetve a rekordok mezőinek hosszát a programban határozzuk meg, a fájllétrehozásakor. Eltérően a szekvenciális fájloktól, a véletlen elérésű fájlok nem használnak szeparátorokat (olyan jeleket, melyek elválasztják a rekordokat egymástól, mint például a szekvenciális fájlnál a CR+LF karakterpár) az egyes mezők és rekordok kezdetének és végének jelölésére. Az állandó hosszúság teszi lehetővé, hogy a fájlban egy tetszőleges sorszámú rekordra ugorhassunk.

Például ha a rekordméret 100 bájt, akkor tudni lehet, hogy az 55. rekord az 5401. bájtnál kezdődik.

4.1.2.3 Bináris fájlok

A bináris fájlok esetében a fájlok szerkezetére semmi sincs megadva. Semmiféle rekordméret, vagy szeparátor nincs kikötve, a fájlt egyszerűen csak bájtok sorozatának tekintjük. A bináris hozzáférés lehetővé teszi, hogy a fájl egyes bájtjait a fájl típusától függetlenül manipuláljuk. Eltérően a véletlen elérésű és a szekvenciális fájloktól, a bináris fájlok tartalma nem csupán ASCII szöveg lehet. A bináris hozzáféréssel bár milyen fájlt olvashatunk, írhatunk, illetve módosíthatunk, viszont nekünk kell kezelni, hogy a fájlban mit és hogyan tárolunk.

4.1.3 Szekvenciális (soros hozzáférésű) fájlok használata

Amint azt a korábbiakban kifejtettük, a szekvenciális hozzáférésű fájlok, rekordok sorozatát tárolják, ahol minden rekord egy szövegsor, melyet egy CR+LF (kocsi vissza + soremelés) karakterpár zár le. Itt emlékeztetünk arra, hogy a szekvenciális fájlok egyszerre csak egy műveletre nyithatók meg: pl. olvasásra (INPUT mód), vagy írására. Ez utóbbi esetben megkell különböztetni az új fájl írása (OUTPUT mód), illetve létező fájl végére történő írás (APPEND mód), Ahhoz, hogy szekvenciális fájloknál az egyik műveletről a másikra váltsunk – például írásról olvasásra – először be kell zárni a fájlt, majd megnyitni a másik módban.

4.1.3.1 Szekvenciális hozzáférésű fájlok írása és olvasása

Ha a tárolandó információ azonos szerkezetű diszkrét (a többi adattói jól elkülöníthető) egységekből áll, akkor használhatunk mezőket a szekvenciális fájlban. Példa lehet erre egy könyvek adatait nyilvántartó fájl. Minden könyvhöz tárolni akarjuk az íróját, a címét, és egy indexet, amely a könyv helyét határozza meg a polcon. Ehhez készíthetünk egy szekvenciális fájlt, melynek szerkezete a következő:

A fájlban minden rekord egy-egy könyv adatait tartalmazza.

Minden rekord három mezőből áll, melyek közül az első a szerző, a második a cím, a harmadik az index.

Legyen az első bejegyzés a „A nagy mesekönyv” című könyv, melyet Benedek Elek írt, az indexszáma pedig 12. Ehhez a bejegyzéshez tartozó rekord a fájlban a következőképpen fog kinézni:

„Benedek Elek”, „A nagy mesekönyv”, 12

Ha a második bejegyzés Dosztojevszkijtől a Bűn és bűnhődés, melynek indexszáma 123, akkor az a következőképpen tárolódik:

„Dosztojevszkij”,”Bűn és Bűnhődés”, 123

Minden rekord külön sorban lesz. Ezekből a példákból világosan látszik, hogy szekvenciális fájlokban vesszők határolják (vesszőt használ szeparátorként) az egyes mezőket, a szöveges adatok pedig macskakörmök között vannak.

Ha egy rekordot akarunk kiírni egy szekvenciális fájlba (szeparátorokkal együtt), akkor a Write # utasítást kell használnunk:

Write # filenum, [lista]

filenum: az a szám, melyet a fájl OUTPUT vagy APPEND módban való megnyitásakor a fájlhoz rendeltünk.

lista: egy vagy több Basic kifejezés, melyet a fájlba akarunk írni. Ha a lista több mint egy elemet tartalmaz, akkor azokat vesszővel kell elválasztanunk. Ha nem adunk meg listát a Write # utasításnak, akkor üres rekord (tehát üres sor) kerül a fájlba.

Íme, egy egyszerű példa:

```
Write # filenum, „Benedek Elek”, „A Nagy Mesekönyv „, 12
```

Egy programban általában változókkal paraméterezzük a Write # utasítást:

```
book.name=„Benedek Elek”
book.title=„A Nagy Mesekönyv”
book.index=12
Write # filenum, book.name, book.title, book.index
```

Minden Write # utasítás egy rekordot ír a fájlba úgy, hogy automatikusan beszúrja a szükséges vesszőket és macskakörmöket. Az ilyen fájlba kerülő adatokban nem szerepelhet a macskaköröm.

Szekvenciális fájlból a mezőkre osztott adatot az Input # utasítással lehet beolvasni.

Az Input # beolvas egy vagy akár több mezőt a fájlból, majd azokat programváltozókba írja:

```
Input # filenum, lista
```

Itt az egyes paraméterek jelentése a következő:

filenum: a fájl- INPUT módban történő - megnyitásakor a fájlhoz rendelt szám.

lista: egy vagy akár több programváltozóból álló lista. Ezekbe kerülnek a fájlból kiolvasott adatok.

Ha a lista egynél több változónevet tartalmaz, akkor azokat vesszővel kell elválasztani.

Az Input # utasítás annyi mezőt olvas be a fájlból ahány változó meg van adva a paraméterlistában, az egyes mezőket pedig sorrendben a megfelelő változókhoz rendeli. Ne felejtsük el, hogy a szekvenciális fájlokat csak szekvenciálisan lehet olvasni, azaz csak az első rekord első mezőjéről lehet indulni, és onnan lehet haladni előre.

Íme egy egyszerű példa:

```
filenum = FreeFile
Open „Konyv,txt” For Output As # filenum
Write # filenum, „Benedek Elek”, „A Nagy Mesekönyv”, 12
Write # filenum, „Dostoyevski”, „Bűn és Bűnhődés”, 123
Close # filenum
```

Majd a program egy másik részében így olvashatjuk vissza:

```
Dim A As String, B As String, C As String, O As String
Dim x As Integer, y As Integer
filenum = FreeFile
Open „Konyv.txt” For INPUT As # filenum
Input # filenum, A, B, x
Input # filenum, C, O, Y
Close # filenum
```

A fenti kód lefutása után az egyes változók értékei a következők lesznek:

A=„Benedek Elek”,B=„A Nagy Mesekönyv”,x=12,
C=„Dosztojevszkij”,D=„Bűn és Bűnhődés",y=123.

Ha a fenti kódban kicseréljük a két Input # utasítást a következő, egy Input # utasításra:

```
Input # filenum, A, B, x, C, O, Y
```

vagy akár az alábbi három utasításra:

```
Input # filenum, A, B
Input # filenum, x, C
Input #filenum, O, y
```

akkor is ugyanezt az eredményt érjük el. Az, hogy az Input # utasítás miként bontja a fájlt mezőkre, az a paraméterlistában szereplő változók típusától függ. Ha az utasítás éppen String típusú változóba olvas, akkor a mező végét a következő jelek valamelyike mutatja:

- Idézőjel, ha mező Idézőjellel kezdődött.
- Vessző, ha a mező nem idézőjellel kezdődött.
- Kocsi vissza + soremelés (CR+LF) karakterpár

Ha numerikus változóba olvas, akkor a mező végét az alábbi karakter sorozatok valamelyike jelzi:

- Egy darab vessző.
- Egy vagy több szóköz.
- Kocsi vissza + soremelés (CR+LF) karakterpár

Ha szekvenciális fájlban tárolunk adatokat, akkor a programban ügyelni kell arra, hogy a Write # és az Input # utasítások az egyes rekordok mezőinek típusát, sorrendjét, és számát tekintve összhangban legyenek. Más szóval az egyes adatelemeket ugyanolyan sorrendben kell kiolvasni a fájlból, ahogy beleírtuk azokat.

Ha ez az összhang felborul, akkor kétféle probléma is felmerülhet:

- Előfordulhat, hogy a program az Input # utasítással a fájlból egy mezőt egy attól különböző típusú változóba akar beolvasni (például Stringet olvas egy numerikus változóba). Ez nem feltétlenül okoz hibát, de valószínűleg váratlan eredményeket produkál. Ha numerikus mezőt olvasunk be String változóba, akkor a változó az adott szám szöveges reprezentációját kapja értékül. Ha String mezőt olvasunk be numerikus változóba, akkor az eredmény a String tartalmától függ. Ha a String nem numerikus karakterrel kezdődik, akkor a változó értéke nulla lesz, egyébként a változó a String elején található szám értékét veszi fel.
- Ha a mezők száma nem azonos, akkor előfordulhat, hogy a program „eltéved” a fájlban, azaz nem tudja, hogy éppen melyik rekordnál tart. Emlékezzünk rá, hogy az Input # mezőket, és nem rekordokat számlál az olvasás során. Tehát a végrehajtott Input # utasítás a következő mezőnél fogja elkezdeni az olvasást, ami viszont nem biztos, hogy a következő rekord elejével azonos.

A szekvenciális elérésű fájlok egyik felhasználási területe a változó hosszúságú Springekből álló tömbök tárolása. A most következő kódrészlet azt mutatja be, hogy hogyan kell az efféle tömböket fájlba írni:

```
Dim Nevek(100) As String, Szam As Integer, filenum As Integer
```

(a Nevek(100) tömböt először adatokkal kell feltölteni, különböző hosszúságú nevekkel).

```
filenum = FreeFile
Open „NOTES.TXT” For OUTPUT As #filenum
    For Szam = 0 to 100
        WRITE #filenum, Nevek(Szam)
    Next Szam
Close # filenum
```

Az alábbi pedig visszaolvassa az adatokat a fájlból, és belepakolja egy Springekből álló tömb be:

```
Open „NOTES. TXT” For INPUT As #filenum
    For Szam = 0 To 100
        Input #filenum, Nevek (Szam)
    Next Szam
Close # filenum
```

Szekvenciális fájlokban számokból álló tömböt is tárolhatunk, de amint a későbbiekben látni fogjuk, erre a célra egy másik elérési mód előnyösebb.

4.1.3.2 A fájl végének vizsgálata

Ha a program szekvenciális fájlból olvas adatot, akkor az EOF függvény segítségével le lehet kérdezni, hogy a program elérte-e a fájl végét:

```
EOF (filenum)
```

Az EOF igazat (True értéket) ad vissza, ha a fájlból már kiolvastuk az utolsó rekordot, hamisat (False), ha még nem. Ha egy szekvenciális fájlból az utolsó rekord kiolvasása után még olvasni próbálunk, akkor hiba keletkezik. A most következő kód mutatja meg, hogy miként lehet egy ciklussal, az EOF függvényt használva beolvasni egy teljes szekvenciális fájlt soronként, majd egy tömbbe rakni:

```
Dim info(1000) As String
Dim Szam As Integer
Open „Sajat.txt” For Input As #filenum
    Szam = 0
    Do While Not EOF(filenum)
        Input #filenum, info (Szam)
        Szam=Szam+1
    Loop
Close # filenum
```

4.1.4 Véletlen elérésű fájlok használata

A véletlen elérésű fájl annyiban hasonlít szekvenciális társához, hogy mindkettő rekordokból áll. A véletlen elérésű fájl rekordjainak viszont egyforma hosszúaknak kell lenniük, hiszen a program az egyes rekordokat fájlbeli pozíciójuk alapján azonosítja. A rekordokat nem választja el speciális karakter. Például ha a fájl minden rekord-ja 100 bájt hosszú, akkor az első rekord a fájl első bájtjától a századikig található (a fájlokban a bájtok sorszámozása 1-gyel kezdődik. A negyedik rekord pedig a 301. pozíción kezdődik, és a 400-ig tart. A véletlen elérésű fájl rekordjai eggyel kezdve, egyesével növekvően vannak számozva. Igazából nem is számozva vannak, hanem a fájlban elfoglalt helyük határozza meg, hogy ők hányadikak. Legfeljebb 2 147 483 647 rekord lehet egy fájlban, ennél többet ilyen módon nem tudunk kezelni. Minden rekordban több mező lehet. Egy véletlen elérésű fájlban minden rekordnak azonos a szerkezete. Más szóval ugyanannyi a mezők száma, és ugyanaz az egyes mezők mérete. Mielőtt létrehozunk, vagy használunk egy véletlen elérésű fájlt, először meg kell adni annak rekordszerkezetét.

4.1.4.1 A véletlen elérésű fájlok rekordszerkezetének megadása

A véletlen elérésű fájlok rekordszerkezetének megadása ugyanúgy történik, mint a felhasználói adattípusok definiálása, azaz a Type ..• End Type kifejezéssel. Ezt az 4.7 Felhasználói Adattípusok fejezetben tárgyaltuk részletesen. 21. old.

Első lépésben készítünk egy olyan adattípust, melynek szerkezete megegyezik a létrehozandó véletlen elérésű fájl rekord-szerkezetével. A könyvnyilvántartó program esetén ez például lehet a következő:

```
Type Konyv_reg
    Szerzo As String * 20
    Cim As String * 25
    Index As Integer
End Type
Dim Könyv_tar(100) As Konyv_reg
```

Most van egy felhasználói típusunk, ami képes lesz kezelni a könyvekre vonatkozó adatokat a programban, továbbá meghatározza az adatfájl rekordszerkezetét, Véletlen elérésű fájlok megnyitásakor az egyik paraméter a fájl rekordjainak hossza (amint azt a fejezet egy korábbi részében láttuk az Open utasítás tárgyalásakor). Ezt a számot könnyen megkaphatjuk, ha meghívjuk a Len függvényt, mely visszaadja a paraméterként kapott típus hosszát bájtokban. Tehát az Open utasítás a következőképpen néz ki:

```
filenum = FreeFile
Open „Konyvtar.txt” For RANDOM As #filenum Len = Len (Konyv_tar(1) )
```

Vegyük észre, hogy a Len függvénynek a tömb egy elemét adtuk át, és nem az egész tömböt. Ez az utasítás a felhasználói adatszerkezet méretével, azonos rekordhosszal nyitja meg a véletlen elérésű fájlt.

4.1.4.2 Véletlen elérésű fájlok írása és olvasása

Véletlen elérésű fájlokba írni a Put utasítással lehet:

```
Put #filenum, [rekordsorszám], felhasználóiTípus
```

Ahol az egyes paraméterek jelentése a következő:

filenum: a fájl megnyitásakor a fájlhoz rendelt szám

rekordsorszám: annak a rekordnak a sorszáma, ahová az adatot akarjuk beírni. Ha nem adunk meg rekordsorszámot, akkor a soron következő helyre kerül az adat (az a legutolsó Get vagy Put művelet helyét követő pozícióra).

Ha a fájl megnyitása óta még nem volt Get vagy Put művelet, akkor a művelet az első rekordra vonatkozik.

felhasználóiTípus: az a változó, mely a kiírandó adatokat tartalmazza, és szerkezete megegyezik a fájl rekordtípusával.

Folytatva a nyilvántartó példát, ha a fájl első rekordjába akarunk írni, akkor a következőt tehetjük:

```
Konyv_tar(1).Szerzo = „Benedek Elek”
Konyv_tar(1).Cim = „A Nagy mesekönyv”
Konyv_tar(1).Index = 12
Put # filenum, 1, Konyv_tar(1).
```

Amikor először létrehozunk egy véletlen elérésű fájlt, akkor kezdhetjük az írást az első rekordnál, mindig növelve a sorszámot eggyel minden egyes sikeresen végrehatott Put utasítás után. A Put utasításból akár el is hagyhatjuk a rekordsorszám paramétert, hiszen a Visual Basic automatikusan nyilvántartja az aktuális pozíciót. Viszont ha létező fájlhoz írunk hozzá újabb rekordokat, akkor magunk kell, hogy kezeljük a rekordsorszám paramétert, ezáltal biztosítva, hogy az adat tényleg oda kerüljön, ahová akarjuk.

Ha egy olyan sorszámú rekordba írunk, mely már tartalmaz adatot, akkor az **Új adat** felülírja a régit. Ha nem akarunk felülírni már meglévő adatokat, akkor az új rekordokat a fájl végére kell írnunk. Más szóval, ha egy **n** rekordból álló fájlhoz írunk hozzá adatsorokat, akkor az új rekordok sorszámai **n+l**-nél kell, hogy kezdődjenek.

A Lof függvény segítségével határozhatjuk meg a fájlban lévő rekordok számát. Ez a Lof függvény a fájl méretét adja meg bájtban:

```
Lof(filenum)
```

Itt a filenum (fájlsorszám) paraméter annak a megnyitott fájlnak a száma, melynek a hosszát meg akarjuk tudni. Miért jó, hogy tudjuk a fájl hosszát bájtban? Tudjuk, hogy a Len függvénnyel a fájl egy rekordjainak hosszát kérdezhetjük le, ebből következik, hogy a fájl hossza osztva egy rekord hosszával megadja a fájlban lévő rekordok számát. Tehát a megoldás, a következő rekord írására:

```
Rec_szam = (Lof(filenum) \ Len(Konyv_tar(1))) + 1
```

```
Put #filenum, Rec_szam, Konyv_tar(Rec_szam).
```

A következő olvasandó vagy írandó rekord sorszámát a Seek függvény adja meg.

Hasonlóan a Loc függvény, mely az utoljára olvasott, illetve írt rekord sorszámát adja vissza.

A Seek a fájlmutató aktuális pozícióját adja meg (azaz a következő olvasandó rekord számát), a Loc pedig a legutoljára kiolvasott illetve beírt rekord helyét.

Szintaxisuk a következő:

```
Seek (filenum)
Loc (filenum)
```

A filenum (fájlsorszám) már megszokott jelentéssel bír.

A Loc által visszaadott érték eggyel kisebb, mint a Seek esetén.

Ha éppen most nyitottuk meg a fájlt, a Loc értéke 0, míg a Seek az 1-et adja vissza.

Véletlen elérésű fájlból olvasni a Get utasítással lehet. Az adat a fájl egy adott rekordjából kerül kiolvasásra, és az előre definiált felhasználói adattípusú változóba kerül. A Get utasítás szintaxisa:

```
Get [#]filenum, [rekordsorszám], felhasználóiTípus
```

Itt az egyes paraméterek jelentése a következő:

filenum: a fájl megnyitásakor a fájlhoz rendelt szám

rekordsorszám: annak a rekordnak a sorszáma, ahová az adatot akarjuk beírni. Ha nem adunk meg rekordsorszámot, akkor a soron következő helyre kerül az adat (az a legutolsó Get vagy Put művelet helyét követő pozícióra).

Ha a fájl megnyitása óta még nem volt Get vagy Put művelet, akkor a művelet az első rekordra vonatkozik.

felhasználóiTípus: az a változó, mely a kiírandó adatokat tartalmazza, és szerkezete megegyezik a fájl rekordtípusával.

Folytatva példánkat, az adatfájl első rekordját a következőképpen olvashatjuk ki:

```
Get [#]filenum, 1, Konyv_tar(1)
```

Ha minden rekordot ki akarunk olvasni egy véletlen elérésű fájlból, akkor írjunk egy ciklust, mely az első rekordról indul, és folyamatosan olvas egészen a fájl végéig, és közben a kiolvasott elemeket belerakja egy megfelelő típusú tömbbe.

4.1.4.3 A fájl végének figyelése

A fájl végének a figyelés itt is az EOF függvénnyel lehetséges, úgy, mint a **Szekvenciális fájlok** végének figyelése, a korábbi szakaszban már tárgyalva lett Hasonlóan a fentiekhez, ebben az esetben is megkaphatjuk az utolsó rekord sorszámát, ha elosztjuk a fájl hosszát egy rekord hosszával.

A Visual Basic minden megnyitott véletlen elérésű fájlhoz nyilvántart egy mutatót. Ez a mutató határozza meg, hogy hányadik rekordot olvassa a Get, vagy hányadikat írja a Put, ha nem adjuk meg a rekordsorszám paramétert. Egy véletlen elérésű fájl megnyitásakor - legyen akár új, akár létező fájl - a rekordmutató az első rekordra mutat. Véletlen elérésű fájlok írásakor és olvasásakor a fájlmutató a következőképpen módosul:

A Get és Put eljárások rekordsorszám paraméter nélküli hívása esetén a fájlmutató értéke eggyel nő.

A Get és Put eljárások rekordsorszám paraméterrel való meghívása esetén a fájlmutató rekordsorszám+1 lesz.

4.1.5 Bináris fájlok használata

A bináris fájlokat formázatlan bájtsorozatnak tekinthetjük. A fájlban nincs rekordszerkezet, ha mégis kell legyen, azt kódból szimuláljuk. A bináris fájlok nagyon rugalmas adattárolási lehetőséget biztosítanak, de strukturálatlanságuk miatt a programozónak kell „fejben tartania”, hogy mi van a fájlban. Azt is nyilván kell tartanunk, hogy éppen hol tartunk a fájlban.

4.1.5.1 Fájlpozíció

Minden megnyitott fájlnak van egy fájlmutatója. A fájlmutató egy numerikus érték, amely a fájlnak arra a pozíciójára mutat, ahol a következő írási vagy olvasási művelet fog történni. A bináris fájlokban minden pozícióhoz egy bájt tartozik. Azaz, ha egy fájl **n** bájtot tartalmaz, akkor a pozíciók **1**-től **n**-ig számozódnak. A fájlmutatót megváltoztatni, illetve lekérdezni a Seek utasítással, illetve a Seek és Loc függvényekkel lehet. Ezek az utasítások ugyanúgy használhatók itt is, mint a véletlen elérésű fájloknál. (A véletlen elérésű fájlok esetében viszont a mutató rekordokra mutat, és nem bájtokra.)

A Visual Basic tudja, hogy egy adott fájl milyen módban lett megnyitva, és ennek megfelelően értelmezi a Seek és Loc függvényeket. Bináris fájlok esetén bájtokra, véletlen elérésű fájlok esetén rekordokra. A Seek és Loc szintaxisa a következő:

Seek (#) filenum, újPozíció

A Seek utasítás a újPozíció pozícióra állítja a rekordmutatót

Itt a filenum (fájlsorszám) a megnyitáskor a fájlhoz rendelt számot jelöli.

```
[poz =] Seek(filenum)
```

A Seek függvény pedig visszaadja a fájlmutató aktuális pozícióját (azaz a következő írás vagy olvasás helyét).

```
[poz =] Loc(filenum)
```

A Loc függvény az utoljára beolvasott vagy kiírt bájt pozícióját adja meg.

Ha nem írjuk saját kezűleg a mutatót, akkor a Loc mindig eggyel kisebb, mint a Seek.

4.1.5.2 Bináris fájlok olvasása, írása

Bináris fájlokból olvasni a Get utasítással lehet, írni beléjük pedig a Put utasítással. Ezek megegyeznek a véletlen elérésű fájloknál használt utasításokkal, ám másként értelmeződnek attól függően, hogy az írandó, illetve az olvasandó fájl milyen módban lett megnyitva:

```
Get [#] filenum, [poz], változó
Put [#] filenum, [poz], változó
```

Ahol az egyes paraméterek jelentése a következő:

filenum: a fájlhoz megnyitáskor hozzárendelt szám.

poz: az olvasás, illetve írás helyét határozza meg. Ha nem adjuk meg a poz paramétert, akkor a fájlmutató által meghatározott helyen hajtódik végre a művelet.

változó: egy akármilyen típusú Basic változó. A Get utasítás ebbe a változóba olvas a fájlból, a Put pedig ennek a változónak a tartalmát írja ki a fájlba. Az egyes utasítások automatikusan annyi bájtot olvasnak, illetve írnak amennyi bájtból a változó áll. Ha a változó egy változó hosszúságú String, akkor az átvitt bájtok száma megegyezik a **változóban** lévő karakterek számával.

Bináris fájlok megnyitásakor a fájlmutató az első pozícióra mutat. A Get és Put utasítások az átvitt bájtok számával növeli meg a fájlmutató értékét.

A bináris fájlok hosszát a Lof függvénnyel lehet meghatározni.

```
[hossz =] Lof(filenum)
```

A Lof által visszaadott érték megegyezik a fájl utolsó bájtjának pozíciójával. A Lof függvénnyel, és egy ciklussal gyorsan be lehet olvasni egy egész bináris fájlt. A következő részlet másolatot készít egy fájlról. A továbbiakban feltesszük, hogy az **Eredeti.txt** nevű fájl létezik, a **Masolat.txt** pedig nem.

Egy **1 karakterből** álló fix hosszúságú String segítségével a program mindig egy bájtot olvas, majd kiírja azt. Ez a kód természetesen működik, de azért megjegyzem, hogy a bájtonkénti olvasás „kissé” nehézkes, és kerülendő módszer.

```
Dim ch As String * 1, Szam As Integer, f1 As Integer, f2 As Integer
f1 = freefile
Open „Eredeti.txt” For BINARY As #f1
   f2 = freefile
Open „Masolat.txt” For BINARY As #f2
   For Szam = 1 TO Lof(f1)
      Get #f1, , ch
      Put #f2, , ch
   Next Szam
Close #f1, #f2
```

Az EOF függvény segítségével lehet figyelni egy bináris fájl végét. Például a fenti kódban a ciklust a következőképpen is megírhattuk volna:

```
Do While Not Eof (f1)
   Get #f1, , ch
   Put #f2, , ch
Loop
```

A bináris fájlok rugalmasságuk miatt rendkívül jól használhatók a program adatainak tárolására. Ez különösen akkor igaz, amikor az adat formátuma nem illeszkedik sem a véletlen elérésű fájlok, sem a szekvenciális fájlok szerkezetéhez. Ne felejtsük el, hogy a bináris fájlokhoz nincs struktúra rendelve, ezért a programnak kell nyilvántartania az adattárolás formátumát. Például, ha a program egy ezer elemű tömböt (dupla pontosságú változókat) használ, akkor a szóban forgó tömböt a következőképpen **Bináris** fájlban tárolni:

```
Dim Data(999) As Double, Szam As Integer
```

(A Data tömböt előre fel kell tölteni, 1000 Double számmal)

```
Open „Ezres_tomb” For BINARY As # 1
   For Szam = 0 To 999
      Put # 1, , Data (Szam)
   Next Szam
Close #1
```

Adatainkhoz később a következőképpen férhetünk hozzá:

```
Open „Ezres_tomb” For BINARY As #1
   For Szam = 0 To 999
      Get #1, , Data2 (Szam)
   Next Szam
Close #1
```

Következzék néhány jó tanács **bináris fájlok** használatához:

Ha létező fájl végére akarunk új adatot írni, akkor állítsuk a mutatót a fájl végére a követező utasítással:

```
Seek (filenum), vagy  Lof (filenum) +1
```

Ha egy létező fájlban a fájl vége előtti pozícióra írunk adatot, akkor az Új adat felülírja a régit.

Ha a fájl vége utáni pozíciókra írunk adatot (például a Lof (filenum) +10 pozícióra) akkor a fájl meghosszabbodik a szükséges méretre **de a köztes területre szemét” kerül**, más szóval definiálatlan értékek.

Ha a fájl vége utáni bájtpozícióról próbálunk meg olvasni, akkor nem történik hiba, de a visszaadott adat-szemét”, definiálatlan érték lesz. Ha nem figyelünk rá, elég érdekes adataink lesznek.

4.1.6 Melyik fájltípust használjuk?

Ha egy kis zavart érzünk a Visual Basic által nyújtott fájlelérési módokkal kapcsolatban, akkor megnyugodhatunk, nem vagyunk egyedül. Nem mindig teljesen világos, hogy melyik fájltípus a legmegfelelőbb egy adott alkalmazásban. A most következő tanácsokat segítségnek szánjuk a döntés meghozatalához:

- Ha az adat szövegsorokból áll, akkor - függetlenül attól, hogy azt egy Visual Basic alkalmazás hozta-e létre, vagy sem - használjunk szekvenciális elérést. A szekvenciális szövegfájlokat a hagyományos, és az FSO megközelítésben egyaránt elérhetjük.
- Ha olyan nem szöveges fájlokat akarunk használni, melyet más alkalmazások hoztak létre, és nem szigorú szerkezetűek, akkor használjuk a bináris elérést.
- Ha az adatnak pontosan megadható a rekordszerkezete, akkor használjunk véletlen elérést vagy szekvenciális fájlt. A programtói függően előfordulhat, hogy a szekvenciális a jobb.
- Numerikus adatok tárolásához használhatjuk a szekvenciális és a bináris módot is. A bináris mód annyiban jobb, hogy gyorsabb, és sokkal kisebb méretű fájlokat készít, de többet kell vele dolgoznunk.
- Változó hosszúságú stringek tárolására használjuk a szekvenciális módot. Fix hosszúságú stringek esetében használjunk vagy szekvenciális, vagy bináris fájlt. A bináris mód előnyösebb a sebesség, illetve a méret szempontjából, de ebben az esetben nem olyan jelentős a különbség, mint a numerikus tömbök esetében.
- A véletlen elérés esetében gyorsabban érhetőek el a fájl egyes rekordjai.
- Szöveges adat esetében viszont pazarló, mert az egyes stringek szóközökkel egészülnek ki, hogy kitöltsék a rekordméretet.
- A szekvenciális mód esetében a fájl egyes rekordjainak elérése lassabb, viszont gazdaságosabb a helykihasználás, mert a stringek nem egészülnek ki szóközökkel.

Ezek persze csak tanácsok. Ne tekintsük őket merev szabályoknak. Fontos viszont szem előtt tartani, hogy a fájlelérés módja nem változtatja meg a fájl fizikai szerkezetét. Csupán azt határozza meg, hogy a program miként olvassa, illetve írja a fájlt. Más szóval egy akármilyen fájl - függetlenül attól, hogy milyen módban nyitjuk meg mindenképp egy lemezen tárolt bájtsorozat. Például megtehetjük, hogy készítünk egy fájlt véletlen eléréssel, és mégis sokszor bináris módban nyitjuk meg. Azzal, hogy ugyanazt a fájlt több elérési móddal használjuk, gyakran drasztikusan lehet növelni a program hatékonyságát, vagy éppen csökkenteni az elkészítéséhez szükséges erőfeszítéseket. Amíg még csak ismerkedünk a fájlkezeléssel, addig mindenesetre egy fájlt csak egy móddal használjunk.

4.2 A „UserForm” használata

4.2.1 Gondolatok a téma megkezdéséhez

A Jegyzet azt a VBA (Visual Basic Application) programnyelvet mutatja be a hallgatók számára, mely a „könyvtárnyi irodalommal” rendelkező VB (Visual Basic) programnyelvnek, „kicsinyített”, a Microsoft Office programokhoz igazított és beépített, Felhasználói háttérnyelve. Ez a VBA már jóval, szerényebb irodalommal rendelkezik… azok is régi kiadásúak.

A VBA minden Office alkalmazásban rendelkezésre áll. Alkalmazásukkal és használatukkal a prezentációk feladatorientáltan kiegészíthetők, az egyes alkalmazásokban rendelkezésre álló eljárások mellé saját Felhasználói eljárásokat is lehet programozni, és mindezt a Windows alap-programnyelv ismerete nélkül. Ezekkel az egyedi Felhasználói programokkal bővített prezentációs feladatok szakmai értéke, színvonala jelentősen emelkedik, a csak Excel függvényeket használó alkalmazáshoz képest.

Jegyzetünkben, az Excelhez igazított VB Application változatával foglalkozunk, amelyben már megtalálhatók a VB-ben még nem létező azon „új parancsok”, amelyek képesek az Excel számolótábla celláival is kommunikálni, valamint képesek az Excel celláit, az Objektum orientált VBA programozásban – tulajdonsággal rendelkező – összetett Objektumként kezelni. A Jegyzet 4. fejezetében tanult programozásban, e Cella objektum használat volt a döntő, hiszen a munka kiinduló adatai a Cellákból származtak és az Eredmény is Cellában végződött.

Ezt a programozást, kicsit pontatlanul, az Excel **Makrózásnak** is nevezik, és ez a hallgatók számára nem is volt „elég vonzó”, mert mint „háttérprogram” valóban a „háttérben fut”, az Excel Munkalapok mögött „valahol láthatatlanul”. A legtöbben nem is szívesen tanulták, talán azért is, mert egyrészt bonyolult és folyamatos logikai gondolkodást igényelt, másrészt a megirt program eredményközlése sem volt látványos. Ráadásul az összemérhetetlenül rövidebb volt a „programfutás” ideje, minta „befektetett programírás” ideje.

De mindazok számára „jó hír”, akik már „átverekedték” magukat tanulmányaikban az Jegyzet 5. teljes fejezetén, hogy most ebben a 6.4. UserForm használat fejezetben, „elérkeztek” ahhoz a „tanulnivalóhoz”, amit nemcsak „megszeretni” lehet, de amibe egyenesen „beleszerelmesedni” lehet! A **Formok – Controlok** építése, használata már „látványos”, mert az Excel munkafüzet-lapok előtt láthatók a képernyőn. A program kezelése ezeken keresztül valósul meg, a programhasználó interaktív közreműködésével, amit nem csak a Felhasználó tud élvezni, de ennek a programnak létrehozása a „Programozónak” is látványos alkotói élményt jelent.

Form (Űrlap), tulajdonképpen egy ablak, és mint ilyen, rendelkezik a Windows (ablak), a Windows-ban megszokott minden jellemvonással. Mint egy Windows ablak jelenik meg a Form is,

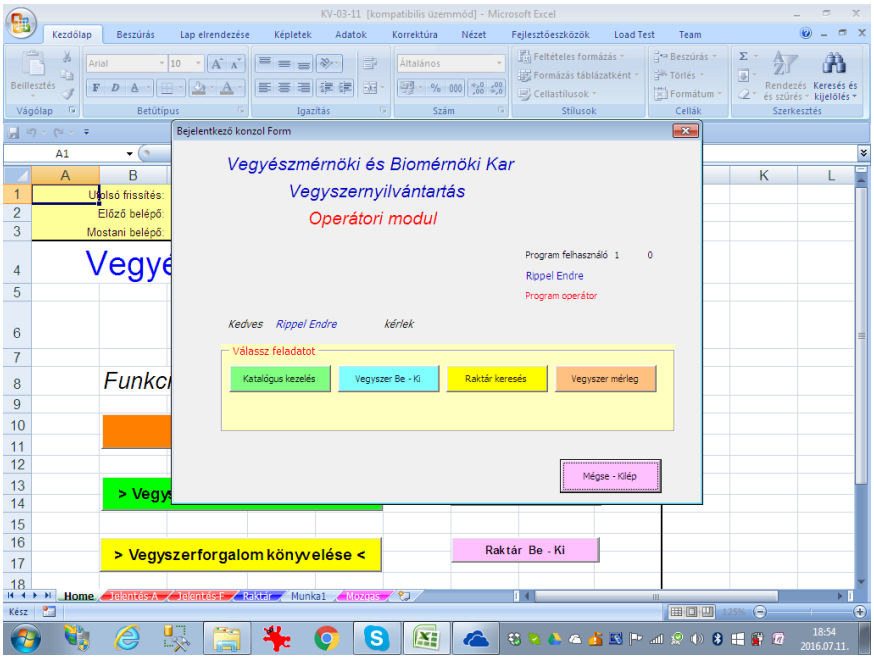
gömbölyített sarkos, négyszögletes keretezett felület, kék fejléccel és piros „ ”-es bezáró gombbal. A VBA kezdetben egy üres felületű **Form**ot ad, amit a programozó tölt fel **Control**okkal, az igényeknek megfelelően.

Control (Vezérlőeszköz Objektum) a Formokra kerülnek alapvetően, de az Excel munkalapokon is ellehet helyezni. Igy ezen objektumokon keresztül tud a Felhasználó Parancsutasításokat kezdeményezni, - folyamatokat indítani!

Példa bemutató: Még 2007-ben, a Kar vezetésében fogalmazódott meg az az igény, éppen a Számítástechnika tárgyra támaszkodva, hogy készüljön egy Vegyszernyilvántartó Demó Tanulmány program. Tanulmányaival tárja fel, és Demóival valóságban is mutatassa be a Kar Vezetőinek, a Vegyszerek Nyilvántartása – Vegyszerforgalom Kezelése – Felügyelete témakörébe tartozó, valamennyi Induló Feltételt, Feldolgozási és Felügyeleti tevékenységet, valamint a Kimeneti Naplózások lehetőségeit, a minden tanszéken rendelkezésre álló Microsoft Excel táblázatkezelőre alapozva kellett megoldani.

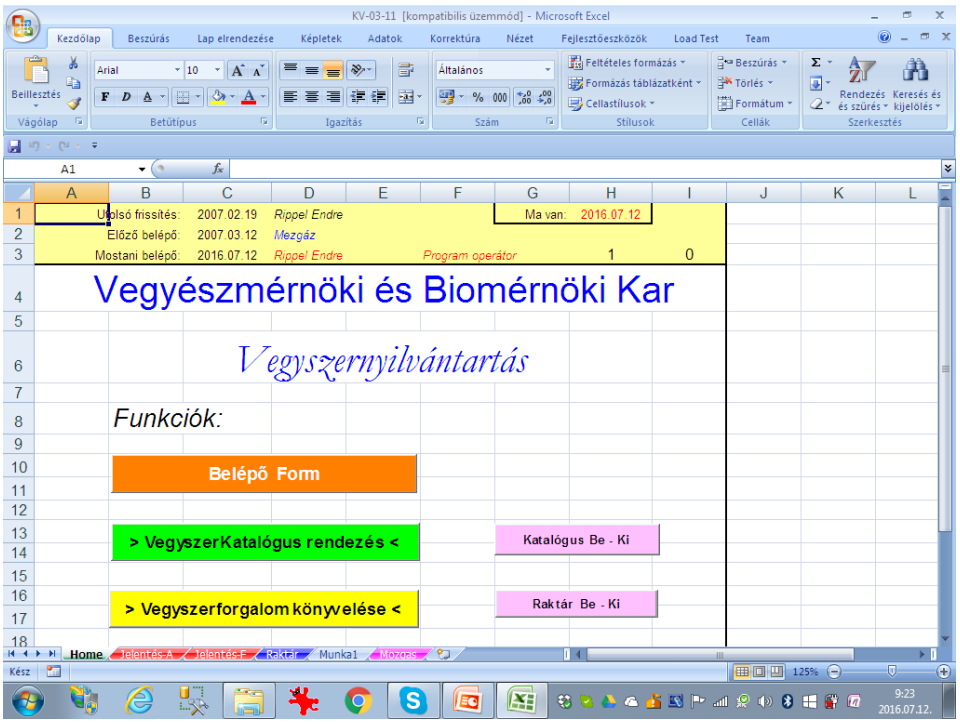
A Kar akkori szervezettségéből adódóan, a vegyszerekről, a Tanszékenként és Vegyszertároló helyek adataiból összerakott Excel táblázat **Raktár** munkalapja **Tanszék – Raktár – Vegyszer** tördelésű összeállításban, (11 Tanszék / 65 Raktár / Vegyszer) közel 5000 soros adatbázist jelentett. E hatalmas adatmennyiség kezeléséhez, VBA-val készült **Form – Control** felületek készültek.

A következőben, egy Vegyszerjelentés összeállításának lépéseit mutatjuk be.



1. kép

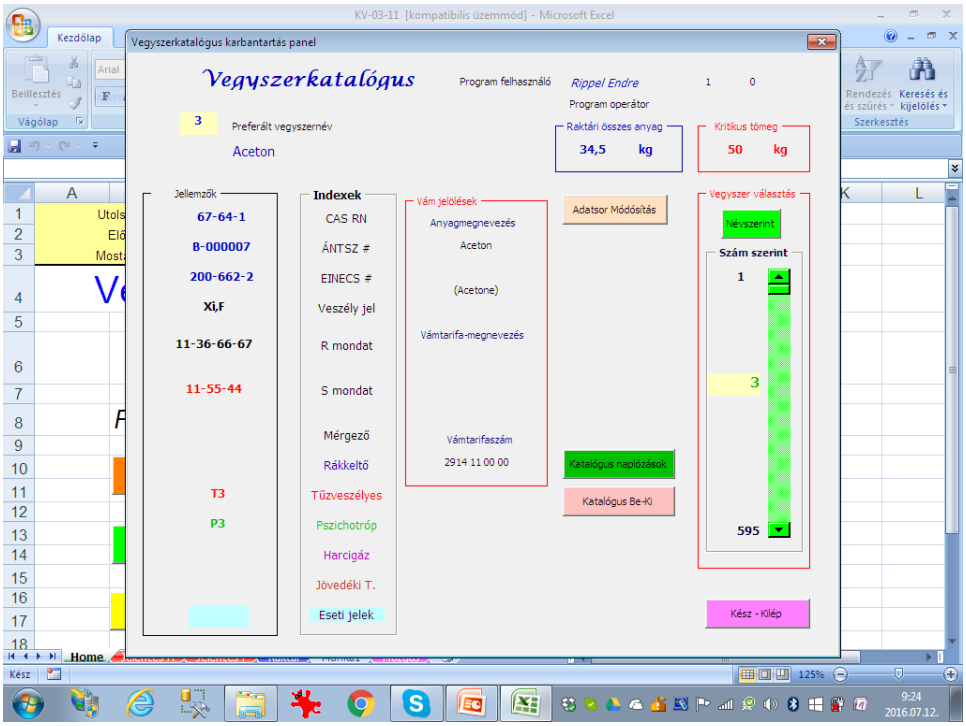
Jelszó (*Password*) bejelentkezés az Operátori modulon keresztül. Ez a Form, a program használat és használati jogosultságok beállítását, és a használat naplózása feladatokat oldja meg. A Form gombjai választási lehetőséget adnak a feladatok közül. A Mégse-Kilép gomb után a 2. kép jön létre.



2. kép

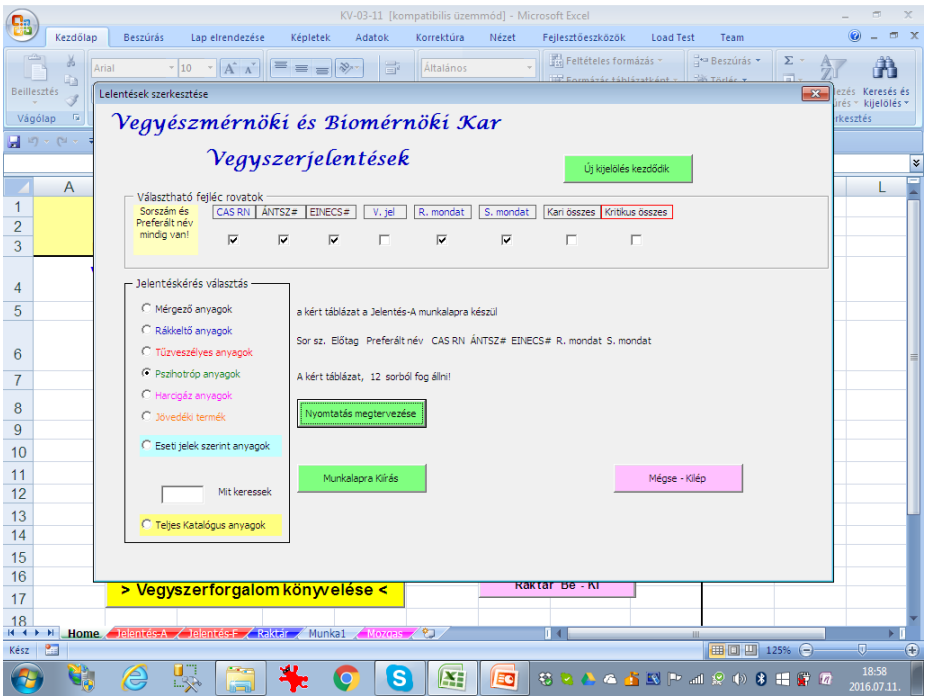
Belépés után feladatválasztó Nyomógomb **Control**ok láthatók a **Home** Munkalapon.

A > Vegyszer Katalógus rendezés < gomb megnyomása után, jelenik meg a 3. kép.



3. kép

A > Vegyszer Katalógus rendezése < nyomógomb adja a Vegyszerkatalógus Formot. Vegyszerek, tulajdonságok kereshetők, módosíthatók, naplózhatók. A Kész – Kilép nyomógomb választása után 4. kép jelenik meg.



4. kép

Vegyszerjelentések Forma jelenik meg, Itt választani kell, a vegyszercsoportok közül és be kell jelölni, milyen fejléc kiosztást kér. Választás után, a Nyomtatás megtervezés nyomógomb átolvassa a mintegy 5000 sort, és a gomb felett válaszol.

5. kép

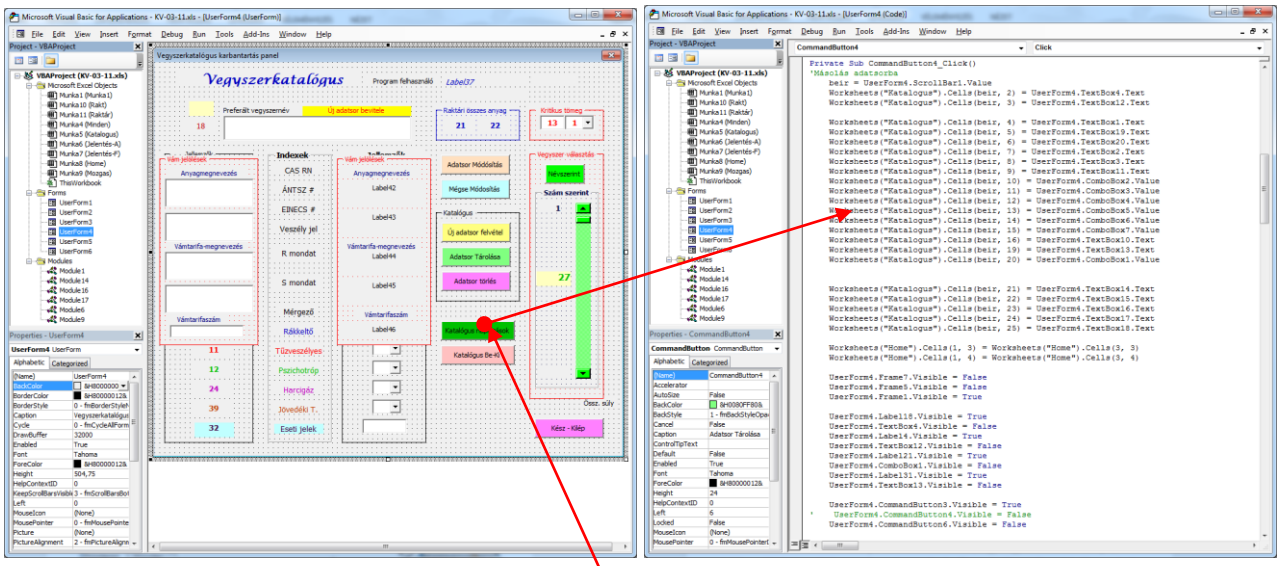
A Munkalapra kiírás nyomógomb után, a program megírja és kinyitja Vegyszerjelentés munkalapot.

A belépéstől, e jelentés megjelenéséig, – még lassú klikkelések esetén is, ~20 másodperc. Ez azért nem rossz!? Már csak nyomtatni kell!

4.2.2 Tudnivaló ismeretek, a téma megkezdéséhez

Ehhez a fejezethez fel kell eleveníteni a Jegyzet, elméleti bevezető fejezeteit, mert ebben a **UserForm** témakörben való munka, e kiemelt témakörök készségszintű ismerete és használata elengedhetetlen, természetesen a teljes 2 fejezet utasításai és parancsai is.

Ebben a UserForm témakörben való munka alatt teljesedik ki igazán, az Objektum-orientált programozás. nagysága, terjedelmessége és szépsége!



Újabb és újabb Objektumok megismerése – használat – megjelenítése lesz a feladat, ezek használata a Project Könyvtárban (*Project Explorer*) követhető, és a szerkesztéshez innen nyithatók ki. Egy-egy Objektumnak a szerkesztő ablakba való nyitásakor, automatikusan nyílik a Tulajdonság ablak (*Properties Windows*), az Objektum tulajdonságaival. A példában, egy VBA Project(KV-03-15.xls) Projectben, a Microsoft Excel **Objects** mappában az Excel Munkalapok fájlok, a **Forms** mappában, a Projectben alkotott Form fájlok, a **Moduls** mappában a programokat tartalmazó Modul lap fájlok láthatók.

A Fejlesztő környezet szerkesztő ablakába (jobb oldal), a kiválasztott UserForm4 képe látható, a rátelepített Controlokkal, a bal oldalon a Properties UserForm4 Tulajdonság ablakkal. Ebben az állapotban **tovább szerkeszthető**! A **Form**on levő pl. az Adatsor Tárolása feliratú nyomógomb választás után a Szerkesztő ablakba, a nyomógomb KódAblaka nyílik ki, az oda írt programsorokkal.

4.2.3 Formok és Controlok

A VBA fejlesztőrendszer alatt készíthető, felhasználói felület. Ezt az alkalmazást a szakirodalom több névvel is használja: pl. **UserInterface – Form**, vagy magyar fordítással: **Űrlap**.

A felhasználói felület egy **Form** (és maradjunk a **Form** névnél), ami **Control**ok, illetve más objektumok hordozója. (A **Control** magyar megfelelője: **Vezérlő**, de maradjunk itt is az angol névnél).

A Formokra helyezett Controlokat fogja látni a Felhasználó, és remélhetőleg interaktív módon használni is. Amikor az alkalmazások, Form felhasználói felületének megépítése után, az alkotóelemek tulajdonságainak beállítását követően megírjuk az elképzelt működéshez szükséges programkódokat, az alkalmazásunkba valójában ekkor lehelünk életet.

A Form és a Control feltételezik egymás meglétét: a Form hordozza a Controlokat. Az Üres Formnak önmagában nincs értelme, míg Control elhelyezhető közvetlenül, az Excel munkalapon is. Mind a Form, mind a Controlok rendelkeznek a saját objektumukra jellemző tulajdonságkészlettel, az adott objektum által érzékelhető eseménykészlettel. és az adott objektumon értelmezhető metódusokkal.

A **tulajdonság** adja az objektum csomagolását, a **metódus** a magatartását, az **események** pedig a felhasználó és az alkalmazás közötti dialektikus kölcsönhatást írják le.

Amikor Controlokat használunk, azt két okból is tehetjük: egyrészt felhasználói **inputként**, másrészt pedig **outputként** valamilyen információ megjelenítésére. A felhasználó tulajdonképpen a Controlokon keresztül interaktív kommunikációval használja alkalmazásunkat.

Ebben a fejezetben sok-sok példán keresztül fogunk megismerkedni a Form, illetve a Controlok tulajdonságaival, főbb metódusaival, és az egyes Controlok adta kommunikációs lehetőségekkel.

Tipp: Mivel a számítástechnika, a „**próbálkozások** tudománya” (szerzők véleménye), így az egyes tulajdonságok jelentését és beállított értékeinek hatását legegyszerűbben, próbálgatással tapasztalhatjuk ki.

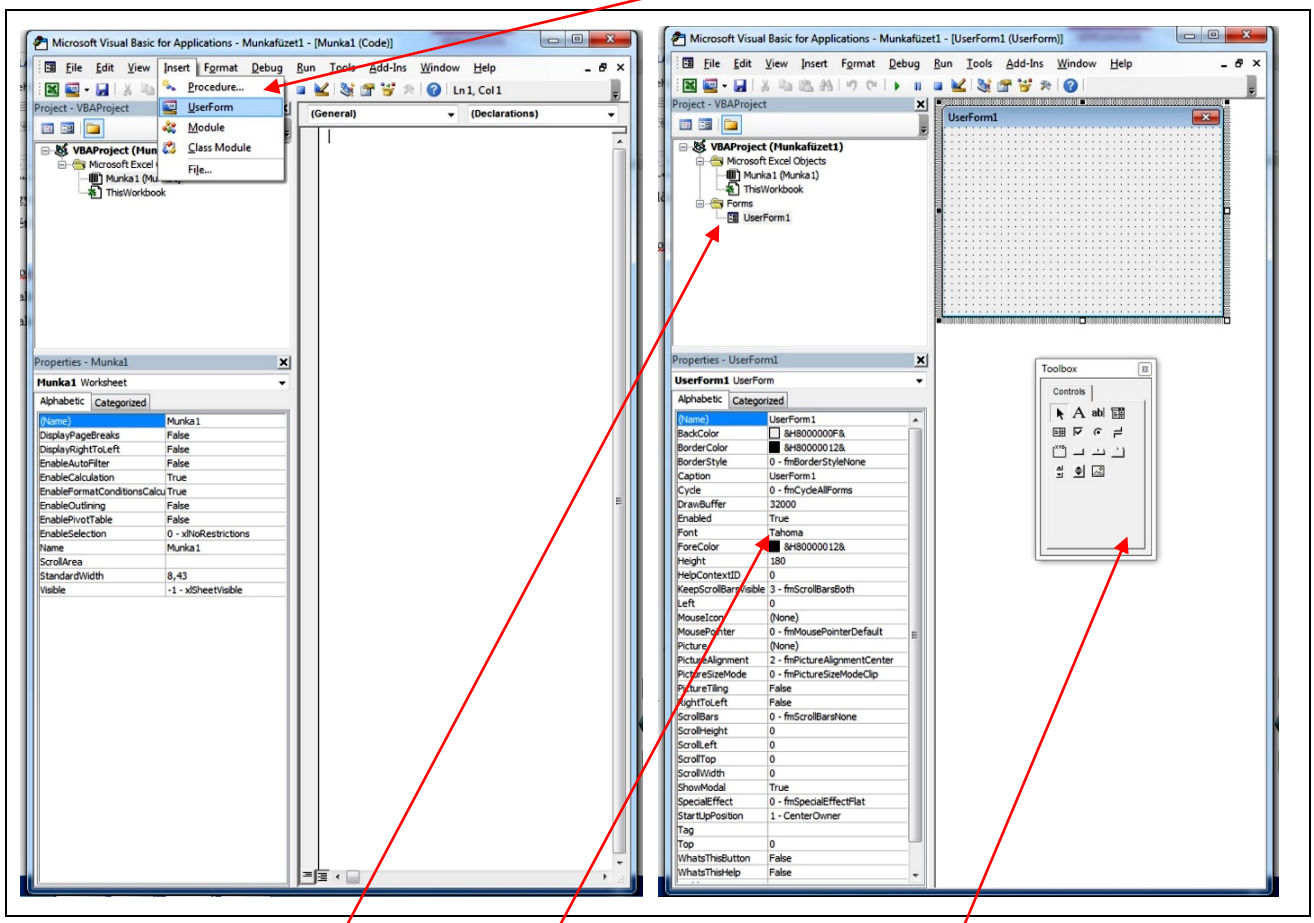
4.2.3.1 Formok

Egy alkalmazás általában egy vagy több, felhasználói interfész céljára szolgáló Formnak az alkalmazás logikája szerinti sorrendű megjelenítéséből áll. A Formok a felhasználóval való kapcsolattartás céljából adatbevitelre és/vagy megjelenítésre, illetve egyéb célokra szolgáló objektumokat tartalmaznak, mindazokkal a fejlesztő által írt programkódokkal, amelyek az objektumok standardtól eltérő működési módját írják le a VBA nyelvén, az egyes objektumokhoz illesztett esemény-kezelő eljárások VBA nyelvű programutasításokkal történő kitöltése révén. Tartalmazhatnak még olyan eljárásokat és ún. függvényeket is, amelyek maguk ugyan nem

eseménykezelő eljárások, de amelyeket – a program jobb áttekinthetőségét elősegítendő – a Formon belüli eljárások hívhatnak meg. Ezek olyan kódokat tartalmaznak, melyeket – nem külön eljárásban történő megfogalmazásuk esetén – az összes hívási helyre szinte változatlanul kellene beilleszteni.

A Project, a Formok és a bennük elhelyezett objektumok, a bárhonnan (pl. a program kódjából) történő hivatkozásuk lehetővé tétele érdekében, azonosító névvel vannak ellátva. A Formok neve Project szinten is egyedi kell, hogy legyen; a bennük levő objektumok neve viszont csak a hordozó Formon belül nem ismétlődhet (tehát két különböző, egy Projectbeli Form tartalmazhat azonos nevű objektumokat).

Egy Első vagy további Új Form felvétele, az Insert menü UserForm választással indítható.



A bal oldali felső Project-VBAProject ablakban megjelenik a Forms könyvtár és alatta a UserForm1 Form. A bal oldali alsó Properties - UserForm1 ablakban megjelennek a UserForm1 objektum tulajdonságai. A UserForm-al (Űrlappal) együtt megjelenik a ToolBox is, ahonnan a Controlokat (Vezérlőket) lehet felrakni.

4.2.3.2 Controlok – ToolBox (Eszközkezelő)

Ez a **ToolBox** objektum tartalmazza azokat az eszközöket, amelyek segítségével különféle Controlokat helyezhetünk el a Formon. Az eszköztárban minden egyes elem egy-egy Controlt reprezentál.

	Pointer	A Formon lévő Controlok kijelölésére és helyzetének meghatározására szolgál.
A	Label control	Megmutat egy szöveget, amit interaktív módon nem, de a program futása alatt változtatható.
ab|	TextBox control	Egyaránt alkalmas szöveg bevitelére és megjelenítésére
	ComboBox	TextBox és ListBox kombinációja. mindig a legördülő listából kiválasztott elemet mutatja
	ListBox	Megmutatja a lista elemeit, amely közül egyet vagy többet választhatunk.
	CheckBox	True/False, Igen/Nem választási lehetőség. A felkínált opciók nem zárják ki egymást.
	OptionButton	A felkínált választási lehetőségek közül egy időben mindig csak egy választható.
	ToggleButton	Arettál – Kétállású nyomógomb
xyz	Frame	A Frame -be tett Controlokat egy csoportba foglalva, vizuálisan is egy egységként kezeli.
	CommandButton	A felhasználói parancs választását érzékeli.
	TabSript	Egyszerűbb lapozható megjelenítő. Nem támogatja a Control csoportokat laponként
	MultiPage	Form -szerű lapozható megjelenítő. Itt már minden Control alkalmazható, laponként.
	ScrollBar	Megadott határok közti érték kiválasztására szolgál, görgető sávként is kapcsolódik a controlokhoz.
	SpinButton	Fel / Le léptető nyomógomb
	Image	Kép-megjelenítő control. Kevesebb lehetőséget biztosít, mint a VB-ben ismert PictureBox.

4.2.4 Építsed a Formot

E Jegyzetnek célja az egyéni tanulás segítése, a meglevő szakirodalmak kijegyzetelése annak érdekében, hogy a kezdő hallgató ennek alapján már eligazodhasson más kiadványokban. Tehát a Jegyzet nem helyettesíti az Irodalomjegyzék kiadványait, és nem is törekszik a Formok – Controlok, „Tételes – Katalógusszerű” ismertetésére, a tulajdonság és alkalmazás tekintetében.

Fontos:

E Form–Control témakörben, az Alkalmazás alapjai, az induló lépései, lehetőségei, egy **Tanuló Form** építése folyamán kerül bemutatásra, a szükséges magyarázatokkal, szoftvertámogatással és a működési bemutatókkal. A további gyakorlat megszerzése, már saját feladat megoldása, saját erőből lehetséges!

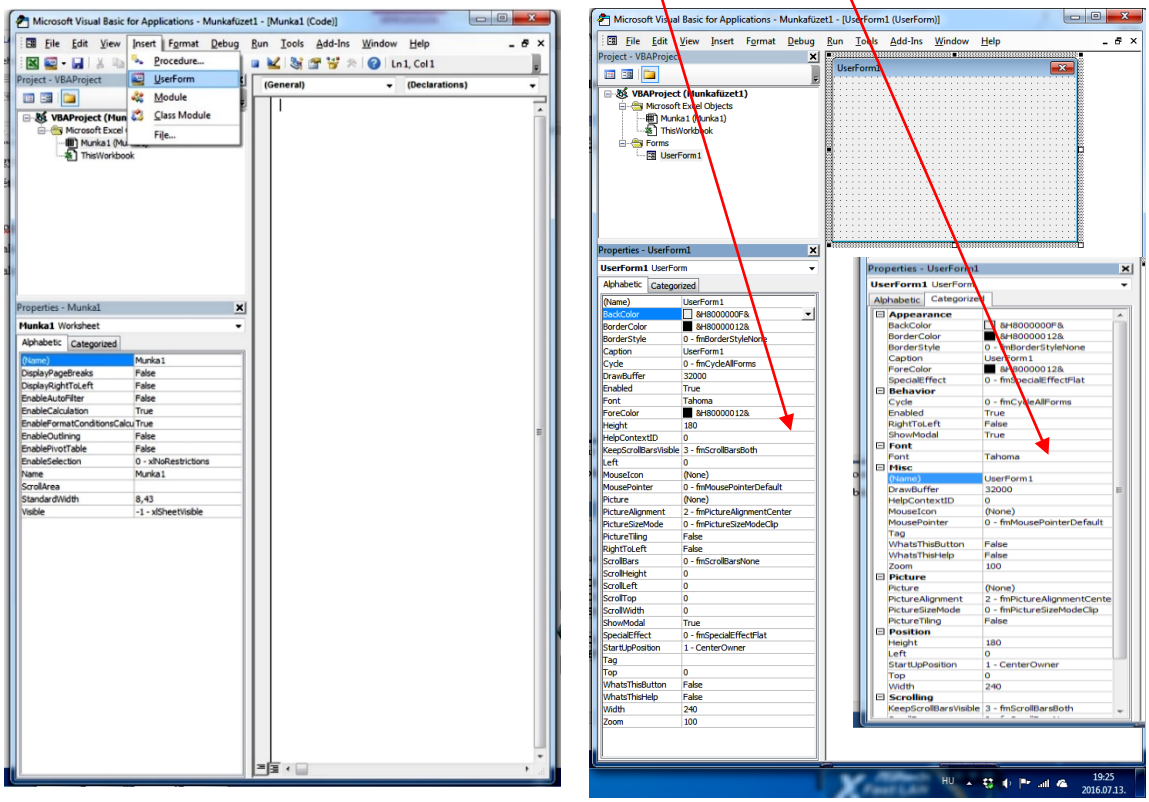
4.2.5 Tanuló Form – TesztForm1 létrehozása – 4 lépésben

4.2.5.1 TesztForm1 létrehozása 4/1 lépés

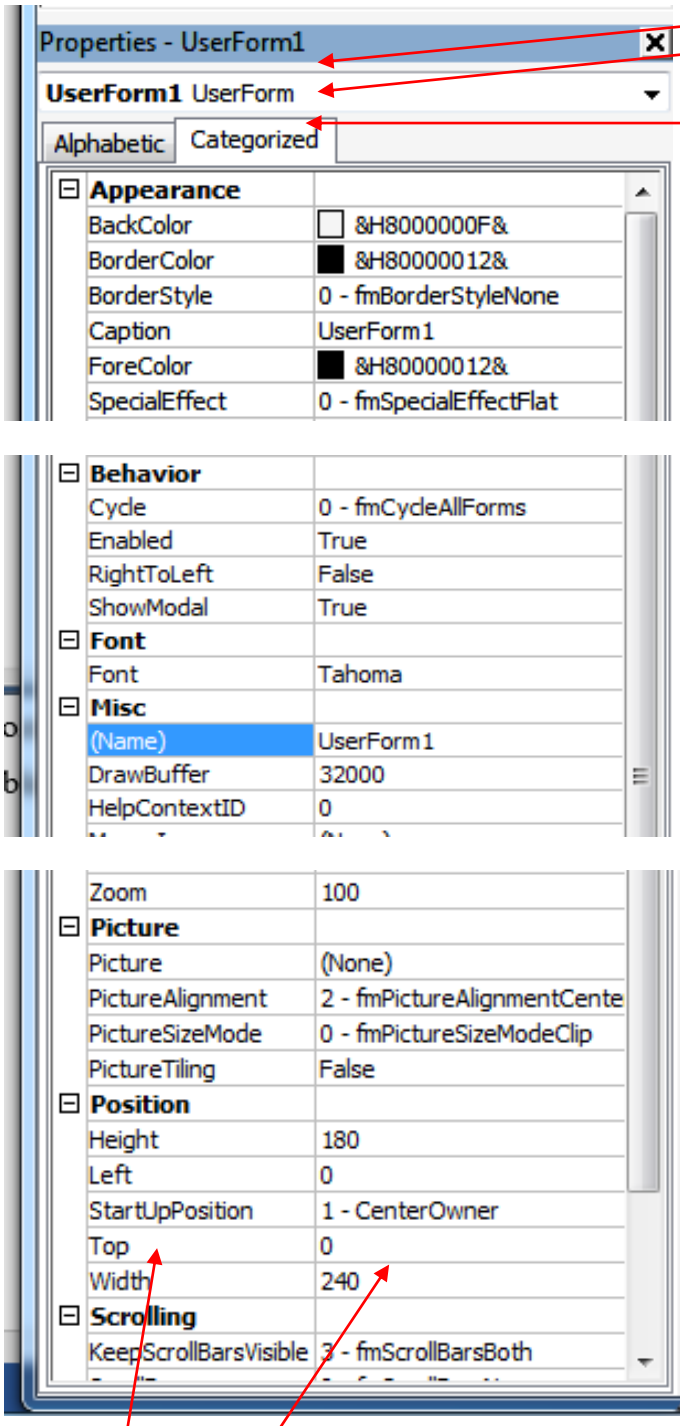
Az előzőekben már ismertettük a Form létrehozását és az önműködően nyíló Properties – Windows – Tulajdonság ablakot. Ez tartalmazza egy kiválasztott Formhoz vagy Controlhoz rendelt tulajdonságainak listáját. Ezen tulajdonságoknak a készlete objektívfüggő, hiszen minden egyes objektumnak más és más a funkciója.

Vannak olyan jellemzők, amelyek minden Form, illetve Control tulajdonságkészletének részét képezik, pl. Kinézeti – Viselkedési – Betűtípus – Megjelenési (Pozíció) tulajdonságok, de minden objektumnak vannak, a csak arra az objektumra jellemző tulajdonságok.

A **Propertis Window** ablak, a tulajdonságokat **Betűrendben** és **Kategória** csoportokban tartalmazza! A programozó szabadon választhat magának „**Listatípust**”, amivel dolgozni szeretne.



A következőben a UserForm- hoz tartozó Tulajdonság lista fontosabbjait mutatjuk be.



Form Programban levő neve (Name)
Form megnevezése és Típusa
Form tulajdonságai,
Alfabetic vagy Categizited sorrendben
Kinézet tulajdonságok
Háttérszín
Keret szín
Keret Stílusa
Form felirati szövege (Form Címe)
Form felirat szöveg színe

Viselkedést tulajdonságok
Form gömbölyítési lehetőségei
Form True/Fase – Aktív/Passzív

Betűtípus tulajdonságok
Font felirati szöveg típusa
Vegyes tulajdonságok
Formnak a programban használt neve,
(Átnevezhető, de a kezdőnek nem ajánlott)

Form megjelenő képének nagyítása
Kép tulajdonságok
Formra helyezhető kép választása
Formra helyezhető kép helyzete
Formra helyezhető kép módozatai

Pozíció tulajdonságok
Form magasság mérete
Form bal-felső csúcsának távolsága balról
Form pozicionálás lehetőségei
Form bal-felső csúcsának távolsága fentről.
Form szélesség mérete
Görgetés tulajdonságok

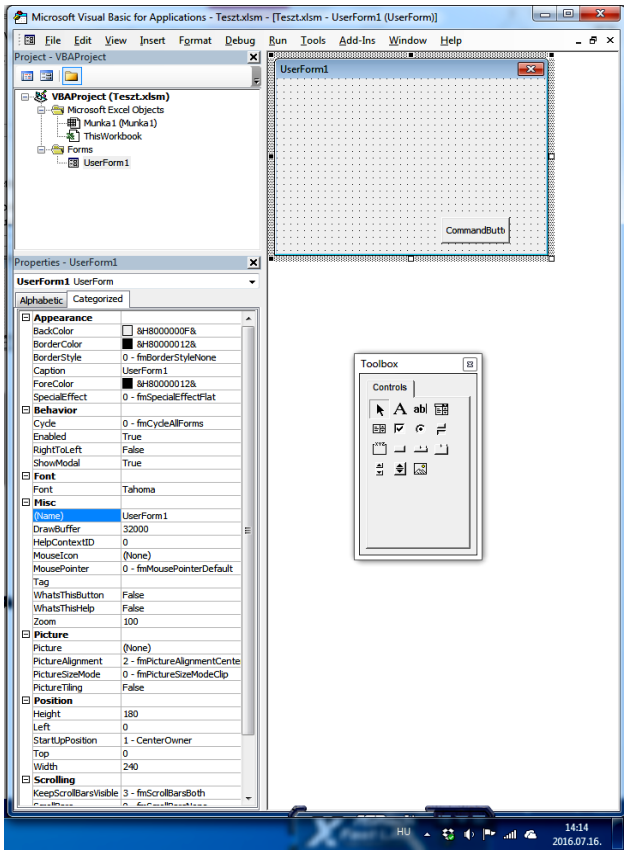
A Properties ablaknak két oszlopa van:

Tulajdonság lista: az ablak bal oldali oszlopa, az objektum választható tulajdonságainak listája,

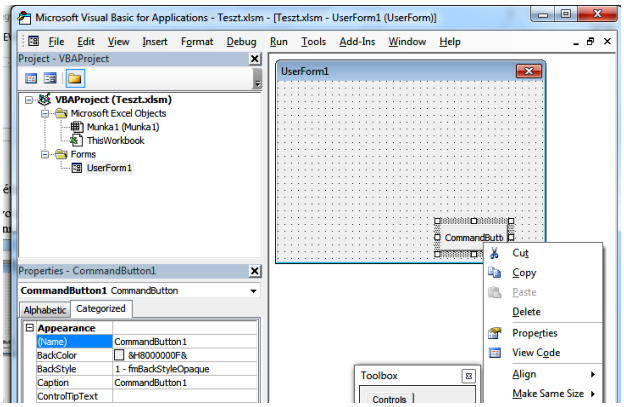
Beállító mező: az ablak jobb oldali oszlopa, mely a tulajdonságok éppen aktuális tulajdonságát mutatja. A mutatott tulajdonságok innen állíthatók, módosíthatók úgy, hogy beleklikkel a kívánt sorba. A tulajdonság jellege szerint, választható lista indul (ebből kell választani), vagy szabadon beírható adatbeírásra van lehetőség. A beállító mező beállításainak megismerése legcélszerűbben úgy valósítható meg, hogy rögtön van lehetőség az élőkép megjelenítése, az Excel munkalap előtt.

4.2.5.2 TesztForm1 létrehozása.4/2 lépés

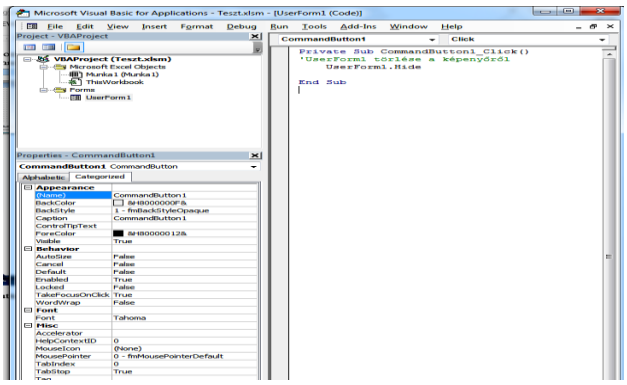
Helyezzünk egy CommandButton nyomógombot a UserForm1 formra. A ToolBox, CommandButton ikonjára klikkelve, a kurzort fel kell vinni a formra, ahol a kurzor „+” re vált.



A kívánt helyre mozgatás után, a bal-egérgomb nyomása alatt felrajzolható a CommandButton mérete: a bal felső ponttól, a jobb alsó pont kijelöléssel. Az egérgomb elengedésével kirajzolódik a Command Button képe, kinyílik a Tulajdonság ablaka, és a nyomógombon megjelenik a Caption tulajdonság (*default*) felirata: CommandButton1



A **jobb** egérgombbal klikkelve a nyomógombon, kinyílik egy menü, ahonnan **bal** egérgomb klikkel, a View Code menüt kell választani. Ekkor a szerkesztő ablakban a form helyett a nyomógomb CommandButton1 Kód-ablaka nyílik ki.



A Kódablakban megjelenik a nyomógomb Subrutinjának Sub és End sora. Ide kell beírni a nyomógombra érvényes Program-utasításokat (Program-törzset), mely a nyomógombra adott, **bal** egérgombbal adott Klikk esemény hatására fut le. A Sub sor után, ajánlatos azonnal egy „komment” sort tenni, utalva a klikk esemény feladatára. Ez eltérő (rendszerint zöld) színnel jelenik meg.

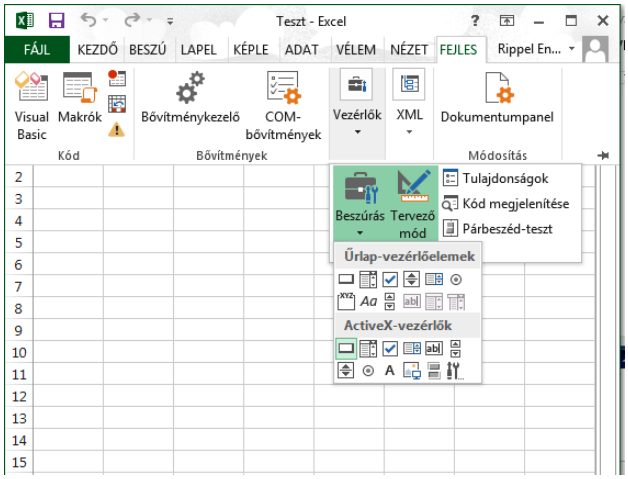
Programtörzsnek Hide metódushívás kell írni, feladata, a Userform levétele a képernyőről. .

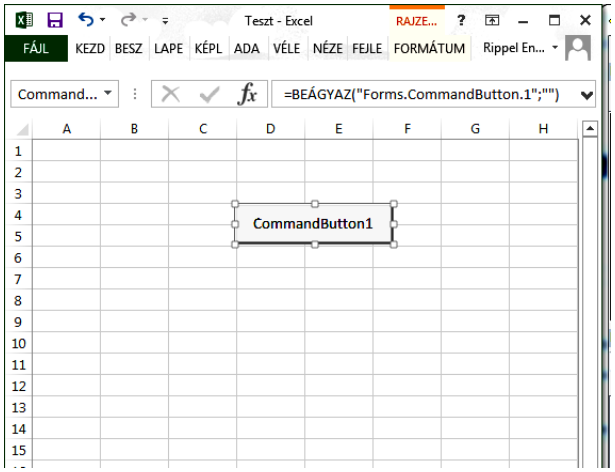
```
<Objektum>.<Metódus>.<paraméter>
Private Sub CommandButton1_Click()
'UserForm1 törlése a képenyőről
    UserForm1.Hide
End Sub
```

4.2.5.3 TesztForm1 létrehozása 4/3 lépés

Ebben a lépésben az eddig kialakított UserForm megjelenítése a feladat. – a UserForm futtatása!

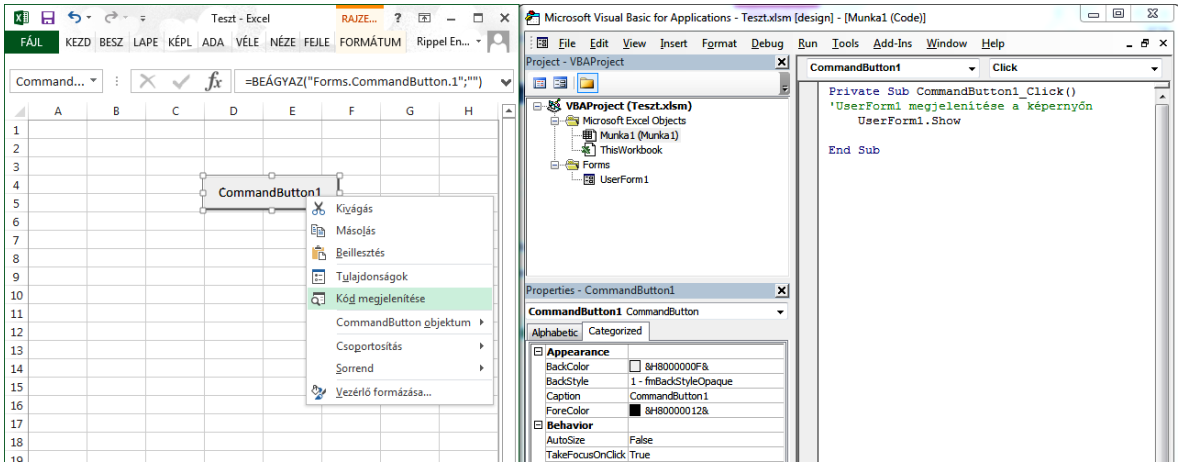
A feladatot, az Excel munkalapra felrakott CommandButton Control (Vezérlő) nyomógombbal lehet megoldani. A felrakás menete: Fejlesztőeszközök menü/Tervező mód ikon, és Fejlesztőeszközök menü/ Beszúrás ikon/ActiveX vezérlők közül/CommandButton vezérlő választással. A kurzor „+” re vált, fel kell mozgatni az Excel munkalap kívánt helyére, ezután a **bal** egérgomb nyomása alatt felrajzolható a CommandButton mérete, a bal felső ponttól, a jobb alsó pont kijelöléssel. Az egérgomb elengedésével kirajzolódik a Command Button képe, kinyílik a Tulajdonság ablaka, és a nyomógombon megjelenik a Caption tulajdonság (default) felirata: CommandButton1





Ezután, – mivel a Tervezőmód be van kapcsolva, ilyenkor a munkalapra helyezett Controlok (Vezérlők) szerkeszthető állapotban vannak –, **jobb** egérgomb klikkeléssel, menü ablak nyílik le, amiből **bal** egér klikkel, a Kódablak nyílik. A Kódablakban megjelenik a nyomógomb Subrutinjának Sub és End sora.

Ide kell beírni a Programutasításokat. A Tervezőmód kikapcsolása után, az Excel munkalapon elhelyezett nyomógomb aktívvá válik, és a rá adott. klikk esemény hatására fut le a program.



A Sub sor után, ajánlatos azonnal egy komment sor írása. Programtörzsnek itt egy Show metódus-hívást kell írni. Írásmód:

```
<Objektum>.<Metódus>.<paraméter>
```

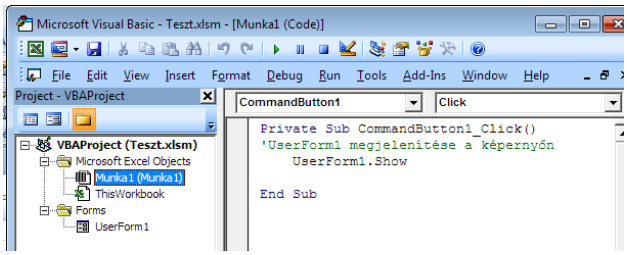
A Show metódus feladata csak a Userform megjelenítése a képernyőn, így nincs paramétere.

```
Private Sub CommandButton1_Click()
'UserForm1 megjelenítése a képernyőről
    UserForm1.Show
End Sub
```

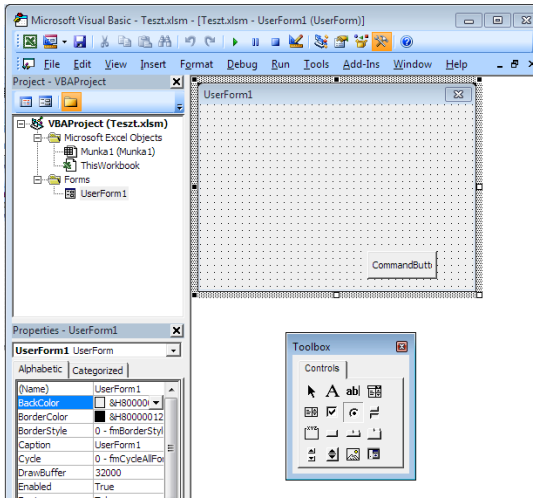
4.2.5.4 TesztForm1 létrehozása 4/4 lépés

Most **nyers** állapotban rendelkezésre áll ez a Teszt Pojekt, kezdődhet a Tulajdonságok beállítása.

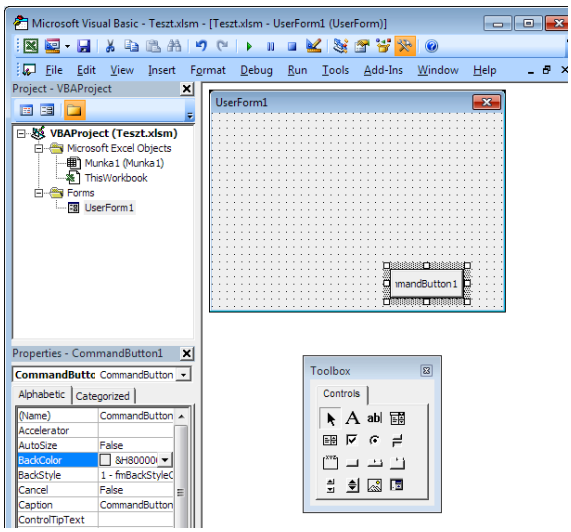
Először a Project ablakot vegyük szemügyre:



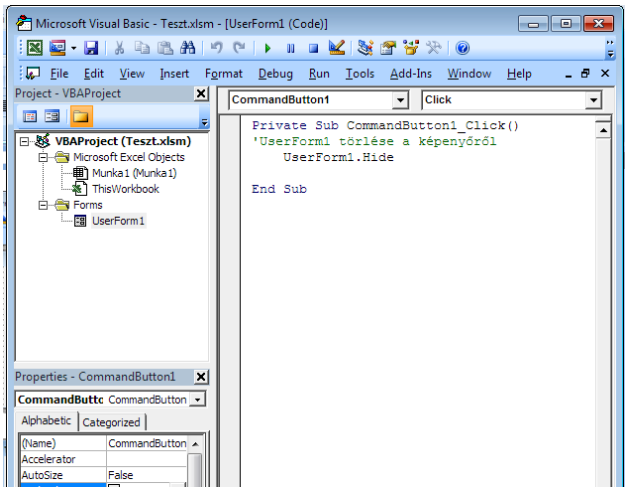
Munka1-re klikkelve a szerkesztőben megjelenik az Excel Munka1 munkalapra tett nyomógomb Subrutinja.



UserForm1-re klikkelve a szerkesztőben megjelenik a Form a rajta levő nyomógombbal, és a Project ablak alatt kinyílik a UserForm1 tulajdonság ablaka. Ott kell megadni a kezdő tulajdonságokat, amelyeket azután bármikor javítgatni lehet még.



CommandButton1-re klikkelve, a Project ablak alatt a nyomógomb tulajdonság ablaka nyílik meg.



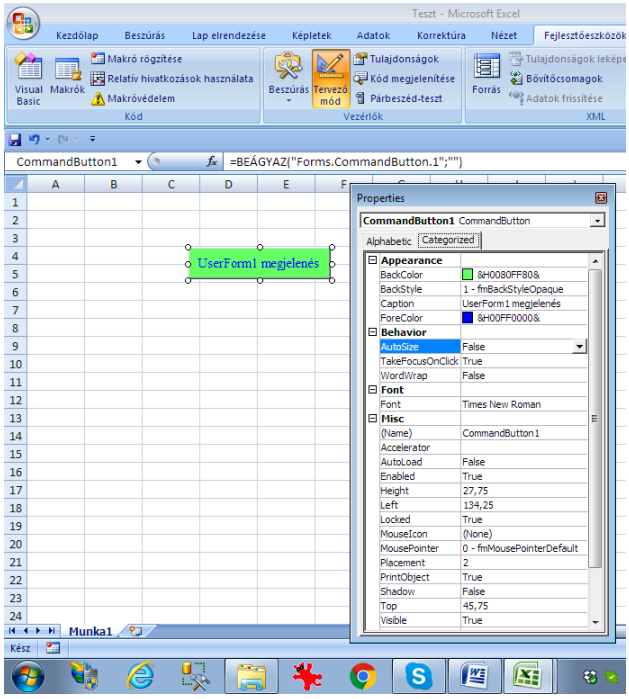
a nyomógombra adott Double Klikk hatására a szerkesztőben a CommandButton1 nyomógomb Subrutinja nyílik ki és szerkeszthetővé válik.

Ez a Project ablakból való választás értelemszerűen igaz a sok Munkalapos Excel, a sok Formot alkalmazó program esetében is. Tehát ez az ablak lesz használva, az egész programírás alatt. Ebből kiindulva lehet programot írni, formokat építeni, tulajdonságokat állítani.

Következő feladat, hogy a nyers kialakítást hogyan öltöztessük fel a kezdő látvánnyal, tulajdonságokkal.

4.2.6 Beállítások

4.2.6.1 Excel Munkalapon levő nyomógomb állítása



Tervezőmód bekapcsolás: **jobb** egér klikkel Tulajdonság ablak kérése

BackColor állítása: klikk a BackColor-ra, klikk a választó nyilacskára, klikk a Palettára és klikk a kívánt színre.

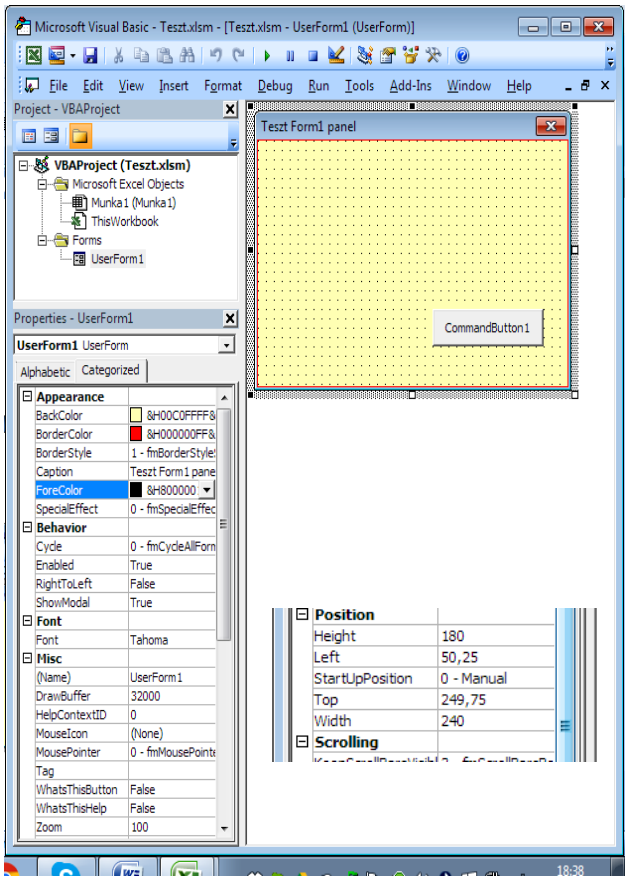
Caption tulajdonság: klikk a Caption-ra és a szerkesztő sávban kell megadni a nyomógomb feliratát.(ha nem fér ki, akkor igazítani kell a nyomógomb méretét.)

Font állítása: klikk a Font-ra, majd a menüből kell választani.

ForeColor állítása: klikk a ForeColor-ra, klikk a választó nyilacskára, klikk a Palettára és klikk a kívánt színre

Tervezőmód kikapcsolás

4.2.6.2 UserForm1 beállítása



BackColor állítása: klikk a BackColor-ra, klikk a választó nyilacskára, klikk a Palettára és klikk a kívánt színre.

BorderColor állítása: klikk a BorderColor-ra, klikk a választó nyilacskára, klikk a Palettára és klikk a kívánt színre

BorderStyle állítása: klikk a BorderStyle-ra, klikk a választó nyilacskára, klikk az 1 típusra.

Caption tulajdonság: klikk a Caption-ra és a szerkesztő sávban kell megadni a Panel címét, pl. Teszt Form1 panel

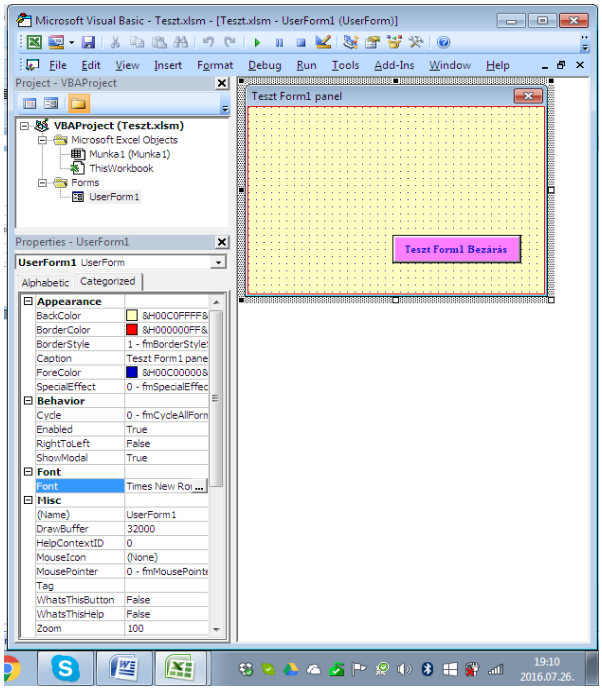
Meg kell adni a megjelenés Pozíciót**.**

StartUpPosition: klikk a StartUpPosition-ra, lenyíló ablakban vannak kész típusok. Ha egyéni helyre kell megjelenni, akkor a 0-Manual típust kell választani.

Left állítása: klikk a Left-re és be kell írni a bal felsősarkának távolságát a baloldaltól, pl.: 50

Top beállítása: klikk a Top-ra és be kell írni a Form bal felsősarkának távolságát a bal felsőoldaltól, pl.: **250** (ekkor kb. a nyomógomb alatt fog megjelenni.

4.2.6.3 Teszt Form1 nyomógombjának beállítás



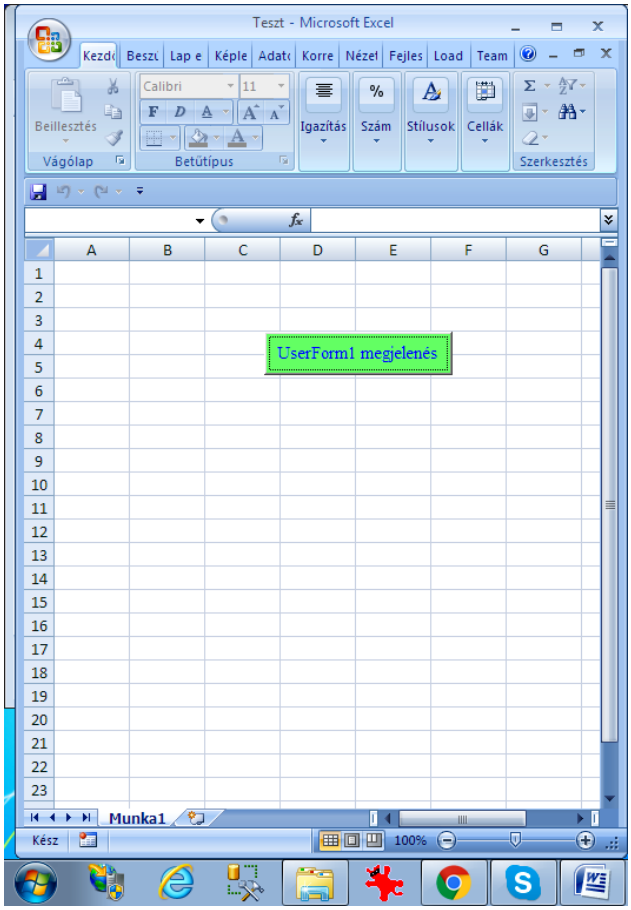
BackColor állítása: klikk a BackColor-ra, klikk a választó nyilacskára, klikk a Palettára és klikk a kívánt színre.

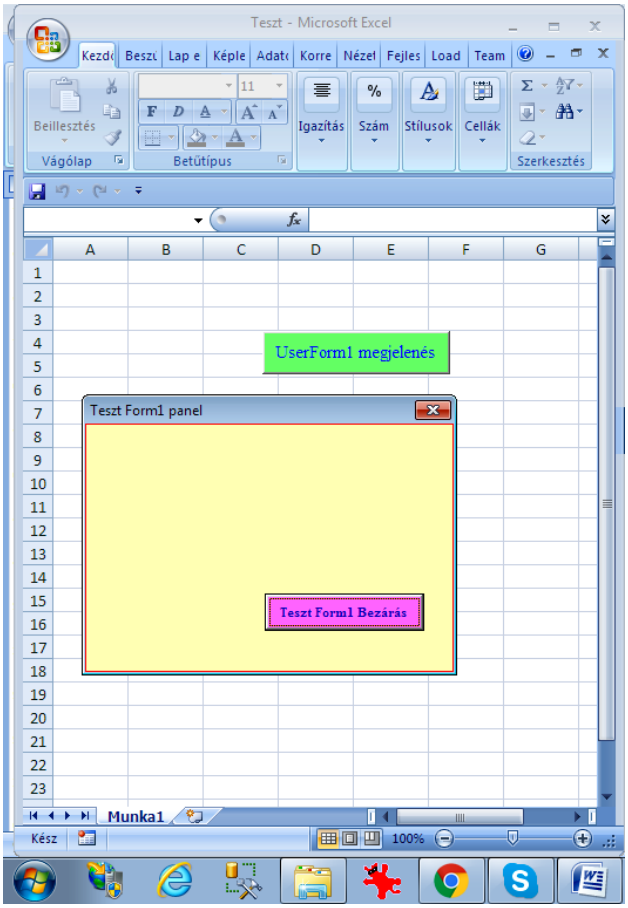
Caption tulajdonság: klikk a Caption-ra és a szerkesztő sávban kell megadni a nyomógomb feliratát.(ha nem fér ki, akkor igazítani kell a nyomógomb méretét.)

Font állítása: klikk a Font-ra, majd a menüből kell választani.

ForeColor állítása: klikk a ForeColor-ra, klikk a választó nyilacskára, klikk a Palettára és klikk a kívánt színre.

Ezzel készen van a TesztForm1 a próbákra.





Az Excel munkalapon a UserForm1 megjelenése gombra klikkelve, a TesztForm1 panel megjelenik.

A TesztForm1-en a Tesztform1 Bezárása gombra klikkelve, a Form eltűnik!

Ezzel egy próba Form áll rendelkezésre, melyre további Vezérlő (*Control*) elemek rakhatók fel, Subrutinjaik megírása és tulajdonságaik beállítása után, azonnal élesben tesztelhető a működés. Ezzel a módszerrel a Control Tulajdonságok mind végig próbálhatók.

4.2.7 TesztForm1 bővítése Vezérlőelemekkel

A Jegyzet további részeiben csak a Vezérlők felhelyezését és egy-két jellemző alkalmazás Subrutin írást mutat be úgy, hogy azokból a működés megérthető legyen, és az egyéni további tanulmányozás, gyakorlás teljesíti majd ki a Vezérlőnek (Controlnak) jobb ismeretét. A sok Vezérlő egyenkénti különböző célokat szolgáló feladatából adódóan, a tulajdonságok és az alkalmazások sokrétű részletes bemutatása e jegyzetben nem lehetséges, ezt csak a szakirodalomból lehet csak egyéni munkával megszerezni.

4.2.7.1 Bővítés: Label, TextBox, CommandButton vezérlőkkel

A Formra telepített Vezérlőket, de magát a Formot is, a VBA program Objektumként kezeli, ezek érzékelik az Egér kurzorát, klikk vagy dupla-klikk eseményre reagálva, futtatják le a hozzájuk kapcsolt Subrutint. A Subrutinban leirt eljárással már vezérlők tulajdonságai módosíthatók.

1) Label (címke) vezérlés

Felhasználói interfész, a Formon **szövegnek** akár statikusan, akár dinamikusan – azaz program-futás közben változtathatóan – történő megjelenítésére szolgál. Tartalma a program futása közben csak programozással módosítható. Formázható is (ám csak teljes egészében). Az e vezérléssel létrehozott objektum a címkemező.

Szövegét a **Caption** tulajdonságban kell megadni, a tulajdonság ablakszerkesztő sáv oszlopában, vagy a program futása közben, a Caption megadásával.

2) TextBox (szöveg) vezérlés

Elsősorban a felhasználói interfész Formok szöveges adatbeviteli mezőinek létrehozására szolgál, de képes információk szöveges megjelenítésére is.

A vezérléssel létrehozott objektumokat szöveg- vagy adatbeviteli mezőknek nevezzük.

Tipikus felhasználási területe a nagy-tömegű adatbevitel, melynek célja lehet pl. egy táblázat feltöltése nagyszámú, azonos típusú objektum tulajdonságainak értékeivel.

3) CommandButton (nyomógomb) vezérlés

A nyomógomb-objektumok feladata a program egyes komplex folyamatainak elindítása vagy lezárása számára egyértelműen megkülönböztethető, és felhasználói interfész eszközökkel könnyen használható lehetőséget biztosítani.

Használatának tipikus céljai:

- folyamat lezárása, elfogadtatással (szokásos szövegek: OK, Apply vagy End, de magyar feliratok is adhatók: Kész – Tárol és kilép – Mentés, stb.);
- a folyamatból történő visszalépés, vagy törlés (szövege: Cancel, vagy Kilép – Mégse Kilép, stb.);
- részfolyamatból vagy programból történő kilépés (szövegei: Exit, Quit vagy End).

Fontos meggondolások a nyomógombok funkcióihoz:

OK – Kész nyomógombok feladata összetett tevékenységet lát el. A Form vezérlőivel végzett adatgyűjtés befejezése, a begyűjtött adatok tárolása a kijelölt helyekre, (Excel munkalap – *.txt fájl – Rekordok), és majd a Form levétele a képernyőről (Hide metódus hívással).

Cancel – Kilép nyomógombnak is összetett feladata van. **Fontos,** hogy szabályosan zárja le a megkezdett folyamatokat, – az indulási alapra kell törölni, állítani a Form vezérlő eszközeit, majd le kell venni a képernyőről a Formot. (Hide metódus hívással).

Sajnos a Form a Windows piros × -re klikkeléssel is eltűnik a képernyőről, de ez sokszor okoz problémát, mert ez a bezárási mód, nem intézkedik a VBA program alapra állítására. Ezért, ezt az eltüntetési módot kerülni kell, és a programot Felhasználóknak is fel kell hívni a figyelmét erre, és a piros × -es bezárást tiltani kell.

A) Feladat 1 leírása

a) TesztForm1-re kerüljön fel egy:

- Label, amely a feladat nevet mutatja, piros, 14-es Bold betűvel.
- TextBox, amelybe egy szöveget lehet beleírni.
- Label, amely fogadni tudja a TextBox szövegét.
- CommandButton, amely a klikk eseményével a TextBox-ba irt szöveget átírja a második Label vezérlőbe, majd törli a TextBox szövegét.

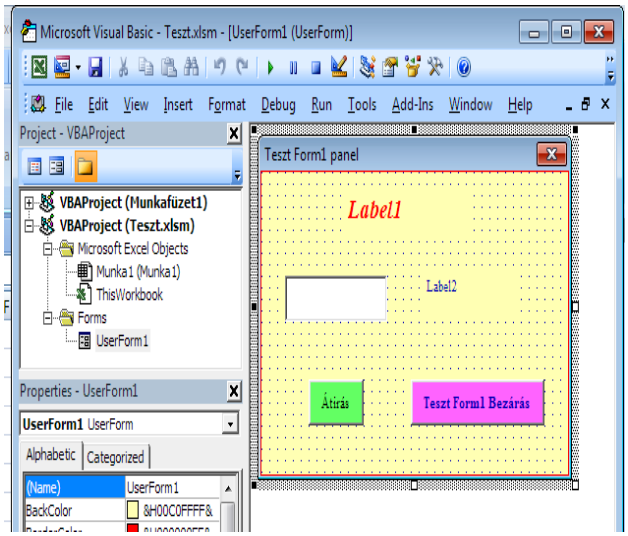
b) A TesztForm1 Kilép Bezárás nyomógombja, a Hide metódus hívása előtt, törölje a TextBoxot üresre.

c) Az Excel munkalapon levő Megjelenés nyomógomb, a Show metódus hívása előtt:

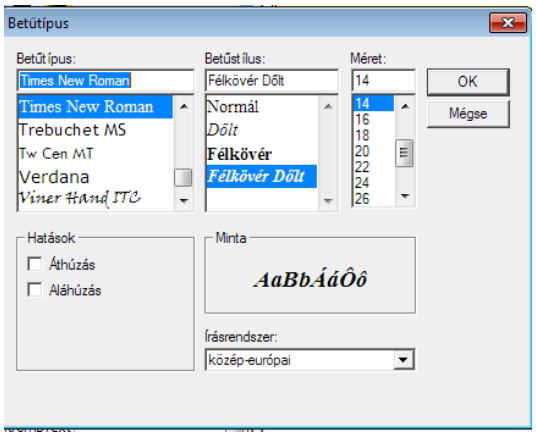
- Írja be a TesztForm Label vezérlőjébe a feladat megnevezését.
- Törölje a TesztForm TextBox-ot
- Induló szöveget írjon, a TesztForm adatfogadó Label vezérlőbe.
- A TextForm1 bővítése az első feladat.

B) Feladat 1 megvalósítása

A ToolBox-ból fel kell tenni egy Label vezérlőt, ezt a szerkesztő mindjárt Label1 névvel látja el, és minden további Label vezérlő a következő sorszámmal lesz kiegészítve. (Ugyanez igaz a többi vezérlőnél is). Ezután a TextBox1 kerüljön fel, majd a Label2 és végül a CommandButton2. Ezután a megjelenési tulajdonságokat kell beállítani, vezérlőnként.



Label1 beállítása: **Font** állítása:



ForeColor állítása: klikk a ForeColor-ra, klikk a választó nyilacskára, klikk a Palettára és klikk a piros színre

Label2 beállítása: **ForeColor** állítása: klikk a ForeColor-ra, klikk a választó nyilacskára, klikk a Palettára és klikk a kék színre

CommandButton2 állítása: **ForeColor** állítása: klikk a ForeColor-ra, klikk a választó nyilacskára, klikk a Palettára és klikk a zöld színre

Caption állítása: Átírás beírása

A Vezérlők Subrutinjainak módosítása, írása

```
Private Sub CommandButton1_Click()
'TesztForm1 Bezár
    'TextBox1 törlése
    UserForm1.TextBox1.Text = „„
    UserForm1.Hide
End Sub
```

```
Private Sub CommandButton2_Click()
    'Átírás nyomógomb
    'Átírás TextBox-ból Label-be
    UserForm1.Label2.Caption = UserForm1.TextBox1.Text
    'TextBox.Text törlése
    UserForm1.TextBox1.Text = „„
End Sub
```

Az Excel munkalap nyomógomb Subrutin bővítése:

```
Private Sub CommandButton1_Click()
' UserForm1 megjelenítése a képernyőn. UserForm feltöltése a kiinduló adatokkal
    UserForm1.Label1.Caption = „A) Feladat”          ' Feladatnév beírása Label1-be
    UserForm1.Label2.Caption = „Induló szöveg”       ' Induló szöveg írás a Label2-be
    UserForm1.TextBox1.Text = „„                     ' TextBox1 törlése
    UserForm1.Show                                   ' UserForm1 megnyitása
End Sub
```

Ebben a fejezetben, az egyes parancssorok után magyarázó kommentek vannak az egyéni tanulás, gyakorlás kedvéért.

A most elkészült TesztForm1-et lehet próbálgatni!

4.2.7.2 Bővítés: Objektum-vezérléssel

A VBA alkalmazásban minden Objektum, még az alapvetően csak passzív, megjelenítő Objektumok is lehetnek aktív vezérlés részesei. Fogadhatnak egér-klikk eseményt, és a Subrutinjukba irt, eljárást végrehajtják.

UserForm és Label klikk-vezérlés bemutatása

Az objektumra egér mutatóval rámutatva és ráklikkelve, az objektum parancsnak, „klikk eseménynek” értékelés, ilyenkor az objektumhoz kapcsolt Subrutin eljárást lefuttatja. Ez a Subrutin is a strukturált felépítésű Project része, ami a Project részeként indítható, de paraméteres (de paraméter nélkül is) Call hívással meghívható.

A bemutató példában mindkettőre lesz megoldás.

C) Feladat 2 programleírás

A Feladat 2 futtatása, a TesztForm1 objektumra adott klikk esemény hatására változtassa meg az Alapszínét (*BackColor*). A Form sárga alapszínnel jelenik meg, s futtatható rajta a Feladat 1, s ennek kiegészítéseként kell megvalósítani, hogy a Formra adott egér-klikkek hatására a Form alapszín pirosra, majd fehérre és zöldre változzon. Egy ilyen 3-as forgót kell programozni úgy, hogy a meglevő Label1 és Label2 felirati színeit is változtassa a láthatóságnak megfelelően. Alkalmazni kell egy Szin számlálót, mely számolja a klikk sorozatot és 3-as ütemre redukálja, az aktuális állást pedig, egy újabb Label objektumon jelez ki. További kiegészítésként a Label objektum egér- klikk hatására Call hívással hívja meg a Form Szin váltó subrutinját, így a rá adott klikk is ugyan úgy fogják a form alapszínét változtatni.

D) Feladat 2 megoldása

A meglevő TesztForm1-re fel kell tenni egy újabb Label vezérlőt, ez a Label3 lesz. Ez lesz Szin számláló kijelzője, az feladat első részében:

- azt a Label3 vezérlőt az Excel Megjelenítő gombjának programjával kell 1-es kijelzésre állítani, a Form címet megvalósító Label1-et Feladat 2-re módosítani.
- ezután a TesztForm1 Subrutinjába kell a már tanult VBA programokkal olyan programot írni, (lásd a bemutatót), amely olvassa a Label3 Caption érétkét, ennek alapján színezi a Formot, majd növeli a Label3 értékét, előkészítve a következő klikknek, de 3 után a Label3- újra 1-re.

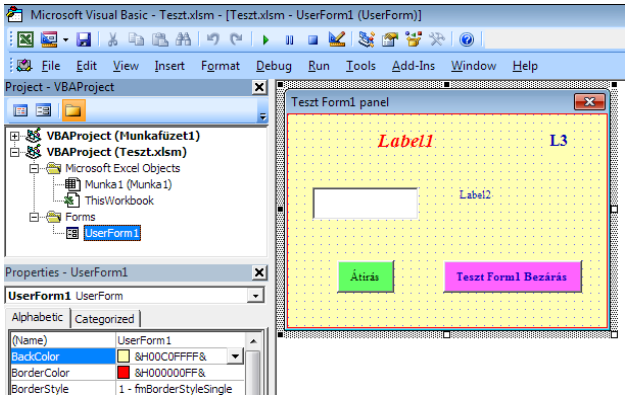
Ezzel már lehet a próbákat kezdeni.

Ha a próbaklikkelések sikeresek, akkor a Label3 Subrutinjába kell egy már tanult Call hívást beírni, mely hívja a Form Subrutinját, paraméter átadás nélkül, hiszen a Form Subrutinja minden adatot ismer.

Ezzel is lehet a próbákat kezdeni. Ha innen is működik minden, akkor kész a Feladat 2.

a) TesztForm1 bővítése TesztForm2-re.

A TesztForm1-re fel kell tenni, egy újabb Label vezérlőt, ez Label3 lesz.



Label3 állítás:

Label3.Caption = L3

Label1 és Label2 háttér színét Transzparensre kell állítani:

Label1.BackStyle = 0 Transparent

Label2.BackStyle = 0 Transparent

b) A Vezérlők Subrutinjainak módosítása, írása

Excel munkalap megjelenítő nyomógomb Subrutin módosítás (a módosítások kommentezve vannak).

```
Private Sub CommandButton1_Click()          'UserForm1 megjelenítése a képernyőn
   'Új feladatnév beírása Label1-be
   UserForm1.Label1.Caption = „A) Feladat 2”
      UserForm1.Label2.Caption = „Induló szöveg”
      UserForm1.TextBox1.Text = „„          'Label3 Szin számoló induló értéke
   UserForm1.Label3.Caption = 1             'TeszForm1 háttérszíne sárga
   UserForm1.BackColor = &HC0FFFF
   UserForm1.Show
End Sub
```

TesztForm1 Subrutin írása, módosítása:

```
Private Sub UserForm_Click()              'UserForm klikk eseménye
    Dim s As Integer
    s = UserForm1.Label3.Caption
    If s = 1 Then
        UserForm1.BackColor = &HFF
        UserForm1.Label1.ForeColor = &HFFFFFF
        s = s + 1
        UserForm1.Label3.Caption = s
    ElseIf s = 2 Then
        UserForm1.BackColor = &HFFFFFF
        UserForm1.Label1.ForeColor = &HFF8080
        s = s + 1
        UserForm1.Label3.Caption = s
    ElseIf s = 3 Then
        UserForm1.BackColor = &HC0FFC0
        UserForm1.Label1.ForeColor = &HFF8080
        s = 1
        UserForm1.Label3.Caption = s
    End If
End Sub
```

```
Private Sub Label3_Click()                ' Szín számláló
    Call UserForm_Click             'Call hívás parancssora
End Sub
```

Most már próbára alkalmas a Feladat 2.

4.2.7.3 Bővítés: Frame, CheckBox, OptionButton vezérlőkkel

Az eddig elkészült TesztForm2-re telepített Vezérlők, de maga a Form is, a további bővítés alapja. Három újabb Vezérlő felhelyezése és bemutatása ennek a pontnak a tartalma.

4) Frame (keret) vezérlés

Az általa létrehozott keret-mező elsődleges feladata az egy logikai csoportot alkotó választó-mezők (lásd előbb) funkcionális összekapcsolása, de használatos más típusú, formra helyezett mezők formai- és/vagy azonosítási célú csoportosítására is. Erre azért képes, mert a hozzá tartozó mezők számára vizuálisan is megjeleníthető (és alapértelmezésben háromdimenziós vonallal meg is jelenített) konténerül szolgál. A keret bal felső részén megjelenő szöveg a Caption tulajdonságában adandó meg, és általában a kereten belül elhelyezett információ-halmaz összefoglaló megnevezését tartalmazza.

5) CheckBox (jelölő) vezérlés

Logikai-jellegű (Igen / Nem) információ állítását lehetővé tevő, felhasználói interfész funkciójú (kapcsoló-mezők létrehozására szolgáló vezérlés-típusok. A jelölő-mező egy kis négyzet, aminek három látható állapota van:

- a szürkén kitöltött négyzetben levő szürke pipa valamiféle meghatározatlan, vagy készenléti (*stand* by) állapotot jelez.
- a négyzet pipát tartalmaz a mező bekapcsolt állapotában van.
- a négyzet üres (pontosabban fehér színnel kitöltött) a mező kikapcsolásakor.

A jelölőmező objektum a négyzet mellett szöveget is tartalmazhat, amely praktikusan a mező funkcióját magyarázza. A jelölő-objektum bármely részére történő kattintás-, vagy a szóköz billentyű egyszeri lenyomása a fókuszban levő mező állapotát – bármi programozás nélkül – váltakozóan (*toggle* jellegűen) pipátlanból bekapcsolt, bekapcsoltból pedig kikapcsolt állapotba állítja. Az egy Formon levő jelölő-négyzetek normális használatuk esetén egymástól független működtetést tesznek lehetővé.

6) OptionButton (választó) vezérlés

Egymást kizáró lehetőségek közül egynek a kiválasztását lehetővé tevő, felhasználói interfész funkciójú kapcsoló-mezők vezérlés-prototípusa.

A kapcsoló-mezőkből a különböző lehetőség-halmazoknak megfelelően **diszjunkt** csoportok képzendők, az egyes csoportokat egymástól vizuálisan is elkülönítő konténereik által. Az így létrehozott választó-mező csoportok működtetését a Windows automatizmusokkal hatékonyan támogatja; különösen, ha az egyes csoportbeli választó-mező objektumok kezelése nem különálló elemekként-, hanem objektum-tömbökbe (lásd előbb) összefogva történik. E rész a választó-csoportok különálló kapcsolókkal megvalósított használatát mutatja be.

A működtethető (azaz az **Enabled** tulajdonságában **True** értékű) választó-mező egy kis kör, aminek a közepén levő pötty jelzi a vele azonos csoportba tartozók közül az aktuálisan kiválasztottat. Nem szükséges, hogy a csoportban mindig legyen egy kiválasztott mező, azonban e készenléti (*standby*) állapot csak programozás útján érhető el.

E) Feladat 3 programleírás

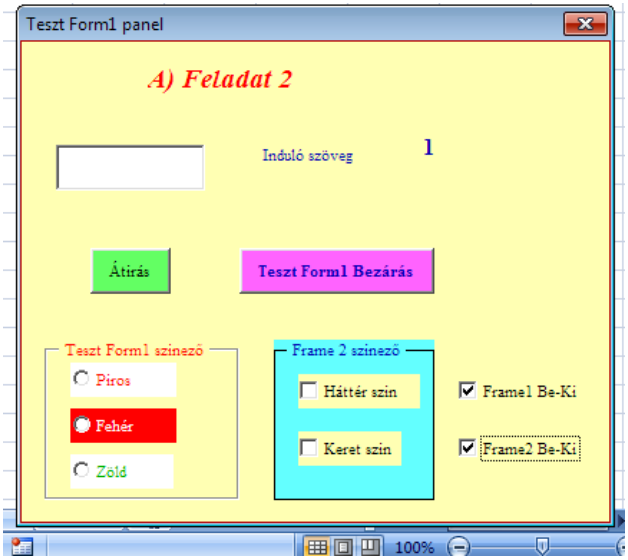
A Feladat 2 futtatása után, bővíteni kell a feladatot a Feladat 3 leírással. Ebben a Frame (keret) vezérlő, valamint az OptionButton (választó) és a CheckBox (választó) vezérlő használatával egészül ki a feladat.

- egyik Frame-be berakott 3 db OptionButton a Subrutinjaikkal kapcsolódjanak a Feladat 1-ben megvalósított TesztForm1 háttér színének mozgatásához. (piros – fehér – zöld).

- másik Frame-be berakott 2 db CheckBox-szal, a saját Frame vezérlő Háttér színét illetve a Keretének színét váltogassa Subrutinba irt program.
- Framen kívül levő 2 db CheckBox-szal, a két Frame láthatóságát (Visible) mozgassa.

F) Feladat 3 megoldása:

A meglevő, a Feladat 2-t megjelenítő TesztForm1-re kell a Feladat 3 kiírását megvalósítani:



- a TesztForm1-et a jobb alsó sarkát megfogva meg kell nagyobbítani. Az addig rajta levő vezérlők helyzete nem változik, mert azok a bal felső saroktól pozícionáltak.
- fel kell tenni a Frame1-t és bele 3 db OptionButtont (1-2-3)
- fel kell tenni a Frame2-t és bele 2 db CheckBoxot (1-2)
- fel kell tenni Framen kívül 2 db CheckBoxot (3-4)
- a felrakott Vezérlők látható tulajdonságait be kell állítani a Form képen láthatónak megfelelően. (Caption – BackColor – BorderColor)
- a 3 OptionButton egymást kizáró választási lehetőség adja azt, hogy programjaikkal, bekapcsolódjanak a TesztForm háttér színének váltogatási programjába. E kapcsolódást, a TesztForm-on levő Szin számláló – Label3, 1-, 2-, 3-ba állításával, – a Label3.Caption = (1-2-3) – és utána a Call hívással meghívni a UserForm Szin váltogató rutinját.

a) TesztForm1 bővítése Feladat 3-ra, a Vezérlők megjelenésének beálltásával kezdődik

```
Private Sub OptionButton1_Click()       'UserForm1 színező Piros
    UserForm1.Label3.Caption = 1
    Call UserForm_Click
End Sub
```

```
Private Sub OptionButton2_Click()       'UserForm1 színező Fehér
    UserForm1.Label3.Caption = 2
    Call UserForm_Click
End Sub
```

```
Private Sub OptionButton3_Click()       'UserForm1 színező Zöld
    UserForm1.Label3.Caption = 3
    Call UserForm_Click
End Sub
```

A Frame2-be rakott 2db CheckBox (CheckBox1 és CheckBox2) – egymástól független működtetési lehetőség – adja azt, hogy programjaikkal, a saját Frame2 háttér és keret színének váltogatását oldják meg programjaikkal.

```
Private Sub CheckBox1_Click()          'Frame háttér színező
    If CheckBox1.Value = True Then
        UserForm1.Frame2.BackColor = &H80FF80
    Else
        UserForm1.Frame2.BackColor = &HFFFF80
    End If
End Sub
```

```
Private Sub CheckBox2_Click()          'Frame keret színező
    If CheckBox2.Value = True Then
        UserForm1.Frame2.BorderColor = &HFF&
    Else
        UserForm1.Frame2.BorderColor = &H80000012
    End If
End Sub
```

A Frame2 Formra rakott 2db CheckBox (CheckBox3 és CheckBox4) egymástól független működtetési lehetősége adja azt, hogy programjaikkal, a Frame1 és Frame2 láthatóságát váltogatását oldják meg programjaikkal.

```
Private Sub CheckBox3_Click()          'Frame1 Visible
    If CheckBox3.Value = True Then
        UserForm1.Frame1.Visible = True
    Else
        UserForm1.Frame1.Visible = False
    End If
End Sub
```

```
Private Sub CheckBox4_Click()          'Frame2 Visible
    If CheckBox4.Value = True Then
        UserForm1.Frame2.Visible = True
    Else
        UserForm1.Frame2.Visible = False
    End If
End Sub
```

b) TesztForm1 bővítése Feladat 3-ra, a Vezérlők megjelenésének Alapra, Megjelenési formájára állításával folytatódik.

Az induló megjelenéshez, a Framek-nek látszani kell, és a Választók és a CheckBox üres mezőt kell, hogy mutasson.

Fontos tudnivaló: a UserForm-ra hívott Hide metódus hatására, csak a Form képe tűnik el a képernyőről úgy, hogy a Vezérlők – Caption és Text – tulajdonságai megmaradnak, nem módosulnak. A Form és Vezérlői viszont aktívak maradnak, programból elérhetők, tulajdonságaik módosíthatók, fogadhatnak adatot, csak a képernyőn nem láthatók. Mivel nincsenek a képernyőn, egér-klikkel nem mozgathatók.

A UserForm-ra hívott Show metódus hatására a Form csak megjelenik a képernyőn abban az állapotban, ahogy a memóriában létezett. Ha viszont a képernyőn egyszerre több Form is látható,

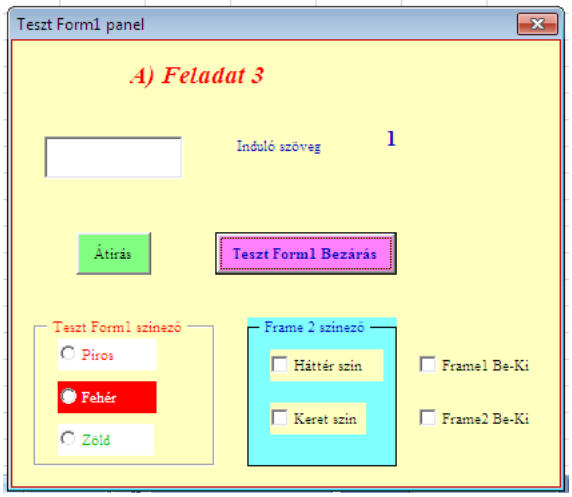
akkor is mind aktív úgy, mint ahogy a memóriában voltak, de egér-klikket csak az éppen Fókuszban levő form tud fogadni.

Ezzel már lehet a próbákat kezdeni.

A TesztForm3 alapba állítását el lehet intézni, - használat után a Bezár nyomógombbal, vagy - használat előtt az Excel munkalap Megjelenítés nyomógombbal. Ügyelni kell arra, ha a Felhasználó az Windows piros ×-szel lép ki, akkor az, bajt ne okozzon.

Az Excel Megjelenítés nyomógomb Subrutinja ezt így oldja meg:

```
Private Sub CommandButton1_Click()          'UserForm1 megjelenítése a képernyőn
    UserForm1.Label1.Caption = „A) Feladat 3”
    UserForm1.Label2.Caption = „Induló szöveg”
    UserForm1.TextBox1.Text = „„
    UserForm1.Label3.Caption = 1
    UserForm1.BackColor = &HC0FFFF          'TesztForm3-ra Bővités
    UserForm1.Frame1.Visible = False
    UserForm1.Frame2.Visible = False
    UserForm1.OptionButton1.Value = False
    UserForm1.OptionButton2.Value = False
    UserForm1.OptionButton3.Value = False
    UserForm1.CheckBox1.Value = False
    UserForm1.CheckBox2.Value = False
    UserForm1.CheckBox3.Value = False
    UserForm1.CheckBox4.Value = False
    UserForm1.Show
End Sub
```



A TesztForm3 Bezár gomb Subrutin kiegészítése.

```
Private Sub CommandButton1_Click()          'TesztForm1 Bezár
    UserForm1.TextBox1.Text = „„            'TesztForm3-ra Bővítés
    UserForm1.Frame1.Visible = False
    UserForm1.Frame2.Visible = False
    UserForm1.OptionButton1.Value = False
    UserForm1.OptionButton2.Value = False
    UserForm1.OptionButton3.Value = False
    UserForm1.CheckBox1.Value = False
    UserForm1.CheckBox2.Value = False
    UserForm1.CheckBox3.Value = False
    UserForm1.CheckBox4.Value = False
UserForm1.Hide
End Sub
```

Ezzel is lehet a próbákat kezdeni. Ha innen is működik minden, akkor kész a Feladat 3.

4.2.7.4 Bővítés: ListBox, ComboBox vezérlőkkel

Az eddig elkészült TesztForm1-re telepített Vezérlők, de maga a Form is, a további bővítés alapja. Két újabb Vezérlő felhelyezése és bemutatása ennek a pontnak a tartalma.

7) ListBox (lista) vezérlés

A vezérlés által létrehozható lista-mezők felhasználói interfész célra szolgáló objektumok, amelyek lehetővé teszik szövegek listaszerű megjelenítését a mező – fix méretű, de szükség esetén automatikusan görgethetővé váló – ablakában, és módot nyújtanak a lista egy vagy több elemének kiválasztására.

A lista-objektum szokásos módon történő létrehozása után a lista szövegekkel való feltöltése vagy még a program fejlesztése közben, a mező **List** tulajdonsága által, vagy a program futása közben, a program aktuális igényei szerint végezhető:

8) ComboBox (kombinált lista) vezérlés

A ComboBox vezérlés egy **egysoros** TextBox és egy **standard** működési módú ListBox kombinációja abból a célból, hogy általa akár lista alapján történő választás, akár más, a listában nem szereplő szöveges információ bevitele is elvégezhető legyen. Az objektum megjelenítését és működési módját a Style nevű, csak a fejlesztőkörnyezetben állítható tulajdonsága az alábbiak szerint határozza meg:

G) Feladat 4 programleírás

A Feladat 3 futtatása után, bővíteni kell a feladatot a Feladat 4 leírással. Ebben a feladatban ListBox (választható lista) vezérlő, valamint az ComboBox (kombinált választható lista) vezérlő használatával egészül ki a feladat. A Teszt.xls munkafüzetbe létrehoztuk a Földrajz munkalapot, melyen hazánk megyéi és megyeszékhelyei, további adatai vannak soronként feltüntetve. A Feladat 4 kiírásban készüljön egy olyan VBA program, mely két listából való választás lehetőségét teszi lehetővé, és választott adat alapján a teljes adatsort írja ki. A két lista legyen a megyék nevei és a székhelyek nevei, s akár Megye, akár Székhely listából választva a Formon jelenjen meg a megyesor főbb adatai. Legyen egy Választás Kezdete gomb, mely több feladatot lát el.

Feliratával felszólít a megye vagy székhely választására, és a választás után a Form fejlécei mellé kiíródnak a főbb adatok, és nyomógomb felirata is átáll a Választás Kész feliratra, színnel is jelezi a feliratváltozást.

A gombra klikkelve, a választott megye teljes adatsorát, a Földrajz munkalapról átmásolja a Munka1 munkalapra, az ugyancsak átmásolt fejléc sor alá. A másolás után törli a Formra irt adatokat, és további választásra ad lehetőséget. A gomb felirata, Választás újra és a szín is változik.

Az ismételt választás eredményét a választás hatására újra a Választás Kész felirat lesz a gombon. Az újabb klikk, ismét átmásolja a Munka1 munkalapra, a választott megye adatait, de a már az addig kiirt sorok alá.

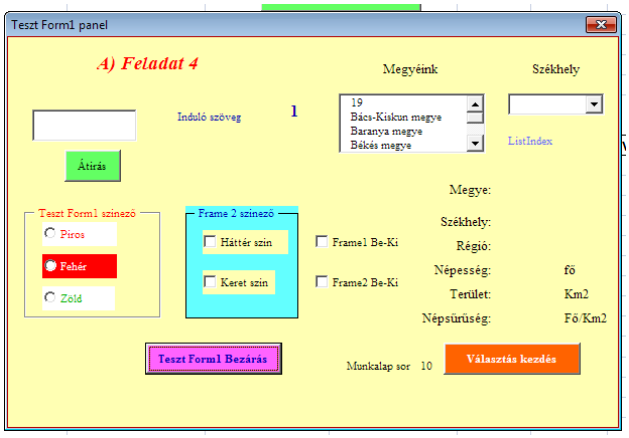
Feladat 4 indításakor az Excel Munka1 munkalap Megjelenítés indítása gomb Subrutinjába gondoskodni kell a UserFom1, ListBox, ComboBox listafeltöltéséről, valamint a Munka1 munkalap előtörléséről, az előzőleg már beirt adatsorok kitörléséről.

H) Feladat 4 megoldása

A meglevő Feladat 3-t megjelenítő TesztForm1-re kell a Feladat 4 kiírását megvalósítani. Először a TesztForm1-et meg kell nagyobbítani. A jobb alsó sarkát megfogva meg lehet nagyobbítani. Az addig rajta levő vezérlők helyzete nem változik, mert azok a bal felső saroktól pozícionáltak.

A munka:

- első fázisában fel kell helyezni – ToolBox-ból a ListBox-ot és a ComboBox-ot;
- ezután sorban fel kell helyezni a vezérlő neveket és a lista neveket hordozó Label vezérlőket;
- majd fel kell rakni a ListIndex felirat és az ennek értékét megjelenítő Label vezérlőket;
- majd fel kell rakni a Munkalap sor felirat és az ennek értékét megjelenítő Label vezérlőket;
- majd a választás eredményeit megjelenítő Label vezérlőket;
- és végül el kell helyezni, a választást érvényesítő CommandButton nyomógombot.



Ezután kezdődhet a programozás. A Subrutin írásokkal együtt, az éppen programozás alatt álló Vezérlők (*Control*) tulajdonságainak a beállításait is el kell végezni úgy, hogy a látható kép szerint nézzen ki.

A programozást, az Excel munkalapon levő Megjelenítő nyomógomb programbővítésével kell kezdeni, mert azokat a vezérlőket is, melyek most kerültek fel, az „Alap Induló” állapotba kell létrehozni. A gomb programja a VBA szerkesztő ablak Project ablak, Munka1 sorára klikkelve áll elő a legkönnyebben. Már látható, hogy milyen sok előkészítés van benne. Ezt kell bővíteni.

Jó tanács: az újabb program részeket mindig komment magyarázza.

Először azokat a deklarációkat kell megadni, ehhez az új vezérlők működtetéseihez szükségesek.

```
'Lista dinamikus tömb deklaráció
 Dim m() As String       ' Megyei lista memória tömbje
Dim sz() As String       ' Székhelyek lista memória tömbje
'Keresésnél - Kiírásnál használt változók
Dim Tag As String
```

```
Dim i As Integer
Dim u As Integer
'ListBox és ComboBox feltöltése
'TesztForm4-re Bővítés
'Beolvasás a Földrajz Excel munkalapról Do ...Loop ciklussal, dinamikus memóriabővítéssel
i = 0
Do
    i = i + 1
    Tag = Worksheets(„Földrajz”).Cells(i + 1, 1)
    ReDim Preserve m(i)
        m(i) = Tag
    Tag = Worksheets(„Földrajz”).Cells(i + 1, 2)
    ReDim Preserve sz(i)
        sz(i) = Tag
Loop Until Tag = „„
    m(0) = i - 1
    UserForm1.ListBox1.List = m()
    UserForm1.ComboBox1.List = sz()          ’TesztForm4-re alapra törlés
    UserForm1.Label12.Visible = False
    UserForm1.Label13.Visible = False
    UserForm1.Label14.Caption = „„
    UserForm1.Label15.Caption = „„
    UserForm1.Label16.Caption = „„
    UserForm1.Label17.Caption = „„
    UserForm1.Label18.Caption = „„
    UserForm1.Label19.Caption = „„
    i = 10                                   'Munkalap Kiírások törlése
Do
    Worksheets(„Munka1”).Rows(i).Select
    Selection.ClearContents
    Range(„A1”).Select
    If Worksheets(„Munka1”).Cells(i + 1, 2) = „„ Then
        t = 1
    Else
        i = i + 1
    End If
Loop Until t = 1 Or i > 20               'Munkalap első kiírási sorának megadása
     UserForm1.Label26.Caption = 10
    UserForm1.CommandButton3.Caption = „Választás kezdés”
    UserForm1.CommandButton3.BackColor = &H80FF&
    UserForm1.CommandButton3.ForeColor = &HFFFFFF
    UserForm1.Show
```

Az első feladat, hogy a Földrajz munkalapról a megyék neveit a ListBox1 listájába, a székhelyek neveit a ComboBox1 listájába kell betölteni.

A Lista feltöltésének egy jól bevált módja van itt bemutatva, a Do …. Loop ciklusban, dinamikus tömb használattal. Ilyenkor nem kell törődni a beolvasandó Adatmennyiséggel. A ciklus elején a

ciklusszámláló és az aktuális adatbeolvasás van a Tag változóba. Minden sorozatos tevékenység (adatbeolvasás, matematikai sorfüggvény számolás) esetén célszerű az Aktuális tagot egy Tag nevű változóba beültetni. Ezután sokkal egyszerűbb az aktuális Tag programszerinti használata.

Itt az m() és sz() String típusú tömbváltozóba gyűjtött névsorokat, lehet átadni egyszerre a ListBox1, illetve a ComboBox1 vezérlők List tulajdonságának. Az m() és sz() üres zárójel párja, a tömbben levő valamennyi memória alatt tárolt adatot átadja.

A Munka1 nevű Excel munkalapnak az előző kiírásokat törlő rutinja, szintén a Do … Loop-os ciklust használja.

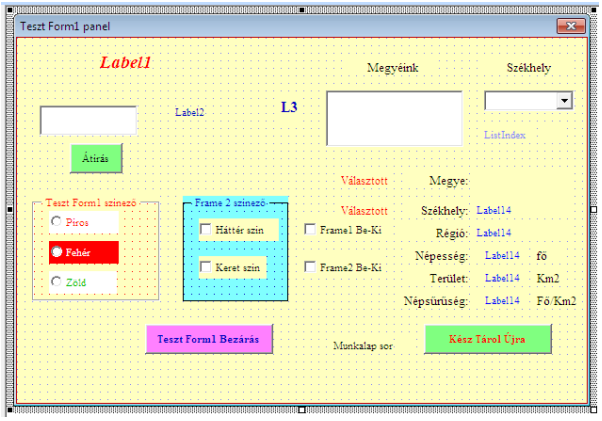
A Loop … Until metódus után a túlfutás megakadályozására, célszerű Or kapcsolattal kettős függvényt írni. A t = 1 a törlés utáni Üres sor esetén áll elő, ilyenkor az Until metódus tovább engedi a futást, a következő parancssorra.

Az Or kapcsolat után irt i > 20 függvény, a sorok vizsgálatának számát korlátozza – most 20-ra,.

Az üresre törölt munkalap sorok után, a Label26 vezérlő Caption tulajdonságába be lett írva a kiírás kezdő sor sorszáma. A következő 3 sor, a választást vezérlő nyomógomb induló beállítása.

Az Excel Munka1 munkalap indító nyomógombja Subrutinjának bővítése után, a UserForm1 bővítésével felkerült újabb vezérlők Subrutinját kell megírni. A ListBox1-ben a Megyenevek, a ComboBox1-ben a Székhelynevek fognak megjelenni. Ezekből klikkeléssel lehet választani, és mindkettőnek a feladata, jelezze a Label12 és Label13 Visible állításával, a választott adat indítást.

A Földrajz munkalapról olvassa ki a választott sor, itt megjelenített információit.



Állítsa be a CommandButton3 választást irányító nyomógomb megjelenés tulajdonságait

```
Private Sub ListBox1_Click()
'ListBox választás
    Dim ind As Integer
    UserForm1.Label12.Visible = True
    UserForm1.Label13.Visible = False
    ind = ListBox1.ListIndex
    UserForm1.Label24.Caption = ind
    UserForm1.Label14.Caption = ListBox1.List(ind)
    UserForm1.Label15.Caption = ComboBox1.List(ind)
    UserForm1.Label16.Caption = Worksheets(„Földrajz”).Cells(ind + 1, 3)
    UserForm1.Label17.Caption = Worksheets(„Földrajz”).Cells(ind + 1, 7)
    UserForm1.Label18.Caption = Worksheets(„Földrajz”).Cells(ind + 1, 8)
    UserForm1.Label19.Caption = Worksheets(„Földrajz”).Cells(ind + 1, 9)
    UserForm1.CommandButton3.Caption = „Választás Kész”
    UserForm1.CommandButton3.BackColor = &HFF00&
```

```
    'UserForm1.CommandButton3.BackColor = &H80FF&
    UserForm1.CommandButton3.ForeColor = &HFFFFFF
End Sub
```

A ComboBox1 Subrutinja egészen hasonló, csak Label12 és Label13 Visible tulajdonságait fordítva kell beállítani.

```
    UserForm1.Label12.Visible = False
    UserForm1.Label13.Visible = True
```

A CommandButton3 Subrutinja megírása következik. A gomb feladata, hogy munkalapra másolja a választott megyesort, törölje a UserForm azon vezérlőinek kiírást, amelyből kimásolta az adatokat, a nyomógomb feliratával és színével kérjen új választást.

A munkalapra való kiírás aktuális sorszámát a Label26.Caption tulajdonága tartalmazza. A deklarációk utáni parancssorban a kiírás sorszám van beolvasva

```
ksor = UserForm1.Label26.Caption
```

A kiíratások után, a sorszámláló léptetése történik:

```
UserForm1.Label26.Caption = ksor + 1
```

A szubrutin pedig:

```
Sub CommandButton3_Click()              'Kész Tárol Újra nyomógomb
    Dim sor As Integer
    Dim ksor As Integer
        ksor = UserForm1.Label26.Caption
    Worksheets(„Munka1”).Cells(ksor, 2) = UserForm1.Label14.Caption
    Worksheets(„Munka1”).Cells(ksor, 3) = UserForm1.Label15.Caption
    Worksheets(„Munka1”).Cells(ksor, 4) = UserForm1.Label16.Caption
    Worksheets(„Munka1”).Cells(ksor, 8) = UserForm1.Label17.Caption
    Worksheets(„Munka1”).Cells(ksor, 9) = UserForm1.Label18.Caption
    Worksheets(„Munka1”).Cells(ksor, 10) = UserForm1.Label19.Caption
        sor = UserForm1.Label24.Caption + 1
    Worksheets(„Munka1”).Cells(ksor, 5) = Worksheets(„Földrajz”).Cells(sor, 4)
    Worksheets(„Munka1”).Cells(ksor, 6) = Worksheets(„Földrajz”).Cells(sor, 5)
    Worksheets(„Munka1”).Cells(ksor, 7) = Worksheets(„Földrajz”).Cells(sor, 6)
    UserForm1.Label26.Caption = ksor + 1
    UserForm1.CommandButton3.Caption = „Választás újra”
    UserForm1.CommandButton3.BackColor = &H80FF&
    UserForm1.CommandButton3.ForeColor = &HFF&
End Sub
```

A fenti vezérlők programjainak megírása után, lehet az Excel Munka1 munkalapról indítani a Feladat 4-re bővített feladatot futtatni.

4.2.7.5 Bővítés: MultiPage, TabStrip, ScrollBar, ToggleButton, SpinButton, Image vezérlőkkel

Az eddig elkészült TesztForm3-ra telepített eddigi Vezérlők bemutatása után, ezen újabb Vezérlők bemutatását, a TeszForm4 mellé kinyíló UserForm2 form alkalmazásával lesz bemutatva.

A UserForm2 Formon való bemutatást, a Feladat 5 megvalósításával, a további hat Vezérlő bemutatása megtörténik, és ezzel vége a VBA Controlok ismertetésének.

9) MultiPage (lapozható vezérlő hordozó) vezérlés

Az ablak azonos területen, több lapot biztosító vezérlő objektum. Ezek a lapok, lapozó füllel rendelkeznek, melyekkel az objektum lapjai egyenként megjeleníthetők. A lapok a Formokhoz hasonlóan hordozhatják a Vezérlőket, igaz a Formoknál kevesebb tulajdonsággal rendelkeznek.

10) TabScrip (egyszerűbb lapozó) vezérlés

A TabScrip ablakok is azonos területen, több lapot biztosító vezérlő objektum. Ez a TabStrip vezérlő hasonló a MultiPage vezérlőhöz, a lapok, lapozó füllel rendelkeznek, de a TabStrip vezérlő kevésbé flexibilis, mint a MultiPage. A TabStrip nem képes tárolni az egyes Vezérlők (*Controlok*) csoportjait. A Vezérlő csoportok láthatóságát, használatát a Visible tulajdonságok kihasználásával, Elrejt / Megmutat stratégiával lehet érni. (használata nehézkes, nem kedvelt vezérlő)

11) ScrollBar (gördítő sáv) vezérlés

Már több vezérlőben lehetett találkozni (TextBox, ListBox, ComboBox) működő Scrollbar vezérléssel, de a VBA alkalmazásban önállóan is használni lehet. A SrollBar-ok egy előre definiálható skálán képes mozgatni a csuszkát, a csuszka állását a Value tulajdonsággal tudja közölni, és más Vezérlőkhöz (Controlokhoz) hasonlóan, függetlenül működnek másoktól, és saját esemény, tulajdonság, metóduskészlettel rendelkeznek.

12) ToggleButton (váltogató nyomógomb) vezérlés

A ToggleButton nyomógomb, a CommandButton nyomógombnak egy változata. Arettáló tulajdonságot mutat: Bennmarad/Kiugrik. Állapotát a Value értékben adja meg. Bent állás Value = 1. Kint állás Value= 0 értéket adja. Ez használható a további programfutások során.

13) SpinButton (léptető nyomógomb-pár) vezérlés

A SpinButton nyomógomb, egy csuszka nélküli Scrollbar-hoz hasonlít, mert előre meghatározható tartományban (min–max tulajdonság) között, Fel / Le léptetést képes végezni. A lépésszám állását, a Value tulajdonságban közli a programozás számára.

14) Image (ábra – kép) vezérlés

Az Image vezérlő a *.bmp, *.ico, *.jpg, *.gif típusú képfájlok megjelenítését végzi, és emellett érzékelni tudja az egérrel való klikk eseményt is. Kép gyors-megjelenítését lehetővé tevő objektum-típusok Visual Basic-ben levő Picture kép-vezérlésnél lényegesen kevesebbet tud, de működtetése kevesebb rendszererőforrást is igényel. Ez különösen az olyan alkalmazások készítésénél lehet előnyös, amikor várhatóan sokszor igénylik képek újra-megjelenítését. A képet – még a nem-átméretezhető képfajták esetén is – a Picture.* tulajdonságai állításával, a mező méreteihez torzítja; úgyhogy a mező-méretek módosítását a kép arányainak megfelelően végezve ezzel a kép nagyítása vagy kicsinyítése érhető el

I) Feladat 5 programleírás

A Feladat 4 futtatása után, bővíteni kell a feladatot a Feladat 5 leírással, melyben, a még létező hat VBA vezérlő lesz bemutatva. A UserForm1 eléggé megtelt az eddigi feladattal, ezért most a Feladat 5 kiírásait, egy újabb User Formon kell bemutatni, a UserForm2 megtervezésével.

A kiírás első kérése legyen mindjárt az, hogy ezen új Form megjelenési helye és mérete igazodjon, a UserForm1 megjelenéséhez. A UserForm2 a UserForm1 mellett jobb oldalon jelenjen meg, és ugyanolyan legyen a Forma magassága.

- a UserForm2 megjelenését, a UserForm1-re felrakott UserForm2 indítása nyomógomb indítsa.
- a UserForm2-re fel kell tenni, egy 3 lapos MultiPage-t vezérlőt, és ennek a lapjain kell bemutatni a további 5 vezérlőt.
- az első lapon egy 3 lapos TabStrip és a ScrollBar vezérlő alkalmazására legyen a példa.
- a 3 lapos TabStrp1 vezérlőn jelenítsen meg 3 db méretű Frame-t, a kereteken belül 3-3 OptionButton vezérlése mellett piros –fekete – zöld színezést mutasson be.
- a ScrollBar vezérlő működését egy Label vezérlő Label.Top tulajdonságának programból való állításával mutassa be. A Label vezérlőnek a ScrollBar melletti Fel/Le mozgatását, a ScrollBar léptetése vezérelje.
- a második lapon ToggleButton és a SpinButton nyomógombok működését mutassa be.
- a ToggleButton működését piros – zöld színezéssel mutassa be.
- a SpinButton léptetető vezérlő működését, a Scrollbar-hoz hasonlóan, egy Label vezérlő mozgatásával mutassa be. A SpinButton léptetését használva, a Label.Top és Label.Height tulajdonság együttes vezérlésével, egy lépésközzel a Top csökkenjen, a Height nőjön. Így a látvány az, hogy Label talppontjáról a léptetés Fel/Le hatására Nő/Csökken.
- a harmadik lapon, az Image vezérlő bemutatása legyen. Az Image vezérlőn megjelenő képeket egy SpinButton léptető léptetése vezérelje.

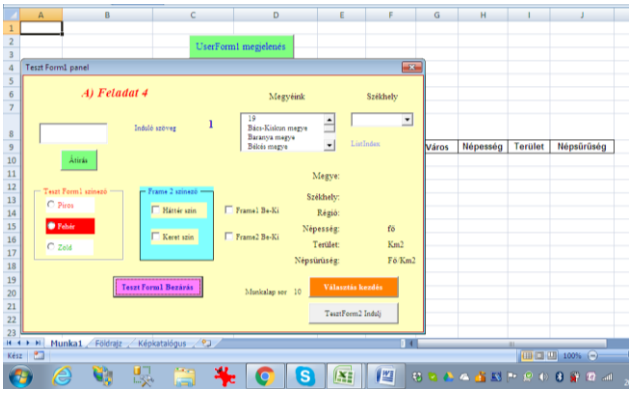
J) Feladat 5 megoldása

A meglevő Feladat 4-et megjelenítő TesztForm1 már eléggé tömött, ezért a Feladat 5 kiírását, egy újabb Formon, a UserForm2 felületén kell megvalósítani. A feladat megvalósítása, most is csak lépések sorozatával lehet megoldani.

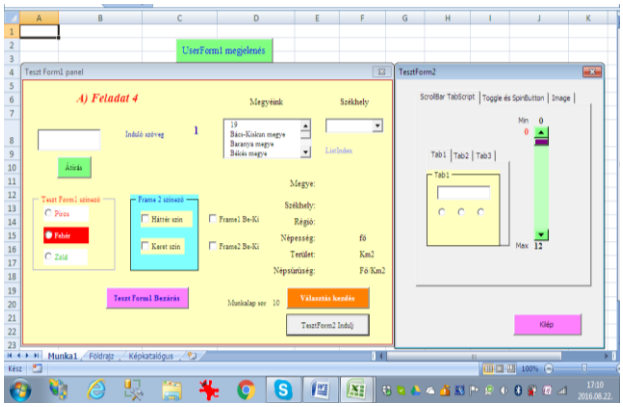
- a UserForm1-re fel kell rakni egy CommadButtont, mellyel a UserForm2-t lehet megjeleníteni.
- a UserForm2.Top és UserForm2.Height tulajdonságát, a UserForm1 form azonos tulajdonságaira kell állítani, míg a UserForm2.Left tulajdonságát pedig úgy kell megadni, hogy azzal a megjelenő UserForm2, pontosan a UserForm1 mellett jelenjen meg. A UserForm2.Width (szélesség) tulajdonsága annyi legyen, hogy, elférjen rajta a 3 lapos MultiPage vezérlő.
- a Formra fel kell tenni, egy Kilép nyomógombot.

A UserForm2-t indító nyomógomb Subrutinja:

```
Private Sub CommandButton4_Click()
'UserForm2 indítása
    UserForm2.Frame4.Visible = True
    UserForm2.Frame2.Visible = False
    UserForm2.Frame3.Visible = False
    UserForm2.TabStrip2.Value = 0
    UserForm2.SpinButton1.Value = 1
UserForm2.Show
End Sub
```



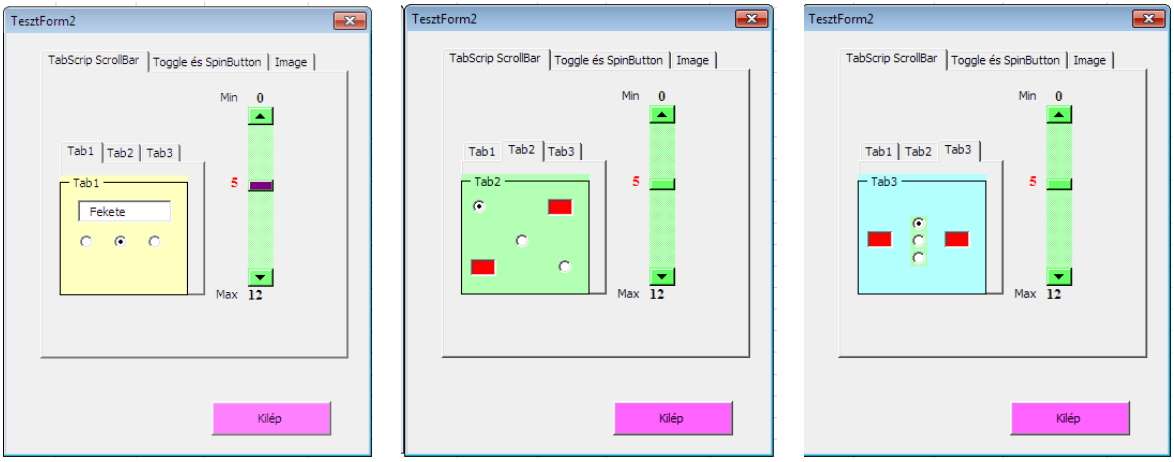
Az indító gomb megnyomása után megjelenik a UserForm1 mellett a UserForm2.



A Formon a 3 lapos MultiPage1 vezérlő, első lapján a TabStrip2 és a ScrollBar1 van bemutatva.

```
Private Sub TabStrip2_Change()
'TabStrip2 vezérlése, a Value értéke alapján
    If UserForm2.TabStrip2.Value = 0 Then
        UserForm2.Frame4.Visible = True
        UserForm2.Frame2.Visible = False
        UserForm2.Frame3.Visible = False
    ElseIf UserForm2.TabStrip2.Value = 1 Then
        UserForm2.Frame4.Visible = False
        UserForm2.Frame2.Visible = True
        UserForm2.Frame3.Visible = False
    ElseIf UserForm2.TabStrip2.Value = 2 Then
        UserForm2.Frame4.Visible = False
        UserForm2.Frame2.Visible = False
        UserForm2.Frame3.Visible = True
    End If
End Sub
```

A TabScrip2 – mivel nem képes a vezérlőket hordozni a lapjain külön-külön – ezért a rajta levő Frame2; Frame3; Frame4 által hordozott vezérlők, a Framek Visible tulajdonságait a TabScrip.Value értékeivel vezérelten lehet egyenként megjeleníteni. Az első lapon a Frame4 lesz látható, az OptionButton1 – 2 – 3.




```
Private Sub OptionButton1_Click()              'Frame4 piros
    UserForm2.TextBox1.Text = „Piros”
End Sub
```

```
Private Sub OptionButton2_Click()              'Frame4 fekete
    UserForm2.TextBox1.Text = „Fekete”
End Sub
```

```
Private Sub OptionButton3_Click()              'Frame4 zöld
    UserForm2.TextBox1.Text = „Zöld”
End Sub
```

A TabScrip2 második lapján a Frame2 lesz látható, a benne levő OptionButton4 – 5 – 6 választó gomb.

A TabScrip2 harmadik lapján a Frame3 lesz látható, a benne levő OptionButton7 – 8 – 9 választó gomb.

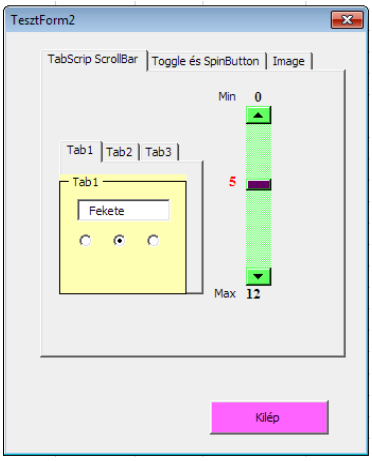
A programjaik egyformák, ezért itt csak OptionButton4 – 5 – 6 választó gomb Subrutinja látható!

```
Private Sub OptionButton4_Click()              'Frame2 Piros
    UserForm2.TextBox2.BackColor = &HFF&
    UserForm2.TextBox3.BackColor = &HFF&
End Sub
```

```
Private Sub OptionButton5_Click()              'Frame2 Fekete
    UserForm2.TextBox2.BackColor = &H0&
    UserForm2.TextBox3.BackColor = &H0&
End Sub
```

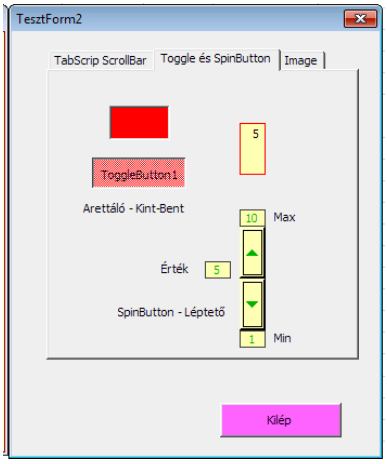
```
Private Sub OptionButton6_Click()              'Frame2 Zöld
    UserForm2.TextBox2.BackColor = &H8000&
    UserForm2.TextBox3.BackColor = &H8000&
End Sub
```

A MultiPage1 első lapján a TabScrip mellett a ScrollBar1 is be van mutatva. Léptetési tulajdonságával, a definiált min–max értékek között a ScrollBar1.Value értékei Fel/Le mozog. Ezt felhasználva, a Label3-Top tulajdonságát vezérelve, a Label szintén Fel/Le mozog, eközben a Value értéket jelzi ki.




```
Private Sub ScrollBar1_Change()          'ScrollBars1
    Dim h As Single
    Dim fe As Integer
    Dim jo As Integer
    Dim m As Single
    m = UserForm2.ScrollBar1.Height - 12
    h = m / UserForm2.ScrollBar1.Max
    UserForm2.Label3.Top = UserForm2.ScrollBar1.Top +_
       (UserForm2.ScrollBar1.Value * h)
    UserForm2.Label3.Caption = UserForm2.ScrollBar1.Value
End Sub
```

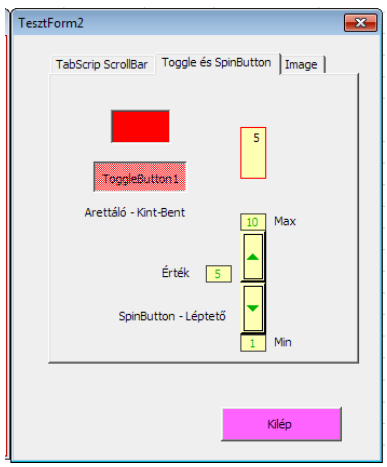
A Formon a 3 lapos MultiPage1 vezérlő második lapján a ToggleButton1 és a SpinButton1 van bemutatva.



A ToggleButton nyomógomb Arettáló nyomógombot utánoz. Bent/Kint állása van, mit Value tulajdonságában jelez. Kint állás: Value = 0, Bent állás: Value = 1. Ezt lehet vezérlő infónak használni.

```
Private Sub ToggleButton1_Click()        'Arettáló nyomógomb
    If ToggleButton1.Value = False Then
        UserForm2.TextBox6.BackColor = &HFF00&
        UserForm2.ToggleButton1.BackColor = &HFF00&
    Else
        UserForm2.TextBox6.BackColor = &HFF&
        UserForm2.ToggleButton1.BackColor = &HFF&
    End If
End Sub
```

A SpinButton1 léptető vezérlő, a definiált Min – Max értékek között Fel/Le léptet, ezeket az értéket a SpinButton1.Value tulajdonságban ad ki.

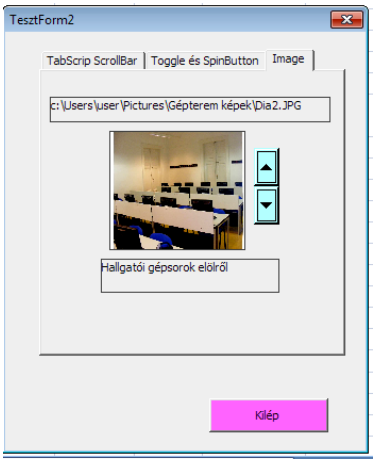



```
Private Sub SpinButton1_Change()         'Léptető gomb
    Dim f As Integer
    Dim m As Integer
        f = 56
        m = 16
    UserForm2.Label8.Caption = UserForm2.SpinButton1.Value
        f = f - (UserForm2.SpinButton1.Value * 4)
    UserForm2.TextBox7.Top = f
```

```
    UserForm2.TextBox7.Text = UserForm2.SpinButton1.Value
        m = m + (UserForm2.SpinButton1.Value * 4)
    UserForm2.TextBox7.Height = m
End Sub
```

Ezt használja a feladat megoldás, amikor ennek Value értéket használva, a TextBox7.Top kezdeti értékét lépésenként egységnyivel csökkenti – egyúttal a TextBox7.Height kezdeti értékét lépésenként egységnyivel növeli. Így áll elő az a látszat, hogy a talpponttól Nő/Csökken a magassága, miközben hordozza a SpinButton1.Value értékét.

A Formon a 3 lapos MultiPage1 vezérlő harmadik lapján az Image1 képvezérlő van bemutatva.




```
Private Sub SpinButton2_Change()
'Kép léptető nyomógomb
    Dim nev As String
    Dim s As Integer
        s = UserForm2.SpinButton2.Value
        nev = Worksheets(„Képkatalógus”).Cells(s, 1)
    UserForm2.Image1.Picture = LoadPicture(nev)
    UserForm2.Label14.Caption = nev
        UserForm2.Label15.Caption =
    Worksheets(„Képkatalógus”).Cells(s, 10)
End Sub
```

Az Image vezérlő, ismert formátumú képeket tud megjeleníteni. Az itt bemutatott rutinnal, a képek megjelenését válogatni lehet. A megjelenítendő kép **könyvtári helyét** kell egy String változóban kell argumentumként a LoadPicture() metódusnak megadni. Ezt **könyvtári utat** a Képkatalógus nevű Excel munkalapról soronként olvassa be a program, nev (String) változóba. A nev változó kerül a LoadPicture(nev) , majd ezzel alakul ki a képmegjelenítő parancssor.

```
    UserForm2.Image1.Picture = LoadPicture(nev)
```

Az Excel Képkatalógus sorait, a korábban bemutatott Application.GetOpenFilname hívással lehet megoldani. Például:

```
Sub feltoltes()
    Dim fajl as String
    Dim i as Integer
    Dim komment as String
    For i = 1 to 9
        fajl = Application.GetOpenFileme
        WorkSheets(„Képkatalógus”).Cells(i,1) = fajl
        komment = InputBox(„Kérem a kép kommentjét = ?”)
        WorkSheets(„Képkatalógus”).Cells(i,4) = komment
Next i
End Sub
```

Ez a rutin 9-szer megkéri, hogy jelöld ki a képet, majd az InputBox megkéri a kép kommentjét, és feltöltik a sorokat.

5 Utószó

Építsed a Formot (ez a gondolat azonos egy korábbi szakasszal)

E Jegyzetnek célja az egyéni tanulás segítése, a meglevő szakirodalmak „kijegyzetelése” annak érdekében, hogy a kezdő hallgató ez alapján már, eligazodhasson más kiadványokban. Tehát a Jegyzet nem helyettesíti az Irodalomjegyzék kiadványait, és nem is törekszik a Formok – Controlok, „Tételes – Katalógusszerű” ismertetésére, a tulajdonság és alkalmazás tekintetében.

Fontos:

A Fom – Control témakörben, az alkalmazás alapjait, az induló lépéseit, lehetőségeit, egy **Tanuló Form** építés lépéseivel lett bemutatva, a szükséges magyarázatokkal, szoftvertámogatással ás a működési bemutatókkal. A további gyakorlat megszerzése, már saját feladat megoldása, saját erőből lehetséges.

E saját munkához ad segítséget az is, hogy ebben a fejezetben bemutatott mintapélda és abból való programrészek mind, a Teszt.xlsm fájlból valók, ezért a tárgy honlapjára, a www.kit.bme.hu –ra a letölthető Jegyzet mellé, letölthetően felkerül ez az Excel fájl is.

További jó munkát!

a szerzők.