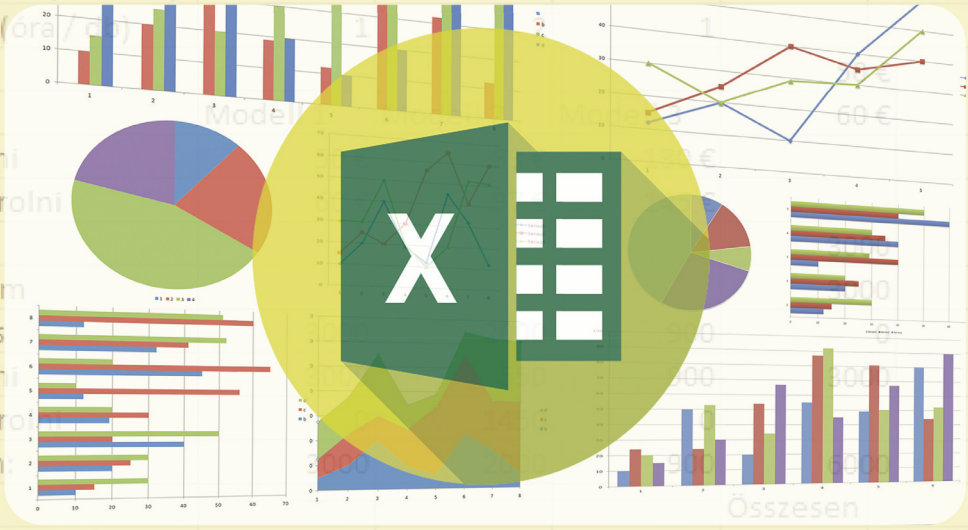
SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM
CSÍKSZEREDAI KAR

jegyzet



FILEP LEVENTE GARDA-MÁTYÁS EDIT OLÁH-GÁL RÓBERT

EXCEL KÖZGAZDÁSZ- ÉS MÉRNÖKJELÖLTEKNEK

Scientia Kiadó | 2021

FILEP LEVENTE
GARDA-MÁTYÁS EDIT
OLÁH-GÁL RÓBERT

EXCEL KÖZGAZDÁSZ- ÉS MÉRNÖKJELÖLTEKNEK

SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
CSÍKSZEREDAI KAR

EXCEL KÖZGAZDÁSZ- ÉS MÉRNÖKJELÖLTEKNEK

FILEP LEVENTE
GARDA-MÁTYÁS EDIT
OLÁH-GÁL RÓBERT

Scientia Kiadó
Kolozsvár ▪ 2021

Megvalósult a Magyar Kormány támogatásával
Miniszterelnökség Nemzetpolitikai Államtitkárság
Bethlen Gábor Alap

Felelős kiadó:
Sorbán Angella

Lektor:
Bíró Piroska (Csíkszereda)

Borítóterv:
Tipotéka Kft.

Kiadói koordinátor:
Szabó Beáta

Első magyar nyelvű kiadás: 2021

Descrierea CIP a Bibliotecii Naţionale a României
FILEP, LEVENTE
Excel közgazdász - és mérnökjelölteknek / Filep Levente, Garda-Mátyás Edit, Oláh-Gál Róbert. - Cluj-Napoca : Scientia, 2021
Conţine bibliografie
ISBN 978-606-975-053-7
I. Garda-Mátyás, Edit
II. Oláh-Gál, Róbert

004

TARTALOM

CONTENTS

CUPRINS

ELŐSZÓ

Az *Excel közgazdász- és mérnökjelölteknek* c. jegyzetünket átlagban egy tanévben 150 hallgató fogja forgatni. Legalábbis átlagban 150 hallgatónak kötelező tananyag. Ezt a tárgyat több mint 10 éve oktatjuk, és azt tapasztaljuk, hogy általában a legtöbb közgazdasági szakon oktatják világszerte. Az ötévenkénti szakakkreditálásoknál a bizottságok állandóan megkövetelik a friss (5 évnél nem régebbi nyomtatott vagy elektronikus) jegyzetek meglétét, ezért döntöttünk úgy, hogy jelen jegyzetünket közreadjuk. Figyelem, az adataink mind példaadatok, saját elképzeléseink, tehát ezért nem szerepel az adatok forrásánál a táblázatok alatt külön kiemelve az adatok forrása. Jegyzetünket egyaránt használhatják középiskolások és más szakok hallgatói is, akik alaposabban meg szeretnének ismerkedni az Excel lehetőségeivel.

Az informatika alapjai c. alapozó tárgyat 22 éve oktatjuk a Csíkszeredai Karon. Az első öt évben két féléves tárgy volt, csak utána lett egy féléves tárgy. Amíg két félévben tanítottuk, addig az első félévben logikai alapokat és Boole-algebrát, a második félévben pedig a Pascal nyelv alapjait adtuk le. Ma már az informatikai műveltségünkhöz tartozik, hogy a Pascal nyelvet didaktikai céllal alkotta meg Nikolas Wirth és Katalin Jensen, és úgy is hívták, hogy a Pascal nyelv lényegében egy futtatható pszeudokód. Nagyjából 10 egyetemi évben adtuk le a Pascal nyelvet, de nem volt hatékony, főleg a közgazdászjelölteknek, ezért a tanszék és a kari tanács úgy döntött, hogy mást kell választani. Mivel nagy gondok voltak a statisztika tárgy megértésében és elsajátításában, ezért az Excelre esett a választás. (Rövid ideig a statisztikát MATLAB szoftver segítségével oktatták, de az sem bizonyult hatékonynak, ezért áttértek az Excelre.) Természetesen a hallgatók érdekében történt, de akkreditációs követelmény is volt, ezért írott jegyzettel is támogattuk ezt az első 10 év informatikaoktatását, és így született meg *Az informatikai alapjai mérnök és közgazdász hallgatóknak* c. jegyzet. Ez a jegyzetünk nem lehetett rossz, mert sokan letöltötték más egyetemekről is, és ma is „pénzért", illegálisan árulja a SCRIBAD weboldal.

Rövid ideig OGR-statisztikát is oktatott, és az Excel hasznosságának igazolására született meg a *Közgazdasági adatok statisztikai feldolgozása Excel segítségével* c. írott jegyeztünk. Ezzel párhuzamosan, következetesen az Excelre alapozzuk az informatika alapjai, informatika és informatika II. tárgyat, sőt a 3. félévben oktatott gazdasági informatika tárgy keretében is főleg VBA-t és az Excel gazdasági alkalmazásait tanítjuk.

Ma már az Excel ismerete nem csak a közgazdászoknak, de szinte minden tisztviselőnek, irodai dolgozónak kötelező, így hozta az élet, és az utóbbi 50 év talán egyik legsikeresebb, legnépszerűbb szoftvere lett. És valóban jól lehet(ne) vele oktatni felsőbb matematikai ismereteket is, a statisztika alapjait, pénzügyi

alapokat, kutatásmódszertant, adatbázis-kezelési alapokat, és a programozás alapjai is jól oktathatók a VBA segítségével. Ezen állításunk igazolására született meg a jelenlegi jegyzetünk.

1. AZ EXCEL FÖLÖTTÉBB HASZNOS VOLTÁRÓL

Az Excel az egyik legnépszerűbb szoftver, talán még a Wordnál is hasznosabb és ismertebb. Egyaránt használhatják a napi gazdaságban (bevásárlásban), a háztartásban, az üzleti életben, az oktatásban, és mérnöki számításokra is alkalmas. Olcsó és könnyen elérhető.

A háztartásban: a bevásárlásaink, kiadásaink és bevételeink nyilvántartására, a víz-, gázfogyasztásunk követésére, a közköltségek követésére, napi munkánk, órarendünk követésére stb. használható.

Legalapvetőbb használatához elegendő, hogy tudjuk: cellái vannak, és azoknak a címei, például a D7 azt jelenti, hogy a 4. oszlop és 7. sor cellájának a tartalma. A cella tartalmazhat adatot vagy képletet. Az adat lehet szöveg, érték, dátum stb. A képletet kiértékeli az Excel, és ebben áll az egyik erőssége. Az általános intelligenciához tartozik, hogy megértsük, mit jelent egy képletet kifejteni, kiértékelni. Például `=A6+B2` azt jelenti, hogy összeadjuk a A6-os cella tartalmát a B2-es cella tartalmával. Az egyenlőség azt jelenti, hogy az Excel ki kell értékeljen valamit, ki kell fejtsen valamit.

A cellák címzése lehet relatív (ez a legtermészetesebb), lehet abszolút és lehet vegyes.

A relatív azt jelenti, hogy ha a 2. sorban és a 3. oszlopban lévő képletet `(=A2+B2)` sokszorosítjuk, lejjebb húzzuk, akkor `=A3+B3` és így tovább `=A4+B4`, …, `=A7+B7` lesz, vagyis relatívan változtatja a címzést. Úgy is kiolvashatjuk, megfelelő sorban, a megfelelő képlet. És ez igaz az oszlopra is. Megfelelő oszlopban a megfelelő képlet. (Ami igaz a sorokra, igaz az oszlopokra is az Excelben!)

Az abszolút címzés azt jelenti, hogy egy cella rögzített, ezt általában a dollárjel jelenti. Pl. `=$A4+B4`, azt jelenti, hogy akárhogy is húzzuk, költöztetjük a cellát, mindig az A oszlopot rögzítettük, míg ha `=A4+B$4`, akkor a 4. sort rögzítettük a második összeadandóban.

Ezeket a szabályokat a gyakorlás szinte észrevétlenül megtanítja. Az Excelben is érvényes az általános szabály (melyet az informatika elsajátítására alkalmazunk): be kell gyakorolni, ki kell próbálni a konkrét esetekben.

2. AZ EXCEL LEGFONTOSABB MŰVELETEI

Az Excel legfontosabb műveletei az összegzés, „kontorizálás”, vagyis hogy miből hány darab van, százalékszámítás, fejléc megadása, ami azt jelenti, hogy az adatokat tételekbe és mezőkbe rendezzük, valamint a legnagyobb és legkisebb értékek kiszámítása.

Fontos aritmetikai műveletei: összeadás, kivonás, szorzás, osztás, hatványozás.

Az Excel alapértelmezésben valós osztást végez, és nagyon fontos lehet a maradékos osztás, ami már nagyobb figyelmet igényel.

Kevésbé gyakori, de fontos: maradékos osztás (`=mod()`), egész hányados (`=quotient()`), számtani átlagszámítás stb.

Természetesen ez attól is függ, hogy ki mire használja az Excelt. A tanárnak az átlagszámítás (médiaszámolás) és rendezés bőven elegendő, ha nem használja az oktatásban. Ha már az oktatásban használja, akkor rengeteg specifikus függvényt érdemes bemutatni.

Az Excel automatikusan felismeri a beírt adat formátumát: a szöveges adatokat alapértelmezés szerint **balra** igazítja a cellán belül; a számokat, dátumokat **jobbra** igazítja, míg a logikai értékeket [Igaz (True), Hamis (False)] vagy az ezeket eredményül adó képleteket **középre** igazítja. Természetesen ez az igazítás később átformázható, de beíráskor jól használható ellenőrzésre.

Ha a beírt szám túl sok számjegyből áll (pl. 12345678456), akkor a számot átalakítja tudományos formátumba (pl. 1,2E+10), vagy ha túl keskeny az oszlop, a szám helyett rácsok jelennek meg a cellában (###), de az oszlop kellő szélesítése után látható lesz a szám a cellában.

2.1. Cellák formázása

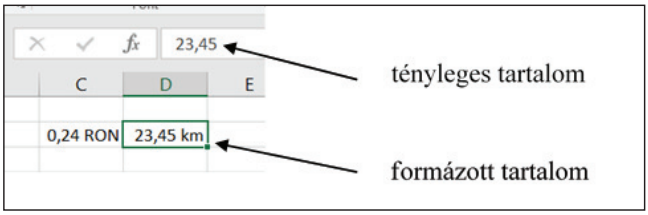
Amikor cellákat formázunk az Excelben, megváltoztatjuk a cella megjelenését, anélkül, hogy magát a tartalmát is megváltoztatnánk. Ha megformáztunk egy cellát, utána bármit írunk, a cellába írt adatok felveszik a formátumot.

A cellákban a formázott tartalom látszik, a szerkesztőlécen a cella igazi tartalma:

a cellák **betűtípus-**formázása és **igazítása** lényegében megegyezik a **Word** lehetőségeivel

- formázások a Cellák (Cells) csoportban: sormagasság, oszlopszélesség, munkalap átnevezése stb.
- formátum törlése: Kezdőlap (Home) → Szerkesztés (Editing) csoport → Törlés (Clear) → Formátum törlése (Clear Formats).

Fontos! Ha számokat írunk egy cellába, akkor **SOHA** ne írjuk be a mértékegységét! A mértékegység az a formátum, amelyet a formázásnál fogunk megadni.

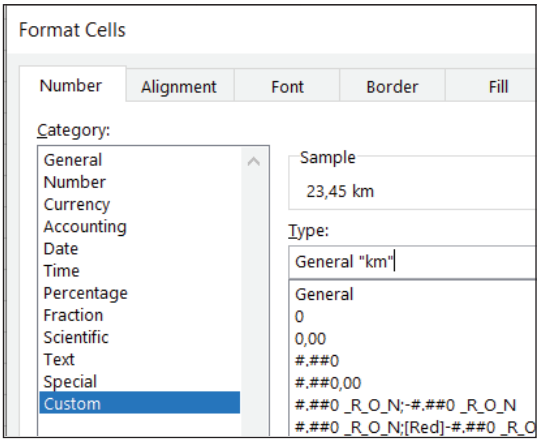


2.1.1. ábra. *Formázott és tényleges tartalom*

Cellaformázás:
Válasszuk ki a formázandó cellát, vagy jelöljük ki a cellatartományt.
- Lehetőségek:
- Kezdőlap (Home) → Szám (Number) legördülő menü vagy
- CTRL + 1 vagy
- kattintsunk a jobb egérgombbal a cellára vagy a cellatartományra, válasszuk a Cella formázása… (Format Cells…) lehetőséget, majd a Szám (Number) csoportot

Formátumok:
- Általános (General): alapértelmezett formátum.
- Szám (Number): a számok általános megjelenítésére szolgál. Megadhatjuk a tizedesjegyek számát, illetve azt, hogy szeretnénk-e ezres elválasztót használni (pl. 3.456.678,14).
- Pénznem (Currency): választhatunk pénznemszimbólumot, megadhatjuk a megjeleníteni kívánt tizedesjegyek számát, ezres elválasztót (pl. 1.234,23 RON).
- Könyvelés (Accounting): pénznemhez hasonló, de igazítja az oszlopban a pénznemszimbólumokat és a számok tizedespontjait.
- Dátum (Date): az Excel a dátumokat sorszámként kezeli (1900 január 1 az 1-es sorszámú). Ezért a dátumok és időpontok értékek, így szerepelhetnek aritmetikai számításokban. Választhatunk területi beállítást (helyet) és típust (pl. 2020. december 12.).
- Idő (Time): az Excel az időt is sorszámként kezeli, a megadott típustól és területi beállításoktól függően (pl. 1:30:54 du.).
- Százalék (Percentage): megszorozza a cella értékét 100-zal, az eredményt százalék (%) szimbólummal jeleníti meg. Kiválaszthatjuk a megjelenítendő tizedesjegyek számát (pl. 35%).

- Tört (Percentage): A számot törtként jeleníti meg, a kiválasztott tört típusának megfelelően (pl. 3/25).
- Tudományos (Scientific): egy számot exponenciális jelöléssel jelenít meg, a szám egy részét E±n-re cserélve, ahol E megszorozza az előző számot 10 a ±n-edik hatványával (pl. −0,0002356 tudományos formátumban való megjelenítése −2,3E-04).
- Szöveg (Text): a cella tartalmát szövegként kezeli, és pontosan úgy jeleníti meg a tartalmat, ahogy azt beírjuk, még számok beírása esetén is (pl. 0123).
- Különleges (Special): a számot irányítószámként, telefonszámként vagy társadalombiztosítási számként jeleníti meg.
- Egyéni (Custom): mértékegységek megjelenítéséhez is használhatjuk. A Formátumkód (Type:) mezőbe az Általános (General) után szóközzel " " között beírhatjuk a kívánt szót, amit a szám után szeretnénk megjeleníteni (pl. 23,45 km).



2.1.2. ábra. *Cellaformázás Egyéni formátummal*

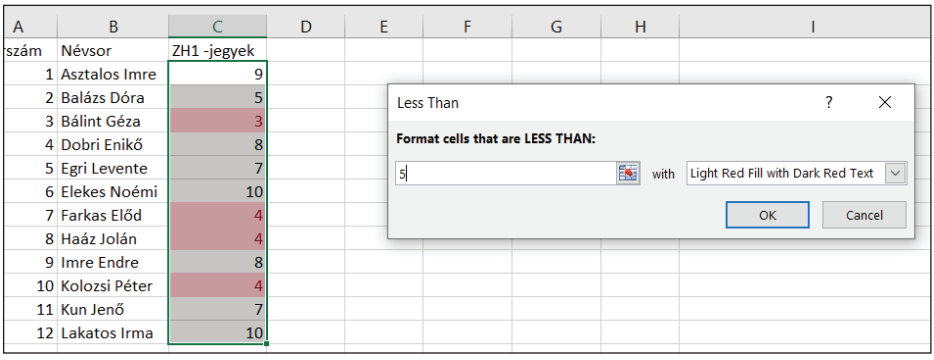
Formátum másolása

- kattintsunk a formátummásoló gombra (egy kattintással csak egyszer másolunk),
- dupla kattintással több, nem összefüggő cellát, illetve tartományt is tudunk egymás után formázni (másolás kikapcsolása: 'Esc' billentyű, vagy kattintsunk ismét a formátummásolóra).

2.2. Feltételes formázás

Sok adat esetén a táblázatra ránézve nem könnyű megállapítani a számunkra fontosakat. Feltételes formázással könnyen kiemelhetőek az érdekes tartalommal bíró cellák, kihangsúlyozhatóak a szokatlan értékek vagy hibás adatok.

Jelöljük ki a kívánt tartományt: Kezdőlap (Home) → Stílusok (Styles) → Feltételes formázás (Conditional Formatting): az adott cellatartomány azon celláira alkalmazza a kiválasztott formátumot, amelyek teljesítik a feltételt. Számos beépített feltétel létezik, és saját feltételeket is létrehozhatunk.



2.1.2.1. ábra. *Az 5-ösnél kisebb jegyek kiemelése*

2.3. Kitűzött feladatok

1. Másoljuk át az alábbi táblázatot:

2.2.1. táblázat. *Egyszerű példaadatok*

Számok	Pénznem	Dátum	Idő	Százalék	Tört	Szöveg	Mérték-egység
12,34	12400	15.03.2021	12:34	0,123	0,24	123	120
12340000	67,34	16.03.2021	12:34:56	0,987	12,34	bokor	25
12,34567	20000	17.03.2021	13:24		67,78		32
-12,34	25	18.03.2021	13:24:12		1,456		410

Formázzuk az adatokat a következő alakba:

Számok	Pénznem	Dátum	Idő	Százalék	Tört	Szöveg	Mértékegység
12,3	$12.400	15 martie 2021	12:34:00	12%	2/10	0123	120 db
12.340.000,0	€ 67,34	21. március 16.	12:34 du.	98,7%	12 1/2	bokor	25 km
12	20.000 HUF	3.17.2021	1 óra 24 perc du.		67 3/4		32 csomag
-12,340	25,00 RON	18.3.2021	1:24:12 PM		1 5/10		410 év

2.2.2. ábra. *Formázott példaadatok*

2. Másoljuk át az alábbi táblázatot:

2.2.3. táblázat. *Gyümölcskészlet példa*

Áru						
termék	ár	mennyiség	származási hely	érkezési idő	szavatossági idő	minőségét megőrzi
alma	2,5	80	Románia	15.08.2020	22	16.01.2021
körte	4,9	35	Románia	16.08.2020	14	22.11.2020
banán	5,2	52	Bolívia	17.08.2020	3	07.09.2020
narancs	4,7	45	Kuba	18.08.2020	5	22.09.2020
őszibarack	11	15	Magyarország	19.09.2020	5	24.10.2020
szőlő	5,7	15	Anglia	20.09.2020	4	18.10.2020
eper	14	12	Németország	21.09.2020	2	05.10.2020

Formázzuk a táblázatot a következő alakba:

Áru						
termék	ár	mennyiség	származási hely	érkezési idő	szavatossági idő	minőségét megőrzi
alma	3 RON/kg	80 kg	Románia	2020-08-15	22 hét	2021-01-16
körte	5 RON/kg	35 kg	Románia	2020-08-16	14 hét	2020-11-22
banán	5 RON/kg	52 kg	Bolívia	2020-08-17	3 hét	2020-09-07
narancs	5 RON/kg	45 kg	Kuba	2020-08-18	5 hét	2020-09-22
őszibarack	11 RON/kg	15 kg	Magyarorsz	2020-09-19	5 hét	2020-10-24
szőlő	6 RON/kg	15 kg	Anglia	2020-09-20	4 hét	2020-10-18
eper	14 RON/kg	12 kg	Németorsz	2020-09-21	2 hét	2020-10-05

2.2.4. ábra. *Formázott gyümölcskészlet*

3. Másoljuk át a következő táblázatot, majd formázzuk az eladást a következő feltételek szerint:

a) A száznál kevesebb eladások félkövéren jelenjenek meg.

b) A100 és 200 közötti eladások dupla vonallal aláhúzva legyenek.

c) A 200-nál nagyobb eladások piros háttérrel jelenjenek meg.

2.2.5. táblázat. *Italeladás példa*

Egységár	Ital	1. hét	2. hét	3. hét	4. hét	5. hét	6. hét	7. hét
4	Rövid kávé	120	118	126	129	135	140	145
5	Hosszú kávé	130	132	130	138	140	125	134
6	Cappuccino	150	155	125	165	170	172	168
6	Mocaccino	170	169	168	171	180	182	190
5	Kakaó	80	90	101	120	125	133	134
4	Tea	110	108	102	101	108	134	124
6	Jeges tea	90	98	92	90	84	40	46
8	Narancslé	70	75	65	64	55	58	56
10	Limonádé	75	90	87	88	102	130	125
1	Cukor	240	250	241	247	259	260	254
2	Tej	190	185	199	230	240	246	233

4. Írjuk be a következő táblázat adatait, formázzuk hasonló kinézetűre, majd a 25%-nál kisebb adatokat tegyük félkövérré, míg a 42%-nál nagyobb adatokat fehér háttérszínnel lássuk el:

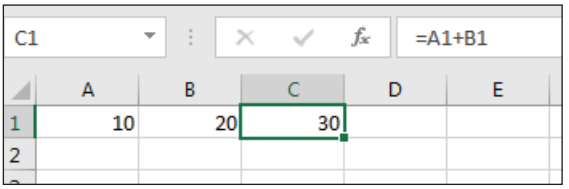
	Alma	Körte	Szilva	Barack
2018	30%	18%	32%	20%
2019	43%	25%	26%	6%
2020	20%	30%	43%	7%

2.2.6. ábra. *Gyümölcsbevétel példa*

3. KÉPLETEK ÉS CELLACÍMZÉSEK

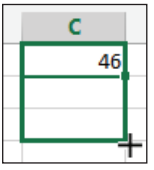
Excelben a cellák oszlopokba és sorokba vannak rendezve. Vízszintesen az oszlopok betűkkel vannak jelölve, míg függőlegesen a sorok számozva vannak, ami lehetővé teszi a cellákra való vonatkozást. Egy ilyen cellacím mindig egybeírt oszlop- és sorszámból áll, például: A1, ahol az ”A” az A oszlopot jelenti és az ”1” pedig az 1-es sort.

Ha az A1 és B1 cella értékeit össze akarjuk adni a C1 cellába, akkor ezt a `=A1+B1` képlet segítségével ki tudjuk számolni:



3.1.1. ábra. *Egyszerű cellaösszeadás képlete*

Mint említettük, a képletek kiértékelése érdekében ”=” jellel kell kezdjük, különben az Excel sima szövegként kezeli ezt és a beírt formában megjeleníti. Egy cella képletét alkalmazni lehet vízszintes vagy függőleges irányban, aminek következtében az Excel kitölti az új cellákba a meglévő képletet (Autofill). Ez nagyon hasznos funkció, amikor egyetlen képletet több, más sorban vagy oszlopban levő adatra is használni akarunk. Ezt a képletes cella jobb oldalában levő zöld kocka segítségével lehet, ezt megfogva és elhúzva a kívánt irányba:



3.1.2. ábra. *Képletek alkalmazása több cellára (Autofill)*

Ilyenkor az Excel lemásolja a kiinduló cellában levő képletet a kívánt cellákba. Mivel az újonnan átmásolt képlet már más sorban vagy oszlopban van, feltételezve, hogy ezek adatait szeretnénk a képlettel használni, ezért az Excel átírja a cellacímeket:

- függőleges (lefele vagy felfele) alkalmazás esetén a cellacímek sorszámait változtatja,

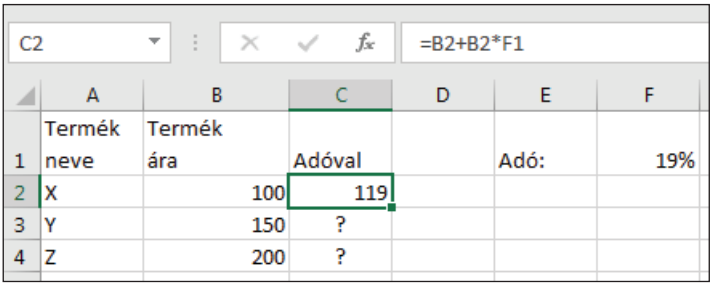
- vízszintes (jobbra vagy balra) alkalmazás esetén a cellacímek oszlopait változtatja.

Példa 1: a C1 cella `=A1+B1` képletét alkalmazzuk lefele pár sort. Az alatta való cellában megfigyelhetjük, hogy a képlet `=A2+B2` lesz, a következő sorban pedig `=A3+B3`, és így tovább. Mivel egy sorral lennebbi adatokkal dolgozunk, ezért egy sorral lennebb a képletben levő cellák sorai nőnek eggyel, ezért lesz az A1-ből A2.

Példa 2: ugyancsak a C1 cella képletét jobbra alkalmazva, a következő cella értéke `=B1+C1` lesz, a rákövetkező pedig `=C1+D1`. Mivel jobbra alkalmazzuk, ugyanabban a sorban maradunk a képlettel, viszont egy-egy oszloppal jobbra lépünk, ezért az Excel úgy veszi, hogy egy-egy oszloppal jobbra levő értékekkel dolgozunk, ezért a cellacímekben az oszlop nevét növeli, így lesz az A1-ből B1, majd C1.

3.1. Rögzített cellacímek

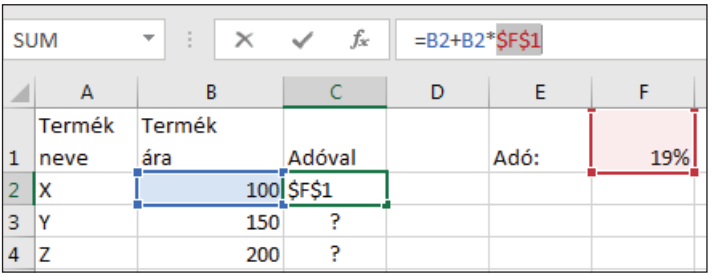
A cellacím átírása igencsak hasznos funkciója az Excelnek, viszont előfordul, hogy nincs erre szükségünk. Tekintsük a következő példát:



3.1.1. ábra. *Egyszerű adószámolás példa*

A táblázat B oszlopában némely termékek ára van, a C oszlopban viszont ki szeretnénk számolni áraikat bizonyos adóval. Ha az adó változhat, akkor nem érdemes ezt a képletbe közvetlen beírni (például: `=B2+B2*19%`), hanem egyszerűbb ennek értékét egy adott cellából venni, például F1, így a képletünk `=B2+B2*F1` lesz. Viszont ha ezt a képletet alkalmazzuk lefele, akkor a következő sorban az F1 cella címe F2 lesz, a rákövetkezőben F3 és így tovább, ami nem lesz jó, mert ezen cellák üresek. Ilyen esetekre a megoldás a cellacímek rögzítése, mely a dollárjel segítségével történik (rögzített cím: `$F$1`), aminek következében a képlet más cellákba való alkalmazása esetén az Excel nem módosítja ezt a rögzített címet. A címben a $F jelentése, hogy az F oszlop, a $1 pedig, hogy az 1-es sor rögzítve van.

Egy kiválasztott cellacím rögzítésére az F4 gyorsbillentyű is a rendelkezésünkre áll (a címet nem kell kijelölni, elég a kurzorral rajta állni):



3.1.2. ábra. *Egyszerű adószámolás példa megoldva*

Feladat: Próbáld ki a fenti példát.

3.2. Vegyes címek

Egy cellacímet akkor nevezünk vegyesnek, ha az oszlop és sor csak egyike van rögzítve. Példaként vegyük az `$A1` címet, melyben az oszlop rögzített, míg az `A$1`-ben a sor rögzített. Képlet alkalmazása esetén a címátírás is követi a rögzítést.

Feladat: Az A1 cellába írj egy értéket, a B1 cellába pedig vedd át ezt az értéket (`=A1`), és a képletben:

- rögzítve az oszlopot (`$A1`) alkalmazd a képletet jobbra, illetve lefele,
- rögzítve a sort (`$A1`) alkalmazd a képletet jobbra, illetve lefele,

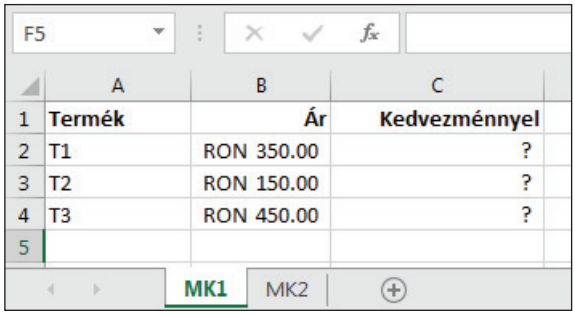
és mindkét esetben figyeld meg a cellacím átírását.

3.3. Munkalapokon keresztüli címzés

A munkalapok nevei alapból Sheet1 (Munkalap1), Sheet2 (Munkalap2) és így tovább. Ezeket dupla klikkel a nevén vagy jobb klikk és a „Rename” opció kiválasztásával meg tudjuk változtatni. Az eddigiekben láttuk, hogyan lehet cellákat címezni, valamint ezeket képletekben használni. Sokszor előfordul, hogy az adatok nem egy azonos munkalapon belül vannak, hanem több munkalapon. Ilyen esetben a munkalap nevét a cellacím elé lehet írni, így hivatkozva egy cellára egy másik munkalapon. A munkalap nevét kötelező módon egyes aposztrófba kell tenni, melynek szerepe, hogy a munkalap neve egyértelmű legyen olyan esetben is, amikor szóköz van a nevében (például 'Munkalap 2'). A munkalap neve és a cellacím mindig egy '!' jellel vannak elválasztva egymástól. Például: `,munkalap neve'!A1`.

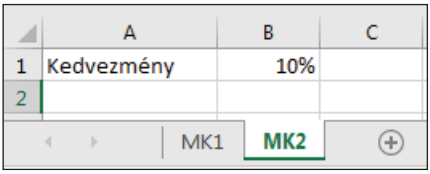
Példa: Legyen MK1 és MK2 két munkalap neve (sorrendjük nem lényeges). Az 'MK1' munkalapon legyen egy terméknév és ár táblázat. A célunk, hogy ezen

termékek kedvezményes árait kiszámoljuk az 'MK2' munkalapon levő kedvezmény szerint. Legyen az 'MK1' munkalapon a következő:



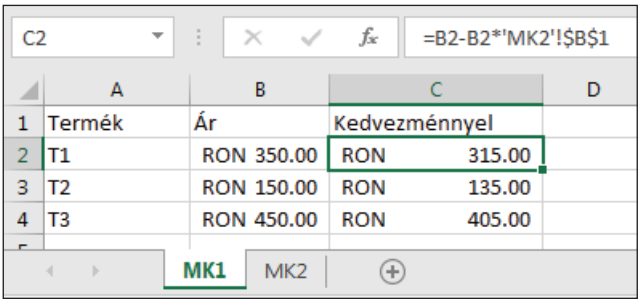
3.3.1. ábra. *Kedvezményszámolás példa (MK1 munkalap)*

A kedvezményes ár kiszámolására a kedvezmény legyen az 'MK2' munkalapon:



3.3.2. ábra. *Kedvezményszámolás példa (MK2 munkalap)*

Megoldás: a képlet beírása közben a kedvezmény cellacímhez, kattintsunk át az 'MK2' munkalapra és ott a B1 cellára. Ennek következtében az Excel beírja ennek teljes cellacímét, a munkalap nevével együtt. Továbbá, mivel ezt a képletet alkalmazni fogjuk lefele, ezért ezt a cellacímet szükséges rögzíteni. Itt megnyomhatjuk az F4 billentyűt (vagy kézzel hozzátesszük a `$` jeleket). A rögzített cellacím a következő formában kell kinézzen: `,MK2'!B1`.



3.3.3. ábra. *Kedvezményszámolás példa megoldva*

3.4. Feladatok

3.4.1. Fibonacci-sorozat, Collatz-sejtés példa

Generáljuk a Fibonacci-sorozat első 20 elemét:
`F0=1, F1=1 és Fn=Fn-1+Fn-2.`
A relatív cella címzés egyik legegyszerűbb és legfontosabb esete: például A1 tartalma 1, A2 cella tartalma 1 és akkor A3 cella tartalma `=A1+A2`, és ezt aztán „lehúzzuk”: A4 cella tartalma `=A2+A3`, A5 cella tartalma `=A3+A4` és így tovább a kívánt generálás eléréséig!

Collatz-sejtés:
Veszünk egy tetszőleges pozitív egészet! Ha páros, elosztjuk kettővel, ha páratlan, megszorozzuk 3-mal és hozzáadunk 1-et. Ez a sorozat mindig az 1-hez tart. Miért? Ez egy nyitott matematikai kérdés! Aki meg tudja indokolni, az világhírű tudós lesz!

Lássuk be (teszteljük le Excelben), hogy igaz. Gépeljünk be az A2 cellába egy tetszőleges pozitív számot, és alá írhatjuk a megoldást az A3 cellába:
`=IF(MOD(A2,2)=0, QUOTIENT(A2,2),3*A2+1)`
Az `=MOD()` függvény a maradékos osztás maradékát adja, tehát két egész szám osztási maradékát, a `=QUOTIENT()` két egész szám egész hányadosát szolgáltatja.

3.4.2. Euklideszi algoritmus

Két egész szám legnagyobb közös osztóját keresi meg (l.n.k.o-val rövidítjük)!
Veszek két egész számot. A nagyobbat elosztom a kisebbikkel. Ha a maradék nulla, akkor az l.n.k.o a kisebbik szám, az osztó. Ha nem, akkor szerepcsere történik: Az osztandó felveszi az osztót, és az osztó felveszi a maradékot, és addig folytatjuk, amíg a maradék nulla lesz!
A feladat az, hogy Excelben modellezzük le ezt a szabályt.

Megoldás:
Képlettel a következőképpen kell megoldani:

3.4.2.1. táblázat. *Euklideszi algoritmus megoldása*

osztandó	osztó	maradék	teszt
345	234	=MOD(A6,B6)	=IF(C6=0, „l.n.k.o.= „&B6,”kell folytatni”)
=B6	=C6	=MOD(A7,B7)	=IF(C7=0, „l.n.k.o.= „&B7,”kell folytatni”)

osztandó	osztó	maradék	teszt
=B7	=C7	=MOD(A8,B8)	=IF(C8=0, „l.n.k.o.= „&B8,”kell folytatni”)
=B8	=C8	=MOD(A9,B9)	=IF(C9=0, „l.n.k.o.= „&B9,”kell folytatni”)

Eredmény:

3.4.2.2. táblázat. *Euklideszi algoritmus eredménye*

osztandó	osztó	maradék	teszt
345	234	111	kell folytatni
234	111	12	kell folytatni
111	12	3	kell folytatni
12	3	0	l.n.k.o.= 3

Természetesen ezt elvégzi a `=GCD(345,234)=3` Excel függvény is, de itt az algoritmuson van a lényeg!

3.4.3. Horner-séma

A következő gyakorlat a Horner-séma, amely arra szolgál, hogy egy n-ed rendű polinom $a=x_0$ pontbeli értékét gyorsan kiszámoljuk:

Adott a következő $P(x) = 3x^4+x^3+x^2-2x+3$ 4-ed rendű polinom. Számítsuk ki a P polinom értékét az $x_0=2, 1, 0, -1, 3$ és 4 pontokban.

Megoldás:

3.4.3.1. táblázat. *Horner-séma megoldása*

	3	**1**	**1**	**-2**	**3**
2	=B2	=$A3*B3+C$2	=$A3*C3+D$2	=$A3*D3+E$2	=$A3*E3+F$2
1	=B3	=$A4*B4+C$2	=$A4*C4+D$2	=$A4*D4+E$2	=$A4*E4+F$2
0	=B4	=$A5*B5+C$2	=$A5*C5+D$2	=$A5*D5+E$2	=$A5*E5+F$2
-1	=B5	=$A6*B6+C$2	=$A6*C6+D$2	=$A6*D6+E$2	=$A6*E6+F$2
3	=B6	=$A7*B7+C$2	=$A7*C7+D$2	=$A7*D7+E$2	=$A7*E7+F$2
4	=B7	=$A8*B8+C$2	=$A8*C8+D$2	=$A8*D8+E$2	=$A8*E8+F$2

Ellenőrzés
=$B$2*A3^4+$C$2*A3^3+$D$2*A3^2+$E$2*A3+$F$2
=$B$2*A4^4+$C$2*A4^3+$D$2*A4^2+$E$2*A4+$F$2
=$B$2*A5^4+$C$2*A5^3+$D$2*A5^2+$E$2*A5+$F$2
=$B$2*A6^4+$C$2*A6^3+$D$2*A6^2+$E$2*A6+$F$2
=$B$2*A7^4+$C$2*A7^3+$D$2*A7^2+$E$2*A7+$F$2
=$B$2*A8^4+$C$2*A8^3+$D$2*A8^2+$E$2*A8+$F$2

Eredmény:

3.4.3.2. táblázat. *Horner-séma megoldása*

	3	1	1	-2	3	Ellenőrzés
2	3	7	15	28	59	59
1	3	4	5	3	6	6
0	3	1	1	-2	3	3
-1	3	-2	3	-5	8	8
3	3	10	31	91	276	276
4	3	13	53	210	843	843

3.4.4. Rendelési lista

Írd be a következő táblázat adatait, és formázd meg a képhez hasonlóan:

	A	B	C	D
1	**Termék**	**Egységár**	**Darabszám**	**Összeg**
2	X	RON 100.00	RON 6.00	?
3	Y	RON 12.61	RON 24.00	?
4	Z	RON 8.00	RON 45.00	?
5	T	RON 43.50	RON 4.00	?
6	U	RON 74.99	RON 2.00	?
7	V	RON 14.35	RON 8.00	?
8				
9			Összesen:	?

3.4.4.1. ábra. *Rendelési lista feladat*

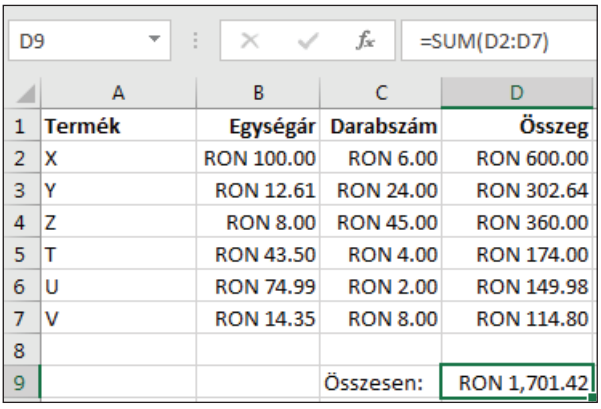
- Formázás: címek félkövér betűkkel (bold), igazítás, pénznem RON-ban és két tizedessel.

- Az összeg oszlopba írj egy olyan képletet, amit lefele lehet alkalmazni bármennyi sorra.
- A *SUM* függvényt használva számold ki, összesen mennyit kell fizetni.

Megoldás: Az összeg oszlop kiszámításához az „Ár” és a „Darabszám” oszlop értékeit össze kell szorozzuk, majd ezeket összeadni. A D2 cellába írjuk a B2 és a C2 cellák szorzatát: `=B2*C2`, mely segítségével megkapjuk az X termékre fizetendő összeget.

Ilyen típusú feladatoknál gondolkodjunk mindig sorokban, azaz mindig egy sor celláinak adataival számolunk. A következő sorok értékeit nem kell újra egyenként összeszorozni, hanem elegendő a D2 cellában levő képletet lefele alkalmazni. Mint említettük, egy képlet lefele való alkalmazásakor az Excel növeli a cellacímekben a sorszámot, így a következő sorban már az új sor celláinak értékeivel dolgozunk. Így a következő sorban a D3 cella képlete: `=B3*D3` lesz, majd a 4. sorban pedig: `=B4*D4`, és így tovább. Itt látható, hogy számos sor esetén a képletek alkalmazása más sorokra nagyban megkönnyíti munkánkat, illetve jelentős mennyiségű időt spórol.

Ha kiszámoltuk az „Összeg” oszlop sorait (D2-D7), akkor már csak az összegző marad. Ezt ki lehet számolni a cellák összeadásával is, mint például: `=D2+D3+D4+D5+D6+D7`, de sokkal egyszerűbb, ha a `SUM` függvénnyel összeadjuk, éspedig: `=SUM(D2:D7)`. A feladat teljesen megoldva az alábbi ábrán látható:



	A	B	C	D
1	**Termék**	**Egységár**	**Darabszám**	**Összeg**
2	X	RON 100.00	RON 6.00	RON 600.00
3	Y	RON 12.61	RON 24.00	RON 302.64
4	Z	RON 8.00	RON 45.00	RON 360.00
5	T	RON 43.50	RON 4.00	RON 174.00
6	U	RON 74.99	RON 2.00	RON 149.98
7	V	RON 14.35	RON 8.00	RON 114.80
8				
9			Összesen:	RON 1,701.42

*3.4.4.***2. ábra.** *Rendelési lista feladat megoldva*

Megjegyzés: mivel a B és a C oszlopok árfolyamban voltak megformázva, ezért a D oszlop celláinak értékei is automatikusan ebben lesznek megformázva.

3.4.5. Eladásiár- és bevételszámolás

Adott a következő terméktáblázat az előállítási árakkal és a F2 cellában meghatározott százaléka a kívánt profitnak. Ez azt jelenti, hogy ennyi profitot kell rászámolni mindegyikre ahhoz, hogy nyereséges legyen az adott termék.

	A	B	C	D	E	F
1	**Termék neve**	**Gyártási költség**	**Darabszám (csomag)**	**Eladási ár**		**Profit**
2	Disznófősajt	RON 4.50	30	?		**21.65%**
3	Szalámi	RON 12.75	15	?		
4	Kolbász	RON 8.65	18	?		
5	Májas	RON 14.80	21	?		
6	Véres	RON 9.65	20	?		
7	Feltűrt	RON 15.50	14	?		
8						
9						
10				Összesen:	?	

3.4.5.1. ábra. *Eladásiár- és bevételszámolási feladat*

Feladat: Számold ki mindegyik termék eladási árát, valamint azt, hogy ha minden terméket eladunk, mennyi lesz a teljes bevétel. Írj lefele alkalmazható képleteket.

Megoldás:

- Termék eladási ára: a termék gyártási ára plusz az F2 cellában levő százaléka a B oszlopnak. Ha csak annyit írunk, hogy `B2 + F2`, akkor az Excel a következőt fogja számolni `B2 + 1 * F2`, és ez nem helyes. Helyesen a B2 cellához hozzáadjuk a B2 cella F2 százalékát, azaz `=B2 + B2 * F2`. Alternatívaként úgy is fel lehet írni, hogy a B2 cella szorozva 100% + F2 -vel, azaz megszorozzuk a 121.65%-kal, tehát `=B2 * (100% + F2)`. Végül ezt a képletet alkalmazzuk lefele a D oszlop összes cellájára. Hibás eredmény esetén ellenőrizzük a cellában levő képletet. Ha hibás a képlet a lefele alkalmazott cellákban, akkor azért van, mert elfelejtettük az F2 cella címét rögzíteni, így a következő sorban F3 lett, s így tovább, ezért jött ki a gyártási ár.
- Ha a D oszlop készen van, akkor számoljuk ki, hogy ha mind eladjuk ezeket a termékeket, mekkora összeg jön be (vigyázat, nem a profit, hanem a bevétel). Ehhez használjunk egy segédoszlopot, éspedig az E oszlopot. Először, minden termék esetén, szorozzuk meg a darabszámot az eladási árral, így megkapjuk az illető termék után bejött összeget. E2 cellában: `=D2*B2`, majd alkalmazzuk a képletet lefele. Ha készen vagyunk, adjuk össze ezeket a *SUM* függvény segítségével: `=SUM(E2:E7)`.

4. EXCEL FÜGGVÉNYEK

Számos esetben nagy mennyiségű adattal kell dolgoznunk Excelben. Kézzel manipulálni ezeket az adatokat igencsak megterhelő, mert sok időt vesz igénybe és pontatlan lehet. Ilyen esetekben segít az Excel a beépített függvényeivel, melyekkel logikai műveleteket tudunk automatizálni, szövegekkel és dátumokkal tudunk dolgozni, illetve ezek együttes használatával számos komplex feladatot is le tudunk egyszerűsíteni.

A következő fejezetekben ezeket a függvényeket részletezzük, de előtte tisztázni kell a fontosabb alapdolgokat, éspedig azt, hogy formailag mit nevezünk függvénynek, általánosan hogyan működnek és mire kell odafigyelni a helyes használatukhoz.

Opcionális paraméterek jelölése a szögletes zárójel segítségével történik, éspedig

```
NEV(paraméter 1, paraméter 2, [opcionális paraméter], ...)
```

míg a "…" jelölés tetszőleges számú opcionális paramétert jelent.

A beépített függvények kiértékelése csak akkor történik meg, ha a cellába írt képletet egyenlőségjellel kezdjük, különben az Excel szövegként értelmezi ezt.

4.1. Függvények használata

A függvények olyan speciális, előre megírt képletek, amelyek segítségével egyszerű vagy összetett számításokat végezhetünk. Az Excel beépített függvényeivel sok számítást hajthatunk végre egyszerűen és nagyon gyorsan.

Egy függvény általános alakja:

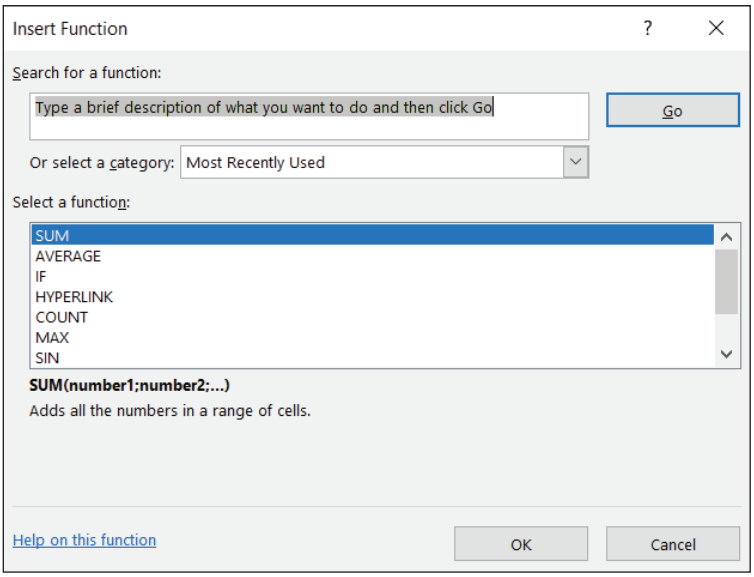
```
függvénynév([paraméter1[,paraméter2[,…]]]).
```

- a függvényeknek egyedi nevük és kerek zárójelek között nulla vagy több paraméterük van (akár 255 is); azokat az értékeket, amelyeket a függvényeknek a műveletek végrehajtásához kell megadnunk, a függvény paramétereinek vagy argumentumainak nevezzük;
- a paraméterek általában pontosvesszővel vannak elválasztva egymástól;
- ha hibásan adjuk meg a függvényt, hibaüzenetet kapunk;
- ha paraméterként egy folytonos tartományt adunk meg egy függvénynek, akkor elég megadnunk a tartomány bal felső sarkában és jobb alsó sarkában levő cellákat, kettősponttal elválasztva (pl. A2:C8);

- kattintsunk a kívánt cellába, ahova a függvényhívás kerül (nem kell duplán kattintani).

Függvénybeszúrási lehetőségek:

1. Használhatjuk a „Függvény beszúrása” gombot (f_x) a szerkesztőlécen, vagy a Képletek (Formulas) lapon a Függvény beszúrása gombot, vagy a Shift+F3 billentyűkombinációt, vagy a Kezdőlap (Home) „Autoszum” gomb Σ jobb oldalán levő legördítő háromszöggel is elérhetjük a függvénybeszúrást:

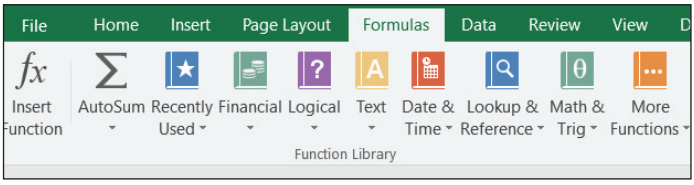


4.1.1. ábra. *Függvény beszúrása párbeszédpanel*

- ha nem tudjuk, hogy hol keressük a függvényt, akkor a „Függvény beszúrása” panelen a „Függvény keresése:” (Search for a function:) mező segítségét is kérhetjük; alatta „A függvény kategóriák:”-ból (Or select a category:) választhatunk, az alsó részen a kategóriákhoz tartozó függvények közül válogathatunk;
- ha a listába kattintunk és ott egy karaktert leütünk, akkor a listán az azzal a karakterrel kezdődő sorra lép (ha van ilyen);
- ha rákattintunk az egyik függvény nevére, akkor a panel alján egy rövid leírást ad róla;
- a függvény nevére kettőt kattintva továbblép a „Függvényargumentumok” (Function Arguments) ablakba, ahol a kötelezően megadandó paraméterek nevét mindig félkövér betűvel írja ki;
- a beírt paramétereket ki is értékeli és kiírja a paraméterek jobb oldalára;
- a lap alján bal oldalon a formázott végeredményt láthatjuk;

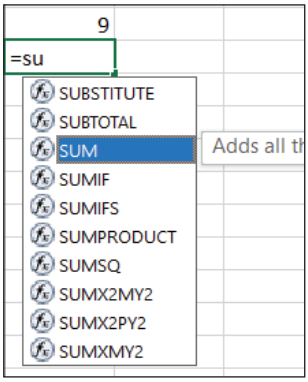
- beírás közben a szerkesztőléc elején levő „Név mező” (Name box) területen a legördíthető listából könnyebben is választhatunk az utoljára használt függvények közül; ez a lehetőség függvények egymásba ágyazására is használható.

2. A „Képletek” (Formulas) lapról az egyes függvénykategóriák közvetlenül is elérhetők:



4.1.2. ábra. *Függvénytár*

3. Egyenlőségjellel kezdve kézzel beírjuk a függvényt (pl. `=SUM(A1:A19)`).
4. Egyenlőségjellel kezdve elkezdjük kézzel beírni a függvény nevét,



4.1.3. ábra. *Az „SU”-val kezdődő függvények listája*

majd a megjelenő függvénylistából kiválasztjuk a kívánt függvényt (duplán kattintva rá),

4.1.4. ábra. *A kívánt függvény, egyelőre paraméter nélkül*

ezután a paraméterlista megadása következik, ami történhet kézzel beírva, vagy a szerkesztőlécen az fx gombra kattintva, melynek hatására megjelenik a „Függvényargumentumok” (Function arguments) panel.

Függvénykategóriák:

- A legutóbb használt (Most Recently Used) kategória: az utolsó pár függvényt mutatja meg, feltételezve, hogy nagyjából azonos függvényeket használunk egy táblázaton belül
- Mind (All) kategória: az összes függvényt felsorolja (ABC sorrendben)
- Pénzügyi (Financial)
- Dátum és idő (Date & Time)
- Matematikai és trigonometriai (Math & Trig)
- Statisztikai (Statistical)
- Mátrix (Matrix)
- Adatbázis (Database)
- Szöveg (Text)
- Logikai (Logical)
- Információ (Information)
- Tervezés, Műszaki (Engineering)
- Kocka (Cube)

4.2. A leggyakoribb függvények

- **SUM –** a függvény összeadja az argumentumaival meghatározott számokat

Szintaxisa:

```
=SUM(szám1 [,szám2,…])
```

A `szám1, szám2,…` argumentumok lehetnek egyedi értékek, cellahivatkozások, tartományok; általában egy vagy több tartomány, jellemzően egy sor- vagy oszloptartomány. Ha az argumentumok között szerepel nem számot tartalmazó cella is, azt figyelmen kívül hagyja.

Példák:

Tekintsük a következő `A1:B5` tartományt:

	A	B
1	1	
2	2	4
3	alma	5
4	4	
5	5	korte

4.2.1. ábra. *Példaadatok*

```
=SUM(A1:A5) → 12
=SUM(B1:B5) → 9
=SUM(B2:B3,A4:A5) → 18
```

• **AVERAGE –** a függvény argumentumainak az átlagát számolja ki
Szintaxisa:

```
=AVERAGE(szám1 [,szám2,…])
```

Az üres cellákat és szövegeket figyelmen kívül hagyja. Összeadja az argumentumaival meghatározott számokat, majd elosztja a számok számával.

Példák (a 4.2.1. ábra adataival):

```
=AVERAGE(A1:A5) → 3
=AVERAGE(B1:B5) → 4.5
=AVERAGE(B2:B3,A4:A5) → 4.5
```

• **COUNT –** megszámolja, hogy a paraméterként kapott tartományban hány számot tartalmazó cella van
Szintaxisa:

```
=COUNT(érték1 [,érték2,…])
```

Példa (a 4.2.1. ábra adataival):
`=COUNT(A1:B5) → 6` (6 számot tartalmazó cella van)

• **COUNTA –** megszámolja, hogy a paraméterként kapott tartományban hány nem üres cella van
Szintaxisa:

```
=COUNTA(érték1 [, érték2] ...)
```

Példa (a 4.2.1. ábra adataival):
`=COUNTA(A1:B5) → 8` (8 nem üres cella van)

• **COUNTIF** – megszámolja, hogy a paraméterként kapott tartományban hány cella teljesíti a paraméterként kapott feltételt
Szintaxisa:

```
=COUNTIF(tartomány, kritérium)
```

Példák (a 4.2.1. ábra adataival):
`=COUNTIF(A1:B5;4) → 2` (2 darab 4-est tartalmazó cella van)

`=COUNTIF(A1:B5;"<=4")` → `4` (4 darab 4-est vagy annál kisebb számot tartalmazó cella van)

- **MAX** – a paraméterként megadott tartományban levő legnagyobb értéket adja vissza.

Szintaxisa:

```
=MAX(szám1 [,szám2,…])
```

Példa (a 4.2.1. ábra adataival):
`=MAX(A1:B5)` → `5` (az `A1:B5` tartományban levő legnagyobb szám az 5-ös)

- **MIN** – a paraméterként megadott tartományban levő legkisebb értéket adja vissza.

Szintaxisa:

```
=MIN(szám1 [,szám2,…])
```

Példa (a 4.2.1. ábra adataival):
`=MIN(A1:B5)` → `1` (az `A1:B5` tartományban levő legkisebb szám az 1-es)

- **SUMIF** – a megadott feltételnek eleget tevő cellákban található értékeket adja össze, vagy a megadott feltételnek eleget tevő celláknak megfelelő értéteket adja össze az összegtartományban

Szintaxisa:

```
=SUMIF(tartomány, kritérium [, összegtartomány])
```

Példák:
Tekintsük a következő `A1:B7` tartományt:

	A	B
1	Nem	Kiadás
2	nő	2000
3	férfi	154
4	nő	1540
5	nő	1200
6	férfi	230
7	nő	1580

4.2.2. ábra. *Példa kiadásokra nemek szerint*

`=SUMIF(B2:B7,”>1500”)` → `5120` (az 1500-at meghaladó kiadások összege 5120)

`=SUMIF(A2:A7,”nő”,B2:B7)` → `6320` (a nők összkiadása 6320)

• **PRODUCT –** argumentumainak szorzatát adja
Szintaxisa:

```
=PRODUCT(szám1 [,szám2,…])
```

Példa (a 4.2.1. ábra adataival):
`=PRODUCT(B2:B3)` → `20`

• **TODAY –** az aktuális dátumot írja ki
Példa: =TODAY() → 16/3/2021

• **WEEKDAY –** megadja, hogy a paraméterként kapott dátum a hét hányadik napjára esik (ha azt szeretnénk, hogy hétfőt vegye a hét első napjának, akkor a függvénynek második paraméterként 2-est adjunk)

Példa: `=WEEKDAY(16/3/2021;2)` → 2 (2021. március 16. a hét második napjára, vagyis keddre esik)

4.2.1. Kitűzött feladatok

1. Adjuk meg, hogy a hét hányadik napján születtünk.
2. Számítsuk ki, hány napot éltünk.
3. Készítsük el az alábbi táblázatot. Számítsuk ki az Áfa értékeket, a Bruttó egységárakat, a Kedvezményeket (ezeket a Bruttó egységárból képezzük), az Eladási árat (a Bruttó egységár és a Kedvezmény különbségéből) és az Összesített értékeket.

	A	B	C	D	E	F
1	Árunyilvántartás					
2	Megnevezés	Nettó egységár	Áfa érték	Bruttó egységár	Kedvezmény	Eladási ár
3	Notebook	5.000 RON	950 RON	5.950 RON	714 RON	5.236 RON
4	Monitor	950 RON	181 RON	1.131 RON	136 RON	995 RON
5	Processzor	1.100 RON	209 RON	1.309 RON	157 RON	1.152 RON
6	Tablet	480 RON	91 RON	571 RON	69 RON	503 RON
7	Videokártya	1.850 RON	352 RON	2.202 RON	264 RON	1.937 RON
8	Összesen	9.380 RON	1.782 RON	11.162 RON	1.339 RON	9.823 RON
9	Áfa	19%				
10	Kedvezmény	12%				

4.2.1.1. ábra. *Árunyilvántartás példa*

4. Készítsük el az alábbi táblázatot, majd töltsük ki a hiányzó adatokat. A januári eladások száma adott (db). Februárban minden italból 50%-kal többet adtak el, mint januárban, márciusban pedig minden italból pontosan 5-tel kevesebbet adtak el, mint februárban. Számítsuk ki a bevételeket havonta italokra lebontva, havonkénti összesítésben, valamint a három hónapra vonatkozóan összesen. Az árutételenkénti bevételi összegeknek csak az egész részét jelenítsük meg. Havonkénti összesítésben határozzuk meg az eladott italok számát is.

	A	B	C	D	E	F	G	H	I
1	ITALFOGYASZTÁS								
2	Megnevezés	Egységár	Január		Február		Március		Összes
3			db	bevétel	db	bevétel	db	bevétel	bevétel
4	Rövid kávé	RON 4.00	120						
5	Hosszú kávé	RON 5.00	150						
6	Cappucino	RON 6.00	50						
7	Moccaccino	RON 6.00	75						
8	Kakaó	RON 5.00	25						
9	Tea	RON 4.00	58						
10	Jeges tea	RON 6.00	82						
11	Narancslé	RON 8.00	75						
12	Limonádé	RON 10.00	102						
13	Összesen								

4.2.1.2. ábra. *Italfogyasztás példa*

5. Készítsük el az alábbi táblázatot. Töltsük ki a hiányzó adatokkal. A kitöltés során használjunk másolást.

	A	B	C	D	E	F	G	H
1	Sorszám	Névsor	Matematika	Informatika	Angol	Átlag	Legnagyobb jegy	Legkisebb jegy
2	1	Asztalos Imre	5	7	8			
3	2	Balázs Dóra	10	10	9			
4	3	Bálint Géza	9	8	8			
5	4	Dobri Enikő	10	6	8			
6	5	Egri Levente	6	7	7			
7	6	Elekes Nóémi	7	9	8			
8	7	Farkas Előd	8	5	9			
9	8	Haáz Jolán	5	6	7			
10	9	Imre Endre	9	10	10			
11	10	Kolozsi Péter	10	9	8			
12	11	Kun Jenő	10	8	8			
13	12	Lakatos Irma	6	7	10			
14		Átlag:						

4.2.1.3. ábra. *Tanulmányi jegyek példa*

6. Az előző táblázatot használva számítsuk ki, hány személynek van 10-ese a különböző tantárgyakból, illetve mindhárom tantárgyból.

7. A táblázat egy meteorológiai állomás néhány napi adatait tartalmazza. A táblázat alapján adjuk meg:

a) Mennyi volt a legnagyobb napi középhőmérséklet?
b) Mennyi volt a napi középhőmérsékletek átlaga?
c) Mekkora volt a legkisebb meleg fokban?
d) Mekkora volt a napi hőmérséklet-ingadozás?
e) Hány olyan nap volt, amikor a napsütéses órák száma meghaladta a nyolc órát?
f) Összesen hány órát sütött a nap, amikor a minimális napi hőmérséklet 21 C° alatti volt?
g) A napok hány %-ában fordult elő eső?

4.2.1.4. táblázat. *Meteorológiai adatok példa*

Dátum	Napi közép-hőmérséklet	Esett-e eső	Maximális napi hőmérséklet	Napsütéses órák száma	Minimális napi hőmérséklet
16.05.2021	25	igen	30	6	21
17.05.2021	24	igen	32	4	20
18.05.2021	28	nem	30	7	22
19.05.2021	30	igen	34	9	24
20.05.2021	31		35	9	26
21.05.2021	30	nem	33	10	20
22.05.2021	29	igen	32	8	19
23.05.2021	30	nem	35	9	22

4.3. Logikai függvények

Az előzőkben bemutatott logikai műveletek igencsak hasznosak, viszont számos esetben előfordul, hogy több logikai művelet eredményét kell összefűzni és/vagy ezek eredményei szerint feltételesen dönteni. Az ilyen esetekre az Excel logikai függvényeket is biztosít, melyek az *AND* (logikai és), *OR* (logikai vagy) és az *IF* (HA döntési függvény).

4.3.1. AND függvény

A logikai *AND* (ÉS) függvény több logikai érték vagy feltétel eredményeinek egyesítésére szolgál. Két vagy több paramétert kell megadni, melyek logikai értékek

kell legyenek, vagy olyan műveletek, illetve függvények, melyek logikai értéket adnak vissza:

```
AND(cella 1 / érték 1, cella 2 / érték 2, …)
```

Visszatérési értéke egy logikai TRUE (igaz) vagy FALSE (hamis) a bemenő paraméterek szerint, éspedig ha minden paraméter **igaz** értékű, akkor az *AND* függvény **igaz** értéket ad vissza, viszont ha csak egyetlen érték is **hamis**, akkor **hamis** értéket ad vissza.

Példa: Írjuk fel és értékeljük ki a következő mondatot Excelben: 1 kisebb, mint 2 és 2 kisebb, mint 3. Mivel azt mondtuk, hogy „és", ezért a logikai *AND* (ÉS) függvényt kell használni, paraméterei pedig a két kifejezés lesz:

```
=AND( 1 < 2, 2 < 3)
```

Ezt beírva az A1 cellába, az Excel kiértékeli ezt a képletet. Mivel az *AND* függvény paramétereibe írt mindkét kifejezés igaz lesz, ezért maga az *AND* függvény TRUE (igaz) értéket fog visszaadni. Ez a visszatérített érték megjelenik az A1 cellában.

A következőben módosítsuk az egyik kifejezést, hogy hamis legyen, vagy adjunk hozzá harmadik paraméternek egy hamis kifejezést, például $2 > 3$. Ebben az esetben, mivel az egyik paramétere az *AND* függvénynek hamis, az A1 cellában FALSE (hamis) értéket fogunk látni.

4.3.2. OR függvény

A logikai *OR* (*VAGY*) függvény, az *AND* függvényhez hasonlóan, szintén több logikai érték vagy feltétel eredmény egyesítésére szolgál. Azonban az *AND* függvénnyel ellentétben, ez akkor ad vissza igaz értéket, ha legalább egy paramétere igaz értékű, valamint hamisat, ha minden egyes paramétere hamis. Szintaxisa:

```
=OR(cella 1 / érték 1, cella 2 / érték 2, …)
```

Példa: Írjuk fel és értékeljük ki a következő mondatot Excelben: 1 nagyobb, mint 2 vagy 1 kisebb, mint 2. Írjuk ezt az A2 cellába, ami a következőképpen fog kinézni:

```
=OR(1 > 2, 1 < 2)
```

Mivel az első feltétel hamis, a második pedig igaz, ezért az *OR* függvény TRUE (igaz) értéket ad vissza. Ha a második kifejezést megváltoztatjuk arra, hogy „$3 < 2$", ami szintén hamis, akkor az *OR* visszatérési értéke, mivel minden egyes paramétere hamis, FALSE (hamis) lesz.

4.3.3. IF függvény

Logikai értékek szerinti döntésre szolgál az *IF* (*HA*) függvény. Mint neve is mondja, „ha”, egy adott feltétel szerint végez döntést: ha igaz vagy ha hamis az adott érték vagy kifejezés értéke. Szintaxisa:

```
=IF(cella / kifejezés, ha igaz, [ha hamis])
```

Első paramétere lehet egy: logikai érték, kifejezés, melynek eredménye logikai érték, illetve egy cella, amely ilyen értéket vagy kifejezést tartalmaz. Fontos, hogy logikai érték legyen, különben az *IF* függvény nem tud dönteni és hibát ad vissza.

A második és a harmadik paramétere az *IF* függvénynek az az érték, amit visszaad, hogyha az első paraméter értéke igaz, illetve ha ez hamis. A harmadik paraméter opcionális, viszont ha ezt nem adjuk meg, és a megadott első paraméter hamis, akkor maga az *IF* függvény FALSE értéket fog visszaadni. Ha az *IF* direkt egy cellában van meghívva, ilyenkor a FALSE érték természetesen megjelenik a cella értékeként.

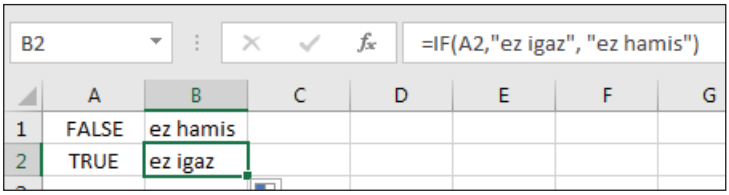
Példa: Ha egy cellába beírunk egy összehasonlító műveletet, akkor ezt az Excel kiértékeli, és TRUE vagy FALSE lesz a cella megjelenített értéke. Példaként vegyünk egy igaz és egy hamis kifejezést, és logikai eredményük szerint jelenítsük meg a cellában a következő két szöveget: „ez igaz”, valamint „ez hamis”.

Legyen az első, hamis kifejezés: $1 > 2$, a második pedig $1 < 2$. Ezt meg lehet oldani kétféleképpen is: egy képlet segítségével vagy két lépésben, egy segédcella segítségével.

Először oldjuk meg két lépésben: használjuk az A1, A2 és a B1, B2 cellákat. Az A1 és A2 cellába írjuk be a kiértékelendő műveletünket, éspedig az „=1 > 2” és „=1 < 2”. A B1 cellába írjuk be az *IF* függvényünket, első paraméternek adjuk meg az A1 cellát, majd a második paraméternek azt az értéket, ha a kifejezés igaz („ez igaz” szöveg), míg a harmadik paraméternek az értéket, ha ez a kifejezés hamis („ez hamis” szöveg). Figyeljünk a paraméterek elválasztására és a zárójelezésre.

```
=IF(A1, ”ez igaz”, ”ez hamis”)
```

Ha készen vagyunk, akkor ezt tudjuk lefele alkalmazni, így a B2-be is bekerül a képletünk. Mivel lefele alkalmaztuk, ezért itt az A1 cellacímet átírja nekünk az Excel A2-re, és ennek értékével dolgozik az IF függvényünk.



4.3.3.1. ábra. *IF függvényt bemutató feladat*

4.3.4. Egyszerű gyakorlófeladat

Az A és B oszlopokba írjuk fel két logikai változó értékeinek kombinációit (0-0, 0-1, 1-0 és 1-1), ezután végezd el a következőket:

- A C oszlopban használd az *AND* (ÉS) függvényt az A és a B oszlop értékeire.
- A D oszlopban használd az *OR* (*VAGY*) függvényt az A és a B oszlop értékeire.
- A C és D oszlopokban megkaptuk a TRUE vagy FALSE értékeit az AND és az OR függvényeknek. Használva az *IF* függvényt, ezek értékeit írjuk ki szavakban, azaz „Ez igaz” vagy „Ez hamis” szövegek formájában az E és az F oszlopokba. Ezt oldd meg kétféleképpen:
  - Mivel a C és D oszlopok eredményei eleve logikai értékek, ezért ezt lehet használni az *IF* függvény feltétel paraméterében.
  - Az *IF* függvénybe ágyazd be az *AND* és az *OR* függvényeket.

A feladat formája:

	A	B	C	D	E	F
1	Értékek		AND és OR		Szavakban	
2	**A**	**B**	**AND**	**OR**	**AND**	**OR**
3	0	0	FALSE	FALSE	Ez hamis	Ez hamis
4	0	1	FALSE	TRUE	Hamis	Igaz
5	1	0	FALSE	TRUE	Hamis	Igaz
6	1	1	TRUE	TRUE	Igaz	Igaz

4.3.4.1. ábra. *AND, OR és IF függvényeket bemutató feladat*

Megoldás:

• C oszlop megoldása: `=AND(A1, B1)`, a D oszlop megoldása: `=OR(A1, B1)`, és mindkét cellát egyszerre kijelölve alkalmazzuk a két képletet lefele. A cellacímeket az Excel megfelelően átírja, így minden sorban a megfelelő A és B értékek lesznek használva.

• E oszlop megoldása: `=IF(C3,"Ez igaz","Ez hamis")`, az F oszlopé pedig: `=IF(D3,"Ez igaz","Ez hamis")`. Itt is mindkét cellát egyszerre kijelölve lehet lefele alkalmazni. Az összetett képlet megoldása: `=IF(AND(A1, B1),"Ez igaz", „Ez hamis")` , illetve az *OR* függvény használatával. Ebben az esetben is lehet a két cella képletét egyszerre alkalmazni lefele.

4.3.5 Gyakorlófeladatok

4.3.5.1. Kiadáslista

Adott a következő táblázat, mely bizonyos termékek súlyát és a rendelt darabszámot tartalmazza. Minden termék külön csomagban érkezik, viszont a szállítócég extraköltséggel terhel, ha túllépjük a megengedett súlyhatárt egy adott csomagra.

	A	B	C	D	E	F	G
1	Vásárolt Termék	Darabszám	Súly	Összesen	Figyelmeztetés		Súlyhatár
2		10	0.82	?	?		12.50
3		18	0.36	?	?		
4		25	1.19	?	?		
5		28	0.76	?	?		
6		180	0.05	?	?		

4.3.5.1.1. ábra. *Kiadáslista feladat*

Feladat: Az E oszlopba írjál egy lefele alkalmazható képletet, mely a megengedett súlyhatár túllépése esetén az adott sorba egy „extraköltség" figyelmeztető szöveget ír. Mivel a szállítócég váltása esetén a képletet nem módosítjuk, ezért a súlyhatár értékét vegyük a G2 cellából.

Útmutató:

Először a D oszlopba az összsúly értékét kell kiszámoljuk. Mivel a termékek adatai soronként vannak, ezért itt a B és a C oszlop celláit összeszorozzuk. Ha ez megvan, akkor ezt a képletet alkalmazzuk lefele, így a D oszlopban minden termék összsúlyát kiszámoltuk.

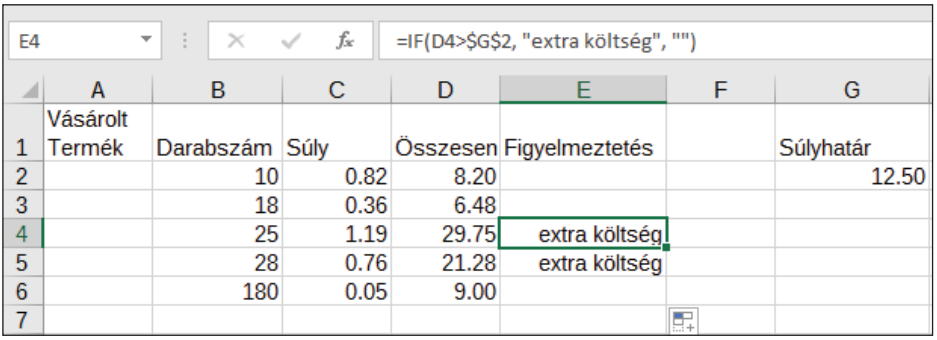
Ha a D oszloppal készen vagyunk, következhet az E oszlop. Mivel dönteni kell, ezért az IF függvényt fogjuk használni az E2 cellában. Feltételnek a termék D2 cella értékét hasonlítjuk a súlyhatárral, azaz a G2 cellával. Ha az E2 nagyobb, mint a G2, akkor az *IF* adja vissza az „extraköltség" figyelmeztető szöveget, különben egy üres szöveget (üres szöveg: „", vigyázat, dupla idézőjel, nem pedig négy darab aposztróf). **Figyelem**: ha nem adunk vissza üres szöveget, akkor a FALSE

értéket adja vissza az *IF* függvény. Ha készen van a képlet, akkor lehet alkalmazni lefelé a többi sorra.

Ellenőrizd, hogy az eredmény minden sorban helyes-e! A 4-es és 5-ös sorban kell megjelenjen a figyelmeztetés. Ha ez nem jelenik meg, ellenőrizd az E oszlop hibás sorainak celláiban levő képleteket. Ha nem volt rögzítve a G2 cella, akkor lefele való alkalmazáskor az Excel ebben a címben is növelte a sorszámot, azaz G3, G4 stb. lett belőle, így helytelen az összehasonlítás.

Megoldások:

- D oszlop: `=B2*C2`
- E oszlop: `=IF(D2>$G$2, "extra költség", "")`



	A	B	C	D	E	F	G
1	Vásárolt Termék	Darabszám	Súly	Összesen	Figyelmeztetés		Súlyhatár
2		10	0.82	8.20			12.50
3		18	0.36	6.48			
4		25	1.19	29.75	extra költség		
5		28	0.76	21.28	extra költség		
6		180	0.05	9.00			
7							

4.3.5.1.2. ábra. *Kiadáslista feladat megoldva*

4.3.5.2. Rendezvény

Legyen a következő táblázat, mely egy rendezvényen részt vevők névsorát tartalmazza, illetve azt, hogy az illető résztvevő előadó és/vagy diák. A rendezvényre a belépés 200 RON. Az előadók 75% kedvezményt kapnak ebből, míg a diákok 25%-ot. A rendezvény elősegítésére készítsünk egy táblázatot, ahol soronként megadva a résztvevőkre, hogy előadók és/vagy diákok, kiszámolja a fizetendő belépési díjat. Ha egy résztvevő előadó vagy diák, akkor a megfelelő cellába be lesz írva az „Igen” szöveg, különben ez üres.

Fontos, hogy a D oszlopba olyan képletet írjunk, ami lefele alkalmazható, így könnyedén lehessen új sorokat hozzáadni. Az esetleges jegyár, illetve kedvezmények változására felkészülve, ezek értékeit minden esetben vegyük a megadott H1, H3 és H4 cellákból. Így, ha ezek változnak, akkor nem kell a képleteket javítani, hanem elegendő a megfelelő cella értékét módosítani.

	A	B	C	D	E	F	G	H
1	**Résztvevő neve**	**Előadó**	**Diák**	**Belépő**			Jegy ára:	**RON 200.00**
2	Kis Pista	Igen		?			Kedvezmények:	
3	Halász Jenő			?			előadó	**75%**
4	Nagy Margit		Igen	?			diák	**25%**
5	Szabó János	Igen	Igen	?				
6	Hagymási Csilla			?				
7								
8			Összesen	?				

4.3.5.2.1. ábra. *Rendezvény feladat*

Ha a képletek bonyolultak, akkor használhatunk segédcellákat, melyeket a nyomtatás érdekében el lehet rejteni, így nem fognak látszani a papíron. Hogy az oszlopszámozás folyamatosságát ne szakítsuk meg, ezeket a segédcellákat mindig vegyük a feladat cellái után (ezektől jobbra).

Útmutató: oldjuk meg két lépésben: használjunk segédoszlopot először a kedvezmények megállapítására, majd a D oszlop celláiban a fizetendő összeget a segédoszlop celláinak segítségével számítsuk ki.

Mivel dönteni kell, ezért használjuk az *IF* függvényt. A két esetünket különkülön tudjuk vizsgálni, ezért el tudjuk kerülni, hogy két *IF* függvényt egymásba ágyazzunk. Vegyük az E és az F oszlopokat segítségnek: az E oszlopban számoljuk ki az előadói kedvezményt egy résztvevőnek, míg az F oszlopban számoljuk ki a diákkedvezményt, amennyiben ezek járnak.

- E oszlop: az *IF* függvény feltételeként vizsgáljuk meg az adott sorban a B oszlopbeli cella értékét: ha itt „Igen” van írva, akkor a kedvezmény értékét vegyük a H3 cellából, különben az *IF* adja vissza a 0 értéket.
- F oszlop: hasonlóan az előző oszlophoz, vizsgáljuk meg a C oszlop celláját: amennyiben „Igen” van írva, az *IF* függvény adja vissza a H4 cella értékét, különben a 0 értéket.

Fontos, hogy a H3 és H4 cella címét rögzítsük, mert ezeket a képleteket lefele fogjuk alkalmazni, így, ha a H3 címből H4, majd H5 stb. lesz, akkor helytelenül fog számolni.

Ha megvannak a segédoszlopok, akkor nem marad más hátra, mint hogy a jegy árából kivonjuk a kedvezmények összegét (mivel két kedvezményünk lehet). Itt is fontos a jegy árának cellacímét rögzíteni.

Megoldás:

- E oszlop: `=IF(B2="Igen", $H$3,0)`
- F oszlop: `=IF(C2="Igen", $H$4,0)`
- D oszlop: Ezt kétféleképpen is ki lehet számolni, például a D2 cellában: `$H$1 - $H$1*(E2 + F2)` vagy `$H$1 * (100% - (E2 + F2))`.

	A	B	C	D	E	F	G	H
1	**Résztvevő neve**	**Előadó**	**Diák**	**Belépő**			Jegy ára:	**RON 200.00**
2	Kis Pista	Igen		RON 50.00	75.00%	0.00%	Kedvezmények:	
3	Halász Jenő			RON 200.00	0.00%	0.00%	előadó	**75%**
4	Nagy Margit		Igen	RON 150.00	0.00%	25.00%	diák	**25%**
5	Szabó János	Igen	Igen	RON 0.00	75.00%	25.00%		
6	Hagymási Csilla			RON 200.00	0.00%	0.00%		
7								
8			Összesen	RON 600.00				

4.3.5.2.2. ábra. *Rendezvény feladat megoldva*

4.3.5.3. Egyszerű vámolás

Importált termékekre a vám a következő: 100 RON felett 19%, 481.20 RON felett pedig extra 7.20%. Adott az alábbi táblázat, ahol a B oszlopban adott egy termék egységára, a C oszlopban pedig ennek a darabszáma.

	A	B	C	D	E	F
1	Tobb mint:	RON 100.00	RON 481.20			
2	Vam:	19.00%	7.20%			
3						
4	Termék	Ár	Darabszám	Érték	Vám	Összesen
5		RON 45.00	1	?	?	?
6		RON 60.00	4	?	?	?
7		RON 98.00	4	?	?	?
8		RON 1,650.00	1	?	?	?
9		RON 180.00	3	?	?	?
10		RON 90.00	2	?	?	?
11						
12				Osszesen:	?	?

4.3.5.3.1. ábra. *Egyszerű vámolás feladat*

Feladat: a D oszlopban számold ki soronként a rendelt termékek összérték, majd az E oszlopban az erre az értékre fizetendő vámot. Az F oszlopban összesítsd termékenként a fizetendő összeget és a vám értékét. Az E12 és F12 cellákban összesítsd az adott oszlopok értékeit. Írj mindenhova lefele alkalmazható képleteket, valamint a vámértékeket vedd a B1, B2 és a C1, C2 cellákból.

Útmutató:

- D oszlop: meg kell szorozni a termék árát a darabszámmal, azaz a D5 cellában: `=B5*C5`, majd ezt alkalmazzuk lefele.
- E oszlop: itt dönteni kell, hogy mennyi a vám, éspedig három esetünk van az érték szerint: kevesebb mint 100 RON, 100 RON és 481.20 RON között, valamint 481.20 RON felett. Használjunk egy segédoszlopot (G oszlop), ahol először megállapítjuk a vám értékét %-ban, majd az E oszlopban ezt használva, kiszámoljuk RON-ban is. Ha a G oszlop megvan, akkor itt egyszerűen kiszámoljuk a vámot, éspedig: `=G5*D5`.
- G segédoszlop: mivel három feltétel van, egymásba lehet ágyazni két *IF* függvényt, éspedig ha a termék összértéke kisebb, mint 100 RON, akkor 0, különben és ide jön a második *IF*, ha több, mint 481.20 RON (C1 cellát használjuk), akkor vegyük a vám értékét `B2+C2`-nek, különben csak az az eset maradt, hogy 100 RON és 481.20 RON között van, ami esetben a vám értékét vesszük a B2 cellából. Mivel a képletet akarjuk lefele is alkalmazni, ezért rögzíteni kell a megfelelő cellákat!
- F oszlop: itt összeadjuk a D és az E oszlop celláit, azaz `=D5+E5`.
- Az E12 cellában *SUM* segítségével összeadjuk az oszlop celláit, éspedig: `=SUM(E5:E10)`, míg az F12 cella képlete: `=SUM(F5:F10)`.

Megoldás:

1. G segédoszlop: `=IF(D5<$B$1,0, IF(D5>$C$1,$B$2+$C$2,$B$2))`

G5 =IF(D5<$B$1,0,IF(D5>$C$1,$B$2+$C$2,$B$2))

	A	B	C	D	E	F	G
1	Tobb mint:	RON 100.00	RON 481.20				
2	Vam:	19.00%	7.20%				
3							
4	Termék	Ár	Darabszám	Érték	Vám	Összesen	
5		RON 45.00	1	RON 45.00	RON -	RON 45.00	0.00%
6		RON 60.00	4	RON 240.00	RON 45.60	RON 285.60	19.00%
7		RON 98.00	4	RON 392.00	RON 74.48	RON 466.48	19.00%
8		RON 1,650.00	1	RON 1,650.00	RON 432.30	RON 2,082.30	26.20%
9		RON 180.00	3	RON 540.00	RON 141.48	RON 681.48	26.20%
10		RON 90.00	2	RON 180.00	RON 34.20	RON 214.20	19.00%
11							
12				Osszesen:	RON 728.06	RON 3,775.06	

4.3.5.3.2. ábra. *Egyszerű vámolás feladat megoldva*

4.3.5.4. Rendelés táblázat

Legyen a következő táblázat, mely kiszállítandó rendeléseket ábrázol. A termékek neve nem lényeges a feladat megoldásánál, ezért ezek a cellák lehetnek üresek is.

	A	B	C	D	E
1	Kedvezmeny:	Min. Darabszám:	Min. Végösszeg	Ingyenes Szállítás	
2		10	500.00 RON	1,000.00 RON	
3		1.75%	2.50%		
4					
5	Termék Neve	Ár	Darabszám	Összeg	Kedvezmény
6		RON 100.00	2.00	?	?
7		RON 50.00	10.00	?	?
8		RON 150.00	2.00	?	?
9		RON 89.10	30.00	?	?
10		RON 2.23	120.00	?	?
11					
12			Összesen:	?	?
13			Fizetni:	?	
14			Ingyen kiszállítás?	?	

4.3.5.4.1. ábra. *Rendelési táblázat feladat*

Feladat:

1. Vidd fel egy új munkalapra a fenti táblázatot, a megfelelő formázással (szöveg, pénznem, tizedesek stb.).

2. Számold ki az „Összeg” oszlopban soronként a fizetendő összegeket.

3. A D12 cellában összesítsd a fenti sorok összegeit.

4. A kedvezmény oszlop (E) celláiba számold ki az adott sor termékei után járó kedvezményt pénznemben. A kedvezmény a következő részekből áll (ezek nem zárják ki egymást, a kedvezmény értékeit a B2, B3, illetve a C2, C3 cellákból kell venni):

a) 10 darab után 1.75% kedvezmény jár az adott termék értékéből,

b) 500 RON után plusz 2.50% kedvezmény jár az adott termék értékéből.

5. Az E12 cellában összesítsd ezeket a kedvezményeket.

6. A C13 cellában számold ki a rendelésre vonatkozó konkrétan fizetendő összeget, azaz a kedvezményeket leszámítva az összárból.

7. A C14 cellában határozd meg, hogy jár-e az ingyenes szállítás a rendelésre vagy sem. A cellában „Igen” vagy „Nem” szöveg jelenjen meg.

Megoldás:

2-3. „Összeg” oszlop (D6-D10 cellák): ez egyszerű összegszámolás, azaz a termék árát megszorozzuk a darabszámmal, tehát: `=B6*C6`, majd ezt a képletet alkalmazzuk lefele a többi cellára. A D12 cellában a *SUM* függvény segítségével összeadjuk az előbb kiszámolt értékeket, azaz `=SUM(D6:D10)`.

4. Kedvezmények: itt már dönteni kell, éspedig két feltétel alapján. Mivel összegről van szó, ezért nem szükséges egymásba ágyazott IF függvényeket használni, hanem meg lehet oldani egyszerűbben is, két külön *IF* függvény segítségével, melyek visszatérési értékeit összeadjuk. Lássuk az első feltételt. Azt mondja a feladat, hogy ha több mint 10 darabot rendelünk egy termékből, akkor adunk 1.75% kedvezményt. Mivel ezek az értékek változhatnak, ezért ezekből a cellákból vegyük. Tehát itt kell használni egy *IF* függvényt, melynek segítségével megvizsgáljuk, hogy több mint 10 darab termék van-e az adott sorban vagy sem. Ha igen, akkor számolunk kedvezményt, különben viszont adjunk vissza 0 értéket. Ez nagyon fontos, mivel a két *IF* értékét összeadjuk, ezért minden esetben a két *IF* függvény visszatérési értéke szám kell legyen. Visszatérve, az első *IF* függvényt írjuk az E6-os cellába, ezért (adott sor adataival dolgozunk) ennek feltétele: `C6>=B2`. Ha ez a feltétel igaz, akkor visszaadjuk a kedvezményt, azaz: `D6*B3`, különben, mint említettük, `0`. Az első *IF* függvényünk a következő formájú lesz:

```
=IF(C6>=B2, D6*B3, 0)
```

Alkalmazzuk a képletet lefele, és ellenőrizzük az eredményt. Ebben a formában, a lefele való alkalmazáskor az Excel növelni fogja a sorszámokat a cellacímekben: A C6 és D6-ból C7 és D7 lesz (és így tovább), ami rendben van, viszont vegyük észre, hogy a B2 és a B3 is változni fog, éspedig B3 és B4 lesz (és így tovább), ami rossz eredményhez vezet. Ezért szükséges a nem változó cellacímeket rögzíteni, éspedig:

```
=IF(C6>=$B$2, D6*$B$3, 0)
```

Ez már helyes eredményt fog adni, viszont ez a kedvezménynek csak a fele. Most írjuk fel a másik *IF* függvényt is, és adjuk össze a kettőt. Mivel egyszerűek a függvények, fel lehet egyből írni egyetlen cellába. A második IF feltétele az, hogy a sorban kiszámolt összeg nagyobb-e vagy egyenlő 500 RON-nal, azaz `D6>=$C$2`. Mivel tudjuk, hogy a képletet lefele fogjuk alkalmazni, ezért a C2 cellát egyből rögzíteni lehet. Ha ez a feltétel igaz, akkor a kedvezményünk ezen része `D6*$C$3`, különben 0. Ha megvan a két feltételünk, adjuk össze, így a képlet a következő lesz:

```
=IF(C6>=$B$2, D6*$B$3, 0) + IF(D6>=$C$2, D6*$C$3, 0)
```

5–6. Ezek összegzése egyszerű. Az E12 cellában a *SUM* függvény segítségével összeadjuk a kedvezményeket, azaz `=SUM(E6:E10)`, majd a C13 cellában kivonjuk a fizetendő összegből (D12) ezt az összkedvezményt (E12): `=D12-E12`.

7. Ebben az esetben ismét döntenünk kell, hogy a kiszállítás ingyenes-e vagy sem. Ez a döntés természetesen az *IF* függvény segítségével történik, a feltételünk pedig, hogy a fizetendő összeg nagyobb-e, mint a D2 cella értéke vagy sem, és ennek függvényében jelenjen meg a cellában az „Igen” vagy a „Nem” szöveg. Az *IF* függvényünk a következőképpen fog kinézni:

```
=IF(D2<=D13, „Igen”, „Nem”)
```

Mivel ezt a képletet nem alkalmazzuk lefele, itt nem szükséges a D2 cellát rögzíteni.

	A	B	C	D	E
1	Kedvezmeny:	Min. Darabszám:	Min. Végösszeg	Ingyenes Szállítás	
2		10	500.00 RON	1,000.00 RON	
3		1.75%	2.50%		
4					
5	Termék Neve	Ár	Darabszám	Összeg	Kedvezmény
6		RON 100.00	2.00	200.00 RON	0.00 RON
7		RON 50.00	10.00	500.00 RON	21.25 RON
8		RON 150.00	2.00	300.00 RON	0.00 RON
9		RON 89.10	30.00	2,673.00 RON	113.60 RON
10		RON 2.23	120.00	267.60 RON	4.68 RON
11					
12			Összesen:	3,940.60 RON	139.54 RON
13			Fizetni:	3,801.06 RON	
14			Ingyen kiszállítás?	Igen	

4.3.5.4.2. ábra. *Rendelési táblázat feladat megoldva*

4.4. Hasznos függvény feltételes párjai

Számos esetben előfordul, hogy a fent említett gyakori függvényeket (*AVERAGE*, *SUM*, *COUNT* stb.) egy szelekción belül nem az összes cellára kell alkalmazni, hanem megadott feltételek mellett, csak bizonyos cellákra. Számos adat esetén időigényes kézzel számolni vagy akár lehetetlen is lehet, például több ezer sor esetén.

Ilyen esetekre biztosít az Excel ezen függvényekhez tartozó feltételekkel kiegészített párokat. Ezen függvények nevei az „IF” szóval végződnek, például *SUMIF*, *AVERAGEIF*, *COUNTIF* stb. Ezen függvények mind egy extra „feltétel” paraméterrel rendelkeznek, mely a számolandó cellák feltételét adja meg, azaz ha ez a feltétel teljesül, akkor a cellát figyelembe veszi és megszámolja, különben figyelmen kívül hagyja. A feltételt szintaktikai okok miatt kötelező módon szövegként kell megadni, ellenkező esetben az Excel külön műveletnek tekinti és megpróbálja végrehajtani ezt. Ugyanakkor ez a követelmény korlátozza is a feltételek sokoldalúságát, de ennek ellenére, egyszerű számolásokra, igencsak hasznosak ezek a függvények.

A fent említett függvényeknek csak egyetlen feltételt lehet megadni, és ez a feltétel a számolandó oszlopra vonatkozik. Ha több feltétel szerint szeretnénk szűrni vagy a feltétel más cellákra vonatkozik, mint a számolandó cellák, akkor rendelkezésre állnak az „IFS”-ben végződő függvények, például *MAXIFS*, *MINIFS*, *COUNTIFS*, *AVERAGEIFS*, *SUMIFS*. Ezek ugyanúgy számolnak, mint az „IF”-ben végződő párjuk, viszont több feltételt is lehet adni: mindegyiket szelekció, feltétel párosban. Nagy előnye, hogy a szelekciók különbözhetnek egymástól, például más-más oszlopokról van szó. Ahhoz, hogy egy cellát megszámoljanak, minden feltételnek egyszerre kell teljesülnie („ÉS” feltételek). „VAGY” feltételek esetére az Excel nem biztosít függvényt; ilyenkor minden feltételnek egy-egy külön függvényt kell használni, majd ezek eredményeit összeadni.

A feltétel részben használható elemek:

- Konkrét érték: ilyenkor csak azokat a cellákat számolja meg, ahol a cella értéke a feltételnek megadott érték.
- Logikai operátorokat: < (kisebb), > (nagyobb), <= (kisebb egyenlő), >= (nagyobb egyenlő), <> (nemegyenlő).
- Helyettesítő karaktereket: „?” (bármilyen karakter), „*” (akármennyi más karakter) „~” (speciális karakterek kivédése) jelek.

Ezek használatát a következő részekben tárgyaljuk. Példáknak vegyük a következő táblázatot:

	A	B
1	**Számla**	**Érték**
2	Gáz	RON 120.00
3	Villany	RON 98.00
4	Gáz	RON 240.00
5	Gáz	RON 182.00
6	Gáz	RON 360.00
7	Villany	RON 14.00
8	Villany	RON 840.00

4.4.1. ábra. *Példa adatok*

A függvények szemléltetésére az A oszlop értékei nem lényegesek, ezért ezek akár üresek is lehetnek.

4.4.1 COUNTIF, COUNTIFS függvények

A *COUNTIF, COUNTIFS* függvények a *COUNT* és az *IF* függvény egyesítése. Szintaxisaik:

```
=COUNTIF(szelekció, feltétel)
=COUNTIFS(szelekció, feltétel1, [szelekció2, feltétel2], ...)
```

A szelekció paraméter, mint a *COUNT* függvény esetében is, a megszámolandó cellákat jelenti. Plusz paraméter a *COUNT* függvényhez képest a fent említett feltétel paraméter.

Feladat: Számoljuk meg, hány olyan számlánk van, aminek értéke több mint 100 RON.

Megoldás: Tetszőleges cellába, a *COUNT*-hoz hasonlóan használjuk a *COUNTIF* függvényt, viszont feltételnek adjuk meg, hogy a számolandó cella értéke legyen több mint 100. Így a képlet a következő formájú lesz: `=COUNTIF(B2:B8, ">100")`, melynek eredménye `5` lesz.

4.4.2. MAXIF, MAXIFS, MINIF, MINIFS függvények

Mint már ismerjük, a „MAX” függvények a maximumot, a „MIN” függvények pedig a minimumot adják adott feltételek szerint. Szintaxisaik:

```
=MAXIF(szelekció, feltétel)
=MAXIFS(szelekció, feltétel1, [szelekció2, feltétel2], ...)
=MINIF(szelekció, feltétel)
=MINIFS(szelekció, feltétel1, [szelekció2, feltétel2], ...)
```

Feladat: Határozd meg a legkisebb, illetve a legnagobb gázszámlát.

Megoldás: Mivel a számla értéke a B oszlop celláiban található, a számla neve, ami a feltételünk pedig az A oszlop celláiban, ezért a *MINIFS* és a *MAXIFS* függvényeket kell alkalmazzuk. Megoldások:

- `=MINIFS(B2:B8,A2:A8,"=Gáz")`
- `=MAXIFS(B2:B8,A2:A8,"=Gáz")`

4.4.3. AVERAGEIF, AVERAGEIFS függvények

Az *AVERAGEIF* függvény átlagot számol egy megadott szelekció azon celláiból, amelyek értékei megfelelnek a megadott feltételnek. Szintaxisa:

```
=AVERAGEIF(számolandó szelekció, feltétel)
=AVERAGEIFS(számolandó szelekció, [feltétel szelekció1, feltétel1], ...)
```

Feladat: Számoljuk ki a 100 RON-nál nagyobb értékű számlák átlagát.

Megoldás: Tetszőleges cellába, az *AVERAGE*-hez hasonlóan használjuk az *AVERAGEIF* függvényt, viszont feltételnek adjuk meg, hogy a számolandó cella értéke legyen több mint 100. Így a képlet a következő formájú lesz: `=AVERAGEIF(B2:B8, „>100”)`, melynek eredménye `1742.00 RON` lesz.

4.4.4. SUMIF, SUMIFS függvények

Az *SUMIF* függvény összeget számol egy megadott szelekció azon celláiból, amelyek értékei megfelelnek a megadott feltételnek. Szintaxisa:

```
=SUMIF(számolandó szelekció, feltétel)
=SUMIFS(számolandó szelekció, [feltétel szelekció1, feltétel1], ...)
```

Feladat: Számoljuk ki a 100 RON-nál nagyobb értékű, illetve kisebb vagy egyenlő értékű számlák összértékét.

Megoldás: Itt is, tetszőleges cellába, a *SUM*-hoz hasonlóan használjuk a *SUMIF* függvényt, viszont feltételnek adjuk meg, hogy a számolandó cella értéke legyen több mint 100. Így a képlet a következő formájú lesz: `=SUMIF(B2:B8, „>100”)`, melynek eredménye `384.40 RON` lesz. A kisebb vagy egyenlő értékű számlák összege esetén pedig a feltételnek `„<=100”` értéket írunk, így a képlet `=SUMIF(B2:B8, „<=100”)` lesz. Ennek eredménye pedig `112.00 RON`.

4.4.5. Egyéb példák

A következőkben vegyünk néhány komplexebb példát. Legyenek a következő táblázatban egy nemrégiben alapult cég alkalmazottai:

	A	B	C
1	Alkalmazottak	Életkor	Régiség (hónap)
2	Kis Pista	20	12
3	Erdész Nóra	21	9
4	Kis János	19	18
5	Halász Anna	20	15
6	Almás János	20	18
7	Nagy Pista	19	6
8	Hagymás János	21	2

4.4.5.1. ábra. *Példa adatok egy cég alkalmazottainak adatairól*

A pont: Számoljuk meg, hány János nevezetű alkalmazottunk van.

Mivel megszámolni kell az ilyen cellákat, ezért a *COUNTIF* függvényt fogjuk használni az A oszlopra, feltételnek pedig az kell, hogy az alkalmazott neve „János" keresztnévvel végződjön.

B pont: Számoljuk meg, hány alkalmazottunk van, akiknek életkoruk nagyobb, mint az F2 cellába írt érték.

Itt is előfordulást kell számolni, ezért a *COUNTIF* függvény használjuk a B oszlopra, viszont a feltételben egy másik cella értékét kell használni. Ezt fel lehet írni szövegek összefűzésével (lásd a szöveges függvények fejezetben), éspedig: `„>=" & F2`.

C pont: számoljuk meg, hány munkatársunk van, aki legalább 6 hónapja dolgozik a cégnél, de nem több mint egy éve.

Itt is számolni kell, tehát a *COUNT* függvény egy variációja, de mivel több feltétel szerint, ezért a *COUNTIFS* függvényt fogjuk használni a C oszlopra. Az első feltétel, hogy az alkalmazott régisége nagyobb vagy egyenlő, mint 6 hónap, a második pedig, hogy régisége kisebb vagy egyenlő, mint 12 hónap.

D pont: Számoljuk ki a 12 hónapnál régebbi alkalmazottak átlagrégiségét.

Mivel átlagot kell számolni egyetlen feltétel szerint, ezért az *AVERAGEIF* függvényt kell használni a C oszlopra, ahol a feltétel, hogy a cella értéke több, mint 12.

E pont: számoljuk ki az átlagfizetésüket a több mint 6 hónap régiséggel rendelkező alkalmazottaknak.

Ebben az esetben *AVERAGEIF* függvényt nem lehet használni, mert a kritérium szelekció nem egyezik az átlagot számolandó oszloppal, ezért az *AVERAGEIFS* függvényt kell használni. Egy kritériumszelekció (D2:D8) és a hozzá tartozó feltétel („>6") megadása elegendő.

F pont: számoljuk meg az összfizetésüket azon alkalmazottaknak, akiknek legalább 2500 RON a fizetésük.

Mivel összeget kell számolni feltétel szerint, ezért a *SUMIF* függvényt kell használni a D oszlop celláira és a „<2500" feltétellel.

G pont: számoljuk meg az összfizetésüket azon alkalmazottaknak, akik legalább 20 évesek és legtöbb 12 hónapja dolgoznak a cégnél.

A *SUMIF* függvényt az F ponthoz hasonlóan itt nem lehet alkalmazni, mert azon kívül, hogy a feltétel oszlop nem egyezik a számolandó oszloppal, több feltételünk is van. Helyette használjuk a *SUMIFS* függvényt. A számolandó cellák továbbra is a D2:D8, viszont két feltételünk van: az egyik a B oszlop celláira, ahol ezen értékek nagyobb vagy egyenlőek, mint 20; a második feltétel pedig a C oszlop celláira vonatkozik, ahol az értékek kisebbek vagy egyenlőek, mint 12.

Megoldások:

- A pont: `=COUNTIF(A2:A8,"*János")`
- B pont: `=COUNTIF(B2:B8,">=" & F2)`
- C pont: `=COUNTIFS(C2:C8,">=6", C2:C8, "<=12")`
- D pont: `=AVERAGEIF(C2:C8,">12")`
- E pont: `=AVERAGEIFS(D2:D8, C2:C8,">6")`
- F pont: `=SUMIF(D2:D8, "<=2500")`
- G pont: `=SUMIFS(D2:D8, B2:B8,">=20", C2:C8, "<=12")`

4.4.6. Gyakorlófeladatok

4.4.6.1. Raktár

Adott a következő táblázat, mely egy cég néhány termékét ábrázolja raktáron, a különböző helységekben, darabszám szerint.

	A	B	C
1	Termék	Darabszám	Raktár
2	A	23	Csíkszereda
3	B	8	Udvarhely
4	A	17	Szentgyörgy
5	C	6	Csíkszereda
6	A	18	Udvarhely
7	C	9	Udvarhely
8	B	4	Csíkszereda

4.4.6.1. ábra. *Raktár feladat adatai*

Feladat: Az eddig említett képletek segítségével oldd meg a következő pontokat:

A. Hány különböző terméktípus van Udvarhelyen raktáron?

B. Hány darab termékünk van összesen Udvarhelyen?

C. Az „A" termékünkből összesen hány darab van raktáron?

Útmutató:

A. Itt azt kell megszámolni, hogy hány különböző típusú termékünk van raktáron Udvarhelyen. Mivel egy adott termék helységekként egy sorba van beírva a táblázatba, ezért elég megszámolni, hogy egy adott helység hányszor szerepel a táblázatban, ezért a *COUNT* függvény feltételes változatát kell alkalmazni, éspedig azokat a cellákat számolja meg a D oszlopban, amelyeknek értéke „Udvarhely".

B. Össze kell adni a termék darabszámait, amelyek Udvarhelyen vannak. A darabszám oszlop a B, a feltételünk pedig a C oszlop, ezért a feltételt külön szelekcióként kell megadni. Mivel összeget kell számolni, ezért a *SUMIFS* függvényt kell alkalmazni.

C. Hasonlóan a B ponthoz, össze kell adni a B oszlop celláit azon sorok esetén, ahol a bejegyzések az „A" termékre vonatkoznak.

Megoldás:

A. `=COUNTIF(C2:C8,"=Udvarhely")`

B. `=SUMIFS(B2:B8,C2:C8,"= Udvarhely ")`

C. `=SUMIFS(B2:B8,A2:A8,"=A")`

4.4.6.2. Eladások

Adott a következő táblázat, mely egy ügynökség valamennyi szerződéskötését tartalmazza:

	A	B	C	D
1	**Szerződés**	**Megye**	**Összeg**	**Ügynök**
2	S0010	HR	RON 3,980.00	Pista
3	S0011	MS	RON 7,850.00	János
4	S0012	HR	RON 4,150.00	Pista
5	S0013	CV	RON 2,980.00	Mari
6	S0014	MS	RON 4,500.00	János
7	S0015	MS	RON 7,150.00	Csilla
8	S0016	HR	RON 4,500.00	Pista

4.4.6.2.1. ábra. *Eladási adatok*

Feladat: Az eddig említett képletek segítségével oldd meg a következő pontokat:

A. Határozd meg a legdrágább szerződés értékeit Maros, Hargita és Kovászna megyében.

B. Határozd meg a szerződések összértékeit Maros, Hargita és Kovászna megyében.

C. Határozd meg az átlagszerződések értékeit Maros, Hargita és Kovászna megyében.

D. Határozd meg János ügynök eladásainak összegét, valamint átlagát.

Útmutató:

A. Három maximum értéket kell számolni a C oszlopból, de csak azokból a sorokból, ahol a B oszlop cella értéke szerre MS, HR és CV. A *MAXIF* függvény nem elegendő, mivel a feltételünk oszlopa különbözik a számolt oszlopétól, ezért a feltételnek külön szelekciót kell meghatározni. Ez a függvény a *MAXIFS*.

B. Az 'A' ponthoz hasonlóak a feltételek, viszont összértéket kell számolni, ezért a *SUMIFS* függvényt kell alkalmazni.

C. A 'B' ponthoz hasonló feltételek mellett, itt az *AVERAGEIFS* függvényt kell alkalmazni.

D. Az eladások számának elég a D oszlopot számolni, viszont az átlag és az összérték esetén a számolt oszlop különbözik a feltétel oszloptól, mivel a szerződések összegét a C oszlop celláiból vesszük, ezért átlagnak az *AVERAGEIFS*, összegnek pedig a *SUMIFS* függvényeket kell alkalmazni.

Megoldások:

A. Maros megyére: `=MAXIFS(C2:C8,B2:B8,"=MS")`

B. Hargita megyére: `=SUMIFS(C2:C8,B2:B8,"=HR")`

C. Kovászna megyére: `=AVERAGEIFS(C2:C8,B2:B8,"=CV")`

D. Eladások száma: `=COUNTIF(D2:D8,"=János")`

összege: `=SUMIFS(C2:C8,D2:D8,"=János")`

átlaga: `=AVERAGEIFS(C2:C8,D2:D8,"=János")`

4.5. Szövegkezelő függvények

Sok helyzetben szöveges értékekkel is kell dolgoznunk, például termék- vagy egyéb kódokkal, ezek részeivel, illetve ezek szerint döntést hozni számításainkban. Az Excel ilyen esetekre rendelkezik beépített függvényekkel, melyek nagyban megkönnyítik feladatainkat.

Mint korábban említettük, Excelben minden cellaérték szöveg, amit nem ismer fel valamilyen formátumnak, illetve minden formátumnak felismert értéket is lehet szövegként kezelni. Tehát minden szöveg karakterekből áll, azaz betűkből, számokból vagy egyéb speciális írásjelekből. Példának vegyük az „Informatika" szöveget. Ez 11 karakterből (11 betűből) áll, melyek balról jobbra számozandóak pozíciójuk szerint a szövegben.

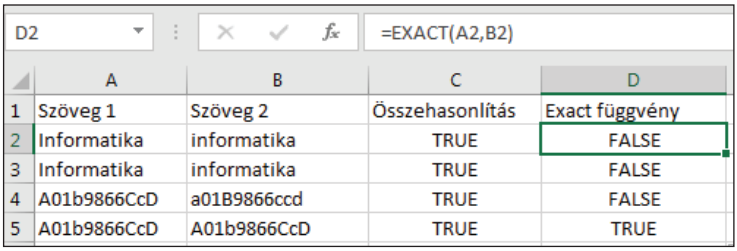
A szöveges függvények lehetővé teszik, hogy egy karakterlánc kívánt részeit kivegyük egy külön szövegbe, így ezekkel tovább tudunk dolgozni, más cellába behelyezni ezen értékeket, összefűzni és új szövegeket alkotni, valamint szövegekben keresni és részeket cserélni. A következő részekben a fontosabb függvényeket, illetve ezek használatát mutatjuk be.

4.5.1. Szövegek összehasonlítása

Excelben az egyenlőségjel segítségével össze tudjuk hasonlítani két cella értékét, például `=A1=B1`, ami TRUE vagy FALSE értéket eredményez annak függvényében, hogy a két érték egyforma-e vagy sem. Azonban ha szöveget tartalmaz a két cella, akkor ez az összehasonlítás nem veszi figyelembe a kis- és a nagybetűket, azaz *case-insensitive* módon történik ez. Önmagában ez nem probléma, viszont sok esetben problémát okozhat, mint például termékkódok vagy más jelölések esetén. Ilyen esetben a megoldás az *EXACT* függvény, mely a kis- és nagybetűket figyelembe véve hasonlít össze két szöveget, azaz *case-sensitive* módon. Szintaxisa:

```
=EXACT(szöveg1, szöveg2)
```

Példa:



	A	B	C	D
1	Szöveg 1	Szöveg 2	Összehasonlítás	Exact függvény
2	Informatika	informatika	TRUE	FALSE
3	Informatika	informatika	TRUE	FALSE
4	A01b9866CcD	a01B9866ccd	TRUE	FALSE
5	A01b9866CcD	A01b9866CcD	TRUE	TRUE

4.5.1.1. ábra. *Szövegek összehasonlítása példa*

4.5.2. LEFT / RIGHT függvények

A *LEFT* függvény egy szöveg bal, a *RIGHT* pedig a jobb oldaláról ad vissza karaktereket. Paraméterei:

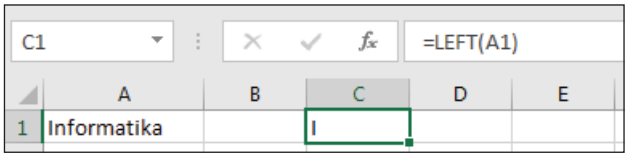
```
=LEFT(szöveg/cella, [darabszám])
=RIGHT(szöveg/cella, [darabszám])
```

Mindkét függvény két paraméterrel rendelkezik. Az első a cél, ami lehet egy adott szöveg vagy egy cella, amelyből kérünk karaktereket. A második opcionális paraméter a darabszám, ezzel mondjuk meg, hány darab karaktert térítsen vissza az adott függvény. Ha nem adjuk meg, akkor alapból *1* lesz az értéke, azaz egy darab karaktert fog visszaadni.

Példák:

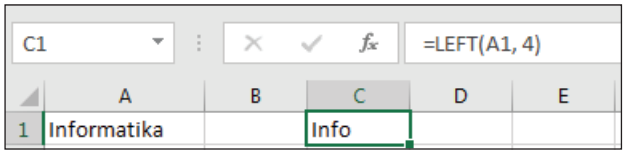
Írjuk be az „Informatika” szöveget az A1 cellába, és vegyünk ki karaktereket ebből. Ilyen esetben a `=LEFT(A1, 1)` képletet beírjuk egy tetszőleges másik cel-

lába. Ez a képlet az „I” betűt fogja visszaadni. Mivel a darabszám opcionális és értéke alapból `1`, ezért ezt el lehet hagyni jelen esetben, így a `=LEFT(A1)` is elég.



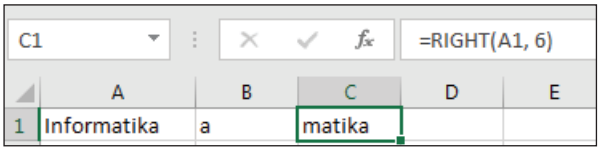
4.5.2.1. ábra. *LEFT függvényre példa*

A következőben vegyük ki az „Info” szót az A1 cellából. Ehhez szintén a cella érték bal oldaláról kell kivegyünk karaktereket, melyre ismét a LEFT függvényt használjuk. Mivel most 4 darab karaktert akarunk kivenni, ezért a második paraméter alapértelmezett `1` értékét felül kell definiálnunk, megadva konkrétan a `4` értéket: `LEFT(A1, 4)`



4.5.2.2. ábra. *LEFT függvényre példa, több karakter kivevése*

Most vegyünk ki karaktereket a szöveg jobb oldaláról, éspedig az utolsó és az utolsó 6 darab karaktert. Az utolsó karaktert az `=RIGHT(A1)` képlet segítségével tudjuk kivenni. Mivel csak egy karaktert kérünk a szövegből, ezért a második opcionális paramétert nem kell kiírni, mivel ennek értéke alapból `1`. Viszont ha `6` darab karaktert szeretnénk kivenni, akkor ennek meg kell adni a 6-os értéket: `=RIGHT(A1, 6)`.



4.5.2.3. ábra. *RIGHT függvényre példa, több karakter kivevése*

4.5.3. MID függvény

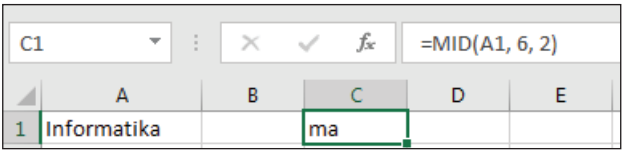
Az előzőkben láttuk, hogy a *LEFT*/*RIGHT* függvények segítségével karaktereket tudunk kivenni egy szöveg bal, illetve jobb oldaláról. Sokszor szükség van, hogy egy szöveg közepéről vegyünk ki karaktereket és ezekkel dolgozzunk. Erre van a *MID* függvény, mely a következő paraméterekkel rendelkezik:

```
=MID(szöveg/cella, pozíció, darabszám)
```

Itt a pozíció a kezdeti pozíciót jelenti, ahonnan kérjük a karaktereket, a darabszám pedig egyértelműen, hogy hány darab betűt kérünk. A pozíció számolása, mint említettük a korábbiakban, balról jobbra történik, az első betű a szövegben az 1-es pozíción található. Tekintsük a következő ábrát példaként a pozíció értékére:

I	n	f	o	r	m	A	t	i	k	a
1	2	3	4	5	6	7	8	9	10	11

Példaként maradjunk az „Informatika” szövegnél. Ha ki szeretnénk venni a „ma” betűket, akkor a következőképpen kell megadnunk a paramétereket: `MID(A1, 6, 2)`



4.5.3.1. ábra. *Példa* MID *függvény használatára*

A *MID* függvénnyel helyettesíteni lehet a *LEFT* függvényt is, viszont ilyen esetben több paramétert kell megadni. A *RIGHT* függvény helyettesítése pedig csak azonos hosszúságú szövegek esetén fog helyes eredményt adni. Ezt könnyű belátni, ha egy ilyen függvényt tartalmazó képletet alkalmazunk több cellára, amelyekben tetszőleges hosszúságú szövegek vannak.

4.5.4. SUBSTITUTE függvény

Sokszor előfordul, hogy számos cella értékeiben akarunk cserét végezni. Ezt manuálisan is lehet, viszont a *SUBSTITUTE* szövegcserélő függvénnyel sokkal hamarabban és precízebben csinálhatjuk ezt. A függvénynek a következőek a paraméterei:

```
=SUBSITUTE(cella/szöveg, mit, mire, [hányszor])
```

Az első paraméter itt is a cél, melyik vagy ahonnan vesszük az értéket, amiben cserélni szeretnénk. A következő két paraméter a „mit”, az a szöveg, amit keresünk, és a „mire”, az a szöveg, amire cseréljük a keresett szöveg előfordulását. Az utolsó opcionális paraméter pedig a cserék számát jelöli. Ha nem adjuk meg ezt, akkor a keresett szöveg összes előfordulását cserélni fogja.

A megadott szöveg keresése a bal oldaláról kezdődik és kis-/nagybetűérzékeny.

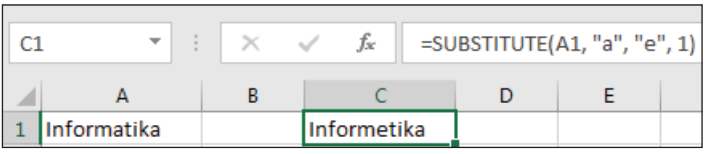
Példaként vegyük szintén az A1 cellában levő informatikaszöveget, és ebben cseréljük ki az „a” betűket „e” betűkre. A képlet a következőképpen néz ki:

```
=SUBSTITUTE(A1, ”a”, ”e”)
```

Ha csak az első előfordulását akarjuk kicserélni, akkor megadjuk az opcionális darabszám paramétert:

```
=SUBSTITUTE(A1, ”a”, ”e”, 1)
```

melynek eredménye:



4.5.4.1. ábra. *Példa a SUBSTITUTE függvény használatára*

4.5.5. Szövegek összefűzése

Gyakran szükség lehet a szöveges függvények által visszaadott eredmények összefűzésére. Ez lehetséges a beépített

```
=CONCATENATE(szöveg1, szöveg2, …)
```

függvény segítségével, viszont egyszerűbb az & karakter használata, mint például

```
= „Infor” & „matika”
```

4.5.6. FIND keresőfüggvény

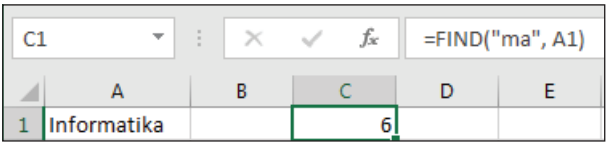
Ha egy szöveg előfordulását akarjuk vizsgálni egy másik szövegben, akkor ezt a

```
=FIND(mit, hol, [pozíció])
```

függvény segítségével végezhetjük. A „mit” paraméter a keresett szöveg, míg a „hol” az a szöveg vagy cella értéke, ahol keressük az illető szöveget. Az opcionális „pozíció” paraméter segítségével meg lehet adni a keresés induló pozícióját abban az esetben, ha nem a szöveg elejétől akarunk keresni. A keresés itt is a bal oldaláról indul és a talált szöveg pozícióját adja vissza. Például az

```
=FIND(”ma”, A1)
```

képlet eredménye 6 lesz, mert a „ma” szócska az „Informatika” szövegben a 6. pozíción található. Ha a „mai” szót keressük, erre `#VALUE` hibát fog visszaadni, mert ez nem található.



4.5.6.1. ábra. *Példa a FIND függvény használatára*

4.5.7. Egyéb függvények

Számos egyéb szövegekkel dolgozó függvény áll a rendelkezésünkre, melyek közül néhány fontosabb:

- szöveg hossza: *LEN(szöveg/cella)*
- szünetek levágása egy szöveg elejéből és végéből: *TRIM(szöveg/cella)*
- szöveg kisbetűvé alakítása: *LOWER(szöveg/cella)*
- szöveg nagybetűvé alakítása: *UPPER(szöveg/cella)*
- minden szó első betűjét nagybetűvé alakítása: *PROPER(szöveg/cella)*

Feladat: próbáld ki ezeket a függvényeket!

4.5.8. Gyakorlófeladatok szöveges függvényekkel

4.5.8.1. Leltári számok

Egy leltári számokat tároló listában cserélni kell bizonyos kódokat, éspedig ha 8-assal kezdődik, akkor ezt át kell írni 7-re. Ha jelentős mennyiségű leltári szám van a listában, akkor ezt a folyamatot kézzel elvégezni és nem tévedni szinte lehetetlen, viszont az Excel által biztosított függvények segítségével ezt automatizálni lehet, így könnyedén és gyorsan el lehet végezni ezt a feladatot. Az egyszerűség

kedvéért tekintsük a következő példa táblázatot, ahol az A oszlop tartalmazza az átírandó leltári számokat, a B oszlopba pedig kerüljenek az átírt kódok:

	A	B
1	Leltári szám	Új szám
2	802AH1146	
3	907KH1147	
4	808KH1148	
5	120TD1149	
6	128KD1150	

4.5.8.1.1. ábra. *Adatok a leltári számok feladathoz*

Megoldás: mivel cserét kell egy végezni szövegben, használjuk a *SUBSTITUTE* függvényt. Emlékezzünk vissza a paramétereire: hol, mit, mire és opcionálisan, hogy hányszor. Viszont ha a B oszlopba ebben a formában alkalmazzuk ezt, akkor nem minden esetben kapunk helyes eredményeket. Próbáljuk ki ezt az esetet, amikor a B2 cellában elvégzünk egy ilyen cserét: `=SUBSTITUTE(A2,"8","7")`, és ezt alkalmazzuk lefele a többi cellára ebben az oszlopban. Mi történik? Az első kódnál, `802AH1146`, helyesen elvégzi a cserét, mivel csak egyetlen 8-as számjegy van benne, és az is pont az első pozíción. Ellenben a `808KH1148` kódban több 8-as van, ezeket mind kicseréli 7-re, tehát ez már helytelen eredmény. Következő ötlet az lenne, hogy a cserét csak egyszer végezze el, ezért adjuk meg az opcionális csereszám paramétert: `=SUBSTITUTE(A2,"8","7", 1)`. Most már a `808KH1148` kód jó lesz a csere után, mert csak az első 8-ast cseréli ki, viszont az utolsó kód esetén, `128KD1150`, ismét helytelen eredményt kapunk, mert van benne 8-as, amit kicserél, viszont ez nem az első pozíción található.

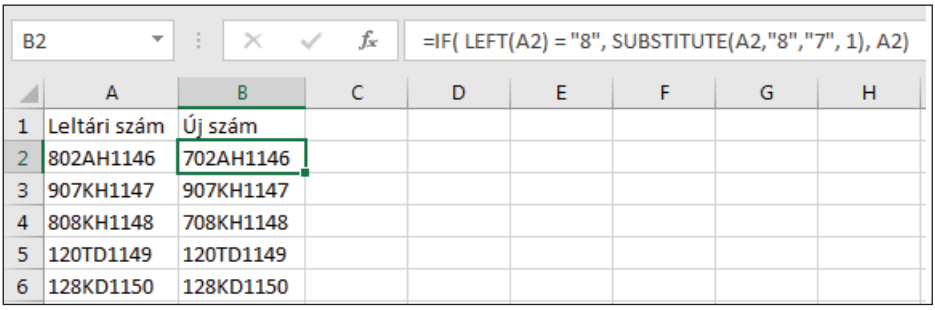
Következésképpen csak a *SUBSTITUTE* függvény használata nem elég, hanem szükséges ellenőrizni, hogy 8-as számjeggyel kezdődik-e vagy sem. Ha igen, akkor elvégezzük a cserét pontosan egyszer (mivel tudjuk, hogy az első 8-as számjegy pont a kód elején található), különben tartsuk meg az eredeti leltári számot. Ez utóbbi fontos, mert különben lyukak lesznek a listában, azaz csak akkor lesz szám, ha elvégeztünk egy cserét. Mivel dönteni kell, ezért használjuk az IF függvényt. Feltételnek meg kell vizsgálnunk, hogy a kód 8-assal kezdődik-e vagy sem. Ehhez viszont kell az első karakter a leltári számból, amit a *LEFT* függvény segítségével kapunk meg, éspedig: `=LEFT(A2)`. Itt ismét lehet segédoszlopot alkalmazni, éspedig a C oszlopba egyszer kivesszük a leltári szám első betűit, majd a B oszlopban IF függvény feltételében elvégezzük az összehasonlítást: `IF(C2="8", ... ,` vagy egybeírjuk: `IF(LEFT(A2) ="8", ...` . Ezután következik az IF függvény másik két paramétere, éspedig milyen értéket adunk vissza, ha a feltétel igaz, és mit, ha hamis. Ha igaz, akkor végezzük el a cserét és a *SUBSTITUTE* függvény visszatérési értékét adjuk vissza, azaz beágyazzuk ezt a függvényhívást az IF második

paraméterébe. Ha nem igaz a megadott feltétel, azaz a leltári szám nem 8-assal kezdődik, akkor adjuk vissza a cella eredeti értékét: ezért beírjuk az A2 cellacímet.

```
=IF( LEFT(A2) = "8", SUBSTITUTE(A2,"8","7", 1), A2)
```

Ezt a képletet lehet lefele alkalmazni a többi sorra. A fenti képlet összefoglalva:

- *IF* függvénnyel döntést végzünk, hogy milyen értéket adjunk vissza.
- Első paraméterben (IF függvény feltétele) a `LEFT(A2)="8"` megnézzük, hogy a leltári szám 8-assal kezdődik-e vagy sem.
- Második paraméterben elvégezzük a cserét, azaz beágyazzuk a *SUBSITUTE* függvényhívást. Ez csak akkor lesz meghívva, ha az *IF* függvény feltétele igaz, azaz 8-as számjeggyel kezdődik a leltári számunk. A függvény csak egyetlen cserét végez (4. paraméter „1"-es értéke), és ez pont a kezdő 8-as számjegy lesz.
- Harmadik paraméterben átvesszük és így visszaadjuk a cella eredeti értékét. Ez akkor történik meg, ha az *IF* feltétele nem igaz, azaz nem 8-assal kezdődik a kód.



	A	B	C	D	E	F	G	H
1	Leltári szám	Új szám						
2	802AH1146	702AH1146						
3	907KH1147	907KH1147						
4	808KH1148	708KH1148						
5	120TD1149	120TD1149						
6	128KD1150	128KD1150						

4.5.8.1.2. ábra. *Leltári számok feladat megoldva*

4.5.8.2. Szállítási költség

Adott a következő táblázat, mely egy rendelés adatait tartalmazza. Az A oszlop a megrendelt termék kódjait tartalmazza, a C oszlop az egységárat, a D oszlop pedig ezek darabszámait. A termékkódban az „M" betű utáni két karakter a termék súlyát jelenti, például az első termékkód esetén ez „06", azaz 6 kg egy termékre. A szállítócég 0.8 RON extraköltséget fog számlázni minden egyes kg-ra, ami túllépi a 200-at (ezek az értékek adottak a H2 és a H3 cellákban).

	A	B	C	D	E	F	G	H
1	Termék kód	Súly	Ár	Darabszám				
2	KM6822M06LM	?	RON 230.11	12			Ingyen szállítás (max. KG)	200
3	KT7801M17XL	?	RON 172.80	20			Extra költség / KG	RON 0.80
4	YT2814M02ST	?	RON 119.25	140				
5	RW2146M08WR	?	RON 310.00	28				
6	AA4002M12RB	?	RON 278.90	17				
7								
8			Termékek:	?				
9			Szállítás:	?				
10			Összesen:	?				

4.5.8.2.1. ábra. *Szállítási költség feladat*

Feladat: számold ki, összesen mennyit fizetünk a termékek után, a szállításra való extraköltség (ha van), illetve összesen a rendelés költségét. Továbbá a B oszlopba írjál egy lefele alkalmazható képletet, mely bármilyen hasonló kiosztású termékkódból kiveszi ennek súlyát.

Megoldás: mivel több mindent kell számolni, használjunk segédoszlopokat.

Termékek összköltsége: ennek kiszámolására meg kell szoroznunk egyenként a különböző termékek árát a rendelt darabszámmal, és ezeket összeadni. Használjuk az E oszlopot arra, hogy soronként kiszámoljuk a termékek költségeit, majd ezt a képletet alkalmazzuk lefele a többi sorra is. Végül a D8-as sorban egy *SUM* függvény segítségével adjuk össze ezeket.

Szállítási költség: ennek kiszámításához szükség van a termékek egyéni súlyára, melyet a termék kódja tartalmaz, éspedig az „M” betű utáni két karakterre. Mivel ezt a szöveg belsejéből kell kivegyük, ezért a *MID* függvényt kell használni. Emlékezzünk vissza paramétereire: honnan, hányadik pozíciótól és hány darab karaktert kérünk. Elsőnek írjuk fel a képletet a B2 cellába. Mivel a 2. sor adataival dolgozunk, a *MID* függvény az A2-es cellából kell vegye a termékkódot, a 8. pozíciótól és 2 darabot. Így a képlet a következő lesz: `=MID(A2,8,2)`, amit alkalmazhatunk is lefele.

Amint a B oszlopbeli képlet megvan, számoljuk ki a termékek súlyát darabszám szerint. Ehhez ismét használjunk egy segédoszlopot, éspedig a következő szabad F oszlopot. Itt össze kell szorozni a termék súlyát a darabszámmal, majd az oszlop alján adjuk össze egy *SUM* függvény segítségével (F7 cella).

Most, hogy megvan az összsúly, a D9-es cellában kell egy döntést végezni, éspedig, hogyha ez az érték több, mint 200 kg, akkor kell extra szállítási költséget számolni. Használjuk az IF függvényt, feltétele legyen `F7>H2`. Ha ez igaz, akkor számoljuk ki az extra szállítási költséget, különben adjunk vissza 0 értéket. Emlékezzünk, hogy az értékeket a H2 és a H3 cellákból vegyük! Mivel ezt a képletet nem alkalmazzuk lefele, ezért nem szükséges a H cellacímeket rögzíteni.

A legutolsó sorban, a D10 cellában kivonjuk a D8 cella értékéből a D9 cella értékét, így megkapjuk a rendelésre fizetendő összeget.

D9 =IF(F7>H2,(F7-H2)*H3,0)

	A	B	C	D	E	F	G	H
1	Termék kód	Súly	Ár	Darabszám				
2	KM6822M06LM	06	RON 230.11	12	RON 2,761.32	RON 72.00	Ingyen szállítás (max. KG)	200
3	KT7801M17XL	17	RON 172.80	20	RON 3,456.00	RON 340.00	Extra költség / KG	RON 0.80
4	YT2814M02ST	02	RON 119.25	140	RON 16,695.00	RON 280.00		
5	RW2146M08WR	08	RON 310.00	28	RON 8,680.00	RON 224.00		
6	AA4002M12RB	12	RON 278.90	17	RON 4,741.30	RON 204.00		
7						RON 1,120.00		
8			Termékek:	RON 36,333.62				
9			Szállítás:	RON 736.00				
10			Összesen:	RON 35,597.62				

4.5.8.2.2. ábra. *Szállítási költség feladat megoldva*

4.5.8.3 Személyi számok

Adott a következő táblázat, melynek az A oszlopában különböző személyi számok találhatóak. A személyi szám formátuma a következő: NÉÉHHNNxxxxxx, ahol az N a nemet jelenti, az ÉÉ a születési év utolsó két számjegyét, a HH a születési hónapot, míg az NN karakterek a születési napot. Mivel a születési évnek csak a két utolsó számjegyét tartalmazza, például 21 az 1921-nek, ezért nem tudunk különbséget tenni azok között, akik 1921-ben és akik 2021-ben születtek. Ez a konfliktus a „nem” karakterrel van megoldva, éspedig az 19xx-ben született személyeknek 1 (férfi) vagy 2 (nő) ez, a 2000 után született személyeké pedig 5 (férfi) vagy 6 (nő).

	A	B	C
1	CNP	Nem	Születési év
2	1800520122892	?	?
3	2810923855196	?	?
4	5011018120810	?	?
5	6021129245765	?	?

4.5.8.3.1. ábra. *Személyi számok feladat*

Feladat:

- A B oszlopban állapítsd meg az adott személy nemét. A cellában jelenjen meg a ’férfi’ vagy ’nő’ szöveg.

- A C oszlopban írd ki a teljes születési évét az illető személynek, éspedig az első sornál: 1980.

Útmutató:

B oszlop: dönteni kell (*IF* függvény használata), éspedig a személyi szám első karakterét kell megvizsgálni, hogy ez '1' vagy '5', ha igen, akkor férfi az illető, különben nő. Az első karaktert a *LEFT* függvénnyel lehet kivenni, pontosabban kétszer, mert először az '1'-es karakterrel hasonlítjuk össze, másodszor viszont a '2'-es karakterrel. Azonban ezt a két összehasonlítást nem lehet direkt vesszővel elválasztva az *IF* függvénybe beírni, mert ennek az első paramétere a feltétel, ezért kell egy egyesítő függvény, amely megvizsgálja a két feltételt, és egyetlen igazat vagy hamisat ad vissza. Ilyen függvények az *AND* és az *OR*. Mivel nekünk a két feltételünk nem teljesülhet egyszerre, mármint a személyi szám első karaktere nem lehet egyszerre '1' és '5' is, hanem vagy egyik, vagy másik, ezért az *OR* függvényt kell használni. Ez a következőképpen néz ki: `OR(LEFT(A2)="1", LEFT(A2)="5")`. Ezt az *OR* függvényt kell beágyazni az *IF* első, feltétel paraméterébe, figyelve a zárójelezésre, ugyanis az *OR*-nak le kell zárni a zárójelét, viszont ha véletlenül két zárójelet írunk, akkor a második az *IF* függvényt fogja zárni, és hibás lesz a képlet. Ha ez a feltétel teljesül, akkor adjuk vissza a „férfi" szöveget, különben a „nő" szöveget.

C oszlop: ez ismét egy összetettebb művelet, és használhatunk segédoszlopot: kivenni a születési évet a személyi számból, majd a C oszlopban ez elé kell fűzni a '19' szöveget vagy a '20' szöveget, attól függően, hogy 19xx-ban született az illető vagy 2000 után.

Először vegyük ki a születési évet. Mivel a szöveg belsejéből kell karaktereket kivennünk, ezért a *MID* függvényt használjuk (D2 cella), éspedig az A2 cellából veszünk ki, a 2. pozíciótól, 2 darabot. Ha nem használunk segédoszlopot, akkor ez a *MID* függvény, majd a C oszlopbeli *IF* függvény belsejébe kerül.

A C oszlopban el kell döntenünk, hogy az illető személy 2000 után született-e vagy sem, mert ha igen, akkor '20'-at kell a születési dátum elé fűzni, különben '19'-et. Ehhez ismét meg kell vizsgálnunk a személyi szám első karakterét, éspedig hogy értéke '1' vagy '2', illetve '5' vagy '6'. Itt ismét két feltételünk van, amit fel lehet írni egy *OR* függvény segítségével, például: `OR(LEFT(A2)="1", LEFT(A2)="2")`, viszont egyszerűbben is lehet. Mivel az Excel össze tud hasonlítani karaktereket is nagyság szerint, nemcsak számokat, így elég, ha azt nézzük, hogy a személyi szám első karaktere kisebb-e, mint a '3'-as szöveg karaktere vagy sem. Vigyázat: szöveg '3'-as karakterrel kell hasonlítsuk, nem numerikus 3-mal, mivel a *LEFT* függvény szöveget ad vissza, és ellenkező esetben a visszaadott ASCII karakter kódját hasonlítjuk a 3-as számhoz. Ebben az esetben az *IF* függvény feltételébe a következő összehasonlítás jön: `LEFT(A2)>"3"`. Ha ez igaz, akkor az illető személy 2000 után született és '20'-at kell a születési éve elé fűzni, különben pedig '19'-et. Emlékezzünk vissza, hogy szövegeket az & jel

segítségével tudunk összefűzni, ezért az első összefűzés a következőképpen fog kinézni: `„20” & D2`. Ha elkészült az *IF* képlet a C2 cellában, akkor ezt tudjuk lefele alkalmazni.

Megoldás:

- B oszlop: `=IF(OR(LEFT(A2)=”1”, LEFT(A2)=”5”), “Ferfi”, “No”)`
- C oszlop (D segédoszloppal): `=IF(LEFT(A2)>”3”, “20” & D2, “19” & D2)`, vagy `=IF(LEFT(A2)>”3”, “20”, “19”) & D2`. A második megoldás esetén az *IF* függvény csak eldönti és visszaadja a ’19’ vagy ’20’-as szövegeket, majd ehhez fűzzük a D2 cellában kivett születési dátumot.
- C oszlop (egy képlettel): `=IF(LEFT(A2)>”3”, “20”, “19”) & MID(A2,2,2)`

C2 | =IF(LEFT(A2)>"3", "20", "19") & MID(A2,2,2)

	A	B	C	D	E	F	G
1	CNP	Nem	Születési év				
2	1800520122892	Ferfi	1980				
3	2810923855196	No	1981				
4	5011018120810	Ferfi	2001				
5	6021129245765	No	2002				

4.4.8.3.2. ábra. *Személyi számok feladat megoldva*

4.6. Dátum- és időkezelő függvények

Excelben a dátum és idő értékek egyszerű számok, dátum formában megjelenítve.

A dátum egy 0 és 2.9 millió közötti szám, mely az eltelt napok számát jelenti 1900. január 01. óta, így az érvényes dátumok az 1900. január 01. és 9999. december 31. közötti intervallumba eső értékek. 1900 előtti dátumokat negatív számként lehet ábrázolni, azonban ezek támogatása verziótól függően jelentősen változik.

	Dátum		Idő
1	1/1/1900	0	0:00
9999	12/31/9999	0.25	6:00
		0.5	12:00
		0.75	18:00
		1	0:00

4.6.1. ábra. *Dátum- és időformátumok Excelben*

Az időt Excel 0 és 1 közötti értékekként tárolja, azaz ezek a tizedes értékek, míg az egész értékek a dátumok.

A következő részekben a dátumok formázásáról, az ezekkel végzendő műveletekről, valamint az Excel által biztosított dátumkezelő függvényekről lesz szó. Az Excelben összesen 25 dátum- és időfüggvény áll a rendelkezésünkre, viszont ezekből csak a fontosabbakat és gyakran használtakat említjük meg.

4.6.1. Műveletek dátumokkal

Mivel a dátumok és idők számok, ezért az Excel könnyedén tud olyan műveleteket végezni ezekkel, mint például az összeadás, kivonás, valamint összehasonlítás.

Példák: Az A1 cellába írjuk be, hogy 01/15/2020, a B1 cellába pedig hogy 01/20/2020, és nézzük meg a következő műveleteket:

- Összehasonlítás:

2. `=A1=B1` képlet hamis értéket fog visszaadni, mert a két cellában levő dátum nem egyenlő.
3. `=A1<B1` képlet igazat fog visszaadni, mivel az A1 cellában levő dátum kisebb, mint a B1-ben levő.

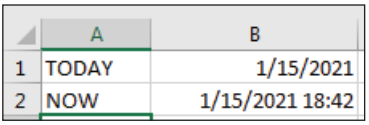
- Kivonás: az `=B1-A1` képlet 5-ös értéket fog visszaadni. Mivel már említettük, hogy az Excel a dátumokat számként tárolja, amik napokat jelentenek, ezért ilyen esetben a két számot egyszerűen kivonja egymásból, és az eredmény a különbséget jelenti napokban. Ha esetleg eredményként dátum jelenik meg a cellában, például ilyen esetben 01/05/1900, akkor ez az 5-ös szám dátumként való megjelenése. Ilyenkor formázni kell a cellát, mint szám.
- Összeadás: az `=A1+5` képlet eredménye 01/20/2020 lesz, azaz az A1 cellában levő dátum 5 nappal való rákövetkező dátumát kapjuk. Itt is, mivel az Excel számokban tárolja a dátumot, ami a napok számát jelenti, ezért 5-tel megnövelve ezt az értéket, 5 nappal későbbi dátumot kapunk.

4.6.2. TODAY és NOW függvény

A *TODAY* függvény az aktuális dátumot adja vissza. Ha a mai nap beírom ezt egy cellába és elmentem, amikor holnap megnyitom a munkalapot, a cella értéke frissül az aznapi dátumra. Más szóval ezt a függvényt dinamikusnak is nevezhetjük. A *NOW* függvény hasonlóan dinamikus, és az aktuális dátumot a teljes idővel adja vissza. Szintaxisaik:

```
=TODAY()
=NOW()
```

Fontos, hogy bár a függvénynek nincs paramétere, mivel függvényről van szó, a zárójeleket ki kell írni a neve után, különben az Excel nem fogja függvénynek tekinteni és hibát ad.



	A	B
1	TODAY	1/15/2021
2	NOW	1/15/2021 18:42

4.6.2.1. ábra. *Példa a TODAY és NOW függvényekre*

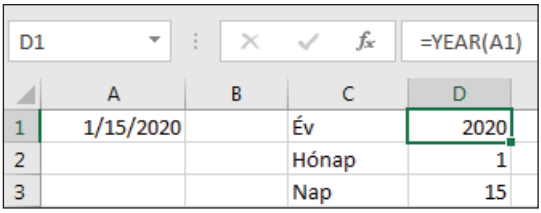
Megjegyzés: az aktuális dátum- és időfüggvények használata esetén az Excel automatikusan frissíti ezek értékeit, valamint az összes képletet újraszámolja, amelyben ezek a függvények használva vannak.

4.6.3. YEAR, MONTH és DAY függvények

A következő három függvény, éspedig a *YEAR*, a *MONTH* és a *DAY*, egy érvényes dátumnak a részeit adja vissza. Amint a nevük is mutatja, ezek az év, hónap és nap részeit adják vissza egy érvényes dátumnak. Szintaxisaik:

```
=YEAR(cella / érvényes dátum)
=MONTH(cella / érvényes dátum)
=DAY(cella / érvényes dátum)
```

Fontos, hogy a paraméterként adott értékek érvényes dátumok legyenek.



D1 =YEAR(A1)

	A	B	C	D
1	1/15/2020		Év	2020
2			Hónap	1
3			Nap	15

4.5.3.1. ábra. *Példa a YEAR, MONTH és DAY függvényekre*

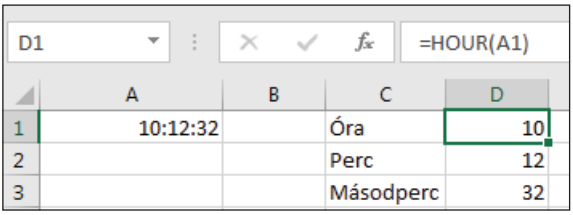
Gyakori hiba, hogy egy cellába beírt dátum formátuma nem talál azzal, amit az Excel elvárt, így nem dátumként ismeri fel, hanem szövegként. Ha a három függvény közül valamelyiknek egy ilyen cellát adunk meg paraméternek, akkor hibát fog adni, mivel szövegből nem tud dátumrészeket kivenni. Fontos, hogy ellenőrizzük, hogy a megadott cella érvényes dátum-e vagy sem. A legkönnyebb ezt ellenőrizni, ha megnézzük, hogy a beírt dátumot a cella melyik oldalára helyezi

el, tudniillik hogy a felismert számformátumokat a cella jobb oldalára helyezi (hacsak nem igazítottuk kézzel a cella értékét más irányba).

4.6.4. HOUR, MINUTE, SECOND függvények

A dátum részeit visszaadó függvényekhez hasonlóan működik az idő részeit visszaadó három függvény is, éspedig a *HOUR*, *MINUTE* és *SECOND*. Mint neveik mutatják, ezek az órát, a percet és a másodpercrészeket adják vissza egy érvényes időből. Szintaxisaik:

```
=HOUR(cella / érvényes dátum)
=MINUTE(cella / érvényes dátum)
=SECOND(cella / érvényes dátum)
```



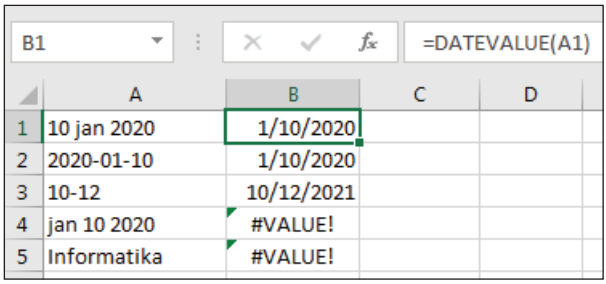
4.6.4.1. ábra. *Példa HOUR, MINUTE és SECOND függvényekre*

4.6.5. DATEVALUE, TIMEVALUE függvények

A *DATEVALUE* függvény egy szövegként megadott dátumot próbál meg dátummá alakítani és ezt visszaadni. A *TIMEVALUE* függvénynek ugyanez a szerepe szövegként megadott időre nézve. Ha ez nem sikerül, akkor értékhibát ad. Szintaxisaik:

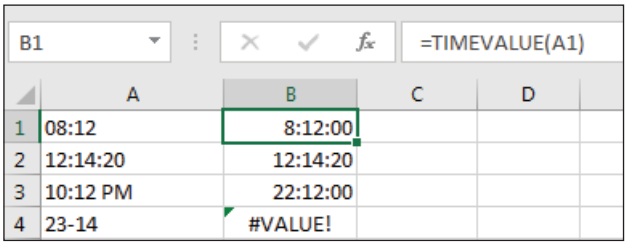
```
=DATEVALUE(cella / szöveg)
=TIMEVALUE(cella / szöveg)
```

A következő példában látni lehet, hogy ez nem minden esetben jár sikerrel, éspedig a „jan 10 2020”-as szöveget nem tudja dátummá alakítani, ezért hibát ad.



4.6.5.1. ábra. *Példa a DATEVALUE függvényre*

A következő ábrán példa látható a *TIMEVALUE* függvényre is. Az „AM” és „PM” jelöléseket is helyesen kezeli:



4.5.5.2. ábra. *Példa a TIMEVALUE függvényre*

4.6.6. DATE függvény

A *DATE* függvény lehetővé teszi egy Excel dátum létrehozását egy paraméterként megadott év, hónap és nap értékekből. A *TIME* függvény Excel időt hoz létre megadott óra, perc és másodperc értékekből. Szintaxisaik:

```
=DATE(év, hónap, nap)
=TIME(óra, perc, másodperc)
```

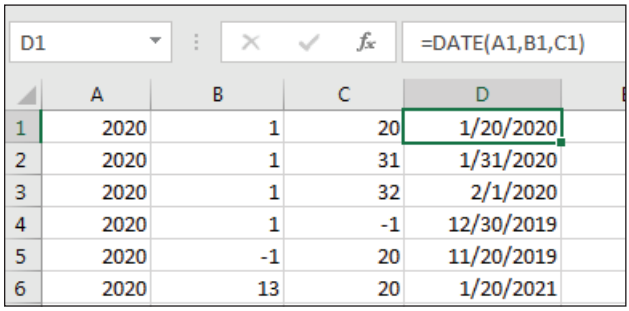
A *DATE* függvény hasznos tulajdonsága, hogy a hónap és nap paraméter értékei lehetnek nagyobbak, mint a hónapok számai, illetve a napok száma egy adott hónapban, vagy akár negatív értékek is. Az Excel ilyenkor eltolódásnak tekinti ezeket a többletértékeket, és ezeket figyelembe véve kiszámolja a megfelelő dátumot. Hasonlóan értékeli ki a *TIME* függvény paramétereit is.

Megjegyzés: mindkét függvény esetén a paraméterek értékei lehetnek szövegként megadott számok is. Ez különösen hasznos, ha szöveges függvények által visszaadott eredményekkel akarunk itt dolgozni.

Példa: az A oszlopba írjunk be éveket, a B oszlopba hónapokat, a C oszlopba meg napokat. Végül a D oszlopban készítsünk ezekből dátumot a *DATE* függvény segítségével. Írjuk be a következő képletet:

```
=DATE(A1, B1, C1)
```

amit lefele tudunk alkalmazni az összes sorra. A példa megoldását a következő ábra illusztrálja:



	A	B	C	D
1	2020	1	20	1/20/2020
2	2020	1	31	1/31/2020
3	2020	1	32	2/1/2020
4	2020	1	-1	12/30/2019
5	2020	-1	20	11/20/2019
6	2020	13	20	1/20/2021

4.6.6.1. ábra. *Példa a DATE függvényre*

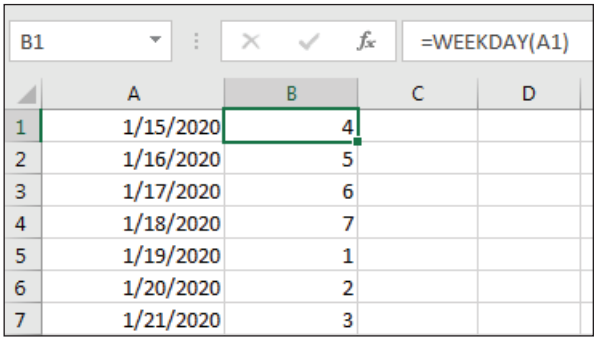
Amint látható, a 3. sorban a január 32-t úgy kezeli, mint január 31. plusz egy nap, amit kiszámolva február 1-jei dátum lesz eredményként. A 4. sorban a -1-es napot, visszafele számolva, előző hónap december 31-ét kapjuk. Végül az 5. sorban a -1. hónap az előző hónapot, a 13. hónap pedig a következő év első hónapját jelenti.

4.6.7. WEEKDAY függvény

A *WEEKDAY* függvény egy dátumból az adott hét napszámát adja vissza. Szintaxisa:

```
=WEEKDAY (cella / dátum, [eredmény típus])
```

Az első paraméter a megszokott dátum vagy érvényes dátumot tartalmazó cella. A második opcionális paraméter a hét napjainak számozását irányítja. Ha nem adunk meg semmit, akkor ezt 1-nek veszi, és a függvény visszatérési értéke 1 (vasárnap) – 7 (szombat), amint az alábbi ábrán is látható:



4.6.7.1. ábra. *Példa a WEEKDAY függvényre*

Az eredmény típusainak fontosabb értékei:

- Üres vagy 1 - eredménye: 1 – 7 (vasárnap, hétfő, …, szombat)
- 2 – eredménye: 1 – 7 (hétfő, kedd, …, vasárnap)
- 3 – eredménye: 0 – 6 (hétfő, kedd, …, vasárnap)

Megjegyzés: a *WEEKDAY* akkor is ad vissza értéket, ha a megadott dátum üres! Csak érvénytelen/hibás dátum esetén ad vissza értékhibát.

4.6.8. WEEKNUM, ISOWEEKNUM függvény

A *WEEKNUM* függvény a hétnek a számát adja vissza egy megadott dátumból, azaz hogy az adott dátum az évnek a hányadik hetében van. Szintaxisa:

```
=WEEKNUM(cella / dátum, [hét első napja])
=ISOWEEKNUM(cella / dátum)
```

Az első paraméter itt is egy érvényes dátum vagy dátumot tartalmazó cella. A második opcionális paraméter a hét kezdőnapját adja meg. Az alapérték az 1-es, melynek jelentése, hogy a hét első napja a vasárnap, 2-es érték jelentése, hogy a hét első napja hétfő, és így tovább. A hetek számozása 1-estől kezdődik, éspedig január első hetébe eső dátumra 1-es értéket ad vissza a függvény.

Az *ISOWEEKNUM* az ISO standard szerinti hétnek a számát adja vissza, amelyikbe esik a paraméterként megadott dátum. Ezen szabvány szerint a hét mindig hétfővel kezdődik, és az év első hete az a hét, amelyik tartalmazza az év első keddi napját.

4.6.9. DATEDIF függvény

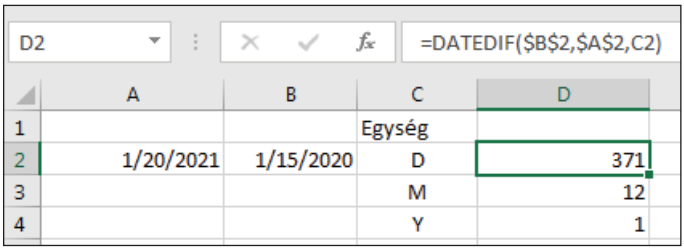
Mint említettük, két dátum közötti különbséget napokban megkaphatunk úgy, hogy kivonjuk a két dátumot egymásból. Viszont ha a különbség hónapban vagy évben kell, akkor egy intuitív, bár rossz megközelítés, a napok számának osztása 30-cal vagy 365-tel. Ilyen esetekre az Excel rendelkezik a *DATEDIF* függvénnyel, mely a két megadott dátum közötti különbséget adja vissza napban, hónapban vagy évben. Szintaxisa:

```
=DATEDIF(dátum1, dátum2, egység)
```

A visszaadott értéket meghatározó „egység” paraméter leggyakrabban használt értékei:

- “D” – napban lesz az eredmény. Ez egyenértékű a két dátum egymásból való kivonásával
- “M” – a különbség hónapokban lesz visszaadva
- “Y” – az eredmény években

Megjegyzés: A *dátum1* paraméter kötelező módon nagyobb kell legyen, mint a *dátum2*. Az egység paramétert szöveges formában kell megadni.



4.6.9.1. ábra. *Példa a DATEIF függvényre*

4.6.10. Gyakorlófeladatok dátumos függvényekkel

A következőkben nézzünk meg néhány dátumot kezelő függvényekkel megoldható feladatot. A képletek ellenőrzése érdekében írjál be olyan dátumokat is, hogy ezek a lehetséges esetek mindegyikét lefedjék.

4.6.10.1. Termékek szavatossága

Egy kisboltban a következő táblázatot használják a termékek nyilvántartására:

	A	B	C	D	E
1	Termék neve	Gyártási idő	Szavatosság (hónap)	Lejárta idő	Figyelmeztetés
2		2/21/2021	1	?	?
3		1/18/2021	2	?	?
4		12/15/2020	3	?	?
5		10/9/2020	6	?	?

4.6.10.1.1. ábra. *Termékek szavatossága feladat*

Minden termék vásárlásakor beírják a táblázatba ennek nevét, gyártási időpontját és szavatossági idejét (azaz a gyártástól kezdve hány hónapig érvényes az illető termék). Automatizáld ezt a táblázatot a következőképpen (használva lefele alkalmazható képleteket):

- A D oszlopban legyen kiszámolva a lejárási időpont.
- Az E oszlopban a lejárási dátumtól függően legyenek a következő figyelmeztetések:
  1. Ha a termék lejárt, jelenjen meg a következő szöveg: „Lejárt!”
  2. Ha kevesebb mint 7 nap múlva jár le: „Leárazni”. Ez a szöveg fog figyelmeztetni, hogy árazzuk le a terméket. Inkább nyerjünk rajta kevesebbet, mintsem teljes vesztesség legyen.
  3. Különben maradjon üres a cella. Ez hasznos, mert könnyebben észrevesszük, ha valamelyik sornál fontos figyelmeztetés van.
- Az F oszlopban, ha még nem járt le a termék, írja ki pontosan, hány nap múlva jár le.

Útmutató:

- D oszlop: az adott gyártási dátumhoz (B oszlop) hozzá kell adni a megfelelő hónapot a C oszlopból. Direkt nem tudjuk ezt hozzáadni, mert az Excel a dátumokat számként tárolja és ez a napok számát jelenti, így például a +1 nem egy hónapot ad hozza, hanem egy napot. A `+C2*30` sem jó megoldás, mert nem minden hónap egyforma számú napból áll. Ezért a B oszlopban levő dátumot fel kell bontani, újra dátumot csinálni, és a hónap részhez hozzáadni a C oszlop celláját. Induljunk ki onnan, hogy dátumot a *DATE* függvénnyel készítünk. Emlékezzünk vissza ennek paramétereire: év, hónap és nap. Ahhoz, hogy ezeket a paramétereket megadjuk, a B oszlop cellájából ki kell vegyük az évet, a hónapot és a napot. Ezeket a *YEAR*, *MONTH* és *DAY* függvény segítségével lehet. Lehet használni három segédoszlopot is, de elég egyszerű egybeírni is. Ne felejtsük el a hónap részhez hozzáadni a C oszlopból a hónapok számát.
- F oszlop: Az E oszlop előtt csináljuk meg ezt az oszlopot. Itt csak két feltételünk van, ezért elég egy darab *IF* függvény. Azt kell vizsgálni, hogy lejárt-e

a termék vagy sem, azaz ha a `D2 >= TODAY()` még nem járt le, és ebben az esetben adjuk vissza a hátramaradt napok számát, azaz `D2-TODAY()`, különben egy üres szöveget adjunk vissza.

- E oszlop: három feltételt kell vizsgálnunk, éspedig: lejárt, kevesebb mint 7 nap van hátra, és különben. Ezt egyetlen *IF* segítségével nem tudjuk, ezért szükségünk van két *IF* függvényt egymásba ágyazni. Először azt nézzük, hogyan tudjuk megállapítani, hogy egy termék lejárt-e vagy sem. Ennek megállapításához mindig a mai dátumhoz kell hasonlítsuk a lejárási dátumot. A mai dátumot a `TODAY()` paraméter nélküli függvény adja vissza. Ha lejárt, akkor azt jelenti, hogy a lejárati dátum egy múltbeli dátum, ezért: `D2 < TODAY()`. Ha ez nem igaz, akkor a termék még nem járt le, így meg kell vizsgáljuk, hogy kevesebb mint 7 nap van hátra, amíg lejár vagy sem. Mint említettük, az Excel a dátumokat számokban tárolja, ami napokat jelent, ezért ha kivonunk két dátumot egymásból, az eredmény a különbség napokban lesz. Így megvizsgáljuk, hogy a `D2 – TODAY()` kevesebb-e, mint 7, vagy sem (itt tudjuk, hogy a termék nem járt le, azaz a D2 nagyobb, mint a mai nap). Most, hogy a feltételek megvannak, egymásba kell ágyazni az *IF* függvényeket és visszaadni a megfelelő szövegeket, vigyázva a zárójelezésre. Ha készen van, akkor lehet alkalmazni lefele.

Megjegyzés: Mivel az Excel a `TODAY()` által visszaadott dátumot automatikusan frissíti, ezért a munkalap minden megnyitásakor a képletek újraszámolódnak az új értékekkel.

Megoldás:

- D oszlop: `=DATE(YEAR(B2), MONTH(B2) + C2, DAY(B2))`
- E oszlop: `=IF(D2 < TODAY(), "Lejárt!", IF( D2 - TODAY() < 7, "Leárazni", "" ))`
- F oszlop: `=IF(D2 >= TODAY(), D2 - TODAY(), "")`

4.6.10.2. Személyi számok (folytatás)

Folytassuk a 3.4.8.3-as feladatot, egészítsük ki a következő oszlopokkal:

	A	B	C	D	E	F	G
1	CNP	Nem	Születési év	Születési dátum	Mikor lesz 40 éves	Hány nap van addig?	Figyelmeztetés
2	180052012	Ferfi	1980	?	?	?	?
3	281092385	No	1981	?	?	?	?
4	501101812	Ferfi	2001	?	?	?	?
5	602112924	No	2002	?	?	?	?

4.6.10.2.1. ábra. *Személyi számok (folytatás) feladat*

Figyelem: a 'CNP' oszlop bevitelekor kezdjük aposztróffal az értéket, különben számként fogja felismerni, és átalakítja tudományos formátumra (pl. `1.78E+12`).

Feladat:

- A D oszlopban határozzuk meg az adott személyek születési dátumát (legyen dátum érték).
- Az E oszlopban számoljuk azt a dátumot, amikor az adott személy pontosan 40 éves lesz.
- Az F oszlopban állapítsuk meg, hány nap van hátra, és írjuk ki a következő formában az értesítést: „XY nap múlva", de csak akkor, ha kevesebb mint egy év van hátra.

A '?' jelek helyére írjunk lefele alkalmazható képleteket, melyek elvégzik a szükséges számolásokat!

Útmutató:

D oszlop (születési dátum): itt dátumot kell generáljunk, amire rendelkezésünkre áll a *DATE* függvény. Emlékezzünk vissza paramétereire: év, hónap és nap. Az évet már meghatároztuk a C oszlopban, ezért csak a hónap és a nap kell. Ezek a személyi szám 4–5. és 6–7. karakterei, amit a *MID* függvénnyel ki lehet venni (emlékezzünk vissza ennek paramétereire: honnan, hányadik pozíciótól és hány darabot. Jelen esetben két-két darab karaktert kérünk a 4. és a 6. pozíciótól). A *MID* függvényhívásokra lehet segédoszlopot használni, vagy direkt be lehet ágyazni őket a *DATE* függvény megfelelő paramétereihez. Ha készen van, alkalmazni kell a képletet, és a többi cellára is ellenőrizni az eredményt.

E oszlop (mikor lesz 40 éves?): itt a konkrét dátumot kell meghatározni, viszont a D oszlophoz nem tudunk direkt 40 évet hozzáadni (ilyen függvénnyel még nem rendelkezik az Excel; ha `40 * 365`-öt hozzáadunk, az nem lesz jó, mert a szökőévek miatt elcsúszik a dátum napja), ezért ismét a *DATE* függvényhez folyamodunk, és két megoldás is a rendelkezésünkre áll:

- A D oszlop képletét átmásoljuk, és a *DATE* függvény év paraméterénél hozzáadunk 40-et, azaz: `C2 + 40` lesz a paraméterben.
- Egy második megoldás a D oszlop celláiban levő dátumot szétszedni év, hónap és napra a *YEAR*, *MONTH* és *DAY* függvények segítségével, míg ezeket betenni a *DATE* paramétereinek, és az évnél itt is hozzáadni 40-et: `YEAR(D2) + 40`.

F oszlop (Hány nap van, amíg 40 éves lesz?): az E oszlopban már kiszámoltuk, mikor lesz az illető személy 40 éves, ezért az E oszlopban levő dátumot a mai dátumhoz kell hasonlítani, amit a `TODAY()` paraméter nélküli függvény ad vissza. Továbbá meg kell állapítani, hogy kevesebb mint egy év, azaz kevesebb mint 365 nap van hátra.

Itt ismét dönteni kell, ezért használjuk az IF függvényt. Először nézzük meg a feltételt vagy feltételeket, amiket meg kell vizsgálni. Megállapítani, hogy hány nap van hátra a mai nap és az E oszlopban levő dátum között, egyszerű: mivel

az Excel számként tárolja a dátumot, ami a napok számát jelenti 1970 óta, ezért a kettőt kivonjuk egymásból. Két eset lehet: ha még nem töltötte be a 40 évet, akkor az E2 cellában levő dátum nagyobb, mint a mai nap, ezért az eredmény pozitív lesz, ellenkező esetben viszont negatív. Ezért nem elég csak azt vizsgálni, hogy ez a különbség kevesebb, mint 365, mivel a negatív szám is kisebb, hanem azt is meg kell vizsgálni, hogy betöltötte-e a 40 évet, azaz az E2 nagyobb, mint a `TODAY()` (alternatív esetben azt is lehet vizsgálni, hogy ez a különbség pozitív legyen). Tehát a helyes eredményért két feltételt kell vizsgálni:

- Több mint 40 éves?
- Kevesebb mint 365 nap van hátra, amíg 40 éves lesz?

Az *IF* függvénynek egy feltételparamétere van, nekünk pedig két feltételünk, ezért szükségünk van egy logikai függvényre, amely mindkét feltételt megvizsgálja, és egyetlen igaz vagy hamis értéket ad vissza. Mivel a két feltétel egyszerre kell teljesüljön, ez a függvény az *AND* lesz, ami bekerül az *IF* feltétel paraméteréhez. Ha ez igazat ad vissza, akkor az „`XY nap múlva`” szöveget adjuk vissza, különben pedig üres kell maradjon a cella, tehát üres szöveget adunk vissza.

Az „`XY nap múlva`” szöveg összeállítása a hátralevő napok száma és a „nap múlva” szöveg összefűzése (emlékezzünk vissza, szöveget az ’&’ jel segítségével tudunk könnyedén összefűzni). Mivel már ellenőrizve volt, hogy az adott személy nem töltötte be még az 40 évet, ezért tudjuk, hogy a mai dátum kisebb, mint az E oszlopban levő dátum, ezért az E oszlop cella dátumából vonjuk ki a mai dátumot. Vigyázat: a hátralevő napok kiszámolásánál a `NOW()` függvény nem lesz jó, mert ez az időt is visszaadja, így a különbségnél a napok száma különbözhet eggyel, ezért szigorúan a *TODAY* függvény által visszaadott dátumot kell használni.

Ha összeállítottuk az *IF* függvényt, akkor alkalmazzuk a többi sor celláira és ellenőrizzük az eredményt.

Megoldás:

G2 =IF(AND(E2 - TODAY() < 365, TODAY() < E2), (E2 - TODAY()) & " nap múlva", "")

	A	B	C	D	E	F	G	H	I	J
1	CNP	Nem	Születési év	Születési dátum	Mikor lesz 40 éves	Hány nap van addig?	Figyelmeztetés			
2	180052012	Ferfi	1980	5/20/1980	5/5/2020	Elmúlt				
3	281092385	No	1981	9/23/1981	9/9/2021	174	174 nap múlva			
4	501101812	Ferfi	2001	10/18/2001	10/10/2041	7510				
5	602112924	No	2002	11/29/2002	11/11/2042	7907				

4.6.10.2.2. ábra. *Személyi számok (folytatás) feladat megoldva*

- D oszlop: `=DATE(C2, MID(A2,4,2), MID(A2,6,2))`
- E oszlop: `=DATE(C2 + 40, MID(A2,4,2),MID(A2,6,2))` vagy `=DATE(YEAR(D2) + 40, MONTH(D2),MONTH(D2))`
- F oszlop: `=IF(TODAY() < E2, E2 - TODAY(), "Elmúlt")`

- G oszlop: `=IF(AND(E2 - TODAY() < 365, TODAY() < E2), (E2 - TODAY()) & " nap múlva", "")`

4.6.10.3. Könyvkölcsönzés

Egy könyvtárban adott a következő kölcsönzési táblázat, ahol az ott dolgozók a B oszlopba beírják egy adott könyvnek a kölcsönzési dátumát, míg a C oszlopba ennek periódusát, azaz hány napig lehet kölcsönözni az adott könyvet. Ha egy könyvet visszahoztak, akkor ennek dátumát beírják az E oszlop cellájába, addig viszont üres ennek az értéke.

	A	B	C	D	E	F	G
1	Leltári szám	Kölcsönzési dátum	Kölcsönzési periódus (nap)	Periódus lejárta	Visszahozatali dátum	Elkésett?	Megjegyzés
2	3-598-21500-2	3/25/2021	14	?		?	?
3	3-598-21501-0	2/14/2021	21	?	3/2/2021	?	?
4	3-598-21504-5	2/28/2021	30	?		?	?
5	3-598-21529-0	3/8/2021	14	?		?	?
6	3-598-21530-4	1/25/2021	60	?	3/10/2021	?	?
7	3-598-21536-3	2/26/2021	21	?		?	?

4.6.10.3.1. ábra. *Könyvkölcsönzés feladat*

Feladat:

- A D oszlopba automatikusan legyen kiszámolva a kölcsönzési periódus vége, hogy kölcsönzés esetén egyből tudják ezt a határidőt közölni.
- Az F oszlopban késés esetén jelenjen meg az „Igen” szó.
- A G oszlopban legyenek a következő megjegyzések:
  1. Ha kevesebb mint 7 nap van hátra a kölcsönzési periódusból: „X nap van hátra”.
  2. Ha elkésett, a visszahozatallal számoljunk késedelmi kamatot az elkésett napok száma szerint. Ennek értékét (késedelmi díj / nap) vegyük a K2 cellából.
  3. Különben: maradjon üres a cella.

Útmutató:

D oszlop: mivel az Excel a dátumot számként tárolja, ami a napok számát jelenti, ellentétben az előző feladattal, itt nem kell felbontani a dátumot év, hónap és nap részekre, hanem elég egyszerűen hozzáadni a C oszlop celláját, azaz két numerikus értéket adunk össze, ahol mindkettő a napok számát jelenti.

F oszlop: két feltételnek kell teljesülnie a figyelmeztetéshez, éspedig: még nem hozta vissza az adott könyvet, illetve a kölcsönzés határideje lejárt. Az első feltételhez meg kell nézni az E oszlop megfelelő celláját, hogy üres-e vagy sem. A második feltétel esetén a D oszlop celláját a mai dátummal kell összehasonlítani,

éspedig ha elmúlt a kölcsönzési határidő, ez azt jelenti, hogy a D oszlop cella értéke kisebb, mint a mai dátum. A mai dátumot természetesen a `TODAY()` függvénytől kapjuk meg, így minden nap a képleteinket újra kiértékeli az Excel. Mivel döntenünk kell, ezért az *IF* függvényt kell használni. Mivel több feltételünk van, ezért szükséges egy logikai függvény használata, amely egyesíti ezt a két feltételt, és egyetlen igaz vagy hamis értéket ad vissza az *IF* függvény feltétel paraméterének. A figyelmeztetéshez mindkét feltétel egyszerre kell teljesüljön, ezért ez az *AND* függvény lesz. Ha ez igaz, akkor az illető személy elkésett a könyvnek a visszahozatalával, ezért az IF függvényünk adja vissza az „Igen" szöveget, különben pedig egy üres szöveget, hogy a cella értéke is üres maradjon.

G oszlop: itt három esetünk van, ezért két *IF* függvényt kell egymásba ágyazni. Nézzük az eseteket és azt, hogy milyen feltételek kell teljesüljenek és milyen értéket kell írjunk a cellába:

- Kevesebb mint 7 nap van hátra a kölcsönzési határidő lejártáig: ez csak akkor igaz, ha még nem hozta vissza a könyvet (E oszlop cellája üres), és a `TODAY()` függvény által visszaadott dátum és a D oszlop cellájában levő dátum között kevesebb mint 7 nap van. Ebben az esetben az „X nap van hátra szöveget" kell összeállítsuk, ahol az X a hátramaradt napok számát jelenti, ami a különbség a mai dátum és a határidő között.
- Elkésett: még nem hozta vissza a könyvet, és a mai dátum nagyobb, mint a D oszlopban levő határidő. Ebben az esetben az elkésett napok számát (a különbség a mai dátum és a határidő között) meg kell szorozzuk a K2 cellában levő napi késedelmi kamattal.
- Különben eset: ha egyik előző feltétel sem teljesül. Ez egyszerűbb, mint külön megvizsgálni, hogy visszahozta-e már a könyvet, vagy a mai dátum és a határidő között több mint 7 nap van.

Ismerve ezeket a feltételeket, meg kell állapítani a feltételek sorrendjét a két egymásba ágyazott *IF* esetére. Fel lehet egy képlet segítségével is írni, viszont egyszerűbb, ha egy segédoszlopot használunk. Fogjuk be a H oszlopot erre a célra, és számoljuk ki a különbséget a mai nap és a határidő között: vonjuk ki a kettőt egymásból (`D2 - TODAY()`). Ha az eredmény dátum lesz, akkor vissza kell formázni numerikus értékre. Ha ez negatív lesz, azaz a mai nap nagyobb, mint a D2, akkor az illető elkésett, különben ezt az értéket meg lehet vizsgálni, hogy kevesebb mint 7 nap, ami szükséges a figyelmeztetéshez. Természetesen mindkét esetben azt is kell vizsgálni, hogy a könyvet még nem hozták vissza. Így a feltételeket a következő sorrendben lehet alkalmazni a két *IF* függvény esetén:

- Még nem hozta vissza a könyvet, és a H2 cella kisebb, mint 0, akkor elkésett, különben:
- Ha még nem hozta vissza a könyvet, és a H2 kevesebb, mint 7, akkor figyelmeztető szöveg a hátramaradt napok számával (itt összefűzzük a H2 cellát a „nap van hátra szöveggel"), különben üres szöveget adunk vissza.

Alternatív esetben ez önmagában lehet kezdő feltétel, ami ha igaz, akkor üres a cella, viszont akkor még három másik feltételünk marad, amihez másik két *IF* függvényt kell alkalmazni, így ez három darab egymásba ágyazott *IF* függvényhez vezet.

Megoldás:

- D oszlop: `=B2+C2`
- F oszlop: `=IF(AND(E2="",D2<TODAY()), "Igen", "")`
- G oszlop: `=IF(AND(H2<0, E2=""),"Elkésett", IF(AND(E2="",H2 <7), H2 & " nap van hátra", ""))`
- H oszlop: `=D2 - TODAY()`

G2 =IF(AND(H2<0, E2=""),"Elkésett", IF(AND(E2="",H2 <7), H2 & " nap van hátra", ""))

	A	B	C	D	E	F	G	H
1	Leltári szám	Kölcsönzési dátum	Kölcsönzési periódus (nap)	Periódus lejárta	Visszahozatali dátum	Elkésett?	Megjegyzés	
2	3-598-21500-2	3/25/2021	14	4/8/2021				14
3	3-598-21501-0	2/14/2021	21	3/7/2021	3/2/2021			-18
4	3-598-21504-5	2/28/2021	30	3/30/2021			5 nap van hátra	5
5	3-598-21529-0	3/8/2021	14	3/22/2021		Igen	Elkésett	-3
6	3-598-21530-4	1/25/2021	60	3/26/2021	3/10/2021			1
7	3-598-21536-3	2/26/2021	21	3/19/2021		Igen	Elkésett	-6

4.6.10.3.2. ábra. *Könyvkölcsönzés feladat megoldva*

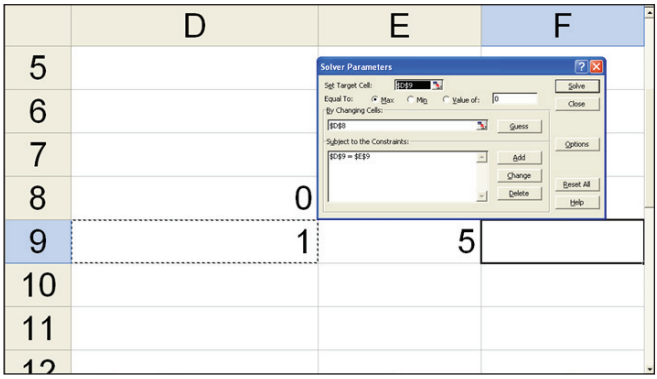
4.7. Pénzfüggvények

Az Excel pénzfüggvényei főleg az amerikai pénzmozgásra, tőkepiacra ad hasznos segítséget. A technikásabb függvényeihez komoly pénzügyi ismeretek szükségesek, vagy például jártasságot feltételez a biztosítási matematikában. Mi csak a legegyszerűbb és a leghasznosabb három pénzügyi függvényt fogjuk bemutatni, ezek a PMT, PV és FV. Nagyon fontosak a pénzügyi függvények. A középiskolás tankönyvekben van is egy olyan fejezet, hogy pénzügyi matematika. Ebbe a fejezetbe tették a kombinatorikát is. Itt olvashatunk a százalékszámításról, a kamatos kamatról, profitszámításról és értékcsökkenésről (amortizációról). Ezeket a pénzügyi számításokat sokkal szemléletesebben lehet tanítani Excel segédlettel. Hiszen a tanulók azonnal látják az unalmas számítás eredményét, és meg kellene tanítanunk azok értelmezésére. Sőt, ha egy picikét érdekesebb a feladat, például hány év alatt ötszöröződik meg a lekötött tőkepénzünk, akkor logaritmustáblázatot is kellene használjunk. Hát annál már közelebb van a számítógép!

Tehát a feladat az, hogy 6,5%-os kamatláb mellett hány év alatt ötszöröződik meg a tőke. Hallottam olyan tanulói véleményt, hogy rossz a feladat, mert nincs

megadva a tőke nagysága. Holott a jövendőbeli pénzügyi szakembernek azt kell megértenie, hogy a feladat lényege éppen az, hogy nem függ a tőke nagyságától, hanem csak a kamatlábtól!

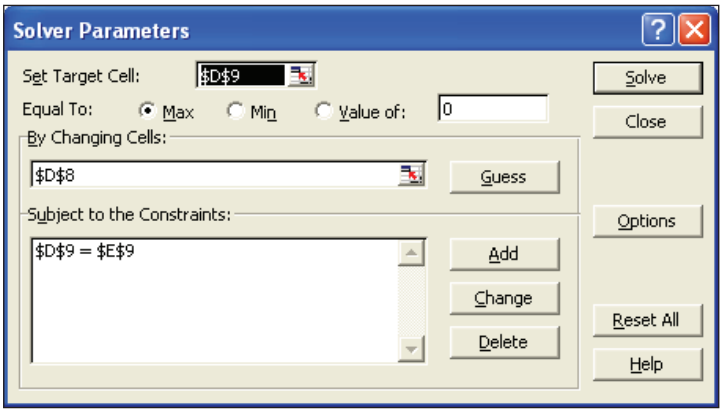
`=(1+6.5/100)^D8=5`



4.7.1. ábra. *Az Excel-lap kitöltése a Solver előkészítésére*

A D8 cellába beírjuk az ismeretlent, jelen esetben egy nullát. A D9 cellába beírjuk az egyenlet bal oldalát, vagyis: `=(1+6.5/100)^D8`. Ez nyilvánvaló 1 értéket ad, mert a kitevő nulla. Az E9 cellába beírjuk az egyenlet jobb oldalát, vagyis az 5-öt.

Ezt Solverrel megoldatni így a legegyszerűbb:



4.7.2. ábra. *A Solver párbeszédpanelje az exponenciális egyenlet megoldásához*

Lényegében itt a Solvert csak a `=(1+6.5/100)^x=5` exponenciális egyenlet megoldására használjuk. A cél, hogy kielégítsük az egyenletet, az ismeretlen a

kitevő, a feltétel is maga az egyenlet. Általában többféleképpen is megoldható Solverrel. A Solver által szolgáltatott eredmény: 25,55686.

Tehát 25,56 év alatt, vagyis 25 év és 7 hónap után.

Nagyon hasznos, ha a tanulókat erről meg is győzzük, sok egyszerű példa segítségével. Erre használatosak az Excel pénzügyi függvényei. Magyar verziónál a `=JBÉ`, vagyis jövőbeni érték, az angolnál az `=FV`, vagyis „future value".

`=FV(6.5%,D7,,-5000)` eredmény `25 000,00 RON`. Persze a D7 cella tartalmazza a 25,55686 értéket.

```
=FV(6.5%,25.56,,-100) = 500,10 RON
=FV(6.5%,D7,,-10) = 50,00 RON
=FV(6.5%,D7,,-1) = 5,00 RON
```

4.7.1. Feladatok

4.7.1.1. Gyakorlófeladatok

1. Mennyi a havi törlesztőrészlet, ha 100 000 RON-t 33%-os kamatláb mellett 15 hónap alatt egyenlő részletekben kell visszafizetni?

Megoldás:

```
=PMT(33%/12,15,100000,,0) = -8225.92
```

2. Következő feladat: Évi 12%-os kamatláb mellett 15 évre előtakarékosságot gyakorlunk. Az időszak végén 500000 RON-nal akarjuk meglepni kedvezményezettünket. Mennyi lesz ebben az esetben a hónap végén rendszeresen befizetendő összeg?

Megoldás:

```
=PMT(12%/12,15*12,0,500000) = -1000.84
```

Ellenőrzés:

```
=FV(12%/12,15*12,B11) =FV(12%/12,15*12,-1000.84031) = 500000
```

3. Évi 28% kamatláb mellett havi 10000 RON életjáradékot szeretnénk kapni 10 éven keresztül minden hónap végén. Az életjáradék ellenértékét az első év elején kell egy összegben befizetni. Mekkora ez az összeg?

Megoldás:

```
=PV(28%/12,10*12,10000,0,0) = -401658.02
```

Ellenőrzés:

```
=PMT(A20/12,10*12,D21,,0) =PMT(28%/12,10*12,401658.02,,0) = 10000
```

4. Évi 28%-os kamatláb mellett 6 hónapon keresztül minden hónap elején beteszünk 1000 RON-t (kirándulásra) egy takarékkönyvbe. Mekkora a 6. hónap végén felvehető összeg?

Megoldás:

```
=FV(28%/12,6,-1000,0,1) = 6509.51
```

5. Egy takarékkönyvben elhelyezünk 100000 RON-t, majd 10 éven keresztül minden hónap végén beteszünk 10000 RON-t, 28%-os kamatláb mellett. Mekkora összeg áll majd rendelkezésünkre?

Megoldás:

```
=FV(28%/12,10*12,10000,100000,0) = -7988445.002
```

Ellenőrzés:

```
=PV(28%/12,12*10,10000,-7988445.002) = 100,000.00 RON
```

4.7.1.2. Műkorcsolyaverseny eredménykijelzője:

4.6.1.2.1. táblázat. *Műkorcsolyaverseny résztvevői*

Helyezés	Versenyzők	Összesen:
4	Alina Zagitova	53.68287
7	Kaetlyn Osmond	46.08581
5	Yevgenia Medvedeva	51.57661
1	Aszada Mao	54.91180
2	Kim Jona	54.36373
3	Carolina Kostner	53.68887
6	Andó Miki	47.79498

A lényeg, hogy a bírók összpontszámából levonják a legnagyobb és a legkisebb osztályzatot, és utána kerül kiértékelésre a helyezés.

```
=IF(LARGE($C$3:$C$9,1)=C3,1,IF(LARGE($C$3:$C$9,2)=C3,2,
IF(LARGE($C$3:$C$9,3)=C3,3,IF(LARGE($C$3:$C$9,4)=C3,4,
IF(LARGE($C$3:$C$9,5)=C3,5,IF(LARGE($C$3:$C$9,6)=C3,6,
IF(LARGE($C$3:$C$9,7)=C3,7,”mas nincs”)))))))
```

Elegánsabban:

4.6.1.2.2. táblázat. *Elegáns megoldás RANK függvénnyel*

Elegáns megoldás
=RANK(C3,$C$3:$C$9,0)
=RANK(C4,$C$3:$C$9,0)
=RANK(C5,$C$3:$C$9,0)
=RANK(C6,$C$3:$C$9,0)
=RANK(C7,$C$3:$C$9,0)
=RANK(C8,$C$3:$C$9,0)
=RANK(C9,$C$3:$C$9,0)

4.8. Keresőfüggvények

A keresőfüggvények táblázatokból, listákból, „adatbázisokból” keresnek nagyon hatékonyan. Olyasmire kell gondolni a keresőfüggvények használatánál, ahogyan szótárakban, lexikonokban keresünk a természetes intelligenciánkkal. Tehát olyan keresésekre gondoljunk, mintha egy lexikonban, szótárban keresnénk egy megfelelő szó vagy fogalom alapján.

4.8.1. VLOOKUP, HLOOKUP

A *VLOOKUP* és *HLOOKUP* talán az Excel legerősebb keresőfüggvényei. Ha az adataink oszlopokba vannak rendezve, akkor *VLOOKUP* függvény segítségével, ha pedig sorokba vannak rendezve, akkor a *HLOOKUP* függvény segítségével lehet keresni ezekben. A *VLOOKUP* esetén a keresés mindig egy szelekció első oszlopában történik, és eredménynek a talált cella sorából egy másik oszlop cellaértékét kapjuk. Szintaxisaik:

```
=VLOOKUP(mit keresünk, hol (szelekció), eredmény oszlop,
[megközelítő keresés])
```

```
=HLOOKUP(mit keresünk, hol (szelekció), eredmény sor,
[megközelítő keresés])
```

Paraméterek:

- Mit keresünk: az az érték, amit keresünk. Ezt meg lehet direktbe is adni, vagy egy cellacímet, ahonnan vegye az értéket.
- Hol keressük: ez egy szelekció kell legyen. *VLOOKUP* esetén ennek az első oszlopában fogja a megadott értéket keresni.
- Eredmény oszlop: a szelekcióban megtalált érték sora esetén hányadik oszlopát adja vissza ennek.

Megközelítő keresés: opcionális paraméter, alapból igaz értékű, azaz megközelítő érték is elfogadható. Ha hamis értéket adunk, akkor egzakt keresésről van szó, azaz ha nem találja meg a pontos értéket, akkor az `#N/A` hibát kapjuk. Ehhez szükséges, hogy az első oszlop értékei rendezve legyenek.

Példa 1:

Vegyük a következő leltári tárgyakat ábrázoló táblázatot:

	A	B	C	D	E	F	G
1	Leltári kód	Név	Ár				
2	A01	Samsung tablet	RON 500.00		Keresés		
3	A02	HP nyomtató	RON 250.00		Kód	Név	Ár
4	B05	Epson szkenner	RON 375.00		B05		
5	C07	LG képernyő	RON 450.00				

4.8.1.1. ábra. *Leltári tárgyak példa*

VLOOKUP függvény segítségével, az E4 cella értékét keresve a táblázat bal oldali részén automatikusan töltsük ki az F4 és G4 cellákat, azaz keressük az E4 cellába írt kód szerint a termék nevét és árát.

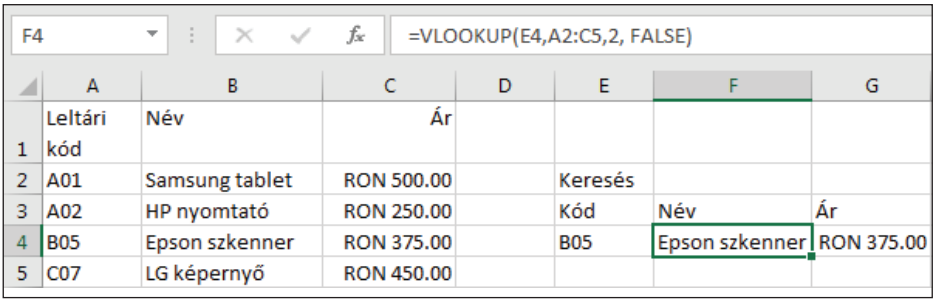
Megoldás: Az F4 oszlopba írjuk fel a *VLOOKUP* függvényt, melynek paraméterei:

- Első paraméternek megadjuk az E4 cella címét, ami azt jelenti, hogy innen vesszük a keresett értéket.
- Második paraméternek megadjuk a keresési szelekciót, azaz A2:C5-ig. Ennek a szelekciónak az első oszlopában fogja keresni az E4 cella értékét, azaz a „B05” kódot.
- A példa esetén a keresett értéket a 4. sorban találja meg. A függvény harmadik paraméterének a szelekció oszlopszámát kell megadjuk, ami azt jelenti, hogy ebből hányadik oszlopot kérjük eredménynek. Jelen esetben a „Leltári kód” az első oszlop, a „Név” a második és az „Ár” pedig a harmadik oszlop

az A2:C5 szelekcióból. Mivel a termék nevét akarjuk meghatározni, ezért a 2. oszlopot kérjük, azaz 2-es értéket írunk ide.

- Negyedik paraméter opcionális. Ha nem írunk ide semmit, akkor ennek értéke TRUE, azaz megközelítő keresést fog végezni. Ez nem minden esetben jó, például írjunk az E2 cellába „B01” kódot, amelyre viszont a „HP nyomtató” értéket fogja visszaadni. Ha ezt nem szeretnénk, akkor ide írjunk TRUE értéket, hogy egzakt keresés legyen.

A G4 oszlopba hasonlóan írjuk fel a képletet. Az egyedüli különbség az eredmény oszlopszáma lesz, éspedig itt a termék ára kell, ezért a 3-as számú oszlopot kérjük.



F4 =VLOOKUP(E4,A2:C5,2, FALSE)

	A	B	C	D	E	F	G
1	Leltári kód	Név	Ár				
2	A01	Samsung tablet	RON 500.00		Keresés		
3	A02	HP nyomtató	RON 250.00		Kód	Név	Ár
4	B05	Epson szkenner	RON 375.00		B05	Epson szkenner	RON 375.00
5	C07	LG képernyő	RON 450.00				

4.8.1.2. ábra. *Leltári tárgyak példa megoldva*

Példa 2:
Legyen a következő táblázat alkalmazottak neveivel és eladásaikkal:

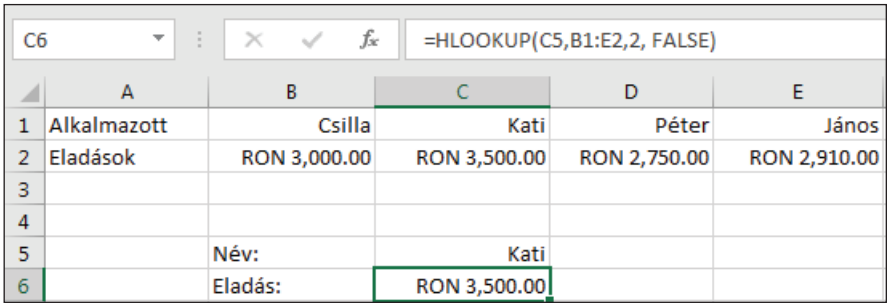
	A	B	C	D	E
1	Alkalmazott	Csilla	Kati	Péter	János
2	Eladások	RON 3,000.00	RON 3,500.00	RON 2,750.00	RON 2,910.00
3					
4					
5		Név:	Kati		
6		Eladás:			

4.8.1.3. ábra. *Alkalmazottak példa*

A *HLOOKUP* függvényt használva keressük meg a C5 cellába írt alkalmazott nevéhez tartozó eladás értékét.

Megoldás: a *VLOOKUP*-hoz hasonlóan, a keresett érték a C5 cella értéke, a keresési szelekció pedig a B1:E2, ellenben itt nem eredmény oszlopról beszélünk, hanem eredmény sorszámról. Mivel az eladásra van szükségünk, ezért a talált

oszlophoz tartozó sorok közül a másodikat kérjük. Ha egzakt keresést akarunk, akkor a negyedik paraméternek itt is FALSE értéket kell adjunk.



C6 =HLOOKUP(C5,B1:E2,2, FALSE)

	A	B	C	D	E
1	Alkalmazott	Csilla	Kati	Péter	János
2	Eladások	RON 3,000.00	RON 3,500.00	RON 2,750.00	RON 2,910.00
3					
4					
5		Név:	Kati		
6		Eladás:	RON 3,500.00		

4.8.1.4. ábra. *Alkalmazottak példa megoldva*

4.8.2. Gyakorlófeladatok keresőfüggvényekkel

4.8.2.1. Adószámolás

Adott a következő táblázat, mely soronként egy-egy terméket tartalmaz (A oszlop), a hozzá tartozó árat (B oszlop), valamint a neki megfelelő adótípust.

	A	B	C	D	E	F	G	H
1	Termék	Ár	Adótípus	Adó (%)	Adó (RON)		Adók	
2		RON 100.20	1	?	?		1	10.00%
3		RON 980.00	3	?	?		2	8.50%
4		RON 476.00	4	?	?		3	5.00%
5		RON 285.00	2	?	?		4	3.00%
6		RON 451.00	3	?	?			
7								
8				Összesen:	?			

4.8.2.1.1. ábra. *Adószámolásos feladat*

Feladat: felhasználva a G-H oszlopokban megadott adótípusokat és ezek értékeit, számoljuk ki a D és az E oszlopok értékeit, valamint az összadó értékét. Az adótípusok száma tetszőlegesen változhat, azaz bármikor jöhetnek be új vagy vehetünk ki meglévő értékeket. Írjunk lefele alkalmazható képleteket.

Megoldás:

- D oszlop: mivel az adótípus táblázatában függőlegesen kell keressünk, ezért a *VLOOKUP* függvényt fogjuk használni (D2 cellában): a keresett érték a termék adótípusa a C2 cellából, a keresési tartomány pedig a G2:H5 szelekció, melyből a 2. oszlop találati sor cellájának kérjük az értékét, a 4. paraméter pedig opcionális, viszont egzakt keresésre van szükségünk, ezért meg kell adjuk a FALSE értéket. Figyelem: mivel a D2 cella képletét lefele fogjuk alkalmazni, ezért rögzíteni kell a megfelelő cellacímeket, hogy ezek sorait ne növelje az Excel, jelen esetünkben ami nem szabad változzon, az a keresési szelekció.
- E oszlop: mivel a D oszlop celláiban megvan az adó % értéke, ezért itt kiszámoljuk a RON értéket, soronként összeszorozva a két oszlop celláit, azaz: `=B2*D2`.
- E8 cella: itt egy *SUM* függvény segítségével összeadjuk a fenti cellákat, éspedig: `=SUM(E2:E6)`.

D2 =VLOOKUP(C2, $G$2:$H$5,2, FALSE)

	A	B	C	D	E	F	G	H
1	Termék	Ár	Adótípus	Adó (%)	Adó (RON)		Adók	
2		RON 100.20	1	10.00%	RON 10.02		1	10.00%
3		RON 980.00	3	5.00%	RON 49.00		2	8.50%
4		RON 476.00	4	3.00%	RON 14.28		3	5.00%
5		RON 285.00	2	8.50%	RON 24.23		4	3.00%
6		RON 451.00	3	5.00%	RON 22.55			
7								
8				Összesen:	RON 120.08			

4.8.2.1.2. ábra. *Adószámolásos feladat megoldva*

4.8.2.2. Jövedelemszámolás

Legyen egy szórakozójegyeket viszonteladó cég. A cég minden eladott jegy után valamennyi százalékot magának tulajdonít. Adott a következő táblázat, és ebben az A oszlopban az értékesített szolgáltatások kódjai, melyek esetén az utolsó karakter szerint lehet számolni az eladások utáni saját jövedelmet. A jövedelemkódok és a hozzá tartozó %-ok az I és J oszlopokban találhatóak.

	A	B	C	D	E	F	G	H	I	J
1	Termék kód	Ár	Eladott Darabszám	Bevétel	Jövedelem (%-ban)	Jövedelem (RON-ban)			**Kód**	**Jövedelem**
2	CT2020M140M	RON 1,450.00	28	?	?	?			A	0.55%
3	CS2021O422L	RON 2,115.75	42	?	?	?			L	0.65%
4	XV2021J858A	RON 985.92	36	?	?	?			M	1.24%
5	HM2020X182N	RON 1,701.50	9	?	?	?			N	0.92%
6	HX2021P080L	RON 1,299.99	17	?	?	?			V	2.25%
7	GV2021W000A	RON 1,583.96	31	?	?	?				
8										
9			Összesen:	?		?				

4.8.2.2.1. ábra. *Jövedelemszámolás feladat*

Feladat: számoljuk ki termékenként, valamint az összbevételt, és határozzuk meg ebből a saját jövedelmet. Írjunk lefele alkalmazható képleteket.

Útmutató:

- D oszlop (bevétel): mivel adott a jegy ára és az eladott darabszám, így a bevétel egy adott jegy értékesítéséből a kettőnek a szorzata, azaz amennyi darabot eladtunk, annyiszor egy jegy ára.
- E oszlop (jövedelem %-ban): ahhoz, hogy megkeressük, mennyi a jövedelem egy adott jegy után, ki kell vegyük ennek kódjából az utolsó karaktert, amihez a *RIGHT* függvényt tudjuk használni, aminek visszatérési értékét átadjuk a *VLOOKUP* függvény első paraméterének. Másképpen mondva, a *RIGHT* függvényt beágyazzuk a keresőfüggvényünkbe. Vigyázat a zárójelezésre! Alternatív esetben lehet segédoszlopot is használni, ahol kivesszük a jegy kódjának utolsó karakterét, majd innen vesszük a keresett értéket. A keresési tartományunk az I2:J6 szelekció lesz, ahonnan a második oszlop celláját kérjük eredménynek. Ez fogja tartalmazni a talált karakterhez tartozó százalékos értéket. A 4. paraméternek FALSE értéket adunk, mivel egzakt értéket keresünk. Mivel ezt a képletet lefele fogjuk alkalmazni, rögzíteni kell a megfelelő cellacímeket.
- F oszlop (jövedelem RON-ban): tudjuk az összbevételt egy adott jegy után (D oszlop), valamint a jövedelem százalékát ebből (E oszlop), így a pénznembeli érték a bevételnek az adott százaléka.
- Az összértékeket (D9, F9 cellák) egy-egy *SUM* függvénnyel összeadjuk.

Megoldások:

- D oszlop: `=B2*C2`
- E oszlop: `=VLOOKUP(RIGHT(A2), $I$2:$J$6, 2, FALSE)`
- F oszlop: `=D2*E2`
- D9 cella: `=SUM(D2:D7)`, F9 cella: `=SUM(F2:F7)`

E2 =VLOOKUP(RIGHT(A2), $I$2:$J$6, 2, FALSE)

	A	B	C	D	E	F	G	H	I	J
1	Termék kód	Ár	Eladott Darabszám	Bevétel	Jövedelem (%-ban)	Jövedelem (RON-ban)			**Kód**	**Jövedelem**
2	CT2020M140M	RON 1,450.00	28	RON 40,600.00	1.24%	RON 503.44			A	0.55%
3	CS2021O422L	RON 2,115.75	42	RON 88,861.50	0.65%	RON 577.60			L	0.65%
4	XV2021J858A	RON 985.92	36	RON 35,493.12	0.55%	RON 195.21			M	1.24%
5	HM2020X182N	RON 1,701.50	9	RON 15,313.50	0.92%	RON 140.88			N	0.92%
6	HX2021P080L	RON 1,299.99	17	RON 22,099.83	0.65%	RON 143.65			V	2.25%
7	GV2021W000A	RON 1,583.96	31	RON 49,102.76	0.55%	RON 270.07				
8										
9			Összesen:	RON 251,470.71		RON 1,830.85				

4.8.2.2.2. ábra. *Jövedelemszámolás feladat megoldva*

5. OPTIMALIZÁLÁSI FELADATOK

Optimalizálási feladatok jelen vannak mind a középiskolai oktatásban, mind a közgazdasági képzésben. Például maximális nyereséget akarunk elérni minimális ráfordítással (minimális költséggel). De ha egy valós egyváltozós függvény szélsőértékeit akarjuk meghatározni (maximumát vagy minimumát), akkor is optimalizálási eljárásokat, technikákat alkalmazunk. A medián kiszámítása is történhet optimalizációval, mert:

Tétel 1:

Ha $x_1 < x_2 < \ldots < x_n$ egy rendezett minta elemei és M_e a hozzájuk rendelt medián, akkor $f : R \to R, f_{(x)} = \sum_{i=1}^{n} |x - x_i|$ függvénynek az $x = M_e$ helyen minimuma van.

Tétel 2:

Ha $x_1 < x_2 < \ldots < x_n$ egy rendezett minta elemei, $f_1, f_2, \ldots, f_n$ a megfelelő gyakoriságok és M_e a hozzájuk rendelt medián, akkor az $x = M_e$ helyen minimuma van.

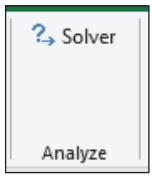
Az Excelben erre van egy nagyon hatékony eljárás, és ez a Solver. Ezt általános megoldónak is nevezhetnénk, mert sokféle feladatot meg tudunk vele oldani, és ezt a jegyzetünk számos példája is igazolja.

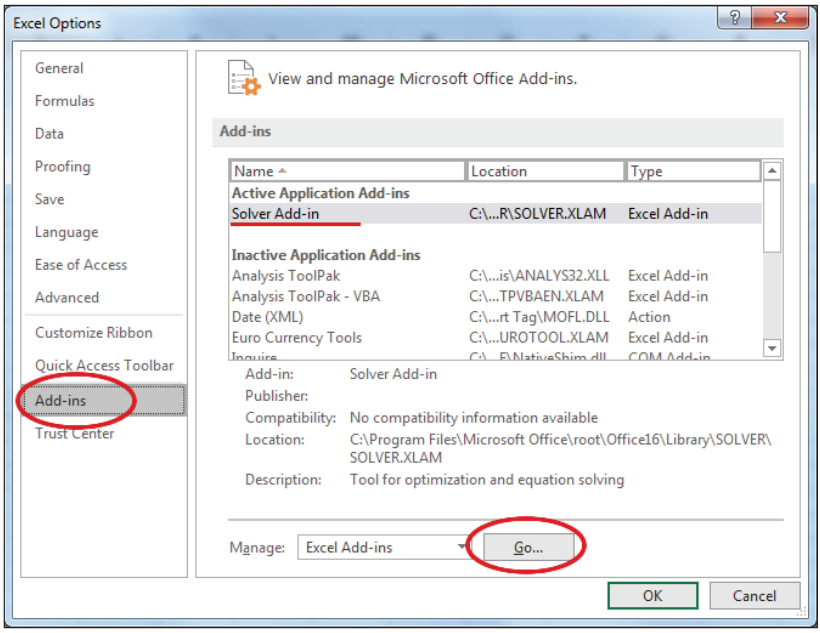
5.1. Solver

A Solver mint kiegészítő van jelen az Excelben, ezért ennek használatához először be kell kapcsolni ezt az „Add-ins” opcióknál. Ezt a „File” menü, „Options” ablakban találjuk meg (5.1.1. ábra).

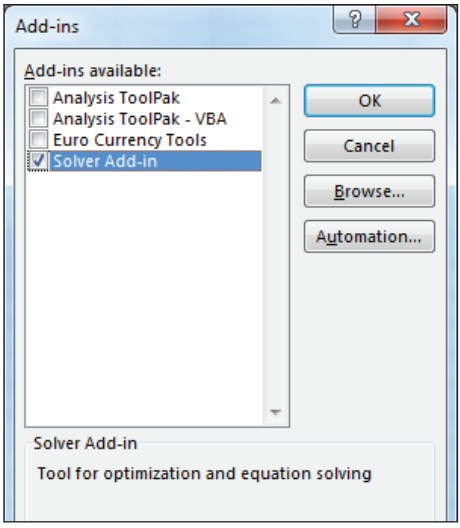
Először ellenőrizzük, hogy telepítve van-e ez, éspedig a felsorolt listában megjelenik-e a „Solver Add-in”. Ha igen, akkor a „Manage: Excel Add-ins” opcióknál be kell pipálni ezt, majd jóváhagyni az OK gomb segítségével (5.1.2. ábra).

A jóváhagyás után a „Data” fülön megjelenik a Solver gomb:



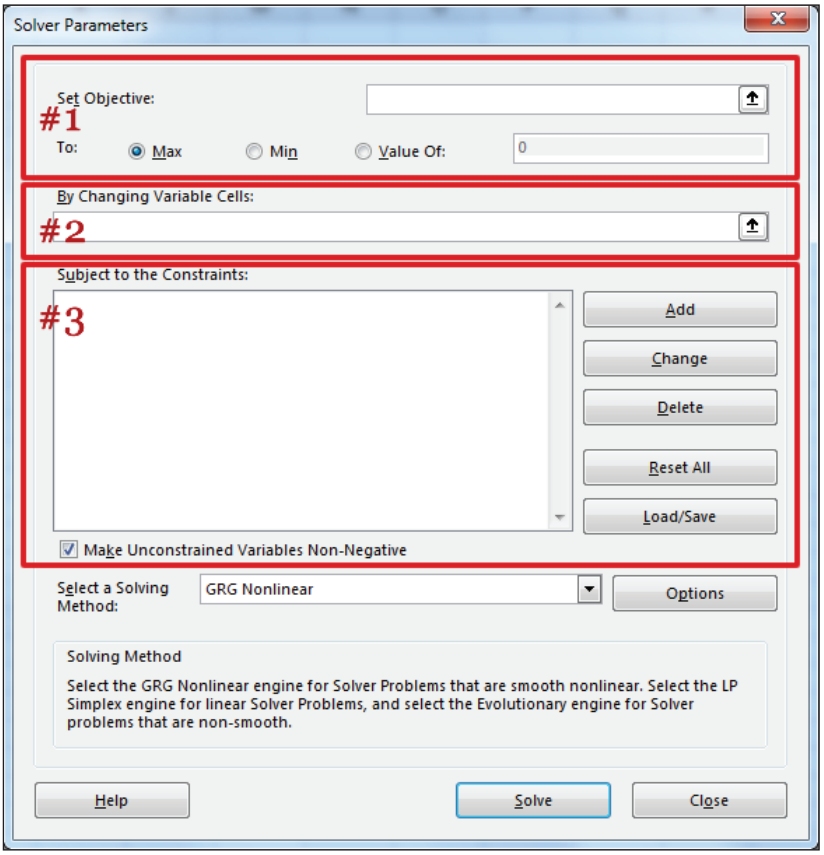


5.1.1. ábra. *Excelkiegészítők az opciókban*



5.1.2. ábra. *A Solver kiegészítő hozzáadása*

Erre rákattintva nézzük meg a Solver ablakot:



5.1.3. ábra. *A Solver részei*

A feladat megoldása szempontjából három fontos részre tudjuk osztani:

- Célcella meghatározása (#1 – „Set Objective”): itt tudjuk megadni a célcellát, illetve a rá, azaz a feladat megoldására vonatkozó feltételt, éspedig úgy oldja meg a feladatot, hogy a célunk, tehát a célcella értéke legyen maximum, minimum vagy egy adott érték.
- Változó cellák (#2 – „By Changing Variable Cells”): amint ennek a résznek a neve is mutatja, itt meg tudjuk határozni a célcella értékét, változtatva ezeket a változó cellákat. Ha több változó cellánk van, akkor ezeket mindig egy szelekcióként kell megadni, ezért fontos, hogy ezeket egymás mellé írjuk.
- Megszorítások vagy kikötések (#3 – „Subject to the Contraints”): itt lehet megadni a feladat megoldásához szükséges kikötéseket, amiket megoldás-

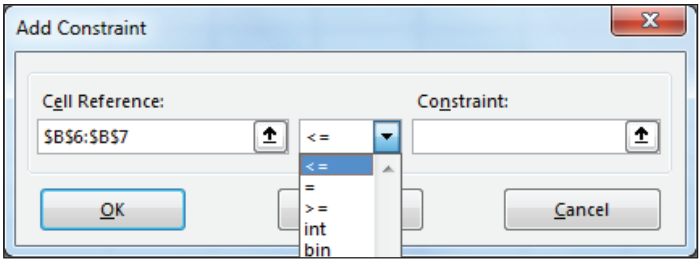
kor figyelembe kell venni a helyes eredmény meghatározásához. Továbbá egy, a megszorításokban explicit nem szereplő „változó cellák értékei ne legyenek negatívak” opciót is biztosít, ami alapból be is van jelölve, bár ez verziótól függ.

Egy további opció a megoldási módszer, ami alapból „GRG Nonlineár” és a hozzá tartozó opciók.

A fejezetben tárgyalt Solverrel megoldható feladatokat könnyedén meg lehet oldani, ha betartjuk a következő három lépést és szabályt:

- Változó cellák meghatározása: a feladat szövegéből kiindulva el kell dönteni, mik a változóink (vagy ismeretleneink), és ezeknek egy-egy cellát befogni. Hogy egyértelmű legyen, melyik cella melyik változóé, írjuk eléjük (ha függőlegesen foglaljuk le ezeket) vagy feléjük a nevüket, szerepüket. Mint említettük, ha több változó cellánk van, akkor ezeket szelekcióként kell megadni a Solver ablakban, ezért fontos, hogy egymás alá/mellé vegyük fel őket. A feladat megoldása során a Solver ezen cellák értékeit fogja változtatni a megadott megszorítások értelmében egészen addig, amíg el nem éri a megadott célt és a konvergencia a cél felé fennáll.
- Célcella meghatározása: mindig egy képlet kell legyen, melynek értéke kell függjön a változó celláktól. Másképpen mondva, ha a változó cellák értéke változik, akkor a célcella értéke is kell változzon. Ez lehet direkt módon, azaz a képlet tartalmazza a változó cellák címeit, vagy indirekt módon. Ha ez nem áll fenn, akkor a Solver hiába változtatja a változó cellák értékeit, ha a cél nem változik, nincs konvergencia, ezért nem tudja megoldani a feladatot.
- Megszorítások meghatározása: sokszor egy-egy ilyen megszorítást is képlet segítségével kell kiszámoljunk. A tévedések elkerülése érdekében itt is jó, ha beírjuk a megszorítás nevét és az esetleges megszorítást is a cella mellé, így a Solver ablakba való bevitelkor ezek átláthatóbbak.

Megszorítások esetén mindig egy vagy több cellacímet (szelekció formájában) kell megadni. Lehetséges értékek a: <=, >=, =, int (a cella értéke egész szám), bináris (0 vagy 1 értékek) és dif (alldifferent – kijelölt cellák értéke egymástól különböző legyen).



5.1.4. ábra. *Megszorítás hozzáadása a Solverben*

Fontos: Figyeljünk a kezdőértékekre. Ha ezek hibásak, vagy értékeik miatt egy képletet nem lehet kiszámolni (például nullával való osztás), akkor a Solverrel sem tudja kezdeni a számításokat. Továbbá, ha lehet, érdemes nem nulla kezdőértékeket adni a változó celláknak. Ha minden nulla, akkor a képletek értékei is mind nullák lehetnek, így nehezebb észrevenni, ha valahol elírtunk valamit. Viszont amennyiben lehet, ha 1-es kezdőértékeket adunk, fejben is könnyű utánaszámolni és ellenőrizni, hogy helyesek-e a képletek. Például: `=2*x + 3*y + z` képlet esetén, ha az x, y, z változó cellák kezdőértékei nullák, akkor a cella értéke is nulla lesz. Ellenben ha 1-et adunk kezdőértéknek, akkor ennek értéke 6 kell legyen. Ha nem ennyi, akkor rögtön észre lehet venni, hogy valamit elírtunk ebben.

Solver modell mentése, betöltése (Save/Load)

Alapból a Solverbe beírt feltételek, cellacímek és értékek a munkalappal együtt el lesznek mentve. Ez azt jelenti, hogy az Excel állomány bezárása, majd későbbiekben való megnyitása után is megmaradnak. Külön feladatok esetén érdemes ezeket külön-külön munkalapon oldani. Azonban ha adott feladat esetén több megoldással próbálkozunk, akkor a „Save/Load" gomb segítségével el lehet menteni a Solverbe éppen beírtakat vagy újra betölteni ezeket. A mentés/betöltés egy megadott cellától kezdve, egymás alatti cellákból történik.

Megoldás és hibák megértése

Mint említettük, a Solver megáll, ha megfelelő megoldást kapott vagy megszűnik a konvergencia, azaz a célcella értéke tovább nem konvergál, vagy akár hiba történik, mint például valamelyik cella, amivel osztunk, nullás értéket kap. Ilyenkor két lehetőséget kínál: tartsuk-e meg a jelenlegi értékeket, vagy állítsa vissza a változó cellák eredeti értékeit. Utóbbi igencsak hasznos, ha sok a változó cella és konkrét értékektől kell indulni.

Egyszerű feladatok esetén a Solver hamar kaphat megoldást, viszont bonyolultabbak vagy esetleg rosszul megadottak esetén ez nem biztos. Ilyenkor a következő leggyakoribb hibák egyikét kaphatjuk:

- Hiba a modellben („Error in model") vagy Érték hiba („Solver encountered az error value"): a számolandó cella értéke hibás, például a szám mező értéke szöveg vagy egyéb érvénytelen érték, amivel a Solver nem tud dolgozni.
- A Solver nem tud javítani a jelenlegi megoldáson („Solver cannot improve the current solution"): a Solver eljutott egy megközelítő megoldásig, de ennél lehet jobb megoldása is. Ennek érdekében finomítani lehet a Solver Opciók beállításain.
- Célcella értéke tovább nem konvergál („The Set Target Cell values do not converge"): olyan esetben fordul elő, ha a Solver tovább változtatja egy vagy több változó cella értékét, mert a megszorítások még nem akadályozzák

meg ezt, viszont a célcella értéke nem változik. Ilyen esetben a Solver nem tud tovább számolni. Ez leggyakrabban hibás képletek esetén fordul elő.

- A Solver nem kap eredményt („Solver could not find a feasible solution”): leggyakoribb előfordulása hibás logika szerint megadott feltételek, rosszul megírt képletek vagy hibás számítás esetén (például nullával való osztás), amikor a Solver egyetlen megoldást sem kap, ami minden megszorítást kielégít.

Példa:

A Solver szemléltetése céljából vegyünk egy nagyon egyszerű feladatot. Legyen a következő táblázat, mely egy diák jegyeit tartalmazza.

	A	B
1	**Tantárgy neve**	**Jegy**
2	Birotika	7
3	Matematika	6
4	Angol	7
5	Tervezés	9
6	Programozás	?
7	Adatbázisok	?
8	**Média:**	**7.25**

5.1.5. ábra. *Diák jegyei példa*

Az eddig kapott jegyek alapján a médiája `7.25`, viszont két tantárgyból, programozásból és adatbázisokból még nem vizsgázott. A kérdés, hogy mekkora jegyeket kell kapjon ebből a két tantárgyból, hogy az átlagmédiája kereken `8.00` legyen.

Megoldás:

Tartsuk be a fent említett lépéseket a feladat megoldása esetén.

1. Változó celláink: mit nem ismerünk? A két tantárgy jegyeit, amiből még nem vizsgázott a diák. Ezek a cellák a B6 és B7. Kezdőértéknek hagyjuk üresen a cellát, mert a ’?’ karakter esetén nem fog tudni számolni a Solver.

2. Célcella: célunk, hogy a média `8.00` legyen. A diák médiáját az *AVERAGE* függvény segítségével ki tudjuk számolni, de vigyázzunk, hogy a két „ismeretlen” cella is legyen benne, azaz `=AVERAGE(B2:B7)`, különben a Solver hiába változtatja a B6, B7 cellák értékeit, a média nem fog változni, ezért nem tudja megoldani.

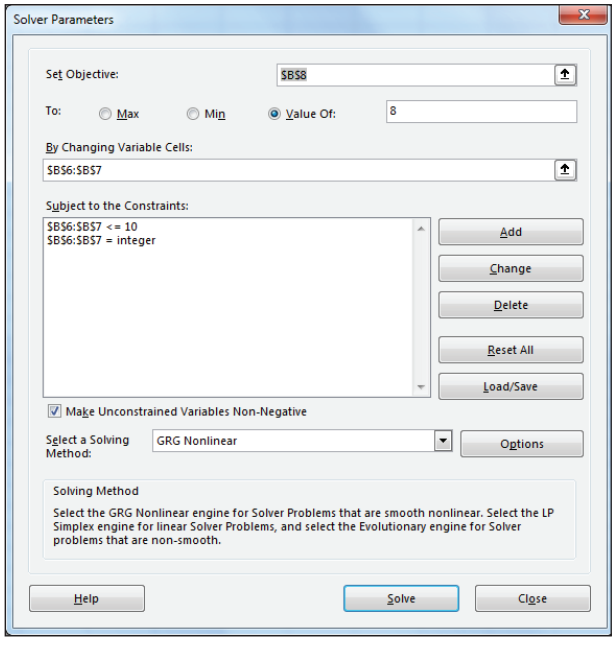
3. Kikötések: ha jelen esetben oldjuk meg a feladatot, akkor a következő megoldást kaphatjuk:

	A	B
1	**Tantárgy neve**	**Jegy**
2	Birotika	7
3	Matematika	6
4	Angol	7
5	Tervezés	9
6	Programozás	10.5
7	Adatbázisok	8.500003
8	**Média:**	**8.000001**

5.1.6. ábra. *Diák jegyei példa részleges megoldás*

Ezzel több probléma is van: nincs 10-esnél nagyobb jegy (10.5), vagy tizedeseket tartalmaz (8.50003), ezért kikötéseknek rögzítsük le a következőket:

- A jegyek legyenek egész számok.
- Egy jegy maximum 10-es lehet és minimum 1-es. Az utóbbit el lehet hagyni, mivel a célcella kezdeti értéke kisebb, mint a kívánt érték, ezért a Solver egy pozitív konvergenciát próbál elérni, így nem fogja a változó cella értékeit negatívba vinni.



5.1.7. ábra. *Egész számok megszorítás hozzáadása a Solverhez*

Ezek után már elfogadható eredményt kapunk, éspedig a két változó cella eredménye 10 és 9 lesz. Természetesen egy másik megoldás a 9 és 10. A Solver nem az összes megoldást fogja megadni, hanem az elsőt, ami megfelel a célnak.

5.2. Feladatok

5.2.1. Közgazdasági optimizálási feladat (1)

Számold ki az $x + y + z \to max$, adva a következő feltételeket:

$x+2y+z \leq 12$

$yz \leq 6$

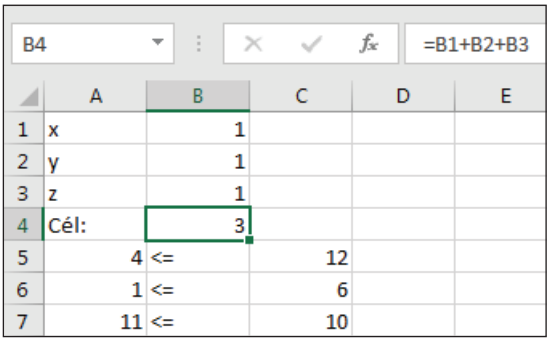
$8x+3z \leq 10$

és y egész szám.

Megoldás: Kövessük a fent említett lépéseket:

- Változó cellák: összesen három ismeretlenünk van: x, y és z, vegyük ezeknek a B1, B2 és B3 cellákat, eléjük (A oszlop) viszont írjuk oda, melyik cella melyik változó értékét jelenti. Kezdőértékeiknek írhatunk 1-et.
- Célcella: itt az $x + y + z$ értékét kell kiszámítsuk, ami maximum kell legyen. Mivel a változóknak 1-es kezdőértéket adtunk, itt a képlet felírása után 3 kell legyen ennek eredménye.
- Kikötések: három feltételünk van, ezeket számoljuk ki egymás alatti cellákba, és írjuk melléjük az értékeikre való feltételt.

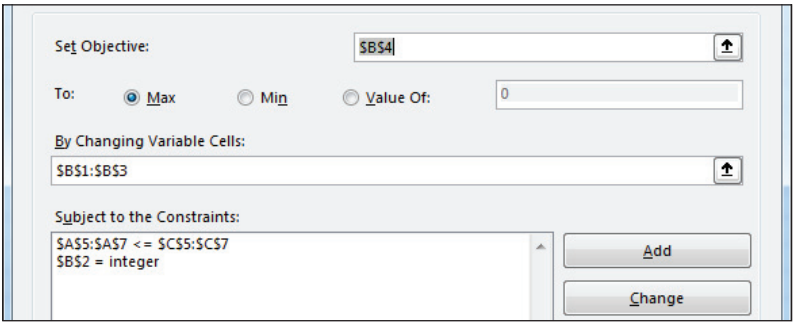
Ezeket felírva, a következő táblázatot kapjuk:



5.2.1.1. ábra. *Feladat Excelben*

Ha ez készen van, akkor Solverbe vigyük fel ezeket. Mivel a megszorításokat egymás alatti cellákba írtuk, ezért ezt a három megszorítást egyetlen feltételben (egy-

egy szelekcióként) meg tudjuk adni a Solverben. Alternatívaként meg lehet adni ezeket, mint három különböző feltételt, ez nem fogja befolyásolni az eredményt.



5.2.1.2. ábra. *Solver előkészítése*

Ha mindent jól vittünk be, akkor célnak 7 és változó celláknak pedig *x*: 0.8, és *y*: 5 és z: 1.2 értékeket kell kapjunk. Ez azt jelenti, hogy a Solver ezekkel az *x*, *y* és *z* értékekkel megkapta a célképletnek a maximumot.

5.2.2. Közgazdasági optimizálási feladat (2)

Oldd meg a következő feladatot:

$$z + \frac{x+y}{2x} \geq 120$$

$$x + 3y \leq 200$$

$$x, y, z \geq 0 \text{ és } z - \textit{egész szám}$$

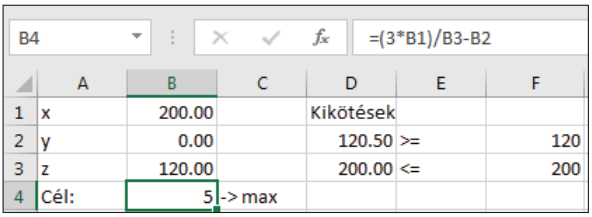
$$\textit{Cél}: \frac{3x}{z} - y \rightarrow max$$

Megoldás: Kövessük a fent említett lépéseket:

- Összesen három változónk van: *x*, *y* és *z*, vegyük ezeknek a B1, B2 és B3 cellákat. Kezdőértékeiknek írhatunk 1-et.
- Ezután vegyünk egy tetszőleges cellát, például B4-et, és írjuk fel a cél képletét, használva a változó cellákat: `=(3*B1)/B3-B2`.
- A megszorításokat, éspedig a két egyenlőtlenséget, valamint azt, hogy *z* egész szám, számoljuk ki egy-egy cellában, és írjunk utánuk ezek értékeire vonatkozó megszorításokat. Vigyázzunk a zárójelezésre a törtek esetén!

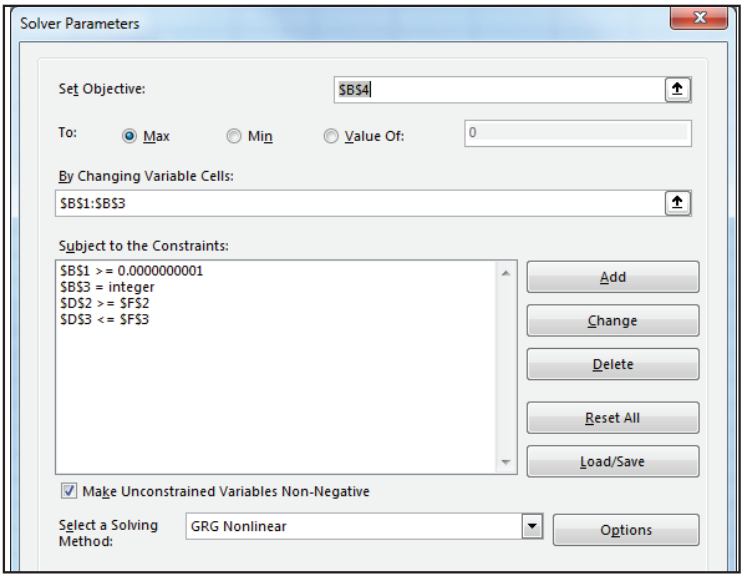
Ha szigorúan csak a feladatban meghatározott feltételeket visszük fel a Solverbe és meg akarjuk oldani, akkor értékhibát kapunk, mivel az *x* változó értéke nulla

lett és osztunk *x*-szel, ezért nullával való osztást kapunk. Ennek elkerülése egy újabb megszorítást igényel. Sajnos Solverben nincs nagyobb vagy kisebb megszorítás, ezért nem lehet megadni, hogy nagyobb legyen, mint 0 (ennek magyarázata, hogy bizonyos esetekben a konvergencia nagyon hosszú időt venne igénybe). Ilyen esetben be kell érni egy megközelítő megoldással, éspedig nagyobb vagy egyenlő, mint egy kicsi szám. Az Excelben a legkisebb pozitív tizedes szám `2.2E-308`, azonban a Solver feltételként ezt nem tudja kezelni, ezért próbálkozni kell egy nagyobb értékkel. Ez nagyban függ a verziótól, de például az `1E-10`-et általánosan elfogadja. A megoldás esetén az x értéke nagyobb lesz ennél, és mivel a feltétel megköveteli, hogy pozitív legyen, ezért ez a kiválasztott alsó küszöb megfelelő.



5.2.2.1. ábra. *Feladat Excelben*

A Solver megoldásként célnak 5-öt kapott, úgy, hogy az *x* értéke 200, *y* nulla és *z* pedig 120.



5.2.2.2. ábra. *Solver előkészítése*

5.2.3. Fürdőkádszerelő cég

Egy helyi luxus fürdőkádat szerelő cég a következő két változatot szereli: AquaSpa és Hydro-Lux. Ezek szereléséhez szükséges a kádon kívül pompa, hozzá tartozó csővezeték és egy bizonyos számú munkaóra:

5.2.3.1. táblázat. *Feladat adatai*

Fürdőkádtípus	Munkaóra	Csővezeték	Pompa	Jövedelem
AquaSpa	9	12	1	350 €
Hydro-Lux	6	16	1	300 €

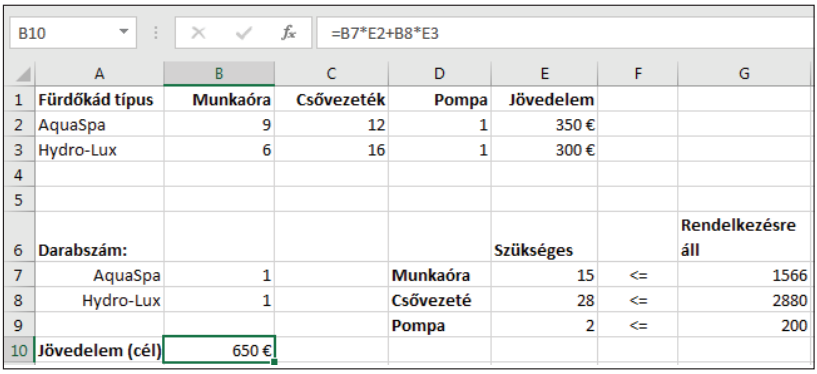
Forrás: FUQUA School of Business – Excel & Solver: Hands-On Modeling Practice Exercises

A nagyszámú kliens miatt a cég megengedi magának, hogy megválogassa a szerelési időpontokat. A kérdés, hogy melyik típusú fürdőkádból hány darabot szereljen egy hónapban, hogy maximalizálja jövedelmét, tudva, hogy 200 darab pompa és 2,880 méter csővezeték áll a rendelkezésre, valamint az alkalmazottak összesen 1,566 munkaórát tudnak szerelésre fordítani.

Útmutató:

- A cég két fürdőkádat szerel, melyek száma határozza meg az összjövedelmet, és ezek darabszáma az ismeretlen. Ebből kifolyólag két ismeretlenünk van, melynek vegyünk egy-egy cellát, például B7 és B8 (az A7 és A8 cellákba írjuk be, melyik darabszám melyik fürdőkádat jelenti). Kezdeti értéknek adjunk 1-et mindkét cella esetén. Ez a következőkben segít könnyedén ellenőrizni a képleteket.
- Az összjövedelmet a szerelt fürdőkádak száma határozza meg. Felhasználva az előző pontban vett cellákat (B7 és B8), számoljuk ki a jövedelmet ezen cellák értékeivel, azaz ahány darabot szerelünk az AquaSpa fürdőkádból, annyiszor 350 €, plusz amennyi darabot szerelünk a Hydro-Luxból, annyiszor 300 € a jövedelem. Mivel 1-et írtunk a darabszám kezdeti értékének, most könnyen lehet látni, hogy 650 € jött ki a célnak. Ha 2-t írunk a változó celláknak, akkor 1300 € kell megjelenjen a célcella értékének. Ha nem, akkor hibás a képlet.
- Megszorítások: darabszám (változócella értékei) szerint ki kell számolni a felhasznált kellékeket (pompa és csővezeték), valamint a szükséges munkaórák számát. Vegyünk ezeknek egy-egy cellát, és számoljuk ki ezeket, valamint írjuk utánuk az értékeikre vonatkozó feltételeket. Ha a változó cellák kezdeti értékeinek egyet-egyet vettünk, akkor könnyű ellenőrizni ezeket a képleteket. Ellenben ha a kezdőértékeket 0-nak vagy üresnek hagytuk volna, akkor minden képlet eredménye most nulla lenne, így nehezebb észrevenni, ha valahol elírtunk valamit.

Példaként a táblázatot fel lehet írni a következő módon:



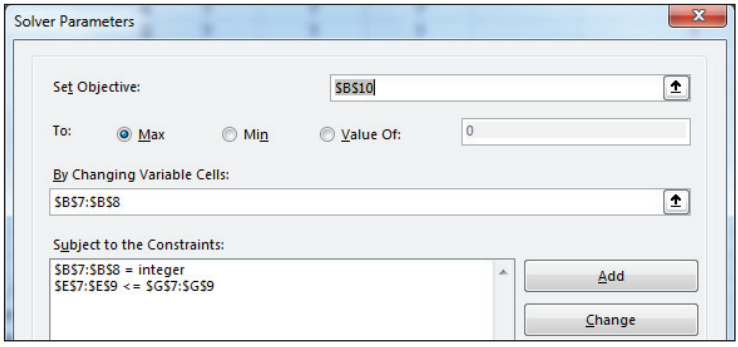
	A	B	C	D	E	F	G
1	**Fürdőkád típus**	**Munkaóra**	**Csővezeték**	**Pompa**	**Jövedelem**		
2	AquaSpa	9	12	1	350 €		
3	Hydro-Lux	6	16	1	300 €		
4							
5							
6	**Darabszám:**				**Szükséges**		**Rendelkezésre áll**
7	AquaSpa	1		**Munkaóra**	15	<=	1566
8	Hydro-Lux	1		**Csővezeté**	28	<=	2880
9				**Pompa**	2	<=	200
10	**Jövedelem (cél)**	650 €					

5.2.3.2. ábra. *Feladat Excelben felírva*

Fontos megfelelően elkészíteni az adatokat, hiszen annál könnyebb a Solverbe felvinni. Ha ezt elkészítettük, akkor nem marad más hátra, mint Solverrel megoldani a feladatot. Eredmény: 66,100.00 €.

Megoldás:

- Célcella: `=B7*E2+B8*E3`.
- Szükséges munkaórák száma: `=B7*B2+B8*B3`.
- Szükséges csővezeték: `=B7*C2+B8*C3`.
- Pompák száma: `=B7+B8` (mivel egy-egy szükséges, ezért nem kell beszorozni 1-gyel, hanem elég a fürdőkádak darabszámát összeadni).
- Solver: plusz feltétel, hogy a szerelt fürdőkádak száma egész szám legyen, ugyanis nem lehet töredék kádat kiszállítani és felszerelni egy adott hónapban. Mivel a szükséges és rendelkezésre álló cellákat egymás alá írtuk, ezért egy feltételben meg lehet adni ezeket.



5.2.3.3. ábra. *Solver előkészítése*

Megoldásnak 66,100.00 € úgy, hogy az AquaSpából 122 darabot, míg a HydroLuxból 78-at szerelünk, így szükség van mind az 1566 munkaórára, valamint 2712 méter csővezetékre és mind a 200 darab pompára.

5.2.4. Asztalosműhely

Egy asztalosműhely három típusú asztalt gyárt exportra, A, B, B festetlenül és C modelleket. A gyártási folyamat az alapanyag előkészítését/felvágását, szerelését és festését foglalja magába.

5.2.4.1. táblázat. *A feladat adatai*

	Munkadíj: óra / darab			
Modell	Előkészítés (vágás)	Szerelés	Festés	Jövedelem
A	1	2	4	35 €
B	2	4	4	40 €
B (festetlen)	2	4	0	20 €
C	3	7	5	50 €
Kapacitás / hónap	200	300	150	

Forrás: FUQUA School of Business – Excel & Solver: Hands-On Modeling Practice Exercises

Melyik típusból hány darabot lehet legyártani ahhoz, hogy a jövedelem maximális legyen (Feladat 1)? Hogy változik ez, ha minden típusból legalább 10 darabot kell gyártani (Feladat 2)?

Útmutató:

- Változó cellák: az asztalok típusai lesznek, azaz 4 darab cella kell a négy típusnak.
- Cél: a jövedelmet kell maximalizálni, ezt pedig a legyártott modellek darabszámai határozzák meg, éspedig minden legyártott A típusú asztal után 35 € a jövedelem, plusz a B típus stb. Ezeket összesítsük egy cellában, darabszámnak felhasználva az előző pontban felvett változó cellákat.
- Megszorítások: havonta csak annyi asztalt lehet legyártani, amennyit a munkakapacitás megenged, ezért számoljuk ki a különböző manoperafázisok igényelt óraszámát a változó cellák függvényében. Ezeket írjuk egymás alatti cellákba, majd melléjük a rendelkezésre álló maximum óraszámokat, hogy könnyebb legyen majd a Solverbe felvinni.

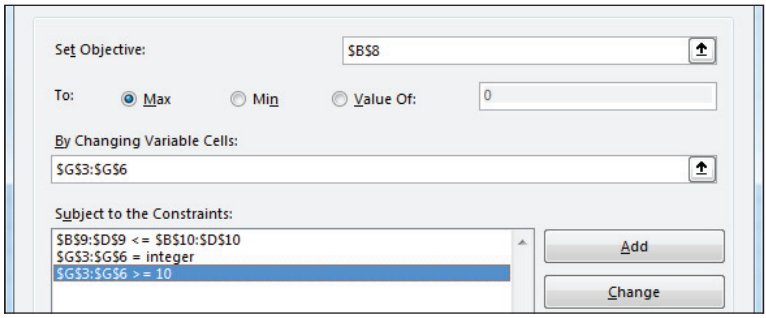
A táblázatot felírhatjuk az alábbi formában. Ha készen vagyunk, akkor a Solver segítségével meg lehet ezt oldani. Elvárt eredmény az első kérdésre: 2,420 €. Ha mindenből legalább 10 darabot le kell gyártani, akkor a jövedelem: 2,225 €.

	A	B	C	D	E	F	G
1		**Manopera: óra / darab**					
2	Modell	Előkészítés (vágás)	Szerelés	Festés	Jövedelem		Darabszám
3	A	1	2	4	35 €		1
4	B	2	4	4	40 €		1
5	B (festetlen)	2	4	0	20 €		1
6	C	3	7	5	50 €		1
7							
8	Összjövedelem:	145 €					
9	Szükséges óraszám:	8	17	13			
10	Rendelkezésre álló óraszám:	200	300	150			

5.2.4.2. ábra. *Feladat Excelben felírva*

Megoldás:

- A fenti ábra szerint válasszuk a G3–G5 cellákat változóknak, soronként az adott típusú asztal darabszámát jelképezve.
- A célcella esetén össze kell szorozni egyenként a legyártott típusok darabszámát (G oszlop) a típusuk szerinti jövedelemmel, például `G3*E3 + …`, viszont egyszerűbben lehet a *SUMPRODUCT* függvény segítségével, éspedig: `=SUMPRODUCT(E3:E6, G3:G6)`.
- A megszorítások esetén a B9, C9 és D9 cellák értékeit szintén könnyebb a *SUMPRODUCT* segítségével, például a B9 cella esetén: `=SUMPRODUCT(B3:B6,$G$3:$G$6)`. Viszont ha rögzítjük a G oszlop celláit, akkor a többit nem is kell felírni, hanem elegendő ezt alkalmazni jobbra a C9 és D9 cellákra. További megszorítás, hogy a legyártott asztalszámok egész számok legyenek, valamint pozitív értékek. Utóbbit viszont nem szükséges megadni, mivel a célcella értékét maximalizáljuk, ezért ezek nem fognak elmenni negatív értékbe (ez csökkentené a célt). Mivel egymás fölött vannak a feltételcelláink, ezért ezeket egy-egy szelekcióként egyetlen feltételben meg lehet adni.
- A második feladat esetén egy plusz feltételt kell betenni, éspedig hogy a változócelláink értéke nagyobb vagy egyenlő legyen 10-zel.



5.2.4.3. ábra. *Solver előkészítése*

Hasonlítsuk össze a két eredményt:

5.2.4.4. táblázat. *Feladat 1 és 2 eredménye*

	Legyártott asztalok száma				
	A típus	B típus	B (festetlen) típus	C típus	Jövedelem
Feladat 1	36	1	56	0	2,420 €
Feladat 2	15	10	40	10	2,225 €

Az eredmények alapján a „B” és a „C” típusú asztalból nem jövedelmező gyártani. Az első eredmény esetén az „A” és a „B (festetlen)” típusú asztalok legyártása után megmaradt munkaórákból még egy darab „B” típusú asztal kijön, ami szintén növeli a jövedelmet. A második feladat esetén a jövedelem kevesebb, mivel munkaórát kell áldozni e két kevésbé jövedelmező asztal 10-10 darabjának a legyártására, így kevesebbet lehet gyártani a jobban jövedelmező másik két modellből.

5.2.5. Cukrászda

Egy cukrászda háromfajta fánkot készít: simát, csokisat és sajtosat. Ezek alapanyagköltségei, munkaerőigényük és eladási áraik a következők:

5.2.5.1. táblázat. *Árusított fánkok adatai*

Fánk	Alapanyagköltség	Munkaerőigény (órában)	Eladási ár
Sima	1.00	0.05	1.50
Csokis	1.50	0.09	3.50
Sajtos	2.00	0.10	4.50

A napi eladásokból ítélve a következők a vásárlók igényei:

5.2.5.2. táblázat. *Cukrászda minimum és maximum eladásai*

Fánk	Minimum	Maximum
Sima	150	250
Csokis	100	200
Sajtos	50	150

Feladat: Állapítsd meg, hogy a különböző típusú fánkból hány darabot készítsen a cukrászda, hogy maximális profitot eredményezzen, tudva, hogy alkalmazottaik nyitás előtt összesen 30 órát tudnak dolgozni, és feltételezve, hogy az összes fánkot eladják.

Útmutató:

- Változó cellák megállapítása: a cukrászda három típusú fánkot készít, a kérdés, hogy ezekből egyenként hány darabot készítsen? Válasszunk három cellát ezek értékeinek. Hogy ne tévesszük össze a cellákat, az A oszlopba írjuk fel a fánkok neveit, és a B oszlopba vegyük ezek darabszámait (ezek lesznek a változócelláink).
- Célnak a maximum jövedelmet kell megállapítani, ezért ki kell számolni, hogy egy-egy darab fánk után mennyi a jövedelem, majd darabszám szerint összesen mennyi ez. Ez utóbbi lesz a célcellánk.
- A megszorítások pedig a minimum, maximum darabszámra és a munkaórára vonatkoznak. Itt ki kell számolni, hogy megadott darabszámú fánk esetén mennyi időt igényel ezek előállítása, valamint mindegyik darabszám a megadott minimum és maximum darabszám között legyen.

Példaként ezt a táblázatot a következő formában lehet felírni:

	A	B	C	D	E	F	G	H	I	J
1				**Jövedelem**					**Munkaidő**	
2	**Fánk**	**Darabszám**	**Alapanyag**	**Eladási ár**	**Összesen**		**/ darab**		Min	Max
3	Sima	?	RON 1.00	RON 1.50	?		0.05		150	250
4	Csokis	?	RON 1.50	RON 3.50	?		0.09		100	200
5	Sajtos	?	RON 2.50	RON 4.50	?		0.10		50	150
6						**Összesen:**	?	<= 30		
7				**Cél: (max)**	?					

5.2.5.3. ábra. *Feladat Excelben felírva*

Ezt felírva, előkészítettünk mindent a *Solver* számára, így a továbbiakban be kell vinni az adatokat, hogy számolja ki.

Megoldás:

- B oszlop: változócellák kezdőértékeinek adjunk 1-et, ezzel könnyebb ellenőrizni a további képleteket.
- E oszlop: egy fánk után a nyereségünk az eladási ár és az alapanyagok közötti különbség, *n* darab után pedig *n*-szer ennyi. Kiszámoljuk ezt az első fánkra: `=(D3-C3) * B3`, majd a képletet alkalmazzuk a másik két sorra. A legvégén egy *SUM* függvénnyel összeadjuk ezeket, megkapva így az összjövedelmet. A célcella képlete indirekt módon függ a változó celláktól, tehát ha ezek értékei változnak, az összjövedelem is újra lesz számolva.
- G6 cella (szükséges munkaidő összesen): itt az egyes termékek darabszámai szerint ki kell számolni, hogy összesen mennyi munkaidőt vesz fel ezek elkészítése. Például a „Sima" fánkból ahány darabot készítünk, annyiszor G3 cellában levő időt vesz igénybe, plusz a többi termék. Így a G3 cella képlete: `=B3*G3 + B4*G4 + B5*G5` lesz. Ennek kiszámítása fontos, mert csak egy megadott idő áll az alkalmazottak rendelkezésére nyitás előtt.

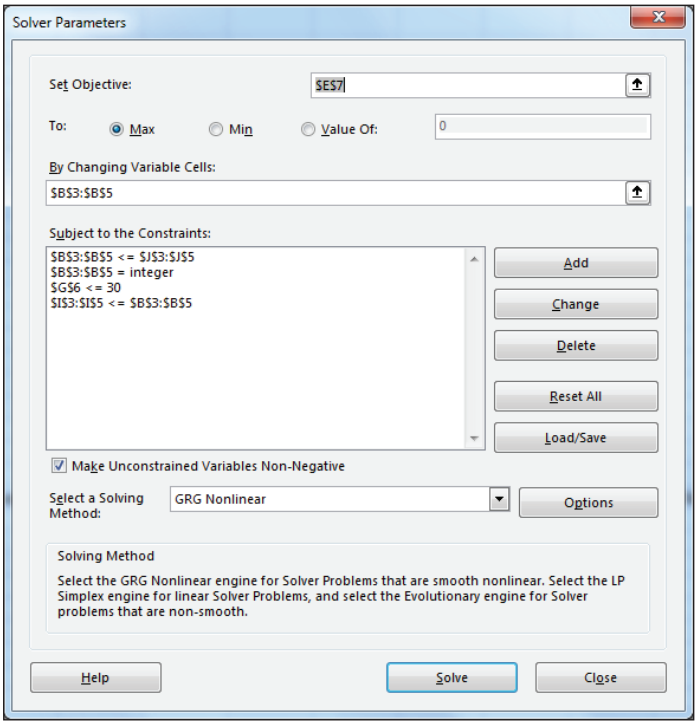
	A	B	C	D	E	F	G	H	I	J
1				**Jövedelem**			**Munkaidő**			
2	**Fánk**	**Darabszám**	**Alapanyag**	**Eladási ár**	**Összesen**		**/ darab**		Min	Max
3	Sima	150	RON 1.00	RON 1.50	RON 75.00		0.05		150	250
4	Csokis	190	RON 1.50	RON 3.50	RON 380.00		0.09		100	200
5	Sajtos	54	RON 2.50	RON 4.50	RON 108.00		0.10		50	150
6						**Összesen:**	30	<= 30		
7				**Cél: (max)**	RON 563.00					

5.2.5.4. ábra. *Cukrászda feladat táblázata előkészítve*

Ha a táblázatot előkészítettük, akkor nem marad más, mint a *Solver*be is bevinni ezeket (5.2.5.5. ábra).

Több cella esetén, ha a szelekció száma talál, akkor a „kisebb vagy egyenlő mint" kikötést egyetlen feltételben is meg lehet adni, például `$B$3:$B$5 <= $J$3:$J$5`. Természetesen extrafeltétel, hogy a fánkok száma egész szám legyen.

Megoldásnak *563.00 RON* jövedelmet kaptunk, úgy, hogy a sima fánkból a minimum *150* darabot kell elkészíteni, a csokisból *190* darabot, míg a sajtosból *54* darabot.



5.2.5.5. ábra. *A Solver paraméterei*

5.2.6. Villanymotorgyár

Egy kínai villanymotorgyár háromfajta elektromos motort gyárt. Ezek gyártása tekercselést és szerelést igényel, minden folyamat egy bizonyos számú munkaórát vesz igénybe.

5.2.6.1. táblázat. *Villanymotorok gyártási adatai*

	Modell 1	**Modell 2**	**Modell 3**
Tekercselés (óra / db)	2	1.5	3
Szerelés (óra / db)	1	2	1

Forrás: FUQUA School of Business – Excel & Solver: Hands-On Modeling Practice Exercises

Az egy hónap alatt legyártandó 750000 € rendelés a következő: Modell 1 – 3000 darab, Modell 2 – 2000 darab és Modell 3 – 900 darab. A rendelés kielégítésére egy hónap áll a rendelkezésre. A gyár havonta 10000 munkaórát tud tekercselésre és 5000 munkaórát pedig szerelésre fordítani. Ha a megrendelt darabszámú villanymotort nem tudja legyártani, akkor a konkurenciától meg tudja vásárolni ezeket, viszont ez költségesebb, mint legyártani:

5.2.6.2. táblázat. *A gyártás és rendelés költségei*

Költség	Modell 1	Modell 2	Modell 3
Legyártani	50 €	83 €	130 €
Megvásárolni	60 €	97 €	145 €

Forrás: FUQUA School of Business – Excel & Solver: Hands-On Modeling Practice Exercises

Feladat: Melyik modellből mekkora mennyiséget kell legyártani, illetve rendelni ahhoz, hogy a rendelés kielégítése a legkisebb költséggel járjon? Továbbá mekkora a rendelés utáni jövedelem, feltéve, hogy az egész gyár csak ezen a rendelésen dolgozik az adott hónapban?

Útmutató:

- A változó cellákat nézve, a feladatot kétféleképpen is meg lehet oldani:
  4. 6 darab változó cellával: 3 darab a 3 legyártandó modellnek és 3 másik cella a megrendelendő modellek darabszámának. Jelöljük a legyártott modellek számát x_1, x_2 és x_3-mal a Modell 1, 2 és 3 esetén, a megrendelt darabszámokat pedig y_1, y_2 és y_3-mal.
  5. 3 darab változó cellával: ebben az esetben a 3 cella a legyártandó modellek darabszámát jelenti. Ebben az esetben azt, hogy hány darabot kell rendeljünk, külön ki kell számolni, például a „Modell 1” esetén ez:$3000 - x_1$ (ugyanígy minhárom modell esetén).
- A célcella esetében a rendelés előállítási költségét kell meghatározni, és ez kell a lehető legkisebb legyen, azaz minimalizálni kell ezt. Ennek értéke a 6 változós megoldás esetén: $50x_1 + 60y_1 + 83x_2 + 97y_2 + 130x_3 + 60y_3 \rightarrow min$.
- Kikötések: tárgyaljuk külön ezeket.
  1. Talán a legfontosabb a rendelés darabszáma, azaz minden modell esetén a legyártott darabszám plusz a megrendelt darabszám összege egyenlő kell legyen a rendelt darabszámmal: $x_1 + y_1 = 3000$, $x_2 + y_2 = 2000$ és $x_3 + y_3 = 900$. Ugyanakkor egyik sem lehet negatív, sem a legyártott, sem a megrendelt darabszám. Mivel a célképlet minimumát keressük, ezért ezt külön meg kell adni a *Solver*nek, ellenkező esetben, ha például a C modell esetén 2776 darabot legyártunk és -1876 darabot megrendelünk, a végeredmény így is 900 darab lesz.

2. Másik fontos feltétel, hogy gyártás esetén az alkalmazottak tekercseléssel csak 10,000 munkaórát, szereléssel pedig 5000 munkaórát tudnak eltölteni. Azaz $2x_1 + 1.5x_2 + 3\,x_3 \leq 10000$ és $x_1 + 2x_2 + x_3 \leq 5000$. Természetesen a megrendelt modellekre nem kell számolni ezeket a munkaórákat, mivel nem az itteni gyárban készülnek.
3. A motorok darabszámai egész számok kell legyenek, ami egyértelmű, mivel nem tudunk például 1/3 motort legyártani, míg a maradék 2/3-át megrendelni. Mivel az összdarabszámot is nézzük, ezért elég csak a legyártott darabszámokat egész számnak venni.

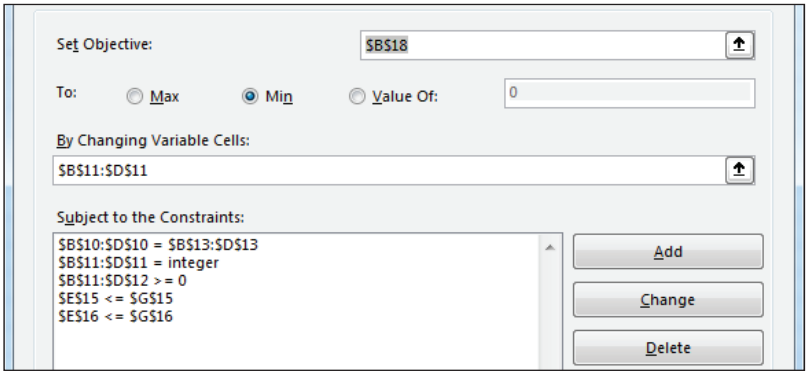
Megoldás:
Írjuk fel a táblázatot a következő formában:

B18 =B11*B6+C11*C6+D11*D6+B12*B7+C12*C7+D12*D7

	A	B	C	D	E	F	G
1		Modell 1	Modell 2	Modell 3			
2	Tekercselés (óra / db)	2	1.5	3			
3	Szerelés (óra / db)	1	2	1			
4							
5	Költség	Modell 1	Modell 2	Modell 3			
6	Legyártani	50 €	83 €	130 €			
7	Megvásárolni	60 €	97 €	145 €			
8							
9	Darabszám						
10	Rendelés	3000	2000	900			
11	Legyártani	3000	550	900			
12	Megvásárolni	0	1450	0			
13	Összesen:	3000	2000	900			
14					Összesen		
15	Tekercselés	6000	825	2700	9525	<=	10000
16	Szerelés	3000	1100	900	5000	<=	5000
17							
18	Költség:	453,300 €	-> min				

5.2.6.3. ábra. *Feladat Excelben felírva*

Ha 6 darab változó cellát használunk, akkor a *Solver*nek a `$B$11:$D$12` szelekciót kell megadni, 3 változó cellás megoldás esetén pedig a `$B$11:$D$11` szelekciót. A 10-es és a 13-as sorban levő cellák értékei mindkét esetben egyenlőek kell legyenek. A tekercselés és a szerelés egy-egy külön feltétel. A példától eltérően, ahol ezek modellenként vannak kiszámolva, majd összegezve (15 és 16-os sor), ezeket egy-egy cellában is ki lehet számolni, megspórolva így a `B15:D16` cellák használatát.



5.2.6.4. ábra. *A Solver paraméterei*

Eredmény: A minimális költség, amiből a rendelést elő lehet állítani, 453300 €, éspedig az 1-es és 3-as modelleket teljes mértékben legyártjuk, a 2-es modellből pedig 550 darabot legyártunk, és a megmaradt 1450 darabot pedig megrendeljük.

6. STATISZTIKA EXCEL SEGÍTSÉGÉVEL

A leggyakoribb statisztikai függvényeket fogjuk használni, az *AVERAGE* (ÁTLAG), *MIN*, *MAX*, *STDEV* (*SZÓRÁS*), *MODE* (*MÓDUSZ*), *MEDIAN* (*MEDIÁN*) stb. Sok függvénye van az Excelnek, a szakterületünktől függően átlagban 20-30 függvény ismerete bőven elegendő. Nagyon jó útmutató és konkrét példa van a függvényekhez rendelve, ezek segítségével könnyen eligazodunk a használatukban.

A közgazdászhallgatóknak talán az Excel statisztikai lehetősége az egyik leghasznosabb alkalmazhatósága. És nem annyira a statisztikai függvényeken van a hangsúly, hanem a módszereken, vagy más szóval az algoritmusokon. Például a módusz egy statisztikai értékhalmaz középértéke, vagyis a rendezett sorozatban a középen álló érték, ha a sorozat elemei páratlanok. Ha páros a számossága a statisztikai sornak, akkor 2 elem áll középen, és annak vesszük az átlagát. De lényegében a gyakorlati életben csak nagyon ritkán tudjuk alkalmazni az Excel *MODE* függvényét, mert az csak szimpla adatsornál használható, és a gyakorlati életben a legtöbbször összetett adatsorral dolgozunk. Ez a megállapításunk igaz nem csak a mediánra, de a szórásra, móduszra stb.

A medián meghatározása: A mediánt csak akkor tudjuk meghatározni, ha az adathalmazunk rendezhető. A medián a rendezett adatsor közepe. Ha az adatsor számossága páratlan, akkor pontosan egy közepe van, *2k+1* db értéknél a *k+1*. Ha páros az adathalmazunk számossága, akkor a medián a középen lévő két adat számtani átlaga. Vagyis *2k* számosságú adatsornál a medián a *k.* és *(k+1).* adat átlaga:

$$Me = \frac{a_k + a_{k+1}}{2}, a_1 \le a_2 \le \ldots \le a_{2k}$$

Nagyon fontos, hogy dolgozhatunk névleges adatokkal, vagyis olyanokkal, amelyek csak minőségi ismérvvel adottak, pl. a hallgatók szeme színe, hajszíne, családi állapota stb. Ilyen adathalmaznál csak a móduszt, vagyis a leggyakrabban előforduló ismérvet (adatot) tudjuk meghatározni. Rendezhető adatoknál számolhatunk mediánt is, míg mérhető adatoknál bármilyen középérték kiszámítható. Például ha van egy névsorunk, akkor abban a középen lévő hallgató, vagy a névsorban középen lévő két hallgató, de ha csak a névsor ismert, még nem tudunk számtani középértéket számolni (minek az átlagát számolnánk?). Ezért fontos, hogy minden középérték jellemezze valamilyen formában a megfelelő adathalmazt, amiből származik.

A fent elmondottakat egy egyszerű példával szeretnénk bemutatni:

Feladat: 95 anyától megkérdezték, hogy hány éves volt, amikor az első gyermekét szülte.

A válaszokat a következő táblázat tartalmazza:

életkor	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
gyakoriság	5	3	7	5	3	4	2	5	7	3	4	6	2	4	8	9	11	7

Adjuk meg, melyik a leggyakoribb életkor az első gyerek szülésére.

Megoldás: Erre a választ a csoportosított adatsor mediánja adja.

Itt csak akkor lehet alkalmazni a *MEDIAN* függvényt, ha kifejtjük, hogy 18 éves volt 5 anya, 19 éves 3 anya, 20 éves 7 anya stb. Ebben az esetben lehet alkalmazni a mediánt, de ez nagyon „favágó" módszer!

Tehát a sorban kifejtettük, hogy melyik értékből hány darab van:

Most már használható a *MEDIAN* függvény `=MEDIAN(A3:K20)`, és meg is adja, hogy az összetett adatsorunk mediánja 28!

életkor	gyakoriság	halmozás	
18	5	5	TRUE
19	3	8	TRUE
20	7	15	TRUE
21	5	20	TRUE
22	3	23	TRUE
23	4	27	TRUE
24	2	29	TRUE
25	5	34	TRUE
26	7	41	TRUE
27	3	44	TRUE
28	4	48	FALSE
29	6	54	FALSE
30	2	56	FALSE
31	4	60	FALSE
32	8	68	FALSE
33	9	77	FALSE
34	11	88	FALSE
35	7	95	FALSE
Összegzés	95		

6.1. ábra. *Gyakoriségi táblázat a medián kiszámítására*

A medián kiszámítására nincs képlet. Rendezni kell az adatokat növekvő sorrendben, és a középen álló érték adja a mediánt. (Ha páratlan a próbák száma, akkor van középen álló érték, ha páros, akkor két érték áll középen, és ennek a két középen álló értéknek vesszük a számtani átlagát).

18	18	18	18	18						
19	19	19								
20	20	20	20	20	20	20				
21	21	21	21	21						
22	22	22								
23	23	23	23							
24	24									
25	25	25	25	25						
26	26	26	26	26	26	26				
27	27	27								
28	28	28	28							
29	29	29	29	29	29					
30	30									
31	31	31	31							
32	32	32	32	32	32	32	32			
33	33	33	33	33	33	33	33	33		
34	34	34	34	34	34	34	34	34	34	34
35	35	35	35	35	35	35				

6.2. ábra. *A gyakoriság kifejtése a MEDIAN egyszerű használata céljából*

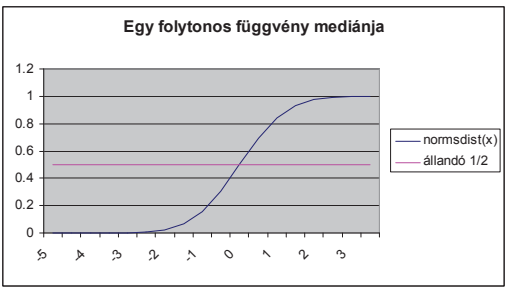
Empirikusan úgy oldjuk meg, hogy rendezzük az adatsort, majd halmozzuk a gyakoriságokat. Így könnyen látható, melyik adat áll középen. De ha beillesztjük azt a képletet, hogy `=D25<95/2`, vagyis az aktuális cella kisebb, mint a sokaság számosságának a fele, akkor így is megkaphatjuk.

(Ahol átvált az „igaz” logikai érték „hamissá”.)

Ha folytonos az eloszlásunk, akkor a medián az $F(x_{med})=1/2$ egyenlet megoldása, ahol F az eloszlásfüggvény. Az $F(x)=1/2$ legtöbb esetben egy implicit függvény x-re, ezért érdemes, hogy Excelben lássunk rá egy megoldási technikát.

Először is rajzban érdemes megérteni:

x	normsdist(x)	1/2
-5	2.86652E-07	1/2
-4.5	3.39767E-06	1/2
-4	3.16712E-05	1/2
-3.5	0.000232629	1/2
-3	0.001349898	1/2
-2.5	0.006209665	1/2
-2	0.022750132	1/2
-1.5	0.066807201	1/2
-1	0.158655254	1/2
-0.5	0.308537539	1/2
0	0.5	1/2
0.5	0.691462461	1/2
1	0.841344746	1/2
1.5	0.933192799	1/2
2	0.977249868	1/2
2.5	0.993790335	1/2
3	0.998650102	1/2
3.5	0.999767371	1/2

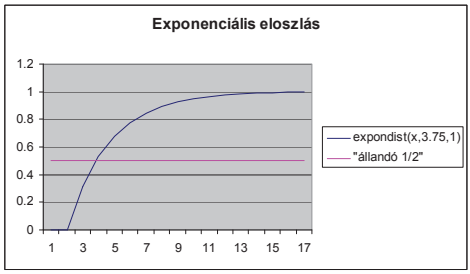


6.3. ábra. *A medián szemléltetése normál (folytonos) eloszlás esetén*

Ez egy olyan „standard normális” folytonos eloszlás, hogy $F(x)=1/2$ egyenlet megoldása $x=0$!

De a legtöbb esetben nem ilyen egyszerű a megoldás. Például nézzük az exponenciális függvényt.

x	expondist(x)	1/2
0	0	1/2
0.1	0.312710721	1/2
0.2	0.527633447	1/2
0.3	0.675347533	1/2
0.4	0.77686984	1/2
0.5	0.846645033	1/2
0.6	0.894600775	1/2
0.7	0.927560243	1/2
0.8	0.950212932	1/2
0.9	0.965781882	1/2
1	0.976482254	1/2
1.1	0.983836505	1/2
1.2	0.988891003	1/2
1.3	0.992364906	1/2
1.4	0.994752482	1/2
1.5	0.996393437	1/2



6.4. ábra. *A medián szemléltetése exponenciális (folytonos) eloszlás esetén*

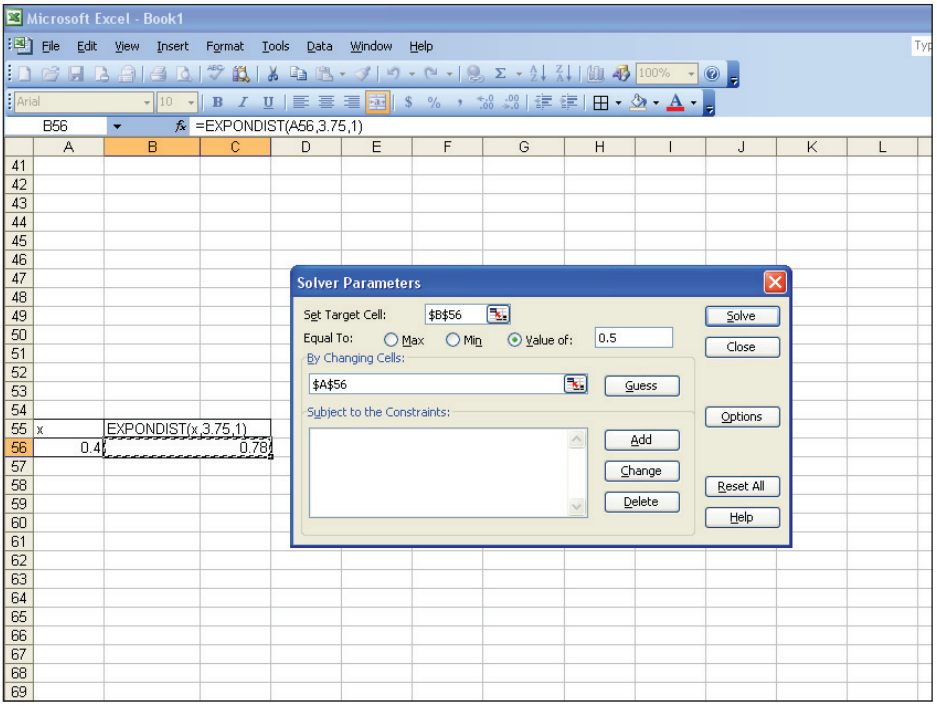
Itt érdemes használni a *Solver*t. Felvesszük a következő cellatartományt, és kényszerítjük, hogy a 0.78 érték alatt 0.5 legyen

x	EXPONDIST(x,3.75,1)
0.4	0.78

6.5. ábra. *Solver használatának előkészítése*

életkor	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
gyakoriság	5	3	7	5	3	4	2	5	7	3	4	6	2	4	8	9	11	7

6.6. ábra. *Adattábla*



6.7. ábra. *Solver használata a medián kiszámítására*

És a Solver szépen megoldja:

x	EXPONDIST(x,3.75,1)
0.184839	0.50

6.8. ábra. *A Solver eredménye*

6.1. A szórás kiszámítása

Az átlag után a másik legfontosabb statisztikai mutató a szórás, mely az adatok változatosságát, szóródását méri, azaz hogy mennyire szóródnak az átlag körül. Ha az összes adatunk egyforma, vagyis konstans adatsorunk van, akkor azok szórása nulla, mert nincs szóródás, nincs változatosság stb. A közgazdasági adatsorokban általában összetett adatsorral dolgozunk, és ezért képlettel kell kiszámítani a szórást. A helyzet teljesen analóg a mediánnál elmondottakkal. Ezért itt is a következő adatsoron ismertetjük a szórás kiszámítását:

A feladat: 95 anyától megkérdezték, hogy hány éves volt, amikor az első gyermekét szülte.

A válaszokat a következő táblázat tartalmazza:

6.1.1. táblázat. *Gyakorisági adattábla*

életkor	gyakoriság	=fi*(xi-átlag)^2
18	5	466.8831025
19	3	225.1509141
20	7	411.0679224
21	5	221.9883657
22	3	96.21407202
23	4	86.9801662
24	2	26.83745152
25	5	35.46204986
26	7	19.36265928
27	3	1.31933518
28	4	0.453850416
29	6	10.72288089
30	2	10.92166205
31	4	44.53806094
32	8	150.4655956
33	9	256.3369529
34	11	441.7112465
35	7	376.8047645
összesen	95	2883.221053

Ha így vannak megadva az adataink, akkor képlettel kell kiszámítani az adatok szórását.

És így a szórás egyenlő 5.51-gyel.

$$\sqrt{\frac{f_i(x_i-átlag)^2}{n}} = \text{stdevp vagy } \sqrt{\frac{f_i(x_i-átlag)^2}{n-1}} = \text{stdev}$$

De figyelem: az Excel belső függvényét, *STDEV* vagy *STDEVP* csak akkor tudjuk alkalmazni, ha az adatokat kifejtettük, akárcsak a mediánnál:

18	18	18	18	18						
19	19	19								
20	20	20	20	20	20	20				
21	21	21	21	21						
22	22	22								
23	23	23	23							
24	24									
25	25	25	25	25						
26	26	26	26	26	26	26				
27	27	27								
28	28	28	28							
29	29	29	29	29	29					
30	30									
31	31	31	31							
32	32	32	32	32	32	32	32			
33	33	33	33	33	33	33	33	33		
34	34	34	34	34	34	34	34	34	34	34
35	35	35	35	35	35	35				

6.1.2. ábra. *Az adattábla kifejtése a STDEV egyszerű használatához*

Ebben az esetben rá tudunk mutatni a felső téglalap alakú tartományra, és meghívjuk a *STDEV* vagy *STDEVP* függvényünket. (A kettő között csak annyi a különbség, hogy kevés adat esetén n-1-gyel osztunk. Hogy mennyi a kevés adat, azt a téma és statisztikai interpretáció dönti el! Ezt korrigált szórásnak nevezzük.)

6.1.3. táblázat. *A számítások összehasonlítása*

Excel belső függvényével	stdev	=STDEV(A3:K20)	=SQRT(C43/(B43-1))
	stdevp	=STDEVP(A3:K20)	=SQRT(C43/B43)

Az eredmény pedig:

6.1.4. táblázat. *Az összehasonlítás eredménye*

Excel belső függvényével	stdev	5.53828	5.538282
	stdevp	5.50906	5.509056

Az Excelben a leíró statisztika alkalmazása nem a függvényeken alapszik, hanem a módszereken, és igazából csak az eloszlás és sűrűségfüggvények alkalmazása az, ami igazán értékessé teszi az Excel használatát. Régebben a klasszikus

eloszlásokat: Binomiális, Poisson, Normális, Exponenciális, táblázatolták, és a hallgatók ezekből a táblázatokból keresték ki a megfelelő értékeket. Ma már sokkal nehezebb ezekhez a táblázatokhoz hozzájutni, mint például az Excelhez.

Lássuk ennek alkalmazását:

A csibai deszkagyárban a deszkák kiszabására beállított gáter úgy működik, hogy a deszkák hossza normális eloszlású, $m = 2.5$ m várható értékkel és $\sigma = 4$ cm szórással.

1. A deszkák hány százaléka lesz 2.4 és 2.6 m között?
2. Mekkora annak a valószínűsége, hogy a deszkák hossza 2.5 m-től legfeljebb 2 cm-rel térjen el?

Excellel tudjuk a leggyorsabban megválaszolni az előbbi két kérdést:

Fontos, hogy átalakítsuk: 4 cm=0.04 m, tehát $\sigma = 0.04$ m.

Így:

$$P(2{,}4 \le x \le 2{,}6) = \text{NORMDIST(2 .6,2.5,0.0 4,1) - NORMDIST(2 .4,2.5,0.0 4,1)} = 0.987581.$$

A második kérdésre a választ így írhatjuk le:

$$P(|2{,}5 - x| \le 0.02) = P(2{,}5 - 0{,}02 \le x \le 2{,}5 + 0{,}02) =$$
$$\text{NORMDIST(2.52,2.5,0.04,1) - NORMDIST(2.48,2.5,0.04,1)} = 0.382925.$$

Ha nem lenne Excel a kezünk alatt, illetve számítógép sem, akkor a normális eloszlást át kellene alakítsuk standard normális eloszlássá, és táblázatból kellene kinézzük a megfelelő értékeket! De például a csíkszeredai hallgatók is könnyebben hozzáférnek az Excelhez, mint az ilyen táblázatokhoz!

Lássuk a normális eloszlásnak a használatát Excelben:

`NORMDIST(x,m,σ,0/1)` m átlagú és σ szórású eloszlás x értékét számolja ki. Ha az utolsó paraméter értéke 1, akkor növekvő eloszlásfüggvény, ha 0, akkor sűrűségfüggvény. (Magyar változatban `NORM.ELOSZL(x,m,σ,0/1)`). Ha a standard normális eloszlásra lenne szükségünk, akkor `NORMSDIST(x)`, magyar változatban pedig `STNORMELOSZL(x)`. Itt is megjegyezzük, hogy a román változatban az Excel függvények nevét nem fordították le románra, azokat a román verzióban is angolul kell használni.

Nagyon fontos, hogy az elméletből ismert tulajdonságokat az Excelben is azonnal alkalmazzuk, vagyis ha X ismérv normális eloszlású m átlaggal és σ szórással, és az x1, x2 ismérvértékek, akkor

$$P(x \le a) = NORMDIST\ (a, m, \sigma, 1)$$

$$\mathrm{P(x \ge a) = 1 - NORMDIST\ (a, m, \sigma, 1)}$$

$$P(x \in [x1, x2]) = P(x1 \le x \le x2) = NORMDIST(x2, m, \sigma, 1) - NORMDIST(x1, m, \sigma, 1).$$

Természetesen azt is tudjuk az elméletből, hogy ha X folytonos valószínűségi változó (amit a statisztikában így is mondhatunk, hogy X folytonos jelenséget modellező ismérv), akkor:

$$P(a \le x \le b) = P(a < x \le b) = P(a < x < b) = P(a \le x < b).$$

Lássunk ennek alkalmazására egy fontos példát:

Egy megoldott statisztikai feladat

Egy őröltkávé-csomagoló gép 20 dkg várható súlyú csomagokat készít 3 g szórással.

a) Mekkora annak a valószínűsége, hogy egy vásárolt csomagban 22 dkg-nál több kávét vásároljunk?
b) Mekkora annak a valószínűsége, hogy egy vásárolt csomagban 19 dkg-nál kevesebb kávét vásároljunk?
c) Mekkora annak a valószínűsége, hogy egy vásárolt csomagban a kávé súlya 19 és 21 dkg között legyen?
d) Mekkora annak a valószínűsége, hogy egy vásárolt csomagban 15 dkg-nál kevesebb kávét vásároljunk?
e) Mekkora annak a valószínűsége, hogy egy vásárolt csomagban 20 dkg-nál kevesebb kávét vásároljunk?

Megoldás:

$$P(x \ge 22) = 1 - \text{NORMDIST}(2\ 2,20,0.3,1\ ) = 1.3084E\ -11.$$

$$P(x < 19) = \text{NORMDIST}(1\ 9,20,0.3,1\ ) = 0.00042906.$$

$$P(19 \le x \le 21) = \text{NORMDIST}(2\ 1,20,0.3,1\ ) - \text{NORMDIST}(1\ 9,20,0.3,1\ ) = 0.999142.$$

$$P(x < 15) = \text{NORMDIST}(1\ 5,20,0.3,1\ ) = 1.14507E\ -62.$$

$$P(x < 20) = \text{NORMDIST}(2\ 0,20,0.3,1\ ) = 0.5.$$

Érdemes egy kicsit kommentálni a kapott valószínűségeket!

a) Az 1.3084E-11 érték gyakorlatilag nulla, hiszen nagyon-nagyon kicsi szám! Világos, hogy a cég tönkremenne, ha 22 dkg-nál több kávét csomagolna.
b) Az is világos, hogy annak a valószínűsége „nagyobb”, hogy 19 dkg-nál kevesebbet kapjunk, mint az a) kérdés, de lényegileg ez is nulla. Másképpen a fogyasztóvédelem büntetné meg a céget.
c) Annak a valószínűsége, hogy a csomagok súlya 19 és 21 dkg között legyen, gyakorlatilag biztos, vagyis a valószínűség szinte 1.
d) Világos, hogy a d) valószínűség jóval kisebb, mint a b) valószínűség, hiszen az eloszlásfüggvény egy monoton növekvő függvény.

e) Annak valószínűsége, hogy 20 dkg-nál kevesebb kávét vásároljunk, elég nagy, 0,5, természetesen annak is 0,5 a valószínűsége, hogy 20 dkg-nál több kávét vásároljunk! Ugyanis 20 dkg a medián is a folytonos eloszlások esetén! Érdemes ezeken egy picikét elgondolkodni, hogy kialakuljon a helyes szemléletünk és valószínűségi intuíciónk.

7. MAKRÓK ÉS ŰRLAPELEMEK

A makró egy rögzített műveletsor, melyet a felhasználó könnyedén újra és újra végre tud hajtatni a számítógéppel. Segítségével az adott programban elvégezhető feladatok szinte mindegyike automatizálható. A makrók használata időt takarít meg és kibővíti a nap mint nap használt alkalmazások lehetőségeit.

Makrót létrehozhatunk rögzítéssel, vagy magunk is írhatjuk őket. A két módszert kombinálva is lehet használni. A makró egy Visual Basic modulban tárolt program, vagy parancs- és függvénysorozat, előre rögzített tevékenységek sora.

A makrók ugyan programkódok, de használatuk nem követel meg programfejlesztői szaktudást, de még programozási ismereteket sem. Az Office-alkalmazásokban létrehozható makrók többsége a Microsoft Visual Basic for Applications (VBA) elnevezésű programnyelven készül. A VBA-ról a következő fejezetben írunk részletesebben.

A makrók a dokumentumhoz kapcsolódva tárolódnak, azok részét képezik, ezért vírusok esetén veszélyt is jelenthetnek. Ebből kifolyólag a makrók használatát engedélyeztetni kell, így a makrókészítés előtt bizonyos előkészületeket kell tennünk:

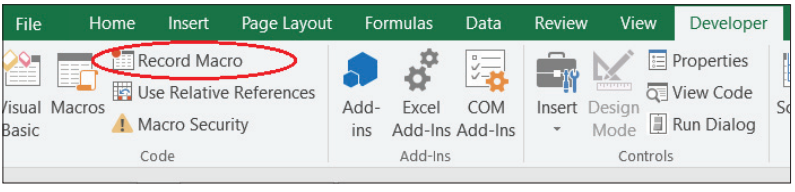
Fejlesztőeszközök lap megjelenítése: Fájl (File) menü → Beállítások (Options) → Menüszalag testreszabása (Customize Ribbon) → Fejlesztőeszközök (Developer) jelölőnégyzet kipipálása a Fő Lapok (Main Tabs) terület jobb oldalán.

Összes makró ideiglenes engedélyezése: Fejlesztőeszközök Lap (Developer) → Kód (Code) csoport → Makróvédelem gomb (Macro Security) → Összes makró engedélyezése (Enable all macros).

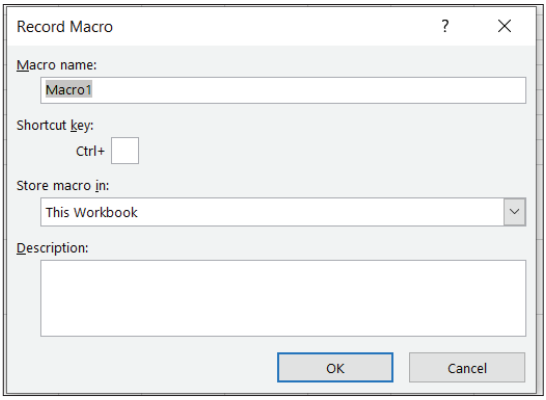
7.1. Makrórögzítés

Az ismétlődő feladatok automatizálásához célszerű makrókat rögzíteni az Excel makrórögzítőjével.

Makró rögzítése: Fejlesztőeszközök Lap (Developer) → Kód (Code) csoport → Makró rögzítése gomb (Record Macro) → megjelenik a Makrórögzítés panelje



7.1.1. ábra. *Fejlesztőeszközök lapon a Makrórögzítés*



7.1.2. ábra. *Makrórögzítés panel*

- Makrónév (Macro name): Maradhat az automatikus név, illetve meg is változtathatjuk: betűkből és számokból állhat, betűvel kezdődjön, nem tartalmazhat szóközt.
- Billentyűparancs (Shortcut key): Ctrl+… Ez egy gyorsindítókód lesz, pl. Ctrl+b, vagy Ctrl+Shift+B. Két makrónak nem lehet ugyanaz a gyorsindítókódja, viszont ideiglenesen felülírhatnak már létező alapértelmezett Excel-billentyűparancsokat (pl. Ctrl+c).
- A makró helye (Store macro in): *Egyéni makró-munkafüzetben* (minden Excel alkalmazásban elérhető) vagy Új munkafüzetben vagy *Ebben a munkafüzetben*.
- A makró leírása (Description): Röviden leírható, hogy mit végez el az adott makró, a rögzítés dátumával, valamint a rögzítő személynevével befejezve. A leírás tetszés szerinti.

Az OK gombra kattintva elkezdődik a rögzítés, ami addig tart, amíg meg nem nyomjuk a *Rögzítés vége* gombot (Stop Recording). A rögzítés vége előtt felvett tevékenységsorozat a makró feladata, ilyenkor az Office lépésenként tárolja el az egyes parancsok végrehajtásának adatait.

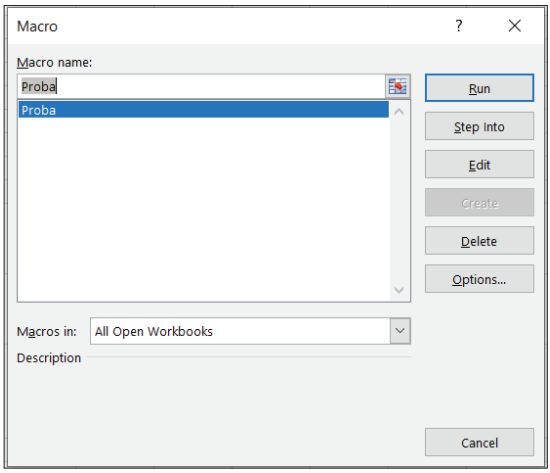
Fontos, hogy makrórögzítéskor a makrórögzítő szinte az összes elvégzett műveletet rögzíti.

- Ha rögzítés közben jelölünk ki egy adott tartományt, és utána hajtunk végre egy bizonyos feladatsort, a rögzítés felveszi a kijelölést is, és a kijelölt tartományon hajtja végre a feladatsort.
- Ha a feladatot tetszőleges tartományon szeretnénk végrehajtani, akkor a rögzítés megkezdése előtt jelöljünk ki egy tetszőleges tartományt, és a rögzítés alatt ne kattintsunk egyetlen cellára sem.
- Ha hibát követünk el a makrórögzítés során (pl. nem megfelelő tartományt jelöltünk ki, vagy olyan gombra kattintottunk, amire nem kellett volna),

a makrórögzítő azt is rögzíti. Ilyenkor vagy rögzítsük újból a teljes folyamatot (miután töröltük a nem tökéletes makrónkat), vagy módosítsuk a makrónk VBA-kódját a feladatnak megfelelően.

Makró futtatása (lejátszása):

1. A létrehozáskor megadott billentyűparancs-kombináció lenyomásával.
2. Készíthetünk egy indító vezérlőelemet, amihez hozzákapcsoljuk a makrót.
3. Fejlesztőeszközök Lap (Developer) → Makrók (Macros) gomb.



7.1.3. ábra. *Makrók panel*

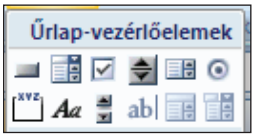
- Indítás (Run): lefuttathatjuk az adott makrót.
- Lépésenként (Step Into): az adott makrót soronként futtatjuk le.
- Szerkesztés (Edit): megtekinthető a makró forráskódja, itt a Visual Basic szerkesztőablakában a meglévő makrót módosíthatjuk.
- Törlés (Delete): a kiválasztott makrót törölhetjük vele.
- Egyebek (Options): a kiválasztott makróhoz billentyűkombinációt rendelhetünk, és megadhatunk egy leírást.
- Mégse (Cancel): bezárjuk a párbeszédpanelt.

7.2. Űrlapvezérlő elemek

Űrlapok célja: adatbevitel megkönnyítése – adatok megadása, módosítása egérrel; adatkijelölés könnyebbé tétele

Űrlapvezérlők használatával makrók is futtathatók. Egy meglévő makrót hozzákapcsolhatunk egy vezérlőhöz, vagy új makrót írhatunk vagy rögzíthetünk. Amikor a felhasználó egy ilyen vezérlőre kattint, az elindítja a makrót.

Vezérlők beszúrása: Fejlesztőeszközök Lap (Developer) → Beszúrás (Insert) → Űrlapvezérlő elemek (Form Controls)



- **Gomb** (Button) : Olyan vezérlő, amelyet virtuálisan meg lehet nyomni. A Gomb megnyomásakor végrehajtódik a hozzárendelt makró.
- **Kombi panel** vagy **legördülő lista** (ComboBox): Egy vagy több szöveges elemből álló listát jelenít meg, melyből a felhasználó választhat. 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.
- **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-e jelölve vagy sem, vagyis egyidejűleg több is kiválasztható.
- **Léptetőnyíl (SpinButton)**: Két egymástól elfelé mutató nyíl. Segítségével egy cella értékét tudjuk növelni vagy csökkenteni a megadott határok között a megadott lépésközzel.
- **Listapanel (Listbox)** : Olyan mező, amiben egy felsorolás található, melyből a felhasználó választhat. A csatolt cellában a választott elem sorszáma lesz.
- **Választógomb vagy Rádiógomb (Optionbutton)** : Egy kis karika felirattal. Egyetlen választást tesz lehetővé a több lehetőség közül. Kijelölt és nem kijelölt állapota lehet. A kijelölt állapotot a karikában egy pont jelzi. Ezek a vezérlők együtt működnek. Több választókapcsolóból egyszerre csak egy lehet kijelölve. Ha másikat választunk ki, az előző kijelölésünk megszűnik.
- **Görgetősáv (ScrollBar)** : A léptetőgomb 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.

Űrlapelemek tulajdonságai:

Miután beszúrtuk az űrlapelemet, az űrlapelemre jobb egérgombbal kattintva a *Vezérlő formázása (Format Control)* menüponttal tudjuk beállítani az adott elemet.

Vezérlő formázása – Vezérlő (Control) fül – Cellacsatolás (Cell Link): megadjuk, hogy az elem értéke melyik cellába kerüljön be (ezt az értéket később fel lehet használni a képletekben, függvényekben). Ha a vezérlő értéke módosul, akkor a csatolt cella értéke is, illetve fordítva:

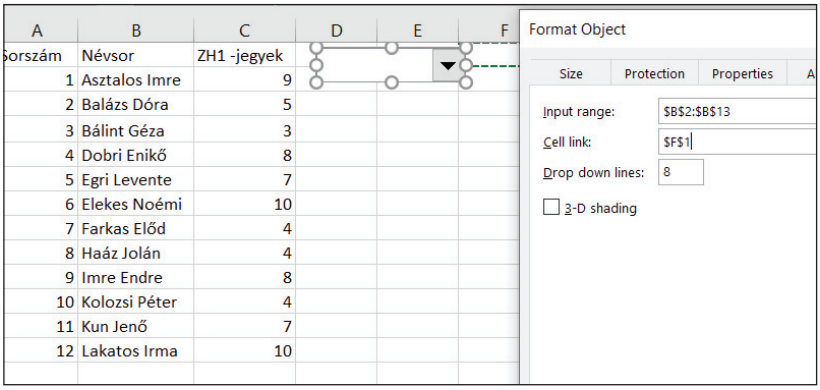
- Jelölőnégyzet esetén: IGAZ vagy HAMIS érték.
- Beviteli lista: a kiválasztott sor száma.
- Választógombok: annak az opciónak a sorszáma, amelyik éppen be van jelölve.
- A Legördülő lista és a Listapanel esetén a kiválasztott listaelem sorszáma. Ennél a két űrlapelemnél a beviteli listához meg kell adni azt a tartományt, ahonnan a felajánlott adatok származnak. Megadható a lista legördülő elemeinek száma is, ízlés és hely szerint módosítható.
- A léptetőnyilak és a görgetősáv egy számot ír a csatolt cellába, ennek a legkisebb és legnagyobb értékét is beállíthatjuk, de megadható az is, hogy a nyíl gombokra kattintva mekkora legyen a léptetési távolság (Incremental change). A görgetősáv esetén az ugrási távolság is megadható (Page change), vagyis hogy mennyivel növelje vagy csökkentse a csatolt cella értékét, ha a nyíl és a csúszka közé kattintunk.
- Választógomboknál sok beállítási lehetőség nincs, csak a kezdeti értéket adhatjuk meg a cellacsatoláson kívül. Fontos, hogy ha több választógomb-csoportot akarunk kezelni, ezeket külön-külön csoportokba is kell foglalnunk, mivel egy csoportnyi választógomb közül egy lehet kijelölve. A csoportba foglalást a Csoportpanel (Group Box) elemmel lehet megtenni [XYZ], ami ugyanott található, mint a többi űrlapelem.

Makró-hozzárendelés (Assign macro...): segítségével makrót rendelhetünk a megfelelő vezérlőhöz.

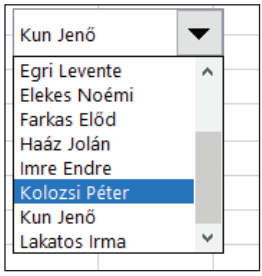
Miután jobb egérgombbal kattintunk egy vezérlőre, annak méretét és elhelyezését szabadon változtathatjuk az egér segítségével a vezérlők oldalain és sarkain lévő méretező négyzetek segítségével, illetve a vezérlő közepére kattintva húzással.

Példák:

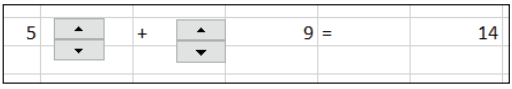
1. Legördülő lista, ahol a beviteli tartomány a neveket tartalmazza: 7.2.1., 7.2.2. ábra.
2. Két léptetőnyíl két szám összeadására: 7.2.3. ábra.
3. Három választógomb, az első kiválasztva: 7.2.4. ábra.



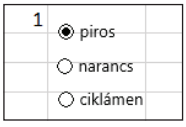
7.2.1. ábra. *Legördülő lista formázása*



7.2.2. ábra. *Legördülő lista tartalma*



7.2.3. ábra. *Léptetőnyilak használata*



7.2.4. ábra. *Választógombok használata*

7.3. Kitűzött feladatok

1. Rögzítsünk makrót, amely az A1:D2 tartomány háttérszínét pirosra, betűszínét fehérre, szegélyét zöldre, betűtípusát 12-es Times New Roman dőlt betűsre állítja.

2. Rögzítsünk makrót, amely a C oszlopot sárgára, a 4:9 sorokat kékre, a metszetet pedig zöldre színezi.

3. Rögzítsünk makrót, amely az egész munkalap formázását visszaállítja automatikusra (fehér háttérszín, fekete betűszín, Calibri betűtípus 11-es betűméret, nem dőlt, nem félkövér).

4. Rögzítsünk makrót, amely a kijelölt tartomány háttérszínét sárgára állítja. Próbáljuk ki különböző cellatartományok kijelölésével.

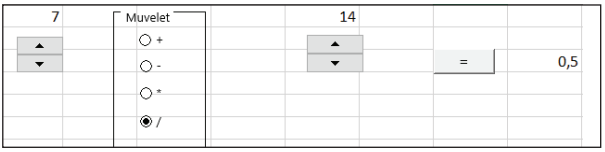
5. Rögzítsünk makrót, mely a kijelölt tartomány külső szegélyét kék színű vastag vonalra állítja, rácsvonalait szaggatott világos kékre.

6. Rögzítsünk makrót, amely az aktuális cellába sárga alapon piros, félkövér betűkkel beírja az aktuális dátumot.

7. Az eddig írt makrók mindegyikét egy gomb lenyomásával is el tudjuk indítani.

8. Tegyünk fel egy jelölőnégyzetet a munkalapra. Ha a jelölőnégyzet ki van jelölve, akkor az A2-es cellába a mai dátum kerüljön, különben az A3-as cellában jelenjen meg, hogy a hét hányadik napja van ma.

9. Készítsünk számológépet, amely a 4 alapműveletet tudja két szám között! A műveletet választógombokkal lehessen kiválasztani, a számokat meg léptetőnyilakkal.



7.3.1. ábra. *Példa számológép készítésére*

10. Készítsünk másodfokú függvényt ábrázoló diagramot! Az x értékek változzanak −10 és +10 értékek között, kéttizedenként! Az fx=a*(x^2)+b*x+c függvény a, b és c paramétereit görgetősávval lehessen megadni! A diagramot formázzuk úgy, hogy jól követhető legyen!

11. Készítsünk szinusz- vagy koszinuszfüggvényt ábrázoló diagramot! Az x értéke 0-tól 2Pi-ig változzon kéttizedenként. Hogy melyik függvény legyen (szinusz vagy koszinusz), legördülő listából válasszuk ki. A függvényeket tudjuk csúsztatni az x tengely mentén: a csúsztatást görgetősávval oldjuk meg!

8. VBA

A VBA (Visual Basic for Application) a Visual Basic variánsa, az Office programcsomag makrónyelve, amely csak bizonyos Microsoft-alkalmazásokon (Office), valamint néhány más alkalmazáson (pl. AutoCAD) belül futtatható programok írására szolgál, értelemszerűen nagyban igazodva az adott gazdaalkalmazás céljához és lehetőségeihez. A makróprogramozás betölti a feladatautomatizálás és a testreszabás szerepét.

A VBA egy objektumokat kezelő eljárásorientált programozási nyelv.

A VBA a számítások elvégzéséhez a központi memóriát használja. A programok minél kevesebb hibával való működése érdekében ezeket a memóriaterületeket előre lefoglaljuk (deklarálás) a program számára.

A programok utasításokból épülnek fel. Az utasítás egy elemi műveletre való felszólítás. Egy programblokk utasítások sorozata. Az első és az utolsó utasítás ebben az esetben azonosítja a blokk kezdetét és végét, valamint megfogalmazza azt, hogy milyen feltételek teljesülése esetén kell a blokk belsejében levő utasításokat végrehajtani.

Az Excel felől tekintve a makró egy, a felhasználó által végrehajtott műveletsorozat, amit a későbbi megismétlés céljára eltárolunk. Informatikai szempontból az Excel makrói tulajdonképpen programok, amelyek az őket felépítő utasítások sorrendjében hajtódnak végre.

Mivel a VBA-t az Excelen keresztül érjük el, ezért a táblázatok celláit fel tudjuk használni a feldolgozandó adatok bevitelére, és a kiszámított adatokat könnyedén el tudjuk helyezni valamelyik táblázati mezőbe.

Hivatkozás egy cellára:

```
cells(sorszám, oszlopszám)
```

Fejlesztőkörnyezet

Hogy a programozónak csak a problémára kelljen koncentrálnia, a programgyártó cégek egyre több szolgáltatást nyújtó integrált fejlesztőrendszert biztosítanak.

Integrált fejlesztőrendszer: erős szövegszerkesztő, mögötte a programíráshoz, fordításhoz, teszteléshez, futtatáshoz, programmentéshez, dokumentáláshoz szükséges programok – ezek a programok integráltan működnek együtt, a programozó minimális beavatkozása mellett.

Az automatikusan működő fordítóprogramnak (compiler) az alapja az ember számára olvasható programleírás, ami a VBA szabályai szerint sorokba írt utasítássorozat.

A fordítóprogram feladata, hogy az „olvasható és értelmes” nyelven írt forrásprogramot a számítógép számára „érthető” kettes számrendszerű kódsorozattá alakítsa át. Ez a fordítóprogram automatikusan, a programozó közreműködése nélkül végzi a feladatát. A VB fordítója „nem natív” (a gép által közvetlenül végrehajtható) utasításokká kódolja a forrásnyelvű szöveget, hanem egy közbenső nyelvre (p-kódra). A program végrehajtásakor a VB futtató magja ezt a p-nyelvű programot interpretálja. Ennek az interpreternek a feladata, hogy a p-nyelvű utasításokat natív-kódú utasításokká fordítsa, majd azokat a processzorral végrehajtassa.

A programozás közben elkövetett szintaktikai hibákra a szerkesztő rögtön felhívja a figyelmet, illetve az automatikusan felajánlott választási lehetőségek ki is zárják ezen hibák elkövetésének a lehetőségét.

A Visual Basic fejlesztőkörnyezet több módon is segíti a programírót a hibátlan kód elkészítésében és a hibás működés okának felderítésében:

- Helyzetérzékeny súgó: a VBA szerkesztője felismeri az utasítások kulcsszavát, és az F1-et használva segítséget nyújt.
- Futtatás: a Visual Basic modulban a Run menü → Run, vagy F5 billentyű lenyomásával, vagy futtatás nyomógombbal, vagy a makró futtatásával.
- Lépésenkénti programvégrehajtás: F8.
- Töréspontok elhelyezése – a lépésenkénti programvégrehajtás gyorsított üzemmódja: ráállunk a kívánt sorra, és lenyomjuk az F9 funkcióbillentyűt, vagy egérrel kattintunk a szerkesztőpanelben a sor előtt a szürke lécen.
- Figyelés ablak: a vizsgálóablak egy újabb lehetőség a hibák kiszűrésére, az ellenőrzésre (View → Watch windowval nyitjuk meg). A figyelni kívánt változókat, kifejezéseket a Debug → Add Watchcsal tehetjük a figyelőablakba vagy a változó kijelölésével és odahúzásával.

8.1. Bevezetés a programozásba

A program a számítógép számára értelmezhető utasítások pontosan, sorrendben végrehajtandó sorozata, melyek adott feladat megoldására szolgálnak. Elképzelhetjük úgy a programot, mintha egy receptet olvasnánk, amelyben le vannak írva a hozzávalók, illetve hogy sorjában mit kell tennünk az elkészítéshez.

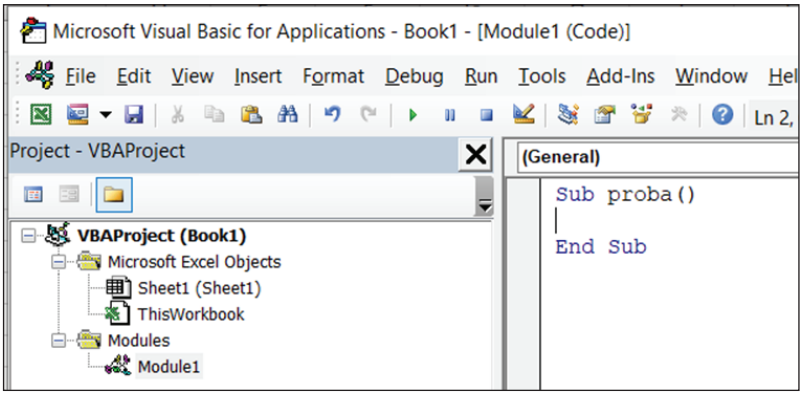
A program célja: egy felhasználó számára hasznos feladat elvégzése.

A végrehajtandó feladatok (alkalmazások) lehetnek egyszerűbbek, de sok számolnivalót tartalmazóak, vagy ismétlőek, amikor ugyanazt a rutint kell sokszor elvégezni. Akkor érdemes programot írni, ha sok számolnivaló van, illetve ha ugyanazt a(z esetleg rövid és egyszerű) számítást akarjuk sokszor elvégeztetni, vagy sorozatos kiíratás a feladat.

Egy új programnyelvet mindig sokkal könnyebb megtanulni, ha az ember ismer már egy másikat.

Egy programnyelv hatékony tanulása csak gyakorlással érhető el!

VBA **kód egyszerű létrehozása** makróírással: Fejlesztőeszközök (Developer) lap a menüszalagon → Makrók (Macros) → Makró neve (Macro name) – adunk egy új nevet a makrónknak → Létrehozás (Create) → beírjuk a kódot.



8.1.1. ábra. *A Visual Basic fejlesztő felülete*

Megjegyzések, kommentárok a kódban: ‘ (aposztróf) után – ezeket a fordító figyelmen kívül hagyja.

8.1.1. Változók

A program az adatokat változókban tárolja. A változó egy azonosítóval (névvel) ellátott memóriaterület, ahol a változó típusának megfelelő értéket (pl. adatot, részeredményt) tárolhatunk. A programok futása során sokszor előfordul, hogy bizonyos értékeket el kell tárolnunk, hogy később is fel tudjuk azokat használni. Erre valók a változók. Egy változóban tárolt érték a program végrehajtása során megváltozhat (innen van az elnevezés), ekkor a változó új értéke felülírja a régit.

A változók azonosítói (változónév) hivatkozások, melynek segítségével el tudjuk érni a megfelelő memóriarekeszeket.

A változónév csak betűket, számjegyeket és aláhúzásjelet tartalmazhat, de csak betűvel kezdődhet; célszerű olyan nevet adni a változónak, ami utal a tartalmára; hosszuk maximum 255 karakter lehet. A VBA nem tesz különbséget kis- és nagybetűk között. A változó neve nem lehet foglalt szó (pl. Sub, Dim, If, Case, For stb.).

Mivel az adatokat többféleképpen kódolhatjuk, ezért a változóknak is többféle típusuk van. Akkor van egyértelműen definiálva egy változó, ha a neve mellett az adattípusa is meg van adva. Általában a változók használatát megelőzi a változók deklarálása, mellyel megadjuk a változó típusát is.

Változók deklarálása

A VBA az adatokat a RAM-ban tárolja. A RAM az adatok tárolására bináris értékeket, nullák és egyesek sorozatát használja. Az adatok típusa adja meg, hogy milyen fajta adat használható, milyen műveletek végezhetők vele, illetve hogyan tárolódik a memóriában.

Alapvető adattípusok Visual Basicben:

- string($) – szöveges adattípus
- byte – egész szám (0-tól 255-ig) – memóriaigénye 1 byte
- integer(%) – egész szám (–32768-tól 32767-ig) – 2 byte-on
- long(&) – hosszú egész – 4 byte-on
- single(!) – valós szám – 4 byte-on
- double (#) – valós szám – 8 byte-on
- date – dátum – 8 byte-on
- boolean – igaz/hamis
- variant – általános – 16 byte-on

A legtöbb programozási nyelvben kötelező a változók deklarálása, de a Visual Basic megengedi a változók deklaráció nélküli használatát is (implicit megadási mód). Implicit megadás esetén a használatba vételt nem előzi meg deklaráció, pontosabban maga a használatbavétel egyben deklaráció is. Ebben az esetben *variant* típussal jönnek létre első használatkor. Viszont amellett, hogy kényelmes egy változót deklarálás nélkül használni, könnyen lehetséges hibaforrások (pl. egy változó azonosítójának elgépeléséből adódó hiba csak a program végrehajtása esetén derül ki). Ha a változót definiáltuk valamilyen típusra (explicit megadási mód), akkor a megadott típusnak megfelelő műveleteket hajthatunk végre vele. A programozási környezet csak akkor figyeli, hogy explicit módon minden változót deklaráltunk-e, ha a modul első utasítása az `Option Explicit`.

Deklarálás:

```
Dim változónév [As típus]
```

Példák:

```
Dim valos!, egesz As Integer

Dim nev As String, x
```

Ezekben a példákban a *valos* egy Single típusú változó, *egész* egy Integer típusú, *nev* egy String, *x* pedig egy Variant típusú változó.

8.1.2. Műveletek

- **Aritmetikai műveletek**: + (összeadás), – (kivonás), * (szorzás), / (osztás), \ (egész osztás hányadosa), ^ (hatványozás), MOD (egész osztás maradéka)
  Példák:
  - x = 1 + 2 → 3
  - x = 4 – 2 →2
  - x = 2 * 5 →10
  - x = 6 / 4 →1,5
  - x = 6 \ 4 →1
  - x = 2 ^ 3 →8
  - x = 6 mod 4 →2

- **Logikai műveletek**: And (és), Or (vagy), Not (tagadás), Xor (kizáró vagy: az eredmény akkor igaz, ha a feltételek között van olyan, amelyik teljesül, és van olyan, amelyik nem), Eqv (azonos: az eredmény akkor igaz, ha az összes feltétel igaz vagy hamis)
- **Relációs műveletek**: =, <> (nem egyenlő), <, <=, >, >=
- **Szöveges műveletek**: összefűzés: & vagy +
  Példa: `s="al"+"ma"` → `s="alma"`, vagy `s="al"&"ma"` → `s="alma"`

8.1.3. Utasítások

Értékadó utasítás: `változónév = kifejezés` (a bal oldalon levő változó felveszi a jobb oldalon levő kifejezés értékét)

Példák:

`x = 3` → x „legyen egyenlő” 3-mal
`y = x+2` → y „legyen egyenlő” az x értéke 2-vel növelve
`cells(1, 2) = x` → a B1-es cellába az x változó értéke kerül
`z = cells(2, 1)` → a z változó értéke az A2-es cella tartalma lesz

8.1.4. Adatbevitel (beolvasás)

A változók beolvasásának egyik lehetősége az **Üzenőablakok használata.**

A program futása során kapcsolatot tarthatunk a felhasználóval. Kérhetünk tőle adatokat az InputBox (a VBA adatbeviteli objektuma) segítségével, és üzenhetünk neki a MsgBox (a VBA adatkiviteli objektuma) segítségével.

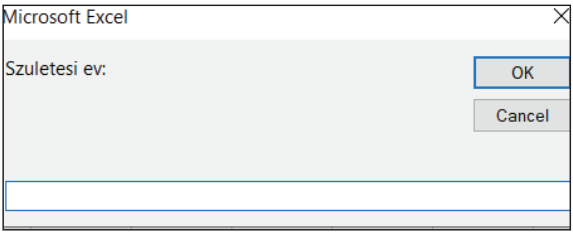
Az InputBox szintaxisa:

```
változónév = InputBox(szöveg[, cím] [, alapértelmezett érték]
[, x_pozíció] [, y_pozíció])
```

Az InputBox egy kis ablak, amely megjelenik a képernyőn. Az ablakban lévő beviteli mezőbe írhat a felhasználó. A program a változóban tárolja, amit a felhasználó beírt. A zárójelben felsorolt paraméterek közül a szöveget kötelező megadni.

- a *szöveg* egy tájékoztató üzenet arról, hogy mit vár a felhasználótól, ez fog megjelenni az ablakban a beviteli mező fölött;
- a *cím* lesz az ablak fejlécében;
- az *alapértelmezett érték* a beviteli mezőben lesz, kijelölve, felülírható;
- az *x_pozíció*-ban lehet megadni a képernyő bal oldala és az üzenőablak bal oldala közti távolságot twipben (1 cm = 567 twip);
- az *y_pozíció*-ban lehet megadni a képernyő teteje és az üzenőablak teteje közti távolságot twip-ben; Alapértelmezésben az üzenőablak a képernyő közepén jelenik meg.

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:



8.1.4.1. ábra. *Beolvasás*

Példák:

```
x = InputBox(“Kérek egy egész számot”)

nev = InputBox(“Kérek egy vezetéknevet”, “Beolvasás”)

nem = InputBox(“Kérem a nemét”, ,”nő”)
```

8.1.5. Adatkivitel (kiíratás)

A program során az üzeneteinket külön kis ablakban jeleníthetjük meg. Különösen hasznos ez figyelmeztetésekkor, eredmény kiírásakor.

A MsgBox szintaxisa:

```
MsgBox szöveg [, gombok] [,cím]
```

A MsgBox egy kis ablak, amely egy üzenettel jelenik meg a képernyőn. A felsorolt paraméterek közül a *szöveg*et kötelező megadni. Ez fog megjelenni az ablakban. A *cím* lesz az ablak fejlécében. A *gombok*nál 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!) Gombok: vbOKCancel, vbYesNoCancel, vbYesNo, vbRetryCancel, vbCritical, vbQuestion, vbExclamation, vbInformation.

Kiíratáskor szükségünk lehet arra, hogy szó szerinti szövegeket és változó tartalmakat együtt jelenítsünk meg. A szó szerint megjelenítendő szöveget idézőjelbe kell tenni, a változó nevét idézőjel nélkül kell leírni. Ha össze akarjuk fűzni őket, akkor az & jelet kell használnunk.

Példa: Ha a *nev* nevű változóban tároljuk egy személy nevét (legyen az értéke mondjuk „Éva") és a *kor* változóban az életkorát (legyen ez mondjuk 18), akkor annak kiíratása, hogy „Éva 18 éves", a következőképpen tehető meg:

```
MsgBox nev & " " & kor & " éves"
```

8.1.6. Kitűzött feladatok

1. Készítsünk makrót, amely Inputboxban bekéri a felhasználó nevét, aztán Msgboxban köszönti a felhasználót a saját nevén!

2. Készítsünk makrót, melyben az A1-es cellába beírja az Inputboxban beolvasott felhasználó életkorát (alapértékként 19 évet adjunk meg).

3. Készítsünk makrót, mely egy Msgboxban a következő üzenetet jeleníti meg: „Gratulálok! Szeretne új feladványt?" A válaszlehetőségek „Yes" vagy „No" legyenek.

4. Készítsünk makrókat, melyek a vbCritical, vbQuestion, vbExclamation, vbInformation típusú Msgboxban adnak tetszőleges üzeneteket.

5. Írjunk makrót, amelyben beolvasunk két egész számot, és a program kiírja az összegüket, különbségüket, szorzatukat, hányadost, osztási maradékot.

6. Írjunk makrót, amely egy Inputbox segítségével beolvasott szám utolsó számjegyét adja vissza egy Msgboxban.

7. Készítsünk makrót, amely egy Inputboxban bekért számból gyököt von. Az eredmény az A1-es cellában jelenjen meg.

8. Írjunk programot, mely kiszámítja, hogy egy egyenletes sebességgel egyenes vonalban haladó test mennyi idő alatt tesz meg egy adott útszakaszt (bemeneti adatok a sebesség és a hossz).

9. Írjunk makrót, amely beolvassa az autóba tankolt benzin mennyiségét, majd kiírja, hogy várhatóan mekkora utat lehet ezzel a tankolással megtenni, ha az autó átlagos fogyasztása 6,4 liter/100 km.

10. Írjunk programot, mely bekéri egy téglalap oldalainak hosszát, és kiszámítja a téglalap területét és kerületét.

11. Írjunk makrót, mely az x és y valós típusú változók értékét felcseréli. Írassuk az A1 és A2 cellákba a változók csere előtti, a B1 és B2 cellákba a változók csere utáni tartalmát.

12. Jancsi bácsi fizetést kap (beolvassuk a kapott pénzösszeget). A kifizetéshez 500, 200, 100, 50, 10 és 5 lejes bankjegyek állnak rendelkezésre – a maradékot Jancsi bácsi nem kapja meg. Feltételezzük, hogy minden címletből van elég, és a lehető legkevesebb számú bankjegy kerül kiosztásra. Milyen bankjegyből hányat kap Jancsi bácsi, és mennyi pénzt hagy ott?

8.2. Programozási struktúrák (1)

8.2.1. Szekvencia

A program utasításai egymás után, az utasítások megadási sorrendjében hajtódnak végre.

Egy tevékenységsorozat (program) hívása során nem csak utasítások egymás utáni végrehajtását szeretnénk, hanem szükség lehet ismétlésekre, vagy valamilyen feltétel szerinti elágazásra. Ilyenkor programjainkban az utasítások nem sorrendben egymás után hajtódnak végre (szekvencia), hanem használnunk kell az elágazásokat és a ciklusokat.

8.2.2. Elágazások

Feltételhez kötjük, hogy melyik utasítás (utasítások) hajtódjon (hajtódjanak) végre.

8.2.2.1. Az IF utasítás

Szintaxisai Visual Basicben:

```
1. If feltétel Then utasítás [Else utasítás]
```

```
2. If feltétel Then
      utasítások, amik akkor futnak le, ha a feltétel igaz
   [Else
      utasítások, amik akkor futnak le, ha a feltétel hamis]
   End If

3. If feltétel Then
      Utasítás(ok)
   [ElseIf feltétel2 then
      Utasítás(ok)
       …]
   [Else
      Utasítások]
   End If
```

Végrehajtás: mindhárom esetben, ha a feltétel teljesül (vagyis igaz), akkor a **Then** után következő utasítást (vagy utasításokat) hajtja végre, majd az **End If** után következő utasításra tér (az első esetben a következő sorra).

Az **Else** ág elhagyható, ilyenkor

1. ha a feltétel nem teljesül, akkor a következő sorban folytatódik a program,
2. ha a feltétel nem teljesül, akkor az **End If** utáni sorban folytatódik a program,
3. ha egyik feltétel sem teljesül, akkor az **End If** utáni sorban folytatódik a program.

- Az első esetben az egész utasítás egy sorba kerül (ekkor nem kell **End If** az utasítás végére), míg a másik két esetben több sorba (ahol az egyes ágakban lévő utasítások beljebb tagolása nem kötelező, de javítja az áttekinthetőséget).
- A harmadik esetben, ha a feltétel nem teljesül, akkor a következő feltételt nézi meg, hogy teljesül-e (és így tovább). Ha valamelyik feltétel teljesül, akkor az ott levő utasítás(oka)t hajtja végre és az **End If** után folytatódik.

Példa: Olvassunk be egy egész számot, majd írjuk ki szövegesen, hogy a beolvasott szám negatív, pozitív vagy nulla.

```
Sub elojel()
Dim x As Integer
x = InputBox("Kerek egy egesz szamot")
If x > 0 Then
   MsgBox "Pozitiv"
 ElseIf x < 0 Then
   MsgBox "Negativ"
 Else
   MsgBox "Nulla"
End If
End Sub
```

8.2.2.2. A Select Case utasítás

A Select Case utasítás a többágú elágaztatás egy eszköze, melynek lényege, hogy egy adott kifejezés vagy változó értékeinek megfelelően választjuk ki a programútvonalat. Főleg akkor használjuk, ha az elágazásokat kettőnél több feltétel szerint akarjuk végrehajtani. Ilyenkor az elágazási feltételeket a **Case** kulcsszó után kell megadni.

Szintaxisa Visual Basicben:

```
Select Case változó vagy kifejezés
Case kifejezéslista1
   Utasítás(ok)1
[Case kifejezéslista2]
   [Utasítás(ok)2]
…
[Case Else
   [utasítás(ok)]]
End Select
```

A kifejezéslista lehet:

- *kifejezés* (egy érték) (pl. 7 – a változó értéke megegyezik-e 7-tel?)
- *kifejezés1, kifejezés2, …* (több érték) (pl. 7, 8, 9 – a vessző itt vagy-ot jelent: a változó értéke megegyezik-e a felsoroltak közül valamelyikkel – 7 vagy 8 vagy 9?)
- *kifejezés1 to kifejezés2* (értéktartomány) (pl. 1 to 10 – a változó értéke benne van-e az [1, 10] intervallumban?)
- *Is = (<>, <, <=, >, >=) kifejezés* (pl. Is <0 – a változó negatív-e?)

Ha az adott változó aktuális értéke megegyezik valamelyik **Case**-t követő értékkel, akkor az ott levő utasítás(oka)t hajtja végre, majd az **End Select** utáni utasításra ugrik. A **Case Else** ág utasításai akkor hajtódnak végre, ha egyik megelőző ágba sem lépett be, vagyis ha az egymás után felkínált feltételek egyike sem teljesült.

Példa 1: Olvassunk be egy 1 és 12 közötti egész számot, mely megfelel egy hónap sorszámának. A program írja ki a beolvasott hónapnak megfelelő évszakot:

```
Sub evszak()
Dim ho As Integer
ho = InputBox("Kerem egy honap sorszamat")
Select Case ho
  Case 12, 1, 2
    MsgBox "Tel"
```

```
  Case 3 To 5
    MsgBox "Tavasz"
  Case 6 To 8
    MsgBox "Nyar"
  Case 9 To 11
    MsgBox "Osz"
  Case Else
    MsgBox "Hibas honap-sorszam"
End Select
End Sub
```

Példa 2: Egy bizonyos termékre a kedvezmény a vásárolt mennyiségtől függ:
10% – ha a vásárolt mennyiség 0 és 25 között van
15% – 26 és 50 közötti darabszám vásárlása esetén
20% – 51 és 75 közötti darabszám esetén
25% – 75 darab feletti vásárlás esetén

Írjunk programot, mely a vásárolt mennyiségtől függően kiírja a 25 lejes alapárú termékért fizetendő összeget.

```
Sub vasarlas()
  Dim menny As Long
  Dim kedv As Double
  menny = InputBox("Kerem a vasarolt mennyiseget")
  Select Case menny
    Case 0 To 25
      kedv = 0.1
    Case 26 To 50
      kedv = 0.15
    Case 51 To 75
      kedv = 0.2
    Case Is > 75
      kedv = 0.25
  End Select
  MsgBox "Fizetendo osszeg: " & 25 * (1 - kedv) * menny
End Sub
```

8.2.3. Kitűzött feladatok

1. Készítsünk makrót, amely deklarál egy egész típusú változót, InputBoxban beolvassa a felhasználó életkorát, majd az A1-es cellába beírja a beolvasott értéket.

Javítsuk át a makrót úgy, hogy csak akkor írja be az A1-es cellába a beolvasott értéket, ha az 0 és 100 között van, különben egy MsgBoxban jelenjen meg valamilyen üzenet.

2. Írjunk makrót, amely az A1-es cella tartalmáról eldönti, hogy páros szám-e vagy sem. A választ egy MsgBoxba írassuk ki.

3. Készítsünk makrót, amely egy InputBoxban bekért számból gyököt von, ha a szám pozitív. Az eredmény az A1-es cellába kerüljön, különben írassunk ki egy hibaüzenetet MsgBoxban.

4. Írjunk makrót, amely egy InputBoxban beolvasott számról eldönti, hogy a [-10,10] intervallumba esik-e.

5. Írjunk makrót, amely beolvas két számot, és a bekért számokat oly módon írja ki, hogy közéjük teszi a nagyobb-, kisebb- vagy az egyenlőségjelet!

6. Ellenőrizzük, hogy milyen feltételek esetén oldható meg egy másodfokú egyenlet a valós számok körében. Írjuk ki, hány valós gyök van, illetve a megoldásokat, ha van.

7. Írjunk programot, amely 3 beolvasott nevet ábécésorrendben ír ki! A nevek éppúgy összehasonlíthatók, akár a számok: pl. „Aladár”<”Bandi”, itt az ábécésorrend számít. Az a kisebb, amelyik az ábécében előbb van.

8. Olvassunk be két egész számot, majd MsgBox segítségével írassuk ki a nagyobbiknak a kisebbikkel való osztási maradékát.

9. Olvassunk be egy 1 és 7 közötti egész számot, mely megfelel a hét egy napjának. Írassuk be az A1-es cellába a számnak megfelelő napot, vagy ha a szám nincs 1 és 7 között, akkor MsgBoxban jelenjen meg egy hibaüzenet.

10. Olvassuk be InputBoxban a felhasználó nemét: „f” vagy „F”, ha férfi, és „n” vagy „N”, ha nő. Majd ennek megfelelően írassuk ki MsgBoxban: „Férfi” vagy „Nő” vagy „Hibás adat”.

11. Készítsünk makrót, mely Inputboxban bekéri a személy életkorát (0–120 év). A beolvasott életkortól függően a következő üzenetek jelenjenek meg:

- csecsemő – ha a beolvasott életkor 0 és 1 között van,
- kisgyerek – 1 és 6 év között,
- nagygyerek – 6-tól 12 évig,
- kamasz – 12-től 16 évig,
- fiatal – 16-tól 25 évig,
- felnőtt – 25-től 60 évig,
- idős – 60 és 120 között,
- hibás adat az összes többi esetben.

12. Olvassunk be egy –100 és 100 közötti számot. Írassuk ki, hogy a beolvasott szám POZITÍV, NEGATÍV vagy NULLA, illetve ha a beolvasott szám nincs az adott intervallumban.

13. Olvassuk be egy hónap és egy nap sorszámát. Írassuk ki MsgBoxban, hogy lehetséges-e a megadott dátum (pl. 2. hónap 30-a nem egy lehetséges dátum).

14. Írjunk programot, amely eldönti, hogy 3 beolvasott szám lehet-e egy derékszögű háromszög oldalhosszai!

15. Írjunk programot, amely eldönti, hogy az X és Y beolvasott számpárhoz tartozó pont a koordináta-rendszer melyik negyedében van (illetve valamelyik tengelyen)!

16. Írjunk programot, amely eldönti, hogy az X és Y beolvasott számpárhoz tartozó pont benne van-e az origó középpontú, R sugarú körben! (A pont benne van a körben, ha koordinátáira érvényes, hogy: x^2+y^2<R^2.)

8.3. VBA-alapfogalmak

Mielőtt folytatnánk a programozási struktúrákat, ismerjük meg az Excel objektumait.

8.3.1. Az Excel objektumai

– Excel-elemek: munkafüzet, munkalap, oszlop, sor, tartomány, cella, diagram stb.,
– az Excel objektumai rendelkeznek tulajdonságokkal és metódusokkal,
– a műveletek elvégzése előtt mindig azonosítani kell az objektumot,
– az azonos objektumok együtt is kezelhetők (csoport),
– példák: Application, Workbook, Worksheet, Range.

Használat*:*
– objektum.tulajdonság =…
– objektum.metódus
– események, eseménykezelő eljárások

Metódus:
– az objektumhoz vagy csoporthoz rendelt műveletek, tevékenységek
pl. egy munkafüzet lezárása:

```
Workbooks("Munkafuzet.xls").Close
```

Tulajdonság:
– az objektum attribútumai: szín, név, méret, érték stb.
pl. egy cellában levő képlet kiíratása:

```
MsgBox Worksheets("Lap1").Range("C5").Formula
```

8.3.1.1. Néhány Excel objektum

• Application: MS Excel alkalmazás, programszintű beállítások és elemek
pl. `Application.Round(…)`: Excel kerekítés függvény elérése (függvénynév angolul!)

- WorkBooks: munkafüzetek objektum
  - metódusok: Open, Close, Add, Select
    pl. a Feladat.xlsx munkafüzet bezárása:
    `WorkBooks("Feladat.xlsx").Close`
- WorkSheets: munkalapok objektum
  - metódusok: Select, Copy
    pl. Az Adatok nevű munkalap aktívvá tétele:
    `WorkSheets("Adatok").Select`

Cellaazonosítások
- Cella: `Range("A1")` vagy `Cells(sorszám,oszlopszám)`
- Tartomány: `Range(Cells(4,3),Cells(6,7))` vagy `Range("D7:F10")`
- Oszlop: `Columns("A")` vagy `Columns(1)` vagy `Range("A:A")`
- Egymás melletti oszlopok: `Columns("A:D")` vagy `Range(Columns(1),Columns(4))`
- Nem egymás melletti oszlopok: `Range ("A:A,D:D,H:H")`
- Minden oszlop: `Columns`
- Sor: `Rows(1)` vagy `Range("1:1")`
- Egymás melletti sorok: `Range(Rows(1),Rows(4))` vagy `Range("1:4")` vagy `Rows("1:4")`
- Nem egymás melletti sorok: `Range("1:1,4:4,7:7")`
- Összes sor: `Rows`
- A munkalap összes cellája: `Cells`
- Meghatározott munkalap cellái: `Sheets1.Cells`

Aktív objektumok:
- `ActiveWorkBook` – az aktuális munkafüzet
- `ActiveSheet` – az aktuális munkalap
- `ActiveCell` – az aktuális cella
- `ActiveChart` – az aktuális diagram
- `Selection` – kijelölt tartomány

8.3.1.2. Excel objektumok tulajdonságai

Tartományok, ill. cellák esetében:
- `Value`: a tárolt érték (szám, szöveg, dátum stb.)

Példa: `Sheet1.Range("A1").Value = 2021`
- `Formula`: Excel képlet (szöveges)
- `Interior`: a háttér (Color, ColorIndex, Pattern)
  - `Pattern: a mintázat (pl. xlSolid, xlPatternCrissCross`)

- `Color`: a cella háttérszíne szövegesen megadva. Az angol színelnevezéseket lehet használni, csak elé kell írni, hogy vb (Black, White, Red, Green, Blue, Yellow, Magenta, Cyan) vagy RGB(0..255, 0..255, 0..255)
  Példák: `Cells(1,1).Interior.Color = vbRed` vagy
  `Cells(1,1).Interior.Color = RGB(255, 0, 0)`
- `ColorIndex` (a színskála sorszáma, `xlAuto, xlNone`):
  `Interior.ColorIndex`: a cella háttérszíne számmal megadva. A Visual Basic 0-tól 56-ig számozza a színeket.
  Példa: Az A1 cella háttérszíne a következőképpen is pirosra állítható (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 (`Name, Size, Color, ColorIndex, Bold, Italic, Underline, SubScript, SuperScript`)
  Példák:
  – betűtípus Arialra 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`
  – aláhúzott: `Cells(1,1).Font.UnderLine = True`
- Cella betűstílusának állítása egyben:

```
With Cells(1,1).Font
    .Name =…
    .Color = …
End with
```

- `Border`: a szegély (`ColorIndex`, `LineStyle`, `Weight`)
- Igazítások:
  - vízszintes igazítás: `Cells(1,1).HorizontalAlignment =xlLeft` (balra) vagy `xlCenter` (középre) vagy `xlRight` (jobbra)
  - függőleges igazítás: `Cells(1,1).VerticalAlignment =xlTop` (fent) vagy `xlCenter` (középre) vagy `xlBottom` (lent)
- A kijelölt rész sormagassága: `Selection.RowHeight`
- A kijelölt rész oszlopszélessége: `Selection.ColumnWidth`

8.3.1.3. Néhány metódus

- Kijelölés: `Select`
  – aktuális cella megváltoztatása: `Range(cellaazonosító).Select`
  – munkalapváltás: `Munkalapnév.Select`
  – a kijelölt objektum: `Selection`

- Beszúrás: `Insert`
  - új sor beszúrása: `Rows(7).Insert`
  - új oszlop beszúrása: `Columns(3).Insert`
- Törlés: `Delete`
  - sor törlése: `Rows(7).Delete`
  - oszlop törlése: `Columns(4).Delete`
- Tartalom törlése: `ClearContents`
- Formátum törlése: `ClearFormats`

8.3.2. Kitűzött feladatok

1. Írassuk ki az aktuális cella tartalmát egy előugró ablakba.
2. Ha az aktív cella üres, írjunk bele 1-et.
3. Ha az aktív cellában 10-nél nagyobb szám van, töröljük ki.
4. Ha az aktív cellában 10-nél kisebb szám van, növeljük 1-gyel, különben töröljük a tartalmát.
5. Ha az aktív cella a keresztnevünket tartalmazza, írjuk felül a cella tartalmát a teljes nevünkkel, különben töröljük a cella tartalmát.
6. Írassuk ki üzenetben, hogy az aktív cella tartalma nagyobb, kisebb vagy egyenlő, mint a fölötte lévő ActiveCell.Offset (hozzá képest a sor, hozzá képest az oszlop)).
7. Írassuk ki üzenetben, hogy az aktív cella tartalma kétszerese-e, mint a balra mellette lévő.
8. Írjunk makrót, mely a kijelölt területet kitölti egy beolvasott szöveggel!
9. Írjunk makrót, amely a kijelölt területet kitölti az A1 cella tartalmával!
10. Írjunk programot, amely InputBoxban bekér egy háttérszínkódot. Ha a beolvasott szín nem 0 és 56 között van, írjon ki hibaüzenetet, különben színezze az összes cella hátterét a megadott kódú színre.
11. InputBoxban beolvasunk egy sor- és egy oszlopszámot. A beolvasott sor- és oszlopszám által megadott cellát, illetve a tőle balra fent és a tőle jobbra lent levő cellák hátterét színezzük ki egy beolvasott színre. Ha a beolvasott sor- vagy oszlopszám valamelyike 1-es, akkor adjunk hozzá egyet.
12. Írjunk makrót, amely egy Inputbox által bekért nagyságú négyzet alapú cellatartományt dupla vonallal bekeretez. A bekeretezett tartomány bal felső sarka az aktuális cella legyen (vagy olvassuk be a bal felső sarok koordinátáit).
13. Készítsünk makrót, mely az aktív tartomány hátterét az A1-es cellában levő számnak megfelelő színűre színezi.
14. Írjunk makrókat, melyek:
    i. A C1:H30 tartományt feltölti 1-esekkel.
    ii. Oszlopokat szúr be (a C, az F és a H elé).
    iii. Kitörli a C, F és H oszlopokat.

15. Készítsünk makrót, amely a kijelölt cellákat egybenyitja, a tartalmat vízszintesen és függőlegesen középre igazítja és 12 pontos betűméretűre állítja.
16. Írjunk makrót, amely egy előugró üzenetben kiírja az aktív munkalap nevét.
17. Írjunk makrót, amely beleírja az aktuális cellába a munkafüzet nevét.
18. Írjunk makrót, amely az aktuális cella tartalmát az alatta levő cellába másolja, az eredetiből pedig törli.
19. Írjunk makrót, amely kiírja egy előugró ablakba az aktuális cella 100-szorosát.
20. Írjunk makrót, amely az aktuális cella értékét a 100x-osára cseréli.
21. Írjunk makrót, amely az aktuális cella tartalmát megszorozza a mellette balra lévő cella tartalmával, és az eredményt a tőle jobbra levő cellába írja.
22. Ugyanaz, mint az előző feladat, csak most MsgBoxban jelenjen meg az eredmény.
23. Írjunk makrót, amely egy üzenetben figyelmeztet, hogy törölni fog, és ha a felhasználó OK gombot nyomott, akkor az aktuális cellából törli az értéket.
24. Írjunk makrót, amely a kijelölt területről törli az értékeket, majd a cellák hátterét pirosra állítja.

8.4. Programozási struktúrák (2)

8.4.1. Ciklusok

Utasítások ismételt végrehajtására használjuk, amikor ugyanazt az utasítássort (azonos vagy hasonló tevékenységeket) szeretnénk többször megismételtetni a programmal (például egy táblázat minden egyes oszlopán kell egy adott műveletsort végrehajtani).

A VBA-ban is többféle ciklustípus áll rendelkezésünkre:

- Számláló vagy iterációs ciklus
- Feltételes ciklusok:
  - előltesztelő ciklus
  - hátultesztelő ciklus

Ha tudjuk előre pontosan, hogy hányszor szeretnénk ugyanazt az utasítássort végrehajtani, akkor számláló (iterációs) ciklust használunk: FOR ciklus.

8.4.1.1. For ciklus

Szintaxisa Visual Basicben:

```
For ciklusváltozó = kezdőérték To végérték [Step lépésköz]
   Ciklusmag
Next
```

Végrehajtás:

- A Ciklusmag végrehajtási feltétele:
  - ciklusváltozó<= végérték, ha a lépésköz>=0 (vagy ha nincs megadva lépésköz)
  - ciklusváltozó>= végérték, ha a lépésköz<0
- Első lefutáskor a ciklusváltozó felveszi a kezdőértéket, aztán minden újabb lefutáskor a lépésközzel változik az értéke, amíg meg nem haladja a végértéket. Amikor a ciklusváltozó meghaladta a végértéket, akkor a Ciklusmag már nem hajtódik végre.
- A lépésköz megadása nem kötelező: ha nem adunk meg lépésközt, akkor annak értéke: +1.
- Lépésköznek megadható törtszám vagy negatív szám is (utóbbi esetben a végérték kisebb vagy egyenlő kell legyen, mint a kezdőérték).
- A ciklusváltozó bármilyen számváltozó lehet.
- A Ciklusmag tetszőleges utasítássorozat.

Példák:

1. Írassuk ki az 1 és 100 között levő négyzetszámokat:

```
Dim i As Integer
For i = 1 To 10
   MsgBox i^2
Next
```

2. Töltsük fel az A1:A10 tartományt tetszőleges számokkal. A B oszlopba írjuk az A oszlopbeli számok háromszorosát:

```
Dim i As Integer
For i = 1 To 10
  Cells(i, 2) = 3 * Cells(i, 1)
Next
```

3. Másoljuk át a következő tőzsdeindex-változásokat tartalmazó táblázatot az A1-es cellától kezdődően:

8.4.1.1.1. táblázat. *Tőzsdeindex-változások példa*

6%	-3%	-0,20%	0,80%	1,30%	3,20%	6%	7%	8%	6%

Egy adott tőzsdei kereskedési nap besorolása 1, 2, 3 aszerint, hogy a tőzsdeindex aznapi változásának abszolút értéke kisebb, mint 1,8%, vagy 1,8% és 5%

között van, avagy 5%-nál nagyobb. Írjunk makrót, amely egy adott napi hozamhoz hozzárendeli a besorolást és beírja az alatta levő cellába.

```
Sub tozsde()
Dim i As Integer
For i = 1 To 10
    Select Case Abs(Cells(1, i))
        Case Is < 0.02
            Cells(2, i) = 1
        Case Is < 0.05
            Cells(2, i) = 2
        Case Else
            Cells(2, i) = 3
    End Select
Next
End Sub
```

8.4.1.2. Feltételes ciklusok: A DO … LOOP típusú ciklusok

Akkor használjuk, amikor nem tudjuk előre, hogy hányszor kell végrehajtani egy adott utasítássorozatot, hanem egy feltételhez akarjuk kötni a ciklus végét.

Vannak elöltesztelős és hátultesztelős ciklusok: elöltesztelős ciklusnál a feltételt a ciklus elején adjuk meg, ilyenkor előbb ellenőrzi a feltételt, és attól függően hajtja végre vagy sem a ciklus magját; hátultesztelős ciklusnál egyszer mindenképpen végrehajtja a ciklusmagot, és csak utána ellenőrzi a feltételt.

Visual Basicben 4-féle feltételes ciklus létezik:

```
1.Do While feltétel
   utasítások
  Loop
```

- elöltesztelős ciklus – mindaddig ismételi az utasításokat, amíg a feltétel igaz
- előfordulhat, hogy már az első lefutáskor sem teljesül a feltétel, ilyenkor egyszer sem fut le a ciklus belsejében levő utasítássorozat

```
2. Do Until feltétel
      utasítások
   Loop
```

- elöltesztelős ciklus – mindaddig ismételi az utasításokat, amíg a feltétel hamis: amikor a feltétel igazzá válik, kilép a ciklusból

- előfordulhat, hogy már az első alkalommal teljesül a feltétel, ilyenkor egyszer sem fut le a ciklus belsejében levő utasítássorozat

```
3. Do
     utasítások
   Loop While feltétel
```

- mindaddig ismételi az utasításokat, amíg a feltétel igaz
- hátultesztelős ciklus – egyszer mindenképpen végrehajtja a ciklusban levő utasításokat, és csak utána ellenőrzi a feltételt

```
4. Do
     utasítások
   Loop Until feltétel
```

- mindaddig ismételi az utasításokat, amíg a feltétel hamis: amikor a feltétel igazzá válik, kilép a ciklusból
- hátultesztelős ciklus – egyszer mindenképpen végrehajtja a ciklusban levő utasításokat, és csak utána ellenőrzi a feltételt

Példák:

1. Olvassunk be egy egész számot, és írjuk be az A oszlop első üres cellájába az aktuális munkalapon:

```
Sub elso_urescella()
Dim x As Integer, i As Integer
x = InputBox("Kerek egy egesz szamot")
i = 1
Do While Cells(i, 1) <> ""
    i = i + 1
Loop
Cells(i, 1) = x
End Sub
```

2. Olvassunk be egy pozitív számot (a beolvasást mindaddig ismételjük, amíg a szám pozitív nem lesz), majd írassuk ki a szám négyzetgyökét.

```
Sub pozitiv_gyok()
Dim x As Integer
```

```
Do
    x = InputBox("Kerek egy pozitiv szamot")
Loop Until x > 0
MsgBox "A beolvasott szam negyzetgyoke: "
        & x ^ (1 / 2)
End Sub
```

8.4.2. Véletlen számok generálása

A véletlen számok generálására szolgáló függvény az `Rnd`. Ez a függvény egy véletlen valós számot ad eredményül a [0, 1) intervallumból. Ennek a függvénynek a transzformálásával lehet megadni, hogy milyen számokat szeretnénk generálni. Általánosan, ha [a,b] intervallumba eső egész számokat szeretnénk generálni:

```
(int(Rnd*(b-a+1))+a)
```

A `Randomize` utasítás a véletlenszám-generátor inicializálását végzi el. Ekkor a véletlen számok előállítására használt `Rnd` függvény futásonként eltérő véletlen számokat fog előállítani. A `Randomize` függvényt az `Rnd` függvény használata előtt kell hívni.

Példa:
Töltsük fel az A1:A20 tartományt 20 kockadobás eredményével (dobókockával 1 és 6 közötti számokat lehet dobni; a számokat véletlenszerűen generáljuk), majd számoljuk meg, hány „6-os dobás volt”:

```
Sub kockadobas()
Dim i As Integer, db As Integer
Randomize
db = 0
For i = 1 To 20
    Cells(i, 1) = Int(Rnd * 6) + 1
    If Cells(i, 1) = 6 Then db = db + 1
Next
MsgBox "6-os dobasok szama: " & db
End Sub
```

8.4.3. Kitűzött feladatok

1. Írjunk makrót, amely n darab “a” betűt ír az első sorba. Az ’n’ értékét InputBoxban olvassuk be.

2. Írjunk 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.

3. Írjunk makrót, amely 1-től egy InputBoxban bekért számig a C oszlopba kiírja a számok négyzeteit.

4. Írjunk makrót, amely a B oszlopba kiírja a 8-as szorzótáblát 1-től 30-ig!

5. Írjunk makrót, amely kiírja egy beolvasott egész szám osztóit az A oszlopba.

6. Írjunk makrót, amely az A1-es cellában megadott szám faktoriálisát számolja ki a C1-es cellába.

7. Írjunk makrót, amely InputBoxban beolvasott két szám közötti páratlan számok összegét számolja ki.

8. Írjunk makrót, amely összeszorozza az A1:A9 cellatartományban található pozitív számokat!

9. Olvassunk be egy egész számot, majd írassuk ki, hogy a beolvasott szám prímszám-e vagy sem.

10. Töltsük fel az A oszlop első 20 sorát véletlen egész számokkal. Számoljuk meg, hogy hány 3-mal osztható szám van ott.

11. Írjunk 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.

12. Írjunk makrót, amely az 1. sorban véletlen darabszámú számú cellát pirosra színez.

13. Írjunk makrót, amely –6 és +7 közötti véletlen egész számokkal feltölti az A oszlopot a 2. sortól a 18. sorig, majd a páros számok betűszínét kékre, a páratlanok háttérszínét pedig sárgára változtatja.

14. Írjunk makrót, amely a B3:B11 cellatartományt véletlenszerűen színezi ki.

15. Írjunk makrót, amely InputBoxban bekért darabszámú számot generál, és ezeket beírja az első sorba. Majd kiszámolja a generált számok szorzatát, és az eredményt MsgBoxban írja ki.

16. Írjunk makrót, amely –20 és +20 közötti véletlen számokkal feltölti az E5:K20 cellatartományt, majd a negatív számok hátterét sárgára, a többiekét kékre színezi.

17. Olvassunk be egy 1 és 10 közötti számot (a beolvasást mindaddig ismételjük, amíg a szám a megfelelő intervallumban nem lesz), majd írassuk ki a szám negyedik hatványát.

18. Írjunk makrót, amely addig megy az első sorban jobbra és színezi pirosra a cellák hátterét, amíg üres vagy nullaértékű cellát nem talál.

19. Írjunk olyan VBA programot, amelyben a számítógép kigondol egy 1 és 100 közé eső egész számot (véletlenszerűen generáljuk), majd felszólítja a felhasználót,

hogy találja ki azt (MsgBox utasítás). A felhasználó tippjére (az InputBoxban beolvasott értékre) közli, hogy az nagyobb vagy kisebb az általa gondoltnál, illetve a helyes tipp esetén gratulál és felajánlja, hogy új játékot kezdjenek.

20. Írjunk makrót, amely egy oszlop elemein végigmegy (Inputboxban olvassuk be az oszlop sorszámát), és ha egymás utáni ismétlődést talál, törli az ismétlődő elemet tartalmazó cellát. Figyeljünk oda, hogy a makró akkor is jól működjön, ha egymás után többször ismétlődik egy elem!

21. Írjunk be számokat az A és B oszlopba az első sortól kezdődően, majd írjunk egy makrót, mely összehasonlítja az A oszlopban levő számokat a B oszlopban levő számokkal (az egy sorban levőket hasonlítja), és ahol az érték nem egyezik, azok hátterét sárgára színezi.

22. Írjunk az A oszlopba véletlen számú véletlenszerűen generált pozitív egész számokat egymás alá. MsgBoxban írassuk ki a számok közül a legnagyobbat.

23. Mindaddig olvassunk be neveket és hozzájuk tartozó életkorokat, amíg a névnél nem olvasunk be semmit. A beolvasott neveket az első oszlopba, a korokat a második oszlopba írjuk be.

24. A program [0,1) közötti véletlen valós számokat generáljon, és ezeket írja egymás alatti cellákba, amíg egy megadott határértéknél nagyobbat nem generál, ekkor hagyja abba a generálást, és írja ki a generált számok összegét és átlagát. A határértéket a generálás megkezdése előtt egy Inputbox utasítással kérdezze meg (az alapérték legyen 0.5).

25. Olvassunk be egy egész számot, és nézzük meg, hogy megegyezik-e a tükörképével.

26. Írjunk programot, amely bekér egy egész számot és felbontja prímtényezők szorzatára. (Pl. 12 = 2*2*3 – a szorzatban csak prímszámok vannak).

8.5. Tömbök

Az eddig deklarált és használt változók mindegyike egyetlen adat tárolására volt alkalmas. Gyakran jó lenne, hogy a valamiféle módon összetartozó adatokat valamilyen könnyen kezelhető adatszerkezetekben használhassuk.

A tömb olyan adatszerkezet, melyet azonos típusú (a legtöbb programozási nyelvben) elemek rendezett csoportja alkot, és az elemekre a sorszámukkal (indexükkel) hivatkozunk.

A tömböket dimenziójuk szerint csoportosítjuk:

- Egydimenziós tömbök (vektorok): elemei egy sorozatot alkotnak; úgy lehet elképzelni, mint egy Excel tábla egyetlen sora vagy oszlopa, ahol minden egyes cella a tömbnek egy eleme. A vektor elemeit egy sorszámmal azonosítjuk.

- Kétdimenziós tömbök (mátrixok): elemei táblázatban (sorokban és oszlopokban) helyezkednek el. A mátrixok elemeit két indexszel azonosítjuk.
- Többdimenziós tömbök: a háromdimenziós tömböt úgy képzelhetjük el, mint lapokból álló kartotékrendszert, amelyben minden lap egy kétdimenziós táblázatot tartalmaz, és a lapokat egymás után soroljuk. Az n dimenziós tömbök elemeit n darab sorszámmal azonosítjuk.

A VBA legfeljebb 60 dimenziós tömbök létrehozását engedi meg!

8.5.1. Tömbök deklarálása

Egydimenziós tömb deklarálása:

```
Dim tömbnév ([alsóhatár To] felsőhatár) As típusnév
```

- a változónév (tömbnév) után kerek zárójelek között megadjuk az indexhatárokat, majd az As kulcsszóval az elemek (közös) típusát
- az indexhatár számérték, számkonstans vagy ilyenekből képzett aritmetikai kifejezés lehet
- alsó_határ ≤ felső_határ
- az alsó határt nem kötelező megadni; ha nem szerepel, az alapértelmezés szerinti érték lesz
- az alsó indexhatár alapértelmzés szerint a VBA-ban 0, hacsak nem adjuk meg a modul előtt az Option Base 1 utasítást, mellyel 1-re állítjuk a tömbök alsó indexhatárát
- ha nem adunk meg típusnevet, a tömb alapértelmezett típusú lesz (Variant)

Példák: 10 elemű tömbök deklarálása:

`Dim t(9) As Long` ⇔ `Dim t(0 To 9) As Long` – a t tömbnek 10 eleme van, mindenik eleme long típusú

`Dim u(1 To 10) As Integer` – az u is egy 10 elemű tömb, elemei integer típusúak

Többdimenziós tömbök esetén – vesszővel elválasztva – annyi indexhatárpárt, illetve indexhatárt kell felsorolni, ahány dimenziós a tömb:

```
Dim tömbnév([alsóhatár1 To] felsőhatár1, ([alsóhatár2 To]
felsőhatár2[, ...]) As típusnév
```

Példák: 3x3-as mátrixok deklarálása:

`Dim mat(2, 2) As Integer` – a mat tömbnek 3 sora és 3 oszlopa, azaz 9 darab integer típusú eleme van

`Dim a(1 To 3, 1 To 3) As Integer` – szintén 9 elemű mátrix

Az ilyen formában deklarált tömbök fix elemszámúak lesznek a programunk teljes szakaszán. Viszont előfordulhat, hogy több vagy kevesebb elem is elég adataink tárolására. Szeretnénk a programunk egyes részein a tömbünk elemeinek számát szabadon növelni vagy csökkenteni. Ilyenkor dinamikus tömbökről beszélünk, amelyeknek deklarálásakor nem adunk meg indexhatárokat:

```
Dim Változónév() As Változótípus
```

A program során a Redim utasítással tudjuk az elemszámot meghatározni, ugyanolyan formában, mint a fix elemszámú tömbök esetében:

```
ReDim Változónév(felső_indexhatár) As Változótípus
```

vagy

```
ReDim Változónév(alsó_indexhatár To felső_indexhatár) As
Változótípus
```

Mindig figyelni kell arra, hogy a deklarálást követően és a használat előtt ne felejtsük el meghatározni az elemek számát, valamint arra, hogy a használat során ne lépjük túl a megadott indexhatárokat. Mindkét esetben ugyanis a programunk hibajelzéssel leáll. Az egyszer megadott dimenziószám később már nem változtatható meg, de az egyes dimenziók indexhatárait többször is, tetszőlegesen újra lehet definiálni.

A Redim használata során a tömb aktuális tartalma elvész. Ennek elkerülése végett használhatjuk a Preserve szócskát közvetlenül a Redim után:

```
ReDim Preserve Változónév(felső_indexhatár) As Változótípus
```

vagy

```
ReDim Preserve Változónév(alsó_indexhatár To felső_indexhatár)
As Változótípus
```

Variant típusú változókhoz az Array függvénnyel rendelhetők – rögzített hosszúságú karakterláncok és saját típusú változók kivételével – tetszőleges típusú, valamint tetszőleges méretű és dimenziószámú tömbök:

Példa: `Dim x As Variant`
`x = Array("alma", 12, "korte")` – ekkor az x egy 3 elemű egydimenziós tömb, melynek indexei 0, 1 és 2

8.5.2. Hivatkozás a tömb elemeire

A tömbelemekre a tömbnévvel és utána kerek zárójelbe írt indexszel hivatkozunk.

Példák:

1. `Dim t(5) As Integer`
   t(0) – a tömb első eleme, t(1) – a tömb második eleme, t(2) – a harmadik, t(3) – a negyedik, t(4) – az ötödik
2. `Dim mat(2, 2) As Integer`
   a tömb elemei:
   első sor: mat(0, 0), mat(0, 1), mat(0, 2)
   második sor: mat(1, 0), mat(1, 1), mat(1, 2)
   harmadik sor: mat(2, 0), mat(2, 1), mat(2, 2)
3. `Dim a(1 To 3, 1 To 2) As Integer`
   t(1, 1) – első sorbeli első oszlopbeli elem, t(1, 2) – első sorbeli második oszlopbeli elem, ...
   t(3, 2) – a tömb utolsó eleme

– minden tömbelemnek annyi indexe van, ahány dimenziós a tömb,
– az egydimenziós tömbök elemeinek indexét sorszámnak, a kétdimenziós tömbök elemeiét sor- és oszlopszámnak, a háromdimenziósokéit lap-, sor- és oszlopszámnak is nevezzük,
– az index számérték, számkonstans, numerikus változó vagy számot eredményező kifejezés lehet,
– az indexhatárok meghatározzák az indexek lehetséges értékét és természetesen az elemek darabszámát is.

Példák:

`1. Dim lista(10) As Integer`

A tömb 5. eleme legyen 32: lista(4) = 32 (azért 4 az index, mert az alsó index 0-tól kezdődik).

2. Deklaráljunk egy tízelemű tömböt, és töltsük fel 11 és 20 közötti véletlen egész számokkal. Írjuk ki az átlagukat az A1-es cellába.

```
Sub tomb2()
Dim t(1 To 10) As Integer, i%, ossz%
ossz = 0
Randomize
For i = 1 To 10
    t(i) = Int(Rnd * 10) + 11
    ossz = ossz + t(i)
Next
```

```
Cells(1, 1) = ossz / 10
End Sub
```

3. Egydimenziós, 7 Variant típusú elemből álló tömb; indexhatárok 1-től 7-ig:

```
Dim tömb(1 to 7)
```

Lehetséges értékadások például: `tömb(1) = „alma”, tömb(2) = 3.14, tömb(3) = True, … .`

4. Kétdimenziós tömb, amely 48 Boolean (logikai) típusú elemből áll; indexhatárok 1-től 4-ig és 11-től 22-ig:

```
Dim t(1 To 4, 11 To 22) As Boolean
```

5. Variant típusú változó, 7 elemű String típusú tömbbel feltöltve:

```
Dim napok
napok=Array(”Hétfő”, ”Kedd”, ”Szerda”,
”Csütörtök”, ”Péntek”, ”Szombat”, „Vasárnap”)
```

A napok(0) értéke „Hétfő”, ..., a napok(6) értéke „Vasárnap”

Megjegyzés: Ha Variant típusú változóhoz az Array függvénnyel rendelünk tömböt, az alsó indexhatár mindig 0 lesz, akkor is, ha 1-re állítottuk az alapértelmezést!

6. Definiálunk egy egész típusú dinamikus tömböt, majd olvassuk be az elemek számát, futás közben deklaráljuk át a tömböt:

```
Dim szamok() As Integer
n = Inputbox(“Elemek szama:”)
Redim szamok(1 To n) as Integer
’Feltöltjük a tömböt és ha később szeretnénk még például 2
elemmel bővíteni, úgy, hogy az előző értékek megmaradjanak:
Redim Preserve szamok(1 to n + 2) as Integer
...
```

8.5.3. Kitűzött feladatok

1. Olvassunk be n darab egész számot egy tömbbe (max. 10-et), majd írassuk ki a tömb elemeit fordított sorrendben, vagyis először a legutoljára beolvasottat (a kiíratás történhet az Excel munkalapon egy sorba vagy egy oszlopba, de MsgBox-ban is kiírathatjuk).

2. Olvassunk be n darab valós számot egy tömbbe, és vizsgáljuk meg, hány eleme pozitív.

3. Olvassunk be egy n elemű tömböt, majd írassuk ki a tömb legnagyobb elemét és ennek tömbbeli indexét.

4. Töltsünk fel véletlenszerűen egy n elemű tömböt, majd olvassunk be egy x egész típusú változót. Vizsgáljuk meg, hogy az x változó szerepel-e a tömb elemei közt.

5. Írjunk programot, amely beolvas egy mxn-es mátrixot, megszámolja a mátrix páros elemeinek számát.

6. Írjunk programot, amely beolvas egy nxn-es mátrixot, és kiszámolja a főátló alatti elemek összegét.

7. Lottóhúzás szimulálása: 90 darab elem, 5 húzás: kötelezően különbözőeket kell húzni.

8. Írjunk makrót, amely 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.

9. Képezzünk 100 darab 10 és 20 közötti véletlen számot. Ezután írjuk ki, hogy melyik számból összesen hány szám keletkezett.

10. Egy börtönben 1000 cella van, mindegyikben egy rab ül. Kezdetben minden cella zárva van. A börtönőrnek játszani támad kedve: végigmegy az összes cella előtt, és mindegyik ajtó zárján fordít egyet. Fordításkor a nyitott cellát bezárja, illetve a zártat kinyitja. Ha végigment, elkezdi elölről, és minden második cella zárján fordít egyet. Aztán minden harmadikon fordít, és így tovább. Legvégül fordít egyet az ezrediken, és kész. Ezután amelyik cella ajtaja nincs bezárva, abból a rab elmehet. Kik a szerencsés rabok?

8.6. Alprogramok

Nagyobb feladatot célszerű lépésekre bontani. Gyakran szükséges, hogy adott műveleteket többször hajtsunk végre a program különböző területein (pl. ugyanazon adatokat többször szükséges beolvasni, eredményeket kiírni, részfeladatokat leíró utasítássorozatokat többször kell végrehajtani, stb.).

A program strukturálásának alapvető célja, hogy a program szétbontható legyen apróbb, jól áttekinthető részekre, úgynevezett alprogramokra.

A VBA kétféle strukturálást ismer: eljárást és függvényt. Mindkettő képes a hívás helyén számára megadott információkat átvenni, azokon módosításokat is végezni.

8.6.1. Eljárások

Olyan önálló program, mely hívása céljából saját azonosítónévvel és az információcsere céljából paraméterekkel rendelkezik.

Eljárást akkor írunk, ha egy-egy hasonló feladatot többször akarunk a programban elvégezni, vagy a program olvashatósága ettől javul.

8.6.1.1. Általános forma

```
Sub eljárásnév ([formális paraméterlista])  ← rutinfej
   [deklarációs rész] ←  lokális változók deklarálása
    eljárástörzs
End Sub
```

Formális paraméterlista:

- a formális paraméterlista elhagyható, ebben az esetben az eljárás önállóan futtatható; de ha szerepel, akkor a formális paraméterek nevének és típusának vesszővel elválasztott felsorolásából áll:

```
fparnév1[As típusnév1][, fparnév2 [As típusnév2][,... ]]
```

- a formális paraméterek száma tetszőleges, és bármelyikük típusa tetszőlegesen megadható vagy elhagyható,
- definiálásuk egyenértékű a lokális változók deklarálásával: nevük nem ütközhet más lokális deklarációkkal, és nem léteznek az eljáráson kívül,
- az eljáráson belül ugyanúgy használhatók, mint bármely másik lokálisan deklarált változó, és típusuk is tetszőleges beépített vagy felhasználói típus lehet,
- amelyik paraméter neve után nincs típusmegadás, alapértelmezett (Variant) típusú lesz,
- ha egy formális paraméter neve megegyezik egy magasabb szinten deklarált konstans vagy változó nevével, akkor az eljáráson belül a formális paraméter érvényes, és a magasabb szintű objektum nem érhető el,
- a formális paraméterek az eljárás aktiválásakor a hívó rutintól kapnak kezdőértéket
- a hívónak pontosan ugyanannyi és ugyanolyan (vagy illeszthető) típusú adatot (aktuális paraméterek) kell ugyanabban a sorrendben átadnia, ahány formális paramétere van a meghívott eljárásnak.

8.6.1.2. Paraméterátadás

Két paraméterátadási mód létezik:
- Cím szerinti
- Érték szerinti

Cím szerinti paraméterátadás (ez az alapértelmezett):
- a `ByRef` kulcsszóval lehet jelezni, de kiírása nem kötelező,
- a formális paraméter megkapja a hívó rutin megfelelő aktuális paraméterének címét,
- a cím szerinti paraméterátadásnál az aktuális és a formális paraméterek a memóriában ugyanazon a címen helyezkednek el, ilyen esetben az alprogram a paraméter címét veszi át,
- ha egy művelet megváltoztatja a formális paraméter „értékét”, akkor a címen keresztül az aktuális paraméter értéke is megváltozik; szokás ezért az ilyen formális paramétert kimenő paraméternek is nevezni,
- cím szerint átadott paraméternek nem kötelező kezdőértékkel rendelkeznie az alprogram hívása előtt, hiszen épp az a cél, hogy az alprogram törzsében levő utasításokkal ezeknek a paramétereknek adjunk értéket,
- cím szerinti aktuális paraméter nem lehet konstans!

Érték szerinti paraméterátadás:
- a `ByVal` kulcsszóval adjuk meg,
- a formális paraméter megkapja a hívó rutin megfelelő aktuális paraméterének értékét,
- ha egy művelet megváltoztatja a formális paraméter „értékét”, akkor ez sehogy nem hat az aktuális paraméter értékére,
- szokás ezt bemenő paraméternek nevezni,
- az érték szerinti paraméterek helyén az alprogramnak megadhatunk konstanst, változót, illetve kifejezést is; ha kifejezést adunk meg, akkor az először kiértékelődik, majd annak az értéke kerül a formális paraméterbe,
- az érték szerint átadandó paraméternek kötelező értéke legyen, hiszen az alprogram ezzel az értékkel dolgozik.

Lokális (alprogram szintű) deklarációk:
- az alprogramokban is explicit deklarációt használunk, minden lokális konstanst és változót deklarálunk,
- a lokális nevek megegyezhetnek a modális vagy a globális nevekkel → félreértésekhez vezethet! Ilyen esetekben mindig a legalacsonyabb szintű név van érvényben,
- alprogramon belül nem lehet globális konstanst vagy változót deklarálni.

Konstansok deklarálása:

```
Const konstnév1 = kifejezés1 [,konstnév2 = kifejezés2[,..]]
```

Ezek az alprogram aktiválásakor, az inicializálás során kapnak értéket.

Változók deklarálása:

```
{Dim|Static} név1 [As típusnév1] [, név2 [As típusnév2] [...]
```

– a szabályok ugyanazok, mint a globális deklarációnál: ha a névlista egynél több változónevet tartalmaz, ezeket vesszővel kell elválasztani; ha egy név után nem adunk meg típusnevet, az illető változó az alapértelmezett (Variant) típust kapja,
– az alprogramok aktiválásakor a lokális konstansok megkapják értéküket,
– a VBA az első alprogramhívás alkalmával típusuknak megfelelő kezdőértékkel feltölti az összes lokális változót,
– újabb meghívás esetén azonban a statikus változókat nem kell inicializálni,
– a Static kulcsszóval deklarált változók értéke az alprogram befejeződése után is megmarad addig, amíg a modullap nyitott állományban van,
– a Dim kulcsszóval deklarált változók értéke az alprogram befejeződésekor elvész.

Az alprogramok törzsét utasítások alkotják.
Hatásuk szerint megkülönböztethetünk:
– aritmetikai, logikai és szöveges értékadó,
– vezérlő (elágaztató, ciklusképző, ugró, rutinhívó),
– be/kimeneti (adatbekérő és üzenetkiíró, állománykezelő),
– objektumkezelő (létrehozó, átalakító, tulajdonságmódosító) utasításokat.

Egy alprogram törzsében nem állhat újabb rutinfej, azaz az alprogramokat nem lehet egymásba ágyazni!

8.6.1.3. Eljárás meghívása

Az eljárásokat egyszerűen nevük leírásával vagy a Call kulcsszóval lehet meghívni:

```
Eljárásnév aktuális-paraméterek
```

vagy

```
Call Eljárásnév (aktuális-paraméterek)
```

- Cím szerinti paraméterátadás esetén az aktuális paramétereket változókban kell megadni, érték szerinti átadás esetén konkrét értékeket is megadhatunk.
- Ha csak nevével hívjuk meg az eljárást: az aktuális paramétereket csak fel kell sorolni (sorrend és típus betartásával!); ilyenkor a zárójeleket tilos kitenni.
- Ha az eljárásnak nincs formális paramétere, tilos kitenni az „üres” zárójelpárt.
- Ha az eljárást Call kulcsszóval hívjuk: a paraméterlistát kötelező zárójelbe tenni; ha az eljárásnak nincs formális paramétere, az „üres” zárójelpárt ki lehet tenni, de a VBE (Visual Basic Editor) automatikusan letöröli.

Mivel a rutinok alapértelmezésben nyilvánosak (Public), az egy munkalapon levő vagy a különböző modulokban írt rutinok „látják” egymást.

Különböző munkalapon levő rutinok hívásakor meg kell adni a rutint tartalmazó objektum (munkalap vagy párbeszédlap) hivatkozási nevét is.

Paraméterátadás:

– az aktuális paraméterlista változók és/vagy kifejezések vesszővel elválasztott felsorolása,
– az aktuális paraméterek számának és típusának – a megfelelő sorrendben – általában meg kell egyeznie a formális paraméterekével (e tekintetben mindegy, hogy a formális paraméterek ByRef vagy ByVal módúak-e),
– ha a formális paraméter Variant típusú, a megfelelő aktuális paraméter bármilyen típusú kifejezés, karakterlánc vagy (akár több dimenziós) tömb is lehet,
– az elvárt egyezéseket a fordító ellenőrzi; ha a formális és az aktuális paraméter típusa nem egyezik meg, azt a fordító jelzi,
– a fordító a futás során (majd) keletkező túlcsordulásokat (előre) nem tudja kezelni; ha például egy Byte típusú formális paraméterre rákényszerítettünk egy Long típusú aktuális paramétert, és futáskor az utóbbi értéke kívül esik a byte határokon, rutinunk futási hibával elakad.

Példa: Írjunk eljárást, amely a kijelölt tartomány betűszínét, betűméretét, betűstílusát, illetve háttérszínét a paraméterként kapott értékekre állítja. Írjunk makrót, amely két tetszőleges tartományra meghívja az eljárást, különböző tetszőleges aktuális paraméterekkel.

```
Sub formazas(ByVal bszin As String, ByVal bmeret As Integer,
ByVal bfelk As Boolean, ByVal hszin As String)
    Selection.Font.Color = bszin
    Selection.Font.Size = bmeret
    Selection.Font.Bold = bfelk
```

```
    Selection.Interior.Color = hszin
End Sub

Sub teszt_formazas()
Range("A1:B4").Select
formazas vbRed, 14, True, vbYellow
Range("A6:b9").Select
formazas vbBlue, 12, False, vbYellow
End Sub
```

8.6.1.4. Kitűzött feladatok

1. Töltsünk fel egy 10×10 mezőt véletlen számokkal, melyek értéke 1 és 100 közé esik. Majd a hárommal osztható számokat tartalmazó cellákat színezzük pirosra, az öttel oszthatókat kékre, a héttel oszthatókat zöldre. A színezéshez használjunk szubrutint (eljárást), amelynek a paraméterei: egy string (a szín megadása), két integer (a színezendő cella koordinátái).

2. Írjunk eljárást, amely a paraméterként kapott karakterlánc karaktereit egyenként véletlenszerűen előállított koordinátájú cellákba írja.

3. Írjunk programot, amely 50 darab −50..50 intervallumba eső véletlen egész szám közül először kiválogatja, majd egyszerre kiírja az összes pozitív számot! (Írjuk eljárásba vagy függvénybe a tömb generálását, az elemek kiválogatását és kiíratását.)

4. Írjunk eljárást, amely megfordít egy tetszőleges szöveget. A program tegye az A oszlopban levő szövegek tükörképét a B oszlopba.

5. Írjunk eljárásokat, melyek paraméterként egy tömböt kapnak, és választ adnak az alábbi kérdésekre:

a) Van-e 0 a tömbben?

b) Van-e 100-nál nagyobb szám?

c) Mi az első 100-nál nagyobb szám?

d) Hány darab 100-nál nagyobb szám van?

e) Hány darab 0 van?

f) Mi a legnagyobb szám a tömbben?

g) A 100-nál nagyobb számokat írjuk át egy másik tömbbe.

6. Írjunk eljárásokat, egyik a paraméterként kapott 2 dimenziós tömböt feltölti elemekkel, a másik kiírja a tömböt, a harmadik összeadja a két tömb elemeit egy új tömbbe.

7. Írassuk ki a Pascal-háromszög egy részletét a munkalapra! InputBoxban beolvassuk a háromszög sorainak számát, ezt adjuk át paraméterként a Pascal-háromszöget kiíró eljárásnak.

Például n = 5 sorra:

```
1
1       1
1       2       1
1       3       3       1
1       4       6       4       1
```

Szépen:

```
                        1
                    1       1
                1       2       1
            1       3       3       1
1           4           6           4           1
```

8.6.2. Függvények

A függvények pontosan úgy működnek, mint az eljárások, azzal a különbséggel, hogy visszatérési értéket is szolgáltatnak.

8.6.2.1. Általános forma

```
Function függvénynév ([formális paraméterlista]) [As típusnév]
   [deklarációs rész]
   függvénytörzs
   függvénynév = kifejezés
End Function
```

Nem kötelező megadni a függvények visszatérési típusát ([As típusnév]), ha ezt elhagyjuk, a függvény alapértelmezett (Variant) típust ad vissza.

A formális paraméterlista tartalmazza azokat a paramétereket, melyeket át kell vennie a hívó rutintól (olyan, mint az eljárások esetén – lehet cím szerinti és érték szerinti).

8.6.2.2. Függvény hívása

```
{[változónév =|Call]} [projektnév.] [objektumnév.]
függvénynév[(aktuális paraméterlista)]
```

– ha függvényként kívánjuk meghívni, a függvény nevének értékadó utasítás jobb oldalán vagy kifejezés tényezőjeként kell szerepelnie, és az aktuális paraméterlistát zárójelek között kell felsorolni,
– ha a függvénynek nincs formális paramétere, az „üres" zárójelpárt ki lehet tenni, de a VBA automatikusan letöröli,
– ha egy függvényt Call kulcsszóval hívunk meg, az eljáráshívási szabályokat kell alkalmazni; ilyenkor a függvény eljárásként működik, és visszatérési értéke elvész.

8.6.2.3. Függvény visszatérési értéke

A függvény típusát (visszatérési értékének típusát) a függvény deklarálásánál lehet megadni az As kulcsszó után. A függvény törzsében legalább egy olyan értékadó utasításnak kell szerepelnie, amelynek bal oldalán a függvény neve áll; ha a futás során egyetlen ilyen utasítás sem hajtódik végre, a függvény a típusától függő „üres" kezdőértékkel tér vissza. A String típusú függvény karakterláncot ad vissza, amelynek maximális hossza 255 karakter lehet, a Variant típusú függvény visszatérési értéke bármilyen típusú változó (akár String is) lehet, tömb is megengedett. Ha nincs szükség a visszaadott értékre, a függvényt eljárásként hívjuk meg, a Call kulcsszóval.

Példák:

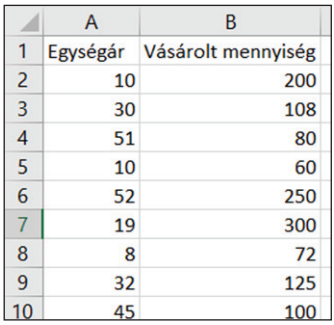
1. Írjunk függvényt, mely kiszámítja két szám szorzatát:

```
Function szorzat(ByVal x As Integer, ByVal y As Integer) As Integer
   szorzat = x * y
End Function
```

Függvény meghívása:

```
Private Sub CommandButton2_Click()
   Dim a%, b%, c%
   a = InputBox("Elso szam:")
   b = InputBox("Masodik szam:")
   c = szorzat(a, b)
   MsgBox "A szorzat: " & c
End Sub
```

2. Egy Excel munkalap A oszlopa bizonyos termékek egységárát tartalmazza, a B oszlopban az adott termékből vásárolt mennyiség található. Tudjuk, hogy a vásárolt mennyiségtől függően a következő kedvezmények járnak az egyes termékekre: 50 darab alatt a kedvezmény 2%, 50 és 99 között a kedvezmény 5%, 100 és 199 között a kedvezmény 8%, legalább 200 darab vásárlása esetén 10%-os kedvezményt kapunk az egységárból. Számítsuk ki a C oszlopban a termékek kedvezményes árát.



	A	B
1	Egységár	Vásárolt mennyiség
2	10	200
3	30	108
4	51	80
5	10	60
6	52	250
7	19	300
8	8	72
9	32	125
10	45	100

8.6.2.3.1. ábra. *Terméknyilvántartás példa*

```
Function kedvezmenyes_ar(ByVal ar As Double, ByVal m As
         Integer) As Double
Dim k As Double
Select Case m
    Case Is < 50
        k = ar * 0.02
    Case Is < 100
        k = ar * 0.05
    Case Is < 200
        k = ar * 0.08
    Case Else
        k = ar * 0.1
    End Select
  kedvezmenyes_ar = ar - k
End Function

Sub kedvezmenyesar_feltoltes()
   Dim i As Integer
   Cells(1, 3) = "Kedvezmenyes ar"
   i = 2
   Do While Cells(i, 1) <> ""
    Cells(i, 3) =
    kedvezmenyes_ar(Cells(i, 1), Cells(i, 2))
    i = i + 1
   Loop
End Sub
```

8.6.2.4. Kitűzött feladatok

1. Írjunk függvényt, amely a sugár megadása után kiszámítja a gömb térfogatát (V = 4r3∏/3).

2. Írjunk függvényt, amely két valós számot összead.

3. Írjunk függvényt, amely a három bemenő paraméter átlagának négyzetgyökét adja vissza.

4. Írjunk függvényt, amely megszámolja, hogy a kapott szövegben mennyi szóköz van.

5. Írjunk két függvényt, amely az oldalak megadása után kiszámítja a téglalap kerületét és területét.

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

7. Írjunk függvényt, amely a bemenő karakterlánc minden második betűjét adja vissza nagybetűvé alakítva.

8. Írjunk függvényt, amely megszámolja, hogy egy karakterlánc hány darab számjegyet tartalmaz.

9. Írjunk függvényt, amely visszaad egy véletlen hosszú, véletlen betűkből álló karakterláncot. A karakterlánc maximális hosszát paraméterben adjuk meg.

10. Írjunk programot, amely tartalmaz egy függvényt, ami a paraméterként kapott tömb elemeinek átlagát adja vissza.

11. Írjunk függvényt, amely a bemenő karakterláncot titkosítja: 1-gyel elcsúsztatja a karaktereket (egy karakter Ascii kódját az Asc függvénnyel kapjuk meg, míg egy kódnak megfelelő karaktert a Chr függvénnyel).

Pl. Titkosit(“ablak”) > ”bcmbl”

12. Dobjon a számítógép két hatoldalú kockával (véletlenszerűen generáljuk), és a dobások eredményét egymás melletti oszlopokba írja ki. Ezt addig ismételje, míg ötször nem dob azonosat a két kockával. Ezután a dobások számát írja ki MsgBoxban. A dobásokat külön függvény végezze. Az azonos dobásokat tartalmazó cellákban a szöveg színe legyen piros.

13. Dobjon a program hatoldalú kockával százszor, miközben a dobások értékét egymás alatti cellákba írja ki. Hatos dobás esetén dobjon újra, és az értéket a mellette lévő cellába írja be, ismételt hatos dobás esetén az új számot a harmadik oszlopba, és így tovább, míg végül nem hatost dob. Az egy sorba írt számokat egy dobásnak véve (pl. két hatos és egy kettes dobás értéke: 14) keresse meg a legnagyobb dobást, és a dobást tartalmazó sor számát írja ki.

8.7. Rekurzió

A rekurzió egy nagyon fontos programozási technika, melynek ötletét a természetből kölcsönöztük. A genetikai kód, az élet kialakulása, a természeti jelenségek egész sora a rekurzió elve szerint működik. A programozásban a lényege, hogy önmagát hívó eljárás vagy függvény. Az egyik legegyszerűbb példa az első n természetes szám összege. Világos, hogy ezt meg lehet oldani iterációval, de mi most a rekurzió lényegét szeretnénk megértetni. Az iterációval ellentétben most az utolsó tagtól kezdjük, és nem az elsőtől. Mert ha iterációval kezdenénk, akkor természetesen az első taggal kezdjük, és leutánozzuk a természetes összeadást: 0 + 1 + 2 + … + n.

Ahogy az alábbi program is ezt elvégzi:

```
Sub Összeg()
Dim n As Integer
Dim S As Integer
S = 0
n = InputBox("Meddig összegezzünk")
For i = 1 To n
 S = S + i
Next
MsgBox S
End Sub
```

Azonban a rekurziónál pont fordítva lesz: Ha például 5-ig összegzünk, akkor az utolsó tagtól kezdjük, az 5-től, és úgy haladunk visszafelé 0-ig! Ezt szemléletesen az alábbi függvény mutatja:

```
Function S(n As Integer) As Integer
If n = 0 Then
 S = 0
Else
 S = n + S(n - 1)
   End If
End Function
Sub foprogram()
MsgBox S(5)
End Sub
```

Létezik egy olyan programozási tétel, hogy minden rekurzió átalakítható iterációvá (hiszen a számítógép memóriájában rekurzió kifejtésénél lényegében iteráció történik).

Programozástechnikailag azonban a rekurzióval jóval könnyebb és szemléletesebb a munka. Ezt akkor értjük meg, ha a későbbiekben átvesszük a Hanoi-tornyai nevű eljárást, vagy a Koch-görbe, Sierpinski-csipkék rajzoltatását.

A rekurzió komoly odafigyelést érdemel, mert például a Fibonacci-sorozat értelmezése lényegében rekurzív, ezért nem szabad rekurziót használni a Fibonacci-sorozat elemeinek a származtatásánál. Mégpedig azért nem, mert ott rekurziós robbanás történik, ami azt jelenti, hogy ugyanazt a részt a gép többször kifejti (fölöslegesen).

```
Function Fib(n As Integer) As Integer
If n = 0 Or n = 1 Then
      Fib = 1
Else
      Fib = Fib(n - 1) + Fib(n - 2)
End If
End Function

Sub fibonacci()
MsgBox Fib(13)
End Sub
```

A Fibonacci-sorozat generálását a legegyszerűbben éppen az Excel relatív címzése illusztrálja:

C	D
0	1
1	1
2	=D2+D1
3	=D3+D2
4	=D4+D3

0	1
1	1
2	2
3	3
4	5

8.7.1. ábra. *Fibonacci-sorozat generálása*

A C oszlopban a generáció van, és a D oszlopban lesznek a Fibonacci-számok, a lényeg az előtte lévő kettő összege.

A számelméletben tökéletes számnak nevezzük azokat a természetes számokat, amelyek megegyeznek az önmaguknál kisebb osztóik összegével. Vagy, ami ezzel ekvivalens, hogy tökéletes szám minden olyan n egész, amelyre az összes osztójának összege pont a szám 2-szerese.

A definíció az ókorból származik, már Eukleidész *Elemek* című művében is megjelenik,[1] τέλειος ἀριθμός (tökéletes, ideális vagy teljes szám) néven. Eukleidész meghatározott egy képzési szabályt is,[2] miszerint [q(q+1)]/2 páros tökéletes szám, amennyiben $q = 2^p - 1$, p és q prímek – az ilyen alakú számokat jelenleg Mersenne-prímeknek nevezzük. Jóval később Euler igazolta, hogy az összes páros tökéletes szám ebben az alakban írható fel. Ez az Eukleidész–Euler-tétel.

Nem ismeretes, hogy létezik-e páratlan tökéletes szám, ahogy az sem, hogy létezik-e végtelen sok tökéletes szám.

Az ókori görögök csak a négy legkisebb tökéletes számot (6, 28, 496, 8128) ismerték. Nyitott kérdés, hogy léteznek-e páratlan tökéletes számok. Számos eredmény született ebben a témában, de egyik sem mutatott rá egy páratlan tökéletes számra vagy cáfolta ezek létezését. Többen vélik úgy heurisztikus érvek alapján, hogy páratlan tökéletes számok nem léteznek.

Írjunk programot, amely kiírja az első 4 tökéletes számot.

Didaktikailag építjük fel a megoldást, hogy könnyen érthető legyen:

Első lépésben megírjuk azt a programot, amely az első n természetes szám összegét számolja ki:

```
Sub osszeg()
,az elso n termesztes szam összegét adja meg
Dim a As Integer
Dim s As Integer
Dim i As Integer
n = InputBox(„kerem az n-et“)
s = 0
For i = 1 To n
  s = s + i
Next
MsgBox “Az osszeg = “ & s
End Sub
```

Második lépésben megírjuk azt a programot, amely egy egész szám osztóit fogja kiírni a munkalapra szépen egymás alá:

```
Sub osztok1()
,kiírja egy egész szám osztóit, szépen egymás alá
Dim szam As Integer
Dim i As Integer
sor = 1
szam = InputBox(„Kerem a egesz szam“)
```

1 VII.22. (7. Könyv 22. Paragrafus.)

2 IX.36.

```
For i = 1 To szam \ 2
 If szam Mod i = 0 Then
  Cells(sor, 1) = i
  sor = sor + 1
 End If
Next
End Sub
```

És akkor most már megírhatjuk azt a programot, amely kiírja a tökéletes számokat, mondjuk 100 000 (százezerig). Figyelem! A program kb. 2 percet fut. Nem figyeltünk a programunk hatékonyságára, nem működik gyorsan a programunk. Az már a jó, tapasztalt programozók feladata, hogyan kell ezt a programot gyorssá, hatékonnyá tenni.

Másfelől nem tudjuk, hogy létezik-e páratlan tökéletes szám, de ha igen, az biztos, hogy nagyobb, mint 10^{1500}. Ez elképzelhetetlenül nagy szám, mert a világmindenség kora is csak 10^{13} nagyságrendű.

```
Sub tokeletes()
,kiírja 100000 százezerig a tökéletes számokat
Dim szam As Long
Dim i As Long
Dim osztok As Long
sor = 1
For szam = 3 To 100000
osztok = 0
For i = 1 To szam \ 2
 If szam Mod i = 0 Then osztok = osztok + i
Next i
If osztok = szam Then
 Cells(sor, 1) = szam
 Cells(sor, 2) = "ez tokeletes"
 sor = sor + 1
End If
Next szam
End Sub
```

Programozástechnikailag a következő lépés, hogy írassuk ki a barátságos számokat!

A számelméletben azokat a számpárokat, amelyekre igaz, hogy az egyik szám önmagánál kisebb osztóinak összege a másik számmal egyenlő és fordítva, barátságos számoknak hívjuk.

Ilyen például a (220; 284) számpár.

220 önmagánál kisebb osztói: 1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110.

1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284

284 önmagánál kisebb osztói: 1, 2, 4, 71, 142.

1 + 2 + 4 + 71 + 142 = 220.

Írjunk programot, amely kiírja a barátokat 1-től 10000-ig!

```
Sub baratok()
Dim peter As Long
Dim pal As Long
Dim peter_osszeg As Long
Dim pal_osszeg As Long
Dim i As Long
Dim fel_peter As Long
Dim fel_pal As Long
Dim j As Integer
peter = 20
j = 1
Do While peter < 10000
  peter_osszeg = 0
  fel_peter = peter \ 2
 For i = 1 To fel_peter
  If peter Mod i = 0 Then peter_osszeg = peter_osszeg + i
 Next
 pal = peter_osszeg
 pal_osszeg = 0
 fel_pal = pal \ 2
 For i = 1 To fel_pal
  If pal Mod i = 0 Then pal_osszeg = pal_osszeg + i
 Next
 If (pal_osszeg = peter) And (peter <> peter_osszeg) Then
  MsgBox "baratok" & peter
  Cells(j, 1) = peter
  MsgBox "baratok" & pal
  peter = pal + 1
  Cells(j, 2) = pal
  j = j + 1
 Else
  peter = peter + 1
```

```
 End If
Loop
End Sub
```

8.8. Grafika Excelben

A klasszikus programozási nyelvekben megszokott grafikai képességekhez hasonló lehetőségeket alapból nem biztosít az Excel. Ellenben biztosít rajzobjektumokat (Draw objects), mint például vonalakat és különböző formákat. Mivel ezek VBA-ban objektumok, ezért kódból lehet velük dolgozni, mint például ilyen objektumokat létrehozni és megjeleníteni, tulajdonságaikat módosítani, valamint ezeket hozzá lehet adni egy rajzlaphoz. A rajzlapot Excelben nem egy tradicionális rajzlapként kell elképzelni, amire tetszőleges formákat tudunk rajzolni, hanem csak az előre definiált Draw objektumokat tudjuk elhelyezni rajta. A grafikai lehetőségek hiányának pótlására létrehoztunk egy VBA osztályt *DrawClass* névvel, mely a Pascal programozási nyelv néhány jellegzetes függvényéhez hasonló megoldásokat kínál.

8.8.1 Draw osztály

Bár VBA-ból a Draw objektumokat direkt egy munkalapra is el lehet helyezni, a *DrawClass* segítségével való rajzolás megvalósítása egy rajzlap (Canvas) segítségével történik. Ez lényeges, mert így egyben lehet tartani a rajzlapokat a rajtuk elhelyezett Draw objektumokkal, egyszerre több rajzlapot is meg lehet jeleníteni, függetlenül bármilyen más, a munkalapon meglévő objektumtól. Továbbá, a *DrawClass*, mint osztály, elkülöníti a saját változóit a többi VBA változóitól, így a külön rajzlapok esetén minden példány saját, egymástól független tulajdonságokkal rendelkezik. A következőkben a rajzosztály felépítését, használatát és néhány példát mutatunk be.

Fő metódusok:

- Init(Width, Height): osztály inicializáló metódusa, mely létrehozza a rajzlapot a megadott szélességgel és magassággal.
- Clear(): rajzlap tartalmának törlése.
- Focus(): a *DrawClass* objektumhoz tartozó rajlap fókuszba való helyezése.
- MoveTo(X, Y): rajzolási kurzor mozgatása az X, Y pozícióra.
- DrawLineTo(X, Y): vonal rajzolása a kurzor jelenlegi pozíciójától a megadott X, Y koordinátákig.
- DrawLineRGBTo(X, Y, R, G, B): hasonlóan a DrawLineTo metódushoz, vonalat húz a megadott RGB (piros, zöld, kék) színekkel. Az RGB csatornák byte-változók, értékei 0–255 között lehetnek.

- DrawLineOptionsTo(X, Y, R, G, B, LineDashStyle): hasonló a DrawLineRGBTo, az extra LineDashStyle paraméterrel a vonal stílusát lehet megadni (például folytonos, szaggatott, pontozott stb.). Értékei az MsoLineDashStyle felsorolásban definiáltak.
- DrawLine(X1, Y1, X2, Y2): vonal rajzolása X1, Y1 koordinátáktól X2, Y2 koordinátákig.
- DrawLineRGB és DrawLineOptions: hasonló a DrawLine metódushoz, a szín és a vonal stílusának megadásával.
- DrawRectangle(X1, Y1, X2, Y2): téglalap rajzolása az X1, Y1 koordinátájú (bal felső sarok) és az X2, Y2 koordinátájú (jobb alsó sarok) pontok között.
- TranslateX(X) és TranslateY(Y): átszámoló függvények, melyek egy fiktív 0–1 közötti értékű, tizedes pontossággal megadott léptékű rajzlap koordinátáinak átalakítását végzik az aktuális Width*Height méretű rajzlapra.
- DrawSelection(): egy kétoszlopos szelekció rajzolása. A két oszlop sorai egy-egy X és Y koordinátát jelentenek, a rajzolás pedig ezek vonalakkal való összekötése lesz.
- DrawSelectionLines(): a *DrawSelection* metódushoz hasonlóan működik, a szelekció viszont négyoszlopos kell legyen, ahol minden sor egy X1, Y1, X2, Y2 vonal koordinátáit jelenti.

Rajzlap inicializálása és használata

A DrawClass használatához szükséges ezt hozzáadni a megnyitott Excel munkafüzethez. Ez a DrawClass.cls állomány importálásával történik, éspedig a VBA szerkesztő ablakon a „File” menüből az „Import File” segítségével.

A rajzolás a *DrawClass* egy példányának létrehozásával és az `Init` metódus meghívásával kezdődik. Példa egy 500 pixel széles, 500 pixel magas rajzlap létrehozására egy „rajz” változóba:

```
Dim rajz As New DrawClass , rajzoló osztály példány létrehozása
Call rajz.Init(500, 500)  , a rajz példány inicializálása
```

A rajzlap inicializálása után a „rajz” változón meghívott metódusok a hozzá tartozó rajzlapra fognak rajzolni. Például húzzuk be a rajzlap átlóit:

```
rajz.DrawLine(0, 0, 500, 500) , a rajz változó által jelölt
rajzlapon rajzolás
rajz.DrawLine(0, 500, 500, 0)
```

Az aktív munkalapról a rajzlapokat tartalmazó diagramok akár kézzel, akár az alábbi különálló kódrészlet segítségével törölhetők. A hiba elkerülése érdekében először ellenőrizni kell, hogy létezik-e diagram objektum (`ChartObjects` elemek

száma nagyobb, mint nulla). Ha igen, akkor ki lehet adni ezekre a törlési parancsot (`ChartObjects.Delete`).

```
If ActiveSheet.ChartObjects.Count > 0 Then
    Application.ActiveSheet.ChartObjects.Delete
End If
```

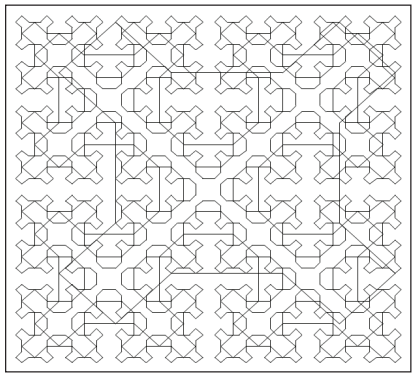
A fenti kódrészlet nemcsak a *DrawClass* által létrehozott diagramokat törli, hanem az aktív munkalapon az összeset!

Korlátok: mivel a rajzok egyedi objektumokból tevődnek össze, amiket az Excel egyenként nyilvántart, szükség esetén, például a munkalap frissítése esetén, ezeket is egyenként újrarajzolja. Ennek következtében a rajz komplexitásával egyenletesen több processzoridőt igényel az Excel, azaz minél nagyobb terjedelmű egy rajz, annál lassúbb lesz maga az Excel.

8.8.2. Példák

8.8.2.1. Sierpinsky:

A Sierpinsky nevet adtuk az eljárásnak, de lényegében egy négyzetet kitöltő görbe, így talán helyesebb lett volna a Peano-görbe megnevezés. A geometriában a Peano-görbe az első példa a térkitöltő görbére. Giuseppe Peano 1890-ben fedezte fel. A Peano-görbe egy szürjektív, folytonos függvény, egy intervallumot képez egy négyzetre, azonban ez nem injektív. Peanót Georg Cantor korábbi eredménye motiválta, hogy ennek a két halmaznak ugyanaz a kardinalitása (számossága). E példa miatt egyes szerzők a Peano-görbe kifejezést használják általánosabban bármely térkitöltési görbére.

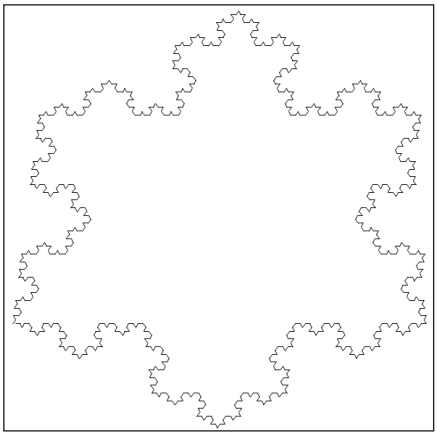


8.8.2.1.1. ábra. *Sierpinsky térkitöltő görbe*

8.8.2.2. Koch:

A Koch-görbe vagy Koch-hópehely Helge von Koch svéd matematikus által 1904-ben leírt fraktál, mely ilyen minőségében az egyik legelső. Ha csak az egyik oldalát vesszük figyelembe, akkor szép példa egy folytonos függvényre, amelyik egy pontjában sem deriválható!

A görbét úgy állíthatjuk elő, hogy egy szabályos háromszög oldalait elharmadoljuk, majd a középső harmadára ismét egy szabályos háromszöget rajzolunk. Ezen háromszögek oldalait szintén harmadoljuk, és háromszöget rajzolunk rájuk. Ezt a végtelenségig folytatjuk. A görbe hossza az n-edik lépés után $(4/3)^n$ A határértékként kapott görbe végtelenül finoman strukturált, és csak közelítőleg lehet ábrázolni. Azok a pontok alkotják, amiket egy iterációs lépés után a további iterációs lépések megőriznek, vagy torlódási pontjai ennek a ponthalmaznak. Sokszor ennek az önmagába záródó görbének harmadát hívják Koch-görbének.



7.8.2.2.1. ábra. *Koch-görbe*

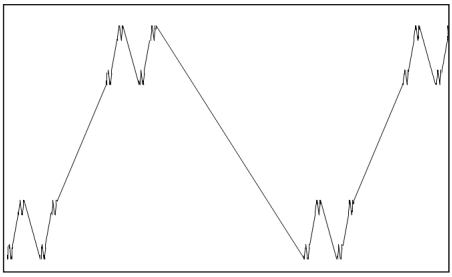
8.8.2.3. Ene-féle függvények:

Ezek a függvények, amelyeket Ene-féle függvényeknek nevezhetünk, példákként és ellenpéldákként szolgálnak bizonyos függvényosztályokra, azok létezésére, vagy azért, hogy egyik függvényosztálynak a másikban való bennfoglalása szigorú-e vagy sem. Ezek a függvények az úgynevezett Cantor-féle perfekt halmaznak karakterisztikusai, és egy példa éppen a Cantor-féle harmadolási függvény, de más formában bemutatva, mint a hagyományos.

Ezen függvények vizuális bemutatására Ene egy speciális technikát használ (doboz-képek), vagyis a grafikus kép úgy képződik, hogy téglalapok szakaszokkal

vannak összekötve, és a téglalapokba rekurzíven újra beépítjük az egész konstrukciót. Amikor a téglalapokba beépítjük az egészhez hasonló szerkezetet, akkor egy sorozatot értelmezünk, és ezek metszete fogja adni a keresett folytonos függvényt.

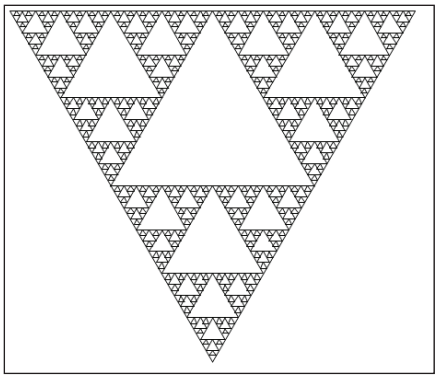
Ezekkel a függvényekkel, amelyeket nevezhetünk Ene-féle függvényeknek, jutnak kifejezésre azok az új tulajdonságok, amelyeket az eddigi valós integrálelmélet nem vizsgált. Természetesen ahhoz, hogy ezeket meg tudjuk fogalmazni, szükségünk van a pontos matematikai nyelvre. Ezt végezte el a fentiekben Vasile Ene.



7.8.2.3.1. ábra. Ene-féle függvény

8.8.2.4. Sierpinski-háromszög

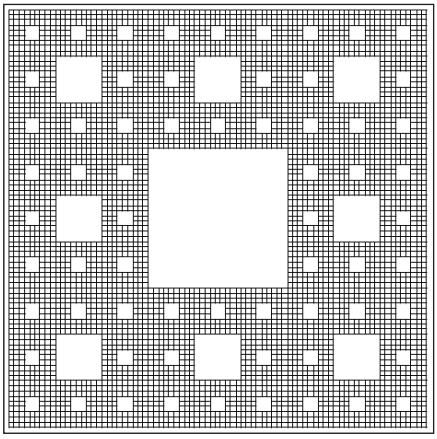
Az eljárásnak a Csipke nevet adtuk, de ez lényegében a Sierpinski-háromszög. A Wacław Sierpinski lengyel matematikus által megtalált fraktál úgy áll elő, hogy egy szabályos háromszögből elhagyjuk az oldalfelező pontok összekötésével nyert belső háromszöget, majd az így maradt három háromszögre rekurzívan alkalmazzuk ugyanezt az eljárást.



7.8.2.4.1. ábra. *Sierpinski-háromszög*
Hausdorff-dimenziója log(3)/log(2) ≈ 1,585

8.8.2.5. Cantor

A matematikában a Cantor-halmaz, Cantor-por vagy Cantor-diszkontinuum a valós számok egy meghatározott részhalmaza, amely több különleges tulajdonsággal bír. Logikailag analóg az előbbi, általunk Csipkének nevezett Sierpinksi-háromszöggel. Az Excel lényegében nem egy grafikus szoftver, ezért ez az eljárás nagyon-nagyon lassú! Csak didaktikai célra használható! Azonban az volt a célunk ezzel, hogy a matematikában nehezen érthető, absztrakt halmazokat is szemléltessük az Excel használatával, és a jegyzetünk tudományos része, amely új lehetőségeket és távlatokat nyújt a hallgatóknak.



7.8.2.5.1. ábra. Cantor-halmaz ábrázolása

A LEGFONTOSABB FÜGGVÉNYEK ANGOL–MAGYAR MEGFELELŐI

Statisztikai függvények

Angol	Magyar	Leírás
AVEDEV	ÁTL.ELTÉRÉS	Az adatpontoknak átlaguktól való átlagos abszolút eltérését adja eredményül, mely egy adathalmaz szóródásának a mérőszáma
AVERAGE	ÁTLAG	A számtani átlagot adja eredményül
BETADIST	BÉTA.ELOSZLÁS	A béta-eloszlás sűrűségfüggvénye
BINOMDIST	BINOM.ELOSZLÁS	A binomiális eloszlás sűrűség-, illetve eloszlásfüggvénye
KURT	CSÚCSOSSÁG	Egy adatsorozat eloszlásfüggvényének a normális eloszláshoz viszonyított csúcsosságát vagy laposságát adja meg
COUNTA	DARAB2	Megszámolja, hogy argumentum tartományában hány nem üres érték található
COUNTA	DARAB	Megszámolja, hogy argumentum tartományában hány szám található
FORECAST	ELŐREJELZÉS	Lineáris regresszióval előre jelez
EXPONDIST	EXP.ELOSZLÁS	Exponenciális eloszlás, illetve sűrűségfüggvény
FDIST	F.ELOSZLÁS	F-eloszlás két adathalmaz eltérési fokának megállapítása
FTEST	F.PRÓBA	F-próba értékét számítja ki
SKEW	FERDESÉG	Egy eloszlás ferdeségét számolja ki
FISHER	FISHER	Fisher-transzformációt végez
GAMMADIST	GAMMA.ELOSZLÁS	Gamma-eloszlás sűrűséget, illetve eloszlásfüggvényt számol
GAMMALN	GAMMALN	A gamma-függvény természetes alapú logaritmusát számítja ki
FREQUENCY	GYAKORISÁG	Gyakorisági tömböt ad vissza
HARMEAN	HARM.KÖZÉP	Harmonikus átlagot számol
HYPGEOMDIST	HIPERGEOM.ELOSZLÁS	Hipergeometrikus eloszlást számol
BETAINV	INVERZ.BÉTA	A béta-eloszlás sűrűségfüggvényének inverzét számítja ki

Angol	Magyar	Leírás
FISHERINV	INVERZ.FISHER	Inverz Fisher-transzformációt végez
FINV	INVERZ.F	Az F-eloszlás inverzét számolja ki
GAMMAINV	INVERZ.GAMMA	A gamma-eloszlás inverzét számolja ki
CHIINV	INVERZ.KHI	A khi-négyzet-eloszlás inverzét számítja ki
LOGINV	INVERZ.LOG.ELOSZLÁS	A lognormális eloszlásfüggvény inverzét számítja ki
NORMINV	INVERZ.NORM	A normális eloszlás eloszlásfüggvényének inverzét számolja ki
NORMSINV	INVERZ.STNORM	A standard normális eloszlás inverzét számolja ki
TINV	INVERZ.T	A Student-féle t-eloszlás inverzét számítja ki
CHIDIST	KHI.ELOSZLÁS	A khi-négyzet-eloszlást számolja ki
CHITEST	KHI.PRÓBA	Függetlenségvizsgálatot hajt végre khi-próbával
SMALL	KICSI	Egy adatsor rendezése után a k. legkisebb elemét adja meg
CORREL	KORREL	Két adatsor korrelációs együtthatóját számítja ki
COVAR	KOVAR	Két adatsor kovarianciáját adja
CRITBINOM	KRITBINOM	A minőségvizsgálathoz kiszámítja, mekkora a megengedhető legnagyobb hibás termékszám
QUARTILE	KVARTILIS	Egy adathalmaz negyedszintjét számítja ki
LINEST	LIN.ILL	A legkisebb négyzetek módszerével az adatpárokra legjobban illeszkedő egyenes paramétereit határozza meg
LOGNORMDIST	LOG.ELOSZLÁS	A lognormális eloszlásfüggvény értékét adja vissza
LOGEST	LOG.ILL	Az adatokra legjobban illeszkedő exponenciális függvény paramétereit számítja ki
MAX	MAX	A legnagyobb értéket adja
MEDIAN	MEDIÁN	Az adathalmaz mediánját számítja ki
CONFIDENCE	MEGBÍZHATÓSÁG	Egy statisztikai sor várható értékének a középérték mindkét oldalán azonos méretű megbízhatósági intervallumát adja eredményül
SLOPE	MEREDEKSÉG	Egy lineáris regressziós egyenes meredekségét számolja ki

Angol	Magyar	Leírás
GEOMEAN	MÉRTANI.KÖZÉP	A mértani közepet számolja ki
INTERCEPT	METSZ	A regressziós egyenes y-tengellyel való metszéspontját számolja ki
MIN	MIN	A legkisebb értéket számolja ki
MODE	MÓDUSZ	A statisztikai sor móduszát számolja ki
LARGE	NAGY	Egy adatsor sorbarendezés utáni k. legnagyobb elemét adja
NEGBINOMDIST	NEGBINOM.ELOSZL	A negatív binomiális eloszlás értékét adja
NORMDIST	NORM.ELOSZL	A normális eloszlást számolja ki
STANDARDIZE	NORMALIZÁLÁS	A normalizált értéket adja eredményül
GROWTH	NÖV	Becslést ad az exponenciális regresszióra
PEARSON	PEARSON	Két adatsor közötti lineáris kapcsolat szorosságának mértéke
PERCENTILE	PERCENTILIS	Egy adattartomány százalékosztályát (percentilisét) adja eredményül
POISSON	POISSON	Poisson-eloszlást számol
TRIMMEAN	RÉSZÁTLAG	Egy adathalmaz középső részének átlagát adja eredményül
RSQ	RNÉGYZET	Pearson-féle szorzatmomentum korrelációs együtthatójának négyzetét adja eredményül
RANK	RANG.EGY	Egy szám rangját adja az adatsorban
DEVSQ	SQ	A középértéktől mért eltérés négyzetösszegét számítja ki
STEYX	STHIBAYX	A regresszióanalizísben a standard hibát számítja ki
NORMSDIST	STNORMELOSZL	A standard normális eloszlás
PERCENTRANK	SZÁZALÉKRANG	Egy adathalmazon belüli érték rangját számítja ki
STDEVP	SZÓRÁSP	Szórást számol
STDEV	SZÓRÁS	Korrigált empirikus szórás
TDIST	T.ELOSZLÁS	Student-féle t-eloszlás
TTEST	T.PRÓBA	Student-féle t-próba
TREND	TREND	Lineáris trend
PROB	VALÓSZÍNŰSÉG	Geometriai valószínűség a és b intervallumon
PERMUT	VARIÁCIÓK	Variációk

Angol	Magyar	Leírás
VARP	VARP	Varianciát (szórásnégyzetet) számol
VAR	VAR	A varianciára ad becslést
WEIBULL	WEIBULL	Weibull-féle eloszlás
ZTEST	Z.PRÓBA	Z-próba

Matematikai függvények

Angol	Magyar	Leírás
ABS	ABS	Szám abszolút értéke
ACOS	ARCCOS	Egy szám arkusz koszinuszát számítja ki
ASIN	ARCSIN	Egy szám arkusz szinuszát számítja ki
ATAN	ARCTAN	Egy szám arkusz tangensét számítja ki
CEILING	PLAFON	Felfelé kerekít egy számot egy adott szám legközelebbi többszörösére
COMBIN	KOMBINÁCIÓK	Adott számú elemcsoportra vonatkozó kombinációk számát számítja ki
COS	COS	Egy adott szög koszinuszát számítja ki
DEGREES	FOK	Radiánban kifejezett szögértéket fokra számít át
EVEN	PÁROS	A legközelebbi egészre kerekített számot adja meg
EXP	KITEVŐ	Eredménye az e szám-adik hatványa
FACT	FAKT	A megadott szám faktoriálisát számítja ki
GCD	GCD	Két szám legnagyobb közös osztóját adja vissza
INT	INT	Tizedes szám egész részét adja vissza
LN	LN	Egy szám természetes logaritmusát adja meg
LOG	LOG	Adott szám adott (vagy 10) alapú logaritmusát számítja ki
LOG10	LOG10	Adott szám 10-es alapú logaritmusát számítja ki
COUNTIF	DARABTELI	Megszámolja egy adott kritériumnak megfelelő cellák számát az adott szelekcióból
MDETERM	MDETERM	Tömb mátrix-determinánsát számítja ki
MINVERSE	INVERZ.MÁTRIX	Tömbben tárolt mátrix inverz mátrixát számolja ki
MMULT	MSZORZAT	Két tömb mátrix szorzatát számolja ki
MOD	MARADÉK	Két szám osztásából adódó maradékot adja eredményül
ODD	PÁRATLAN	Felfelé kerekít egy számot a legközelebbi páratlan egészre

Angol	Magyar	Leírás
PI	PI	A pi értékét adja vissza
POWER	HATVÁNY	Egy szám adott kitevőjű hatványát számítja ki
PRODUCT	SZORZAT	Összeszoroz minden argumentumként megadott számot
RADIANS	RADIÁN	Fokot radiánná alakít át
RAND	VÉL	Egyenletes eloszlású és véletlenszerű valós számot generál
ROMAN	RÓMAI	A megadott számot római számokkal kifejezve, szövegként adja vissza
ROUND	KEREKÍTÉS	Egy számot adott számú számjegyre kerekít
ROUNDDOWN	KEREKÍTÉS.LE	Lekerekítés
ROUNDUP	KEREKÍTÉS.FEL	Felkerekítés
SIGN	ELŐJEL	A megadott szám előjelét határozza meg, eredménye 1, ha pozitív, 0, ha negatív
SIN	SIN	Egy szög szinuszát számítja ki
SQRT	GYÖK	A megadott szám pozitív négyzetgyökét számítja ki
SUBTOTAL	RÉSZÖSSZEG	Listában vagy adatbázisban levő cellák részösszegét adja vissza
SUM	SZUM	Cellákat ad össze
SUMIF	SZUMHA	Megadott feltételt teljesítő cellák értékeit adja össze
SUMPRODUCT	SZORZATÖSSZEG	A megadott szelekciók vagy tömbök megfelelő elemeinek szorzatát, majd ezek összegét adja vissza
SUMSQ	NÉGYZETÖSSZEG	Paraméterei négyzetének összegét számítja ki
TAN	TAN	Megadott szög tangensét adja eredményül
TRANSPOSE	TRANSZPONÁLÁS	Függőleges cellatartományokat vízszintessé, vízszintes cellatartományokat függőlegessé alakít
TRUNC	CSONK	Adott szög tangensét adja eredményül

Logikai függvények

Angol	Magyar	Leírás
AND	ÉS	Két vagy több kifejezés ÉS logikai értékét határozza meg
IF	HA	A megadott feltétel igazságértéke szerint adja vissza a második vagy harmadik paraméter értékét
NOT	NEM	Logikai érték tagadása

Angol	Magyar	Leírás
OR	VAGY	Két vagy kifejezés VAGY logikai értékét határozza meg
FALSE	HAMIS	Hamis logikai érték
TRUE	IGAZ	Igaz logikai érték

Dátumfüggvények

Angol	Magyar	Leírás
DATEVALUE	DÁTUMÉRTÉK	Szövegként tárolt dátumot dátumértékké alakít át
DATEDIF	DÁTUMTÓLIG	Két dátum közé eső napok, hónapok vagy évek számát számítja ki
DATE	DÁTUM	Év, hónap, nap értékekből dátumot készít
YEAR	ÉV	Megadott dátumból visszaadja az évet
WEEKDAY	HÉT.NAPJA	Megadott dátumból visszaadja a hét napját
MONTH	HÓNAP	Megadott dátumból visszaadja a hónapot
TIMEVALUE	IDŐÉRTÉK	Szövegként tárolt időt időértékké alakít át
TIME	IDŐ	Megadott óra, perc, másodperc értékekből időértéket készít
TODAY	MA	Mai dátumot adja vissza
NOW	MOST	Az aktuális időértéket adja vissza (dátum és idő)
SECOND	MPERC	Megadott időértékből a másodpercet adja vissza
DAY	NAP	Egy adott dátumból a napot adja vissza
HOUR	ÓRA	Megadott időértékből az órát adja vissza
MINUTE	PERC	Megadott időértékből a percet adja vissza
WORKDAY	KALK.MUNKANAP	Egy dátum megadott számú munkanappal megelőző vagy követő dátum értékét adja vissza
YEARFRAC	TÖRTÉV	Két dátum közötti teljes napok számát számítja ki törtévként

Szövegfüggvények

Angol	Magyar	Leírás
CHAR	KARAKTER	A szám argumentummal megadott kódú karaktert adja eredményül
CLEAN	TISZTÍT	Eltávolít minden nem nyomtatható karaktert a szövegből
CONCATENATE	ÖSSZEFŰZ	Két vagy több szöveget fűz össze

Angol	Magyar	Leírás
DOLLAR	FORINT	Megadott értéket szöveges pénznemformátumban alakít át
EXACT	AZONOS	Két szöveg egyenlőségét vizsgálja
FIND	SZÖVEG.TALÁL	Szövegben másik szöveget keres
FIXED	FIX	Egy számot kerekít a tizedeshelyek paraméterrel megadott számú tizedesre, majd ezresenként tagolja, elhelyezi a tizedesvesszőt, és az eredményt karakterláncként adja vissza
LEFT	BAL	Karaktereket ad vissza egy szöveg bal oldaláról
LEN	HOSSZ	Szöveg hosszát adja vissza
LOWER	KISBETŰ	Szöveg karaktereit kisbetűvé alakít
MID	KÖZÉP	Karaktereket ad vissza egy szöveg megadott indexétől
PROPER	TNÉV	Egy szöveg első betűjét, valamint a nem betű után álló betűket nagybetűsre változtatja
REPLACE	CSERE	Szövegben cserét végez
REPT	SOKSZOR	Megadott alkalommal megismétel egy szövegrészt
RIGHT	JOBB	Karaktereket ad vissza egy szöveg jobb oldaláról
SEARCH	SZÖVEG.KERES	Szövegben másik szöveget keres
SUBSTITUTE	HELYETTE	Szövegben cserét végez
TEXT	SZÖVEG	Számok megjelenítésének módosítását végzi el a megadott feltételek szerint
TRIM	KIMETSZ	Minden szóközt töröl a megadott szöveg elejéről, illetve végéről
UPPER	NAGYBETŰS	Szöveg karaktereit nagybetűvé alakítja
VALUE	ÉRTÉK	Megadott szöveget számmá alakítja

Keresőfüggvények

Angol	Magyar	Leírás
ADDRESS	CÍM	Cellacímet képez a megadott oszlop, sor és esetleges munkalap nevéből
AREAS	TERÜLET	Egy hivatkozásban található területek számát adja eredményül
CHOOSE	VÁLASZT	Az érték argumentumok közül az index sorszámút adja vissza
COLUMNS	OSZLOPOK	A hivatkozásban vagy egy tömbben lévő oszlopok számát adja eredményül

Angol	Magyar	Leírás
INDEX	INDEX	A megadott táblázatból vagy egy tartományból egy értéket vagy egy értékre mutató hivatkozást ad vissza
INDIRECT	INDIREKT	Eredménye a szövegként megadott hivatkozás
HLOOKUP	VKERES	Vízszintes keresés
LOOKUP	KERES	Egyetlen sorban vagy oszlopban való keresés
MATCH	HOL.VAN	Egy cellatartományban tartomány adott elemet, majd az elem tartományon belül való relatív pozícióját adja eredményül
OFFSET	ELTOLÁS	Egy megadott magasságú és szélességű hivatkozást ad eredményül egy másik hivatkozástól számított megadott számú sornyi és oszlopnyi távolságra
ROWS	OSZLOP	A hivatkozásban vagy tömbben található sorok számát adja meg
VLOOKUP	FKERES	Függőleges keresés

IRODALOM

ANDRÁS Szilárd–BARICZ Árpád
2007 *Statisztika közgazdászoknak*. Csíkszereda, Státus Kiadó.
BIJI, Mircea–BIJI, Maria Elena
2002 *Tratat de statistic*. Editura Economică.
CSEKE Vilmos
1982 *A valószínűségszámítás és gyakorlati alkalmazásai*. Kolozsvár, Dacia Könyvkiadó.
DENKINGER Géza
1999 *Valószínűségszámítás*. Budapest, Nemzeti Tankönyvkiadó.
DENKINGER Géza
1999 *Valószínűségszámítási gyakorlatok*. Budapest, Nemzeti Tankönyvkiadó.
EZEKIEL, M.–FOX, K. A.
1970 *Korreláció és regresszióanalízis*. Budapest, Közgazdasági és Jogi Kiadó.
HARNOS Zsolt–LADÁNYI Márta
2005 *Biometria agrártudományi alkalmazásokkal*. Aula Kiadó.
HUNYADI László–MUNDRUCZÓ György–VITA László
1999 *Statisztikai képletgyűjtemény*. Budapest, Aula Kiadó.
HUNYADI László–MUNDRUCZÓ György–VITA László
2001 *Statisztika*. Budapest, Aula Kiadó.
JUHÁSZ Györgyné
2001*Feladatmegoldások és tesztek a Statisztikai módszerek könyvhöz*. Budapest, Aula Kiadó.
KORPÁS Attiláné
1996, 1997 Általános statisztika I., II. Budapest, Nemzeti Tankönyvkiadó.
KRÖPFL, B.–PESCHEK, W.–SCHNEIDER, E.–SCHÖNLIEB, A.
2000 *Alkalmazott statisztika*. Budapest, Műszaki Könyvkiadó.
KERÉKGYÁRTÓ Györgyné–MUNDRUCZÓ György–SUGÁR András
2001 *Statisztikai módszerek és alkalmazásuk*. Budapest, Aula Kiadó.
KERÉKGYÁRTÓ Györgyné–MUNDRUCZÓ György–SUGÁR András
2001 *Statisztikai módszerek és alkalmazásuk a gazdasági, üzleti elemzésekben*. Budapest, Aula kiadó, Budapest.
KOVALCSIK Géza
2005 *Az Excel programozása*. Budapest, Computerbooks.
LUKÁCS Ottó
1987 *Matematikai statisztika példatár*. Budapest, Műszaki Könyvkiadó.
MIHOC, Ghe.–MICU, N.
1979 *Elemente de teoria probabilităţilor şi statistică matematică* (manual pentru clasa a XII-a). Bucureşti, Editura didactică şi pedagogică.

MOLNÁR Máténé–TÓTH Mártonné
2001 *Általános Statisztika Példatár I*. Budapest, Nemzeti Tankönyvkiadó.
MORONEY, M. J.
1970 *Számoktól a tényekig*. Budapest, Gondolat Kiadó.
MUNDRUCZÓ György
1981 *Alkalmazott regressziószámítás*. Budapest, Akadémiai Kiadó.
OBÁDOVICS J. Gyula
2001 *Valószínűségszámítás és matematikai statisztika*. Budapest, Scolar kiadó.
REIDMACHER, Heinz Peter
2000 *Excel közgazdászoknak*. Budapest, Aula Kiadó.
RÓTH Józsefné–SUGÁR András
2002 *Statisztika*. Budapest, Nemzeti Tankönyvkiadó.
SZARVAS Beatrix–SUGÁR András
1999 *Példatár a statisztika című könyvhöz*. Budapest, Aula kiadó.
ŢIŢAN, Emilia–GHIŢĂ, Simona–BĂCESCU-CĂRBUNARU, Angelica
2002 *Bazele statisticii*. Meteora Press.

REZUMAT

Excel pentru studenţi la economie şi ştiinţe inginereşti

Excel este unul dintre cele mai populare programe, poate chiar mai util şi mai familiar decât Word. Este relativ ieftin şi accesibil. Programul poate fi utilizat şi în viaţa de zi cu zi (cumpărături), gospodărie, dar şi pentru afaceri, educaţie sau calcule inginereşti. În gospodărie, poate fi folosit pentru a înregistra achiziţiile, cheltuielile şi veniturile, pentru a urmări diferite consumuri (de apă, gaz), pentru a urmări cheltuielile publice, sau a ţine evidenţa muncii zilnice, a orarului etc. Nu în ultimul rând, Excel se poate folosi în predarea unor elemente de matematică, ale elementelor de bază ale statisticii, fundamentelor financiare, metodologiei de cercetare, elementelor de bază ale gestionării bazelor de date, şi, totodată, a elementelor de bază ale programării cu ajutorul VBA. Prezentele note de curs au fost realizate pentru a justifica aceste afirmaţii.

În cadrul Facultăţii de Ştiinţe Economice, Socio-Umane şi Inginereşti din Miercurea Ciuc aceste note de curs constituie baza a trei discipline fundamentale: Bazele informaticii, Informatică şi Informatică aplicată II, şi totodată materialul de bază pentru cursul Informatică Economică, în cadrul căruia predăm în principal VBA şi aplicaţiile economice ale programului Excel.

Acest material este extrem de util şi elevilor de liceu, deoarece conţine cunoştinţe de bază despre Excel necesare pentru absolvire. Înţelegerea acestui material nu presupune aproape nicio cunoştinţă prealabilă, dacă cititorul nu doreşte să dobândească informaţii specifice în domeniul statisticii sau a metodologiilor de cercetare. Deoarece se adresează candidaţilor la economie şi ştiinţe inginereşti, este firesc că majoritatea exemplelor sunt legate de acest subiect.

În finalul materialului de faţă încercăm, de asemenea, să oferim câteva informaţii de programare de bază şi să prezentăm o aplicaţie grafică interesantă şi inovatoare. Această secţiune este scrisă în special pentru a ajuta studenţii în lucrări pentru comunicări ştiinţifice sau lucrări de licenţă.

ABSTRACT

Excel for Economist and Engineer Candidates

Excel is one of the most popular programs, perhaps even more useful and familiar than Word. It is relatively inexpensive and easily accessible. It can be used in everyday economics (shopping), household, business, education, and engineering calculations. In the household: it can be used to record our purchases, expenses and income, to track our water and gas consumption, to track public spending, to keep track of our daily work, schedule, etc. Last but not least, Excel can be used in teaching math elements, basic statistics, financial fundamentals, research methodology, basic database management, and also basic programming with VBA. The current course notes were made to justify these statements.

Within the Faculty of Economic, Socio-Human and Engineering Sciences in Miercurea Ciuc, this lecture note is the basis of three core disciplines: *Fundamentals of Informatics*, *Informatics and Applied Informatics II*, and also the base material for the course *Economic Informatics*, in which we mainly teach VBA and the economic applications of Excel.

We also respectfully recommend this material to high school students as it contains basic knowledge about Excel required for graduation. Understanding this material requires almost no prior knowledge unless the reader wants to gain knowledge in statistics or research methodology. At the same time, being intended for candidates in economics and engineering, it is natural that most examples are related to these topics.

At the end of this material, we also try to provide some basic programming knowledge and present an interesting and innovative graphics application. This section is written primarily to assist students in work for scientific communications or undergraduate papers.

A SZERZŐKRŐL

Filep Levente (Kolozsvár, 1982. április 2.) egyetemi tanársegéd. A kolozsvári Babeş–Bolyai Tudományegyetem Matematika és Informatika Karán 2007-ben végzett informatika szakon, majd 2008-ban ugyanitt, a mesteri diplomáját modellezés és szimulálás szakon szerezte meg. 2016-ig az IT-iparban tevékenykedett mint projektvezető és „Full Stack" webfejlesztő. 2016-tól visszatért a Babeş–Bolyai Tudományegyetemre mint PhD-hallgató. Ugyanettől az évtől a Sapientia EMTE Csíkszeredai Karának oktatójaként tevékenykedik, ahol számos, a diákoknak szánt, informatikával kapcsolatos tevékenység szervezését bonyolította le, valamint több nemzetközi konferencián is előadást tartott.

Garda-Mátyás Edit (Székelyudvarhely, 1977. szeptember 20.) egyetemi tanársegéd. A kolozsvári Babeş–Bolyai Tudományegyetem Matematika és Informatika Karán végzett 2001-ben, informatika szakon. Mesteri diplomáját ugyanitt szerezte, számítógépes matematika szakon, 2004-ben. 2011-től a Debreceni Egyetem Matematika- és Számítástudományok Doktori Iskola doktoranduszhallgatója, PhD-fokozatát 2021-ben szerezte meg. Kutatási témája a Függvényegyenletek. 2001-től a Sapientia EMTE Csíkszeredai Karának oktatója előbb gyakornoki, majd 2008-tól tanársegédi munkakörben. Több éve tart közgazdászhallgatóknak Excel- és VBA-laborgyakorlatokat, illetve Alkalmazott informatika II. laborgyakorlatokat mérnökhallgatóknak. Számos kutatási programban és konferenciaszervezésben vett részt, nemzetközi tudományos konferenciákon tartott előadást, több tudományos cikknek a társszerzője vagy egyedüli szerzője.

Oláh-Gál Róbert (Marosvásárhely, 1958. június 13.) egyetemi adjunktus, matematikatörténész és Bolyai-kutató. A bukaresti egyetemen végzett matematika szakot 1982-ben. Három évig középiskolai tanár Csíkszeredában, majd ugyanott programozó a Területi Számítóközpontban 2000-ig. 1993-ban doktorált Debrecenben. 2000–2010 között a kolozsvári Babeş–Bolyai Tudományegyetem csíkszeredai kihelyezett informatika szakán adjunktus. 2010-től a Sapientia Erdélyi Magyar Tudományegyetem adjunktusa, a csíkszeredai helyszínen tanít matematikát és informatikát. Különböző hazai és külföldi szakmai társulatok tagja. Matematikai tárgyú és matematikatörténeti írásai, valamint Bolyai János életéről szóló tanulmányai hazai és tekintélyes külföldi lapokban jelentek meg. Könyvei: *Bolyai-breviárium*, Scientia Kiadó, Kolozsvár, 2012. ISBN 978-973-1970-73-8 (2. kiadás: 2015), *Az értől az óceánig: Réthy Mór (1846–1925) akadémikus élete és munkássága*, MATI Magyar Tudománytörténeti Intézet, Budapest, 2013, *Források az erdélyi magyar matematikai élet 1785–1918 közötti történetéhez*, Magyar Tudománytörténeti Intézet, Budapest, 2015.

Scientia Kiadó
400112 Kolozsvár (Cluj-Napoca)
Mátyás király (Matei Corvin) u. 4. sz.
Tel./fax: +40-364-401454
E-mail: scientia@kpi.sapientia.ro
www.scientiakiado.ro

Korrektúra:
Szenkovics Enikő

Műszaki szerkesztés:
Metaforma Kft.

Tipográfia:
Könczey Elemér

Nyomdai munkálatok:
F&F INTERNATIONAL Kft.
Felelős vezető: Ambrus Enikő igazgató

Az Excel az egyik legnépszerűbb szoftver, használata igen elterjedt. Alkalmas mérnöki számításokra, a segítségével jól lehet oktatni felsőbb matematikai ismereteket, a statisztika alapjait, a kutatásmódszertant, a pénzügyi adatbázis-kezelési és programozási alapokat, hasznos az üzleti életben és a háztartási bevételek/kiadások követésében egyaránt.

A jegyzet elsősorban a Sapientia EMTE közgazdász- és mérnökjelöltjei számára készült, így többségben ezekhez a szakterületekhez kapcsolható példák és gyakorlatok találhatók benne.

ISBN 978-606-975-053-7