

MICROSOFT EXCEL VISUAL BASIC FOR APPLICATIONS

PROGRAMOZÁSI SEGÉDLET

Ez a kiadvány azoknak a szakembereknek készült, akiknek nincs informatikai vagy programozói végzettségük, de munkájuk során, az Excel alkalmazásakor gyakran olyan feladatokat kell elvégezniük, amelyeket a legismertebb függvényekkel nem, vagy csak jelentős időráfordítással lehet megoldani. Ennek megfelelően elsősorban az egyszerűsége és az érthetőségre törekedtem. A gyakorlatban gyakran előforduló feladatokhoz hasonló példákon keresztül mutatom be az egyes nyelvi elemeket, a szükséges – de nem teljes – mélységükig részletezve. Az ismeretanyagot 12 egymásra épülő leckére osztottam, melyek elsajátításával Ön képes lesz olyan programok írására, amelyek jelentős időmegtakarítást eredményeznek, és megkönnyítik a napi feladatok megoldását.

Excel Expert www.excelexpert.hu

JELÖLÉSEK

A tananyagban a jobb elkülöníthetőség érdekében az alábbi jelöléseket alkalmazzuk:

- Programkódok esetén írógépbetűkkel (Courier New 10 pt): For x = 1 to 100 Step 5
- Legördíthető menüpontok esetén (Ariel félkövér 10 pt): **Nézet/Eszköztárak**

A PROGRAMOZÁSRÓL

A program olyan utasítások sorozatából áll, amelyet a számítógép egymás után sorban végrehajt. A programozó feladata az, hogy a kapott feladatot „lefordítsa” a gép nyelvére; ez a kódolás folyamata. Lássunk egy egyszerű példát!

Kaptunk egy táblázatot, ami nincs teljes egészében kitöltve, vannak benne üres sorok is.

A feladat egyszerű: töröljük ki a táblázatból az üres sorokat!

- A folyamat lépései:
- Megnézzük az első sort, innen haladunk lefelé a táblázat végéig
 - Ha üres cellát találunk:
 - kijelöljük a sort,
 - töröljük a sort.
 - Továbblépünk a következő sorra, megnézzük, van-e üres cella.

Ugyanez egy program formájában a gép nyelvére lefordítva:

```
Private Sub urestorol() ' üres sorok törlése
sor = 5 ' az 5. sorban kezdődik a táblázat

Do Until Worksheets("Mintalap01").Cells(sor, 3) = "" ' folytasd a
táblázat végéig

    If Worksheets("Mintalap01").Cells(sor, 4) = "" Then ' ha üres
cellát találunk
        Rows(sor).Select ' kijelöljük a sort
        Selection.Delete Shift:=xlUp ' töröljük a sort

    End If

    sor = sor + 1 ' következő sor száma

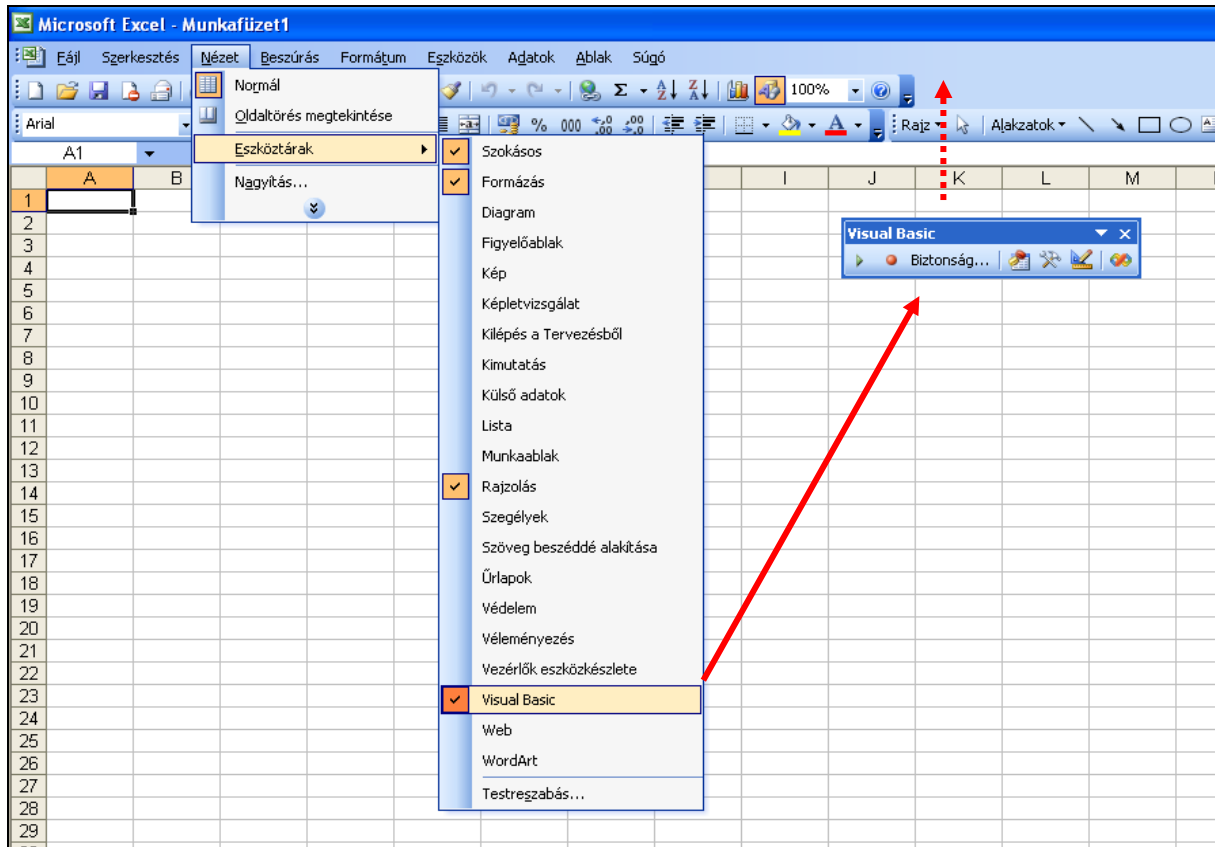
Loop ' továbblépünk a következő sorra

End Sub ' elfogytak a sorok, vége a programnak
```

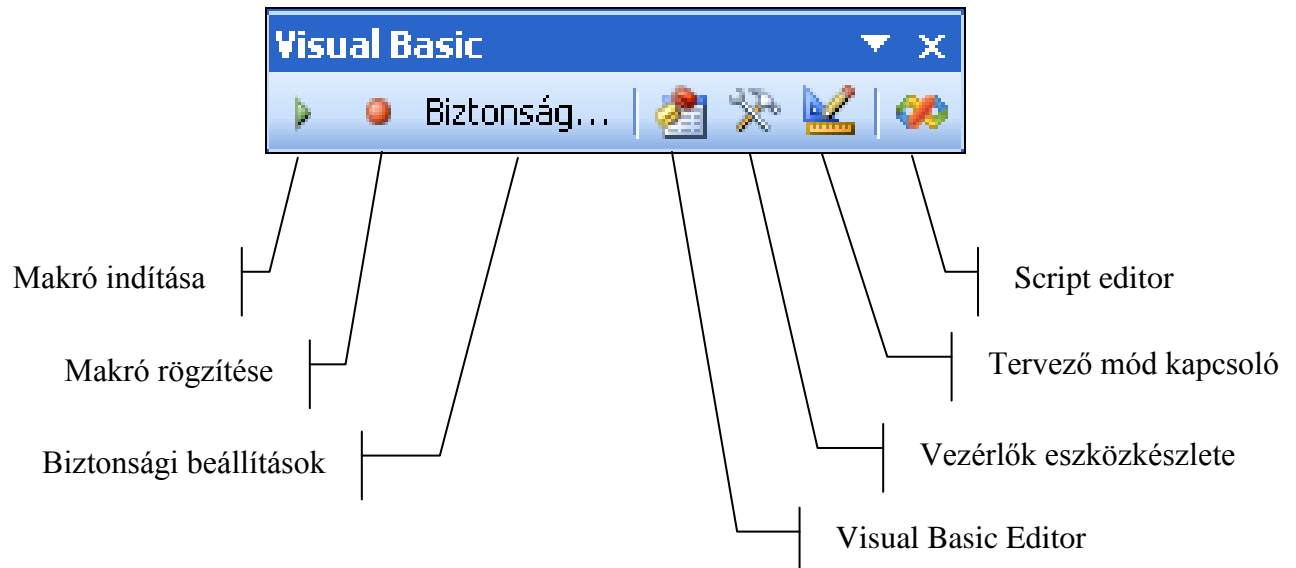
Ebben a kiadványban a Microsoft Excel alkalmazást fogjuk programozni VBA (Visual Basic for Applications) nyelvi környezetben. Ez a programozási nyelv nem a profi, hosszú éveig képzett programozók, hanem az általános ismeretekkel és képességekkel rendelkező felhasználók számára készült, így könnyen elsajátítható.

AZ EXCEL KÖRNYEZET BEÁLLÍTÁSA – OFFICE 2003-AS KÖRNYEZETBEN

A VBA szerkesztőt, amelyben dolgozni fogunk, illetve az elkészült a programokat láthatjuk, a **Nézet/Eszköztárak/Visual Basic** menüpontokon keresztül érhetjük el. Ez az eszköztárat a folyamatos használat miatt érdemes kihelyezni a többi közé, a szokásos eszköztárak mellé.



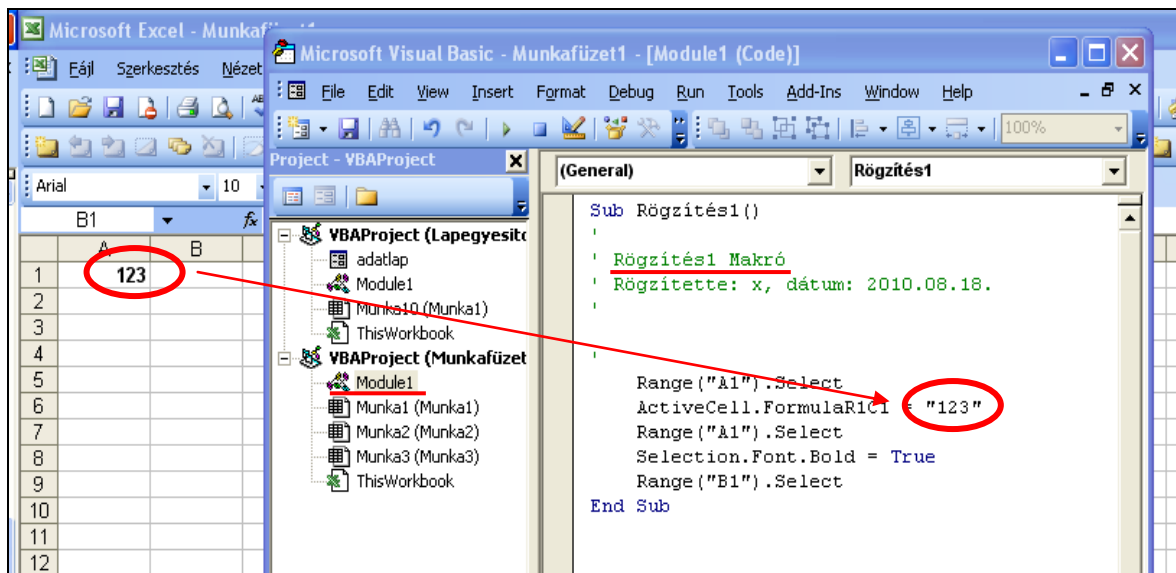
Az eszköztáron belül az alábbi gombokat találhatjuk:



Az egyes gombok jelentése:

- **Makró indítása:** ezzel a gombbal indíthatjuk, szerkeszthetjük, vagy törölhetjük a makró programokat. A munkafüzetek megnyitásakor ezzel a gombbal érhetjük el a minta-programokat is.
- **Makró rögzítése:** ha lenyomjuk a **Felvétel** gombot, akkor az Excel a gomb ismételten lenyomásáig minden tevékenységünket feljegyzi (mint egy diktafon), és programkód formájában rögzíti (vagyis kódolja).


Az első rögzített program `Rögzítés1()` néven érhető el egy modulban, ami a programok egyik tárolási egysége. Mint látható, a magyar Excel ékezetes betűket használ a program elnevezésében, mi azonban a nyelvi kompatibilitási hibák elkerülése érdekében lehetőleg ne használjunk ékezetes karaktereket. A modulokat ugyanúgy be lehet helyezni egy munkafüzetbe, mint a munkalapokat. Az alábbi kép középső részén látható, hogy a **Module1** alatt az egyes munkalapok is felsorolásra kerültek – mint a munkafüzet részei.

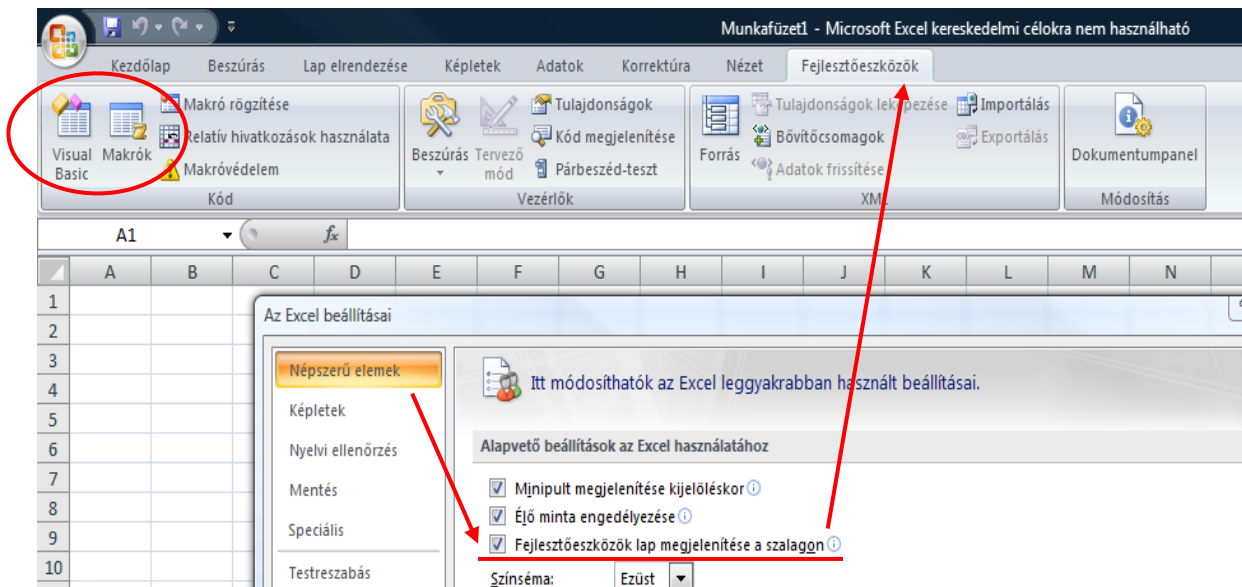


Érdeemes a rögzítést próbálgatni, mivel az így készült programokból sokat lehet tanulni, vagy akár kész kódrészleteket „kölcsonözni”.

- **Biztonsági beállítások:** itt határozhatjuk meg a makrókra vonatkozó biztonsági szintet.
- **Visual Basic Editor:** ezzel a gombbal lehet megnyitni azt az ablakot, ahol programjainkat szerkeszthetjük.
- **Vezérlők eszközkészlete:** megnyomásával egy újabb menühöz jutunk, amely segítségével ún. User Formra (felhasználói felületre) helyezhetünk el különböző gombokat, legördülő listákat, szövegmezőket. A haladó szintű CD tananyagában foglalkozunk vele.
- **Tervező mód kapcsoló:** bekapcsolásával tervező módba juthatunk, amelyben a munkalapra és a UserFormra elhelyezett vezérlőket állíthatjuk be. A haladó szintű CD tananyagában foglalkozunk vele.
- **Script Editor:** HTML kódszerkesztő, ebben az anyagunkban nem használjuk.

AZ EXCEL KÖRNYEZET BEÁLLÍTÁSA – OFFICE 2007-ES KÖRNYEZETBEN

1. Kattintson az **Office**  gombra , majd **Az Excel beállításai** gombra.
2. Kattintson a **Népszerű elemek** kategóriára, és jelölje be a **Fejlesztőeszközök lap megjelenítése a szalagon** jelölőnégyzetet.



ÁLLANDÓK, VÁLTOZÓK ÉS TÖMBÖK DEFINIÁLÁSA A PROGRAMOK ELEJÉN

- Állandók azok az adatok, amelyek a program futása közben nem változtatják értéküket.
- Változók azok az adatok, amelyek a folyamat során különböző értékeket kaphatnak.
- Tömbök azok az adathalmazok, amelyek egy név alatt több elemet is tartalmazhatnak. Az egyes elemekre sorszámukkal hivatkozhatunk.

Állandókat, változókat és tömböket az alábbi módon definiálhatunk:

Dim név As típus

A programok helyes működésének érdekében a programok legelején a programban használni kívánt állandók, változók és tömbök nevét, valamint a hozzájuk tartozó típusokat meg kell határozni. Ez azt jelenti, hogy előre meg kell mondanunk, hogy például a `nev` elnevezésű változóba szöveg típusú adatokat, neveket fogunk helyezni.

- A definiálásnak ezt a folyamatát ne hagyjuk el, mert segít a hibák előzetes kiküszöbölésében. Használatával elkerülhetjük például azt az alábbi hibát, amikor nem egyforma típusú adatokat szeretnénk összegezni:

$x = 15 + \text{„Alma”}$

Ezt az összeadást az Excel nem tudja elvégezni, mivel egy számhoz egy főnevet kellene hozzáadni, ami nem megoldható.

- Egy másik példa: a moziban a székek számozása egész számokkal történik (Integer típus – magyarázatát lásd néhány sorral alább). Zavarba jönnénk, ha mozijegyünk az 5,3-as székre szólna. Ha a jegy kiadásakor meg lett volna határozva, hogy az egyes székek sorszáma csak pozitív egész szám lehet, már a pénztárban kiderült volna a hiba.

A következő főbb típusokat szoktuk használni:

- `Integer`: egész számok -32 768 és 32 767 közötti értékben (Pl.: -155, 34, 4345)
- `String`: karakterlánc (szöveg) maximálisan 65 535 db karakter részére (póló, nadrág)
- `Boolean`: logikai, kétféle adat lehet benne: Igaz vagy Hamis (54 > 12? → Igaz.)
- `Date`: dátumok és időpontok tárolására (2010. 01. 01. 13:00)
- `Variant`: ez a típus bármilyen típusú adatot tartalmazhat

Példák:

- A fenti példában a mozsizékek helyes (egész számokkal történő) sorszámozásához egy „szeksorszam” nevű változó, amely a székek sorszámaikat tartalmazza:

```
Dim szeksorszama As Integer
```

- Egy „nev” elnevezésű változó, amelyben neveket (karakter-sorozatokat) fogunk tárolni:

```
Dim nev As String
```

- Egy „darabszamok” nevű tömb 10 db elemmel, melyek egész számok (az elnevezés utáni zárójelben az adott tömb elemszámát határozzuk meg):

```
Dim darabszamok(10) As Integer
```

Így képzelhetjük el a „darabszamok” nevű tömböt:

23	45	56	43	76	78	54	25	57	26
----	----	----	----	----	----	----	----	----	----

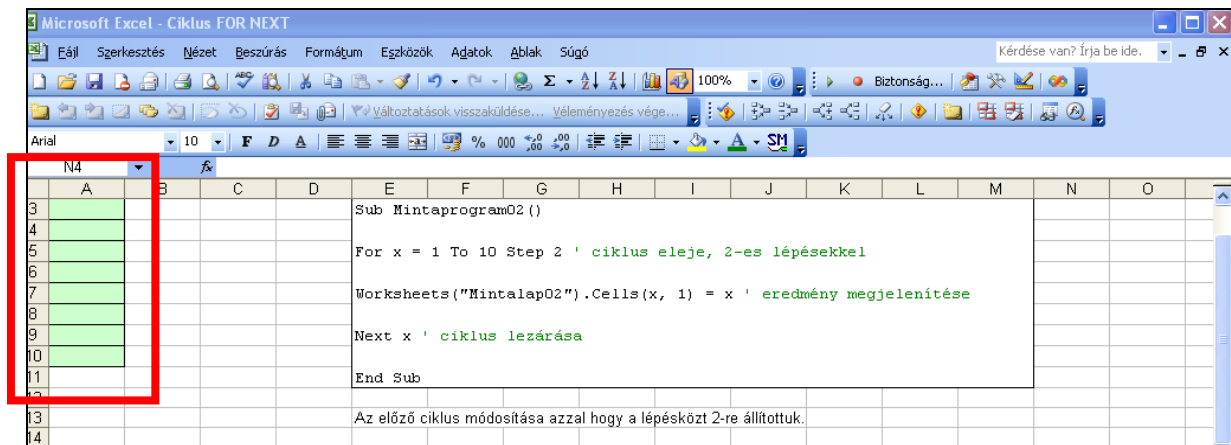
Adott elemre való hivatkozás módja (például a tömb második elemére): `darabszamok(2)`

Bizonyos nyelvi elemek már foglaltak; ha állandóink, változóink vagy tömbjeink nevét erre szeretnénk definiálni, hibajelzést fogunk kapni. Nem használható elnevezésként például a `For`, mert ez egy ciklus kezdetét jelöli. A név kiválasztásakor érdemes olyan elnevezést választani, ami utal a későbbi tartalomra. Például az aktuális sor számát eltárolhatjuk az `aktualissorszama` elnevezésű változóban, vagy a kezdősor számának állandó értékét megadhatjuk a `kezdosorszama = 5` formában.

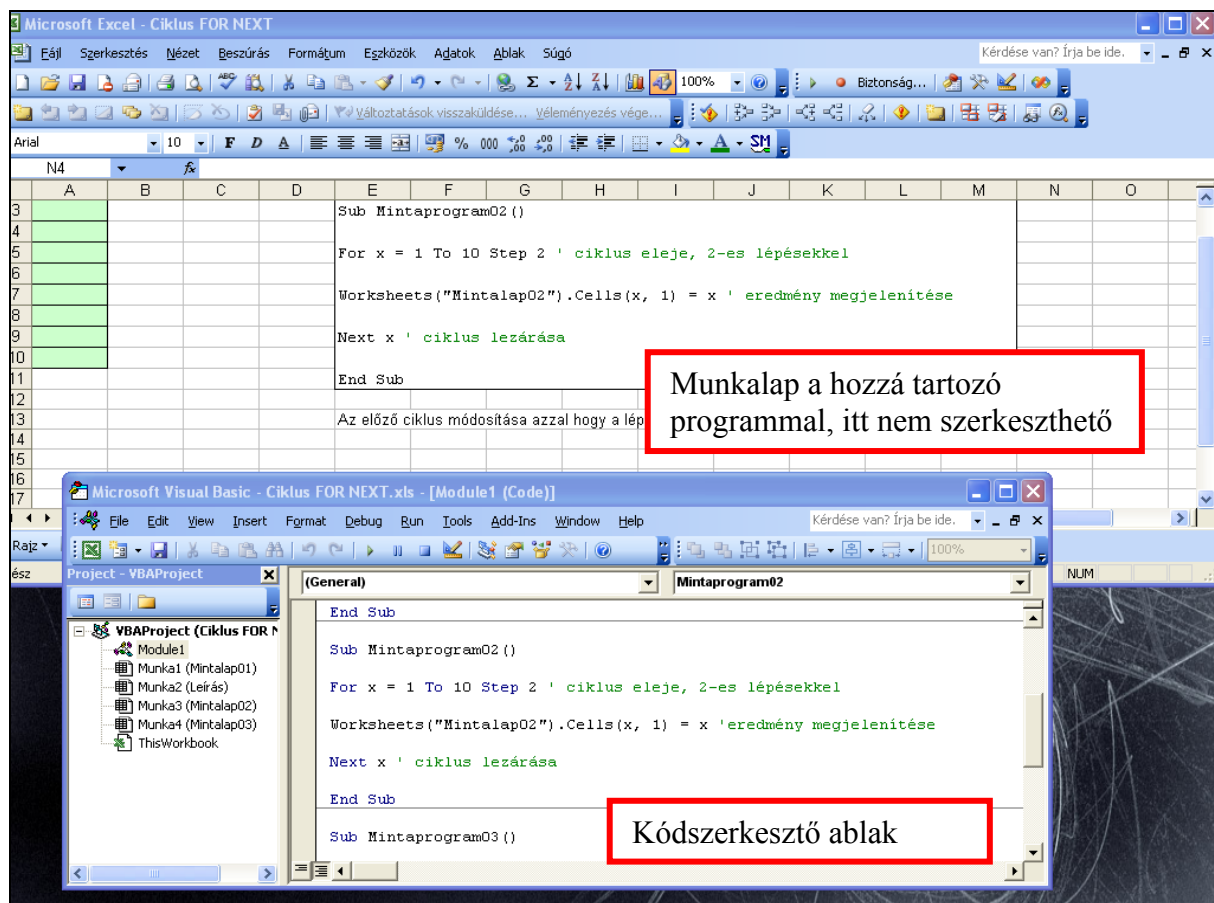
Az Excel a nem definiált változók használatát is megengedi, ez egyfelől rugalmasságot, más szempontból viszont hibalehetőséget jelent számunkra. Ha a program első sorába beírjuk az `Option Explicit` utasítást, akkor a program írása során csak a már egyszer definiált változóneveket fogadja el. Anyagunk példaprogramjaiban ez az utasítás nem szerepel, azért, hogy a próbálgatásnál és a változtatásoknál ne okozzon gondot. Ha viszont új programot írunk, akkor mindenképpen javasolt a használata.

ÉS MOST LÁSSUNK MUNKÁHOZ!

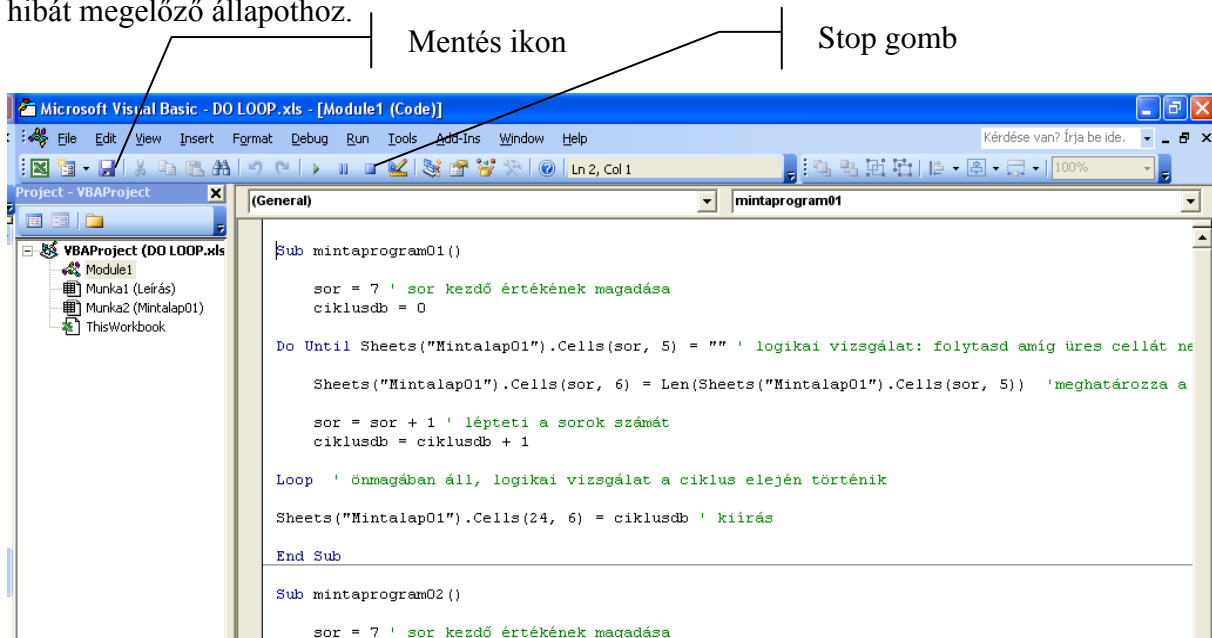
Nyissuk meg az Excel alkalmazást, azon belül a következő leckét tartalmazó fájlt. Minden munkafüzet első lapja a soron következő új tananyaghoz tartozó leírást tartalmazza. Ezt átnézve és megértve átléphetünk a mintalap(ok)ra, ahol lefuttathatjuk a mintaprogramokat, és láthatjuk is azok eredményét. Az egyes mintalapokon mindig az aktuálisan aktív mintalappal megegyező sorszámú mintaprogram működését láthatjuk. Amennyiben egy mintaprogramot nem a hozzá tartozó munkalapon futtatunk, ez a program futásában hibát okozhat, és hibaüzenetet kaphatunk. A témához tartozó program kódja az adott mintalapon külön keretben szerepel. A program minden kódsora egy megjegyzéssel végződik, amely magyarázatot ad az előtte lévő programsor feladatára, működésére vonatkozóan. A program futásának eredménye általában a zöld háttérű cellákban jelenik meg. (Az ábrát lásd a következő oldalon.)



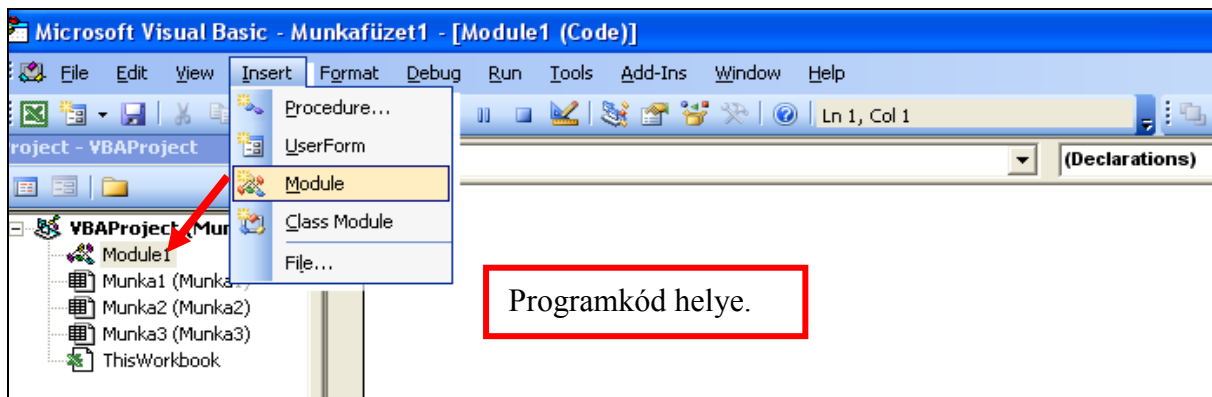
Ha módosítani szeretnénk a programon, akkor azt nem itt, hanem a programkód-szerkesztő ablakban tehetjük meg. A **Visual Basic Eszköztár/Visual Basic Editor** gombjának megnyomásával nyissuk meg a programkód-szerkesztő ablakot. Itt láthatjuk a mintaprogramokat a Modul1-ben. Ebben az ablakban úgy szerkeszthetjük, másolhatjuk vagy javíthatjuk a kódsorokat, mint egy egyszerű szövegszerkesztőben. Kísérletezzünk bátran a kódsorokkal, ne aggódjunk, hogy gyakorlásunk közben visszavonhatatlan hibát okozunk – a programkód eredeti verziója ugyanis a munkalapon megmarad, innen soronként bármikor visszamásolható.



Fontos tudni, hogy ha a gyakorlás részeként a program kiindulási paramétereit megváltoztatjuk, és a programot lefuttatjuk, annak eredménye nem visszavonható. Például, ha egy táblázat ötödik sora helyett a táblázat – a program futása szempontjából meghatározóan fontos adatokat tartalmazó – első sorát töröljük, ez a művelet a program lefuttatását követően többé már nem visszavonható! Az ehhez hasonló hibák elkerülése érdekében célszerű a munkafüzetet a munka során rendszeresen menteni, így ha valamilyen hiba történik, akkor a munkafüzet mentés nélküli bezárásával, majd a legutóbb mentett változat megnyitásával visszatérhetünk a hibát megelőző állapothoz.



Ha teljesen új programot szeretnénk írni, akkor a felső menüsorban található **Insert/Module** paranccsal helyezünk be egy új programmodult.



Ezt követően a jobb oldali ablakban máris kezdetjük programunk írását az `Option Explicit`, majd a `Sub programnév()` utasításokkal. Az `End Sub` automatikusan a helyére kerül. E sorok közé lehet írni a további kódokat.

```
Option Explicit

Sub elsoprogramom()

End Sub
```

HASZNOS TUDNIVALÓK

A programsorokat a jobb áttekinthetőség érdekében érdemes egymástól eltolni, köztük üres sorokat hagyni, és az összetartozó kódokat egymás alá, azonos behúzással elhelyezni (lásd az alábbi ábrán kék szaggatott vonallal és nyíllal jelölve).

A sorok végére megjegyzések szúrhatók be. Ezek a program futását nem befolyásolják, de a későbbiekben hasznos útmutatást adhatnak, ha módosításra kerül sor. Megjegyzést felső vessző (apoztróf) begépelése után írhatunk, ezután a megjegyzés színe automatikusan zöldre vált.

```
Sub Mintaprogram03()  
  For x = 1 To 10 ' külső ciklus  
    For y = 1 To 5 ' belső ciklus  
      Worksheets("Mintalap03").Cells(x, y) = x 'eredmény megjelenítése  
    Next y ' belső ciklus zárása  
  Next x ' külső ciklus lezárása  
End Sub
```

HIBAKERESÉS ÉS ELLENŐRZÉS

Ha hibás sort írunk be (FOR x = 1 To 10 egy sorban kell lennie), a szöveg azonnal pirosra vált, és hibajelzést kapunk:

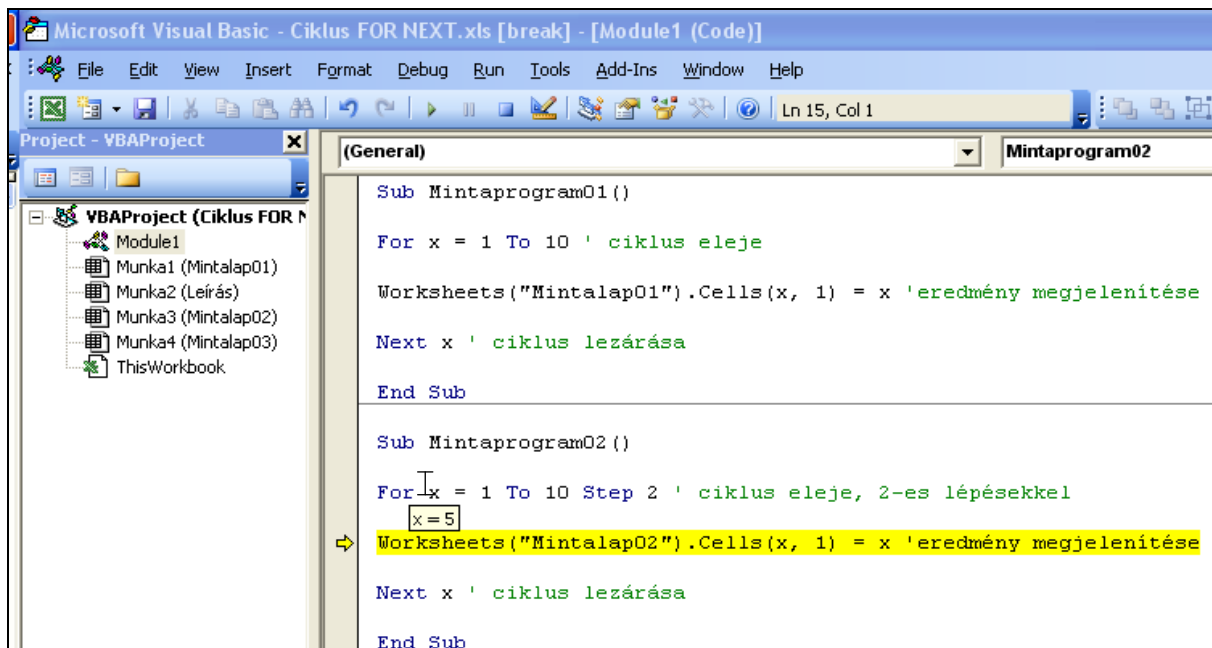
```
For x  
= 1 To 10 ' külső ciklus
```

Ha a sor helyes, de mégsem értelmezhető, például azért, mert egy nem létező (pl. „Mintalap05” nevű) munkalapra akarunk írni, akkor a program futása közben hibajelzést kapunk, a program pedig megáll. A hibajelzésen belüli **Debug** gomb megnyomásával visszatérhetünk a szerkesztőbe, ahol a hibás sor sárga nyíllal és sárga háttérrel van jelölve.

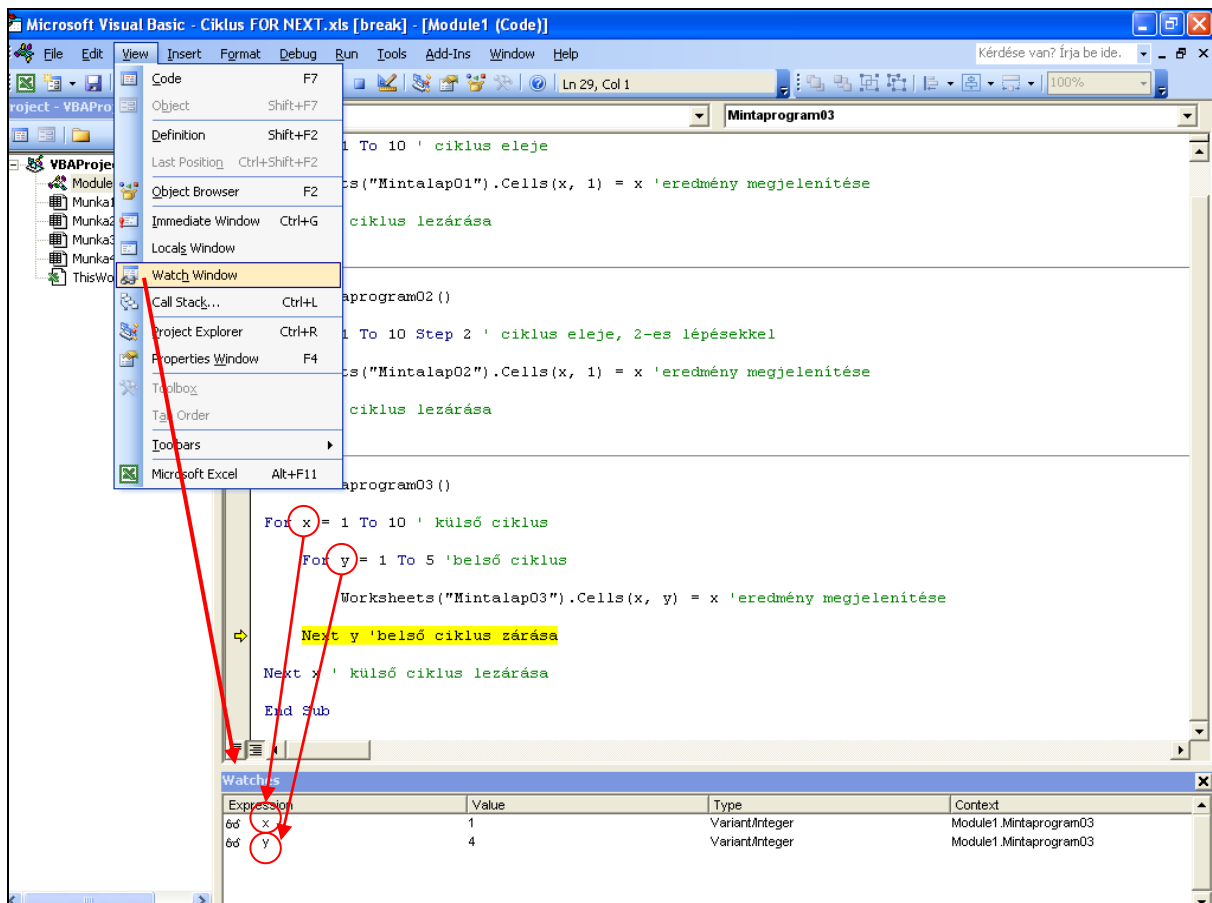
```
⇒ Worksheets("Mintalap05").Cells(x, y) = x ' eredmény megjelenítése
```

A hiba javítása után a **Mentés** ikonnal egy sorban lévő **Stop** gomb megnyomásával léphetünk ki a hibakezelési folyamatból. A program ezután újra elindítható.

Programunkat utasításonkénti végrehajtással is tesztelhetjük. Ehhez kattintsunk a kurzorral a program első sorára, majd nyomjuk meg egymás után többször az **F8** funkcióbillentyűt. Ilyenkor a hibajelzéshez hasonlóan az aktuális sor sárga háttérrel jelölt. Ha közben a kurzorral (□) egy változóra mutatunk, megjelenik annak aktuális értéke (A következő oldalon látható ábrán: x = 5).



Ha a léptetés során folyamatosan szeretnénk a változók értékét nyomon követni, akkor a felső menüsorból válasszuk ki a **View/Watch Window** funkciót. Ekkor a kódszerkesztő alatt megjelenik egy újabb ablak, amelybe a változókat egérrel behúzva (az egér bal gombját kijelölés után lenyomva tartva a kiválasztott objektum az egérrel mozgatható), vagy a változókra jobb egérgombbal rákattintva a megjelenő menüből kiválasztva az **Add Watch** menüt, lépésenként figyelhetjük a kiválasztott változók aktuális értékének alakulását.



TARTALOMJEGYZÉK

Alapműveletek és beépített függvények alkalmazása

Az Excel alkalmazás használata során a felhasználó általában a megszokott, táblázatos kialakítású munkalapokon dolgozik, ott vannak a kiindulási adatok, és oda várjuk az eredményt is, miközben a programok a háttérben futnak. A cellák kezelésével felvehetjük az adott program lefuttatásához szükséges adatokat, elvégezzük a számításokat, majd a végeredményt visszaírhatjuk a táblázatba.

Lekérdezések és beállítások

Lekérdezésekkel az aktuális cella, munkalap és munkafüzet beállításairól és paramétereiről kaphatunk információkat, illetve a lekérdezés megfordításával beállíthatjuk az általunk kívánt értékeket.

Üzenet megjelenítése

Az üzenetpanel segítségével információkat adhatunk a program felhasználójának, választási lehetőségeket kínálhatunk fel, vagy a program futásának eredményét jeleníthetjük meg.

Számláló típusú ciklus

A ciklus segítségével többször végrehajthatunk egy vagy több olyan utasítást, amelyet a ciklus magjában helyezünk el. A `For-Next` elől tesztelő, iteráló típusú számláló ciklus. Ebben a típusban a ciklus kezdetekor előre meghatározzuk a futás darabszámát.

Feltételes típusú ciklus

A ciklus segítségével többször végrehajthatunk egy vagy több olyan utasítást, amelyet a ciklus magjában helyezünk el. A `Do-Loop` egy elől vagy hátul tesztelő, logikai feltételt vizsgáló ciklus. Ebben a típusban a ciklus kezdetekor nem határozzuk meg a futás darabszámát, hanem azt egy feltétel teljesüléséhez kötjük.

Feltételes elágazás

A feltételes elágazás segítségével utasítások két csoportjának egyikét hajthatjuk végre a logikai vizsgálat eredményétől függően.

- Egy klasszikus példa eldöntendő kérdésre: Vigyek, vagy ne vigyek magammal esernyőt?
A logikai vizsgálat: kinézek az ablakon és megnézem, esik-e az eső.
A válasz: Igen vagy Nem.
 - Ha igen, esik: viszek ernyőt (A).
 - Ha nem esik: nem viszek ernyőt (B).

Esetvizsgálat

A `Select Case` utasítást használhatjuk feltételes elágazás létrehozására abban az esetben, ha kettőnél több kimenetre van szükségünk. A kifejezést összehasonlítjuk a mintákkal, és amelyekkel egyezik, az ahhoz tartozó programblokk fog lefutni.

Tömbök használata

Tömbök segítségével nagyszámú adatot tárolhatunk és kezelhetünk egy csoportban. A tömböt alkotó adatokat elemeknek nevezzük, melyekre a tömb nevével és a tömbindexek segítségével hivatkozhatunk.

Táblázat-, cella- és tartalom-formázási lehetőségek

A formázások segítségével átláthatóvá tehetjük a táblázatokat, kiemelhetjük a fontosabb adatokat, eredményeket, vagy adott esetben megjeleníthetjük a hibás adatokat, eltéréseket.

Saját függvény készítése

Saját (felhasználói) függvény definiálásának segítségével bonyolultabb, többlépcsős számításokat előre definiálhatunk egy függvényben. A függvényen belül lehetőség van kisebb programok elhelyezésére is. A saját függvény leírása az eddigi programokkal megegyező módon, egy modulban kerül elhelyezésre, meghívása viszont – az Excel saját függvényeivel azonos módon – a munkalap egyik cellájából történik. A makró programok listájában a függvények nem szerepelnek, működésükhöz nincs szükség makró indítására.

Program hívása

Egy programot a `Call` utasítás segítségével indíthatunk el egy másik programból. Ekkor az első program átadja a vezérlést az elindított második programnak, így inentől a második program utasításai kerülnek végrehajtásra. A második program futásának befejeztével a vezérlés visszakerül az első programhoz, annak a második programot elindító `Call` utasítást követő, soron következő sorára. Inentől tovább folytatódik az első program futása.

Szubrutin hívása

A szubrutin egy, a főprogramon belül található kisebb, külön névvel ellátott programrészlet. Ha a főprogram futása közben a `GoSub` utasítással meghívjuk a szubrutint, akkor az abban foglalt utasítások kerülnek végrehajtásra, ahonnan a `Return` utasítás hatására a program futása visszatér a főprogram `GoSub` utasítást követő, soron következő sorára. Maga a szubrutin a programtörzs végét jelölő `Exit Sub` utasítás után található. A szubrutin neve után kettőspontot kell tenni. Szubrutin alkalmazása akkor célszerű, ha egy utasítást a főprogram futása során többször végre kell hajtani.

Készítette: Excel Expert

www.excelexpert.hu

Kiadó: Management Kiadó Kft., 1135 Budapest, Kerekes u. 1/b.

www.managementkiado.hu