Lévayné dr. Lakner Mária – Baksa-Haskó Gabriella

Excel 2003 táblázatkezelés és programozás a gyakorlatban

120 feladattal

RÉSZLET

ComputerBooks
Budapest, 2006.

Tartalomjegyzék

9. Űrlapok készítése Excelben

Ebben a fejezetben azokat a lehetőségeit mutatjuk meg az Excelnek, amelyek segítségével új objektumokat helyezhetünk el a munkafüzetben. Lehetőségünk van az objektumok tulajdonságainak beállítására és ezáltal hasznos kiegészítői lehetnek a munkánknak. Egyúttal átvezetés is ez a fejezet az ezután következőhöz, az Excel programozásához.

9.1 Vezérlők

A **Nézet** menüben az **Eszköztárak** közül kapcsoljuk be a **Vezérlők eszköztár**at! Az eszköztáron megtaláljuk a Windows párbeszédpanelekről már ismert vezérlőket, melyeket elhelyezhetünk a munkalapon, valamint néhány ikont, ami a vezérlők szerkesztésében segít.

Vezérlők eszköztára

Megjegyzés: Az űrlap eszköztáron található vezérlők ugyanazon funkciókat valósítják meg, mint a vezérlő eszköztáron lévők, csak a felhasználónak sokkal kevesebb beállítási lehetősége van. Az űrlap vezérlői például programkódból nem elérhetőek. Ezért ezt az eszköztárat nem fogjuk használni.

A felhelyezhető vezérlők az eszköztáron lévő sorrendben: (A vezérlőknek sok neve van a köztudatban, mi magyarul és angolul is az Excel által használt neveket tüntetjük fel.)

– **Jelölőnégyzet** (**CheckBox**): Egy kis négyzet felirattal. Kijelölt és nem kijelölt állapota lehet. Kijelölt állapotban a négyzetben egy pipa van. A vezérlőkről **egyesével dönthető el**, hogy ki legyenek jelölve, vagy sem, vagyis egyidejűleg több is kiválasztható. Értéke kijelölt állapotban **IGAZ**, jelöletlen állapotban **HAMIS**.

Jelölőnégyzet

– **Beviteli mező** (**TextBox**): Olyan mező, amibe a tervező mód kikapcsolása után a felhasználó írni tud. Értéke a beleírt karaktersorozat szövegként.

Beviteli mező

Parancsgomb – **Parancsgomb** (**CommandButton**): Olyan vezérlő, amelyet virtuálisan meg lehet nyomni. Hatására egy eljárás indul el. Csak a későbbiekben fogjuk használni. **Értéke nincs.**

Választó-kapcsoló – **Választókapcsoló** (**OptionButton**)**:** Egy kis karika felirattal**.** Kijelölt és nem kijelölt állapota lehet. Kijelölt állapotot a karikában egy pont jelzi. A vezérlők **együtt működnek**. Több választókapcsolóból **egyszerre csak egy lehet kijelölve**. Ha másikat választunk ki, az előző kijelölésünk megszűnik. Értéke kijelölt állapotban **IGAZ**, jelöletlen állapotban **HAMIS.**

Listapanel – **Listapanel** (**ListBox**): Olyan mező, amiben egy felsorolás található, amiből a tervező nézet kikapcsolása után a felhasználó választhat. Értéke a kiválasztott karaktersorozat **szövegként.**

Beviteli lista – **Beviteli lista** (**ComboBox**): Beszélőbb nevén **legördülő lista.** Mint a listánál, itt is több lehetőség közül választhatunk. Alaphelyzetben egy üres mező látszik mellette egy lefelé mutató nyíllal. A nyílra kattintva láthatjuk a lista elemeit és választhatunk közülük. A választás után a lista eltűnik és csak a kiválasztott elemet látjuk. A listapanellel ellentétben itt a **felhasználó** maga **is beírhat a mezőbe**, ezáltal olyan értéket választva, amit a lista nem ajánlott fel. Értéke a kiválasztott, vagy beleírt karaktersorozat **szövegként.**

Váltógomb – **Váltógomb** (**ToggleButton**): A váltógomb a jelölőnégyzethez hasonlóan viselkedik, csak a kijelölt állapotot nem egy négyzetben lévő pipa mutatja, hanem az, hogy a gomb be van nyomva, vagy nincs. Értéke kijelölt állapotban **IGAZ**, jelöletlen állapotban **HAMIS.**

Léptetőnyíl – **Léptetőnyíl** (**SpinButton**): Két ellentétes irányba mutató nyíl. Segítségükkel számlálni tudunk a megadott határok között a megadott lépésközzel. Az alsó határértéknek kisebbnek kell lennie, mint a felsőnek, az értékek nem lehetnek negatívak és a lépésköz csak egész lehet. Értéke a kiválasztott szám számként.

Görgetősáv – **Görgetősáv** (**ScrollBar**): A léptetőnyíl rokona**,** azzal a különbséggel, hogy a két nyíl között egy sáv is található, aminek a segítségével nagyobb lépéseket is tehetünk. A sávon csúszka is van, amit a felhasználó a két szélső érték között szabadon tud csúsztatni.

Felirat – **Felirat** (**Label**): E vezérlő segítségével helyezhetünk el feliratokat a munkalapon, amelyeket a tervező mód kikapcsolása után a felhasználó nem tud változtatni. **Értéke nincs.**

Kép – **Kép** (**Picture**): E vezérlő segítségével tudunk képet felhelyezni a munkalapunkra**. Értéke nincs.**

A vezérlők eszköztáron található, a szerkesztést és programozást segítő ikonok:

 ikon segítségével tudjuk ki-be kapcsolni, hogy az adott vezérlőt szerkeszteni, vagy használni szeretnénk. (**Tervező mód - Kilépés a tervezésből**)

Tervező mód ikon

 ikon: Miután egy vezérlőt elhelyeztünk a munkalapon ezzel az ikonnal tudjuk a tulajdonságait beállítani. A tulajdonságok ablak tartalma nincs magyarra fordítva. (**Tulajdonságok**)

Tulajdonságok ikon

 ikon: Az adott vezérlőhöz kapcsolódó programkód megtekintése. Csak a későbbiekben fogjuk használni. (**Kód megjelenítése**)

Kód megjelenítése ikon

Néhány hasznos tulajdonság, melyeket a **Properties (Tulajdonságok)** ablakban abc sorrendben (**Alphabetic**), vagy témák szerint (**Categorized**) láthatunk és állíthatunk be:

Tulajdonságok

– **(Name)**: A vezérlő neve, ami áll a vezérlő típusnevéből, például Label, és egy sorszámból. A vezérlők átnevezhetőek, de ez kezdő felhasználók esetén nem javasolt.

Name

– **LinkedCell**: Ez a tulajdonság kitűntetett jelentőséggel bír. A vezérlőt ennek segítségével tudjuk cellához kapcsolni. Ha a vezérlő értéke módosul, akkor a kapcsolt cella értéke is, illetve fordítva. A vezérlők értékének **adattípusa a vezérlőtől függ**: a jelölőnégyzet, a választókapcsoló és a váltógomb logikai értéket, a léptetőgomb és a görgetősáv számot, a beviteli mező és a listák pedig szöveget adnak. A parancsgombnak, a feliratnak és a képnek nincs értéke, így LinkedCell tulajdonságuk sincs. Ha képletekben szeretnénk olyan cellára, mint számra hivatkozni, amit szövegként vittünk be, akkor az **ÉRTÉK**(cella) függvénnyel számmá kell konvertálnunk. A kapcsolt cellát az Excelben megszokott módon, tehát az oszlop betűjelével és a sor számával tudjuk megadni.

LinkedCell

Caption

– **Caption**: A felirattal rendelkező vezérlők feliratát adhatjuk meg.

– **Text**: A beviteli mezőnek és a beviteli listának nem felirata van, hanem szövege. Ezt a Text tulajdonság megadásával tudjuk beállítani.

Text

– **Font**: A felirattal, vagy szöveggel rendelkező vezérlők feliratának betűtípusát adhatjuk meg.

Font

– **ForeColor**: A felirattal, vagy szöveggel rendelkező vezérlők feliratának színét adhatjuk meg.

ForeColor

BackColor
- **BackColor**: Háttérszínt adhatunk a vezérlőnek.

Enabled
- **Enabled**: A vezérlő elérhetőségét lehet állítani. Két értékű tulajdonság: ha az értéke **True**, akkor a vezérlő használható (például a gomb megnyomható, vagy a beviteli mezőbe lehet írni), ha az értéke **False**, akkor a vezérlő nem használható.

Visible
- **Visible**: A vezérlő láthatóságát lehet vele állítani. Szintén két értékű tulajdonság: ha az értéke **True**, akkor a vezérlő látható, ha **False**, akkor nem látható. Természetesen amíg tervező módban vagyunk, addig minden látható, de ha a fent bemutatott ikonnal kilépünk a tervezésből, akkor a láthatatlanra állított vezérlők eltűnnek.

Value
- **Value**: Az értékkel rendelkező vezérlők aktuális értéke. Az egyes vezérlők bemutatásánál leírtuk, hogy milyen értékeket vehetnek fel.

Picture
- **Picture**: Azoknál a vezérlőknél, ahol megadhatunk feliratot (**Caption**), ott ezt helyettesíthetjük képpel is. Ezen kívül a Kép vezérlőnek tudunk még kép tulajdonságot beállítani. **Tallózás**sal tudjuk megadni a kívánt kép elérési útját.

ListFillRange
- **ListFillRange**: Itt adhatjuk meg azt a cellatartományt, amely a kiválasztható értékeket tartalmazza listát tartalmazó vezérlők esetén. (**Listapanel, Beviteli lista**) A cellatartományt az Excelben megszokott módon a két sarokcellát kettősponttal elválasztva tudjuk megadni.

Min, Max, SmallChange LargeChange
- **Min, Max, SmallChange, LargeChange**: a léptetőgomb és a görgetősáv tulajdonságai, amelyekkel meghatározhatjuk hogy milyen értékeket vegyenek fel. (LargeChange tulajdonsága csak a görgetősávnak van) Mivel negatív számokat és nem egész lépésközt nem lehet beállítani, ha ilyenre van szükségünk, akkor a vezérlő értékét transzformálni kell.

Méret, elhelyezés
- A vezérlők méretét és elhelyezését a tervező nézet bekapcsolása után szabadon változtathatjuk az egér segítségével a vezérlők oldalain és sarkain lévő méretező négyzetek segítségével, illetve a vezérlő közepére kattintva húzással. Ha pontos méretet vagy elhelyezést szeretnénk, azt elérhetjük a **Height** (magasság), **Width** (szélesség), **Top** (lap tetejétől való távolság), **Left** (lap bal szélétől való távolság) tulajdonságok képpontokban történő megadásával.

A vezérlők segítségével készíthetünk olyan űrlapokat, melyek hátterében az Excel végez számításokat, de a kiinduló értékeket a felhasználó az általunk meghatározott keretek között adhatja meg.

Feladat

Nyissuk meg a statisztika.xls munkafüzetet! A táblázatban az Agyabugya Bt. bevételei láthatók, mint ahogy az ábra is mutatja.

	A	B	C	D	E	F	G	H	I	J	K
1	Az Agyabugya Bt. forgalma 10 héten át ezer Ft-ban										
2											
3		1. hét	2. hét	3. hét	4. hét	5. hét	6. hét	7. hét	8. hét	9. hét	10. hét
4	Hétfő	586,29	442,69	864,66	1013,53	1065,99	261,4	461,56	1183,46	975,99	363,99
5	Kedd	865,97	972,38	993,93	565,73	1197,71	956,32	249,13	904,81	387,39	804,66
6	Szerda	948,9	774,23	942,37	256	341,11	278,77	1050,26	842,59	277,48	362,2
7	Csütörtök	372,31	387,8	795,78	546,68	1167,77	845,9	758,89	308,11	1196,16	711,13
8	Péntek	653,43	1176,43	936,73	802,75	1036,88	407,42	450,32	786,43	740,37	384,65

statisztika.xls

Helyezzünk fel egy léptetőnyilat, ami csak két értéket tud felvenni, 1-et és 2-őt! A B10-es cellában jelenjen meg a „Minimum” felirat, ha a léptetőnyíl értéke 1 és a „Maximum” felirat, ha a léptetőnyíl értéke 2. A mellette lévő cellában a megfelelő értéket számoljuk is ki a táblázatban szereplő 50 adatból az Excel beépített függvényeivel! (Ha a feladathoz felhasználunk olyan cellát, amit „el szeretnénk dugni”, akkor azt tegyük a felhelyezett vezérlő alá.)

Kapcsoljuk be a **vezérlők eszköztár**at és válasszuk róla a **léptetőnyil**at! A megjelenő hajszálkereszttel rajzoljunk egy fekvő téglalapot úgy, hogy az eltakarja az A10 cellát! Ha nem tetszik a léptetőnyíl mérete, akkor a határoló fekete pontokkal még utólag is tudunk rajta módosítani. Ha megvagyunk a rajzolással, akkor állítsuk be a megfelelő tulajdonságokat a következőképpen: A felsorolásban nem szereplő tulajdonságokat hagyjuk változatlanul!

LinkedCell	A10
Max	2
Min	1
SmallChange	1

A **tulajdonságok beállítása** után zárjuk be a **Properties** ablakot és **kapcsoljuk ki a tervező módot**! Ha most a léptetőnyíl gombjaira kattintgatunk nem fogunk látni semmit, mert a kapcsolt cellát eltakartuk a vezérlővel, de a kurzormozgató billentyűkkel rá tudunk állni az A10-es cellára és akkor a szerkesztőlécen látjuk a tartalmát. Ha ezután kattintunk a léptetőnyíl két gombjára felváltva, akkor már látszik, hogy valóban az 1 és 2 értékeket veszi fel.

Most álljunk a B10 cellába és írjuk be a megfelelő függvényt, hogy a Maximum és Minimum felirat helyesen váltakozzon! A megoldáshoz a **HA** függvény kell:

=HA(A10=1;"Minimum";"Maximum")

A C10 cellában hasonló módon a **HA** függvényt hívjuk segítségül, csak most nem szövegkonstansok lesznek a paraméterek, hanem beágyazva a megfelelő statisztikai függvények:

=HA(A10=1;MIN(B4:K8);MAX(B4:K8))

Ezzel a feladat elkészült.

statisztika.xls megoldása

C10 fx =HA(A10=1;MIN(B4:K8);MAX(B4:K8))

	A	B	C	D	E	F	G	H	I	J	K
1	Az Agyabugya Bt. forgalma 10 héten át ezer Ft-ban										
2											
3		1. hét	2. hét	3. hét	4. hét	5. hét	6. hét	7. hét	8. hét	9. hét	10. hét
4	Hétfő	586,29	442,69	864,66	1013,53	1065,99	261,4	461,56	1183,46	975,99	363,99
5	Kedd	865,97	972,38	993,93	565,73	1197,71	956,32	249,13	904,81	387,39	804,66
6	Szerda	948,9	774,23	942,37	256	341,11	278,77	1050,26	842,59	277,48	362,2
7	Csütörtök	372,31	387,8	795,78	546,68	1167,77	845,9	758,89	308,11	1196,16	711,13
8	Péntek	653,43	1176,43	936,73	802,75	1036,88	407,42	450,32	786,43	740,37	384,65
9											
10		Maximum	1197,71								
11											

9.2 Feladatok

1. Feladat

Egy üres Excel munkalapra helyezzük fel a felsorolt vezérlőket és a LinkedCell tulajdonságokat állítsuk be különböző cellákra. Ezután lépjünk ki a tervező módból és próbáljuk ki a vezérlőket. Láthatjuk, hogy a választókapcsolónál, a jelölőnégyzetnél és a váltókapcsolónál szürkén jelzi az Excel, ha a kapcsolt cella tartalmát kitöröljük.

2. Feladat

Egészítsük ki a fejezetben megoldott feladatot úgy, hogy a minimum és a maximum mellett más statisztikai mutatókat is kiszámoljon, pl. átlagot, szórást, mediánt móduszt! *(Ha túl sokfélét szeretnénk a lehetőségek között felsorolni és a HA függvények egymásba ágyazása kényelmetlenné válik, akkor egy külön táblázatba számoljuk ki az összes értéket és a megfelelő cellába egy FKERES függvénnyel írassuk ki a megfelelő értékeket. A külön táblázatot helyezhetjük egy külön munkalapra, aminek a visible tulajdonságát hamisra állítjuk, így elrejthetjük a felhasználó elől.)*

3. Feladat

Nyissuk meg a bevetelek1.xls munkalapot! A munkalapon helyezzünk el egy listát, amelyből ki lehet választani egy üzletet! Egy beviteli mezőben az adott üzlet átlagforgalma jelenjen meg! A megoldást egyetlen összetett képlettel számoljuk ki! Használjuk a HOL.VAN és az INDEX függvényeket!

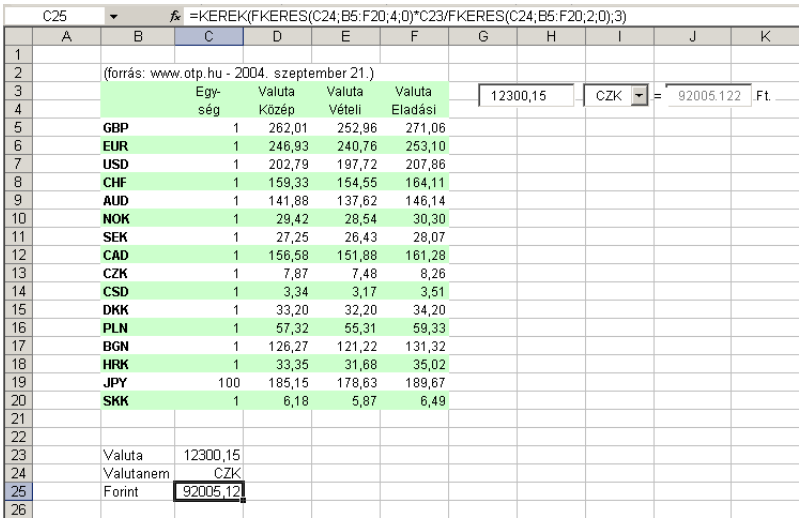
4. Feladat

Nyissuk meg a valutavalto.xls munkafüzetet és készítsünk egy valutaváltó űrlapot!

	A	B	C	D	E	F	G
1							
2		(forrás: www.otp.hu - 2004. szeptember 21.)					
3			Egy-		Valuta	Valuta	
4			ség	Közép	Vételi	Eladási	
5		**AUD**	1	141,88	137,62	146,14	
6		**BGN**	1	126,27	121,22	131,32	
7		**CAD**	1	156,58	151,88	161,28	
8		**CHF**	1	159,33	154,55	164,11	
9		**CZK**	1	7,87	7,48	8,26	
10		**CSD**	1	3,34	3,17	3,51	
11		**DKK**	1	33,20	32,20	34,20	
12		**EUR**	1	246,93	240,76	253,10	
13		**GBP**	1	262,01	252,96	271,06	
14		**HRK**	1	33,35	31,68	35,02	
15		**JPY**	100	185,15	178,63	189,67	
16		**NOK**	1	29,42	28,54	30,30	
17		**PLN**	1	57,32	55,31	59,33	
18		**SEK**	1	27,25	26,43	28,07	
19		**SKK**	1	6,18	5,87	6,49	
20		**USD**	1	202,79	197,72	207,86	
21							

Az átváltani kívánt valutaösszeg beviteli mezőbe kerüljön, mellette beviteli listából lehessen kiválasztani a valutanemet! A fenti két vezérlőt csatoljuk az Excel munkafüzet egy-egy cellájához, melyek alapján egy harmadik cellában FKERES függvényt használva határozzuk meg az eredményt! Ezt szintén beviteli mezőben jelenítsük meg, melynek Enabled tulajdonságát False-ra állítva megtilthatjuk a módosítást! *(Vegyük figyelembe a számításnál, hogy nem mindegyik valutánál ugyanaz az egység!)*

Megoldás:



C25 =KEREK(FKERES(C24;B5:F20;4;0)*C23/FKERES(C24;B5:F20;2;0);3)

	A	B	C	D	E	F	G	H	I	J	K
1											
2		(forrás: www.otp.hu - 2004. szeptember 21.)									
3			Egy-	Valuta	Valuta	Valuta		12300,15	CZK	= 92005.122	Ft.
4			ség	Közép	Vételi	Eladási					
5		GBP	1	262,01	252,96	271,06					
6		EUR	1	246,93	240,76	253,10					
7		USD	1	202,79	197,72	207,86					
8		CHF	1	159,33	154,55	164,11					
9		AUD	1	141,88	137,62	146,14					
10		NOK	1	29,42	28,54	30,30					
11		SEK	1	27,25	26,43	28,07					
12		CAD	1	156,58	151,88	161,28					
13		CZK	1	7,87	7,48	8,26					
14		CSD	1	3,34	3,17	3,51					
15		DKK	1	33,20	32,20	34,20					
16		PLN	1	57,32	55,31	59,33					
17		BGN	1	126,27	121,22	131,32					
18		HRK	1	33,35	31,68	35,02					
19		JPY	100	185,15	178,63	189,67					
20		SKK	1	6,18	5,87	6,49					
21											
22											
23		Valuta	12300,15								
24		Valutanem	CZK								
25		Forint	92005,12								
26											

5. Feladat

Készítsük el a Forintról idegen valutára váltó verziót is!

6. Feladat

Nyissuk meg az arajanlat.xls munkafüzetet és készítsünk árajánlatkérő űrlapot! A nyomdai munka ára: példányszám * oldalszám * oldalár. Erre jön példányonként a kötés ára és szintén példányonként a borító ára.

7. Feladat

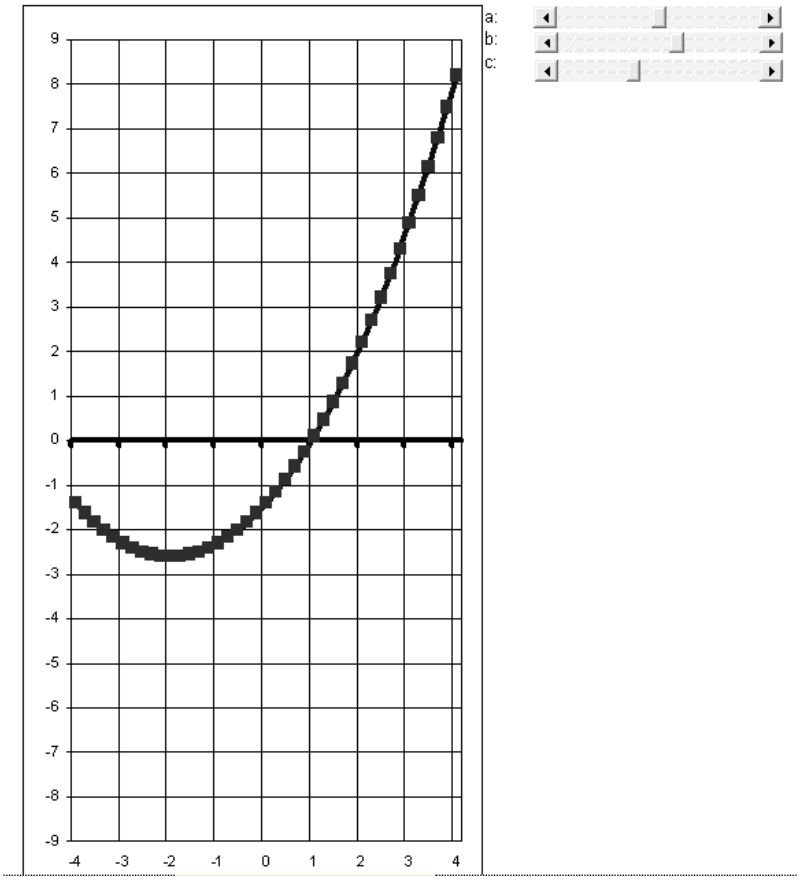
Készítsünk kerekítő űrlapot! Beviteli mezőben lehessen megadni, hogy milyen számot szeretnénk kerekíteni, a kerekített érték is beviteli mezőben jelenjen meg (ne lehessen megváltoztatni) és görgetősávval lehessen megadni, hogy mire kerekítsünk! A KEREK függvény hány_számjegy paramétere –5 és +5 között legyen változtatható!

8. Feladat (+)

Készítsünk másodfokú függvényt ábrázoló diagrammot! Az x értékek változzanak –10 és +10 értékek között két tizedenként! Az

f_x=a*(x+b)^2+c függvény a, b és c paramétereit gördítősávval lehessen megadni szintén –10 és +10 között egy tizedes változással! A diagrammot formázzuk úgy, hogy jól követhető legyen! *(Figyeljünk arra, hogy a gördítősáv alsó határértéke sem lehet negatív és a lépésköz is csak egész szám lehet, így nem tudjuk közvetlenül előállítani a feladat által kért értékeket csak külön cellákba transzformációval.)*

Megoldás: csuszka_diagram_mo.xls



9. Feladat

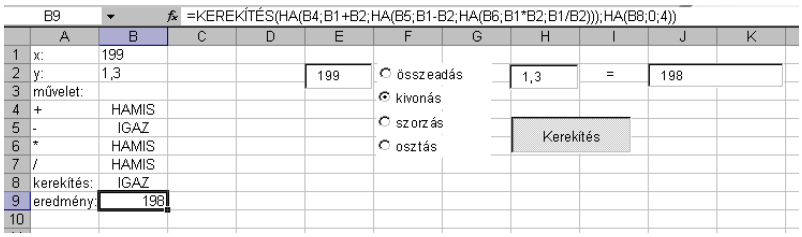
Készítsünk űrlapot, amely segítséget nyújt betétlekötéshez! Nyissuk meg a betet.xls munkafüzetet és abba dolgozzunk! Hogy melyik bankba helyezzük el a betétet azt beviteli listából lehessen kiválasztani, az éves kamatot függvénnyel határozzuk meg a táblázatból, a havi kamat

az éves kamat 1/12-e. A hónapokban mért futamidő megadásához használjunk gördítősávot (legfeljebb 3 évre lehessen számolni), a lekötött összeget pedig beviteli mezőbe lehessen beírni. A végösszeg kiszámításához a JBÉ pénzügyi függvényt használjuk (kamat, időszakok_száma, mai_érték)!

10. Feladat

Készítsünk számológépet, amely a 4 alapműveletet tudja két szám között! A két számot beviteli mezővel lehessen megadni, a műveletet választókapcsolóval lehessen kiválasztani és legyen egy váltógomb, aminek benyomott állapota mellett egészre kerekíti az eredményt, egyébként pedig 4 tizedesre!

Megoldás:



9.3 Mit tanultunk meg

– A Vezérlők eszköztár ikonjai
– A vezérlők típusai és legfontosabb tulajdonságaik

10. Az Excel programozása Visual Basicben, makró készítés

Az Excel táblázatkezelővel sok probléma megoldható a beépített függvények és eljárások segítségével. Vannak más problémák, amelyekre az Excelnek nincsenek közvetlenül függvényei ill. eljárásai, de rendelkezik olyan lehetőségekkel, amelyeket felhasználva a megoldást magunk készíthetjük el.

Írhatunk programokat és fejleszthetünk az Excel saját függvényein kívüli saját felhasználói függvényeket. Az Excel Visual Basic for Applications (VBA) háttere használható fel ezekre a célokra.

A továbbiakban bepillantást kapunk a VBA programozásba. Rövid elméleti bevezetők után feladatokkal próbáljuk segíteni az elsajátítást. A feladatok egy része mellett megoldás is található. A nem megoldott feladatok között vannak a minta alapján egyszerűen megoldhatók, vannak összetettebbek és van néhány kiemelt, amelynek megoldásához plusz ötletre is szükség van.

10.1 Előkészületek

A **makró** egy rögzített műveletsorozat, melyet a felhasználó könnyedén újra és újra végre tud hajtatni a számítógéppel. Makrót létrehozhatunk rögzítéssel, vagy magunk írhatjuk őket. A két módszert kombinálva is lehet használni.

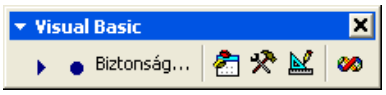
Makró

A makrórögzítést általában arra használjuk, hogy az Excel által megírt programrészleteket később saját programjainkban felhasználjuk.

Makrókészítés előtt bizonyos előkészületeket kell tennünk:

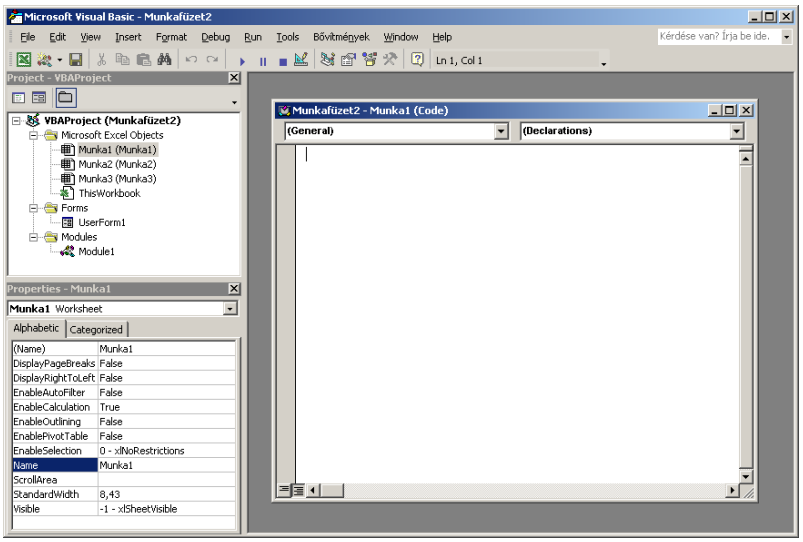
Kapcsoljuk be a Visual Basic eszköztárat! Az eszköztár ikonjai megfelelnek az Eszközök menü Makró almenüjének menüpontjainak:

Visual Basic eszköztár



– **Makró indítása** (Eszközök/Makró/Makrók…): Ezzel a gombbal nyithatjuk meg azt a párbeszédpanelt, amelyben kiválaszthatjuk a már meglévő makróinkat és eldönthetjük, hogy mit akarunk velük csinálni:
  - **Indítás**: lefuttathatjuk az adott makrót
  - **Mégse**: bezárjuk a párbeszédpanelt
  - **Lépésenként**: az adott makrót soronként futtatjuk le
  - **Szerkesztés**: felnyílik a Visual Basic szerkesztő ablak, ahol a meglévő makrót módosíthatjuk
  - **Létrehozás**: ha nem a listából választunk, hanem egy új nevet írunk be, akkor megnyomhatóvá válik a Létrehozás gomb, ami az előző pontban is említett ablakot nyitja meg, ahol az általunk megadott néven létrejövő üres makrót szerkeszthetjük, vagyis írhatjuk meg
  - **Törlés**: a kiválasztott makrót törölhetjük vele
  - **Egyebek**: a kiválasztott makróhoz billentyűkombinációt rendelhetünk és megadhatunk egy leírást
– **Makró rögzítése** (Eszközök/Makró/Új makró rögzítése…): Ezzel a gombbal nyithatjuk meg azt a párbeszédpanelt, amelyben megadhatjuk a rögzítendő makró tulajdonságait:
  - nevét
  - billentyűparancsát
  - helyét
  - és leírását
– Biztonság... **Biztonság** (Eszközök/Makró/Biztonság…): Itt három biztonsági szint közül választhatunk. Számunkra a közepes szint a megfelelő.
– **Visual Basic** (Eszközök/Makró/Visual Basic): Ezzel a gombbal nyithatjuk meg a Visual Basic szerkesztőfelületet.
– **Vezérlők eszköztára**: Ezzel a gombbal ki-be kapcsolhatjuk a Vezérlők eszköztárat.
– **Tervező mód**: Ezt a gombot már ismerjük a Vezérlők eszköztárról. Ezzel tudjuk ki-be kapcsolni, hogy a vezérlőinket szerkeszteni, vagy használni tudjuk.
– **Microsoft Script Editor**: HTML scriptek írásához nyújt segítséget. Ezt a gombot nem fogjuk használni.

Kattintsunk a már többször említett Visual Basic ikonra, és tekintsük meg részletesebben a megjelenő ablakot!



Visual Basic szerkesztő

A **Visual Basic szerkesztő** egy új ablakban nyílik meg saját menüvel és eszköztárral. A View menüben bekapcsolhatjuk azokat az ablakokat, amelyek a fenti ábrán is láthatók:

– **Project Explorer**: Ebben az ablakban láthatjuk az összes megnyitott munkafüzetünk elemeit fa szerkezetben. Ha csak egy munkafüzetünk van nyitva és még nem írtunk makrót, akkor egy törzs van: **VBAProject** (Munkafüzet neve). Ebből a törzsből nyílik a **Microsoft Excel Objects** könyvtár, ami tartalmazza a **munkafüzet munkalapjai**t egyesével zárójelben az általunk adott munkalapnevekkel, és egy **ThisWorkbook** nevezetű objektumot, ami az egész munkafüzetet együtt jelenti. Az **Insert** menüből, vagy az eszköztárról további elemeket szúrhatunk be a munkafüzetbe amelyek szintén könyvtárakba rendeződnek:

Project Explorer

- **Modules**: ha beszúrunk egy modult, rögtön létrejön egy Modules nevű könyvtár és ezentúl minden modulunk ide kerül. Modult az Insert menü Module menüpontjával, vagy az eszköztár ikonjának listájából tudunk beszúrni.
- **Forms**: ha beszúrunk egy UserFormot, rögtön létrejön egy Forms nevű könyvtár és ezentúl minden formunk ide kerül.
- Az ablak tetején látható 3 ikon a következő:
- **View Code**: a listából kiválasztott elem kódját mutatja a jobb oldali ablakban

  - **View Object**: a listából kiválasztott objektumot mutatja
  - **Toggle Folders**: a fent bemutatott mappaszerkezetet lehet vele elrejteni és akkor az objektumok „ömlesztve” jelennek meg

Properties Window

– **Properties Window**: Mindig a kiválasztott objektum tulajdonságait látjuk hasonlóan, mint az előző fejezetben megismert Tulajdonságok ablakban.

Makrót a Project Explorer ablaknál felsorolt minden elemhez írhatunk. Ha az ablakban a kiválasztott elemre **duplán kattintunk**, akkor jobb oldalt a szürke területen nyílik egy újabb ablak, ahol az adott elemhez tartozó makrók láthatók, illetve ide írhatók az új makrók.

Minden makrónak nevet kell adnunk. A **makrónév** egy szóból állhat, mely **nem kezdődhet számmal**, és nem tartalmazhatja a legtöbb írásjelet. **Ékezetes betűk használata** korábbi Windows verziókkal való kompatibilitás miatt **nem ajánlott**. Névnek már foglalt Visual Basic és Excel kulcsszavakat sem szabad adni, mert keveredéseket okozhat.

A makrók az Excel munkafüzettel együtt mentődnek.

10.1.1 Feladatok

11. Feladat

Nyissuk meg a makro.xls munkafüzetet! A munkafüzet tartalmaz egy elso nevű makrót, amely a munkalap első cellájába piros vastag betűkkel beírja, hogy ELSŐ. Futtassuk le a makrót! Nézzük meg a kódját a Visual Basic szerkesztőben!

12. Feladat

Nyissunk meg egy üres Excel munkafüzetet és a Visual Basic szerkesztőben hozzunk létre egy modult!

10.1.2 Mit tanultunk meg

– A Visual Basic eszköztár ikonjai
– A Visual Basic szerkesztő felépítése
– A makró neve

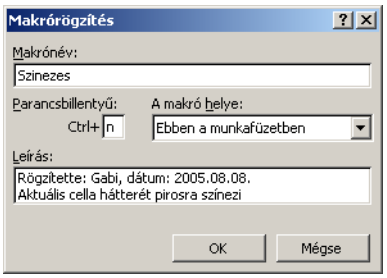
10.2 Makrók rögzítése

Makró rögzítése

Álljunk egy üres munkalap A1-es cellájára! Az előző fejezetben látott **Visual Basic eszköztár**on nyomjuk meg a **Makró rögzítése** gombot!

A felugró párbeszédablakban megadhatjuk a **makró nevét** (érdemes az alapértelmezettet megváltoztatni valami beszélő névre), a **parancsbillentyűt** (amit ha a Ctrl gombbal egyidejűleg megnyomunk, akkor azonnal futtatható a makró) a makró **helyét** és egy **leírást**, ami alapértelmezésben a rögzítő személyét és a rögzítés dátumát tartalmazza.

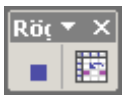
Makró rögzítése ablak



Az ábrának megfelelően névnek írjuk be a „Szinezes” szót, billentyűparancsnak az „n” betűt, a makró helyét ne változtassuk, a leírást pedig egészítsük ki a leendő program rövid leírásával!

Az **Ok** gomb megnyomása után **a gép minden mozdulatunkat rögzíti**, a kijelöléseket, formázásokat és beírásokat is, egészen a **Rögzítés vége** (bal oldali gomb) megnyomásáig.

Rögzítés vége eszköztár



Ha a rögzítés közben nem váltunk kijelölést, akkor a makrót mindig a kijelölt cellákra tudjuk majd lefuttatni, hiszen a cellák kijelölését ebben az esetben nem rögzítjük. A rögzítés vége eszköztár jobb oldali gombja a **Relatív hivatkozás.** Ezt megnyomva ha kijelölést váltunk a program nem a cella abszolút címét rögzíti, hanem az elmozdulás irányát és mértékét.

Relatív hivatkozás

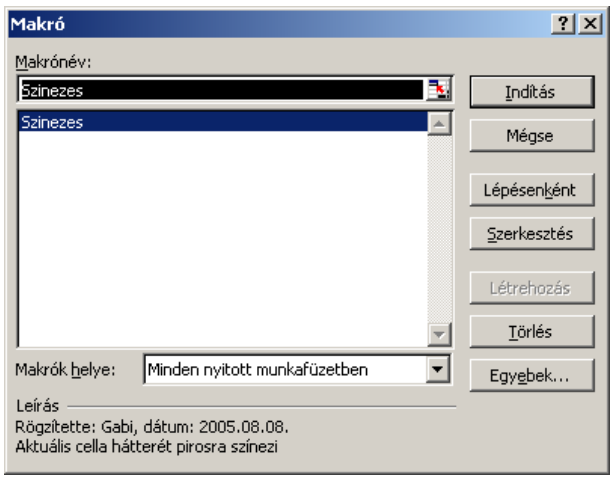
Először a leírásnak megfelelően a kijelölt cella hátterének színezését szeretnénk rögzíteni, tehát az adatok megadása és az Ok gomb megnyomása után, vigyázva, hogy **ne tegyünk felesleges lépéseket**, színezzük az aktuális (azaz az A1-es) cella hátterét pirosra a Formázás

eszköztár segítségével. Rögtön ezután nyomjuk meg a Rögzítés vége gombot.

Megjegyzés: makró rögzítése után soha ne feledkezzünk meg a **Rögzítés vége** gomb megnyomásáról, különben a makró rögzítése tovább folytatódik, esetleg a program lefagyásához vezető hibát okozhat.

A Szinezes nevű makrónk bekerült a munkafüzetbe. A Visual Basic eszköztár **Makró indítása** gombjával megnyíló ablakban szerepel a makrók listáján.

Makró indítása ablak



Makró indítása

Ha már több eleme lenne a listának, akkor rákattintással választhatnánk közülük. Most, mivel csak egy elemünk van, ezen állunk és az **Indítás** gomb megnyomásával lejátszhatjuk a makrót. Ha az A1 cellán állva indítjuk el a lejátszást, semmit nem fogunk látni, hiszen a már eddig is piros cellát színezi újra pirosra a program. Most álljunk át egy másik cellára és próbáljuk ki a makró indításának másik módját! Megadtuk **gyorsbillentyű**nek az „n” betűt, tehát ha kijelöltük azt a cellát, vagy tartományt, amit színezni szeretnénk, nyomjuk meg a **Ctrl + n** billentyűket. A kijelölt terület háttérszíne piros lesz.

Megjegyzés: a makró műveletei nem vonhatók vissza a Visszavonás paranccsal.

Készítsünk egy második makrót is rögzítéssel, ahol kihasználjuk, hogy a kijelöléseket is rögzíti az Excel!

Feladat

Készítsünk makrót, amely az aktuális cellától kezdődően beírja egymás alá a hét napjait, majd az A1-es cellát narancssárgára színezi!

Álljunk a munkalap egy tetszőleges cellájára és indítsuk el a rögzítést, adjuk a makrónak a „Hivatkozasok” nevet, majd nyomjuk meg az Ok gombot! Mielőtt bármit tennénk, győződjünk meg róla, hogy a **Relatív hivatkozás** be van kapcsolva (a bekapcsolt állapotot egy szürkés háttér jelzi)! Írjuk be a cellába, hogy „hétfő”, aztán **Enter**-rel lefelé haladva sorba a hét napjait vasárnapig. Ezután kapcsoljuk ki a Relatív hivatkozást, álljunk az A1-es cellára és a Formázás eszköztár segítségével színezzük narancssárgára! Most fejezzük be a rögzítést!

A makró rögzítése után a munkalap

	A	B	C
1			
2			hétfő
3			kedd
4			szerda
5			csütörtök
6			péntek
7			szombat
8			vasárnap

A kipróbáláshoz kattintsunk a munkafüzetünk egy másik munkalapjára és a fent tanult módon futassuk a makrónkat! Láthatjuk, hogy a napokat az aktuális cellától kezdődően írja (relatív elmozdulás), míg a színezés ez esetben is az A1-es cellára vonatkozik (abszolút hivatkozás).

Az előző fejezetben már láttuk a **Visual Basic szerkesztő** ablakának felépítését. Most a Visual Basic eszköztáron lévő gombbal nyissuk meg és nézzük meg az Excel által automatikusan írt két makrót!

Rögzített kódok

```
Sub Szinezes()
'
' Szinezes Makro
' Rögzítette: Gabi, dátum: 2005.08.08.
' Aktuális cella hátterét pirosra színezi
'
' Billentyűparancs: Ctrl+n
'
    With Selection.Interior
        .ColorIndex = 3
        .Pattern = xlSolid
    End With
End Sub
```

```
Sub Hivatkozas()
'
' hiv Makro
' Rögzítette: Gabi, dátum: 2005.08.09.
'

'
    ActiveCell.FormulaR1C1 = "hétfő"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "kedd"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "szerda"
…
    Range("A1").Select
    With Selection.Interior
        .ColorIndex = 45
        .Pattern = xlSolid
    End With
End Sub
```

Mindkét kód elején a **Sub** kulcsszó után a makró általunk adott neve látható, majd két zárójel. A következő sorok **'** (aposztrófot) jellel kezdődnek és zöldek, ami azt jelenti, hogy valójában nem a program részei, csak információk, **megjegyzés**ek. Az ezután következő sorok a program valódi utasításai, míg zárásul mindig az **End Sub** parancssor áll.

A program utasításait egy kis angoltudással megérthetjük, de megtanulnunk nem szükséges őket, mert az általunk írt saját makrók egy kissé ettől eltérőek lesznek.

10.2.1 Feladatok

13. Feladat

Készítsünk rögzítéssel makrót, amely a munkalap A1:H8 celláit sakktáblaszerűen kiszínezi!

14. Feladat

Nyissuk meg a bevetelek2.xls munkafüzetet! A Munka1 munkalapon egy táblázatot látunk a Gyakorlás Kft. üzemeinek 2004-es bevételeiről. Készítsünk egy-egy kördiagramot az egyes üzemek bevételeinek havonkénti megoszlásáról úgy, hogy a diagram készítésekor semmi mást ne adjunk meg, csak az ábrázolandó adatokat és a diagramok

címét! Rögzítsünk „Diagram” néven makrót, amelyben megadjuk az A üzemről készült diagram kategóriafeliratainak a hónapok nevét, feliratnak százalékot kérünk és kihúzással kiemeljük a körből a júniusi bevételt! A kész makrót futassuk le a másik 3 diagram kijelölése után is!

10.2.2 Mit tanultunk meg

- Makró rögzítése abszolút és relatív hivatkozásokkal
- Rögzített makró lejátszása
- Rögzített makró kódjának megnézése

10.3 Makrók írása Excelben

Ha olyan makrót szeretnénk írni, amely nem használ vezérlőket és a munkafüzetünk minden munkalapján egyformán szeretnénk tudni futtatni, akkor egy **modul** elemhez kell írni. Mint ahogy a rögzített makróknál már láttuk, a makró kezdő- és zárósora mindig az alábbi:

```
Sub makrónév()
…
End Sub
```

Kezdő és zárósor

A programkódban hivatkozhatunk a munkafüzetünk különböző részeire. A hivatkozást különbözőképpen használhatjuk. Mi az általunk programozás szempontjából legpraktikusabb módot foglaljuk össze a következő táblázatban (láthattuk, hogy rögzítéskor az Excel nem ezeket használja):

	Az Excelbeli elnevezések	**Programkódbeli megfelelő**
1.	A B3 cella	Cells(3,2)
2.	A C4:G6 tartomány	Range(Cells(4,3),Cells(6,7))
3.	A B oszlop	Columns(2)
4.	A H, I, J oszlopokból álló tartomány	Range(Columns(8),Columns(10))
5.	A 2. sor	Rows(2)
6.	A 13.-tól a 16. sorig tartó tartomány	Range(Rows(13),Rows(16))
7.	A munkalap összes cellája	Cells

Cella-hivatkozások a programban

8.	Meghatározott munkalap cellái	Sheets(2).Cells
9.	Az éppen aktuális cella (ha tartomány van kijelölve, aktuális cella akkor is csak egy van, aminek a tartalma a szerkesztőlécben látszik)	ActiveCell
10.	Az éppen kijelölt objektum (cella, tartomány, rajz, diagramm)	Selection

Alárendelt objektum

A táblázat 8. sorában láthatunk egy példát arra (**Sheets(2).cells**), hogy az alárendelt objektumokat ponttal tudjuk elválasztani egymástól.

Cellák tulajdonságai

A celláknak is vannak tulajdonságaik, mint a vezérlőknek. Programból úgy tudjuk ezeket a tulajdonságokat megadni, hogy a cella, vagy tartomány után írjuk **ponttal elválasztva** a tulajdonságot és **=** jel után adunk neki értéket. A leggyakrabban használt tulajdonságok:

– **Value**: a cella értéke, ha szöveg, akkor idézőjelek közé kell tenni. Ez az alapértelmezett tulajdonság, ezért ha nem írunk tulajdonságot, akkor ezt tudjuk megadni. A következő két sor eredménye ugyan az: az A1-es cellába az Alma felirat kerül:

```
Cells(1,1).Value = ”Alma”
Cells(1,1) = ”Alma”
```

– **Interior.Color**: a cella háttérszíne szövegesen megadva. Az angol színelnevezéseket lehet használni, csak elé kell írni egybe vele, hogy vb *(csak az alapszínek: fekete, fehér, sárga, piros, kék és zöld)*:

```
Cells(1,1).Interior.Color = vbRed
```

– **Interior.ColorIndex**: a cella háttérszíne számmal megadva. A Visual Basic 0-tól 56-ig számozza a színeket. A 0 színkód alapértelmezett színt jelent: háttér esetén fehér, karakter esetén fekete. Az A1 cella háttérszínét a következőképpen is pirosra lehet állítani *(a 3-as a piros szín kódja)*:

```
Cells(1,1).Interior.ColorIndex = 3
```

– **Font**: a cella betűtípusa. Újabb ponttal elválasztva lehet megadni stílust, típust, színt a fent ismertetett módokon. A következő táblázatban példákat találunk az A1 cella betűformázásaira:

Betűtípus Arial-ra állítása	Cells(1,1).Font.Name = ”Arial”
Betűszín pirosra állítása	Cells(1,1).Font.Color = vbRed vagy Cells(1,1).Font.ColorIndex = 3
Betűméret 12-esre állítása	Cells(1,1).Font.Size = 12

Félkövér betűre állítás	Cells(1,1).Font.Bold = True
Dőlt betűre állítás	Cells(1,1).Font.Italic = True

- **Row, Column**: a cella sor és oszlopszáma. Olyan tulajdonságok, amit csak lekérdezni lehet, megváltoztatni nem.
- **Formula**: idézőjelek között megadhatjuk a képletet, amit a cellába írnánk. Ugyanúgy, mint a Value, elhagyható.
- **FormulaR1C1**: a fenti tulajdonság egy változata, ahol a képletben a sor és oszlopszámok megadásával hivatkozhatunk a cellákra. Ugyanúgy, mint a Value, elhagyható.

Több tulajdonság állítása

Egy objektum **több tulajdonságának megadásakor** nem kell az objektum megnevezését minden sorban leírni, hanem a **With, End With** kulcsszavak között elég egyszer megadni, hogy melyik objektumról van szó és aztán csak a tulajdonságokat és a tulajdonságértékeket kell egymás alatt sorolnunk minden sort ponttal kezdve. Például ha egy cella betűtípusát a fenti táblázat alapján akarjuk beállítani, akkor azt megtehetjük a következőképpen is:

```
Sub formaz()
With Cells(1, 1).Font
    .Name = "Arial"
    .Color = vbRed
    .Size = 12
    .Bold = True
    .Italic = True
End With
End Sub
```

Megjegyzés

A programkódban használhatunk megjegyzéseket, hogy megírt eljárásainkat később is könnyen értelmezni tudjuk. A megjegyzéseket **'** jel után írjuk (láthattuk a makró rögzítésénél is). Ezeket a program futásakor a számítógép nem veszi figyelembe és a kódban zöld színnel jelenik meg.

Túl hosszú sorok

Túl hosszú sorok írása esetén mód van a sortörésre az **_** (**alulvonás**) karakter segítségével. Ahol szeretnénk a sort eltörni, ott beírjuk ezt a karaktert és Enter-t nyomunk a program így egy sornak fogja tekinteni.

10.3.1 Képletek írása

Képletek

A program írása során a cellákba írhatunk **képlet**eket is. Mint a fentiekben már említettük, erre több lehetőségünk van, a **Formula**, a **FormulaR1C1** tulajdonságok megadásával, vagy tulajdonságnév

elhagyásával. A programozás során írt képleteinknél is fontos, hogy **abszolút** vagy **relatív hivatkozás**t használunk.

A Formula tulajdonság megadásánál ugyanúgy a dollárjelek segítségével adhatjuk meg, hogy a hivatkozás abszolút, mint a cellába közvetlenül gépelve.

A FormulaR1C1 megadásánál a sor számát mindig egy **R**, az oszlop számát pedig egy **C** betű után kell írni. Azaz például a B1-es cella az R1C2 kóddal hivatkozható. Ez a hivatkozás abszolút. Relatív hivatkozásnál a sorszám helyett azt kell számmal megadnunk, hogy hány cellát kell elmozdulnunk. Ha nem váltunk sort, vagy oszlopot, akkor nem kell írni semmit, ha váltunk, akkor pedig az elmozdulás mértékét szögletes zárójelek között kell megadni. Azaz az eggyel fel és kettővel jobbra lévő cellára hivatkozhatunk az R[-1]C[2] kóddal.

Az alábbi négy sor ugyanazt fogja eredményezni, azaz a **B1** cellába az **=2*A1** képletet írja.

```
Cells(1, 2).Formula = "=2*A1"
Cells(1, 2) = "=2*A1"
Cells(1, 2).FormulaR1C1 = "=2*RC[-1]"
Cells(1, 2) = "=2*RC[-1]"
```

Az itt következő négy sor a **B1** cellába az **=2*A1** képletet írja, azaz annyiban különbözik az előzőtől, hogy itt a hivatkozás abszolút.

```
Cells(1, 2).Formula = "=2*$A$1"
Cells(1, 2) = "=2*$A$1"
Cells(1, 2).FormulaR1C1 = "=2*R1C1"
Cells(1, 2) = "=2*R1C1"
```

Beépített Excel függvény

Ha nem egyszerű képletet írunk, hanem az Excel valamelyik **beépített függvény**ét szeretnénk használni, akkor már nehezebb dolgunk van. Ha a magyar Excelben megszokott módon a fent mutatott cellahivatkozások valamelyikével írjuk a függvényt, akkor azt jól írja ugyan be a cellába, de mégsem értékeli ki, hanem hibaüzenetet küld (**#NÉV**). Ha ezek után **szerkesztő üzemmódb**a megyünk a cellába, majd minden változtatás nélkül Enter-t, vagy pipát nyomunk a függvény kiértékelődik. Lássuk be ez tömeges függvényhasználat esetén munkaigényes és nem is túl elegáns megoldás.

Egyből jó eredményt kapunk, ha az **angol** függvényneveket és paramétermegadási sorrendet használjuk. Ehhez azonban elkerülhetetlen az angol függvénynevek ismerete. Ehhez segítséget nyújtunk a könyv végén található mellékletben. Nagyon kell vigyáznunk, mert a képletben

az argumentumokat elválasztó **;** (**pontosvessző**) helyett mind a magyar, mind az angol függvénynevek esetén **,** (**vessző**)-t kell használni. A tartomány kezdő és végpontja között a **:** (kettőspont) változatlan.

Helyes függvény-megadások

Az alábbiakban látható néhány példa (a teljesség igénye nélkül) függvények helyes megadására.

- `Cells(1, 2) = "=FAKT(A1)"` – Faktoriális függvény magyarul
- `Cells(1, 2).Formula = "=FACT(A1)"` – Faktoriális függvény angolul
- `Cells(1, 2) = "=HATVÁNY(A1,2)"` – Hatvány függvény magyarul
- `Cells(1, 2) = "=POWER(R1C1,2)"` – Hatvány függvény angolul

10.3.2 Feladatok

15. Feladat

Készítsünk makrót, amely az aktuális munkalap B9-es cellájába az „Ezaz!” szöveget írja!

16. Feladat

Készítsünk makrót, amely az A1:D2 tartomány háttérszínét pirosra, betűszínét fehérre, betűtípusát 12-es Times New Roman dőltbetűsre állítja!

17. Feladat

Készítsünk makrót, amely a C oszlopot sárgára, a 4.-9. sorokat kékre, a metszetet pedig zöldre színezi!

18. Feladat

Készítsünk makrót, amely a Munka2 munkalap A1-es cellájába írt számnak megfelelőre színezi az aktuális munkalap C1-es celláját!

Megoldás:

```
Sub masiklap()
Cells(1, 3).Interior.ColorIndex = Sheets(2).Cells(1,
1)
End Sub
```

19. Feladat

Készítsünk makrót, amely az egész munkalap formázását visszaállítja automatikusra (0-s háttérszín, 0-s betűszín, Arial betűtípus 10-es betűméret, nem dőlt, nem félkövér)!

Megoldás:

```
Sub visszaallitas()
With Cells
    .Interior.ColorIndex = 0 'automatikus háttérszín
    .Font.ColorIndex = 0 'automatikus betűszín
    .Font.Name = "Arial" 'betűtípus
    .Font.Size = 10 'betűméret
    .Font.Bold = False 'nem félkövér
    .Font.Italic = False 'nem dőlt
End With
End Sub
```

20. Feladat

Készítsünk makrót, amely a kijelölt cellákat sárgára, az aktív cellát pirosra színezi!

21. Feladat

Készítsünk makrót, amely a kiválasztott objektum háttérszínét sárgára állítja. Próbáljuk ki különböző cellatartományok kijelölésével, illetve készítsünk egy diagrammot, és annak elemeit kijelölve is futtassuk le!

22. Feladat

Nyissuk meg a bevetelek2.xls munkafüzetet! Készítsünk makrót, amely az aktuális cella felett lévő 4 cella szórását számítja ki és futtassuk le minden hónapra! (A makrót értelem szerűen csak a 8. sor celláiban futassuk!)

23. Feladat +

Nyissuk meg a nevek.xls munkafüzetet! Készítsünk makrót, amely a kijelölt cellákba a tőlük balra található nevekből kódokat készít! A kód úgy áll össze, hogy a vezetéknév első két karaktere után a keresztnév első két karaktere következik, mindezt csupa nagy betűkkel, tehát például Kiss Ágoston kódja KIÁG lesz. (Szövegfüggvények kombinálására lesz szükség.)

10.3.3 Mit tanultunk meg

– A makróírás nyitó és záró kulcsszava
– Az Excel celláira való hivatkozás módja
– A cellák tulajdonságai
– Képletek írása

10.4 Programozási struktúrák

Egy tevékenységsorozat (program) hívása során nem csak utasítások egymás utáni végrehajtását szeretnénk, hanem szükség lehet **ismétlés**ekre, vagy valamely feltétel szerinti **elágazás**ra. (Ilyennel már találkoztunk az Excel HA függvényénél.) Ilyenkor programjainkban az utasítások nem sorrendben egymás után hajtódnak végre (**szekvencia**), hanem használnunk kell a **ciklus**okat és az **elágazás**okat.

Elágazás: feltételhez kötjük, hogy melyik utasítások hajtódjanak végre. Ennek szintaxisa Visual Basic-ben: Elágazás

```
If feltétel Then
      utasítások, amelyek akkor futnak le,
      ha a feltétel igaz
Else
      utasítások, amelyek akkor futnak le,
      ha a feltétel hamis
End If
```

Ebből az **If** és az **End If** kötelező, az **Else** ág elhagyható, ilyenkor ha a feltétel nem teljesül, akkor az End If utáni sorra ugrik a program.

Ciklus: ugyanazt az utasítássort szeretnénk többször megismételtetni a programmal. A megfelelő kulcsszavak közé írt, ismétlendő utasítássort **ciklusmag**nak nevezzük. Ciklus

Iterációs ciklus

Ciklus írására több lehetőségünk van. A legegyszerűbb, ha előre tudjuk, **pontosan hányszor** szeretnénk futtatni az utasítássort. Ekkor használjuk a **leszámláló (iterációs) ciklus**t. Ennek szintaxisa Visual Basic-ben:

```
For ciklusváltozó = kezdőérték To végérték Step_
lépésköz
      ciklusmag
Next
```

A **ciklusváltozó** bármilyen számváltozó lehet (leggyakrabban az i, j, k betűk valamelyikét használjuk). Első lefutáskor a változó a **kezdőérték**et veszi fel, aztán minden újabb lefutáskor **lépésköz**zel változik az értéke, amíg **meg nem haladja a végértéket**. Amikor a változó értéke a végértéknél nagyobb, akkor a ciklusmag már nem fut le. (Ha nem adtunk meg lépésközt, akkor annak értéke: +1).

Előfordul, hogy nem tudjuk előre, hányszor kell lefutnia az utasítássornak, hanem egy **feltételhez** akarjuk **kötni** a ciklus végét. Ilyenkor használjuk a **feltételes ciklus**ok valamelyikét. Visual Basicben összesen 4 féle feltételes ciklus létezik, ezek közül kettőt fogunk megismerni. A ciklus elejét a **Do**, végét a **Loop** kulcsszó jelzi.

Elöltesztelős feltételes ciklus

Elöltesztelős ciklusnál a feltételt a ciklus elején a **Do While** kulcsszavak után adjuk meg. Ebben az esetben a **feltétel** teljesülése estén történik az ismétlés. (Előfordulhat, hogy már az első lefutáskor sem teljesül a feltétel, ilyenkor **egyszer sem fut le** a ciklus belsejében lévő utasítássorozat.)

Hátultesztelős feltételes ciklus

Hátultesztelős ciklusnál a feltételt a ciklus végén a **Loop Until** kulcsszavak után adjuk meg. Ebben az esetben mindaddig ismétlődik a ciklsumag, amíg a **feltétel hamis**. A hátultesztelős ciklus mindig **lefut legalább egyszer**. A lehetséges szintaxisok:

```
Do While feltétel
      ciklusmag
Loop
```

```
Do
      ciklusmag
Loop Until feltétel
```

Ciklusok átírása

A leszámláló ciklus átírható feltételes ciklussá. Ez esetben létre kell hoznunk egy változót a ciklus előtt, és **kezdőérték**et kell neki adnunk, majd a ciklusmagban a változó értékének megfelelő növekedését, vagy csökkenését nekünk kell biztosítanunk. A feltételben azt kell megfogalmazni, hogy ha a változó meghaladta a végértéket, akkor legyen vége az ismétlésnek.

Az alábbiakban egy példát látunk egy számlálós ciklus átírására a 2 feltételes ciklusba. Az utasítás az ossz változóba összegzi a számokat 1-től 10-ig.

Ciklusok átírása, példa

```
ossz = 0
For i = 1 To 10
  ossz = ossz + i
Next
```

```
ossz = 0
i = 1
Do While i <= 10
  ossz = ossz + i
  i = i + 1
Loop
```

```
ossz = 0
i = 1
Do
  ossz = ossz + i
  i = i + 1
Loop Until i > 10
```

Általában úgy érdemes a ciklusokat használni, hogy csak akkor használunk **feltételes ciklus**t, ha **nem tudjuk előre**, hogy hányszor kell lefutnia az utasítássornak.

Cella-tartomány

Ha Excelben cellák egy **tartomány**ára ugyanazt a műveletet akarjuk végrehajtatni, akkor két lehetőségünk van: vagy **ciklus**sal végigmegyünk a tartomány minden egyes celláján és a ciklusmagban leírjuk az utasítást, ami mindig a megfelelő cellára vonatkozik, vagy ciklus nélkül a megfelelő **cellatartomány**ra hivatkozva adjuk ki az utasítást. A két megoldás azonban nem minden esetben egyenértékű.

A két következő programkód ugyanazt eredményezi, mindkét esetben az A1:A10 cellatartomány háttere kék lesz:

```
For i = 1 To 10
Cells(i,1).Interior.Color = vbBlue
Next
```

```
Range(Cells(1,1),Cells(10,1)).Interior.Color = _
vbBlue
```

A következő példában képzeljük el, hogy a C3 cella tartalma: **= VÉL()** Ekkor az alábbi programkódok eredménye **különböző** lesz:

```
For i = 1 To 10
Cells(i, 2) = Cells(3, 3)
Next
```

```
Range(Cells(1, 1), Cells(10, 1)) = Cells(3, 3)
```

Az első esetben minden egyes cella külön töltődik fel és így mindegyik után változik a C3 cella tartalma is. Tehát a B oszlop 10 cellájában különböző számok lesznek. A második esetben egyszerre töltjük fel a 10 cellát az A oszlopban, tehát mindegyik cella ugyanazt a számot fogja tartalmazni.

10.4.1 Feladatok

24. Feladat

Készítsünk makrót, amely az A oszlopot feltölti az egész számokkal 0-tól 56-ig! A B oszlop celláinak háttérszíne legyen az A oszlopban lévő számoknak megfelelő!

25. Feladat

Készítsünk makrót, amely feltölti az A1:J10 tartományt a 10*10-es szorzótáblával! A 3-mal osztható számok hátterét színezzük sárgára, az egy maradékot adókét kékre, a 2 maradékot adókét zöldre!

26. Feladat (+)

Oldjuk meg az előző feladatot a színkódok segítségével úgy, hogy ne használjunk benne elágazást!

27. Feladat

Nyissuk meg a tozsde.xls munkafüzetet! A tábla részvények alakulását mutatja. Színezzük a növekedők sorának hátterét zöldre, a csökkenőkét pirosra, a stagnálókét kékre!

28. Feladat

Nyissuk meg a trend.xls munkafüzetet! A táblázatban a magyar ipari termelés volumenindexének változását látjuk 2004. januártól 2005. áprilisig. 2004 februártól kezdve színezzük az egyes időszakok sorait a

következőképpen: ha az index magasabb, mint az előző hónapban, akkor legyen a cellák háttérszíne zöld, ha pedig alacsonyabb, akkor piros!

29. Feladat

Nyissuk meg a valahol.xls munkafüzetet! Készítsünk makrót, amely megkeresi az „oszlop” munkalap A oszlopában található adatsort és beszínezi a hátterét sárgára! Az oszlopban az adatsor előtt szereplő cellák háttere legyen fekete! A makró indításakor mindig tűnjön el minden háttérszínezés! Úgy írjuk meg a programot, hogy bármilyen munkalapon lefuttatható lehessen, függetlenül attól, hogy hol van az adatsor és mekkora!

30. Feladat

Nyissuk meg a valahol.xls munkafüzetet! Készítsünk makrót, amely megkeresi a „táblázat” munkalapon található A1 cellától induló táblázat jobb alsó sarkát és az egész táblázat hátterét beszínezi sárgára! A makró indításakor mindig tűnjön el minden háttérszínezés! Úgy írjuk meg a programot, hogy bármilyen munkalapon lefuttatható lehessen, függetlenül attól, hogy mekkora a táblázat!

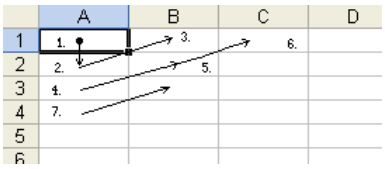
31. Feladat (+)

Nyissuk meg a valahol.xls munkafüzetet! Készítsünk makrót, amely megkeresi a „valahol” munkalapon található táblázatot és beszínezi a hátterét sárgára. A sorban és oszlopban a táblázat előtt szereplő cellák háttere legyen fekete! A makró indításakor mindig tűnjön el minden háttérszínezés! Úgy írjuk meg a programot, hogy bármilyen munkalapon lefuttatható lehessen, függetlenül attól, hogy hol van a táblázat és mekkora!

Megoldásrészlet:

```
Sub keres_tablazat()

Cells.Interior.ColorIndex = 0      'háttérszínezés
                                   'visszaállítása

i = 1                              'A1 cellától
j = 1                              'indulunk

Do While Cells(i, j) = ""          'Keressük a
    i = i + 1                      'táblázat bal
                                   'felső sarkát
    Do While Cells(i, j) = "" And i > 1
        i = i - 1
        j = j + 1
    Loop
    If Cells(i, j) = "" Then
        i = j
        j = 1
    End If
Loop
tsork = i
toszlk = j
Do While Cells(i, toszlk) <> ""    'A táblázat bal
    i = i + 1                      'felső sarkától
Loop                               'kezdve
tsorv = i - 1                      'keressük a
Do While Cells(tsork, j) <> ""     'jobb alsó
    j = j + 1                      'sarkot
Loop
toszlv = j - 1

For i = tsork To tsorv             'A táblázat
For j = toszlk To toszlv           'színezése
    Cells(i, j).Interior.Color = vbYellow
Next
Next

End Sub
```

A bal felső sarok megkeresésekor a következő sorrendben vizsgáltuk végig a cellákat, amíg meg nem találtuk az első cellát, ami már nem üres:



32. Feladat

Bizonyos kimutatásokat is készítő számviteli programok úgy exportálják Excelbe az eredményt, hogy egy összetartozó adathalmaz két sorban van és a második sor nem tartalmaz plusz információt az elsőhöz képest. Egy ilyen kimutatást láthatunk a kimutatas.xls munkafüzetben. Készítsünk makrót, amely a felesleges sorokat kitörli! Úgy készítsük el a makrót, hogy minden további hasonló szerkezetű, de eltérő méretű kimutatásra is használható legyen! Azt, hogy hogyan lehet egy sort kitörölni makrórögzítéssel nézzük meg!

10.4.2 Mit tanultunk meg

– Az elágazás szintaktikája
– A számlálós ciklus szintaktikája
– A feltételes ciklusok szintaktikája

10.5 Néhány hasznos VB függvény

A program futása során **kapcsolat**ot tarthat a felhasználóval. Kérhet tőle adatokat az **InputBox** függvény segítségével és üzenhet neki a **MsgBox** függvény segítségével.

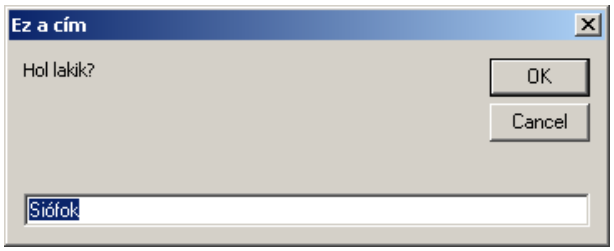
Az InputBox szintaxisa: InputBox

```
valtozo = InputBox(szöveg, cím, alapértelmezés)
```

Az **InputBox** egy kis ablak, ami megjelenik a képernyőn. Az ablakban lévő **beviteli mező**be írhat a felhasználó. A program a változóban tárolja **szövegként**, amit a felhasználó beírt. A zárójelben felsorolt paraméterek közül a szöveget kötelező megadni. Ez fog megjelenni az ablakban a beviteli mező fölött. A cím lesz az ablak fejlécében, az **alapértelmezés**

pedig a beviteli mezőben, de át lehet írni. A két gomb Ok és Cancel. Az **Ok** gomb megnyomásával bekerül a beviteli mező tartalma a változóba, a **Cancel** gomb megnyomásával a **változó tartalma üres lesz**.

InputBox ablak



MsgBox

A MsgBox szintaxisa:

```
MsgBox szöveg, gombok, cím
```

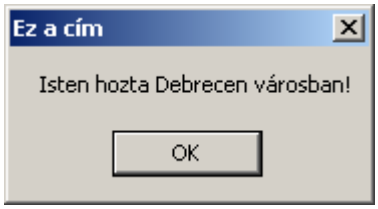
A MsgBox egy kis ablak, ami egy üzenettel jelenik meg a képernyőn. A zárójelben felsorolt paraméterek közül a szöveget kötelező megadni. Ez fog megjelenni az ablakban. A cím lesz az ablak fejlécében. A gomboknál azt lehet megadni, hogy milyen gombok jelenjenek meg az ablakban. Ha nem adunk meg semmit, vagy 0-t írunk, akkor csak Ok gomb lesz. *(Vigyázzunk arra, hogy az InputBoxnál kell zárójel, míg a MsgBox itt bemutatott használatánál nem!)*

Szövegek kiírásakor szükségünk lehet arra, hogy szó szerinti szövegeket és változótartalmakat együtt jelenítsünk meg. A szó szerint megjelenítendő szöveget **idézőjel**be kell tenni, a változó nevét idézőjel nélkül kell leírni. Ha össze akarjuk fűzni őket, akkor az **&** jelet kell használnunk. Pl. ha egy varos nevű változóban tároljuk a város nevét, és szeretnénk egy üdvözletet küldeni, hogy „Isten hozta … városban!”, akkor azt a következőképpen tehetjük meg:

```
MsgBox „Isten hozta „ & varos & „ városban!”
```

Azért tettünk az idézőjelen belülre szóközt, hogy ne írja egybe a hozta, ill. a városban szót a város nevével.

MsgBox ablak



Rnd véletlen függvény

A **véletlen** számok generálására szolgáló függvény az **Rnd**. Ez a függvény így argumentum nélkül használható és egy véletlen számot ad eredményül **0 és 1 félig zárt intervallumban**, 0 lehet, 1 nem. Ennek a függvénynek a transzformálásával lehet megadni, hogy milyen számokat akarunk generálni. Például –10 és +10 közötti páros számokat szeretnénk, úgy hogy a –10 és a +10 is közte legyen:

```
(Int(Rnd*11)-5)*2
```

Az Int() függvény a szám egészrészét veszi. Pontos tárgyalására a változókról szóló fejezetben kerül sor.

Ahhoz, hogy az Rnd függvény valóban véletlen számokat generáljon a függvény használata előtt be kell írnunk a **Randomize** kulcsszót a programunkba. *(Mielőtt a Randomize kulcsszót beírnánk, teszteljük le a programunkat! A Randomize nélkül az Rnd függvény minden programfutáskor ugyanazt a véletlen számsort fogja generálni.)*

10.5.1 Feladatok

33. Feladat

Készítsünk makrót, amely InputBoxban bekéri a felhasználó nevét, aztán MsgBoxban köszönti a felhasználót a saját nevén!

34. Feladat

Egészítsük ki az előző feladatot úgy, hogy ismételje meg a név bekérését, ha a felhasználó a Cancel gombot nyomta meg!

Megoldás

```
Sub udvozlet()
Do
    nev = InputBox("Hogy hívnak?", "Kérdés")
Loop Until nev <> Empty
MsgBox "Szerbusz kedves " & nev & "!", 0, "Üdvözlet"
End Sub
```

35. Feladat

Készítsünk makrót, amely InputBoxban számokat kér be és a számokat egymás alá beírja az A oszlopba! Mindaddig folytassa a bekérést, amíg a felhasználó a Cancel gombot meg nem nyomja!

36. Feladat

Készítsünk makrót, amely –6 és +7 közötti véletlen egész számokkal feltölti az A oszlopot a 2.-tól a 18. sorig, majd a páros számok betűszínét kékre, a páratlanok háttérszínét pedig sárgára változtatja! Vigyázzunk arra, hogy ha többször egymás után le akarjuk futtatni a programot, akkor az előző formázásokat meg kell szűntetnünk a program elején!

Megoldás:

```
Sub veletlen()
Cells.Interior.ColorIndex = 0
Cells.Font.ColorIndex = 0
Randomize
For i = 2 To 18
    Cells(i, 1) = Int(Rnd * 14 - 6)
    If Cells(i, 1) Mod 2 = 0 Then
        Cells(i, 1).Font.Color = vbBlue
    Else
        Cells(i, 1).Interior.Color = vbYellow
    End If
Next
End Sub
```

37. Feladat

Készítsünk makrót, amely 10 és 30 közötti véletlen egész számokkal feltölti az A1:G10 cellatartományt, majd a páros számok hátterét sárgára színezi, a páratlanoknak pedig a betűtípusát megváltoztatja kék félkövérre, majd jelenjen meg egy MsgBox, ami közli, hogy hány darab páros számunk van! Vigyázzunk arra, hogy ha többször egymás után le akarjuk futtatni a programot, akkor az előző formázásokat meg kell szüntetnünk a program elején!

38. Feladat

Készítsünk makrót, amely –20 és +20 közötti véletlen páros számokkal feltölti az E5:K20 cellatartományt, majd a negatív számok hátterét sárgára, a nem negatívakét kékre színezi!

39. Feladat

Egészítsük ki az előző feladatot úgy, hogy a nullák háttere legyen zöld!

40. Feladat (+)

Készítsünk makrót, amely InputBoxban bekér egy alsó, egy felső határt, majd egy lépésközt! Generáljon az A1:J10 cellatartományba a megadott feltételeknek megfelelő véletlen egész számokat!

41. Feladat

Nyissuk meg az urlap.xls munkafüzetet! A munkafüzethez készítsünk új makrót! A makró ellenőrizze az űrlapot a következő szempontok alapján:

- Az irányítószám 1000 és 9999 közé essen
- A születés dátuma 1900 és 2004 közé essen
- Az összegek és különbségek értelemszerűen egyezzenek

Hiba esetén a kitöltő üzenetablakban kapjon figyelmeztetést hibájáról!

42. Feladat (+)

Készítsünk makrót, amely sakktáblát készít a következő feltételekkel!

- A sakktábla első kockája az A1-es cella legyen!
- Az A1-es cella sötét legyen!
- A sakktábla szabályainak megfelelően váltakoznak a sötét és világos cellák!
- A cellák alakját ne változtassuk!
- A sakktábla méretét InputBoxban adhassa meg a felhasználó (ugyanannyi sor és oszlop legyen)!
- Próbáljuk meg a lehető legrövidebben megírni a makrót!

43. Feladat

A 23. feladatban a nevek.xls munkafüzet neveiből képeztünk kódot. Ez a kódolási módszer viszont nem teljesen megfelelő, mert előfordulhatnak azonos kódú egyének. (A konkrét feladatban például a SZAN és a SZIS kódok duplán szerepelnek.) Egészítsük ki a feladatot úgy, hogy a 4 betűből álló kódot egészítsük ki egy 100 és 999 közé eső véletlen számmal. (Elvi lehetőség így is marad, hogy kódismétlődés legyen, de sokkal kisebb az esélye. Ki lehet egészíteni a feladatot úgy, hogy kódismétlődés esetén másik számot generáljon.)

10.5.2 Mit tanultunk meg

- Az InputBox függvény
- A MsgBox függvény
- Szövegösszefűzés
- Az Rnd függvény

10.6 Változók

Programjaink során gyakran használunk **változók**at.

A következő táblázatban felsoroljuk a leggyakrabban használt változótípusokat Visual Basicben:

Változó-típusok

típusnév	**mit tárol**
String	szöveg
Integer	egész szám (32768-ig)
Single, Double	valós szám
Long	hosszú egész
Date	dátum
Boolean	logikai(igaz/hamis)
Variant	általános

Változó-deklarálás

Visual Basicben **nem kötelező** a változók **deklarálás**a (definiálása). A változók **Variant** típussal jönnek létre első használatkor. Ha a változót definiáltuk valamilyen típusra, akkor a megadott típusnak megfelelő műveleteket hajthatunk végre vele. Ez gondokat is okozhat, ha nem megfelelő tartalmat adunk meg. Változót a **Dim** kulcsszó segítségével deklarálhatunk:

```
Dim valtozonev As valtozotipus
```

Több, azonos típusú változót létrehozhatunk úgy is, hogy a Dim és az As kulcsszavak között vesszővel elválasztva felsoroljuk őket.

```
Dim i,alma As integer
```

Általános adattípus Variant

Ha a változó **Variant** típusú, akkor viselkedése attól függ, hogy milyen módon adunk neki értéket. Programon belüli értékadásnál létrejöhet valós, dátum, boolean és szöveg típusú tartalom is. Ha azonban InputBox függvénnyel, vagy szöveg típusú vezérlővel (például beviteli mező, vagy kombi panel) adunk értéket, akkor az **szöveg**ként fog viselkedni akkor is, ha alakilag más adattípus is lehetne.

Automatikus típus-konverzió

A VBA-ban elég gyakran működik az **automatikus típuskonverzió**, azaz ha például egy szöveg típusú számot tartalmazó változóval számra jellemző műveletet végzünk (szorzás, osztás, stb.), akkor az számként fog viselkedni. Van azonban néhány **kivétel**. A két legfontosabb az **összeadás** és az **összehasonlítás.**

Összeadás

Összeadás: két szám összege a matematikai szabályoknak megfelelően képződik, két szöveg összege megfelel a két szöveg összefűzésének, egy szám és egy szöveg összeadása viszont bizonyos esetekben hibát okoz: ha a szöveg tekinthető számnak is, akkor összeadás, ha nem, akkor hibaüzenet az eredmény.

A programkód	Az eredmény

`Dim a As Double`
`Dim b As Double`
`a=10`
`b=3.2`
`Cells(1,1)=a+b`
	Az A1 cella tartalma: 13,2 számként

`Dim a As String`
`Dim b As String`
`a=10`
`b=3.2`
`Cells(1,1)=a+b`
	Az A1 cella tartalma: 103,2 szövegként

`Dim a As Double`
`Dim b As String`
`a=10`
`b=3.2`
`Cells(1,1)=a+b`
	Az A1 cella tartalma: 13,2 számként

`Dim a As Double`
`Dim b As String`
`a=10`
`b=”alma”`
`Cells(1,1)=a+b`
	’Run-time error 13’ nem megadott típust kapott a rendszer

Összeha-sonlítás

Összehasonlítás: két számot a matematika szabályai szerint hasonlít össze a program, ha azonban az összehasonlításban legalább az egyik elem szöveg, akkor már az (angol) **abc** határozza meg a sorrendet. Ebben az összehasonlításban tehát 100 < 85, ha valamelyik szöveg típusú, ugyanis az abc-ben karakterenként történik az összehasonlítás és 1 < 8.

Néha az is okozhat problémát, ha meghatározzuk a változó típusát. Nézzünk mi lesz a következő feladat eredménye különböző bemenő adatok hatására!

Egész számot várok a kért cellába

```
Sub beker()
Dim szam As Integer        'egész típusú változó
                           'deklarálása
szam = InputBox("Kérem a születési évedet!")
Cells(3, 4) = szam
End Sub
```

Beírt érték	Eredmény
1983	1983
1975,5	1976
alma	'Run-time error 13' nem megadott típust kapott a rendszer

Típus ellenőrzése

A problémán javíthatunk. Általános típust definiálunk (vagy elhagyjuk a definíciót), ez a **Variant**. A változó tartalmát **ellenőrizhetjük**, így csak a kívánt típust fogadjuk el. Először csak hibaüzenetként megjelenítjük, a következő programrészletben megmutatjuk, hogy hogyan javíthatjuk. A rosszul megadott típus így ellenőrizhető.

Az **IsNumeric**(*Változó*) függvény eredménye igaz, vagy hamis. Ha a változó tartalma numerikus érték eredménynek **True**-t (igazat) kapok. Ezt használhatjuk ellenőrzésre.

```
Sub bekervariant()
Dim szam As Variant
szam = InputBox("Kérem a születési évedet!")
If Not IsNumeric(szam) Then    'szam változó
                               'tartalmát
                               'vizsgáljuk
    MsgBox "Ismételje meg, nem számot adott!"
End If
Cells(3, 4) = szam
End Sub
```

A fenti feladatot ciklusba szervezve, mindaddig bekéri az adatot, míg a megfelelő típusú értéket nem adjuk meg. Utána lehet akár nagyságrendet is vizsgálni.

```
Sub bekerciklus()
Dim szam As Variant
szam = InputBox("Kérem a születési évedet!")
Do While IsNumeric(szam) = False
    MsgBox ("Ismételje meg, nem számot adott")
    szam = InputBox("Kérem a születési évedet!")
Loop
Cells(3, 4) = szam
End Sub
```

Ha egyes műveleteknél a típuskonverzió nem automatikus, akkor a VBA beépített **konvertáló függvény**eivel tudjuk az értékek típusát változtatni:

Konvertáló függvények

Val(*érték*)	valós számmá konvertál *(Az első karaktertől kezdve a számkaraktereket veszi, ha egyet sem talál, akkor az eredmény 0 lesz. Problémát okozhat, hogy a VBA-ban tizedes pont van, míg az Excelben tizedes vessző. Ha tizedesvesszőt talál, akkor azt is nem szám karakternek tekinti.)*
Str(*érték*)	szöveggé konvertál *(Csak nem szöveg típusú adatokat konvertál szöveggé.)*
Int(*érték*)	egész számmá konvertál *(Csak számjegyeket tartalmazó értéket tud konvertálni és a számnak csak az egészre kerekített értékét adja vissza eredményül.)*

Mint a legtöbb programnyelvben, Visual Basic-ben is lehet tömbváltozókat is deklarálni. A tömbváltozóknak az a jellemzője, hogy egy névvel és egy indexszámmal azonosítjuk őket. Az azonos nevű, azonos típusú és különböző indexszámú változók alkotnak egy tömböt. Visual Basic-ben az indexszámot ()-be kell írni. Deklarálásnál meg kell adnunk a To kulcsszó segítségével az indexek kezdő és záró értékét, vagy csak a záróértéket, ilyenkor az indexek 0-tól kezdődnek és egyesével emelkednek. Tehát az alábbi két sor ugyanazt eredményezi: létrejön egy 6 elemű, szöveg típusú tömb.

```
Dim tomb(5) As String
Dim tomb(0 To 5) As String
```

Az alábbi kis példában definiálunk egy 5 elemű szöveg tömböt. A tömb minden elemébe beolvasunk egy nevet és a neveket kiírjuk egymás alá az A oszlop celláiba.

```
Dim nev(1 To 5) As String
For i = 1 To 5
    nev(i) = InputBox(i & ". név:")
    Cells(i, 1) = nev(i)
Next
```

Lehetőség van többdimenziós tömváltozók definiálására is. Ilyenkor a dimenziókat vesszővel választjuk el egymástól.

10.6.1 Feladatok

44. Feladat

Készítsünk makrót, ami induláskor egy InputBoxban bekér egy számot! Ha nem számot adunk meg, akkor MsgBoxban küldjön figyelmeztetést és kérje be a számot még egyszer! Ha valóban szám, akkor az A1-es cellába írjuk ki számként, az A2-be pedig szövegként! Figyeljük meg, hogy a tizedesvessző és a tizedes pont eltérő használata az Excel-ben és a Visual Basic-ben milyen problémát okoz!

45. Feladat

Készítsünk makrót, ami induláskor egy InputBoxban bekér egy számot, a számnak megfelelő méretű szorzótáblát készít az A1-es cellától kezdődően, majd egy újabb InputBoxban bekér egy újabb számot! Az újonnan beadott számnál nagyobb számok cellájának háttérszínét színezze pirosra!

46. Feladat

Készítsünk makrót, ami egy 20 elemű tömbváltozóba generál 20 véletlen 100-nál kisebb pozitív egész számot! Kérjünk be inputboxban egy számot! A bekért számnál kisebb tömbelemek sorszámát írassuk ki egymás alá az A oszlopban, magukat a számokat pedig a B oszlopban!

10.6.2 Mit tanultunk meg

- A változók használata
- A változók deklarálása
- A változók típusai
- A változók típuskonverziója

– Tömbváltozók

10.7 Algoritmusok

Algoritmus

Az **algoritmus** egy feladat megoldására szolgáló egyértelmű szabályokkal követhető lépések (utasítások) sorozata. (A kis példaprogramokban feltételezzük, hogy a munkalapon az A1:A10 cellatartomány számokat tartalmaz és erre a tartományra vonatkoznak a feladatok.)

Minimum-keresés

Minimumkeresés: első lépésként feltesszük, hogy a halmaz első tagja a legkisebb, és megjegyezzük az értékét. Végigmegyünk a halmazon és megvizsgáljuk, hogy találunk-e az eddig feltételezettnél kisebbet. Ha találtunk, akkor ez az eddigi legkisebb és megjegyezzük az értékét. A halmaz végére biztosak lehetünk abban, hogy a feltételezett legkisebb valóban a halmazban előforduló legkisebb elem.

```
minimum = Cells(1, 1)
For i = 2 To 10
    If Cells(i, 1) < minimum Then
        minimum = Cells(i, 1)
    End If
Next
```

Természetesen a **maximumkeresés** is hasonló módon működik.

Előfordulhat, hogy nem a legkisebb szám értékére vagyunk kíváncsiak, hanem arra, hogy a **halmaz melyik eleme** a legkisebb. Ilyenkor a feltételezett legkisebbnek nem az értékét, hanem a helyét jegyezzük meg.

```
hely = 1
For i = 2 To 10
    If Cells(i, 1) < cells(hely,1) Then
        hely = i
    End If
Next
```

Összegzés

Összegzés: egy változóban fogjuk tárolni az összeget. Ez az érték kezdetben 0. Végigmegyünk a halmazon és a halmaz minden elemét hozzáadjuk az eddigi összeghez.

```
osszeg = 0
For i = 1 To 10
    osszeg = osszeg + Cells(i, 1)
Next
```

Keresés

Keresés: egy adott értékről el szeretnénk dönteni, hogy az adott halmazban megtalálható-e és ha igen, akkor hányadik helyen. Végigmegyünk a halmazon addig, amíg meg nem találjuk a keresett elemet, vagy amíg el nem értük az utolsó elemet. Ha megtaláltuk az elemet, akkor kiírjuk, hogy hányadik sornál álltunk meg, ha végigmentünk az egész halmazon, de nem találtuk meg az elemet, akkor kiírjuk, hogy „nincs”. (A példában a keresett nevű változó tárolja a keresett elemet.)

```
j = 0
Do
    j = j + 1
Loop Until Cells(j, 1) = keresett Or j = 11
If j < 11 Then
    MsgBox "Van! A(z)" & Str(j) & ". elem."
Else
    MsgBox "Nincs"
End If
```

Sorba rendezés maximum-kiválasztással

Sorba rendezés maximumkiválasztással: Felhasználva a maximumkeresés algoritmusát megkeressük a halmaz legnagyobb elemét és kicseréljük a halmaz utolsó elemével. Ezt mindig eggyel rövidebb halmazon ismételjük, amíg az utolsó elem is a helyére nem kerül. Két elemet egy **segédváltozó** segítségével tudunk kicserélni egymással: először az egyik elemet kitesszük a segédváltozóba, aztán a másik elemet átírjuk az egyik helyére, majd a segédváltozóból visszaírjuk az egyik elemet a másik helyére.

```
For i = 10 To 2 Step -1
    hely = 1
    For j = 2 To i
        If Cells(j, 1) > Cells(hely, 1) Then
            hely = j
        End If
    Next j
    s = Cells(hely, 1)
    Cells(hely, 1) = Cells(i, 1)
    Cells(i, 1) = s
Next i
```

Sorba rendezés buborék módszerrel

Sorba rendezés buborék módszerrel: végigmegyünk a halmazon és páronként összehasonlítjuk az elemeket. Ha a két elem sorrendje megfelelő, akkor úgy hagyjuk őket, ha nem megcseréljük őket. Az összehasonlítgatást többször meg kell tennünk, hogy a végleges sorrend kialakuljon. Legrosszabb esetben elemszám – 1-szer kell végigmennünk a halmaz elemein páronként összehasonlítva őket. Egy végighaladás során legalább egy elem a helyére kerül, így minden végighaladásnál egyel

rövidebb halmazt kell vizsgálnunk. Két elemet egy **segédváltozó** segítségével tudunk kicserélni egymással: először az egyik elemet kitesszük a segédváltozóba, aztán a másik elemet átírjuk az egyik helyére, majd a segédváltozóból visszaírjuk az egyik elemet a másik helyére.

```
For i = 10 To 2 Step -1
    For j = 1 To i - 1
        If Cells(j, 1) > Cells(j + 1, 1) Then
            s = Cells(j, 1)
            Cells(j, 1) = Cells(j + 1, 1)
            Cells(j + 1, 1) = s
        End If
    Next j
Next i
```

A Visual Basic nem csak számokat tud összehasonlítani egymással, hanem karaktereket is az ASCII kódjuk alapján, így névsorba rendezés is lehetséges az angol ábécé keretein belül. *(A magyar ábécé különleges karakterei az ASCII kódjuk alapján nincsenek a „helyükön”.)*

10.7.1 Feladatok

47. Feladat

Nyissuk meg az adatbazis.xls munkafüzetet! Készítsünk makrót amely a G1 cellától kezdődően kiírja a legfiatalabb diák(ok) nevét és átlagát!

48. Feladat

Nyissuk meg az adatbazis.xls munkafüzetet! Készítsünk makrót amely a I1 cellától kezdődően kiírja a legrosszabbul tanuló diák(ok) nevét és születési dátumát!

49. Feladat

Nyissuk meg az adatbazis.xls munkafüzetet! Másoljuk le az adatbázist a másolat munkalapra! Az adatbázis munkalapon rendezzük az adatbázist makróval, az átlag alapján növekvő sorrendbe!

50. Feladat

Módosítsa az előző feladatot úgy, hogy egy inputboxban adhassuk meg a program elején, hogy hányadik oszlop szerint szeretnénk a sorbarendezést!

51. Feladat

Készítsünk makrót, melyben egy 10 elemű változó tömbbe beolvasunk 10 nevet, majd azokat ABC sorrendben írjuk ki az A oszlop első 10 cellájába!

10.7.2 Mit tanultunk meg

- Minimum és maximumkeresés
- Összegzés, keresés
- Sorba rendezés

10.8 Objektumokhoz kapcsolódó makrók

A 10.1 fejezet végén említettük, hogy nem csak **modul**ba lehet makrót írni, hanem az egyes munkalapokhoz és a munkafüzethez is. Ha a makrókat az eddig megtanult módon, de nem modulba, hanem az egyik munkalaphoz írjuk, akkor akármelyik munkalapról indítjuk el a makrót, a cellahivatkozások mindig annak a munkalapnak a celláira fognak vonatkozni, amelyikről indítottuk őket (ha nincs bennük a munkalapot is megjelölő direkt hivatkozás).

A 9. fejezetben **vezérlők**et helyeztünk el a munkalapokon, de nem írtunk hozzájuk programot. A munkalapokhoz írt makrókkal ezt is megtehetjük.

Vezérlő a munkalapon

Nyissunk meg egy üres Excel munkafüzetet! A Munka1 munkalapra helyezzünk fel egy **parancsgomb**ot (**CommandButton**) a **Vezérlők eszköztár**ról! Menjünk át a **Visual Basic szerkesztő**felületre és válasszuk a Project Explorerben a Microsoft Excel Objects közül a Munka1 munkalapot, kattintsunk rá kétszer! A jobb oldalon megjelenő ablak tetején két legördülő listát látunk. A baloldaliban vannak a munkalaphoz kapcsolódó **objektum**aink, a jobboldaliban pedig a kiválasztott objektumhoz tartozó **események**. Ha legördítjük a baloldali listát, abban 3 elemet látunk:

– (General): zárójelben van, mert ez igazából nem objektum, ide lehet írni a globális változódeklarálásokat.
– **CommandButton1**: ez az a parancsgomb, amit az előbb felhelyeztünk. Ha lenne több vezérlőnk a munkalapon, azok mind megjelennének itt.
– **Worksheet**: maga a munkalap is egy objektum, amihez tartoznak események.

Esemény-vezérelt programok

Eddigi makróinkat mindig menüből indítottuk. A makrókat azonban indíthatja egy-egy **esemény** is. Ezeket választhatjuk ki a jobb oldali listából. A programunk indulhat pl. egy parancsgomb megnyomásakor, vagy egy cella tartalmának megváltozásakor automatikusan. Tekintsünk át néhány eseményt a munkalaphoz és a vezérlőkhöz!

A **munkalap** néhány eseménye:

A munkalap eseményei

SelectionChange	ha a munkalapon mást jelölünk ki
Change	ha bármelyik cella tartalma, vagy értéke megváltozik a munkalapon
BeforeDoubleClick	dupla kattintáskor a szerkesztő üzemmódba menetel előtt (természetesen nem a dupla kattintás előtt)
BeforeRightClick	jobb-egérgomb kattintáskor mielőtt a gyorsmenü megjelenik (természetesen nem a kattintás előtt)
Calculate	ha a munkalapon számítás történt

A **Change** és a **Calculate** eseményekkel nagyon kell vigyázni, mert ha olyan utasítássort kapcsolunk hozzájuk, ami egy újabb változást, újabb számolást okoz, akkor **végtelen ciklus**hoz juthatunk. *(A végtelen ciklus leállítása a **Ctrl+Break** billentyűkombinációval lehetséges)*

A **vezérlők** néhány eseménye (nem mindegyik vezérlőhöz kapcsolódik mindegyik esemény, a táblázat azt tartalmazza, hogy a 4 gyakori esemény közül az egyes vezérlőkhöz melyek kapcsolódhatnak):

A vezérlők eseményei

Vezérlők	**Change**	**Click**	**DblClick**	**MouseMove**

	amikor az értéke változik	kattintás	dupla kattintás	amikor az egérkurzor fölé kerül
Jelölőnégyzet	van	van	van	van
Beviteli mező	van	van	van	van
Parancsgomb	nincs	van	van	van
Választókapcsoló	van	van	van	van
Listapanel	van	van	van	van
Beviteli lista	van	van	van	van
Váltógomb	van	van	van	van
Léptetőnyíl	van	nincs	nincs	nincs
Görgetősáv	van	nincs	nincs	nincs
Felirat	nincs	van	van	van
Kép	nincs	van	van	van

10.8.1 Feladatok

52. Feladat

Helyezzünk el a Munka1 munkalapon egy parancsgombot, a feliratát változtassuk meg arra, hogy „Szia”! A parancsgomb megnyomásakor jelenjen meg egy MsgBox a következő felirattal: „Üdvözöllek a VBA világában!”!

53. Feladat

Írjunk makrót a Munka1 munkafüzethez, ami egy cellára való dupla kattintáskor beleírja a nevünket az aktuális cellába!

Megoldás:

```
Private Sub Worksheet_BeforeDoubleClick(ByVal _
Target As Range, Cancel As Boolean)
ActiveCell = "Neumann"
End Sub
```

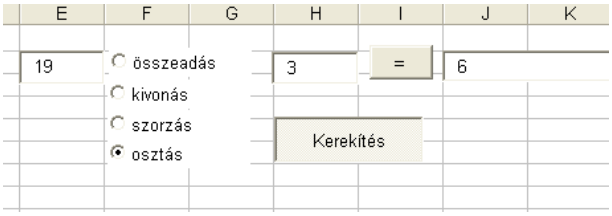
54. Feladat

Írjunk makrót, amelynek a hatására, ha rákattintunk egy cellára, annak véletlenszerűen megváltozik a háttérszíne!

55. Feladat (+)

Készítsük el a 10. feladatban látott számológépet makró írással! Az egyenlőségjel helyén legyen parancsgomb, melynek megnyomása indítja el a programot! A kerekítés gomb megnyomásakor automatikusan (az egyenlőségjel újbóli megnyomása nélkül) változzon az eredmény megjelenítése! Ha műveletet váltunk, akkor az egyenlőségjel megnyomásáig tűnjön el az eredmény!

Megoldás:



A programkód:

```
Dim eredm As Double    'Azért itt deklaráljuk,
                       'hogy mindkét eljárásból
                       'lássuk majd
```

```
Private Sub CommandButton1_Click()
Dim elso As Double     'Valósként deklaráljuk a
Dim masodik As Double  'változókat

If IsNumeric(TextBox1.Text) Then 'Vizsgáljuk,
    elso = TextBox1.Text         'hogy a megadott
Else                             'érték szám-e
    MsgBox ("Nem számot írtál be az első _
mezőbe!")
End If

If IsNumeric(TextBox2.Text) Then
    masodik = TextBox2.Text
Else
    MsgBox ("Nem számot írtál be a második _
mezőbe!")
End If

If OptionButton1.Value = True Then
    eredm = elso + masodik       'Vizsgáljuk,
End If                           'hogy melyik
```

```
                                    'opció van
                                    'kiválasztva
If OptionButton2.Value = True Then
    eredm = elso - masodik
End If
If OptionButton3.Value = True Then
    eredm = elso * masodik
End If
If OptionButton4.Value = True Then
    If masodik <> 0 Then            'Vizsgáljuk,
        eredm = elso / masodik      'hogy nem 0-val
    Else                            'akarunk-e
                                    'osztani
        MsgBox ("Nullával nem tudok osztani!")
    End If
End If
If ToggleButton1.Value = True Then
    TextBox3.Text = Int(eredm)      'Vizsgáljuk,
Else                                'hogy kerekített
                                    'értéket
                                    'várunk-e

    TextBox3.Text = Round(eredm, 4)
              'A round függvény paraméterei
              'a kerekítendő szám és az, hogy
              'hány tizedesjegyet jelenítsen meg
End If
End Sub
```

```
Private Sub OptionButton1_Click()
    TextBox3.Text = Empty
End Sub
```

```
Private Sub OptionButton2_Click()
    TextBox3.Text = Empty
End Sub
```

```
Private Sub OptionButton3_Click()
    TextBox3.Text = Empty
End Sub
```

```
Private Sub OptionButton4_Click()
    TextBox3.Text = Empty
End Sub
```

```
Private Sub ToggleButton1_Click()'Ha már megvan
If TextBox3 <> Empty Then        'az eredmény,
    If ToggleButton1.Value = True Then
        TextBox3.Text = Int(eredm)  'akkor is
    Else                         'tudjuk
        TextBox3.Text = Round(eredm, 4)
    End If                       'változtatni,
End If                           'hogy
End Sub                          'kerekítünk-e
```

56. Feladat

Nyissuk meg a jatek.xls munkafüzetet! A táblázatban rejtvényeket találunk, mindegyik megfejtése egy szám. Helyezzünk el a táblázat mellé egy parancsgombot „Ellenőrzés" felirattal! Ha a játékos megnyomja a gombot, a gép ellenőrizze, hogy a megoldásai jók-e! A helyes megoldások mellé írja ki hogy „Helyes"! Ha minden megfejtés megvan, MsgBox-ban gratuláljon is!

Ennyi játékos van egy focicsapatban.	11
Ennyi rabló volt Alibabához.	40
Ennyi byte egy Kilobyte	1024
Ennyi képviselői hely van a parlamentben	386
Az ötszög belső szögeinek összege	540
Ennyi másodperc egy óra	3600
Ebben az évben hat meg Karl Marx.	1883
Ennyi naponta van telihold	21
Ennyi ezer km a Föld sugara	6

57. Feladat

Javítsuk ki az előző feladatot úgy, hogy a játékot tetszőleges számú feladattal ki lehessen egészíteni, azaz az ellenőrzést az első üres sorig hajtsuk végre!

58. Feladat (+)

Módosítsuk a 56. feladatot úgy, hogy ha a játékos új megoldást ír a táblába, a gép rögtön ellenőrizze, hogy a megoldás jó-e! Ha igen írja a megoldás mellé, hogy „Helyes"! *(Vigyázzunk arra, hogy nehogy végtelen ciklusba bonyolódjunk! Ezt úgy tudjuk elkerülni, ha mindig csak annak a cellának az értékét változtatjuk, aminek tényleg változik az értéke, tehát*

ha egy cellában már szerepel, hogy „helyes”, akkor nem írjuk ezt felül, vagy ha egy cella üres, akkor nem tesszük újra üressé.)

59. Feladat

Az előző feladatot tovább szépíthetjük, ha nem azt írjuk a helyes megoldás mellé, hogy „Helyes”, hanem egy pipát teszünk, *(a pipa a Wingdings betűtípus ü betűje)* valamint ha a Megoldás munkalapot elrejtjük. *(Munkalapot úgy tudunk elrejteni, hogy a Visual Basic szerkesztőben a Properties ablakban a munkalap Visible tulajdonságánál a 0xlSheetHidden értéket választjuk)*

60. Feladat

Nyissunk meg egy üres munkafüzetet! Helyezzünk el egy fekvő és egy álló gördítősávot, mindegyiket 2 és 10 között lehessen léptetni egyesével! Minden léptetésnél a gördítősávok értékének megfelelő cella háttérszíne legyen zöld, a többié legyen a szokásos! Például, ha a vízszintes gördítősáv értéke 4, a függőlegesé pedig 7, akkor a D7 cella háttérszíne változzon!

61. Feladat

Nyissunk meg egy üres munkafüzetet! Töltsük fel az A1:H10 cellatartományt 1 és 100 közötti véletlen egész számokkal (1 és 100 is lehessen)! Helyezzünk el egy gördítősávot, amely 1 és 100 közötti egész értékeket vehet fel! Ha a gördítősáv értéke változik, akkor az A1:H10 cellatartományban azoknak a celláknak, amelyeknek az értéke megegyezik a gördítősáv értékével a háttere legyen sárga, a körülötte levőké pedig legyen piros! (Figyeljünk arra, hogy a tartomány szélein kívül ne színezzünk akkor sem, ha a megtalált szám a tartomány szélére esik!)

62. Feladat

Nyissunk meg egy üres munkafüzetet! Helyezzünk el egy kép vezérlőt, aminek kép tulajdonságánál állítsunk be egy tetszőleges képet! Ha a képre kattintunk, jelenjen meg egy MsgBox a kép címével! Például tegyünk fel egy virágot és ha rákattintunk, akkor a MsgBox-ban az jelenjen meg, hogy „Ez egy virág”!

10.8.2 Mit tanultunk meg

– A munkalap eseményei
– A vezérlők eseményei

10.9 Függvények írása Excelben

Saját függvények

Excelben **saját függvények**et is készíthetünk. Ezek megjelennek a **függvényvarázsló**ban is a **Felhasználói függvénykategória** alatt. A függvényeket külön **modulba kell írni** nem abba, ahol a saját makróink vannak, de egy modulba több függvény is írható. A függvény névadására a makróknál tanultak érvényesek. A függvény kezdő és záró sora mindig az alábbi:

```
Function függvénynév()
      …
End Function
```

Függvény-készítés lépései

Első lépésként hozzunk létre egy **modul**t függvényeinknek. Az új modulunkba írjuk be a fenti két sort (függvénynév legyen a „Proba”) és nézzük meg Excelben, hogy mi történik!

Megjegyzés: a függvények elnevezésére is ugyanaz a szabály, mint a makrók elnevezésére, azaz egy szóból kell álljon, mely **nem kezdődhet számmal**, és nem tartalmazhatja a legtöbb írásjelet. **Ékezetes betűk használata** korábbi Windows verziókkal való kompatibilitás miatt **nem ajánlott**. Névnek már foglalt Visual Basic és Excel kulcsszavakat sem szabad adni, mert keveredéseket okozhat. Foglalt kulcsszónak számít minden angol és magyar beépített függvénynév is. (A könyv mellékletében táblázatosan megtalálhatók a magyar és angol függvényelnevezések.)

Excelben hívjuk meg a **függvényvarázsló**t és válasszuk ki a függvénycsoportok közül az utolsót, a **felhasználói**t! A függvények között egyetlen függvény lesz felsorolva, az amit az előbb írtunk meg. Válasszuk ki! Megjelenik a szokásos ablak, ahol meg lehetne adni az argumentumokat, de helyette csak egy felirat van: „Ennek a függvénynek nincs argumentuma”. Persze, hogy nincs, hiszen nem adtuk meg, hogy mi legyen az. Azt is látjuk, hogy az előre számolt érték 0. Ez is azért van mert nem írtunk semmit a függvényünkbe. Nyomjuk meg a **Kész** gombot!

A **függvény eredményét** úgy adhatjuk meg, hogy a függvény neve után írjuk egy = jel után. Tehát írjuk a következőt:

```
Function masodik()
      masodik = ”Ez az eredmény”
End Function
```

Konstans-függvény

Most már ez a függvény is benne lesz a függvényvarázslóban. Argumentuma ennek sincs, de az eredmény most már nem 0, hanem az a felirat, amit idézőjelben beírtunk. Így készíthetünk **konstansfüggvény**t.

Argumentumok

Következő lépésként legyen a függvényünknek **argumentum**a is. Lehet egy vagy több, a függvény neve után a zárójelbe adhatjuk meg őket **vesszővel** elválasztva. Első példánkban egyetlen argumentumunk lesz és a függvény annyit fog tenni, hogy az argumentumot adja vissza eredményül.

```
Function ugyanaz(arg)
      ugyanaz = arg
End Function
```

A következő függvényünknek nem csak egy argumentuma van. A beírt két argumentumot egymás mellé írva, összefűzve jeleníti meg.

```
Function egymasmelle (arg1, arg2)
      egymasmelle = arg1 & arg2
End Function
```

Ha függvényünkben számolni akarunk, akkor természetesen az argumentumoknak számoknak kell lenniük. A legtöbb esetben, ha nem számot adunk meg, akkor a szokásos **#ÉRTÉK** hibaüzenet érkezik, ami jelzi a felhasználónak, hogy rossz értéket adott meg. Ha egyéb **megkötés**ünk is van az argumentumra, akkor azt a függvény elején **elágazás**ban ellenőrizhetjük, hogy teljesül-e és ha nem teljesül, akkor a cellába az előzőhöz hasonló hibaüzenetet írathatunk ki, a függvény értékét pedig csak akkor számíttatjuk ki, ha a megadott értékek megfelelők. A függvény argumentumának is megadhatunk típust. Ilyenkor egy nemmegfelelő érték megadásakor vagy szintén az **#ÉRTÉK** hibaüzenet a visszatérési érték, vagy pedig automatikus típuskonverzió jön létre.

```
Function ketszerezo (szam As Double)
      ketszerezo = szam * 2
End Function
```

Tömb-függvény

Készíthetünk olyan függvényt is, amelyiknek nem csak egy érték az eredménye, hanem egy **tömb**. Ezeket tömbfüggvénynek hívjuk. Ilyenkor a függvény neve és az egyenlőségjel után az **Array** kulcsszót írjuk és

zárójelben vesszővel elválasztva soroljuk fel a tömb elemeit. A függvény kipróbálásakor a Gyakoriság függvénynél megismert módszert kell használni: **F2, Ctrl+Shift+Enter**. Oda kell azonban figyelnünk, mert a Gyakoriság függvény függőlegesen adta az eredményeket, míg saját tömbfüggvényünk **vízszintesen** fogja.

```
Function negyzet_kob (szam)
      negyzet_kob = Array(szam^2, szam^3)
End Function
```

A fenti függvény a megadott szám négyzetét és köbét adja eredményül tömbként.

10.9.1 Feladatok

63. Feladat

Készítsünk konstansfüggvényt, amely az e-t adja eredményül! (e értéke: 2,718282)

64. Feladat

Készítsünk függvényt, amely a sugár megadása után kiszámítja a gömb térfogatát! ($V = 4r^3\Pi/3$)

65. Feladat

Készítsünk függvényt, amely két számot összead! Készítsük el az argumentumok típusának megadása nélkül és úgy is, hogy valós számként kérjük az argumentumokat! Próbáljuk ki mindkét függvényt két szám, két szöveg és egy szöveg és egy szám típusú argumentumok megadásával!

66. Feladat

Készítsünk tömbfüggvényt, amely az oldalak megadása után kiszámítja a téglalap kerületét és területét!

67. Feladat

Készítsünk függvényt, amely kiszámítja a megadott szám faktoriálisát!

68. Feladat

Készítsünk függvényt, amely egy vezetéknév és egy keresztnév megadása után előállítja a teljes nevet! Vigyázzunk, hogy a név két része ne legyen egybeírva!

69. Feladat

Készítsünk tömbfüggvényt, amely az első elem, a hányados és az elemszám megadása után tömbként visszaadja a mértani sorozat első és utolsó elemét! ($a_n=a_1 * q^{n-1}$)

70. Feladat (+)

Készítsünk tömbfüggvényt, amely az $a*x^2+b*x+c=0$ másodfokú egyenletet oldja meg az a, b és c paraméterek megadása után! A megoldásra nem vezető paraméterek esetén értelmes, a hiba típusára utaló hibaüzenetet kapjunk!

71. Feladat

Készítsünk függvényt amely kiszámítja a Fibonachi sor n-edik elemét! (Fibonachi sor első két eleme 1, 1, aztán minden új elem az előző két elem összege.)

10.9.2 Mit tanultunk meg

- A függvények nyitó és záró kulcsszava
- A függvény bemenő paramétereinek megadása
- A függvény eredményének megadása
- A tömbfüggvény készítése

10.10 Hibák a programban

A programozás során követhetünk el hibákat, amelyek megtalálásában és kijavításában is segítségünkre tud lenni a **Visual Basic** szerkesztő.

Különböző szintű és típusú hibákat különböztethetünk meg.

Szintaktikai hibák

A **szintaktikai** hibákat már **írás közben** jelzi a program. Hiányzó szóköz, befejezetlen sor esetén azonnal piros betűsre váltja a szöveget és egy hibaüzenetet is küld. Ezek kijavítása nélkül **nem tudjuk futtatni** a programot. Ha **kulcsszavak**at gépelünk el, azt nem jelzi ily módon, de azért felfedezhető a hiba. A Visual Basic nem érzékeny a kis és nagybetűkre, sőt ha felismeri a beírt kulcsszót automatikusan átalakítja meghatározott karaktereit nagybetűsre. Tehát jól tudjuk ellenőrizni magunkat, ha mindig mindent **csupa kisbetűvel** írunk, mert ha nem változtatja meg, akkor bizonyosan elgépeltünk valamit az adott kulcsszóban.

Fordítási hibák

Bizonyos szintaktikai hibák csak **futás közben** derülnek ki. Amikor egy kötelező sor kimarad, például nem zárjuk le az elágazást, vagy gépelés közben mégsem vesszük észre, hogy elgépeltünk egy kulcsszót. Ilyenkor a program elindulás után rögtön megszakad és a „**Compile error**” (Fordítási hiba) hibaüzenet jön, alatta megfogalmazva a talált hibát. *(A talált hiba nem mindig egyezik meg az elkövetett hibával. Ilyenkor minden szerkezeti pontját nézzük végig a programnak, hogy minden nyitásnak megvan-e a zárás párja és az a megfelelő helyen van-e!)*

Reset gomb

Ezzel egyidőben átvált a képernyő a Visual Basic szerkesztőre. Az **Ok** gomb megnyomása után javíthatjuk a hibát. A program első sorának **sárga kiemelés**e mutatja, hogy a program nem véget ért, hanem **megszakadt**. A programot csak akkor tudjuk újra futtatni, ha előbb megállítjuk a megszakadt futást. Ezt az eszköztáron a **Reset** gombbal tudjuk megtenni.

Futási hiba

Szintén a program futása közben derül ki a **futási hiba.** Ebben az esetben a program elkezd futni és csak ott áll meg, ahol a hibát találja. A hibaüzenet: „**Run-time error ’13’**” (Futási hiba), majd a hiba megfogalmazása. Ilyen hiba például, ha egy változóba nem megfelelő típusú értéket akarunk beírni, vagy olyan cellára, olyan objektumra hivatkozunk, ami nem létezik. A hibaüzenet alatt ilyenkor a Súgó gomb mellett a **Debug** és az **End** gombok jelennek meg. Az **End** gomb megnyomására a program futása azonnal megáll. A **Debug** megnyomására a hibásnak talált sort **sárgával emeli ki** a program és lehetőség van a használt **változók aktuális értékének megtekintés**ére. Ehhez csak a változó neve fölé kell vinni az egeret és egy kis sárga mezőben megjelenik annak aktuális értéke. Ez segíthet a hiba felismerésében. A sárga szín most is arra figyelmeztet, hogy a program nem befejeződött, csak megszakadt. Tehát a **Reset** gomb megnyomása után tudjuk a munkánkat folytatni.

Logikai hiba

Előfordulhat, hogy a program hibaüzenet nélkül végigfut, de mégsem az történik, amit szerettünk volna. Lehet hogy rosszul találtuk ki az algoritmust, de az is, hogy az algoritmus jó volt, csak íráskor követtünk el hibát.

Lépésenkénti futtatás

A hiba megtalálásában segíthet a **lépésenkénti futtatás**. A szerkesztőben, ha egy programsor mellett a bal oldali sávon kattintunk, ott megjelenik egy bordó pont és a programsor is bordó kiemelést kap. Így helyezünk el **töréspont**ot. Ha ezután futtatjuk a programot, akkor az adott sorhoz érve, megáll a futás és a szerkesztőképernyőn sárga lesz a **kiemelt sor**. Ez ugyanolyan állapot, mint amikor a futási hiba után a **Debug** gombot nyomtuk meg. Ha az egeret a változóink fölé visszük, megnézhetjük az aktuális értéküket. Most viszont eldönthetjük, hogy a **Reset** gombbal megállítjuk a program futását teljesen, vagy a mellette található **Continue** gombbal folytatjuk. A leírt módszerrel **több töréspontot is elhelyezhetünk** a programban, így az mindegyiknél meg fog állni.

10.10.1 Feladatok

72. Feladat

Smaug városában három légszennyezettség-mérő berendezés működik. Nyissuk meg a hibas.xls munkafüzetet! Az A:C oszlopok a három műszer által egy év alatt mért NO_2 értékeket tartalmazzák (napi átlagértékek, a határérték százalékában). A táblázathoz makrót akartunk írni, ami kiemeli azokat a sorokat pirossal, ahol a legnagyobb és a legkisebb közti eltérés nagyobb, vagy egyenlő, mint 40 százalékpont, zölden pedig azokat, ahol 10 vagy annál kisebb. A makrónk összesen 6 hibát tartalmaz. Keressük meg és javítsuk ki ezeket a hibákat, hogy megfelelően működjön a makró!

10.10.2Mit tanultunk meg

- hibák típusai
- hibakeresési és javítási lehetőségek
- töréspontok a programban