Boros Norbert, Fehérvári Arnold, Fülep Dávid,
Kallós Gábor, Lovas Szilárd, Pukler Antal, Szörényi Miklós
SZE-MTK, Matematika és Számítástudomány Tanszék

Informatikai rendszerek alapjai

2013

Műszaki és természettudományos alapismeretek
tananyagainak fejlesztése a mérnökképzésben
Pályázati azonosító: TÁMOP-4.1.2.A/1-11/1-2011-0054

A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

IMPRESSZUM

Szerkesztette: Pukler Antal
Széchenyi István Egyetem, Műszaki Tudományi Kar, Matematika és Számítástudomány Tanszék
Lektor: Dr. Füvesi István, Szegedi Tudományegyetem, Informatikai tanszékcsoport

ISBN 978-963-7175-85-5
Kiadó: Széchenyi István Egyetem, Műszaki Tudományi Kar

Támogatás:
Készült a TÁMOP-4.1.2.A/1-11/1-2011-0054 számú, "Műszaki és természettudományos alapismeretek tananyagainak fejlesztése a mérnökképzésben" című projekt keretében.

Kulcsszavak: *adatábrázolás, hardver, operációs rendszerek, hálózatok, táblázatkezelés, szöveg- és kiadványszerkesztés*

Tartalmi összefoglaló: A tananyag első felében – a kapcsolódó tantárgyak célkitűzésének megfelelően – az informatikai eszközök gyakorlati használata során nélkülözhetetlen alapismereteket foglaltuk össze. Az itt érintett fontosabb témák a következők: adatábrázolás, számítógép-történelem, tömörítés, titkosítás, hardver ismeretek, operációs rendszerek, hálózatok. A tananyag második felében külön modulként szerepel az általános táblázatkezelés és a szöveg-, ill. kiadványszerkesztés bevezető szintű tárgyalása.

Technikai megjegyzések a jegyzet használatához.

Ez a tananyag egy *elektronikus jegyzet*.

2013-ban, a megjelenés évében annyira elterjedtek az elektronikus tartalomfogyasztásra alkalmas eszközök, hogy bátran feltételezhetjük: az egyetemisták túlnyomó többsége rendelkezik saját számítógéppel, tablet-géppel vagy elektronikus könyvolvasóval. A tananyag elektronikus formája sok előnnyel rendelkezik a nyomtatotthoz képest:

- Aktív tartalmak: az elektronikus változatban belső kereszthivatkozások, külső linkek, mozgóképek, stb. helyezhetők el. A tartalomjegyzék fejezetszámai, az egyenlet- és ábrasorszámok automatikusan belső linket jelentenek, így biztosítják a kényelmes és gyors belső hivatkozást, de a Szerző tetszőleges helyre tud akár a dokumentum belsejébe, akár egy külső webhelyre mutató linket elhelyezni, ami a szokásos klikkentéssel aktivizálható.
- Rugalmasság: a nyomtatott könyv statikus, míg az elektronikus jegyzet esetében könnyű hibajavításokat, frissítéseket alkalmazni.
- Erőforrás-takarékosság, környezetvédelem: az elektronikus formában való terjesztés sokkal kisebb terhelést jelent a környezetre, mint a nyomtatott. Különösen igaz ez, ha a tananyagban sok a színes ábra.

A használt fájlformátum: *PDF*.

A Portable Document Format az Adobe által kifejlesztett formátum, mely igen széles körben elterjedt. Sok helyről szerezhetünk be programot, mely a PDF fájok olvasására alkalmas. Ezek egy része azonban nem tartalmazza a teljes szabvány minden elemét, ezért speciális tartalmak nem, vagy nem pontosan jelenhetnek meg, ha nem az Adobe olvasóját, az AdobeReader-t használjuk. (Letölthető innen.)

A legtöbb megjelenítőprogram jól fogja kezelni az alapszöveget, ábrákat és linkeket, de gondok lehetnek a speciálisabb funkciókkal, pl. a beágyazott dokumentumok kezelésével, az aktív tesztek, kérdőívek használatával.

A jegyzet *képernyőn való megjelenítésre* lett optimalizálva.

A jelenlegi általánosan elérhető könyvolvasó hardverek mérete és felbontása kisebb, mint a nyomtatott könyveké és a számítógépek monitorai általában fektetett helyzetűek. Ehhez igazítottuk a formátumot arra optimalizálva, hogy fektetett kijelzőn teljes képernyős üzemmódban lehessen olvasni. Ehhez állítottuk be a karaktertípust és -méretet valamint azt is, hogy csak kis margót hagyunk, minél több pixelt biztosítva ezzel a tartalomnak. Azért, hogy teljes képernyős üzemmódban is lehessen navigálni, a margón kis navigáló-ikonokat helyeztünk el, melyek a megszokott módon kezelhetők:

- Lapozás előre és hátra: a függőleges oldalak közepén elhelyezett, nyújtott nyilakkal.
- Címoldalra ugrás: kis házikó szimbólum a bal felső sarokban.
- Vissza és előreugrás a dokumentumban: két kicsi szimbólum a bal felső részen. Ezek nem azonosak a lapozással, hanem a web-böngészők vissza- és előrelépéséhez hasonlóan a hiperlinkeken való navigálást szolgálják.

A jegyzet *segítséget nyújt a tanulás ütemezésében*.

A megtanulandó tanagyag a szokásos fejezet-alfejezet felosztáson túl leckékre való bontást is tartalmaz. A leckék különböző számú alfejezetből állhatnak, de közös bennük, hogy a Szerző megítélés szerint egy lecke „együltő helyben” megtanulható, azaz várhatóan 1–1,5 óra alatt feldolgozható.

A leckék elején rövid leírás található a tárgyalt témakörökről, a szükséges előismeretekről, a végén pedig önellenőrző kérdések, melyek sok esetben a PDF fájlban (AdobeReader-rel) aktív tartalomként jelennek meg feleletkiválasztós teszt, számszerű vagy képletszerű kérdés formájában. Érdemes tehát leckénként haladni a tanulásban, mert ez segít az ütemezés tervezésében illetve a leckevégi ellenőrzések segítenek annak eldöntésében, tovább szabad-e haladni vagy inkább ezt vagy az előző leckéket kell újra elővenni.

Ha a tananyag indokolja, nagyobb egységeket „modulokba” szervezünk és a modulok végén a leckevégi önellenőrzéshez képest komolyabb feladatblokkot találhatunk.

Tartalom

4. lecke

5. A számológép és a számítógép története

5. lecke

6. Számítógép-generációk

III. MODUL | Operációs rendszerek és számítógép-hálózatok

6. lecke

7. lecke

8. lecke

IV. MODUL | Kiadványszerkesztés

V. MODUL | Táblázatkezelés modul

12. lecke

1. Előszó

Ebben a jegyzetben azokat az ismereteket tárgyaljuk, amelyek a Széchenyi István Egyetemen a nem informatikus hallgatók alap-számítástechnikai és alapinformatikai oktatásában szerepelnek. A kialakításnál azt az elvet követtük, hogy minden hallgató – függetlenül attól, hogy konkrétan milyen szakon tanul, és mennyire mélyen kell megismerkednie az adott részterülettel – haszonnal tudja forgatni a jegyzetet, és megtalálja benne azokat az ismereteket is, amelyek – a rendelkezésre álló korlátozott időkeret miatt – az előadásokon és gyakorlatokon csak rövidebben kerülhetnek terítékre.

Az informatikai tárgyak oktatásának kísérőjelensége az állandó változás, ez a tankönyvíró szerző feladatát sem könnyíti meg. Az összeállításnál az vezérelt bennünket, hogy ezt a dinamikus – esetleg helyenként nehezen érthető – nagyobb képet egy magyarázó stílusban megírt, bármikor fellapozható gyűjteménnyel támogassuk meg.

Fontos célunk volt, hogy a tananyag alkalmas legyen az önálló feldolgozásra. Ezt több eszköz is segíti: modulokra és leckékre bontás, ellenőrző kérdések, önálló aktivitások (gyakorlati feladatok).

A felépítés során az általános felhasználó számára szükséges ismeretek klasszikus tárgyalási sorrendjét követtük (első fejezetek), emellett a tömörítés-titkosítás, a kiadványszerkesztés és a táblázatkezelés kapott helyet. Az egyes részek és szerzőik:

- A számolás története, számírási rendszerek, kódolás (Boros Norbert, Pukler Antal, Szörényi Miklós)
- Tömörítés, titkosítás (Boros Norbert, Kallós Gábor, Pukler Antal)
- Számítógép történelem, számítógép-generációk, a PC felépítése (Boros Norbert, Lovas Szilárd, Pukler Antal)
- Operációs rendszerek, hálózatok (Fülep Dávid)
- Kiadványszerkesztés (Fehérvári Arnold)
- Táblázatkezelés (Boros Norbert, Kallós Gábor)

A szerkesztés Pukler Antal gondos munkája.

Az anyag elsajátítása akkor tekinthető sikeresnek, ha a hallgató képessé válik a jegyzetünkben kitűzött feladatok megoldására is. Ennek az állapotnak az elérése természetesen függ a korábbi egyéni felkészültségtől, a tanulási sebességtől, de több-kevesebb idő ráfordításával mindenki eredményes lehet.

Reméljük ugyanakkor, hogy a jegyzetet a zárthelyikre és a vizsgára való felkészülésen túl is eredményesen használják majd a hallgatóink.

Köszönetnyilvánítás: A szerkesztő és a szerzők köszönetüket fejezik ki a Széchenyi István Egyetem munkatársainak, név szerint Bauer Péternek, Csábi Bélának, Hatwágner Miklósnak, Keresztes Péternek, Környei Lászlónak, Pusztai Pálnak, Takács Gábornak, Varjasi Norbertnek, valamint volt hallgatónknak Balics Ákosnak a munka során nyújtott értékes segítségükért. Ugyancsak köszönet illeti Füvesi Istvánt az alapos és segítő lektori munkájáért.

Győr, 2012. november

A Szerkesztő és a Szerzők

I. MODUL

A számolás története és a kódolás

Kulcsszavak: számrendszer, adatkódolás, titkosítás, tömörítés.

1. LECKE

A számolás kezdetei, számok leírása

Az első leckében megismerkedhetünk a számolás kialakulásának egy lehetséges változatával. A fejezetnek ez a néhány sora olvasmány gyanánt ajánlott az érdeklődőknek.

A számok leírására használt módszerek közül a római számmegadás szintén olvasmány, a hindu-arab módszer viszont már mindenkinek szóló anyag. Itt összefoglalva megtaláljuk a középiskolában már megismert számrendszereket, kiegészítve újabb ismeretekkel is.

Ilyenek például a tetszőleges alapú számrendszerek ismertetése, átváltás a számrendszerek között és a Boole-algebra alapjainak ismertetése.

2. Történeti áttekintés, a számolás története

Arra a kérdésre, hogy mikor és hogyan alakult ki az emberiség történetében a számolás, nehéz egyértelmű és pontos választ adni. A történészek a számolással kapcsolatos őskori leletek alapján a kezdeteket a beszéd kialakulásának idejére teszik. Ahogy a kőkorszakban (Kr. e. 500 000 – Kr. e. 10 000) a beszéd megjelent az emberiség történetében úgy jelent meg a számolás is. Természetesen nem a mai módszerekkel számolt ősünk. Nem voltak hatékony számolást segítő eszközei, nem tudott esetleg írni sem, sőt még az írást sem ismerte. legfeljebb az ujjait vagy a környezetében fellelhető apró tárgyakat használhatta a számolásra.

Hogy pontosan hogyan is jelent meg a mennyiségek kifejezéséhez a számfogalom, nem tudjuk. Kialakulásának csak közvetett bizonyítékait ismerik az őstörténettel foglalkozó történészek, és ezek magyarázatára is többféle elmélet létezik, így közöttük is vitatéma a számolás kifejlődésének módja.

A kezdetekben a mennyiségek megadására talán a mai egy, kettő, sok, később a kevés, majd a semmi (=0) szavaknak megfelelő szavak szolgáltak, és hosszú évszázados (évezredes) fejlődés következményeképpen alakultak ki a ma ismert számnevek közül a kisebb mennyiségek nevei. Valószínűnek látszik, hogy az egész számokkal párhuzamosan jelentek meg a törtszámok, hiszen a részekre osztás a mindennapok természetes rendjében számtalanszor előfordult, az így kialakult részmennyiségek megjelenése, megnevezése elkerülhetetlenül hozzá tartozott a számolás fejlődéséhez. Az is természetesnek tűnik, hogy kezdetekben a számokat nem önállóan, hanem valaminek a mennyiségét, nagyságát kifejezve használták. Az absztrakt számfogalom, azaz a számok önálló élete csak a számolás fejlődésének későbbi szakaszán alakult ki. Régészeti leletek alapján ez az időszak a Kr. e. 20 000 környékére tehető.

A fejlődés során egy-egy szám kitüntetett szerephez jutott, és erre épült fel az egész számrendszer. Ezt a domináns számot a számrendszer alapjának tekintjük. Így beszélhetünk kettes, hármas, ... tízes, ... tizenhatos, sőt akár hatvanas számrendszerről is. Gyakran előfordult az is, hogy az egyes számrendszerek keveredtek egymással, ami akár a fejlődés nem mindig szisztematikus voltából vagy akár a különböző kultúrák egymásra hatásából következhetett. A Föld különböző területein kialakult civilizációk számrendszereit vizsgálva megállapíthatjuk, hogy a tizenkettesig bezárólag minden számrendszerre akadt példa. Természetesen a fejlődési folyamat sem időben sem fejlettségi fokban nem volt egyforma. Voltak területek, ahol az ott élő népek mai

mértékek szerint is nagyra értékelhető rendszerben számoltak, másutt alig jutottak túl a számolás kezdetein. Fejlett számolási technikákkal és rendszerekkel rendelkezett Eurázsiában a kínai, a hindu, a mezopotámiai, a görög, a római, az arab, Afrikában az egyiptomi, később az arab, Amerikában a maja kultúrkör, bár a történelem nem ugyanazon korában voltak meghatározó tényezői a Föld kultúrájának.

A számolás műveleti közül a kezdetekkor megjelenhetett az összeadás, az osztás, a kivonás, a szorzás (a négy alapművelet). Az ókori matematikában már tudtak hatványozni, nyoma van a gyökvonás, a logaritmus, sőt az integrálás (!) kezdeteinek is.

Hogy a fejlődés valahogyan a fentiek szerint történhetett, a régészeti leleteken kívül a mai népek nyelvében is fellelhető szavak támasztják alá. Ma az egész világon a tízes alapú rendszer a „hivatalos” számrendszer. De például az angol vagy a német nyelvben a tizenegy, a tizenkettő neve nem a tízes rendszer alapján képződik (eleven, elf, illetve twelve, zwölf), ami arra utal, hogy az angol (szász, normann) és a német (germán) népek természetes módon használták a tizenkettes számrendszert. Az orosz, a magyar nyelvek a tízes rendszer szerint képezik a tizenegy és a tizenkettő számneveket, de vannak nyomok mindkét nyelvben a tizenkettes rendszer ismeretére és esetleges használatára is. Az orosz (dgyuzsina), a magyar tucat ugyancsak a tizenkettő neve. A tizenkettes rendszer ismeretére és használatára utal az év tizenkét részre való osztása, a nap kétszer tizenkét órával való mérése, az óra ötször tizenkét percre, a perc ötször tizenkét másodpercre való felosztása is.

2.1. A számok leírása

Az írásbeliség megjelenése ugyanúgy, mint a beszéd többi szavát, a számok neveit is leírhatóvá tette. A számneveknek a többi szavakhoz hasonló megjelenítése viszont nem segítette a számolást vagy a számokkal való műveletvégzést, ezért a beszéd szavainak leírására használt módszer helyett olyan formalizmusok alakultak ki, amikben néhány számot önálló jellel láttak el, és ezekből a jelekből, különböző szabályok szerint építették fel a többit megadó jelsorozatot. Nem célunk ezeknek a módszereknek mindegyikét felsorolni, de kettővel – mivel ezeket ma is használják –, részletesebben foglalkozunk.

Az egyik a római számírás és -rendszer, a másik a hindu eredetű arab közvetítéssel Európába kerülő, úgynevezett arab számjegyeken alapuló helyiértékes rendszer.

2.1.1. A római számírás

Az ókori Róma által használt számleírás, amelyik néhány hatékony számolási módszert is támogatott, Európában egészen a 13. századig általánosan elterjedt forma volt. Manapság viszonylag ritkán, de még mindig használjuk. A római rendszer a tízes számrendszeren alapult, de az ötös, sőt a kettes (!) számrendszer elemeit is fellelhetjük benne. Mivel a nulla jelet (a középkorig a nullát nem is tartották számnak!) nem ismerte, valamint a törtek használata nehézkessé tenné a rendszer ismertetését, ezért csak a természetes számok leírásával és a velük való számolással foglalkozunk. A rendszer ismertetése során a műveletek közül is csak az összeadást használjuk, és az ezernél nagyobb nagyságrendű számokkal nem foglalkozunk.

A latin nyelvben önálló neve volt a számoknak egytől tízig, a száznak, az ezernek, a milliónak. A többi számot a tízes rendszer szerint szóösszetétellel vagy több szóból álló szókapcsolattal képezték. (Mint ahogyan a magyar nyelv is teszi.) A számleírás az összeadás-műveletre épült, az összeadandó mennyiségekhez a latin ábécé nagybetűi közül választottak jelet. A 2.1. táblázat tartalmazza az önálló betűvel jelölt számokat és betűjelüket. A többi szám jeleit ezekből a betűkből az összeadás segítségével úgy alakították ki, hogy a számot felbontották olyan összegre, amelynek tagjai csak a táblázatban szereplő számok lehettek úgy, hogy a különböző tízhatványok maximum négyszer, a különböző tízhatványok ötszörösei maximum egyszer szerepelhettek, és a tagok nem növekvő sorrendben követték egymást. Ezután a tagoknak megfelelő betűt az összeadás sorrendjében egymás után írták. Jegyezzük meg: a fenti összeg egyértelmű tagokra való bontását adja minden számnak, így a számhoz tartozó betűsorozat mindig ugyanaz, bárki és bármikor végzi is el az összeg megállapítását a megadott szabály szerint.

2.1. táblázat. *A római számok jelei*

Szám	1	5	10	50	100	500	1000
Tízhatványok segítségével felírva	10^0	$5 \cdot 10^0$	10^1	$5 \cdot 10^1$	10^2	$5 \cdot 10^2$	10^3
Jele	I	V	X	L	C	D	M

Nézzünk ezek után egy példát az elmondottak illusztrálására. Írjuk fel római számként a 2634-et! $2634=1000+1000+500+100+10+10+10+1+1+1+1$. A táblázat jeleit behelyettesítve kapjuk: MMDCXXXIIII. Természetesen a rómaiak, illetve a római számokat használó kultúrák számolni tudó polgárai a 2634 felbontását formalizálva nem tudták elvégezni, hiszen nem ismerték hozzá az eszközöket (arab-hindu számjegyek, helyiértékes számírás), de a római számrendszer szerint gondolkodva a számrendszerükben „természetes” módon tudták leírni a számokat. A fenti gondolatmenet a mi számokról alkotott elképzelésünk alapján adja meg a számok római számjegyekkel való leírásának módját. (Valójában nem más, mint egy módszer arra, hogy hogyan kell a világon általánosan használt számleírásból a római számformát előállítani.)

Megjegyzendő, hogy a római rendszernek egy másik változata is ismert. Ebben a változatban az összeg felírásának szabályrendszere más. Nevezetesen csak három egyforma különböző tízhatvány követheti egymást, a különböző tízhatvány ötszöröse maximum egyszer szerepelhet. Ha négy tízhatványra lenne szükség, akkor a negyedik tag helyett a megfelelő tízhatvány-ötszörös és tízhatvány kivonásával kell kialakítani az összeget. Az összegre (különbségre) való bontás ebben a rendszerben is egyértelmű.

A 2634 ebben a formában a következő lesz: $1000+1000+500+100+10+10+10+5–1$. A kivonásnak megfelelő felírás pedig a jelek fordított sorrendjével történik, azaz a szám római rendszerű alakja: MMDCXXXIV.

2.1.2. Az arab-hindu számírás, a tízes, helyiértékes számrendszer

A római számok nem helyiértékes rendszerben vannak felírva. Itt is igaz ugyan, hogy a nagyobb számot jelentő jelek a szám felírásakor megelőzik a kisebb értékűeket, de nem függ a képviselt érték attól, hogy a sorban hányadik helyen vannak. Például a C akkor is százat jelent, ha egyedül áll, és akkor is, ha vannak még utána más jelek is, mondjuk: CLI.

Hogyan is épülnek fel a helyiértékes számrendszerek? Ennek a rendszernek a kialakulásában fontos lépés volt azt észrevenni, hogy a nulla is szám, illetve a 0 is számjegy. Ez a felismerés a hindu számolás és számrendszerben már az ókorban megtörtént.

Európa csak a 13., 14. században kezdte használni a nullát, mégpedig a hindu számjegyek megismerését követően. Mivel ezeket a jegyeket (jeleket) arab közvetítéssel ismertük meg, ezért hívjuk őket még manapság is

arab számjegyeknek.

Az arab közvetítők egyik legjelentősebb képviselője Muhammad Al-Hvarizmi (780–845), akinek latin nyelven megjelenő művei nemcsak megismertették Európával a hindu számírást és a számjegyeket (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), de két művének címe alapján alakult ki az algebra és az algoritmus szavunk is. (Az algebra a matematika egyik tudományterülete, az algoritmus pedig egy-egy feladathoz javasolt szisztematikus számolási módszer, amely a feladatot egyértelműen, véges lépésben megoldja.) A számrendszer alapja a tíz volt, ezért tízes – idegen szóval decimális – számrendszernek hívjuk. A számírás pedig azon a tényen alapult, hogy minden nemnegatív valós szám felírható olyan tízhatványok összegeként, amelyben minden különböző tízhatvány maximum kilencszer szerepelhet. Ennek az összegnek a rövidített felírásából keletkezik a szám helyiértékes alakja. Nem kell mást tenni, csak az összegben szereplő tízhatványok együtthatóját jelentő számjegyeket a kitevők nagyság szerinti sorrendjében leírni, amelyik kitevő nem szerepel az összegben, annak az együtthatója nyilván nulla, így ennek a kitevőnek megfelelő helyre nulla kerül. A nulladik kitevő együtthatója után vesszőt teszünk, ezzel jelezve, hogy befejeződött az egészrész, és a törtrész következik.

2.1. ábra. *Muhammad Al-Hvarizmi*

Példa: Az 1948,4 valójában az $1 \cdot 1000 + 9 \cdot 100 + 4 \cdot 10 + 8 \cdot 1 + 4 \cdot 1/10 = 1 \cdot 10^3 + 9 \cdot 10^2 + 4 \cdot 10^1 + 8 \cdot 10^0 + 4 \cdot 10^{-1}$ összeg rövidítése. Ebben valóban maximum kilencszer szerepel minden előforduló tízhatvány.

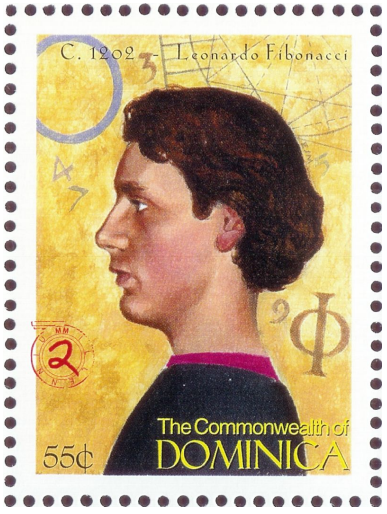
Általánosan, ha a számot a-val jelöljük, az előzőek képletben:

$a = a_{n-1}10^{n-1} + a_{n-2}10^{n-2} + \ldots + a_1 10^1 + a_0 10^0 + a_{-1}10^{-1} + \ldots + a_{-m}10^{-m} = \sum_{i=0}^{n-1} a_i 10^i + \sum_{i=1}^{m} a_{-i}10^{-i}$,

ahol az n az egészrész , m a törtrész számjegyeinek számát jelenti. És minden lehetséges i indexre a_i elem a {0,

1, 2, 3, 4, 5, 6, 7, 8, 9} halmaznak, azaz a_i valamelyik számjegy. Az összeg segítségével az a számot egyszerűen úgy lehet felírni, hogy az a_i együtthatókat az indexek szerint csökkenő sorrendben egymás után írjuk, az a_0 után tizedesvesszőt teszünk. Azaz $a = a_{n-1}a_{n-2}\dots a_1a_0{,}a_{-1}\dots a_{-m}$. A pozitivitást jelző előjelet (+) – ha akarjuk – a szám elé írhatjuk, de kiírni nem szükséges.

Ha a szám negatív, akkor igaz, hogy $a = -|a|$, de az $|a|$ már nemnegatív, így a hatványok összegére való bontás az előzőek szerint elvégezhető, és $a = -a_{n-1}a_{n-2}\dots a_1a_0{,}a_{-1}\dots a_{-m}$. Amint látjuk ezzel a módszerrel bármilyen valós szám tíz számjeggyel (plusz három kiegészítő jellel, az esetleges előjellel és a tizedesvesszővel) felírható. Az alapműveletek a rendszer segítségével könnyen automatizálhatóvá váltak, ezért elvégzésük könnyen tanulható lett. Segédeszközre az írótáblán és íróvesszőn (ma a papíron és ceruzán) kívül nem volt szükség. Ezt ismerte fel Leonardo Pisano (1170–1250), ismertebb néven Fibonacci, aki Al-Hvarizmi nyomán könyvet írt az arab-hindu számokról. A kortársak mégis idegenkedtek az új számírástól és számolástól, mivel azt tartották, hogy ezekkel a jelekkel írt feljegyzések könnyen hamisíthatók. Ami a kézzel való írás esetében igaz is lehetett, hiszen a számok végére utólag beírt nulla nagyságrendben változtathatta meg az értékét, és ugyancsak könnyen lehetett a nullát hatosra vagy kilencesre átrajzolni. Egy időre be is tiltották az arab-hindu számok használatát, ennek ellenére az új módszer hamarosan kiszorította a római rendszert.



2.2. ábra. *Leonardo Pisano*

Jegyezzük meg: a tízes számrendszerben a számok jeleinek helyiértékes felépítéséhez pontosan tíz számjegy kell.

2.1.3. Tetszőleges A-alapú számrendszerek

A 16., 17. században rájöttek arra is, hogy a helyiértékes felírási mód nemcsak a tízes, hanem tetszőleges számrendszer esetén is használható, csak a számjegyeket kell megfelelően megválasztani és megmutatni, hogy az adott számrendszerben is egyértelműen felírható minden szám a számrendszer alapjának hatványait megfelelően összegezve, és az összegben minden hatvány legfeljebb az alap értékénél eggyel kevesebbszer szerepelhet. Ez természetesen a tízes rendszerhez hasonlóan azt jelenti, hogy a számjegyek száma minden számrendszerben pontosan az alapszám értékével egyezik meg!

A számjegyek kiválasztása tíznél kisebb alap esetén nem okozott fejtörést, hiszen a tízes rendszer számjegyei közül a feleslegeseket, az éppen vizsgált alapnál nagyobb vagy egyenlő értékű jegyeket elhagyva, a maradék alkalmas lesz a számrendszer jegyeinek jelölésére. Ha az alap tíznél nagyobb, akkor a szükséges számjegyek száma is nagyobb, mint tíz. Tehát új számjegyekre van szükség. A mai gyakorlat szerint nem új jeleket szokás új számjegyként kreálni, hanem az latin ábécé elejéről választunk annyi nagybetűt, amennyit az alap értéke megszab.

2.3. ábra. *Gottfried Wilhelm Leibnitz*

Manapság a tízes számrendszer mellett használjuk a kettes (bináris) és a tizenhatos (hexadecimális) számrendszereket is.

A kettes helyiértékes számrendszer pontos, precíz leírását Gottfried Wilhelm Leibnitz készítette el az „Explication de l'Arithmétique Binaire" című könyvében.

A tízes rendszerhez hasonlóan igaz, hogy kettes rendszerben is minden nemnegatív valós szám felírható az alap, azaz a kettő hatványainak olyan összegeként, amelyben minden kettőhatvány maximum egyszer szerepel. Képlettel: tetszőleges nemnegatív a valós számra igaz, hogy

$$a = a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \ldots + a_1 2^1 + a_0 2^0 + a_{-1}2^{-1} + \ldots + a_{-m}2^{-m} = \sum_{i=0}^{n-1} a_i 2^i + \sum_{i=1}^{m} a_{-i}2^{-i},$$

ahol minden lehetséges i indexre a_i a 0 vagy az 1 számjegy valamelyike. Az a kettes számrendszerbeli alakja

pedig: $a = a_{n-1}a_{n-2}\ldots a_1a_0{,}a_{-1}\ldots a_{-m}$. Negatív szám esetén ugyanúgy járunk el, mint ahogyan a tízes számrendszernél láttuk.

A kettes számrendszer számjegyeit (a 0-t és az 1-et) az angol nevük (binary digit) rövidítéséből bitnek is szokás nevezni.

Tizenhatos számrendszert alkalmazva a tízes és a kettes rendszernél már leírtakat kapjuk, hogy minden nemnegatív valós a-ra

$$a = a_{n-1}16^{n-1} + a_{n-2}16^{n-2} + \ldots + a_1 2^1 + a_0 16^0 + a_{-1}16^{-1} + \ldots + a_{-m}16^{-m} = \sum_{i=0}^{n-1} a_i 16^i + \sum_{i=1}^{m} a_{-i}16^{-i},$$

ahol a_i minden lehetséges indexre a {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F} halmaz valamelyik eleme. Az A, B, C, D, E és F jelek tizenhatos rendszerben rendre a 10, 11, 12, 13, 14 és 15 tízes számrendszerbeli számnak megfelelő számot jelentik. Az a tizenhatos számrendszerbeli alakja pedig: $a = a_{n-1}a_{n-2}\ldots a_1a_0{,}a_{-1}\ldots a_{-m}$. Ha a szám negatív, akkor ugyanúgy járhatunk el, mint a tízes és kettes alapú rendszer esetében láttuk.

Mivel a számok számjegyekkel való helyiértékes leírásakor a kapott jelsorozatból nem lehet egyértelműen eldönteni, hogy milyen számrendszerben van ezért jelölni kell valamilyen módon a szám számjegyei mellett azt is, hogy melyik számrendszert használtuk a szám felírására. A számrendszer jelölésére többféle megoldás is használatos. Mi a következőkben a szám jegyei után alsóindexben és zárójelben a számrendszer alapját (tízes számrendszerben) megadva jelezzük a használt rendszert. (Pl.: $176_{(16)}$ [kimondva tizenhatos alapú százhetvenhat], $\mathrm{A1F}_{(16)}$ tizenhatos, $1011_{(2)}$, $10_{(2)}$ kettes, $10_{(10)}$, $176_{(10)}$ tízes számrendszerben megadott számok. Felhívjuk a figyelmet arra, hogy $10_{(2)}$ nem egyenlő $10_{(10)}$-zel és $176_{(16)}$ nem egyenlő $176_{(10)}$-tal.) Nem írjuk ki az alapot, ha számrendszer egyértelműen azonosítható.

A hétköznapi életben a tízes számrendszer a megszokott, így nem kell az alapot megadni, ha egy számot leírunk. A későbbiekben az alap nélkül leírt számokat mindig tízes számrendszerben megadott számnak tekintjük, és a tízes alapot csak akkor írjuk ki, ha hangsúlyozni szeretnénk azt, hogy a szám tízes rendszerben van leírva, megadva.

Aktivitás: A fentiek alapján gondoljuk át, hogy hogyan lehet leírni egy számot nyolcas (oktális) számrendszerben.

2.1.4. Számok átírása egyik számrendszerből másik számrendszerre

Ezek után felmerülhet az a kérdés, hogy hogyan kell – ha szükségessé válik – egyik számrendszerben leírt számot egy másik számrendszerbe átírni.

Mivel tetszőleges számrendszerből tízes számrendszerre való átírás az egyszerűbb, először ezzel foglalkozunk. Az átalakításhoz semmi más nem kell, mint egyszerűen felírni azt az összeget, amelynek rövidítéséből a szám jelsorozatát kaptuk. Azokat a számjegyeket, amelyek a tízes rendszerben nincsenek benne helyettesíteni kell az értékük tízes rendszerbeli alakjával. Az így kialakult összegben már minden tízes számrendszerben van felírva, ha elvégezzük a műveleteket az eredmény is tízes számrendszerbeli lesz, és ez éppen a szám tízes számrendszerben megadott alakját adja.

Példa: $\mathrm{A1F}_{(16)} \longrightarrow \mathrm{A}\cdot 16^2 + 1\cdot 16^1 + \mathrm{F}\cdot 16^0 \longrightarrow 10\cdot 16^2 + 1\cdot 16^1 + 15\cdot 16^0 = 10\cdot 256 + 16 + 15\cdot 1 = 2591_{(10)}$.

$1011_{(2)} \longrightarrow 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 8 + 2 + 1 = 11_{(10)}$.

$\mathrm{AA},8_{(16)} \longrightarrow \mathrm{A}\cdot 16^1 + \mathrm{A}\cdot 16^0 + 8\cdot 16^{-1} \longrightarrow 10\cdot 16 + 10\cdot 1 + 8\cdot 1/16 = 170{,}5$

Aktivitás: Adjuk meg a $8\mathrm{A3E1}_{(16)}$, $726_{(8)}$, $1010101{,}01_{(2)}$ számok tízes számrendszerbeli alakját!

Egy tetszőleges b valós szám tízes számrendszerből tetszőleges A-alapú rendszerben való felíráshoz nem kell mást tennünk, mint megkeresnünk A azon hatványainak összegét, amely (tízes számrendszerben) b-vel egyenlő. Azaz meg kell állapítani, hogy a

$$b = a_{n-1}A^{n-1} + a_{n-2}A^{n-2} + \ldots + a_1A^1 + a_0A^0 + a_{-1}A^{-1} + \ldots + a_{-m}A^{-m} = \sum_{i=0}^{n-1} a_iA^i + \sum_{i=1}^{m} a_{-i}A^{-i}$$

egyenlőség milyen n, m és a_i értékekre teljesül. Ezekről az értékekről csak annyit tudunk, hogy az összes $a_i < A$ és nemnegatív.

Vizsgáljuk először az összeg első felét. Azt, amelyben az a_i indexei nemnegatívok. Ez a részösszeg a b-nek A-alapú egészrészét adja meg, és látható, hogy az utolsó tag kivételével minden más tag osztható A-val. Mivel $a < A$, ezért az is nyilvánvaló, hogy az összeget – azaz b egészrészét – maradékos osztással A-val elosztva éppen a_0-t kapunk maradékul, a hányados pedig $a_{n-1}A^{n-2} + a_{n-2}A^{n-3} + \ldots + a_1A^0$ lesz. Ezt A-val újra elosztva

a_1-et kapunk maradékul. Ezeket a lépéseket az újabb és újabb hányadosra elvégezve rendre megkapjuk a többi pozitív indexű a_i-t. Az osztásokat akkor fejezzük be, ha a hányados nullává válik. Ez éppen az n-edik lépésben következik be, és ekkor az utolsó együtthatót, az a_{n-1}-et is megkapjuk. Így nyilvánvaló, hogy a maradékok rendre az a_0, ..., a_{n-1} értékeket adják, azaz a b-t előállító A számrendszerbeli alakot definiáló összeg első fele már felírható.

Az összeg második felét vizsgálva először azt kell megállapítanunk, hogy ez a részösszeg éppen b-nek A-alapú törtrészét adja meg. Vegyük észre, hogy ha ezt az összeget – azaz b törtrészét – A-val szorozzuk, akkor a kapott összeg egy olyan A-alapú számot határoz meg, amelynek egész része éppen a_{-1}, a törtrésze pedig az összeg, amely ugyancsak egy törtszám, csak a számjegyeinek száma az A-alapú alakban eggyel kevesebb, mint az előző törtrészé volt. Ezt a törtszámot A-val szorozva az egészrész a_{-2}, a törtrész egy újabb törtszám, amelynek A-alapú alakjában a_{-3}-tól a_{-m}-ig szerepelnek a számjegyek. A szorzásokat addig folytatva, amíg a törtrész nullává nem válik, megkaphatjuk a b A-alapú alakját megadó összeg összes negatív indexű a_i értékeit és m értékét is, ami nem lesz más, mint a nulla törtrész eléréséhez szükséges szorzások száma.

Sajnos előfordulhat, hogy a törtrész sohasem válik nullává. Ilyen esetben a b szám A-alapú alakja végtelen sok számjeggyel leírható törtrészből áll. Ekkor a szorzást addig kell folytatni, amíg elő nem kerül egy olyan törtrész, amelyik már a szorzások eredményeként megjelent. Ettől kezdve ugyanis periodikusan ismétlődni fog minden, a két egyforma törtrész közti szorzat, így a b A-alapú alakja periodikus végtelen törtrészű lesz.

Ezek alapján az átalakítást a következők szerint kell elvégezni:

1. Válasszuk a számot egészrészre és törtrészre.
2. Határozzuk meg az egészrész új számrendszerbeli alakját.
3. Határozzuk meg a törtrész új számrendszerbeli alakját.
4. Vesszővel elválasztva írjuk az egészrész mögé a törtrészt.

Példa: $1848{,}4_{(10)}$ bináris, oktális és hexadecimális átalakítása.

1. A egészrész: $1848_{(10)}$, a törtrész $0{,}4_{(10)}$
2. Az egészrész átalakítása. (Lásd a 2.2. táblázatot. Csak az osztási eredményeket tartalmazza a kiindulási egészrészt nem!)

2.2. táblázat. *Az egészrész átalakítása 2-es, 8-as és 16-os számrendszerbe*

	2		8		16	
index	hányados	maradék	hányados	maradék	hányados	maradék
0	$924_{(10)}$	0	$231_{(10)}$	0	$115_{(10)}$	8
1	$462_{(10)}$	0	$28_{(10)}$	7	$7_{(10)}$	3
2	$231_{(10)}$	0	$3_{(10)}$	4	$0_{(10)}$	7
3	$115_{(10)}$	1	$0_{(10)}$	3		
4	$57_{(10)}$	1				
5	$28_{(10)}$	1				
6	$14_{(10)}$	0				
7	$7_{(10)}$	0				
8	$3_{(10)}$	1				
9	$1_{(10)}$	1				
10	$0_{(10)}$	1				

Az egészrész alakjai a megadott számrendszerekben: $11100111000_{(2)}$, $3470_{(8)}$, $738_{(16)}$.

3. A törtrész átalakítása. (Lásd a 2.3. táblázatot. Csak a szorzási eredményeit tartalmazza az átalakítandó szám törtrészét, amit az első szorzáshoz használunk, nem!)

2.3. táblázat. *A törtrész átalakítása 2-es, 8-as és 16-os számrendszerbe*

	2		8		16	
	szorzat		szorzat		szorzat	
index	egészrész	törtrész	egészrész	törtrész	egészrész	törtrész
-1	0	$,8_{(10)}$	3	$,2_{(10)}$	6	$,4_{(10)}$
-2	1	$,6_{(10)}$	1	$,6_{(10)}$	innen ismétlődik	
-3	1	$,2_{(10)}$	4	$,8_{(10)}$		
-4	0	$,4_{(10)}$	6	$,4_{(10)}$		
	innen ismétlődik		innen ismétlődik			

A törtrész új alakjai: $0,\dot{0}11\dot{0}_{(2)}$, $0,\dot{3}14\dot{6}_{(8)}$, $0,\dot{6}_{(16)}$. A számjegyek feletti pontok a periodikusan ismétlődő szakaszt jelölik ki.

4. Az új teljes alakok: $11100111000,\dot{0}11\dot{0}_{(2)}$, $3470,\dot{3}14\dot{6}_{(8)}$, $738,\dot{6}_{(16)}$.

Aktivitás: Számítsuk ki, mekkora hibát vétünk, ha az átalakításkor keletkező végtelen tört jegyeit egy adott $-i$ kitevőtől elhagyjuk!

Valamely valós szám tetszőleges A alapról tetszőleges B alapra való átalakítása, ha sem A sem B nem tíz, két menetben történhet. Először alakítsuk át a számot tízes alapú formára, majd ezt alakítsuk tovább B-alapú alakra.

Példa: Mi a kettes számrendszerbeli alakja a $738{,}\dot{6}_{(16)}$-nak?

1. Átalakítás tízes számrendszerre:
   $738{,}\dot{6}_{(16)} = 7 \cdot 16^2 + 3 \cdot 16^1 + 8 \cdot 16^0 + \sum_{i=1}^{\infty} 6 \cdot 16^{-i} = 1848_{(10)} + (6_{(10)}/16_{(10)})/(1_{(10)} - 1_{(10)}/16_{(10)}) = 1848_{(10)} + 2_{(10)}/5_{(10)} = 1848{,}4_{(10)}$
   A számolásban felhasználtuk azt, hogy a $\sum_{i=1}^{\infty} 6 \cdot 16^{-i}$ egy $a_1 = 6/16$ kezdőelemű $q = 1/16$ kvóciensű mértani sor, amelynek értéke az $a_1/(1-q)$ képlettel számolható.
2. A tízes számrendszerbeli alak továbbalakítása kettes számrendszerre (lásd a 2.4. táblázatot).

2.4. táblázat. *Az* $1848{,}4_{(10)}$ *átalakítása 2-es számrendszerbe*

egészrész osztása			törtrész szorzása		
index	hányados	maradék	egészrész	törtrész	index
0	$924_{(10)}$	0	0	$,8_{(10)}$	-1
1	$462_{(10)}$	0	1	$,6_{(10)}$	-2
2	$231_{(10)}$	0	1	$,2_{(10)}$	-3
3	$115_{(10)}$	1	0	$,4_{(10)}$	-4
4	$57_{(10)}$	1	innen ismétlődik		
5	$28_{(10)}$	1			
6	$14_{(10)}$	0			
7	$7_{(10)}$	0			
8	$3_{(10)}$	1			
9	$1_{(10)}$	1			
10	$0_{(10)}$	1			

A kettes számrendszerbeli alak: $11100111000{,}\dot{0}11\dot{0}_{(2)}$.

Aktivitás: Mi az alakja nyolcas számrendszerben a $1010101{,}01_{(2)}$ és a $8A3E1_{(16)}$ számoknak?

Természetesen a számokkal minden számrendszerben lehet műveleteket végezni. A négy alapművelet elvégzése nem is olyan nehéz, ha felismerjük, hogy ugyanolyan szabályok érvényesek, mint amiket a tízes számrendszerben megismertünk. A problémát legfeljebb ezeknek a szabályoknak az adaptálása okozza. Nehezebb a bonyolultabb műveletek hozzáigazítása tetszőleges, de nem tízes alapú számrendszerhez.

„Nagy" számokkal, segédeszköz nélkül, még a tízes rendszerben is problémát okozhat a négy alapművelet elvégzése. Manapság – a különféle számoló eszközök korában – a segédeszközök mindenki számára elérhetők. Sőt, a négy alapművelet mellett lehet hatványozni, gyököt vonni, és néhány egyszerű függvény értékét is ki tudjuk számítani ezekkel az eszközökkel. Némelyik nemcsak tízes számrendszerben tud számolni, hanem kettes vagy tizenhatos rendszerben is.

2.1.5. Műveletek nem csak számokkal, a Boole-algebra

A 19. század közepétől a számokkal való műveletvégzés mellett megjelentek más objektumokon végezhető műveletek is, és kialakult az absztrakt algebra.

Számítástechnikai szempontból ennek a tudományágnak legfontosabb területe a Boole-algebra, amit George Boole munkássága alapozott meg.

A Boole-algebra egy olyan struktúra, amely egy kételemű halmazból és a rajta elvégezhető műveletekből épül fel. A halmaz egyik elemét I(gaz), másik elemét H(amis) értéknek tekintjük. Ezeken az értékeken maximum 4 egyoperandusú és 16 kétoperandusú művelet definiálható. Mi ezek közül egy egyoperandusú műveletet és három kétoperandusú művelet fogunk megismerni.

A tárgyalandó egyoperandusú műveletet negációnak nevezzük. A műveleti jele $\neg$ jel legyen. A művelet elvégzése pedig a következő: ha az operandus értéke I, akkor az eredmény legyen H, ha az operandus értéke

H, akkor az eredmény I. Formálisan: ¬I=H, ¬H=I. A művelet leírását úgynevezett igazságtábla segítségével is megadhatjuk.

operandus: A	eredmény: ¬A
I	H
H	I

A kétoperandusú műveletek egyikét „és”, a másikat „vagy”, a harmadikat „kizáró vagy” műveletnek nevezzük, a műveleti jelük: $\wedge$, $\vee$, valamint $\neq$. A műveletek igazságtáblája:

1. operandus: A	2. operandus: B	eredmény: A$\wedge$B
I	I	I
I	H	H
H	I	H
H	H	H

1. operandus: A	2. operandus: B	eredmény: A$\vee$B
I	I	I
I	H	I
H	I	I
H	H	H

1. operandus: A	2. operandus: B	eredmény: A≠B
I	I	H
I	H	I
H	I	I
H	H	H

A logikai műveletek számunkra a későbbiekben felhasználható fontosabb tulajdonságai:

Az „és” valamint a „vagy” művelet:

- asszociatív, azaz $(A \wedge B) \wedge C = A \wedge (B \wedge C) = A \wedge B \wedge C$ és $(A \vee B) \vee C = A \vee (B \vee C) = A \vee B \vee C$.
- kommutatív, azaz $A \wedge B = B \wedge A$ és $A \vee B = B \vee A$.

A „kizáró vagy” kommutatív azaz $(A \neq B) = (B \neq A)$, de nem asszociatív azaz $((A \neq B) \neq C)$ nem egyenlő $(A \neq (B \neq C))$-vel.

Az „és” művelet disztributív a „vagy” műveletre, illetve a „vagy” az „és”-re, azaz: $(A \vee B) \wedge C = (A \wedge C) \vee (B \wedge C)$, illetve $(A \wedge B) \vee C = (A \vee C) \wedge (B \vee C)$

Természetes ezek a legegyszerűbb műveleti tulajdonságok, többre nem is lesz szükségünk.

Önellenőrzés

1. Számítsa át tízes számrendszerre a következő számokat: $AF5_{(16)}$, $123_{(4)}$, $23{,}4_{(16)}$, $23{,}4_{(5)}$.
2. Számítsa át kettes számrendszerre a következő számokat: $1756{,}5_{(10)}$, $1756{,}5_{(16)}$, $1756{,}5_{(8)}$.
3. Bizonyítsuk be, hogy minden kettes számrendszerben felírható, véges sok nullát tartalmazó racionális szám tízes számrendszerben véges sok számjeggyel felírható. (Használjuk a mértani sorozatra megtanult összefüggéseket!)
4. Bizonyítsuk be, hogy a tízes és a tizenhatos számrendszerekben pontosan $n > 1$ darab számjeggyel felírható számok között van olyan pár, amelyiknek egyik tagja tízes, a másik tizenhatos számrendszerből való, és a nagyobbik nagyobb, mint a kisebbik 16-szorosa.
5. Jelölje meg az igaz állításokat!
   - ☐ Tízes számrendszerben leírt, véges sok tizedesjegyet tartalmazó szám kettes számrendszerben is véges sok számjegyet tartalmazó törtrészből áll.
   - ☐ Kettes számrendszerben leírt szám, amelynek törtrésze véges sok számjegyet tartalmaz, tízes számrendszerben is véges sok tizedesjegyű lesz.
   - ☐ Tízes számrendszerben véges sok számjeggyel leírt szám kettes számrendszerben is véges sok számjeggyel leírható.
   - ☐ Előfordulhat, hogy tízes számrendszerben véges sok számjeggyel leírt szám kettes számrendszerben végtelen sok számjeggyel lesz leírható.
   - ☐ Bármelyik egész szám kódolható valós számként is.

2. LECKE

Kódolás

Ebben a leckében megismerjük a kódolás fogalmát. Olyan kódrendszereket ismerünk meg amiket a számítástechnikában szokás alkalmazni a hétköznapokban használt kódrendszerek helyett.

Karakterek, jelek kódolásával kezdünk, és eljutunk az olyan összetett objektumok kódolásának egy lehetséges változatáig, mint amilyenek a képek. A képkódolás vázlatos leírása inkább csak olvasásra ajánlott, de nem érdektelen senki számára, hiszen olyan ismeretek találunk a leírásban, mint a digitalizálás, a felbontás és annak mértékegysége. Ezek az ismeretek a hétköznapi életben is fontosak lehetnek.

Sokak számára nem biztos, hogy nyilvánvaló miért is van szüksége a hétköznapi számítógép-használónak a kódolások ismeretére. A válasz nagyon egyszerű: már a táblázatkezelők használata során is előkerülnek olyan problémák, amiknek megoldásában nélkülözhetetlen a kódolás alapjainak ismerete. Nem kell tehát számítástechnikusnak lenni ahhoz, hogy a kódolás mibenlétét ismernünk kelljen.

3. Kódolás

Amint az előzőekből látszik, szinte a kezdetektől megjelentek azok a módszerek, amelyek segítségével a számok megjeleníthetők, később leírhatók voltak, azaz kialakult a számok kódolása.

3.1. definíció: Tágabb értelemben kódolásnak hívjuk azt a módszert, amely segítségével mások számára is elérhetővé, érthetővé tesszük gondolatainkat.

Ilyen kódolás a beszéd, az írás bármilyen formája, így a számok leírása is. Ilyen kódolás eredménye például a nyolcnak a szokásos tízes számrendszerben leírt $8_{(10)}$, a kettes számrendszerben megadott $1000_{(2)}$ vagy a római számú VIII alakja is.

3.2. definíció: Szűkebb értelemben kódolásról akkor beszélünk, ha szokásos eszközökkel (számjegyekkel, írással, képpel, videóval, hanggal) megadott objektumokat valamilyen egységes rendszerben újra megadunk, leírunk.

A továbbiakban a szűkebb értelmű kódolást értjük kódolás alatt.

Különösen fontossá vált a kódolás, amikor a számolás segítésére különböző eszközöket kezdett az emberiség használni. A következőkben a jelen eszközeinek használatához nélkülözhetetlen kódolásokkal fogunk foglalkozni. Ezeknek a kódolásoknak alapja a kettes számrendszer, mivel eszközeink elektronikus eszközök, és a kettes számrendszer két különböző bitje elektronikus eszközökkel könnyen megjeleníthető vagy könnyen továbbkódolható úgy például, hogy az 1-nek egy magas, míg a 0-nak egy alacsony feszültségszintet feleltetünk meg.

A különböző kódrendszereket vizsgálva megkülönböztethetünk fix hosszúságú és változó hosszúságú kódokból álló kódrendszert. A kódok hosszát a kódban lévő jelek számával szokás definiálni. A számítástechnikában mindkét rendszert használják.

3.1. A Boole-algebra objektumainak kódolása

A legegyszerűbb kódolás, hiszen csak két elemnek kell kódot választani (I, H). Legfeljebb az okozhat fejtörést, hány jelből álljon a kód, ha ragaszkodunk ahhoz, hogy a kódokat bitekből építjük fel. Egyetlen jelből álló kód esetén például az I objektumot 1-gyel, a H objektumot 0-val kódolhatjuk, 8 bites kód esetén az I kódja a 11111111 jelsorozat, a H-é a 00000000 jelsorozat lehet.

3.2. Betűk és egyéb jelek valamint tetszőleges szöveg kódolása

Fix hosszúságú kódolást használunk, a kód hossza általában 8 vagy 16 bináris jel, ritkábban előfordul a 24 és 32 kódhossz is.

Mivel 0 és 1 jelekből 8 hosszúságú kódot 256-féleképpen lehet előállítani, ezért ezzel a kódrendszerrel összesen 256 különböző jel kódolható. Ezeket a jeleket, illetve kódjaikat szabványok határozzák meg. A legelterjedtebb szabványok egyike az ASCII (American Standard Code for Information Interchange = amerikai szabványkód információcseréhez). Az első 128 kód sztenderd, azaz állandósult jeleket tartalmaz, a második 128 kód az úgynevezett kiterjesztett jelek kódjai.

3.1. ábra. *Az ASCII kódtáblázat első fele*

DEC	OKT	HEX	BIN	Jel	DEC	OKT	HEX	BIN	Jel	DEC	OKT	HEX	BIN	Jel	DEC	OKT	HEX	BIN	Jel
0	0	0	00000000	NUL	32	40	20	00100000		64	100	40	01000000	@	96	140	60	01100000	`
1	1	1	00000001	SOH	33	41	21	00100001	!	65	101	41	01000001	A	97	141	61	01100001	a
2	2	2	00000010	STX	34	42	22	00100010	"	66	102	42	01000010	B	98	142	62	01100010	b
3	3	3	00000011	ETX	35	43	23	00100011	#	67	103	43	01000011	C	99	143	63	01100011	c
4	4	4	00000100	EOT	36	44	24	00100100	$	68	104	44	01000100	D	100	144	64	01100100	d
5	5	5	00000101	ENQ	37	45	25	00100101	%	69	105	45	01000101	E	101	145	65	01100101	e
6	6	6	00000110	ACK	38	46	26	00100110	&	70	106	46	01000110	F	102	146	66	01100110	f
7	7	7	00000111	BEL	39	47	27	00100111	'	71	107	47	01000111	G	103	147	67	01100111	g
8	10	8	00001000	BS	40	50	28	00101000	(	72	110	48	01001000	H	104	150	68	01101000	h
9	11	9	00001001	HT	41	51	29	00101001	)	73	111	49	01001001	I	105	151	69	01101001	i
10	12	0A	00001010	LF	42	52	2A	00101010	*	74	112	4A	01001010	J	106	152	6A	01101010	j
11	13	0B	00001011	VT	43	53	2B	00101011	+	75	113	4B	01001011	K	107	153	6B	01101011	k
12	14	0C	00001100	FF	44	54	2C	00101100	,	76	114	4C	01001100	L	108	154	6C	01101100	l
13	15	0D	00001101	CR	45	55	2D	00101101	-	77	115	4D	01001101	M	109	155	6D	01101101	m
14	16	0E	00001110	SO	46	56	2E	00101110	,	78	116	4E	01001110	N	110	156	6E	01101110	n
15	17	0F	00001111	SI	47	57	2F	00101111	/	79	117	4F	01001111	O	111	157	6F	01101111	o
16	20	10	00010000	DLE	48	60	30	00110000	0	80	120	50	01010000	P	112	160	70	01110000	p
17	21	11	00010001	DC1	49	61	31	00110001	1	81	121	51	01010001	Q	113	161	71	01110001	q
18	22	12	00010010	DC2	50	62	32	00110010	2	82	122	52	01010010	R	114	162	72	01110010	r
19	23	13	00010011	DC3	51	63	33	00110011	3	83	123	53	01010011	S	115	163	73	01110011	s
20	24	14	00010100	DC4	52	64	34	00110100	4	84	124	54	01010100	T	116	164	74	01110100	t
21	25	15	00010101	NAK	53	65	35	00110101	5	85	125	55	01010101	U	117	165	75	01110101	u
22	26	16	00010110	SYN	54	66	36	00110110	6	86	126	56	01010110	V	118	166	76	01110110	v
23	27	17	00010111	ETB	55	67	37	00110111	7	87	127	57	01010111	W	119	167	77	01110111	w
24	30	18	00011000	CAN	56	70	38	00111000	8	88	130	58	01011000	X	120	170	78	01111000	x
25	31	19	00011001	EM	57	71	39	00111001	9	89	131	59	01011001	Y	121	171	79	01111001	y
26	32	1A	00011010	SUB	58	72	3A	00111010	:	90	132	5A	01011010	Z	122	172	7A	01111010	z
27	33	1B	00011011	ESC	59	73	3B	00111011	;	91	133	5B	01011011	[	123	173	7B	01111011	{
28	34	1C	00011100	FS	60	74	3C	00111100	<	92	134	5C	01011100	\	124	174	7C	01111100	|
29	35	1D	00011101	GS	61	75	3D	00111101	=	93	135	5D	01011101	]	125	175	7D	01111101	}
30	36	1E	00011110	RS	62	76	3E	00111110	>	94	136	5E	01011110	^	126	176	7E	01111110	~
31	37	1F	00011111	US	63	77	3F	00111111	?	95	137	5F	01011111	_	127	177	7F	01111111	

3.2. ábra. *Az ASCII kódtáblázat Latin1 jeleket tartalmazó második fele*

DEC	OKT	HEX	BIN	Jel	DEC	OKT	HEX	BIN	Jel	DEC	OKT	HEX	BIN	Jel	DEC	OKT	HEX	BIN	Jel
128	200	80	10000000	€	160	240	A0	10100000		192	300	C0	11000000	À	224	340	E0	11100000	à
129	201	81	10000001		161	241	A1	10100001	¡	193	301	C1	11000001	Á	225	341	E1	11100001	á
130	202	82	10000010	‚	162	242	A2	10100010	¢	194	302	C2	11000010	Â	226	342	E2	11100010	â
131	203	83	10000011	*ƒ*	163	243	A3	10100011	£	195	303	C3	11000011	Ã	227	343	E3	11100011	ã
132	204	84	10000100	„	164	244	A4	10100100	¤	196	304	C4	11000100	Ä	228	344	E4	11100100	ä
133	205	85	10000101	…	165	245	A5	10100101	¥	197	305	C5	11000101	Å	229	345	E5	11100101	å
134	206	86	10000110	†	166	246	A6	10100110	¦	198	306	C6	11000110	Æ	230	346	E6	11100110	æ
135	207	87	10000111	‡	167	247	A7	10100111	§	199	307	C7	11000111	Ç	231	347	E7	11100111	ç
136	210	88	10001000	ˆ	168	250	A8	10101000	¨	200	310	C8	11001000	È	232	350	E8	11101000	è
137	211	89	10001001	‰	169	251	A9	10101001	©	201	311	C9	11001001	É	233	351	E9	11101001	é
138	212	8A	10001010	Š	170	252	AA	10101010	ª	202	312	CA	11001010	Ê	234	352	EA	11101010	ê
139	213	8B	10001011	‹	171	253	AB	10101011	«	203	313	CB	11001011	Ë	235	353	EB	11101011	ë
140	214	8C	10001100	Œ	172	254	AC	10101100	¬	204	314	CC	11001100	Ì	236	354	EC	11101100	ì
141	215	8D	10001101		173	255	AD	10101101		205	315	CD	11001101	Í	237	355	ED	11101101	í
142	216	8E	10001110	Ž	174	256	AE	10101110	®	206	316	CE	11001110	Î	238	356	EE	11101110	î
143	217	8F	10001111		175	257	AF	10101111	¯	207	317	CF	11001111	Ï	239	357	EF	11101111	ï
144	220	90	10010000		176	260	B0	10110000	°	208	320	D0	11010000	Ð	240	360	F0	11110000	ð
145	221	91	10010001	‘	177	261	B1	10110001	±	209	321	D1	11010001	Ñ	241	361	F1	11110001	ñ
146	222	92	10010010	’	178	262	B2	10110010	²	210	322	D2	11010010	Ò	242	362	F2	11110010	ò
147	223	93	10010011	“	179	263	B3	10110011	³	211	323	D3	11010011	Ó	243	363	F3	11110011	ó
148	224	94	10010100	”	180	264	B4	10110100	´	212	324	D4	11010100	Ô	244	364	F4	11110100	ô
149	225	95	10010101	•	181	265	B5	10110101	µ	213	325	D5	11010101	Õ	245	365	F5	11110101	õ
150	226	96	10010110	–	182	266	B6	10110110	¶	214	326	D6	11010110	Ö	246	366	F6	11110110	ö
151	227	97	10010111	—	183	267	B7	10110111	·	215	327	D7	11010111	×	247	367	F7	11110111	÷
152	230	98	10011000	˜	184	270	B8	10111000	¸	216	330	D8	11011000	Ø	248	370	F8	11111000	ø
153	231	99	10011001	™	185	271	B9	10111001	¹	217	331	D9	11011001	Ù	249	371	F9	11111001	ù
154	232	9A	10011010	š	186	272	BA	10111010	º	218	332	DA	11011010	Ú	250	372	FA	11111010	ú
155	233	9B	10011011	›	187	273	BB	10111011	»	219	333	DB	11011011	Û	251	373	FB	11111011	û
156	234	9C	10011100	œ	188	274	BC	10111100	¼	220	334	DC	11011100	Ü	252	374	FC	11111100	ü
157	235	9D	10011101		189	275	BD	10111101	½	221	335	DD	11011101	Ý	253	375	FD	11111101	ý
158	236	9E	10011110	ž	190	276	BE	10111110	¾	222	336	DE	11011110	Þ	254	376	FE	11111110	þ
159	237	9F	10011111	Ÿ	191	277	BF	10111111	¿	223	337	DF	11011111	ß	255	377	FF	11111111	ÿ

A kiterjesztés többféleképpen is történhet, a bemutatott példa a Latin1 kiterjesztésű változat. A kódrendszer első felét a 3.1. ábra, a második részt a 3.2. ábra tartalmazza. A bináris kódok mellett megtaláljuk ezek decimális és hexadecimális értékét is. Az utóbbi két számban az értéket nem képviselő, az elől lévő nullákat elhagytuk. (A bináris kódban fontosak a kódok kezdő nullái is, mivel ezek a kód szerves részei. Ha elhagyhatók lennének, akkor a kód nem lenne fix hosszúságú!)

Látható, hogy ebben a rendszerben összefüggő tömböt alkotnak az angol ábécé nagybetűi és a kisbetűi, valamint a számjegyek. Megtalálhatók a jelek között az írásjelek, a gyakran használt matematikai jelek, sok ékezetes betű jele és néhány speciális jel is.

Az ASCII rendszerben nem kódolhatók a távolkeleti nyelvek betűi, sőt némely európai ábécé különleges betűje sem. Ennek a problémának a kiküszöbölésére alkották meg az Unicode, az UTF-8 rendszereket, Ezekben a kódok 16, 24, 32, 40, 48 hosszúságú 0 és 1 jelből álló sorozatok. Ezekben a rendszerekben 16 bittel 256^2, 24 bittel 256^3 különböző jelet lehet kódolni.

Ha jeleket tudunk kódolni, akkor már szöveget is. Hiszen nem kell mást tennünk, mint a szöveg betűit, számjegyeit, írásjeleit, betűközeit a jelek kódtáblázatát felhasználva kódoljuk, és a kódokat egymás mögé írva megkaphatjuk a kódolandó szöveg kódját. Vegyük észre, hogy a kapott kód hossza függ a jelek kódhosszától és a szöveg jeleinek számától is. Mivel a jelek száma más és más szövegben általában nem egyforma, ezért a szöveg ilyen kódolása változó hosszúságú kódot eredményez.

3.3. Számok kódolása

Természetesen az előző részben leírt kódolási módszer számok kódolására is jó, de az így kódolt számokkal a műveletvégzés nehézkes lesz, tehát alkalmasabb rendszert célszerű választani. Sokféle számkódolás lehetséges. A következőkben négyféle módszerrel fogunk foglalkozni.

3.3.1. Nemnegatív egész számok kódolása

Ezeket a számokat számítástechnikában előjeltelen (angolul unsigned) számoknak is szokás nevezni. Kódolásukra 8, 16, 32, 64 bites kódok a legelterjedtebbek. Nyolc bittel 0 és $2^8 - 1 = 255$, 16 bittel 0 és

$2^{16} - 1 = 65535$, 32 bittel 0 és $2^{32} - 1 = 4294967295$, 64 bittel 0 és $2^{64} - 1 = 255$ közé eső számokat szokás kódolni. A kód megállapítása nagyon egyszerű. Írjuk fel a kódolandó számot kettes számrendszerben, írjunk eléje annyi nullát, amennyi ahhoz kell, hogy a kód hossza megfelelő legyen. Amit kaptunk az a szám kódja! Fontos megjegyezni, hogy a fenti módszerrel a megadott intervallumokon kívüli egész számok nem kódolhatók!

3.3.2. Egész számok kódolása kettes komplemens kóddal

Ezeket a számokat számítástechnikában előjeles (angolul signed) számoknak szokás nevezni. Szintén 8, 16, 32, 64 bites kódokat használunk a kódoláshoz. Nyolc bittel a $-2^7 = -128$ és $2^7 - 1 = 127$, 16 bittel a $-2^{15} = -32768$ és $2^{15} - 1 = 32767$, 32 bittel a $-2^{31} = -2147483648$ és $2^{15} - 1 = 2147483647$, 64 bittel -2^{63} és $2^{63} - 1$ közé eső számokat kódoljuk. A megadott intervallumokon kívül eső számok ezzel a módszerrel nem kódolhatók!

A kódolás a következő (csak a nyolcbites változattal foglalkozunk, a másik három változatban ennek mintájára történik a kódok megállapítása). Ha a szám nemnegatív, akkor – ugyanúgy, mint az előző kódolási módszernél – vesszük a kettes számrendszerbeli alakot, eléje írunk annyi nullát, hogy nyolc jelből álljon, és már készen is van a kód. Ha a szám negatív, akkor hozzáadunk $2^8 = 256$-ot. Az eredmény 127-nél nagyobb, 256-nál kisebb pozitív szám lesz. Ennek vesszük a kettes számrendszerbeli alakját, és az lesz a negatív számunk kódja.

Vegyük észre, a nemnegatív szám kódja mindig nullával, a negatívé mindig eggyel kezdődik, ezért a kód vezető bitje megadja azt is, hogy kód negatív vagy nemnegatív számot jelent-e. Ezért ezt a bitet szokás előjelbitnek is nevezni. Ez az elnevezés bár helytelen, de annyira elterjedt, hogy itt is megemlítjük, de mindenképpen kihangsúlyozzuk, hogy nem az előjel kódolására használjuk!

A negatív szám kódját kettes komplemens kódnak is nevezik.

A kettes komplemens kód megállapítására egy kevesebb számolást igénylő módszert is lehet használni. A módszer azon alapul, hogy a 2^{n-1} szám kettes számrendszerben pontosan n darab egyesből áll, nulla nélkül. Ebből a számból könnyű kivonni bármilyen n-nél kevesebb bitből álló pozitív bináris számot, hiszen nem kell mást tenni, csak 0-t írni arra a helyiértékre, ahol a kivonandóban 1 van, és 1-et arra, ahol 0 van. Az eredmény nem más, mint az a szám, amit úgy kapunk, hogy a kivonandóban minden bitet az ellentettjére – 0-t 1-re, 1-et 0-ra – cserélünk, más szóval negáljuk, és annyi 1-gyel kiegészítjük a szám elején, hogy n jegyű legyen. Ez

a szám eggyel kisebb, mint a kivonandó -1-szeresének kettes komplemens kódja, ezért a kettes komplemens eléréséhez 1-et hozzá kell adni. Amit kaptunk, az nem más, mint a kivonandó (pozitív) szám -1-szeresének (ez már negatív) kódja.

Példa: Számítsuk ki a -1848 kettes komplemens kódját! A megoldás táblázatba foglalva:

1848	111 00111000
negálás	000 11000111
kiegészítés	11111000 11000111
$+1$	00000000 00000001
-1848 kódja	11111000 11001000

Érdemes megjegyezni, hogy a fenti módszert mechanikusan alkalmazva a kettes komplemens kódra, a kódhoz tartozó negatív szám abszolút értékének kettes számrendszerbeli alakját, azaz a kódját kapjuk. -1848 esetén:

-1848 kódja	11111000 11001000
negálás	00000111 00110111
kiegészítés	nem szükséges
$+1$	00000000 00000001
1848 kódja	00000111 00111000

Aktivitás: Számítsuk ki a -1848 32 bites komplemens kódját!

Ebben a kódrendszerben két szám összegének kódját rögtön megkaphatjuk, ha a kódokat összeadjuk, tehát nem kell az összeadás elvégzéséhez oda-vissza kódolgatni, a számok összeadása helyett elég a kódokat összeadni. Ezen kívül megspórolhatjuk a kivonást is, ugyanis $a - b = a + (-b)$ tetszőleges a és b esetén, az a és a $-b$

kódjának összege ebben a kódrendszerben éppen az $a - b$ szám kódja lesz. Ez pedig az előbb megismert mechanikus módszerrel könnyedén előállítható.

Jegyezzük meg, hogy ha két szám összege vagy különbsége nagyobb, mint a kódolható számok intervallumának végpontja vagy kisebb, mint a kezdőpontja, akkor a kódok összege nem állít elő kódot, hiszen a kérdéses összeg vagy különbség ezzel a módszerrel nem kódolható!

Példa: $1592 - 1848 = 1592 + (-1848) = -256$. Kódokkal:

1592	00000110 00111000
-1848 kódja	11111000 11001000
összeg	11111111 00000000

Aktivitás: Ellenőrizzük, hogy az eredmény valóban a -256 kódja-e!

3.3.3. Egész számok kódolása feszített vagy többletes kóddal

A komplemens kódhoz hasonlóan ennél a kódolásnál is 8, 16, 32 vagy 64 bites kódokat szokás használni. Ezzel a módszerrel kódolható számok 8 bit esetén $-2^7 + 1 = -127$ és $2^7 = 128$ közé 16 bittel $-2^{15} + 1 = -32767$ és $2^{15} = 32768$ közé, 32 bites kódnál $-2^{31} + 1 = -2147483647$ és $2^{31} = 2147483648$ közé, 64 bittel $-2^{63} + 1$ és 2^{63} közé eső számok lehetnek. Általánosan: n bites kóddal tetszőleges – n bittel kódolható – szám kódját a következőképpen kapjuk:

1. Számítsuk ki az $a + 2^{n-1} - 1$ számot. (A $2^{n-1} - 1$-et többletnek szokás nevezni.)
2. Alakítsuk át kettes számrendszerbe.
3. Ha szükséges tegyünk eléje annyi nullát, hogy a bitek száma pontosan n legyen.

A kapott jelsorozat lesz az a feszített vagy más névvel a többletes kódja.

3.3.4. Valós számok kódolása

Minden a valós szám (a 0-t kivéve) tetszőleges A számrendszerben egyértelműen felírható a következő módon. $a = mA^k$, ahol $|m|$ az $[1,A)$ intervallumba eső A-alapú valós szám. m neve mantissza. k szintén A-alapú egész szám, a karakterisztika, de szokás exponensnek is nevezni. Az $a = mA^k$ alakot az a szám A-alapú normál alakjának nevezzük. Ha $A = 2$, akkor a normál alak kettes számrendszerbeli és ebből az alakból kiindulva szokás a valós számok egyfajta kódolását felépíteni.

Nem kell ugyanis mást tenni, mint megállapítani a kód hosszát, majd kódolni kell az m előjelét, az $|m|$-et, és a k-t.

Az előjel kódolásához elég egy bit. Ezt a bitet már valóban nevezhetjük előjelbitnek. Legyen a kód 0, ha pozitív számot kódolunk, 1, ha negatív számot kódolunk.

$|m|$ kódja legyen az a bitsorozat, amit úgy kapunk, hogy az $|m|$-ből elhagyjuk az egészrész bitjét (ez a bit mindig 1) és a (kettedes) vesszőt. A megmaradó bitsorozatot a végén kiegészítjük nullákkal, ha nem elég hosszú, illetve a végét elhagyjuk, ha hosszabb, mint ahány bittel az $|m|$-et kódolnunk kell. Jegyezzük meg, hogy az kódja azt is meghatározza, hogy a kód mennyire pontosan adja vissza az eredeti számot, hiszen előfordulhat, hogy az $|m|$-ből el kell hagyni biteket, és emiatt a kódból az a-nak csak közelítő értékét kaphatjuk vissza.

A k kódolására a már megismert – k tervezett kódhosszának megfelelő – többletes kódot használjuk. Az így kialakított három kód egymásutánírásával kapjuk az a valós szám kódját. A sorrend általában a következő: az előjel kódja, a karakterisztika kódja, végül a mantissza kódja.

A pozitív és a negatív nulla megkülönböztethető, mivel a 0 kódja ebben a rendszerben kétféle lehet. A pozitív nulla kódja a csupa nullából álló, a kódhossznak megfelelő bitsorozat, a negatívé az előjelbiten egyet, máshol csupa nullát tartalmazó bitsorozat.

Ez a kódolás az úgynevezett lebegőpontos kódolási módszerek egyike, és ha a későbbiekben lebegőpontos kódról lesz szó, erre a kódolásra kell gondolni.

Azt, hogy a kód milyen hosszú legyen, és ezen belül hány bitet használhatunk a karakterisztika, és hányat a mantissza kódolására, szabványok határozzák meg.

Ilyen szabványok például az IEEE 754–1985, IEEE 854–1987, IEEE 1596.5–1993 szabványrendszerek. Ezek négy alaptípust definiálnak, leírásuk a 3.5. táblázatban található.

3.5. táblázat. *Valós számok szabványkódjainak neve és kódhossza*

Pontosság		Bitek száma			
Magyarul	Angolul	Hossz	Előjel	Kitevő	Mantissza
Egyszeres	Single float	32	1	8	23
Dupla	Double float	64	1	11	52
Kiterjesztett	Extended float	80	1	15	64
Négyszeres	Quadruple float	128	1	15	112

Mivel a későbbi tanulmányaink során olyan számítógépes rendszereket ismerünk meg, amelyekben a valós számokat a Double float szerint kódolták, így ezzel a változattal kicsit többet foglalkozunk.

Amit a kódhoz tartozó adatokból rögtön leolvashatunk: a karakterisztika kódolásánál az eltolás 1023, a mantisszából maximum 52 jelet tudunk felhasználni. A kódrendszernek – ugyancsak a számítógépes felhasználása miatt, a szabvány szerint – néhány kódját másképpen kell értelmezni, mint ahogyan a leírásból következne. Ezeket a különlegességeket foglalja össze a 3.6. táblázat.

3.3. definíció: A nem normalizált szám az a kettes számrendszerbeli szám, aminek alakja $m \cdot 2^{11111111110}$, ahol $0 < |m| < 1$, és a törtrész számjegyeinek száma legfeljebb 52.

3.6. táblázat. *A Double float különleges kódjai*

Értelmezés	Előjel kódja	Karakterisztikának megfelelő kód	Mantisszának megfelelő kód
$+0$	0	00000000000	Minden bit 0
-0	1	00000000000	Minden bit 0
+ végtelen	0	11111111111	Minden bit 0
– végtelen	1	11111111111	Minden bit 0
+ Nem szám kódja	0	11111111111	legalább egy nem nulla bit
– Nem szám kódja	1	11111111111	legalább egy nem nulla bit
Pozitív nem normalizált szám	0	00000000000	legalább egy nem nulla bit
Negatív nem normalizált szám	1	00000000000	legalább egy nem nulla bit

Példa: Állapítsuk meg a $-1848{,}4$ kódját az előző kódrendszer szerint!

A negatív előjel kódja 1, a karakterisztika kódja 1010 + 1111111111 = 10000001001, a mantisszáé 11001110000110011001100110011001100110011001100110011001. Ezeket egymás után írva kapjuk a kódot:

11000000100111001110000110011001100110011001100110011001100110011001.

Aktivitás: Állapítsuk meg, hogy a fenti kódból melyik valós számot lehet visszakódolni, és ez a szám mennyivel tér el a $-1848{,}4$-től!

Ezzel a módszerrel tárolt számok néhány nevezetes értékét és szabványos nevét tartalmazza a 3.7. táblázat:

3.7. táblázat. *Nevezetes értékek a Double float kódolásban*

Megnevezése	Leírása	Értéke 10-es számrendszerben
DBLMIN	a kódolható legkisebb szabványos pozitív szám	2.22507385850720138E–308
DBLMAX	a kódolható legnagyobb szabványos pozitív szám	1.79769313486231571E+308
nincs	a kódolható lehető legkisebb (nem szabvány szerinti) pozitív szám.	4.94065645841246544E–324
DBLEPSILON	kód-epszilon: az a legkisebb pozitív érték, amit 1-hez adva olyan számot kapunk, aminek kódja nem egyezik meg 1 kódjával.	2.2204460492503125E–16

A táblázat utolsó oszlopában szereplő E jelentése: az E előtti számot meg kell szorozni 10-nek azzal a hatványával, amit az E utáni szám határoz meg.

3.3.5. Összetett objektumok (pl. képek) kódolása

Nem célunk, hogy az összetett objektumok néha mélyebb ismereteket is igénylő kódolását és kódrendszereiket ismertessük, de annak megértéséhez, hogy egy nem számjegyekkel való tágabb értelemben kódolt objektum számokkal is kódolható, egy vázlatos példát adunk a kódoláshoz anélkül, hogy magát a kódrendszert ismertetnénk.

Legyen egy ilyen objektum egy kép. Első lépés a kódolásban hogy a képet diszkrét egységek rendszerére bontjuk. Ezek az egységek legyenek az úgynevezett képpontok (a képpont angolul pixel). Minden képpont egy-egy téglalap alakú képrész lesz. (A nyomdatechnikában hasonló felbontást alkalmaznak, itt a képpontok neve raszter, és nem mindig szabályos síkidom, sőt nem is mindig érintkeznek a raszterek egymással.) A felbontás annál jobb – és ennek következményeképpen a pontokból összerakott kép annál jobban hasonlít az eredeti képhez –, minél több pontból áll. Ezért szokás a felbontást egy adott hosszúságegységre (inch) eső pontok számával jellemezni. A mértékegység a pont per inch (angolul: Dot Per Inch), röviden DPI. Minél nagyobb a DPI érték, annál nagyobb és jobb a felbontás. Mivel a képek kétdimenziósak, és a vízszintes-függőleges irányú pontra bontás nem szükségképpen egyezik meg, ezért előfordulhat, hogy a vízszintes és a függőleges felbontást szorzat alakban adják meg, ahol az első tényező a vízszintes, a második a függőleges irányú egy inch-re eső pontszámot jelenti.

A kapott képpontok a képet egy téglalap alakú pontrendszerre bontják. Ennek a rendszernek a pontjait a méretük és a színük jellemzi. A méret a DPI értékből minden pontra egységesen megállapítható, a szín minden pontban más és más lehet.

A kép bitekből álló kódját tehát úgy lehet megadni, hogy megadjuk a bitkódját annak, hogy hány pontból áll a kép vízszintesen, hány pontból függőlegesen, megadjuk a bitkódját a vízszintes és függőleges pontméretnek, majd egyenként kódoljuk a pontok színét is. A bitkódokat egymás után írva egy 0-1 sorozatot kapunk, és ez lesz a kép kódja. Érdemes megemlíteni, hogy ezzel a kódolási mechanizmussal visszakódoláskor az eredeti képnek egy egyszerűbb változatát kapjuk. Ennek megértéséhez elég azt látni, hogy más és más felbontásra más és más kód alakul ki ugyanarról a képről, és a visszaalakítás után a képek csak „hasonlítani” fognak egymásra is és az eredeti képre is.

Azt a folyamatot, amely egy valós, folytonos objektumhoz diszkrét, legtöbbször bináris kódot állít elő, digitalizálásnak nevezzük.

Természetesen ez a leírás egy elnagyolt leírása az egyik lehetséges kódolásnak (legjobban az úgynevezett pixelgrafikus kódoláshoz hasonlít), de ahhoz, hogy a bonyolultabb objektumok bináris kódolását megértsük – elegendő.

A pixelgrafikus tárolással kapcsolatban célszerű vázlatosan megismerni az RGB modellt. Ebben a modellben minden színt három alapszín összetételével állítunk elő. Ezek a színek: vörös (Red), zöld (Green) kék (Blue). 16 bites színkód esetén a vörös összetevő kódolására 5, a zöldére ugyancsak 5, a kékre 6 bitet szokás használni. Ez 65536-féle szín kódolását teszi lehetővé, ami az emberi szem teljesítőképességét figyelembe véve egy hétköznapi kép kódolására bőségesen elegendő. Sok ember a képpontok színének 8 bittel (256 féle szín) való kódolásakor kapott kódból visszaállított képről sem tudja megmondani, hogy mi a különbség az eredeti és a visszaállított kép között.

A képpontokhoz tartozó direkt kódok mellett a kép kódjában általában még bizonyos kiegészítő vagy járulékos információt is kódolnak (pl. hibajavításra). Ez a kód végső hosszát, méretét csak minimális mértékben megnöveli.

A pixelgrafikus kódolást elsősorban fényképszerű képek kódolására használják. A vonalakból összerakott képek kódolására nem célszerű módszer. Az ilyen típusú ábrákat matematikai módszerek – vektorok – segítségével lényegesen rövidebb kóddal lehet leírni.

Önellenőrzés

1. Adja meg az ASCII kódját a Nemzeti dal első sorának.
2. Mi lesz a kódja a 100-nak, ha karaktersorozatként, nemnegatív egészként, ha egész számként és ha valós számként kódoljuk?
3. Mi lesz az egyszeres pontosságú kódja annak a számnak, amelyiknek a dupla pontos kódja: 1100000010011100111000011001100110011001100110011001100110011001? Útmutató: állapítsuk meg a kódhoz tartozó szám normál alakját, majd kódoljuk újra, most már egyszeres pontosságú kóddal.
4. Jelölje meg az igaz állításokat!
   - ☐ A Boole-algebra elemeit egy bittel is lehet kódolni.
   - ☐ Minden egész számként kódolható szám kódolható valós számként is.
   - ☐ Minden negatív szám kódolható egész számként.
   - ☐ Véges sok olyan valós szám van, amelynek kódja ugyanaz.
   - ☐ Nem minden valós számot lehet valós számként kódolni.
5. Van-e olyan szám, amelynek
   - ☐ Nemnegatív számként megállapított kódja ugyanaz, mint az egész számként megállapított kód?
   - ☐ Valós számként megállapított kódja megegyezik az egész számként megállapított kódjával?
   - ☐ Valamilyen kódja éppen az a betű ASCII kódjával egyenlő?

3. LECKE

Tömörítés, titkosítás

Ebben a leckében azzal foglalkozunk, hogyan lehet egy létező kódot rövidebbel (tömörítés), illetve mindenki számára nem visszakódolhatóval (titkosítás) helyettesíteni.

A tömörítéssel foglalkozó fejezetek viszonylag egyszerűbb matematikai apparátus ismeretével megérthetők. Néhány olyan fogalom, aminek ismeretével nem feltétlenül rendelkezik a hallgató, ha nem is teljes precizitással, de talán a megértéshez megfelelő mélységben s felhasználás helyén megtalálható.

A titkosítás már komolyabb matematikai ismeretek igényel. A modulhoz megadott irodalomban ezeknek tárgyalása megtalálható, a szövegben hivatkozunk ezekre.

Ez a lecke a nem műszaki szakos hallgatóinknak teles egészében olvasmányként szolgál. A műszaki szakok hallgatói számára a 4.2.6 fejezettől kezdődő rész is csak érdekességként ajánlott.

Érdeklődő olvasók a témák részletesebb tárgyalását az [1,4] tankönyvekben megtalálják.

4. Tömörítés, titkosítás

4.1. Tömörítés

A kódok tömörítésénél a legfőbb célunk az, hogy a kód rövidebb legyen, így kisebb helyet foglaljon el a kódot tartalmazó hordozón. A tényleges tömörítést az teszi lehetővé, hogy a kód – főleg ha hosszú – tartalmaz ismétlődő elemeket. Ezeket átkódolva, rövidebb kóddal helyettesítve rövidíteni lehet az eredeti kódot. Az átkódolás kétféle lehet. Visszaállítható kódolásról beszélünk, ha a kódból a kiindulás reprodukálható, adatvesztésesről, ha a kiindulás csak közelítőleg állítható vissza. Az egész számokra megismert kódolási módszerek például visszaállítható kódolások, a valós számokra megismert módszer nem.

A hétköznapi életben a tágabb értelemben vett kódolás eredményeit vizsgálva sok olyan jellegzetességet állapíthatunk meg, amely a tömörítésben felhasználható.

- Írott szövegben például egyes betűk előfordulása gyakoribb, mint másoké;
- Vonalas ábránál – ha képpontokra bontjuk az ábrát – a nagy összefüggő fehér terület sok egyforma egymást követő képpontból áll, ezért ezekkel a pontokkal leírt területek sokkal rövidebben kódolhatók, mintha minden pontot külön kódolnánk;
- Képpontokra bontott fényképnél az egymás melletti képpontok sokszor csak kismértékben vagy egyáltalán nem különböznek egymástól, (a különbözőség elhanyagolható, mivel visszakódolás után az új kép és a régi kép közti különbséget a szemünk nem veszi észre) ezért itt is rövidíthető a kód ahhoz képest, mintha minden képpontot külön kódolnánk;
- Filmek esetén az egymás utáni kockák sokszor csak kismértékben különböznek.

4.1.1. Tömörítő eljárások

A tömörítő eljárások megfelelő tömörítő módszerek alkalmazásával két fő műveletet végeznek:

- Átkódolás (becsomagolás) – az eredeti kódot „becsomagolják”, azaz megpróbálják a kódot rövidebb kóddal helyettesíteni, amit egy „kódburokkal” vesznek körbe. A kódburok azt írja le, hogy a tömörítés milyen módszerrel történt. (Általában az új kód – a burokkal együtt – kisebb méretű lesz);
- Dekódolás (kicsomagolás) – a tömörített, rövidített kódból visszanyerik az eredeti kódot. Ez a két fő funkció gyakran még kiegészül további tevékenységekkel, amelyek szerepe a biztonság növelése:
- Adatvédelem – a tömörítéshez jó hatásfokú hibajavító kódolást használnak, amely „folthibák” (kód egyes részeinek hibái) esetén is lehetővé teszi a visszakódolást;
- Titkosítás – a tömörített kódból csak egy kulcs (jelszó) megadása után fejthető vissza az eredeti kód. Már az eddig leírtakból is kikövetkeztethetjük, hogy kétféle tömörítési, kódolási módszert különböztetünk meg.
- Veszteségmentes tömörítés esetén visszakódolás (dekódolás) után teljesen az eredeti kódot nyerjük vissza. Szöveg, program kódja csak így tömöríthető (nyilvánvaló, hogy nem lenne értelme egy programot veszteségesen tömöríteni). Sok esetben képek kódját is érdemes veszteségmentesen rövidíteni, de fontos megjegyeznünk azt is, hogy nem minden kép kódja alakítható át kisebb méretűre veszteségmentesen!
- Veszteséges tömörítésnél az eredeti és dekódolt kód között kisebb eltérések adódnak, de ezek – a módszert megfelelő körültekintéssel alkalmazva – nem zavaróak, vagy számunkra nem is észlelhetőek (észlelésünk fiziológiai korlátai miatt). Ezt a módszert nyilvánvalóan képek, hanganyagok kódjának rövidítésére célszerű használni.

Aktivitás: Bizonyítsuk be, hogy nem minden kép kódja rövidíthető veszteség nélkül! (Magyarázat: vannak olyan képek, amelyek kódja nem tartalmaz veszteségmentes tömörítést lehetővé tevő részeket.)
Igazolásvázlat: Vizsgáljuk csak azokat a képeket, amelyek kódja pontosan ezer bitből áll. Ilyen kép pontosan 2^{1000} darab van. tegyük fel, hogy mindegyik kódot sikerült rövidíteni. Ezer bitnél rövidebb kód összesen $2^1 + 2^2 + \ldots + 2^{999} = \sum_{i=1}^{999} 2^i = 2^{1000} - 2$ darab van, ami kettővel kevesebb, mint az összes ezer bitből álló. Így szükségképpen van a rövidített kódok között egyforma. Ezekből az egyforma kódokból az eredeti két különböző kód valamelyike nem dekódolható, így az a kép is csak veszteséggel állítható vissza, amelynek kódját nem tudjuk dekódolni.

Főbb veszteségmentes tömörítő algoritmusok:

- LZW-algoritmus (kidolgozói: Abraham Lempel, Jacob Ziv és Terry Welch).
- Huffman-kódolás (kidolgozója: David Albert Huffman);

A veszteséges tömörítés fontosabb algoritmusai:

- DCT-kódolás (az angol Discrete Cosine Transform kifejezésből)
- JPEG-tömörítés (Az angol Joint Photographic Experts Group kifejezésből. Ez egy cég neve. Ők alkották meg a képek kódolására használható egyéb más szabványok mellett az úgynevezett JPEG szabványt is.)

4.1.2. Az LZW-algoritmus

Az algoritmus egy szótár felépítésével kódol úgy, hogy a szótárban szereplő elemeket a tömörítendő kódban a szótárbeli indexükkel helyettesíti [4]. A kódolás lépései:

1. Állítsunk elő egy szótárat, amely a tömörítendő kód elemi kódjait fogja tartalmazni. Az elemi kódnak azt a kódszeletet, kódrészletet tekinthetjük, amiből a teljes kód összerakható. Szövegben egy betű.

2. Az olvasási pozíciót állítsuk egyre. Ez a pozíció mutatja azt, hogy a kódolásban hol tartunk.
3. Az olvasási pozíciótól kezdve vegyünk le a tömörítendő kódból annyi elemi kódot, hogy a levett kódsorozat szerepeljen a szótárban, de az eggyel több elemi kódból álló kódsorozat már ne.
4. Helyettesítsük ezt a kódsorozatot a szótárbeli sorszámmal, azaz az indexével. Vegyük fel a szótár végére azt kódsorozatot – ha van ilyen –, amelyik egy elemi kóddal hosszabb, mint amit a szótárba betettünk, majd állítsuk az olvasási pozíciót arra az elemi kódra, amelyikkel kiegészítettük a szótárba való felvételhez a sorozatunkat.
5. Ha nem értünk a tömörítendő kód végére, akkor ismételjük meg a lépéseket a 3. lépéstől.
6. Ha a tömörítendő kód végére értünk, akkor az új kód az indexekből álló sorozat lesz.

Példa: Legyen a tömörítendő kód a következő jelsorozat: almafaalattalmavan.

Az eredeti kód (almafaalattalmavan) 18 hosszú, az új kód (12314191558a617) 15 hosszú, ha nem is jelentősen, de rövidebb, mint az eredeti volt.

4.8. táblázat. *Az LZW tömörítő algoritmus alkalmazása*

A szótár		
index	kód	megjegyzés
1	a	A kiindulási szótár
2	l	
3	m	
4	f	
5	t	
6	v	
7	n	
8	al	1. bővítés
9	lm	2. bővítés
a	ma	3. bővítés
b	af	4. bővítés
c	fa	5. bővítés
d	aa	6. bővítés
e	ala	7. bővítés
f	at	8. bővítés
g	tt	9. bővítés
h	ta	10. bővítés
i	alm	11. bővítés
j	mav	12. bővítés
k	va	13. bővítés
l	an	14. bővítés

A helyettesített kód	Az új kód	A kódolás lépései
a	1	1. lépés
l	2	2. lépés
m	3	3. lépés
a	1	4. lépés
f	4	5. lépés
a	1	6. lépés
al	8	7. lépés
a	1	8. lépés
t	5	9. lépés
t	5	10. lépés
al	8	11. lépés
ma	a	12. lépés
v	6	13. lépés
a	1	14. lépés
n	7	15. lépés

4.1.3. A Huffman-algoritmus

A Huffman-kódolás [4] fix hosszúságú kódokból (egymásutánírással) összerakott kódok tömörítésére használatos népszerű, hatékony és viszonylag egyszerű módszer. Az eljárás végrehajtása során táblázatot készítünk az egyes elemek (pl. képpontok vagy karakterek) kódjainak – az elemi kódoknak – előfordulási gyakoriságáról. Ez alapján építjük fel az új kódokat (bináris jelsorozatok); változó hosszú kódszavakat használva, gyakoribb jelhez rövidebb kód tartozik, míg a ritkábbakhoz hosszabb (így érhető el a tényleges tömörítés).

A hagyományos kódolási módszerek fix hosszú kódokat használnak, pl. karakterekre az ACSII kódolást.

A módszer veszteségmentes. Érdekesség, hogy a változó hosszú kódszavak egyértelmű visszafejtését az teszi lehetővé, hogy úgynevezett prefix kód [4] keletkezik (egyik elemi kód sem lehet kezdőszelete semelyik másik elemi kódnak). Az átkódolást, és ezáltal a tömörítést, egy tipikus mohó algoritmussal végezzük el.

4.1. definíció: A mohó algoritmus olyan optimummeghatározó módszer, amelyben minden elemi lépés lokális optimumot határoz meg abban a reményben, hogy ezáltal az összes elemi lépést végrehajtva a globális optimumhoz jutunk.

Bizonyítható, hogy a Huffman-kód optimális tömörítést ad, azaz az eredmény már nem javítható tovább, ill. bármely más módszert alkalmazva sem kaphatunk jobb tömörítést (az igazolást nem tárgyaljuk). Ez azt jelenti, hogy a tömörítést megvalósító mohó algoritmus valóban globális optimumot állít elő. (Nem minden mohó algoritmus szolgáltat globális optimumot!) Az, hogy a tömörítés milyen mértékű lesz, nyilván a konkrét adatsor jellegzetességeitől függ. Nagyon sok szabályosság, ismétlődés esetén akár az eredeti méret töredékét is elérhetjük, de nem teljesen egyenletes eloszlásnál minden esetben az eredetinél rövidebb kódot kapunk.

A módszer a tömörítés elvégzéséhez egy bináris irányított fagráfot épít fel, mégpedig úgy, hogy minden iterációs lépésben egy-egy farészlettel bővül a fa, amíg fel nem épül.

4.2. definíció: A bináris fagráf egy olyan struktúra, amely pontokból és élekből épül fel a következő szabályok szerint

- minden pontból vagy kettő vagy nulla él indulhat ki;
- egy pont – a gyökérpont – kivételével minden pontba pontosan egy él érkezik. A gyökérpontba nem érkezik él.

Azokat a pontokat, amelyekből nem indul ki él, leveleknek nevezzük. Egyetlen pontból is állhat a bináris fa, ebben az esetben a fának nincs éle! Az éleket és pontokat megcímkézhetjük, azaz kódokkal láthatjuk el őket. Ha a fagráfot le szeretnénk rajzolni, akkor a rajzon a pontokat valamilyen síkidommal, az éleket vonallal jelképezzük.

A tömörítő algoritmus a következő:

1. Készítsünk egy táblázatot a tömöríteni kívánt kódban előforduló elemi kódok gyakoriságáról. Adjunk meg annyi – egyetlen pontból álló – fagráfot, ahány elemi kódot találtunk, és minden pontra írjuk rá kódként ezt az elemi kódot és az előfordulás gyakoriságát. Rendezzük növekvően sorba a fagráfokat a gyökérponthoz rendelt gyakoriság szerint.
2. A sorba rendezett fagráfok közül vegyük ki az első kettőt, és készítsünk belőlük egy új fagráfot úgy, hogy veszünk egy új pontot ez lesz a gyökérpont , ebből a pontból élet vezetünk mindkét fagráf gyökérpontjához. Bal él menjen a kisebb, jobb él menjen a nagyobb értékű gyökérponthoz. Ezzel ezek már nem lesznek gyökérpontok, hiszen mindegyikbe vezet már él. Az új pontot címkézzük meg a két felhasznált fagráf gyökérpontjában lévő gyakorisági értékek összegével (ez lesz az új fagráfban lévő elemi kódok gyakorisága). Az új fagráf új bal élére írjunk 0-t az új jobb élére 1-et, majd illesszük be a fagráfok sorába az új gráfot úgy, hogy a gyakoriság szerinti növekvő sorrend megmaradjon. (Megjegyzés: a fagráfok száma ezzel eggyel csökken!)
3. Ha egynél több fagráfunk van, ismételjük meg a 2. lépést.

4. Ha csak egy fagráf maradt, akkor ennek segítségével megállapíthatjuk az elemi kódok helyére írandó új kódot. Ugyanis ha megfigyeljük, ennek az egyetlen maradék fagráfnak a levélcímkéiben éppen az elemi kódok vannak, és ha a gyökérponttól valamelyik levélhez a gráf éleinek felhasználásával elmegyünk, és az érintés sorrendjében felírjuk az él címkéit, a kapott bitsorozat alkalmas helyettesítő kódja lehet a levél elemi kódjának. Az összes levélre felírva ezeket a kódokat, és a tömörítendő kódban ezekre lecserélve az összes elemi kódot, az eredeti tömörebb kódját kapjuk.

Be lehet bizonyítani, hogy az új kód nem hosszabb a réginél. Ha a régivel megegyező hosszúságú, akkor a régi kód veszteség nélkül nem rövidíthető, tömöríthető.

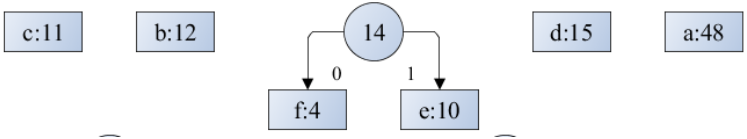
Az algoritmus működését egy konkrét példán mutatjuk be [1]. Egy olyan, 100 000 karakterből álló szöveget szeretnénk tömöríteni, amelyben 6-féle karakter (mondjuk a, b, c, d, e, f betűk) fordulhat elő. Az eredeti kódban a karakterek kódolására hárombites fix hosszúságú kódokat használtunk, a keletkezett kód hossza 300 000 bit. Ezen szeretnénk javítani a tömörítéssel. Az egyes karakterek ezerre redukált előfordulási gyakoriságát a 4.9. táblázat mutatja.

4.9. táblázat. *Egy szövegben előforduló betűk gyakorisága*

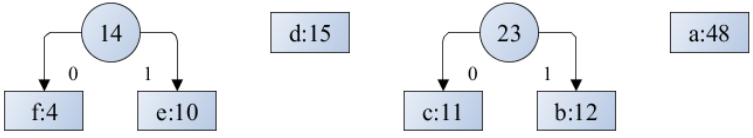
Karakterek	a	b	c	d	e	f
Gyakoriság ezerben	48	12	11	15	10	4

Az eljárás példánkra a következő fát építi fel:

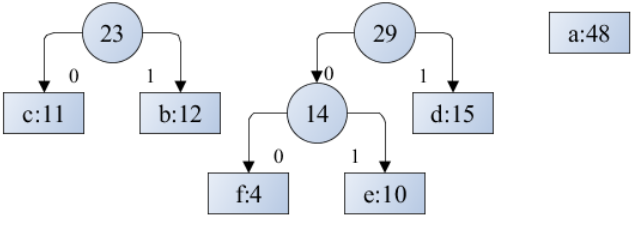
Az algoritmus első lépésének eredménye.



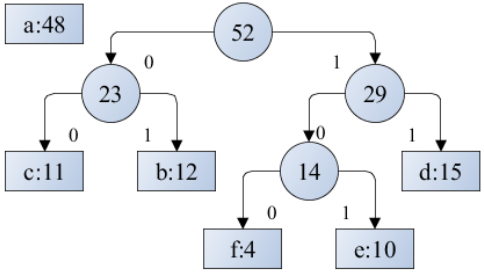
Az algoritmus második lépésének első végrehajtásával keletkezett gráfsorozat.



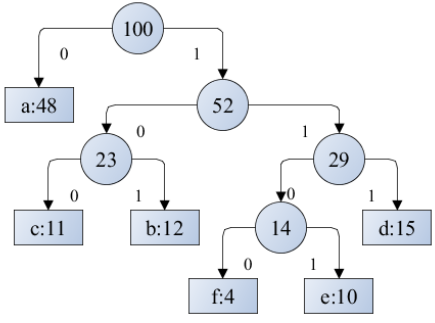
Az algoritmus második lépésének második végrehajtásával keletkezett gráfsorozat.



Az algoritmus második lépésének harmadik végrehajtásával keletkezett gráfsorozat.



Az algoritmus második lépésének negyedik végrehajtásával keletkezett gráfsorozat.



Az algoritmus második lépésének ötödik végrehajtásával keletkezett gráfsorozat. Ez egyben a végeredmény is.

4.1. ábra. *A Huffman-kódoló eljárás működése a fenti példára.*

A fa alapján megállapított új elemi kódok a 4.10. táblázatban vannak összegyűjtve. Ezekkel a régi elemi kódokat helyettesítve kapjuk a tömörített bitsorozatot.

Látható, hogy változó hosszú kódolásnál $(48000 \cdot 1 + 12000 \cdot 3 + 11000 \cdot 3 + 15000 \cdot 3 + 10000 \cdot 4 + 4000 \cdot 4) = 218000$ bit lesz a kód hossza. Ez nagyjából 27%-os megtakarítást eredményez.

A Huffman-kódolást gyakran használják a tömörítő programokban (természetesen más további algoritmusokat is).

4.10. táblázat. *Az LZW tömörítő algoritmus alkalmazása*

Karakterek	a	b	c	d	e	f
Fix hosszú kódszó	000	001	010	011	100	101
Változó hosszú kódszó	0	101	100	111	1101	1100

4.1.4. DCT-kódolás

Veszteséges tömörítésre alkalmas algoritmus. Működése vázlatosan – kép tömörített kódjának megállapítására – a következő:

Osszuk a képet négyzet alakú blokkokra. A négyzetek méretét a lefedett képponttal jellemezzük. (Pl. lehet 88 képpontot tartalmazó négyzet.) Minden négyzethez tartozó pontrendszert külön kódolunk, A négyzet bal felső sarkában lévő pontnak – mint bázispontnak – valamilyen képpont kódolására alkalmas rendszerben megadjuk a kódját. A többi pontra meghatározzuk, hogy a bázisponttól mennyire tér el, ezt az eltérést kódolva kapjuk a pont kódját, ami rövidebb, mintha a bázispontra alkalmazott módszer szerint kódolnánk. Ez a kódolás a kép éles átmeneteit rontja, de megjelenhetnek a képen a blokkhatárok is.

Minél nagyobb a blokk mérete, annál kisebb méretű kódot kapunk, de annál rosszabb lesz a visszaállított kép minősége is.

4.1.5. JPEG-tömörítés

Nagyon egyszerű tömörítő eljárás. Két lépésből áll. Első lépés egy olyan DCT-kódolás, amely úgy módosul, hogy a bázisponttól csak kismértékben különböző pontokat úgy veszi, mintha ezek a pontok a bázisponttal megegyező tulajdonságúak lennének, majd egy Huffman-tömörítés következik.

4.2. Titkosítás

A kriptográfia (titkosítás) több ezer éves tudomány (művészet); írások és üzenetek olyan (kódolt titkos anyagba történő) rejtjelezésével foglalkozik, amely illetéktelen személyek számára megnehezíti – ha sikerül „jól” titkosítani – megakadályozza a visszafejtést. A teljes titkosító módszernek nyilvánvalóan olyannak kell lennie, hogy a jogosult fogadó képes legyen viszonylag egyszerűen és egyértelműen visszafejteni a szöveget (kriptorendszer) [4].

Szintén több ezer éves kapcsolódó – konkurens – társtudomány (művészet) a kriptoanalízis, amely rejtjelezett üzenetek (illetéktelen) feltörésével foglalkozik.

A kriptográfia tipikus alaphelyzete a következő:

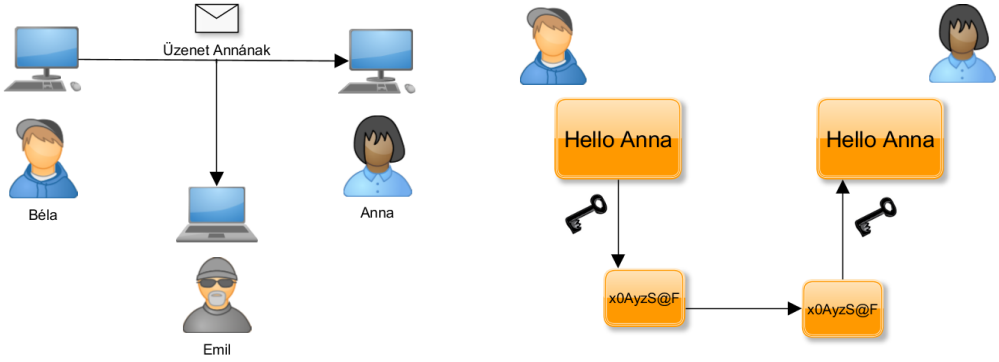
- Kommunikáció zajlik két szereplő között (Anna és Béla), amelyet lehallgathat egy külső szereplő (Emil) (a kommunikáció manapság többnyire a számítógépek között zajlik, az átvitelt az internet teszi lehetővé);
- Emiatt az üzenetküldés egy kulcs segítségével kódolva (titkosítva) történik (encoding – E függvény), a titkosított üzenet pedig egy (másik) kulccsal visszafejthető (decoding – D függvény);

A titkosítás lehet szimmetrikus (titkos) kulcsú és aszimmetrikus (nyilvános) kulcsú.

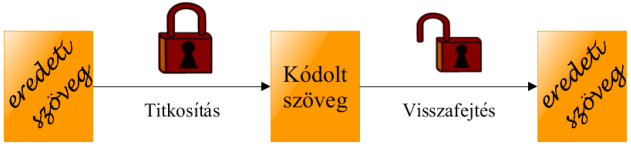
4.2.1. Szimmetrikus és aszimmetrikus kulcsú titkosítás

A szimmetrikus (titkos) kulcsú titkosításnál a kódoláshoz és a visszafejtéshez használt kulcs megegyezik, vagy az egyik könnyen kiszámolható a másikból. Így a kulcsot feltétlenül titokban kell tartani, ugyanis ha valaki külsős mégis hozzáfér a kulcshoz, akkor képes az összes korábbi üzenetet dekódolni, illetve bármelyik fél nevében üzenetet hamisítani.

A régi korok titkosító eljárásai (egészen az elmúlt évtizedekig) mind ezen az elven alapultak. Az első ilyen eljárás az egyszerű betűeltolásos titkosítás volt (az ábécé betűit kódolták, és az eredeti szövegben minden betűt az ábécében megtalálható kódjával helyettesítettek – kódolás: a 4.11. táblázatban, visszakódolás: a 4.12.



4.2. ábra. *A kriptográfia alaphelyzete*



4.3. ábra. *Szimmetrikus kulcsú titkosítás*

táblázatban). Ezt a titkosítást azonban igen egyszerű feltörni. A későbbi – fejlettebb – változatokban már a szövegbeli elhelyezkedéstől függően más és más volt ugyanannak a karakternek a kódja. (Akár egészen bonyolult szabályok alapján az eredeti szöveg egy adott betűjének a titkosított szövegben más és más betű felel meg). Bár sokszor időigényesen, de ezek a kódolások is mind feltörhetőnek bizonyultak (pl. betűk, betűpárok, betűhármasok stb. előfordulásának vizsgálatával).

4.11. táblázat. *NAVIGARENECESSEEST szöveg titkosítása a betűk ábécébeli sorszámának 3-mal való megnövelésével történik. Klasszikus titkosító eljárás*

A feladói oldal:

Üzenet	N	A	V	I	G	A	R	E	N	E	C	E	S	S	E	E	S	T
Betűsorszám	13	0	21	8	6	0	17	4	13	4	2	4	18	18	4	4	18	19
3-mal eltoltan	16	3	24	11	9	3	20	7	16	7	5	7	21	21	7	7	21	22
Titkosítva	Q	D	Y	L	J	D	U	H	Q	H	F	H	V	V	H	H	V	W

4.12. táblázat. *NAVIGARENECESSEEST szöveg visszafejtése a betűk ábécébeli sorszámának 3-mal való csökkentésével történik. Klasszikus titkosító eljárás visszafejtése*

A feladóoldal:

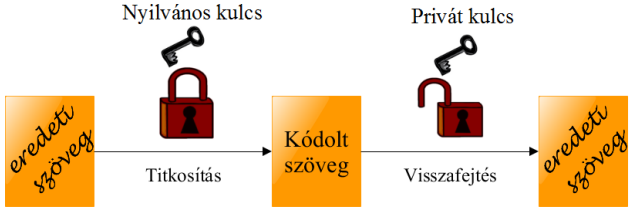
Titkosított	Q	D	Y	L	J	D	U	H	Q	H	F	H	V	V	H	H	V	W
Üzenet	N	A	V	I	G	A	R	E	N	E	C	E	S	S	E	E	S	T
Betűsorszám	16	3	24	11	9	3	20	7	16	7	5	7	21	21	7	7	21	22
3-mal csökkentve	13	0	21	8	6	0	17	4	13	4	2	4	18	18	4	4	18	19
Visszafejtve	N	A	V	I	G	A	R	E	N	E	C	E	S	S	E	E	S	T

A módszer (megfelelő matematikai alapokkal) napjainkban is sok esetben nagyon jól alkalmazható (tipikusan ott, ahol a küldés és a fogadás egy helyen történik), ugyanakkor általános használatuk nem javasolt, mivel számítógéppel viszonylag gyorsan feltörhetők.

A gyakorlati alkalmazásnál nehézséget jelent még, hogy a kulcsot az adatátvitel előtt valahogy (zárt módon) el kell juttatni egyik féltől a másikig, továbbá – sokszereplős környezetben – minden kommunikációs partnerhez különböző kulcsot kell használni, hisz közös kulcs esetén el tudnák olvasni egymás üzeneteit.

Jelöljük e-vel a nyilvános kulcsot, ezt ismeri az üzenet küldője, d-vel a privát kulcsot, amit csak a címzett ismer.

Az aszimmetrikus (nyilvános) kulcsú titkosító eljárások általános jellemzője, hogy $d \neq e$, és „gyakorlatilag nem számítható ki” e-ből d.



4.4. ábra. *Aszimmetrikus kulcsú titkosítás*

A kódolás és visszafejtés lépései a következők:

Egy nyilvánosan elérhető, megbízható forrásból (pl. magától a címzettől) megszerezzük a címzett nyilvános kulcsát;

Az üzenetet kódoljuk ezzel a kulccsal, majd elküldjük a címzettnek;

A megkapott üzenetet a címzett saját privát kulcsával visszafejti (a kódolt üzenet csak ezzel a kulccsal nyitható!), a végeredmény az eredeti, titkosítatlan szöveg lesz.

A legtöbb ma használt kommunikációs szabvány ilyen típusú megoldást alkalmaz a biztonságos adatcseréhez. Ezen az elven alapul a digitális aláírás készítése és ellenőrzése is.

4.2.2. Az RSA algoritmus

Az RSA [4] az egyik leggyakrabban használt nyilvános kulcsú algoritmus. Az elnevezés rövidítés, a kifejlesztő kutatók neveiből (Ron Rivest, Adi Shamir, Leonard Adleman; 1978). A módszer bemutatása annak idején nagy feltűnést keltett, ugyanis itt szerepelt először – biztonságos, jól alkalmazható megoldással – nyilvános kulcs.

Az algoritmus lépései: *kulcsgenerálás, rejtjelezés, visszafejtés*. A következőkben ezeket a tevékenységeket részletesen is áttekintjük (bizonyos matematikai ismeretekre szükség lesz a megértéshez). A korábbiaknak megfelelően E a bekódolás, D a visszafejtés művelete. Felhívjuk a figyelmet arra, hogy a kódjaink bitekből állnak, így minden kód egy kettes számrendszerbeli számként értelmezhető, és ezzel a számmal tízes számrendszerbe átírva minden művelet elvégezhető!

4.2.3. Kulcsgenerálás

4.3. definíció: Az a és b számok akkor relatív prímek, ha lnko$(a,b) = 1$ (azaz a és b legnagyobb közös osztója 1).

Használni fogjuk a $\varphi(n)$ számelméleti függvényt, ami az n-nél kisebb, az n-hez relatívprím-számok számát határozza meg. Pl. $\varphi(2) = 1$, $\varphi(3) = 2$, $\varphi(4) = 2$, $\varphi(5) = 4$ stb. Ha a, b relatív prímek, akkor teljesül a multiplikativitási szabály is, azaz $\varphi(N) = \varphi(a) \cdot \varphi(b)$, ha $N = a \cdot b$.

Tegyük fel, hogy Béla küldi majd az üzenetet Annának – ő lesz a címzett.

Anna választ véletlenszerűen két (nagy, azaz akár százjegyű) prímszámot, p-t és q-t (itt $p \neq q$), kiszámítja az $N = p \cdot q$ számot. Ezek után választ egy e számot úgy, hogy $1 < e < \varphi(N) = \varphi(p)\varphi(q) = (p-1)(q-1)$ és

lnko$(e,\varphi(N)) = 1$.

Anna ezután meghatározza azt az egyértelmű d számot, amelyre $1 < d < \varphi(N)$ és $ed = 1 \pmod{\varphi(N)}$.

A d szám itt az e inverze modulo $\varphi(N)$, a „modulo” művelet (rövidítve mod) pedig az osztási maradékot képzi.

Ezzel a kulcsgenerálás befejeződött, az (N,e) pár lesz Anna nyilvános kulcsa, (N,d) pedig Anna titkos kulcsa.

4.2.4. Rejtjelezés

Osszuk blokkokra Béla üzenetének bitekkel leírt kódját (ezt a kódot valamely ismert kódrendszerrel alakította ki Béla). A blokkokat úgy alakítsuk ki, hogy a blokk kódjának számértéke minden blokk esetén legyen N-nél kisebb. Jelöljük m-mel egy tetszőleges blokk által meghatározott számot!

Béla ismeri Anna nyilvános kulcsát, így a kódoló függvényt a következőképpen definiálhatja: $E(n) = n^e \pmod{N}$. Ez alapján m titkos kódja $E(m) = m^e \pmod{N} = c$. Végezzük el minden blokkra az átalakítást, a keletkező titkos kódokat egymás után írva megkapjuk a titkos üzenetet, amit el lehet küldeni Annának.

4.2.5. Visszafejtés

Jelöljük c-vel a titkos üzenet egy tetszőleges blokk-kódjának számértékét.

Anna a titkos kulcsa alapján definiálja a $D(n) = n^d \pmod{N}$ függvényt. Ezzel ki tudja számítani a $D(c) = c^d \pmod{N} = m$ számot. Be lehet bizonyítani, hogy ez a szám éppen a c által meghatározott titkos kód eredeti kódja lesz, azaz éppen az az m szám, amelyre $c = E(m)$, tehát $D(E(m)) = m$. Minden blokkra elvégezve a számolást visszakapjuk az eredeti üzenet kódját, amit már el tudunk olvasni, hiszen az a kódrendszer, amiben ezt Béla kódolta már mindenki számára ismert. (Ez azt jelenti, hogy az RSA kriptorendszert alkot).

4.2.6. Néhány megjegyzés a titkosítás matematikai alapjaihoz

Az RSA lépéseinek tényleges gyakorlati alkalmazhatóságához több fontos kérdést, problémát tisztázni kell. Ehhez már a komputeralgebra egyes területeit is érintjük. A mélyebben érdeklődő Olvasónak egyéni

felkészültségétől függően szükséges lehet további tájékozódás a témakörben. A részleteket ebben a jegyzetben az adott keretek között nem tudjuk bemutatni.

Az egyik jelentős kérdés, hogyan válasszuk meg a p és q prímszámokat. Ha ezek kicsik lennének, akkor az üzenetet lehallgató Eve az n számot könnyedén faktorizálni tudná (fel tudná bontani szorzattá), és így meg tudná határozni a d titkos kulcsot. (Hatékony faktorizáló algoritmussal „kicsinek” tekinthetőek akár 20-25-jegyű prímek is.)

Mivel d az e inverze modulo $\varphi(N)$, ezért ennek meghatározására léteznek módszerek. Ilyen módszer a bővített euklideszi algoritmus, amelynek leírása [1]-ben megtalálható.

Mindezek miatt a gyakorlatban (a mostani nagy számítógépek teljesítményét és a visszafejtő algoritmusok tudását figyelembe véve) p-t és q-t legalább 80-100-jegyű (decimális) számnak kell választani.

Újabb fontos kérdés, hogy hol és hogyan találunk ilyen prímeket. A javaslat az, hogy alkalmas programmal véletlenül generálunk ilyen sokjegyű számokat, teszteljük őket, hogy prímek-e (a prímek elég „sűrűn” helyezkednek el ahhoz, hogy az eljárás működhessen).

Tudjuk, hogy a prímek száma végtelen, így jó eséllyel megtippelhetjük valamelyiket véletlenül is, de hogyan állapítsuk meg egy véletlenszerűen kiválasztott 80-100-jegyű számról, hogy prím-e?

4.5. ábra. *Egy sokszámjegyű prím*

A prímtulajdonság biztos, pontos tesztelése nehéz feladat! Pierre de Fermat (1601–1665) rájött arra, hogy ha p prímszám és a-nak nem osztója p, akkor (pl. $a = 2$) igaz, hogy $a^{p-1} = 1 \pmod{p}$. Erre a megállapításra épül a Fermat-teszt, amit a következőképpen kell elvégezni. Válasszunk az ellenőrizendő p számhoz egy olyan a számot, amelyre $\text{lnko}(p,a) = 1$. Általában $a = 2$ jó választás, hiszen páros p biztosan nem prím. Ha $2^{p-1} \neq 1 \pmod{p}$, akkor p nem prím.

Sajnos, ha $2^{p-1} = 1 \pmod{p}$, akkor az még nem jelenti azt, hogy p prím, mivel vannak olyan „prímként

viselkedő összetett számok" (pszeudoprímek), amelyekre szintén teljesül, hogy $2^{p-1} = 1 \pmod{p}$. Ezért ha a tesztet olyan szűrőknek tekintjük, amivel megállapítható egy számról, hogy összetett vagy prím-e, akkor nem tökéletes. Kisebb eséllyel lesz álprím a p, ha más alkalmas a-kkal is elvégezzük a tesztet, de akárhány a-val is ellenőrzünk, a tesztek után határozottan nemnulla annak a valószínűsége, hogy a p összetett szám.

Azaz a Fermat-féle teszttel elég gyorsan megállapítható adott p-ről, hogy összetett-e, és eljuthatunk odáig, hogy az adott szám valószínűleg prím.

Más – úgynevezett valószínűségi – prímtesztek is léteznek, ezek elég gyorsak, és gyakorlati szempontból már teljesen megbízhatóan igazolják, hogy a vizsgált szám prím (ilyen pl. a Miller-Rabin-féle teszt, amelyről [1]-ben olvashatunk).

Működési elvük a következő: egy tesztlépés után kétféle választ adhatnak, hogy „a szám összetett" és azt, hogy „a szám valószínűleg prím". Az első esetben a válasz biztos és végleges. A második esetben tévedhetünk, de a hiba esélye minden egyes egymástól függetlenül elvégzett tesztlépésben egy $\bar{p}$, 50%-nál kisebb fix érték. Ha tízszer egy más után elvégezzük egy-egy tesztlépést, és a válasz mindig az, hogy „a szám valószínűleg prím", akkor a tévedés esélye már csak $\bar{p}^{10}$), ami igen kicsi érték. Elég sokszor elvégezve a tesztlépéseket a választ lényegében biztosnak tekinthetjük.

Másik lehetséges megoldás, hogy egy egzakt prímteszttel, matematikailag is korrekt módon megállapítjuk a vizsgált szám prímtulajdonságát. Sajnos az egzakt prímtesztek még a nagyteljesítményű számítógépekkel is éveket vehetnek igénybe!

Foglalkoznunk kell azzal a kérdéssel is, hogy hogyan tudunk hatékonyan nagy hatványra emelni számokat (ez ugyanis az algoritmus több kulcslépésében is szerepel). A számolás gyorsításához a kitevőt kettő hatványok összegére bontva a hatvány olyan szorzattá alakítható, ahol a szorzótényezők is hatványok, de egyik a másikból könnyen kiszámítható, és így a hatványozás néhány szorzással helyettesíthető lesz.

Példa: $697^{30} = 697^{16} \cdot 697^{8} \cdot 697^{4} \cdot 697^{2}$. Egy szorzással kiszámítjuk a 697^2-t, eggyel a 697^4-t, eggyel a 697^8-t, eggyel a 697^{16}-t, majd a kapott négy tényezőt három szorzással összeszorozva megkapjuk a 697^{30}-t. Ez harminc szorzás helyett csak hét szorzás, ami ilyen nagy számok esetén jelentős különbség!

4.2.7. Az RSA kódolással kapcsolatos biztonsági kérdések

Bár azt gondolhatnánk, hogy az RSA eljárás 100%-os biztonságot nyújt, azaz a módszerrel előállított titkos kód feltörhetetlen, ez nem így van. A módszer nyújtotta védelem döntően azon (a sejtésen) alapszik, hogy a nagy számok faktorizációja (szorzótényezőkre történő felbontása) igen nehéz feladat, a végrehajtására nagyon sok idő kell (napok, hónapok, évek). Ez valószínűleg igaz, de még nem sikerült bizonyítani. Tapasztalati alapon kijelenthetjük, hogy a kód feltörése általános esetben, megfelelően nagy p és q választásával olyan sok ideig tartana, hogy nem érdemes megpróbálni. (Ez már garantálja a szükséges biztonságot.)

Lehet azonban hibázni a prímgenerálásnál. Egyes speciális esetekben („ügyetlen” prímgenerálás következményeképpen, p vagy q valamely „kihasználható” tulajdonsággal rendelkezik) meg lehet találni az osztókat nagy N (összetett szám) esetében is. Arra, hogy mikor ügyetlen a prímválasztás, és mit jelent a kihasználható tulajdonság részletesen nem térünk ki, mivel nem rendelkezünk azokkal a matematikai ismeretekkel, amelyekre szükségünk lenne a magyarázathoz. Az alábbiakban néhány nagyon egyszerű példát említünk az ügyetlen prímválasztásra.

Például, ha p-t vagy q-t úgy választottuk, hogy $p-1$, illetve $q-1$ könnyen felbontható törzstényezőkre, akkor létezik olyan módszer, ami jó eséllyel felbonthatja N-t is. (Ilyen például Pollard-algoritmus, ennek részleteit [1]-ben megtaláljuk.)

Más tipikus feltörést segítő hibákat is el lehet követni az RSA kódolásnál. Ilyenek lehetnek a következők:

- Kicsi vagy nagyon speciális e szám választása. (Ekkor az $E(m) = m^e \pmod N$ kiszámítása után az e-edik gyökvonást elvégezve megkaphatjuk m-et, ez alapján az eredeti üzenet titkosítatlan kódját, ami olvashatóvá teszi az üzenetet is.);
- A p és q számok túl közel vannak egymáshoz (ilyen esetre már léteznek visszafejtő algoritmusok!);
- A szöveg karakterenkénti vagy kis blokkonkénti kódolása (a karakterenkénti titkosítás minden azonos karakterre ugyanazt a kimenetet adja, ami az üzenetet ugyanolyan könnyen megfejthetővé teszi, mint a hagyományos titkosírás esetén lásd 4.13. táblázat).

4.13. táblázat. *Az RSA kódolás $p = 3$, $q = 11$, $N = 33$, $\varphi(N) = 20$, $e = 7$, $d = 3$ paraméterekkel*

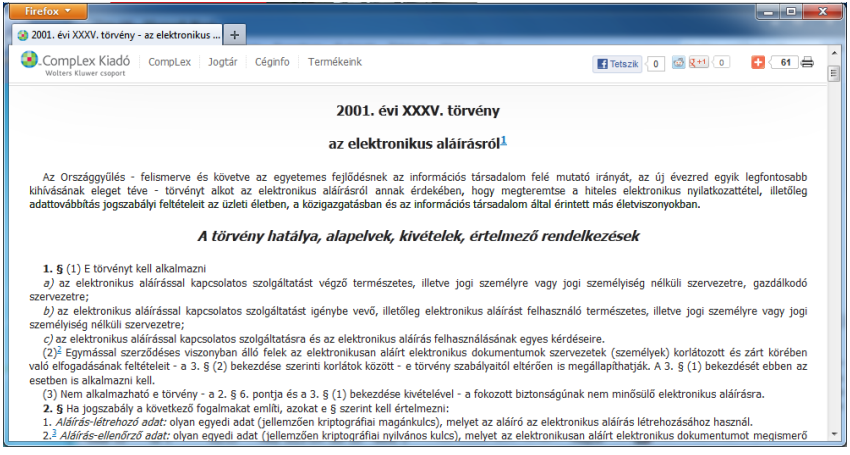
A feladói oldal:																		
Üzenet	N	A	V	I	G	A	R	E	N	E	C	E	S	S	E	E	S	T
Sorszámkód	13	0	21	8	6	0	17	4	13	4	2	4	18	18	4	4	18	19
Titkos kód	7	0	21	2	30	0	8	16	7	16	29	16	6	6	16	16	6	13
Címzett oldala																		
Titkos kód	7	0	21	2	30	0	8	16	7	16	29	16	6	6	16	16	6	13
Visszafejtve	13	0	21	8	6	0	17	4	13	4	2	4	18	18	4	4	18	19
Visszaállítva	N	A	V	I	G	A	R	E	N	E	C	E	S	S	E	E	S	T

4.2.8. Nyilvános kulcsú titkosító eljárások alkalmazása

Az aszimmetrikus kulcsú eljárások segítségével nemcsak direkt titkosítást lehet megvalósítani, hanem *elektronikus aláírást* is.

Az elektronikus aláírásnál nem az üzeneteket rejtjelezése a cél. A titkosítás csak eszköz annak megállapítására, hogy a kódolt üzenet küldője valóban az-e, mint akinek mondja magát, illetve más nem módosította az üzenetet. Ennek ellenőrzésére az ad lehetőséget, hogy a privát kulcs egyedi, és az RSA-kódolást alkalmazva a privát kulccsal kódolt üzenet a nyilvános kulccsal elolvashatóvá konvertálható. Ugyanis az E és D függvények egymás inverzei. Az elektronikus aláírást tehát a következő módon lehet megvalósítani. Az aláíró megállapít saját maga számára egy privát kulcsot, partnereinek pedig egy nyilvános kulcsot ad. A privát kulcsával kódolja az aláírását (pl.: „Én X Y vagyok” szöveget), és ezt elküldi a partnerének ha az kéri, de akár kérés nélkül is küldheti (pl.: internetes weboldalain minden oldalon megjelenteti). A partner a nyilvános kulccsal visszafejti az aláírást. Ha sikerül, akkor azonosította a küldőt, ha nem, akkor a küldő ismeretlen, és ennek megfelelően kell kezelni az üzenetet.

Az elektronikus aláírást Magyarországon szabályozó – 2001. évi XXXV. – törvény az Európai Parlament, és az Európa Tanács irányelvein alapszik, és 2001. szeptember elején lépett hatályba. A törvény néhány kivételtől eltekintve teljesen egyenrangúvá teszi a digitális aláírást a hagyományos aláírással annak minden jogkövetkezményével együtt, ha az elektronikus változat az előírt feltételeknek eleget tesz.



4.6. ábra. *A 2001. évi XXXV. törvény részlete*

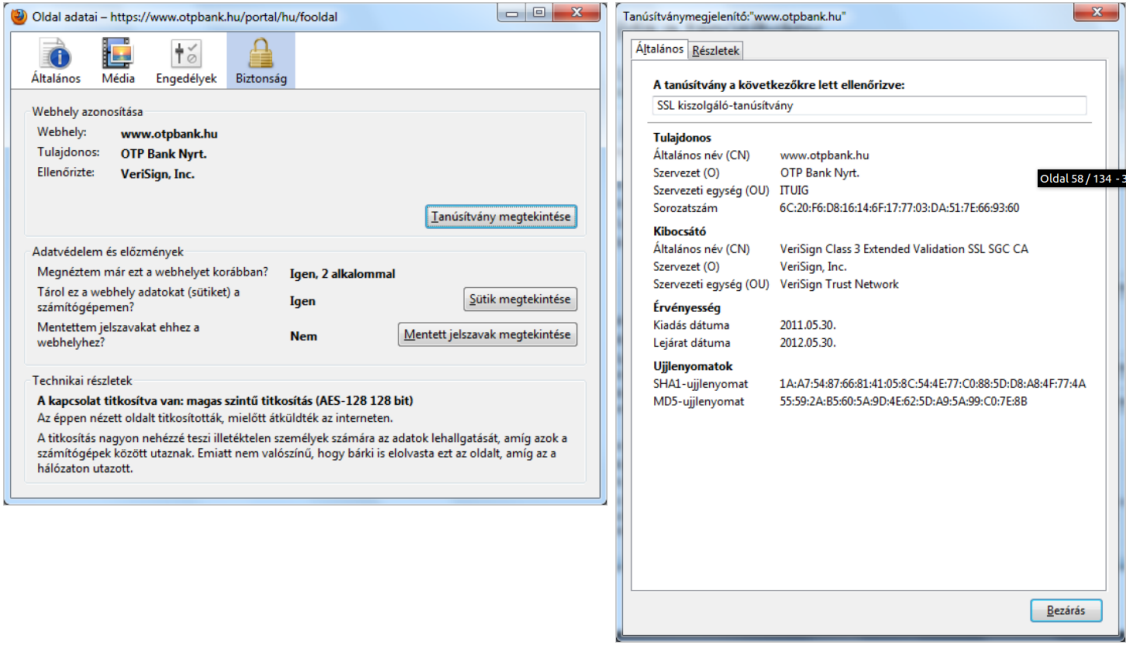
Egy másik fontos alkalmazás a digitális tanúsítvány.

Itt az alaphelyzet az, hogy valamely bizalmas adatokat kezelő partnerrel kommunikálunk. Biztonságunk garantálásához ekkor – a megfelelő titkosított kommunikáció alkalmazása mellett – arra is szükség van, hogy biztosak lehessünk abban, hogy a partner valóban az, akinek kiadja magát.

Már történtek, történnek olyan visszaélések, amely során a csalók készítenek egy valódi bankéhoz hasonló weboldalt (ilyet össze lehet állítani), és erről – hamis e-mail vagy üzenet küldése után – beugrat(hat)ják a nem kellően tájékozott felhasználót, hogy adja meg az adatait, jelszavát, amit ők aztán illegális pénzfelvételre,

átutalásra használnak.

Az esetleges visszaélések úgy védhetők ki, hogy egy alkalmas szervezet – megfelelő biztonságtechnikai módon – igazolja, hogy tényleg az a partner van a túloldalon, akit mi szeretnénk. A háttérben meghúzódó eljárás itt is a nyilvános kulcsú titkosítás.



4.7. ábra. *Digitális tanúsítvány az OTP bank oldalán*

Önellenőrzés

1. A Huffman-kódoló eljárással fát építünk fel a következő gyakoriságokkal: 6, 7, 10, 15, 20, 42. Mennyi lesz a 3. lépésben keletkező fa gyökérértéke?
   Válasz: ☐

2. A Huffman-kódolást alkalmazva mekkora a helymegtakarítás a fix hosszú kódoláshoz képest, ha az előfordulási gyakoriságok rendre:

Karakterek	a	b	c	d	e	f
Gyakoriság ezrekben	45	13	12	16	9	5

   Válasz: ☐ bit

3. Kódolja a következő szöveget LWZ, majd Huffmann-kóddal. „Szerb húsz öt cseh öt török öt görög hány ember” Melyik a rövidebb kód?
4. Titkosítsa eltolásos módszerrel a fenti szöveget!
5. Ha a titkosított szöveget LWZ módszerrel tömörítjük, melyik kód lesz a hosszabb, a titkosított szöveg vagy az eredeti szöveg kódja?
   - ☐ a titkosított szöveg kódja hosszabb
   - ☐ az eredeti szöveg kódja hosszabb
   - ☐ egyforma hosszúak lesznek a kódok

II. MODUL

A számítógép története és a hardverismeretek

4. LECKE

A számológépek és a számítógépek története Neumannig

Ebben a leckében a számolást segítő eszközök történetével foglalkozunk. Az 5.1 fejezet teljes egészében olvasmány, ennek ellenére mindenki számára javasoljuk átolvasását, mivel sok olyan érdekességet tartalmaz, amit egy tájékozott értelmiségi ember jó, ha ismer.

Az 5.2 fejezet a számítógép kialakulását ismerteti, a neumanni számítógépmodellnek, illetve főbb részeinek leírásával együtt.

Megismerkedhetünk a neumanni számítógép működésével is. Ezzel kapcsolatosan érdemes megjegyezni, hogy a mai számítógépek jó része ennek az elvnek megfelelően működik.

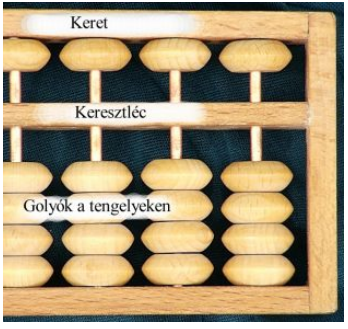
5. A számológép és a számítógép története

A számolás megkönnyítését szolgáló eszközök a számolással párhuzamosan fejlődtek. A kezdetekben lehetséges, hogy az ujjaikat használták őseink, esetleg kavicsokat, csontocskákat stb. Később – a számok absztrakt fogalommá válása során – egyéb, vonalkákat, pontrendszereket tartalmazó eszközök segítették a számolás. A legrégebbi ilyen feltételezett számolóeszköz, amit Afrikában találtak a régészek Kr. e. 20 000 táján készülhetett.

5.1. ábra. *Számolás ujjak segítségével*

5.1. A számológépek az ókortól napjainkig

Kr. e. 2000–1000 táján a számolás megkönnyítésére már olyan eszközök is ismertek voltak, amelyek manapság is használhatók, sőt a Föld egyes területein használják is őket. Ilyen eszköz a japán szorobán (5.2. ábra) és az ókori abakusz (5.3. ábra), amiknek modernizált változatait az iskolai számolásoktatásban mind a mai napig használjuk.



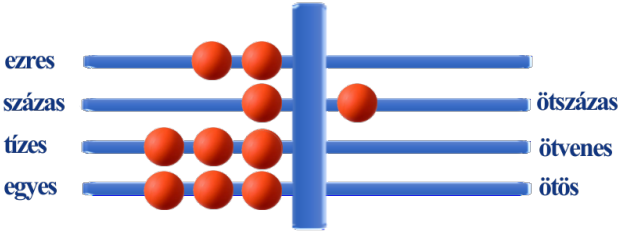
5.2. ábra. *Szorobán*

Érdemes kissé jobban megismerni az abakusz használatát, hiszen a római számokkal való műveletvégzést még mindig ezzel az eszközzel tudjuk a legjobban segíteni.

Római számokkal a műveletvégzés segédeszköz nélkül igencsak nehézkes, az abakusz viszont az összeadáshoz, kivonáshoz, szorzáshoz kiváló segédeszköznek bizonyult.

Bár többnyire kész eszközöket használtak a számoláshoz, ezt a szerkezetet egyszerűsége alkalmassá tette arra, hogy akár az út porában néhány kaviccsal pillanatokon belül abakuszt készítsen a hozzáértő felhasználó. Az 5.4 ábrán látható vízszintes vonalakat és az ezeket kettévágó függőleges vonalat akár a porban megrajzolhatjuk, a golyócskákat kavicsokkal helyettesítve máris készen áll az abakusz a számoláshoz.

5.3. ábra. *Pitagorasz (Kr. e. 582–496) az abakusz mellett*



5.4. ábra. *MMDCXXXIII az abakuszon*

A berendezés használatához csak a számok kirakási módját, az összegezést és az úgynevezett tisztázás-műveletet kellett ismerni, és máris elvégezhető volt az összeadás. (A többi műveletre most nem térünk ki.)

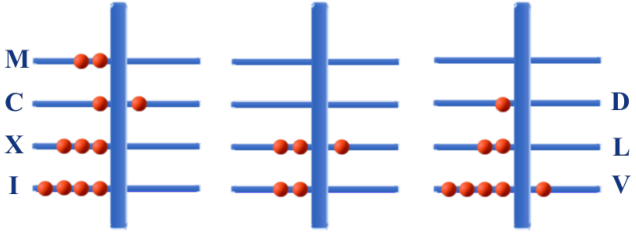
Nézzük először a számok kirakásának szabályát. Nagyon egyszerű dologról van szó, hiszen nem kell mást tenni, mint az elválasztó vonal mellé oda kell tenni a megfelelő vízszintes vonalon a római számban lévő betűjelek számának megfelelő kavicsot (az egyszerűség kedvéért használjuk a csak összeadással megadott formát!). Az eredményt a 2633 esetén az 5.4. ábrán láthatjuk. Az összeadásra annyi vonalrendszert kell használni, ahány tagú az összeg. A tagokat egyenként kirakjuk egy-egy vonalrendszeren. Elvégezzük az összegezést valamelyik vonalrendszeren úgy, hogy az egyes nagyságrendnek megfelelő vonalakon lévő összes kavicsot a kiválasztott vonalrendszeren ugyanarra a nagyságrendű vonalra gyűjtjük. Ezután az úgynevezett tisztázás műveletével kialakítjuk az összeg leírásához a római szám leírási szabályainak megfelelő formát. A tisztázást a legkisebb számnak megfelelő vonalon kezdjük a következők szerint:

1. Minden, az elválasztó vonaltól balra eső (tízhatványokat jelentő) vonalról leveszünk annyiszor öt kavicsot, ahányat csak lehetséges. A jobb oldalon mellette lévő öttöbbszörös vonalra ráteszünk annyi kavicsot, ahány ötös csoportot a mellette lévő vonalról levettünk.
2. Majd leveszünk ugyanerről a jobb oldali vonalról annyiszor két golyót, ahányszor csak tudunk, és minden egyes pár levételekor hozzáadunk az eggyel feljebb lévő vonal bal oldali kavicsaihoz egy-egy kavicsot.
3. Ezeket a lépéseket addig ismételjük, amíg el nem jutunk oda, hogy már nem tudunk sem öt sem kettő kavicsit levenni a sorra kerülő vonalakról.

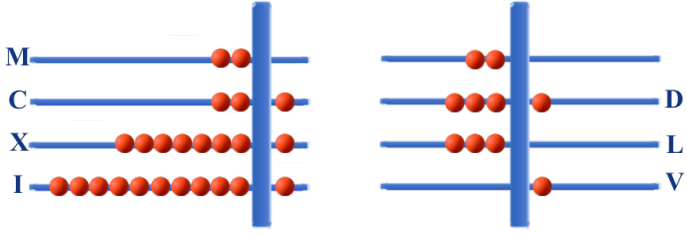
Befejezve a tisztázást leolvashatjuk az összeget.

Példa rajzolt abakuszrendszer segítségével (5.5. ábra). Legyen a kiszámítandó összeg: 2634+72+129. A megfelelő római számok: MMDCXXXIIII, LXXII, CXXVIIII. Az ábra a sora a számok kirakását mutatja, az 5.6. ábrán az összegezést és a tisztázás végeredményét látjuk. Ezt leolvasva a szám római alakja: MMDCCCXXXV, ami a 2835-nek felel meg. Ez valóban a fenti összeadás eredménye.

Az arab-hindu számok és a helyiértékes felírási mód megjelenésével az abakusszal való számolás lassan háttérbe



5.5. ábra. *Az összeadandók az abakuszon*



5.6. ábra. *Az összegzés és a tisztázás eredménye*

szorult, bár érdemes megjegyezni, hogy hozzáértő képes esetleg versenyre kelni, sőt akár megnyerni is a versenyt az 1970-es évek technológiájával készült zsebszámológépet használó számolóval is.

A 16., 17 században a tudomány és a közlekedés fejlődése olyan mennyiségű számítási feladatokat generált, hogy ezek elvégzésére az addig ismert módszerek nem voltak alkalmasak. Elsősorban a csillagászati táblázatok és a hajózással kapcsolatos táblázatok elkészítése igényelt komoly mennyiségű számításokat.

5.7. ábra. *Napier természetes alapú logaritmusa, és erre az elvre kifejlesztett számolóeszköz, a logarléc*

A feladatok elvégzéséhez egy elméleti eredmény és erre az eredményre épülő számolóberendezés nyújtott segítséget. Az elméleti eredmény a természetes alapú logaritmus bevezetése, ami John Napier (1550–1617) skót csillagász, matematikus, filozófus nevéhez fűződik. Ő tette általánossá a logaritmus használatát és neki köszönhetjük a tizedesvesszőt, és ezáltal a tizedestörtek felírásának manapság használt formáját is.

Mint ismeretes, a logaritmus segítségével a szorzás összeadásra, az osztás kivonásra vezethető vissza. Ezek a műveletek sokkal egyszerűbben, és kevesebb munkával végezhetők el. Sőt, a műveletvégzést is meg lehet takarítani, ha olyan táblázatokat készítünk, amelyekből a szorzótényezők (vagy az osztó és osztandó) ismeretében az eredmény kiolvasható. (Ilyen táblázatokat találunk a középiskolában megismert függvénytáblázatok között is.) Ezeknek a táblázatoknak elkészítéséhez konstruált Napier egy számolóeszközt, amit Napier-csontocskák néven ismer a tudománytörténet. Tulajdonképpen olyan csontpálcák rendszeréről van szó, amelyekre logaritmusskála szerinti számértékeket jelképező vonalkák voltak vésve, és ezek alkalmasan egymás mellé illesztésével lehetett szorozni, osztani. Az eszköz továbbfejlesztett modern változata a logarléc,

amely a mérnöki munkában még a 20. század második harmadában is általánosan használt számolóeszköz volt.

Alig több, mint egy évtized múltán Wilhelm Schickard (1592–1635) tübingeni professzor Napier számolóeszközéből egy mechanikus elemekből álló berendezést tervezett. Valószínűleg barátjának, Keplernek készítette, akinek csillagászként nagy mennyiségű számításokat kellett elvégeznie. A gép az összeadást, kivonást teljesen automatikusan, a szorzást, osztást a kezelő segítségét is igénybe véve végezte el. Mivel a gép megsemmisült, elkészültéről az utókor Schickard 1623-ban és 1624-ben Keplernek írt, de csak 1957-ben megtalált leveléből szerzett tudomást.

A levelekben lévő leírás alapján 1960-ban elkészítették a gép működő modelljét.

5.8. ábra. *Schickard számológépe*

Egy újabb évtized elteltével – 1642-ben – Blaise Pascal (1623–1662) francia polihisztor megtervezte és megépítette az „aritmetikai gép” nevű mechanikus számológépét, amelyből többet is készített, sőt ötven körüli prototípust is gyártott, mielőtt a berendezéssel a nyilvánosság elé lépett volna. Két szám összeadására, kivonására közvetlenül, a szorzásra és az osztásra ismételt összeadással, illetve kivonással volt képes. Ebben a berendezésben már fontos szerepet játszottak a forgó alkatrészek, a fogaskerekek, amelyek az óragyártást is lehetővé tették. Nem véletlen, hogy a kor számítógépgyártói az órakészítés mesterei voltak. Pascal gépét az utókor Pascaline néven ismeri.

Az 1670-es évek elején Wilhelm Gottfried Leibnitz (1646–1716) megépítette Pascal gépének az úgynevezett Leibnitz-kerék alkalmazásával továbbfejlesztett változatát. Ez a berendezés – az új alkatrészének köszönhetően – az összeadás és a kivonás mellett teljesen automatikusan képes volt a szorzás és osztás elvégzésére is. A Leibnitz kerék alkalmazása annyira sikeresnek bizonyult, hogy a 19. század végéig fontos része volt a mechanikus számológépeknek. Ez a részegység egy tengelyre szerelt henger volt, amelynek palástján a tengellyel párhuzamosan egyre rövidebb bordák voltak elhelyezve. Ennek következményeképpen a hengert forgatva a hozzá kapcsolódó fogaskerekek egymáshoz viszonyított körbefordulásának száma pontosan

szabályozhatóvá vált. Ez a tulajdonság tette lehetővé a szorzás mechanikus elvégzését is.

Az első igazán használható mechanikus számológépet az úgynevezett Arithmometert egy párizsi órásmester, Thomas de Colmar (1785–1870) készítette a francia hadsereg számára az 1820-as évek elején. Csak néhány készült belőle, és nem vált ismerté egészen 1851-ig, amíg a Párizsban megrendezett Nemzetközi Ipari Vásáron be nem mutatták. Akkora sikert aratott, hogy 1851-től negyven év alatt de Colmar manufaktúrája több mint 5000 darabot gyártott belőle. Egészen az 1900-as évekig szinte egyeduralkodó volt a piacon, bár a 19. század elejétől amerikai cégek is készítettek számológépeket. A 20. század elején viszont megjelentek a német, az orosz és a svéd konkurens gyártmányok is. Ezen cégek által gyártott berendezések Odhner-típusú gépek voltak, mivel a gyártó cégeket Odhner és jogutódjai alapították. Az orosz gyártás még a szovjet időkben is az Odhner-féle konstrukción alapult.

5.9. ábra. *Pascal és a Pascaline*

Willgodt Theophil Odhner (1845–1905) a cári Oroszországba emigrált svéd származású műszerész volt, aki számológép-javítással is foglakozott. Az 1870-es évek elején Colmar-féle számológépet javítva a robusztus Leibnitz-kereket egy könnyebb, egyszerűbben működő úgynevezett pin-kerékre cserélte, és ezzel egy új irányt mutatott a számológépgyártásban. Európában számtalan klónját gyártották gépének, és a 20. század közepéig az irodai és a műszaki számításokhoz szinte kizárólagosan Odhner-gépeket használtak.

Fel kell hívni a figyelmet arra, hogy ezek a számológépek csak segítették a számolást. A felhasználó állította be rajtuk az operandusok értékét, a forgatókarok használatának módja határozta meg azt, hogy milyen műveletet végezzen el a gép. Az eredményt a felhasználó sok esetben a gép állapotának megfelelő értelmezésével olvashatta le. Ennek következtében gyakori volt a számolási hiba, főleg akkor, ha a felhasználó a számológép használatában gyakorlatlan volt.

A mechanikus berendezéseket a múlt század hetvenes éveitől lassan felváltották az elektromechanikus majd az

5.10. ábra. *Odhner számológépe*

elektronikus asztali, illetve zsebszámológépek.

Manapság számológépként már az okostelefonokat használjuk.

5.2. A számítógépek kialakulása

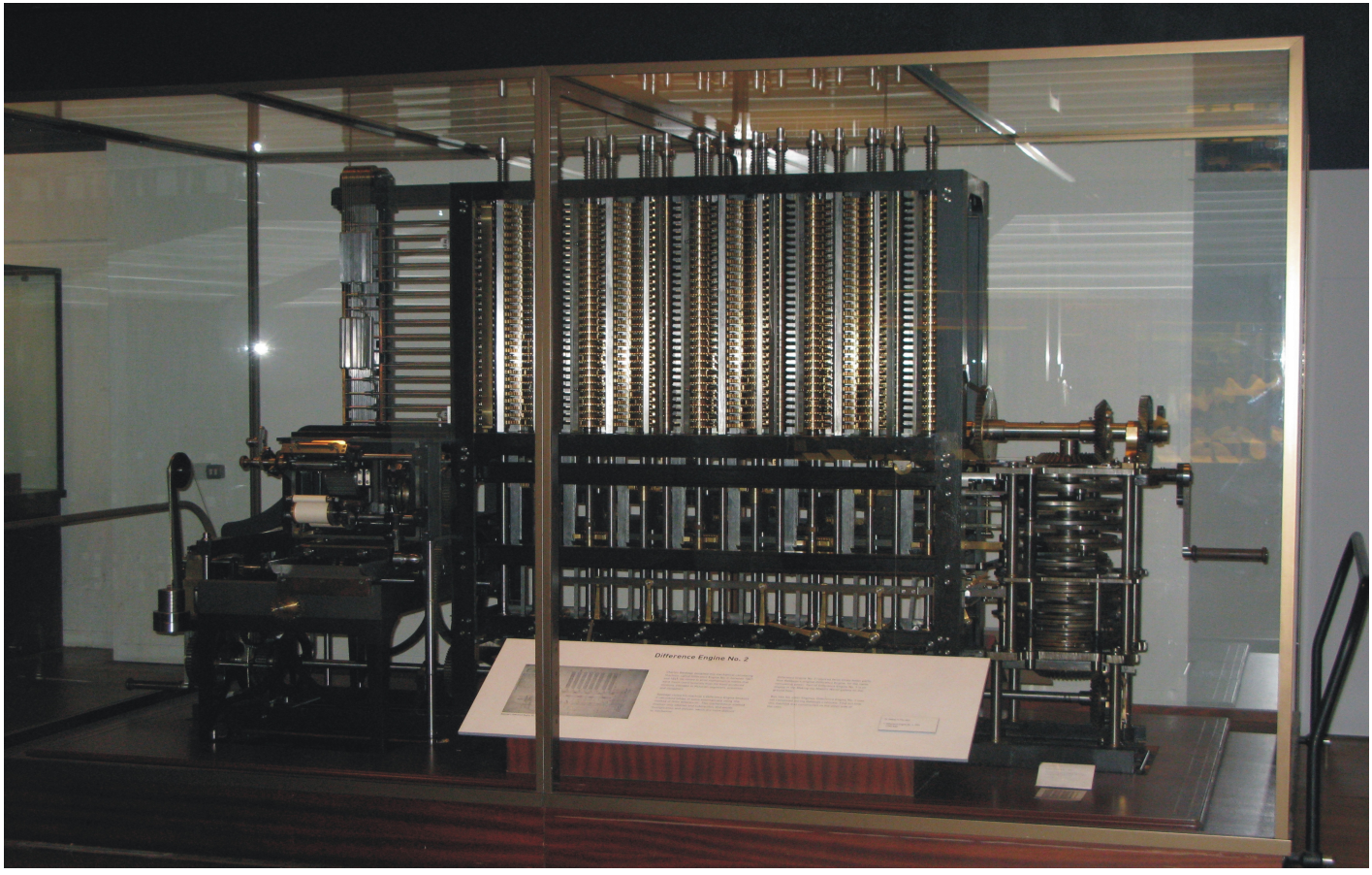
5.2.1. A mechanikus eszközök

A számoláson alapuló ipari berendezések tervezésének és gyártásának kezdetei Joseph Marie Jacquard (1752–1834) francia feltaláló nevéhez köthetők. Ő alkalmazott először a szövetgyártásban olyan gépeket, amelyek előre megállapított minta szerint voltak képesek elvégezni a szövést. A gépek vezérlésére lyukkártyákat használt. Ezek a kártyák kartonlapokból készültek, és különféle lyukkombinációk határozták meg, hogy a gépen milyen mintázatú szövet készüljön. Mondhatnánk azt is, hogy a lyukkártya tartalmazta azt a „programot”, amely meghatározta a szövőszék működését, így ezek a gépek a programmal vezérelt számítógép előfutárainak tekinthetők.

Az 1800-as évek közepén Charles Babbage (1791–1871) angol matematikus volt az első, aki olyan számológép megépítésére gondolt, aminek működése már sokkal kisebb mértékben függött az emberi tényezőktől. Első gépét a különböző hajózási és csillagászati táblázatok újraszámolására tervezte, mivel tapasztalata szerint ezek a táblázatok, bár adataikat számológéppel számolták ki, de az emberi tényezők miatt sok és esetenként durva hibákat is tartalmaztak. Gépe, amit differenciálgépnek nevezett, a matematikából jól ismert véges differenciák segítségével, csak összeadást végezve készítette volna el a táblázatokat – Babbage elképzelése szerint teljesen automatikusan. Mechanikus gépet tervezett, amit gőzgép hajtott volna. Sajnos a kor technikai fejlettsége nem tette lehetővé, hogy a terv megvalósuljon, bár a gép majdnem elkészült, csak működésére a technikai fejletlenség miatt nem volt lehetőség. A tapasztalatok alapján Babbage egy újabb gépet tervezett, sajnos ezt sem sikerült megépítenie. (A 20. század technikáját felhasználva az 1980-as évek közepén az eredeti tervek alapján elkészült gép tökéletesen működik.)

A sikertelenség nem vette el a kedvét. Újabb, most már nemcsak összeadásra, de a másik három alapművelet elvégzésére is alkalmas gépet tervezett. Analitikus gépnek nevezte el. Megvalósítani sajnos ezt sem tudta, de a tervekben olyan elveket alkalmazott, amelyeket a mai modern számítógép-építés során is használnak. A számítások vezérlését a Jacquard-féle lyukkártyával képzelte el. A részeredmények megőrzésére önálló részegységet tervezett. A négy alapművelet elvégzésére alkalmas műveletvégző egysége volt. Az eredményeket nyomtatni lehetett volna. Azaz gépe programozható, bemeneti és kimeneti egységgel volt ellátva, memóriával és műveletvégző egységgel rendelkezett. Ezek a részegységek és a programozható tulajdonság a mai modern számítógépeknél is megtalálhatók. Azt lehet mondani, hogy ez lehetett volna – ha megvalósul – az első olyan gép, amit számítógépnek – angolul computer – nevezhetünk.

5.11. ábra. *Charles Babbage*



5.12. ábra. *Babbage számítógépe (reprodukció)*

Programozásához Ada Lovelace (1815–1852) fűzött értékes megjegyzéseket, amelyek Babbage-hez írt leveleiből váltak ismerté. Erről a hölgyről nevezték el az ADA programozási nyelvet.

A sikertelenség és a technikai színvonal elégtelenségének felismerése meggátolta a mechanikus számítógépek fejlesztését. Egészen 1938-ig kellett várni az első és egyben az utolsó működőképes mechanikus számítógép elkészítéséig. Bár a mechanikai elemek mozgatására már elektromotorokat használt Kondrad Zuse (1910–1995) német mérnök, aki a gépet megtervezte és megépítette. Ez volt a Z1 nevű számítógép. Mivel Zuse a náci Németországban dolgozott, ezért nem volt igazán kedvező a légkör arra, hogy kutatásai nemzetközi elismerésben részesüljenek. Érdemeit az 1950-es, 60-as években kezdték elismerni.

A fejlődés mellékágát jelentette, ennek ellenére fontos eredményeket hozott, ezért meg kell említeni a statisztikai adatok feldolgozásának gépesítése terén elért sikereket is.

Az egyik ilyen statisztikai feladat az országok népességi adatainak kezelése volt. A népszámlálások adatainak feldolgozása kézi eszközökkel olyan sok időt vett igénybe, hogy amire elkészültek a megfelelő táblázatokkal, már a bennük szereplő adatok nem feleltek meg a valóságnak.

Az 1800-as évek végén Herman Hollerith (1860–1929) amerikai statisztikus éppen ezeknek a feladatoknak az elvégzésére szerkesztette meg gépét, amely Jacquard lyukkártyás módszerének alkalmazásával a lyukkártyákra rögzített adatok csoportosítását rendezését végezte el, a kézi módszerhez szükséges időnek töredéke alatt. Ez a berendezés tekinthető az adatfeldolgozó gépek ősének.

Hollerith a gépek gyártására vállalkozást is alapított, amely vállalkozásból nőtt ki a mai számítógéppiac egyik vezető cége az IBM (International Business Machines Corporation).

5.2.2. Elektromechanikus számítógépek

A technikai fejlődés a 20. század elejétől olyan léptékűvé vált, hogy a tudomány minden területén, így az elektronikában is szinte naponta jelentek meg újabb és újabb felfedezések. Ezek lehetővé tették, hogy a mechanikai eszközök egy része elektromechanikussá, majd később elektronikussá váljon. Így történt ez a számítógép esetén is. A jelfogó (relé), majd az elektroncső megjelenése új lendületet adott a számítógép-építésnek.

Az első komoly eredményeket Zuse érte el a már említett mechanikus Z1-nek elektromechanikus átalakításával: Z2 nevű gépe 1940-ben készült el. Az 1941-ben megépült Z3-ban már mindent megtalálunk, amit a mai modern számítógépek legalapvetőbb tulajdonságaiként elvárunk. Minimális változtatásokkal továbbfejlesztett változata a Z4 1945-ben készült el, és ez volt az első kereskedelmi forgalomba került számítógép.

Az utóbbi két gép bináris számrendszert használt, műveletvégző egységük alkalmas volt valós számokkal való műveletvégzésére is, memóriával rendelkeztek, programozhatók voltak. Fő egységeiket a telefonközpontokban is használt jelfogókból építette Zuse. Szinte csak ez az oka annak, hogy nem ezt a gépet tartjuk a kor legmodernebb számítógépének.

Az Egyesült Államokban a már említett IBM cégnél is megindult a számítógépek fejlesztése. 1944-ben befejezték és a Harvard Egyetemre szállították első számítógépüket az IBM Automatic Sequence Controlled Calculatort (ASCC). A tervező Howard Hathaway Aiken (1900–1973) elektromechanikus egységekből – kapcsolókból, relékből – építette gépét, amit a működtető egyetem később Mark I.-nek nevezett el. A gép az Amerikai Tengerészeti Hivatal megrendelésére végzett számításokat. A működését program vezérelte. A programlépések egy szalagra lyukasztott lyukkombinációval voltak kódolva, ezeket olvasta a számítógép, és lépésenként hajtotta végre. Amikor egy lépés végrehajtását befejezte, a következő lépés kódját elolvasva folytatta a munkát. A számítások elvégzésére decimális kódolást használt.

Továbbfejlesztett változata a MARK II. 1946–1947-ben készült el, még mindig kapcsolók és jelfogók felhasználásával.

5.3. Elektronikus számítógépek

5.3.1. Technikai és tudományos alapok

Az elektroncsövek modernebb változatai – bár melegedésük és ennek következményeképpen meghibásodásuk még mindig komoly problémát okozott a számítógép-tervezőknek és -építőknek –, a 20. század negyvenes éveire mégis alkalmassá váltak arra, hogy a számítógépekben alkalmazott jelfogókat helyettesítsék. Így megteremtődött a teljesen elektronikus számítógépek létrehozásának technikai feltétele.

Az elméleti megalapozás is megtörtént ugyanebben az időszakban. Sok neves számítógép-tervezéssel foglakozó tudós nevét felsorolhatnánk, de csak Turing munkásságát említjük. A harmincas években publikált cikkeiben leírta a számítógép olyan elméleti modelljét – az úgynevezett Turing-gépet –, amely mind a mai napig a legteljesebb leírása az absztrakt számítógépnek, illetve a legjobb modellje a mai számítógépeknek. Erre a modellre épül az algoritmus mai értelmezése is, ugyanis minden olyan számítási eljárást, amit Turing-gép képes megvalósítani algoritmusnak nevezünk, és nincs olyan számítási eljárás, amit nem tudnánk Turing-géppel megvalósítani.

Alapvető eredményeket publikált a „számítógépes gondolkodás” – a mesterséges intelligencia – területén; a róla elnevezett Turing-teszt a gépi intelligencia mérésére ma is használatos. Már a negyvenes években olyan műveletvégző egységet képzelt el a számítógéphez, amelynek megvalósítása a mai számítógép-fejlesztőknek is problémát okoz.

5.3.2. A fejlődés főbb állomásai

Az első elektronikus számítógépet az USA-ban, Iowa államban készítették. Tudományos célokra – nevezetesen lineáris egyenletrendszerek megoldására – 1937-ben kezdte építeni John Vincent Atanasoff (1903–1995) és Clifford Berry (1918–1963), 1942-re készült el az Iowai Főiskolán. A gép teszteredményei sikeresek voltak, de működése, elsősorban a részeredmények tárolása megbízhatatlan volt. Amikor Atanasoff a második világháború alatt otthagyta a főiskolát, a fejlesztések megszakadtak.

5.13. ábra. *Allen Turing és az Enigma*

Gépükről hosszú ideig a számítógépekkel foglalkozó társadalom alig tudott valamit, mígnem az 1960-as években felelevenítve az alkotók eredményeit megépítették a gép másolatát. Ekkor vált nyilvánvalóvá, hogy ez az első számítógép, amely elektronikus működésű, bináris rendszerben számol, és az adattárolás elkülönül a végrehajtástól, azaz külön részegysége volt a tárolásra, külön részegység szolgált a

műveletvégzésre is. Programozni nem lehetett, csak egyenletrendszerek megoldására volt képes.

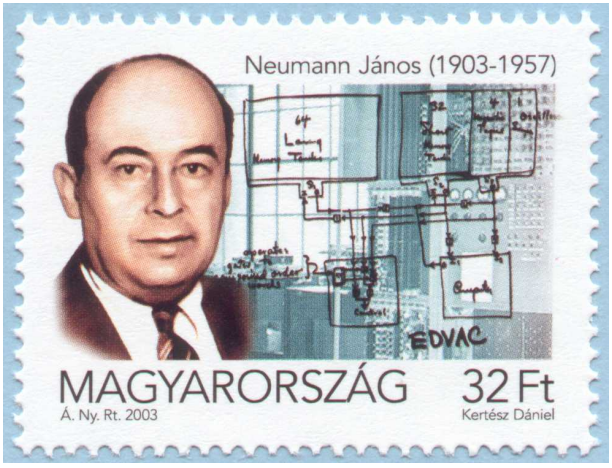
Az elektronikus számítógépek gyors kifejlesztésére a technikai lehetőségek mellett nagy jelentősége volt a második világháborúnak is. Az egymással szembenálló felek terveinek megismerése és megelőzése ugyanis fontos szerepet játszott a háború kimenetelében. A megismerés alapfeltétele a titkos üzenetek megfejtése volt. A németek az ENIGMA nevű rejtjelező géppel kódolták üzeneteiket, és az így rejtjelzett üzenetet megfejthetetlennek tartották. A kriptoanalízissel foglalkozó angol tudósok – köztük Alan Mathison Turinggal (1912–1954) –, részben Turing eredményei alapján, Thomas „Tommy" Harold Flowers (1905-1998) tervei szerint már 1943–44-ben megépítették és használták a Colossus nevű elektronikus számítógépet, amit az ENIGMA-kódú üzenetek megfejtésére használtak – sikerrel. Több tucatot készítettek belőle, és csak kódfejtésre használták. A gépet és terveit titokban tartották, ez a titkosítás oly sikeres volt, hogy a német kódolók még az 1950-es években is használták az ENIGMA-kódot, és még akkor sem tudtak arról, hogy az angolok meg tudják fejteni.

A gépet programozni lehetett, de programozása körülményes volt, mivel kapcsolók, kábelek segítségével lehetett beállítani a programot. Ez a körülményes programozási módszer a továbbfejlesztett változatokban is megmaradt.

Háborús célok motiválták az amerikai fejlesztőket is. Az Egyesült Államok Hadseregének Ballisztikai Kutatások Laboratóriuma tüzérségi táblázatok számítására alkalmas számítógép tervezését és építését rendelte meg a Pennsylvaniai Egyetem Moore Intézetében. A tervezők John William Mauchly (1907–1980) és John Adam Presper Eckert (1919–1995) voltak. A katonai finanszírozás és felhasználás miatt a tervezés, az építés és a munkába állítás is titokban történt. Csak 1946-ban vált publikussá a kutatás, és csak ekkor ismerhette meg a világ az ENIAC (Electronic Numerical Integrator And Computer, magyarul: Elektronikus Numerikus Integrátor és Számítógép) nevű gépet, amely decimális kódolást használt, programozható volt, bár a programozása a Colossuséhoz hasonlóan kapcsolókkal és kábelek segítségével történt. Elektronikus elemekből építették meg, de minimális mértékben reléket is tartalmazott. Fontos tulajdonsága volt a gépnek, hogy nem konkrét feladat elvégzésére, hanem általános célokra készült, ami praktikusan azt jelentette, hogy nemcsak a tüzérségi táblázatok kiszámítására volt alkalmas, hanem minden olyan feladatot képes volt megoldani, amire be tudták programozni. Ezt a tulajdonságot Neumann János (1903–1957) tanácsadói munkájának eredményeképpen

tartják számon. Ő a Los Alamosban zajló atomkísérletekhez szükséges számítások elvégzésére és tesztek kiértékelésére használta a gépet.

Még az ENIAC-projekt befejezése előtt a gép tervezői – ugyancsak az Egyesült Államok Hadserege és a Pennsylvaniai Egyetem Moore Intézete közötti szerződés keretében – 1944 augusztusában új fejlesztésbe kezdtek. Munkájukhoz tanácsadóként csatlakozott Neumann János is. Céljuk egy új számítógép, az EDVAC (Electronic Discrete Variable Automatic Computer, magyarul Elektronikus Diszkrét Változójú Automatikus Számológép) megtervezése és megépítése volt. Az ENIAC kifejlesztése során nyert tapasztalatok továbbfejlesztésével egy elektronikus, bináris kódolást alkalmazó, belső programvezérlésű számítógép megépítésébe fogtak, amelyet 1948-ban már elkészítettek, de csak 1949-ben került a Ballisztikai Kutató Laboratórium birtokába. Elődjétől elsősorban abban különbözött, hogy bináris kódolású volt, nem voltak elektromechanikus alkatrészei, és a működését meghatározó programot a tárolóegységében kódolva tárolta.



5.14. ábra. *Neumann János*

Neumann a tervezés eredményeit, illetve saját javaslatait az 1945. júniusi dátumú „First Draft of a Report on the EDVAC” (Az EDVAC-jelentés első vázlata) című tanulmányában foglalta össze. A jelentésben egy olyan számítógépet ír le, amely az 1970-es évekig meghatározta a számítógép-tervezés és -építés irányelveit. Megjegyzendő, hogy a Neumann által leírt gép soha nem épült meg. A ténylegesen elkészült EDVAC, ha nem is sokkal, de eltér a vázlatban megadott számítógéptől.

5.3.3. A Neumann-elvek

Az EDVAC-jelentésben Neumann a számítógépének nemcsak a tervét adta meg, hanem azt is megfogalmazta, és javaslatait meg is indokolta, hogy mit vár el egy számítógéptől.

1. A gép digitális legyen, azaz kódolt számjegyeket tudjon kezelni, mégpedig bináris számjegyeket.
2. A számítógép teljesen elektronikus legyen, azaz a fő részegységei ne tartalmazzanak mozgó (mechanikus) alkatrészeket.
3. A gép a következő részegységekből épüljön fel:
   - Legyen egy aritmetikai műveletek elvégzésére alkalmas egysége. Ezt nevezhetjük központi aritmetikai egységnek, Neumann elnevezése szerint „Central Arithmetical part”, azaz CA.
   - Legyen egy, a gép működését vezérlő egység, ami egymás után hajtja végre a gép működését meghatározó utasításokat. Az egymásutániság nem a tárolás szerinti sorrendre vonatkozott, hanem azt jelentette, hogy a sorra kerülő utasítással addig ne foglalkozzon a vezérlőegység, amíg az éppen feldolgozás alatt lévőt be nem fejezte. (Ennek a tulajdonságnak a megkövetelése a kor technikai lehetőségei miatt történhetett, nem pedig elvi megfontolásból, hiszen Neumann alig pár év múlva a párhuzamos végrehajtást, mint elvi lehetőséget is vizsgálta.) Az egységnek neve „Central Control”, azaz CC. A CC és a CA egység együtt a vezérlő (Control), röviden C.
   - Legyen egy olyan egysége, amelyben tárolni lehet a működést meghatározó utasításoknak, valamint a számoláshoz szükséges adatoknak, részeredményeknek a kódjait. (Természetesen digitálisan!) Neve „Memory”, azaz M.
   - Tudjon a gép a környezetével kapcsolatot létesíteni, azaz legyen egy olyan részegysége, amely képes a környezettel kommunikálni. A környezetből kapott jelek és az oda küldött jelek az R nevű egységcsoporton (outside Recording medium, azaz külső adathordozó) vannak, lesznek valamilyen formában kódolva. Ezek egy része az ember számára is érthető kódolású, más részük digitális kódú, amit csak visszakódolva tehetünk olvashatóvá. A külső adathordozókat két csoportba sorolta. A

bemenő jeleket vagy kódjaikat kezelő egységeket I-nek (Input, azaz bemenet), a kimenő jeleket vagy kódjaikat tartalmazó egységeket O-nak (Output, azaz kimenet) nevezte.

- A fenti elemek működését szinkronizálandó legyen „órája” a gépnek, amely a zongoristák metronomjához hasonlóan ütemezi a gép működését, ütemenként elkülönítve a végrehajtandó lépéseket.

4. A gép működtetését bináris kódokkal leírt utasítások végrehajtásával képzelte el, amiket a memória tartalmazott. A memóriában lévő utasítások cserélhetők voltak, így a gép minden olyan feladat végrehajtására képes volt, aminek megoldásához utasítássorozatot lehetett készíteni. Ez azt jelenti, hogy a tervezett berendezést Neumann univerzális, belső (program)vezérlésű számítógépként képzelte el.
5. A memória mellett a C egység részeként olyan átmeneti tárolókat tervezett, amelyek az utasításokhoz kapcsolható kódokat, az elvégzendő műveletek operandusainak kódjait és az eredmény kódját tartalmazták.

Ezek után részletesen leírja a tároláshoz használt kódolást. Egyértelműen a bináris kódok mellett teszi le a voksot. Az egész számok kódolására a bináris komplemens kódot javasolja megmutatva előnyeit. Megadja a valós számok egyfajta normalizált rendszerű kódolását is, amely hasonlított ugyan de nem egyezett meg a megismert IEEE 754–1985, IEEE 854–1987, IEEE 1596.5–1993 szabványok által javasolt kódolásokkal.

Felsorolja a CA által mindenképpen elvégzendő műveleteket. Ezek: az összeadás, a kivonás, a szorzás és az egészosztás. Felsorol még néhány más műveletet is, amelyek elvégezhetőségét fontosnak tartja. Megtervezi a CA elektronikus összetevőit, és megadja működési elvüket is.

Megadja azt is, hogyan legyen megszervezve az M.

Végül kitér arra is, hogy a definiált egységek kapcsolata hogyan valósuljon meg, hogyan mozogjon az egyes egységek között a kód.

A tanulmány alkalmat adott az utókornak arra, hogy a benne összefoglalt tulajdonságokat Neumann-elveknek, az ezek alapján felépített számítógépeket Neumann-elvű számítógépnek tekintse. Meg kell jegyezni azt is, hogy szinte minden számítógéppel foglalkozó szakember más és más tulajdonságokat számít a Neumann-elvek

csoportjához. Ez nem is meglepő, hiszen gondoljuk csak végig: a Neumann-jelentés keletkezésének idején a javasolt számítógép minden tulajdonsága megjelent valamelyik tervező elképzelésében, sőt zöme meg is valósult valamelyik szerkezetben. Éppen emiatt a legtöbb kutató Neumann-elvként egyetlen tulajdonságot szokott kiemelni, mégpedig a belső programvezérlés elvét, amit tárolt program elve néven is szokás emlegetni.

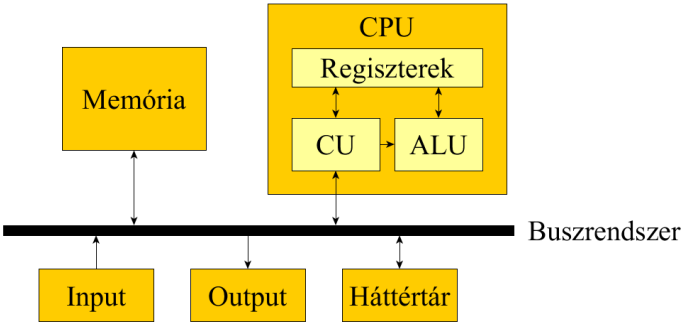
Mivel a neumanni elveknek megfelelő számítógép, az EDVAC építése 1948-ban befejeződött ugyan, de csak 1949-ben helyezték üzembe, ezért mindenképpen említést kell tenni a manchesteri Victoria Egyetemen épített Manchester Small-Scale Experimental Machine (SSEM) – becenevén Baby – számítógépről, amely a belső programvezérlés elve alapján működött. Tervezői, építői Frederic Calland Williams (1911–1977), Tom Kilburn (1921–2001) és Geoff Tootill (1922–) voltak. Mivel 1948-ban már üzemszerűen használták, ezért ez a gép tekinthető az első belsőprogram-vezérlésű számítógépnek. Nem köznapi használatra szánták, inkább laboratóriumi tesztberendezésnek, kísérleti eszköznek készült, mivel elsődleges célja egy Williams-cső nevű elektroncsövekből készült memória kipróbálása volt. Ennek megfelelően primitív berendezésnek számított, és működése sem volt megbízható.

5.3.4. A Neumann-elvű számítógép felépítése és működése, a Neumann-architektúra

Az 1940-es évek végén lezárult a számítógép-építés történetének őskora. Minden készen állt a nagyüzemi gyártáshoz, és elérkezett az idő, hogy a számítógép, ha lassan is, de a mindennapi életünk eszközévé váljon.

Neumann terveihez igazodva, némely praktikus változtatással a számítógép logikai felépítést az 5.15. ábra tartalmazza. Az építőelemeket, illetve a számítógépet alkotó berendezéseket közös néven hardvernek nevezzük. Az ábrán megjelölt egységek funkciója, felépítése és működése a következő:

- Memória
  A számítógép működéséhez szükséges kódokat tárolja. A Neumann elvekhez illeszkedve elektronikus működésű, és kettes számrendszerben megadott kódokat tartalmaz. Legkisebb elemi egysége két stabil állapottal rendelkezik. Ezek az állapotok egymásba átvihetők. Az átkapcsolás az egységre ható külső hatásra következhet be. Az egyik állapot 0-nak, a másik 1-nek tekinthető, így az elemi egység egy bit tárolására alkalmas.



5.15. ábra. *A Neumann-elvű számítógép felépítésének sematikus ábrája*

Az elemi cellákat, mivel egy bináris számjegy tárolására képesek, szintén bitnek nevezzük. (Vigyázat, a bit kétféle jelentését ne keverjük!) Bár elméletileg semmi akadálya, hogy a memóriát elemi cellákkal kezeljük, a gyakorlatban sokkal hatékonyabb, ha ezeket nagyobb tárolási egységbe szervezzük. Leggyakoribb a nyolc elemi cella alkot egy egységet, ezt bájtnak (byte) hívjuk

A kódolásnál megismert kódrendszerekben nyolc bitnél hosszabb kódokat is használtunk, így célszerű a memóriában a bájtnál nagyobb egységeket is definiálni. Ilyenek a félszó, ami két bájtból, a szó, ami négybájtos, a duplaszó, ami nyolc bájtból áll.

A memória méretét ugyancsak bájtokkal szokás kifejezni. Mivel a mai memóriaméret jócskán meghaladja az 1950-es évek számítógépeinek memóriaméretét, ezért a bájt mellett célszerű ennek többszöröseit a memória méretének megadására bevezetni. A számítástechnikai váltószám az egyes mértékegységek között az 1024, ami 2^{10}-nel egyenlő. A SI (Système International d'Unités = Mértékegységek Nemzetközi Rendszere) rendszer, amit az egész világon alkalmaznak, tízes alapú mértékegység-rendszer. A megállapodások szerint számítástechnikában is ezt kellene használni, de kompromisszumos megoldásképpen használatos a kettes alapú rendszer is. A hétköznapi életben gyakran összekeverik a kétféle rendszert, és csak az SI mértékegységeket használják akkor is, ha a méret bináris

rendszerben van.
A memória méretei általában a bináris rendszer szerint vannak megadva, a mértékegységet viszont a tízes rendszer szerint használják – helytelenül. (Megabyte helyett mebibyte-ot kellene mondani, illetve az MB helyett MiB jelet kellene használni.) A háttértárak méretét viszont általában SI egységekben adják meg, és ott – helyesen – megabyte, gigabyte, terabyte a mértékegység, MB, GB, TB a jel.

5.14. táblázat. *A tárolókapacitás bináris rendszere*

A mértékegység neve bináris rendszerben	Nagysága	Jele
bit	egy bináris jegy tárolóegysége	b
bájt (byte)	8 bit (8 b)	B
kibibájt (kibibyte)	1024 bájt (1024 B)	KiB
mebibájt (mebibyte)	1024 kibibájt (1024 KiB)	MiB
gibibájt (gibibyte)	1024 mebibájt (1024 MiB)	GiB
tebibájt (tebibyte)	1024 gibibájt (1024 GiB)	TiB
pebibájt (pebibyte)	1024 tebibájt (1024 TiB)	PiB

Az 1950-es évek számítógépeinek memóriamérete legfeljebb párszáz bájt volt, manapság a memória nagyságát gigabájtokban (helyesen gibibyte lenne) mérik.

Aktivitás: Becsüljük meg, vajon 1 TiB hányszorosa 1 TB-nek?

A tárolóegységek azonosítását sorszámozással oldják meg. A sorszámozást nullával kezdik, és minden egymást követő egység eggyel nagyobb sorszámot kap. A bájtszervezésű memóriában bájtok, a szószervezésűben a szavak kapják a sorszámot. A sorszámokat szintén kettes rendszerben többféle módszerrel is lehet kódolni. Az egyik ilyen módszer a direkt címkódolás, amelyben a címeket egyszerűen kettes számrendszerbe megadva kódolják. Fix hosszúságú kód esetén a kettes számrendszerbeli alak elé

5.15. táblázat. *A tárolókapacitás decimális (SI) rendszere*

A mértékegység neve SI-ben	Nagysága SI-ben	Jele
bit	egy bináris jegy tárolóegysége	b
bájt (byte)	8 bit (8 b)	B
kilobájt (kilobyte)	1000 bájt (1000 B)	kB
megabájt (megabyte)	1000 kilobájt (1000 kB)	MB
gigabájt (gigabyte)	1000 megabájt (1000 MB)	GB
terabájt (terabyte)	1000 gigabájt (1000 GB)	TB
petabájt (petabyte)	1000 terabájt (1000 TB)	PB

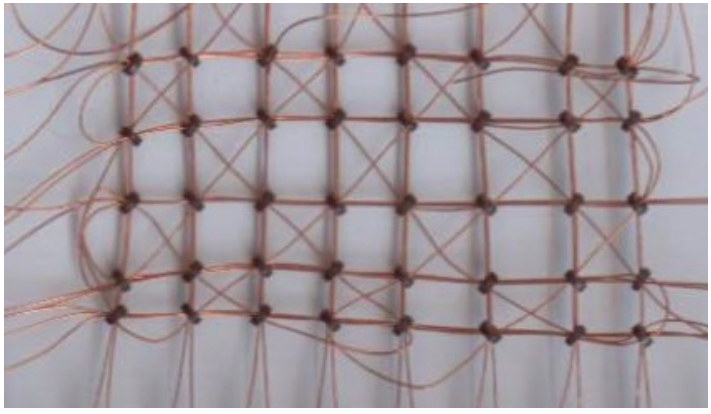
– ha szükséges – annyi nullát kell írni, hogy a megfelelő hosszúság kialakuljon. A kód hossza a memória méretével definiálható. A képlet: $\lceil \log_2(M) \rceil$, ahol M a memória mérete szervezőegységekben $\lceil x \rceil$ pedig x-től legkisebb mértékben eltérő, x-nél nem kisebb egész szám. Ha a nagy memória miatt ez a kód túl hosszú lenne (túl hosszú például, ha 16 bitnél hosszabb), akkor másféle kódolást alkalmazva a cím kódja rövidíthető.

A memória szerepe: a kódok tárolása. Neumann szerint ezek a kódok jelenthetik a számítógépet vezérlő utasításokat, de jelenthetik a számoláshoz szükséges adatokat is. Magáról a kódról általában nem eldönthető, hogy mit jelent. A számítógép vezérlő egységének kell tudnia, hogy utasítás vagy adat kódjának tekintünk-e egy kódot.

A memóriát kezdetben elektroncsövekből, később mágnesezhető apró gyűrűkből (lásd: 5.16. ábra) stb. építették. Később kizárólagossá vált az integrált áramkörös megvalósítás.

5.1. definíció: A programkódot is tartalmazó memóriát operatív memóriának szokás nevezni.

- CPU (Central Processing Unit, magyarul Központi Végrehajtó Egység, Neumann-terminológia szerint C)



5.16. ábra. *Ferritgyűrűs memória*

Neumann két fontos részből állónak tekintette. Célszerűnek látszik e két egység (CU – neumanni terminológia szerint CC – és CA) mellett kiemelni, és harmadik fontos összetevőként kezelni a CPU néhány bájtból álló tárolóegységét, a regisztertömböt is.
Mivel Neumann csak numerikus értékekkel végzendő számolásra gondolt, amikor a számítógépet tervezte, ezért a számolómű nevében is csak az aritmetikai jelzőt szerepeltette. Már abban a korban is voltak olyan számítógépek, amelyek logikai műveletek elvégzésére is képesek voltak, ezért a CA helyett az ALU (Arithmetic and Logic Unit) megnevezést fogjuk használni.
A CPU tehát a CU és az ALU egységből áll, amik elérik a CPU részét képező regisztertömböt is. Részletesebben:

– Regisztertömb

  Speciális tárolóegységek rendszere. A tárolóegységek mérete a tárolt kódok nagyságának, a memóriaszervezésnek, a műveletvégző egységnek paramétereitől függ, egy bájtnál semmiképpen nem rövidebbek, de a mai számítógépekben lehetnek akár nyolcbájtosak is.

Néhány fontosabb egység a regisztertömbben:

* Utasításregiszter, röviden IR (Instruction Register): az éppen végrehajtás alatt álló utasítás kódját tartalmazza.
* Utasításszámláló regiszter, röviden PC (Program Counter): egy memóriacím kódját tartalmazza. Aktuális értéke alapján kerül a megfelelő utasítás az utasításregiszterbe. (Ez a PC rövidítés nem tévesztendő össze a personal computer PC rövidítésével!)
* Akkumulátorregiszter, röviden A (Accumulator): Az ALU által előállított eredmény kódját, és sok esetben aritmetiai-logikai műveletek egyik operandusának a kódját is tartalmazza.
* Státuszregiszter röviden SR (Status Register): minden bitje más és más információt kódol. Egyik bitjéből azt lehet kiolvasni, hogy a művelet eredménye negatív-e vagy sem, a másikból azt, hogy nulla-e vagy sem, a harmadikból azt, hogy az eredmény beletartozik-e a kódolható tartományba vagy sem, stb.

Mivel ezt a tárolót szinte folyamatosan használja a CU és az ALU, ezért gyorsnak és megbízhatónak kell lennie.

– ALU

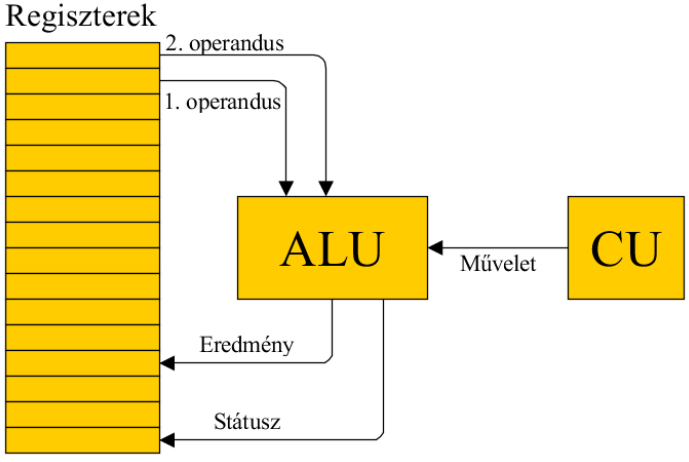
Azokat a műveleteket végzi, amelyek elvégzésére a CU-tól utasítást kap. A műveletekhez szükséges adatok kódjait szintén a CU készíti elő a regisztertömb megfelelő regisztereinek kitöltésével. A művelet elvégzése után az eredmény szintén valamelyik regiszterbe kerül. A műveletvégzés eredményének kódján kívül néhány más kód is keletkezik, ezek a kódok alakítják ki a státuszregiszter megfelelő bitjeit.

Az ALU által elvégezhető műveleteket a következő csoportokba sorolhatjuk:

* Aritmetikai műveletek. Ezek további két csoportba sorolhatók.
  · Az egész számok kódjaival elvégezhető úgynevezett fixpontos műveletek.
  · A valós számok kódjaival elvégezhető úgynevezett lebegőpontos műveletek. Ezeket a régi ALU-k még nem tudták elvégezni.
* Logikai műveletek. Ezeket a műveleteket ismertük meg a Boole-algebra című fejezetben.

* Léptető műveletek. Ezek a kódok bitjeit jobbra vagy balra tolják úgy, hogy a kódból kieső bitek a kódból elvesznek, az eltoltak helyébe nulla vagy egy kerül, illetve lehet az eltolás ciklikus is, ekkor a kipottyanó bit a kód másik végén jelenik meg.

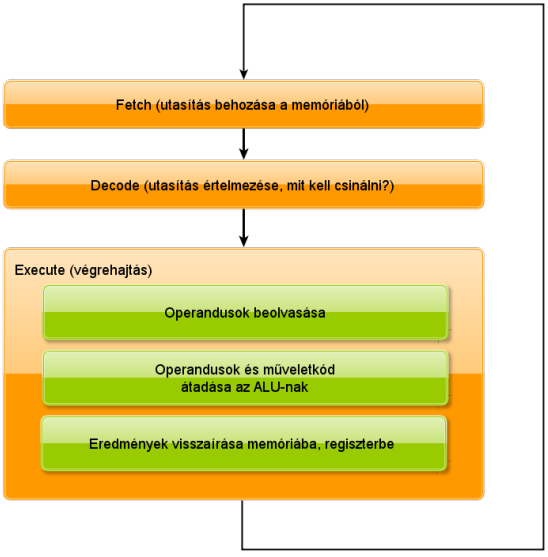
Minél több és összetettebb művelet elvégzésére képes az ALU, annál bonyolultabb áramkörökből épül fel, és természetesen annál drágább is. Egy-egy számítás elvégzésére viszont annál kevesebb művelet kell, minél bonyolultabb a műveletvégző egység. A számítógép-építőknek tehát dönteniük kell egyszerűbb, olcsóbb ALU, így több művelet ugyanannak a számításnak az elvégzésére, vagy bonyolultabb, drágább ALU kevesebb műveletvégzéssel. Napjainkban mindkét irányú választásra van példa; az 1950-es években viszont nem nagyon volt választási lehetőség, hiszen az ALU is elektroncsövekből épült fel. A sok elektroncső hamar túlmelegedett, és a gép leállt. Ezért még a látszólag egyszerű összeadást végző egység is további egyszerűsítésekkel volt csak megépíthető. Az 5.17. ábrán egy kétoperandusú műveletet elvégzésére beállított ALU sematikus rajza látható.



5.17. ábra. *A Neumann-elvű számítógép felépítésének sematikus ábrája*

– CU:

A Control Unit (vezérlőegység) a számítógép alapvető fontosságú részegysége. Működése során néhány egyszerű lépést ismétel ciklikusan (5.18. ábra):



5.18. ábra. *A CU működése*

A vezérlőegység elsőként a memóriából, az utasításszámláló regiszterben lévő címkód alapján beolvas egy utasításkódot, ami az utasításregiszterbe kerül. Miután beolvasta, értelmezi azt, vagyis eldönti, hogy mit kell csinálnia, végrehajtja – ha szükséges az ALU műveletvégző képességét is felhasználva – a felismert utasítást, az eredményt – ha az utasítás alapján erre szükség van – a memóriába juttatja, módosítja a regisztertömb megfelelő regisztereit, majd a következő utasításkód beolvasásával egy újabb ciklus kezdődik.

Ahhoz, hogy ezt a folyamatot pontosabban megértsük meg kell ismernünk a (gépi) utasítás fogalmát. Egy gépi utasítás nem más, mint egy elemi instrukció a CPU számára, amit az felismer, és pontosan úgy működik, ahogyan az instrukció szerint működnie kell. Természetesen egy adott CPU-hoz több különböző instrukció, gépi utasítás tartozik. Ezeknek az utasításoknak az összességét utasításkészletnek, angolul instrucion setnek nevezzük. Ez a készlet a CPU jellemző tulajdonsága, így más-más CPU-nak más-más utasításkészlete van.

Az utasítások binárisan vannak kódolva, ez a kód általában egy állandó és egy változtatható részből tevődik össze. Az állandó résszel lehet megadni, hogy milyen utasítást kódolt a jelsorozat, a változtatható rész adja meg azt, hogy a végrehajtáskor mik lesznek az esetleges műveletvégzés operandusai, ha az utasításhoz nem tartozik operandus, akkor ez a rész elmarad. A kezdetekben csak ezt a kódot használták az utasítás megadására, később minden utasításnak rövid nevet (mnemonikot) adtak, így könnyebb volt megjegyezni, és könnyebb volt összeállítani azt az utasítássorozatot is, amely úgy irányította a számítógép CPU egységét, ahogyan a gépet használó akarta.

A gépi utasításokat a következő csoportokba célszerű sorolni.

* Memóriakezelő utasítások (regiszter vagy memória olvasása, írása, másolása stb.).
* Aritmetikai és logikai utasítások (összeadás, kivonás, osztás, szorzás; bitmanipulációs műveletek, például: eltolás, negáció; összehasonlítás, például: kisebb, nagyobb, egyenlő stb.).
* Vezérlőutasítások (feltételes ugrás, feltétel nélküli ugrás stb.).
* Egyéb utasítások (például energiagazdálkodást szabályzó utasítások, amelyek a mai modern számítógépeken már megtalálhatók).

Nézzük meg egy példán, hogyan is történik a gépi utasítások végrehajtása. Tegyük fel, hogy valamilyen elképzelt számítógép memóriája bájtszervezésű, mérete 64 KiB, az adatok 8 bittel kódolt nemnegatív számok, processzora gépi utasításainak kódhossza egy-, két-, három-, négy-, esetenként ötbájtos lehet. Ismeri az összeadást végző utasítást, aminek végrehajtásakor a CPU a CU és az ALU összehangolt működésével összead két értéket. Az összeadandók kódját vagy a kódok tárolóegységét az utasításban megadható két operandussal lehet kijelölni, az eredményt az első operandusnak megfelelő helyen lehet tárolni.

Mnemonikus formája pedig a következő: ADD op1,op2.

Az op1 és az op2 kétféleképpen jelölheti ki az összeadásban felhasználandó kód helyét; memóriacímmel és regiszternévvel. Az op2 cím helyett az összeadandók egyikének kódja is lehet. Az ADD mnemoniknak ezért több bináriskód-megfelelője lehet, attól függően, hogy az op1 és op2 mit jelent.

Most tegyük fel, hogy az op1 egy memóriacím, az op2 pedig egy konstans. A cím és a konstans megkülönböztetéséhez a konstans elé tett aposztrófot használjuk.

A konkrét utasítás legyen tehát: ADD 1024,'32.

Ezt binárisan 32 bittel kódolva a következő felépítésű kódot kapjuk: az első bájt az utasításkód, a második és harmadik bájt a direkt memóriacím, az utolsó bájt a konstans kódja lesz. Az utasításkód tehát négybájtos, amiben az első bájton a művelet és az operandusok típusa van kódolva. Konkrétan a következő:

10000010 0000000100000000 00100000.

Amikor az utasítás végrehajtása a gép működésekor sorra kerül, ez a kód az IR regiszterbe jut. Praktikusan annyi bájt, amekkora az IR.

Az első nyolc bitből a CU megállapítja, hogy a művelet összeadás, hogy memóriában lévő kód az első operandus, megállapítja, hogy a második operandus kódja az utolsó bájtról olvasható le, és azt is, hogy az eredményt oda kell visszaírnia, ahonnan az első operandus kódját kapja. A PC regiszter értékét – mivel a feldolgozás alatt lévő utasítás négybájtos – 4-gyel megnöveli.

Ezután a 0000000100000000 (=1024) címen lévő kódot az ALU első operandusát tartalmazó regiszterbe, a második operandust tartalmazó regiszterbe a 00100000 kódot tölti, majd utasítást ad az ALU-nak az összeadás elvégzésére. Az eredményt szinten egy regiszterbe kerül.

A negyedik lépésben az eredményregiszter tartalmát (ez egy kód lesz!) a 0000000100000000 címre írja, beállítja a SR-t, majd elkezdi a következő ciklust.

A CPU működési sebességét az egy másodperc alatt végrehajtott utasítások számával mérik. Mértékegysége az IPS (az instructions per second rövidítése). Ennek ezerszerese a kIPS, milliószorosa az MIPS.

A fenti utasítás-végrehajtás példája jól mutatja, hogy hogyan is működik a CPU. Fontos tudni azt is, hogy ezek az utasítások hogyan alkotnak szerves egységet abból a célból, hogy a CPU pontosan úgy működjön, ahogyan a felhasználója szeretné. Másképpen: hogyan kell összerakni a működtető programot, és mi is az a program.

Gépi programnak vagy másképpen gépi kódú programnak nevezzük azt az utasításkód-sorozatot, amit a számítógép memóriájába betölthetünk, és a betöltés után ezek az utasítások vezérlik, irányítják a számítógép CPU egységét, és ezáltal a számítógépet. A programot úgy kell összeállítani, hogy az utasítások megfelelő sorrendben végrehajtva úgy működtessék a CPU-t, ahogyan a felhasználó szeretné. Ennek a munkának az elvégzői, azaz a programok készítői a programozók. A gépi kódokkal programot írni komoly munkát jelentett. Ennek a megkönnyítésére, a gépi kódokat helyettesítő mnemonikok és egyéb más szimbolikus jelek bevezetése az első és egyben legegyszerűbb programíró vagy más szóval programozási nyelv megalkotásához vezetett. Ezt a nyelvet assembly-nyelvnek nevezték, és annyi „nyelvjárása” volt (van), ahányféle CPU létezett. Az assembly nyelven megírt programot a számítógép már nem tudta közvetlenül felhasználni, ezért át kellett alakítani a gépi nyelvre. Az átalakításra programokat használtak. Ezek voltak az assemblerek.

A programozók munkáját tovább egyszerűsítette az úgynevezett magas szintű programozási nyelvek megjelenése. Ezeken a nyelveken az írott nyelvekhez hasonlóan lehet megfogalmazni a programokat. Természetesen a magas szintű nyelven megírt programot sem képes a számítógép megérteni, ezért ezeket is át kell alakítani gépi kódra. Az átalakítást ugyancsak programok végzik, amiket fordítóprogramoknak hívunk. Manapság számtalan magas szintű programozási nyelv létezik.

- Buszrendszer

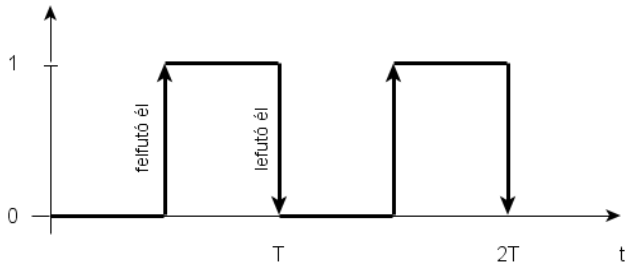
  A számítógép fő részei közötti összeköttetést az elektronikus működés következményeképpen elektromos vezetékek biztosítják. Ezek a vezetékek juttatják el az egyik egységből a másikhoz a működéshez szükséges kódokat.

  Ilyen vezetékek kötik össze a memóriát a CPU-val, az input, output eszközökkel stb. Kezdetben ezeken a rendszereken mindenféle kódot mozgattak, később külön vezetékrendszer készült a memóriacímek, az utasítások, az adatok kódjainak átvitelére. Egy-egy kódcsoporthoz tartozó vezetékrendszert busznak hívunk. Egy számítógép összes busza a buszrendszer. A buszrendszerben nem pont-pont kapcsolattal

mennek a vezetékek, hanem minden busz egy fővezetékből áll, amelyre több egység is csatlakozhat.
A kódok bitjeit kétféleképpen lehet átvinni:

  - Bitenként – egy vezetéken – egymás után. Ezt soros átvitelnek hívjuk.
  - A teljes kódot, vagy annak egy jól meghatározott bitszámú részét annyi vezetéken, ahány bitből áll a megadott rész. Ehhez legalább annyi vezeték kell, ahány bitet egyszerre szeretnénk szállítani. Ezt az átvitelt párhuzamos átvitelnek nevezzük.

- Órajel-generátor
  Egy számítógép megfelelő működéséhez nem csak az fontos, hogy a gépet felépítő elemi áramkörök működjenek, hanem az is, hogy ezek az elemi működések vagy állapotváltozások szinkronban működjenek. Ennek biztosítására a legtöbb számítógépben egy szinkronizáló jelet, elterjedtebb nevén órajelet (5.19. ábra) használnak, amit az órajel-generátor állít elő. Ez az apró áramkör (még az ősi számítógépeken sem volt jelentős méretű) a főbb egységektől függetlenül működik, az előállított jeléhez minden egység hozzáfér. Az órajel egy alacsony (a bináris 0 kódnak felelhet meg) és egy magas (a bináris 1 kódnak felelhet meg) feszültségérték között periodikusan és ugrásszerűen váltakozó folytonos, elektromos jel (5.19. ábra). Fontos tulajdonsága a periódusidő – jele T. A T az mutatja, hogy az órajel-generátor jele bármely időpillanathoz mérten minimum mennyi idő múlva lesz pontosan ugyanolyan értékű. A T reciproka az $1/T$, az órajel frekvenciája, ami azt fejezi ki, hogy egy időegység alatt (pl. másodperc) hány periódus ismétlődik. A mai modern számítógépeken ez az érték már 10^9 (gigahertz, GHz) nagyságrendű is lehet.



5.19. ábra. *Az órajel*

Az áramkörök működését a felfutó és lefutó élnek nevezett 0-1 vagy 1-0 átmenetek szinkronizálják, azaz az órajelben bekövetkező hirtelen feszültségváltozást figyelve képesek a számítógép részei önmaguk és egymás közti összehangolt működésre.
Bár az órajel frekvenciája befolyásolja a számítógép sebességét, mégsem növelhető büntetlenül minden határon túl. Egyrészt a számítógép áramköreinek is képesnek kell lenniük a gyorsabb működésre (általában ez jelenti a korlátot), másrészt a nagyobb működési frekvencia magasabb fogyasztással és hőtermeléssel jár együtt.

- Input és output eszközök
  A számítógép semmit sem ér, ha nem tud a környezetével kapcsolatot teremteni. Ez a kapcsolat kétirányú. Ez azt jelenti, hogy képesnek kell lennie a környezettől adatokat, kérdéseket, stb. fogadni, és meg kell tudni jeleníteni az eredményeket, esetenként a számolás részeredményeit is. Szükség van tehát olyan eszközökre, amelyek megteremtik a kapcsolatot a külvilág és a számítógép között.
  - Bemenő vagy inputeszköznek nevezzük azokat az eszközöket, amelyeken keresztül a számítógép adatokat, instrukciókat fogad. Ezeket inputperifériáknak is szokás nevezni.
  - Kimenő vagy outputeszközöknek nevezzük azokat, amelyeken keresztül a számítógép adatokat küld a

külvilág felé. Ezek az outputperifériák.

Ezeknek az eszközöknek a számítógéphez való csatolására alapvetően kétfajta módszert alkalmaznak:

- A memóriába ágyazott I/O esetén az eszközöket a memória eléréséhez használt buszra kapcsolják. Kód írása vagy olvasása a memóriába való írással illetve a memóriából való olvasással megegyező módon, ugyanazokkal az adatmozgató utasításokkal történik.
- Dedikált I/O esetében az adatforgalom egy külön buszon, külön input, output utasítások hatására jut el a memóriába vagy a CPU regisztereibe.

Korunkban azt a berendezést hívjuk számítógépnek, amelyik struktúrájában és működési elvében hasonlít vagy megegyezik a fentebb vázlatosan ismertetett géppel.

5.3.5. A számítógép működése

Már láttuk, hogy a CU hogyan működik. Azt is láttuk, hogy a CU működése során hogyan szólítja meg az ALU-t, hogyan használja a memóriát, az input, az output, az I/O eszközöket. Most vázlatosan megismerjük a számítógép működését is.

A memória elemei elveszthetik tartalmukat, ha a berendezést kikapcsolják. A belső programvezérlés elvén megépített számítógép viszont csak akkor működik, ha a memóriából a működéshez szükséges utasításhoz hozzájut. Ezért szükség van olyan tárolóegységre, amelyből bekapcsoláskor kiveheti az első működtető utasításokat. Ez az első elektronikus számítógépeken pár bájtból álló kapcsolókból álló tábla volt, ahol a gép kezelője a kapcsolókon állította be az első utasításokat. Ezeket kapta meg a CU, amik hatására felépítette a memóriának a programot tartalmazó részét. Azaz „betöltötte” a működtető gépi programot. A betöltés általában háttértárról történt, de voltak olyan számítógépek is, amikben erre a célra háttértár helyett más, csak input célra alkalmas berendezésről került a memóriába a program. Manapság léteznek olyan memóriák, amelyek nem vesztik el tartalmukat a számítógép lekapcsolása után sem. Bekapcsoláskor erről a memóriatartományból veszi ki a CU az elsőként végrehajtandó utasítást, minden bekapcsoláskor ugyaninnen és mindig ugyanazt. Ennek a memóriának a címét kezdetekben a gépkezelő egy kapcsolótáblán tudta beállítani, később a számítógép

bekapcsolásakor mindig ugyanaz a cím került a PC regiszterbe. Ezen a címen egy olyan memóriarészt lehetett elérni, amelynek tartalma kikapcsoláskor is megmaradt. Ide a gép gyártásakor beírták az indításkor végrehajtandó utasításokat. Ezeknek a végrehajtása során a CU felépítette a további működéshez szükséges memóriatartalmat. A megmaradó tartalommal rendelkező memória mérete a kezdetekben pár bájt volt, manapság több KiB, sőt GiB nagyságrendű.

A működés tehát a következő:

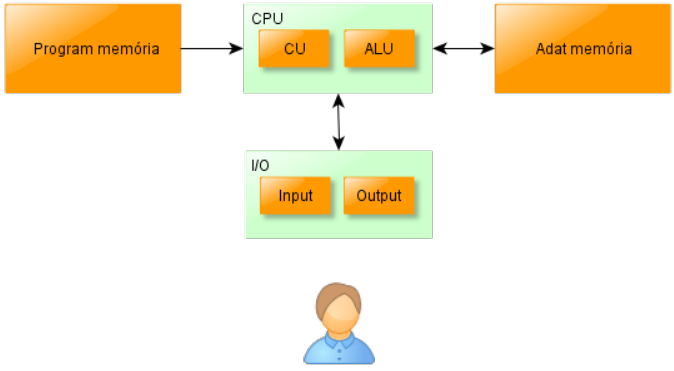
Bekapcsoláskor feszültség alá kerül a számítógép. Ennek a feszültségimpulzusnak a következményeképpen a CPU regisztereiben kezdőérték generálódik. A PC regiszterben lévő cím alapján bekerül az utasításregiszterbe az első gépi utasítás kódja, és a CU a már megismert módon elkezdi ciklikus működését, és ez az állandóan ismétlődő folyamat már a teljes számítógép-működést meghatározza. Ez a működés automatikus, csak akkor van szükség a kezelő beavatkozására, ha a program ezt előírja. (Természetesen a kezelő beavatkozhat a működésbe, amikor csak akar, de ennek gyakran az a következménye, hogy az éppen „futó” program megszakad, és esetleg nem is lesz folytatható.) A számítógép működése kikapcsolással állítható le. A kikapcsolás a fő egységek feszültséggel való ellátásának megszüntetésével lehetséges.

A mai modern gépek, ha éppen várakoznak a gépkezelő beavatkozására, és ez nem történik meg, akkor takarék üzemmódra kapcsolnak, sőt a kezelő utasítására már önmaguk kikapcsolására is képesek.

5.3.6. Harvard-architektúra

A Neumann-architektúrának előnyei mellett voltak hátrányai is. Az egyik ilyen hátránya az volt, hogy a programkódot ugyanott tárolta a számítógép, ahol egyéb más kódokat is. Így előfordulhatott, hogy hibás programozás esetén a gép „átírta” a programot, aminek következményeképpen nem biztos, hogy olyan utasításokat hajtott végre az átírást követően, mint amelyeket a programozó szeretett volna. Ennek a problémának a kiküszöbölésére a memóriát kettéválasztották: külön memória szolgált a programkódok és külön az egyéb kódok tárolására is (5.20. ábra). Az így kialakult architektúra a Harvard-architektúra.

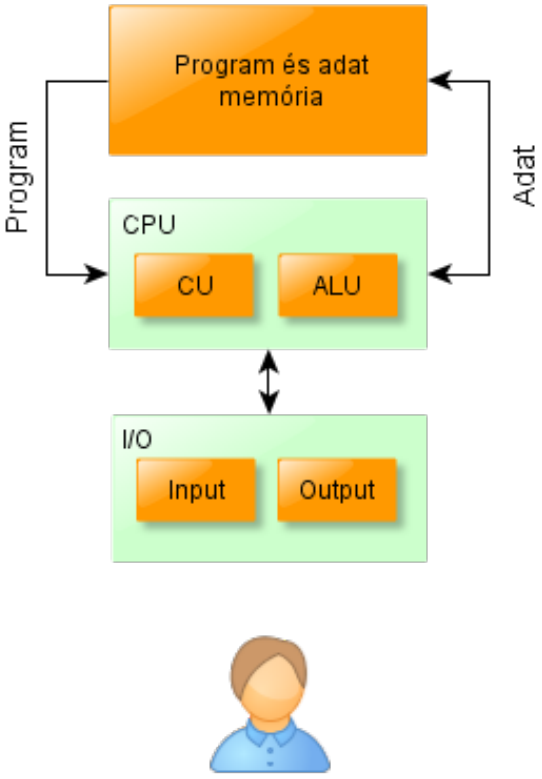
Ennek a szervezési módnak előnye többek között az, hogy a programmemória és a többi kódot tartalmazó memória eltérő szervezésű is lehet, a programkódot és az adatok kódját akár párhuzamosan is lehet a



5.20. ábra. *A Harvard-architektúra*

memóriából a CPU-ba juttatni stb.

A Neumann-architektúra és a Harvard-architektúra ötvözete a módosított Harvard-architektúra, amelyben az összes kódot egyetlen memóriában tároljuk, de a programkód és a többi kód szállítására külön buszok szolgálnak, és ezek a buszok felhasználás előtt két különálló átmeneti memóriába viszik a kódokat. Ebből a két memóriából kerülnek a CPU megfelelő regisztereibe. Fordított irányú kódmozgás ugyancsak elkülönítetten történik, csak a memória közös (5.21. ábra).



5.21. ábra. *A módosított Harvard-architektúra*

Önellenőrzés

1. Melyik állítás igaz?
   - [ ] Az 1 kB ugyanakkora, mint az 1 KiB.
   - [ ] Az 1 KiB nagyobb mint 1 KB.
   - [ ] A CU működési sebességét MIPS mértékegységben is meg lehet adni.
   - [ ] A memória nagyságát Kib-ben szokás megadni.
2. Melyik állítás igaz?
   - [ ] A soros átvitelhez ugyanannyi szálú vezeték kell, mint a párhuzamoshoz.
   - [ ] A soros átvitelhez több szálú vezeték kell, mint a párhuzamoshoz.
   - [ ] A soros átvitelhez kevesebb szálú vezeték kell, mint a párhuzamoshoz.
   - [ ] Az átviteli sebesség mértékegysége a MIPS.
3. Melyik állítás igaz?
   - [ ] A CPU a számítógép egyik fő egysége, ami a CU-t és az ALU-t is tartalmazza.
   - [ ] Az ALU önállóan állapítja meg, hogy milyen műveletet kell elvégeznie egy utasítás végrehajtása közben.
   - [ ] A regisztertömb a memória része.
   - [ ] A memóriába ágyazott input és output eszközök kezeléséhez nincs szükség külön gépi input és output műveletekre.

4. Melyik állítás igaz?

- [ ] Az első elektronikus számítógépet Babbage tervezte.
- [ ] Az EDVAC megépítését megelőző időkben csak elektromechanikus alkatrészekből (relékből) építettek számítógépet.
- [ ] Neumann tanulmányának elkészülését megelőző időkben nem is gondoltak arra, hogy a kettes számrendszer alkalmas lehet a számítógépes kódolásra.
- [ ] Az EDVAC elektroncsöves számítógép volt.

5. Melyik állítás igaz?

- [ ] Az első műveletvégző egységek egyike sem tudott közvetlenül valós számokkal műveletet végezni.
- [ ] Neumann számítógéptervében nem szerepelt logikai műveletek elvégzésére képes műveletvégző egység.
- [ ] Az ALU csak aritmetikai műveletek tud végezni.

6. Melyik állítás igaz?

- [ ] A Neumann-architektúra abban különbözik a Harvard-architektúrától, hogy az előbbinek egy az utóbbinak kettő memóriája van.
- [ ] Neumann nem tervezett órajelet előállító áramkört a számítógépéhez.
- [ ] Neumann János Alan Turinggal közösen tervezte az első elektronikus számítógépet.
- [ ] Neumann számítógépének memóriája kikapcsolás után is megőrizte tartalmát.

5. LECKE

Számítógép-generációk és a PC felépítése

Ebben a leckében megismerkedhetünk azzal a folyamattal, ami napjaink számítógépeinek megjelenéséig vezetett. A fejlődés folyamatát a számítógép-generációk reprezentálják. Az első három generáció bemutatása vázlatos, csak arra koncentrál, hogy a főbb jellegzetességek felsorolásszerűen megjelenjenek.

A negyedik generáció bemutatása részletesebb, mivel a mai számítógépek zömmel ehhez tartoznak.

Az ötödik generációval – mivel még ennek jellegzetességei nem alakultak ki – nem foglalkozunk.

A lecke utolsó témája a PC-bemutatása, mindenki számára fokozottan ajánlott rész, de felhívjuk a figyelmet arra, hogy önmagában ez a fejezet nem feldolgozható, mivel sok olyan fogalommal találkozhatunk itt, amit az előző fejezetek tárgyalnak.

6. Számítógép-generációk

Az 1950-es évektől kezdődően a számítógép-tervezés és -készítés ipari méreteket öltött. Újabb és újabb eszközök és ötletek alapján egyre gyorsabb és megbízhatóbb gépeket építettek. A fejlődés fázisait az úgynevezett számítógép-generációkhoz tartozó számítógépek fontosabb technikai és logikai tulajdonságaival érzékeltethetjük.

6.1. Az első generáció

Az **első generációs gép**eket az elektroncsöves technika jellemezte. Aritmetikai egységeik – néhány kivételtől eltekintve – csak az egész számok kódjaival tudtak számolni. A lebegőpontos kódokkal végzendő műveletekre programot kellett készíteni. Ezek az egyéb, úgynevezett felhasználói programokkal együtt gépi nyelven készültek, azzal a könnyítéssel, hogy nem kettes számrendszerben kellett az utasításkódokat leírni, hanem vagy nyolcas, vagy tizenhatos számrendszerben. Ezt kellett az input eszköz segítségével konvertálni kettes rendszerre és bejuttatni az operatív memóriába. A tartalmat megőrző memória mérete pár bájtnyi volt, a gépet működtető programot jellemzően háttértárról kellett „betölteni” a memóriába. Gyakran előfordult olyan megoldás is, hogy a gép az első végrehajtandó utasításokat is a háttértárról kapta.

E korszak gépeit a processzorcentrikus felépítés jellemezte, a külső berendezéseket (input, output stb.) a CPU közvetlenül vezérelte. Emiatt a számítógép sebessége jóval alatta maradt az elektronikus alkatrészek (a CPU, a memória) sebességének, hiszen a perifériák sok mechanikus alkatrésszel rendelkeztek, és ezek működési sebessége nem érhette el az elektromos egységek működési sebességének nagyságrendjét sem.

Az első generációhoz tartozó gépek logikai felépítése a neumanni architektúrát követte, bár már ekkor megjelentek a Harvard-architektúrás gépek is, és a neumanni struktúrán is voltak minimális változtatások. Ez a korszak az 1950-es évek közepéig tartott.

6.2. A második generáció

A **második generációs gép**ekben (1955–65) a kisméretű és sokkal gyorsabb félvezető diódák, tranzisztorok kiszorították az elektroncsöveket. Ezzel gyorsítani lehetett a CPU működését. Sőt, bonyolultabb áramkör létrehozása is lehetségessé vált, mivel a félvezetőkből készült alkatrészek sokkal kisebb helyet foglaltak el és sokkal kisebb mértékben melegedtek, mint az elektroncsövek. Ezek a memóriaépítésből is kimaradtak, helyettük megjelent a ferritgyűrű mint olyan eszköz, amely egy bit tárolására kiválóan alkalmas. Minden bináris számértéket egy-egy mágnesezhető pici ferritgyűrű tárolt. Az egyik számjegyet az egyik, a másik számjegyet a másik mágneses irány jelentette. A ferritgyűrűből felépített memória nagyságrendben kisebb helyet foglalt el az elektroncsöveshez képest. Ezért sokkal nagyobb kapacitású memóriát lehetett vele építeni. A bináris információk elérését, módosítását a gyűrűkön átfűzött vékony huzalok elektromos jelei biztosították.

Ha a számításokhoz kicsi volt a memória (ekkor ennek még sokkal nagyobb volt az esélye, mint manapság), akkor az éppen nem használt kódokat háttértárak kódtárolóira írták, ahonnan szükség esetén beolvashatók. Mivel ezek a kódtárolók, más szóval adathordozók az egységről levehetők voltak, ezért a nagytömegű ritkán használt adatokat is tárolni lehetett rajtuk. A tárolás elve a mágnesezhetőségen alapult, az adathordozók mágnesezhető anyagból készültek, kivitelük szalag- vagy lemezforma volt.

A második generációs gépek mérete jelentős mértékben lecsökkent, az üzembiztonságuk megnőtt. A teljesítményük, sebességük az 1 MIPS értéket is elérte. A gyorsabb központi egységet a lassú külső berendezések (perifériák) visszafogták, éppen ezért a beviteli-kiviteli műveletekre a CPU helyett önállóan működő perifériaprocesszorokat építettek a gépbe. Ezek vezérelték az input és output tevékenységet, a CPU csak elindította a folyamatot. A művelet elvégzése után a perifériaprocesszor üzenetet küldött a CPU-nak, és várta a következő művelet elvégzésére a felszólítást. Ezzel a megoldással az input vagy output műveletek elvégzése alatt a CPU további utasításokat tudott végrehajtani.

Mivel az önálló vezérléssel rendelkező perifériák, háttértárak megjelenésével a külvilággal történő output kommunikációt a CPU csak elindítja, az átvitelt már a perifériavezérlő felügyeli, ezért az output folyamat befejezését csak akkor érzékelheti, ha erről jelzést kap. Ezt a jelzést, úgynevezett megszakításkérést a processzornak fogadnia kell, amit csak akkor tud megtenni. ha az éppen feldolgozás alatt lévő programot

megszakítja, és csak azt követően folytatja, ha elvégzi azokat az utasításokat, amelyek a kivitel befejezéséhez szükségesek.

Bizonyos esetekben az input perifériák is kezdeményezhetnek kommunikációt. Ilyenkor a megfelelő perifériavezérlő ugyancsak megszakításkérést küld a CPU-hoz, aminek következményeképpen az éppen futó programot megszakítva a CPU elvégzi a szükséges utasításokat az inputperifériáról érkező kódok feldolgozására, és csak ezután folytatódik a megszakított program.

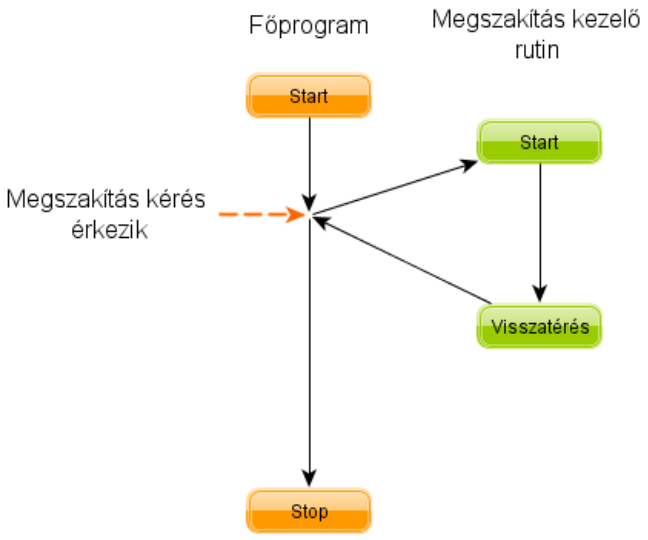
Ez a folyamat részletesebben (a 6.1. ábrán látható):

1. Az I/O eszköz jelzi a tevékenységének kezdetét vagy végét, azaz úgynevezett megszakításkérést küld a CPU-nak.
2. A CPU észleli a megszakításkérést, az éppen működő programot a feldolgozás alatt lévő utasítás végrehajtását befejezve megszakítja (nem a program következő utasítását hajtja majd végre).
3. Elindítja a megszakításkérésnek megfelelő feladat elvégzésére szolgáló eljárást (ez egy olyan program, amelynek befejezése után a megszakított program soron következő utasításával folytatható a programfutás).
4. A megszakításkérést kiváltó problémához tartozó eljárást befejezve folytatja a megszakított programot.

A megszakításkérések kezelését a CPU végezte, de némelyik számítógépbe már megszakításvezérlő áramkört is építettek, ami gyorsabbá és biztonságosabbá tette a megszakításkezelést.

A megszakításkérések hierarchikusan is feldolgozhatók, ami azt jelenti, hogy a számítógépekben ezek feldolgozása nem sorban, egymás után történik, hanem egy magasabb prioritású (elsőbbséget élvező, valamilyen szempontból fontosabb) megszakításkérés megszakíthatott egy épp folyamatban lévő alacsonyabb prioritásút (kevésbé fontosat).

Néhány gyártó megjelent a piacon olyan ALU-val is, amelyek már közvetlenül voltak képesek a lebegőpontos kódokkal való számolásra, így ezeken a gépeken nem kellett program a lebegőpontos kódolású számokkal való műveletvégzéshez.



6.1. ábra. *Programvégrehajtás megszakítás esetén*

A számítógép működéséhez, működtetéséhez segédprogramokat készítettek. Ezek a programok voltak a későbbi operációs rendszerek előfutárai. Operációs rendszernek egy olyan programrendszert nevezünk, amely képes arra, hogy a felhasználó minimális segítségével teljesen automatikusan működtesse a számítógépet. Olyan szolgáltatásokat adjon, amelyek kényelmessé teszik a felhasználók feladatait megoldó programok használatát is.

Megjelentek azok a programrendszerek, amelyek bizonyos feladatkör megoldását, kezelését tették lehetővé, azaz kialakult az ipari szoftvergyártás. Szoftver a különböző programok, programrendszerek gyűjtőneve. A megvásárolható programrendszerek mind-mind szoftvertermékek.

A termékek előállítására hatékony eszközökre volt szükség, ezért ebben korszakban dolgozták ki az első magas szintű programozási nyelveket is.

6.3. A harmadik generáció

A **harmadik generációs gép**ek korszaka a 60-as évek közepétől a 70-es évek végéig tartott. Az 1960-as évek első harmadában megjelentek az első szilícium alapú integrált áramkörök, amelyekbe az évtized végén már néhány száz áramköri elem (dióda, tranzisztor) volt beépítve. A miniatürizálás következménye a számítógépekben: csökkent a méret, arányosan nőtt a működési sebességük és négyzetes arányban csökkent az energiafogyasztás. A gépek tárolási kapacitása megsokszorozódott.

Megjelentek olyan operációs rendszerek, amelyek képesek voltak egyszerre több program futtatására is. Kétféle technológiát alkalmaztak arra, hogy több program egyidőben futtatható legyen.

Az egyik az úgynevezett multiprogramozás, amelynél az operatív memóriában több felhasználói program is lehetett, és ezeket az operációs rendszer felügyeletével a számítógép a következőképpen futtatta: Az első program futtatásával kezdte a munkát. Ha ez a program I/O tevékenységet kezdett, akkor az I/O befejeztéig a második program utasításait hajtotta végre egészen addig, amíg az első program I/O tevékenysége be nem fejeződött, vagy a második el nem indított egy I/O tevékenységet. Az első esetben az első programot folytatta a gép, a második esetben a harmadik program végrehajtására kapcsolt és így tovább.

A másik a multitaszking, a többfeladatos rendszer. Ez szintén több program futását tette lehetővé úgy, hogy minden programhoz hozzárendelt a rendszer egy időt – többnyire ugyanannyit –, és amikor az egyik programra szánt idő lejárt, a következő program utasításai kerültek sorra egészen addig, amíg az utolsó programra szánt idő is letelt. Ekkor újra az első program utasításai kerültek sorra. Természetesen, ha valamelyik program input vagy output végrehajtását indította el, akkor rögtön a következő program utasításaira került sor, azaz a multiprogramozási technika itt is érvényesült. Ha az egyes programokra szánt idő elegendően rövid volt,

akkor a felhasználók úgy érezték, mintha a programjaik párhuzamosan működnének. Ezt a működési módot időosztásos, idegen szóval time sharing technikának nevezzük.

A gépek kihasználtság azzal is fokozták, hogy a gépen egyidőben több felhasználó dolgozhatott. Ezek a felhasználók akár több programot is elindíthattak. Ezek futását multiprogramozott vagy multitaszking módon az operációs rendszer felügyelte.

Amiatt, hogy a számítógépen egyszerre több program is futhatott, gondoskodni kellett a memória megfelelő kezeléséről. Nem szabadott ugyanis megengedni, hogy az együtt futó programok egymás memóriaterületére írjanak, esetleg olvassanak.

Az is előfordulhatott, hogy a memória nem lett volna elég az elindított programok futtatásához. Ennek a problémának a megoldására kialakult az úgynevezett virtuális memóriakezelés. Ezt úgy valósították meg, hogy a memóriát logikailag egyenlő részekre, úgynevezett lapokra osztották, és ezeket a lapokat rendelték a futtatandó programokhoz. Ha a program nagyobb területet igényelt volna, mint amit a kiosztott lapok meghatároztak, akkor a programnak és a futáskor szükséges munkaterületnek csak egy része került az operatív memóriába, a többi háttértáron maradt. Ha szükség volt a háttértáron lévő program- vagy adatkódra, az operációs rendszer kicserélte az éppen nem használt lapot a memóriában arra a részre, amire a program folytatásához szükség volt. Ezt a működési módot memórialapozásnak hívjuk. Ezzel a technikával látszólag az operatív tár méreténél nagyobb (virtuális vagy látszólagos) memória volt használható (6.2. ábra).

Mind a multiprogramozott, mind a többfeladatos, mind a többfelhasználós használatot az operációs rendszerek biztosították. Ez mindhárom esetben olyan memóriamenedzselést jelentett, amely nélkül a biztonságos működés lehetetlen lett volna.

Az operációs rendszerek új szolgáltatásai mellett szerkezetük is jelentősen fejlődött. Megjelent a modularitás, és ezek a modulok piramisszerűen egymásra épülve a hétköznapi felhasználó elől elrejtették a gép valódi működését, olyannyira, hogy ő csak a piramis tetején lévő modulhoz férhetett hozzá. Ezzel megnövekedett a működés biztonsága is.

Megjelentek a modulok létrehozását támogató, sőt speciális feladatok programozására alkalmas magas szintű programozási nyelvek, ezáltal tovább növekedett a programozói munka hatékonysága.

6.4. A negyedik generáció – napjaink számítógépe

A **negyedik generációs gép**ek megjelenését a 70-es évek közepétől számítjuk. A technikai fejlődés következményeképpen lecserélhetők lettek a még mindig elég nagyméretűnek számító elektronikus és mágneses elven működő alkatrészekből álló egységek az úgynevezett integrált áramkörökkel megvalósított egységekre. Ez a számítógépek méretének jelentős csökkenésével járt, és lehetőséget adott arra is, hogy olyan egységek is megjelenjenek a számítógépeken, amelyeknek megépítésére vagy nem volt eddig szükség, vagy nem volt lehetőség. Ezek az új részek egyrészt az operációs rendszerben szoftveres úton megoldott funkciókat váltottak le hardver megvalósításra, másrészt új szolgáltatásokat nyújtottak a számítógép biztonságos működéséhez.

Ilyen új egységek:

- A nagy kapacitású megmaradó tartalmú memória.
- Kisméretű, nagykapacitású közvetlen hozzáférésű írható, olvasható memória-áramkörök.
- Direkt memória-hozzáférést biztosító áramkörök.
- Memória-hozzáférést szabályozó áramkörök.
- Programozható perifériavezérlők.
- Új periféria- és háttértár-csatlakozók
- A processzorok megjelenése a működési sebesség növelésének különböző módszerei.

Ezek az egységek az integrált áramkörök gyártástechnológiájának továbbfejlődésének (az LSI – Large Scale Integration: magas integráltsági fokú áramkör megjelenése) következményeképpen egyre kisebb és kisebb méretűek lettek, és manapság már a zsebben is elfér egy olyan teljesítményű számítógép, amely a generáció kezdetén még szobaméretű volt.

- A tartalmukat megőrző (csak olvasható) memóriák
  Az évek során többféle változatuk alakult ki. A klasszikus változata a ROM memória. Az angol Read Only Memory rövidítése, jelentése csak olvasható memória. Tartalmát a gyártáskor rögzítik. Ez a tartalom nem változtatható, megmarad akkor is, ha nem kap áramellátást, tápfeszültséget. Felhasználása elvileg korlátlan idejű. A memóriacellái általában direkt elérésűek, ami azt jelenti, hogy bármelyik memóriaegysége közvetlenül címezhető, tartalma kiolvasható függetlenül attól, hogy az előtte vagy utána lévő memóriaegységek tartalmát elolvasta-e a számítógép vagy sem.
  A PROM (Programable Read Only Memory) típusba nem gyárilag égetik be a tartalmat, hanem speciális programíró berendezéssel a felhasználója egyszer beégeti, és ezután csak olvasható módon érhető el a tartalom, amit a tápfeszültség megszűnése után is megőriz.
  A memóriatípus továbbfejlesztett változata az EPROM, az angol Erasable Programable Read Only Memory rövidítése. Ez magyarul törölhető programozható csak olvasható memóriát jelent. Ez látszólag ellentmondás, de valójában nem, hiszen ez a memóriafajta csak speciális eszközzel törölhető és programozható újra, az a számítógép, amiben a memória van, ezt a törlést és újraprogramozást nem tudja elvégezni. A gyártáskor általában ebbe a memóriába is beleégetik a kezdő tartalmat. Törlése UV-fénnyel történhet, ezután új tartalom írható rá.
  A legújabb változatai az EEPROM memóriák. A mozaikszó az angol Electrically Erasable Programable Read Only Memory kifejezésből keletkezett. Ennek magyar jelentése: elektromosan törölhető programozható csak olvasható memória. A csak olvasható kitétel valójában itt arra utal, hogy a memória tápfeszültség nélkül is megőrzi tartalmát, nem szükséges újraírással frissíteni, a használat során csak olvasni kell a tartalmát. Ha kell, a tartalom kicserélhető, átírható. Ezt az átírást akár a számítógéppel is elvégezhetjük. Az EEPROM memóriák egy speciális változatát, a flashmemóriát háttértárként is lehet használni. Ilyen memória található a mai pen drive-okban is.

- A tartalmukat elvesztő (írható olvasható) memóriák
  Ezeket a memóriákat RAM-nak (Random Access Memory rövidítése; ennek szó szerinti fordítása: véletlen elérésű memória) nevezzük. Az elnevezés nem igazán szerencsés, mivel a véletlen elérés valójában azt jelenti, hogy a memóriaegységeket tetszőleges sorrendben, a címük alapján lehet kiválasztani és a tartalmat kiolvasni vagy megváltoztatni. Ezeknek a feladatoknak az elvégzésére fordított idő – más szóval az elérési

idő – nem függ a memóriaegység helyétől. Nem szerencsés az elnevezés azért sem, mert a ROM-típusú memóriák tárolóegységeit is így érhetjük el. Ennek ellenére a RAM megnevezés csak az írható-olvasható memóriákat jelenti. Három típusáról teszünk említést.

A dinamikus RAM vagy DRAM. Bitjeit olyan elektronikus elemek (egy kondenzátor és egy szigetelt elektródájú térvezérelt tranzisztor, MOSFET) alkotják, amelyek tartalma kiolvasáskor, illetve az idő előrehaladtával akkor is elveszik, ha feszültség alatt van a memória. Éppen ezért kiolvasás után és bizonyos idő elteltével a tartalmat frissíteni kell. Ennek a tulajdonságnak köszönheti nevében a dinamikus jelzőt. Az állandó frissítések miatt működése lassú, de viszonylag olcsó előállítási költsége miatt általánosan elterjedt operatívmemória-fajta.

A szinkron DRAM, röviden SDRAM (Synchronous dynamic random access memory) olyan memória, amit az órajel segítségével a többi egységgel szinkronban lehet elérni. A szinkronizálatlan memória küldte vagy fogadta a kódokat amint csak tudta, a szinkron működésű memória csak akkor fogad, küld, ha az órajel ezt lehetővé teszi (például egy felszálló ág megjelenésekor).

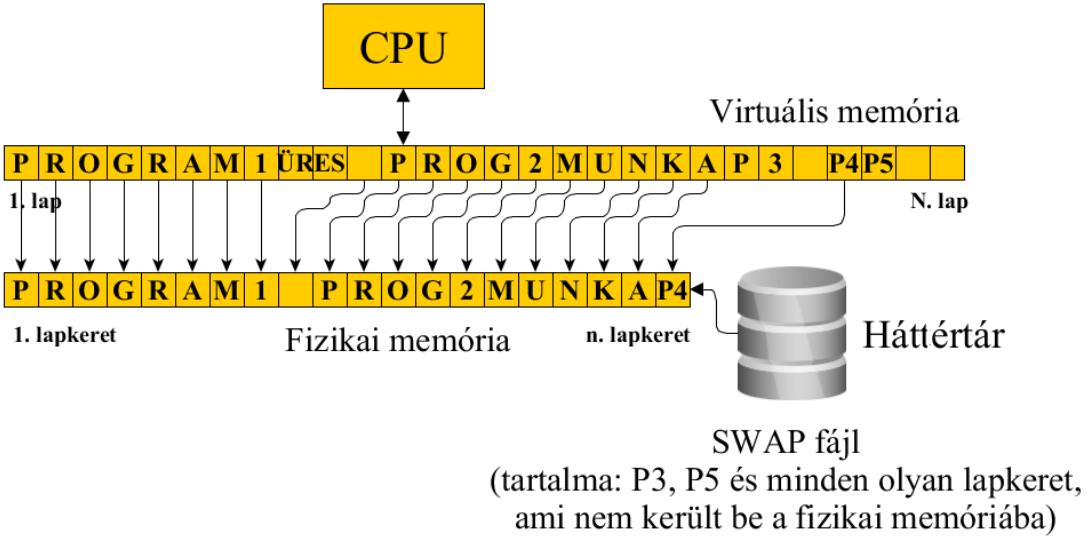
A statikus RAM vagy SRAM bitjei más alkotóelemekből (6 szigetelt elektródájú térvezérelt tranzisztor, MOSFET) épülnek fel. Ezek csak akkor vesztik el a tartalmukat, ha a tápfeszültség megszűnik, így normál működés közben nincs szükség állandó frissítésre. Ezért az elérési idejük kisebb, mint a DRAM-oké. Előállítási költségeik viszont nagyságrenddel nagyobbak, ezért elsősorban speciális memóriaként és nem operatív memóriának használják őket. A CPU regiszterei például SRAM memória lehet. Ugyancsak SRAM-elemekből épülnek meg az úgynevezett gyorsítótárak (cache-memóriák). Ezek viszonylak kiskapacitású (kezdetekkor pár száz B, manapság akár pár KiB is lehet) memóriák, amelyek arra szolgálnak, hogy a lassúbb operatív memóriából a CPU működésétől függetlenül, annak működésével párhuzamosan kódokat másoljunk, hogy a CPU innen olvassa ki, ha szüksége lesz rá. Használhatnak gyorsítótárat a perifériák, a háttértárak és az operatív memória közti kódforgalom lebonyolítására is.

- Közvetlen memória-hozzáférés – DMA

  A mozaikszó a Direct Memory Access kezdőbetűiből keletkezett. A CPU tehermentesítésére alkalmazzák azt az áramkört, amely a közvetlen memória-hozzáférést és ennek vezérlését megvalósítja. Gyakran előfordul ugyanis, hogy nagymennyiségű kódot kell másolni I/O eszközök és a memóriatartomány, valamint memóriatartomány és memóriatartomány között. Gazdaságtalan lenne, ha ezt teljes egészében a CPU

végezné. Elegendő csak elindítania a feladatot, és érzékelni a befejezését. Az elindításhoz a CPU a DMA vezérlőjében beállítja a másolás forrását, a célt és a másolandó bájtok számát, majd átadva a munka folytatását a vezérlőnek olyan utasítások végrehajtásával folytathatja működését, amelyek végrehajtásához a másolás eredménye nem szükséges. Ha a DMA-vezérlő befejezte a másolást megszakításkéréssel jelzi ezt a CPU-nak, ami megszakítja az éppen végrehajtás alatt lévő programot, a másolás befejezésének adminisztrációját elvégzi a megszakításkérést feldolgozó programrésszel, majd folytatja a megszakított program következő utasításának végrehajtásával.
Ezzel a memóriaelérési technológiával természetesen a DMA átveheti az I/O eszközök elindításának és az I/O eszközök megszakításkérésének feldolgozását is. Természetesen nem minden I/O-tevékenység jelent nagy kódmozgást. Pár bájtnyi input kód, aminek ráadásul a számítógép működésére azonnali hatással kell lenni, a DMA-vezérlő nélkül is a CPU-hoz juthat. Ilyen input kód lehet például a modern számítógépek egér-perifériájának elmozdulásából keletkező kódok, a billentyűk leütéséből keletkező kódok stb.

- A memória fizikai menedzselése, az MMU
  A memória menedzselése az operációs rendszerek feladata volt. Szolgáltatásaiknak növekedése egyre bonyolultabbá tette a munkájukat. Ezt a munkát egyszerűsítendő – mivel a technikai fejlődés számítógépekbe újabb és újabb áramkörök beépítését tette lehetővé anélkül, hogy a méret növekedett volna –, a memóriamenedzselés egy részét is ilyen új áramkörök segítségével valósították meg. Ezeknek az áramköröknek a rendszere az MMU (Memory Management Unit), aminek segítségével a virtuális memória kezelése gyorsabbá és biztonságosabbá vált.
  Az MMU a virtuális memóriát kisméretű – általában 2, 4, 8 KiB nagyságú – blokkok, más szóval lapokból álló rendszerként kezeli (6.2. ábra). Ezekhez a lapok sorszámozva vannak, és ezzel a sorszámmal azonosíthatók. Amikor egy programot a memóriába töltünk, az ilyen lapokra kerül. Ezeknek a lapoknak megfeleltetve a valós memória szabad vagy szabaddá tett lapkereteit töltődnek a szükséges kódok a valós memóriába. Nem minden lapnak fog megfelelni lapkeret. Amelyikhez nincs, annak tartalma az úgynevezett swap-fájlban, valamelyen háttértáron érhető el. A gyors működéshez a mai háttértár-kapacitások mellett akár a teljes virtuális memória laprendszere is a swapban lehet.
  A programban lévő címek, valamit a felhasznált lapok adatiból felépít egy táblázatot, ami a lapokhoz való hozzáférést és a program címeinek a fizikai elhelyezés szerinti címekre való átkonvertálását lehetővé



6.2. ábra. *A virtuális memória*

teszi. Ezt a táblázatot saját gyorsítótárjában, a TLB-ben (translation lookaside buffer) helyezi el. Ezáltal a programban előforduló címek nem az operatív memória, hanem egy virtuális memória címei lesznek. Ezeket használhatja a CPU, és ha szükséges az MMU ezeket konvertálja valós címekké (6.2. ábra). A virtuális memória mérete nem szükségképpen egyezik meg a fizikai memóriamérettel. Általában nagyobb, így a lapozási technika segítségével az operatív tárban mindig az a kódrész van, ami éppen szükséges.
A CPU ezen a virtuális memórián keresztül fér a valós memóriához. Ha olyan cím érkezik, amely a hozzáférési információknak ellentmond vagy a fizikai cím kialakításakor nem a programhoz tartozó lapok bájtjainak címét kaptuk, akkor szükséges konverziók elvégzését az MMU áramkörei elvégzik. (Például, ha a CPU-tól kapott cím olyan kódra mutat, ami nincs a memóriában, valamelyik éppen nem használt lapkeretbe

bekerül a szükséges kódegyüttes, és ezután a fizikai címet kiszámolva a szükséges kód elérhető lesz.)

- Perifériavezérlők
  Mivel a perifériák, de még a háttértárak működési sebessége is nagyságrendekkel elmaradt a CPU működésétől, ezért viszonylag hamar, már az 1960-as években is megpróbálták a vezérlésüket függetleníteni a központi vezérlőtől. Ezt a törekvést technikai problémák akadályozták. A miniatürizálás, a gyors memóriatípusok megjelenésével az 1970-es, 80-as években ezek a problémák részben megszűntek, így lehetővé vált, hogy a perifériavezérlést típusonként önálló egységgel oldják meg. Ezeket kezdetben egyszerű vezérlők voltak, amelyek az adott periféria és a memória közti kódforgalmat lebonyolító utasítások végrehajtására voltak képesek. A CPU-tól megkapták a forgalom lebonyolításához szükséges paramétereket, ezekkel önállóan elvégezték a munkát, majd megszakításkéréssel jelezték, hogy készen vannak.
  Később a vezérlőkhöz memóriát, sőt műveletvégző egységet is építettek. Ezek az eszközök a számítógép részei voltak. Némelyiket fixen a gépbe építették, másokat bővítményként, a gép felhasználójának kívánságára – akár utólag – lehetett a számítógépbe építeni.
  Megjelentek az intelligens perifériák is, amik a számítógéptől független memóriával, vezérlővel és műveletvégző egységgel rendelkeztek. Mondhatnánk, speciális célra készült számítógépek voltak. Manapság szinte mindegyik periféria (háttértár) ilyen.

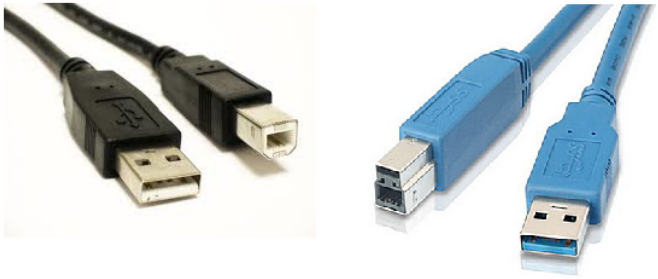
- Periféria-portok
  A számítógépbe épített perifériavezérlők csatlakozóin – más szóval portjain – lehet a perifériákat a számítógéphez csatlakoztatni. Kezdetben minden perifériának külön vezérlője volt, később a perifériacsoportok kaptak vezérlőt, és szoftveres úton, úgynevezett meghajtóprogrammal volt képes a vezérlő a perifériát kezelni.
  Manapság szinte általánosan elterjedt az USB (universal serial bus azaz az általános soros busz), amelynek portjára (magyarul kapujára, csatlakozó pontjára) megfelelő szoftvertámogatással szinte mindenféle periféria csatlakoztatható.
  Ennek megfelelően az USB-t használó eszközök köre igen széles: billentyűzet, egér, játékvezérlő, lapolvasó, nyomtató, digitális fényképezőgép, webkamera, külső merevlemez stb. Az USB soros, pont-pont közötti

kapcsolatot biztosít a számítógép (host) és a külső eszköz (device) között. Az USB egy számítógéphez legfeljebb 127 eszköz csatlakozását teszi lehetővé. Természetesen ennyi csatlakozási pont egyetlen számítógéphez sem építenek, de speciális elosztóval, az úgynevezett USB-hubbal a csatlakozási pontok sokszorozhatók.
A manapság használatban lévő gépeken és eszközökön három verziójának valamelyike található meg. Működésüknek lehetséges sebességfokozatait és értékeit az alábbi táblázat tartalmazza. A sebességet megabit/másodperccel (egy másodperc alatt átvitt bitek száma Mb-ben) jellemezzük.

6.16. táblázat. *USB szabványok maximális átviteli sebességei*

Sebességfokozat	USB 1.1	USB 2.0	USB 3.0
Alacsony sebesség (low speed)	1,5 Mb/s	1,5 Mb/s	1,5 Mb/s
Teljes sebesség (full speed)	12 Mb/s	12 Mb/s	12 Mb/s
Magas sebességfokozat (hi-speed)	x	480 Mb/s	480 Mb/s
Szupersebesség (superspeed)	x	x	5 Gb/s

Az USB szabvány 1.1-es és 2.0-ás változatában azonos csatlakozókat és kábeleket használnak, a 3.0-ás változatban azonban változtatni kellett a megnövekedett sebesség miatt. Az ilyen 3.0-ás csatlakozókat kék színnel különböztetik meg a hagyományos 1.1-es és 2.0-ás csatlakozóktól, melyekkel felülről kompatibilis, ami azt jelenti, hogy 3.0-ás csatlakozó ponthoz használhatunk 1.1 vagy 2.0 szabványkábeleket, de a 3.0-ás kábel sem az 1.1, sem a 2.0 csatlakozásra nem megfelelő.



6.3. ábra. *USB 1.1, 2.0 és 3.0-ás csatlakozók*

- Processzorok
  Azzal, hogy az I/O vezérlése önállóvá vált, a CU-ra csak a memória (esetenként ennek is csak a virtuális változata) kezelése és az ALU irányítása maradt. Az időszak elején, amikor a programok futási idejét a memória gyorsasága határozta meg elsősorban, olyan CU-kat alkalmaztak, amiknek utasításkészlete bonyolult és sok utasításból állt. A lassú memória miatt a bonyolult utasításokhoz szükséges viszonylag hosszú végrehajtási idő sem növelte a programok futási idejét. Az ilyen számítógépeket összetett utasításkészletű számítógépnek hívták (röviden CISC az angol Complex Instruction Set Computer rövidítése).
  Mivel a memória gyorsasága növekedett, ez lehetővé tette az utasítások egyszerűsítését, és a számuk is csökkent. Igaz, hogy ugyanannak az eredménynek eléréséhez négy-öt egyszerűbb utasítás kellett, de ezeket az utasításokat akár tízszer gyorsabban lehetett végrehajtani, ami CU működésében jelentős gyorsulást eredményezett. Az ilyen számítógépet csökkentett utasításkészletűnek nevezték (röviden RISC, az angol Reduced Instruction Set Computer rövidítése).
  A miniatürizálás növekedésével lehetőség nyílott arra is, hogy a különálló regisztertömböt, CU-t és ALU-t egybeépítsék. Ezt az egybeépített egységet hívják processzornak. Az egyetlen chipben megvalósított processzor a mikroprocesszor.

A memóriák további gyorsulása, illetve a különböző memóriakezelési technikák alkalmazása és egyéb más okok következményeképpen a processzorok sebességének növelése fontos kutatási feladattá vált. Az egyik lehetőségnek a Neumann által javasolt soros utasítás-végrehajtás helyett valamiféle párhuzamos végrehajtás látszott.

Neumann – amit tudjuk –, úgy képzelte el számítógépének utasítás-végrehajtását, hogy a CU csak akkor fog neki a következő utasításnak, ha az előzőt már befejezte. Ez a technológia, amely a korai számítógépek CPU-egységét jellemezte, nem volt hatékony módszer. Érezte ezt Neumann János is, hiszen alig írta meg nevezetes jelentését, más számítástechnikával foglalkozó kutatókkal egyidőben máris foglalkozni kezdett a párhozamos végrehajtás lehetőségével.

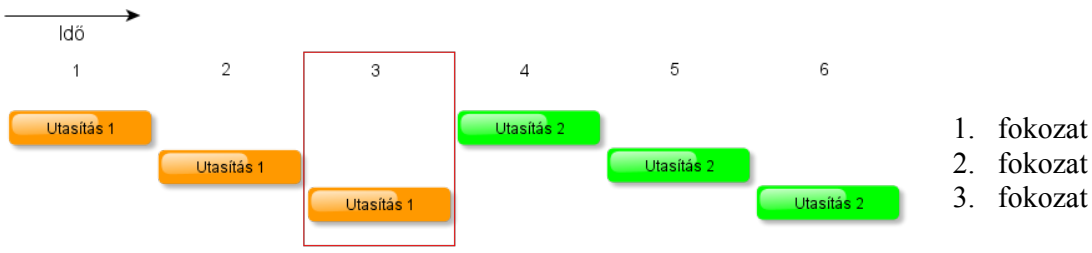
- A párhuzamos végrehajtás

  Az alapprobléma az volt, hogyan is lehetne megoldani, hogy az utasítás-végrehajtáskor a CPU éppen nem működő részeit hogyan lássuk el munkával, illetve milyen módon lehetne egyszerre több utasítást is végrehajtani. Többféle javaslat is született, megvalósításukhoz viszont nem volt megfelelő technikai háttér. Az integrált áramköri technológia kialakulása lehetővé tette ezeknek a javaslatoknak megvalósítását, sőt újabb megoldásokat is eredményezett.

  A modern számítógépek szinte mindegyike alkalmaz valamilyen párhuzamos technológiát, annak ellenére, hogy tudjuk: bizonyos utasítások nem végezhetők el párhuzamosan. Gondoljunk csak arra, hogy mi történne, ha az 5.3.4. fejezetben megismert ADD utasítás következő két változatát hajtanánk végre egyszerre: ADD cím,'2 és ADD cím,'1. Mi lenne a cím sorszámú memóriaegység értéke? Eggyel, kettővel vagy hárommal nagyobb szám kódja, mint amilyen szám kódját a két utasítás végrehajtása előtt tartalmazta? (Ezt a jelenséget adatfüggőségnek nevezzük.)

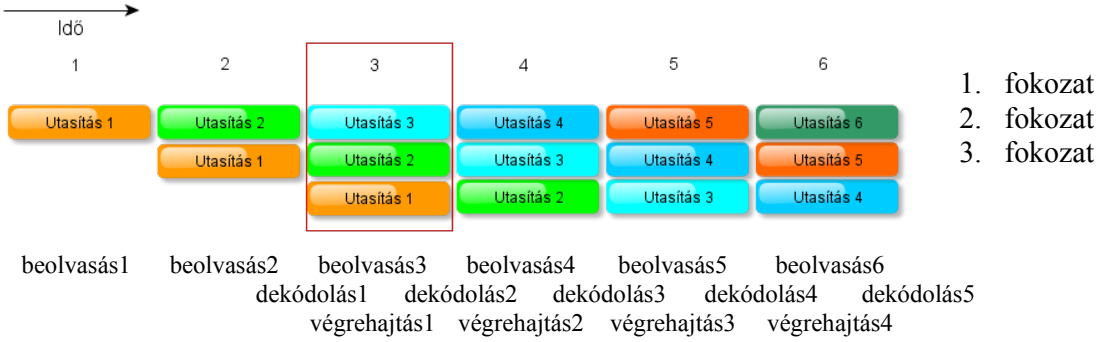
  Mivel mindkét utasításnak a cím által meghatározott memóriaegységben lévő kóddal kell dolgoznia, és az eredmény is ebbe a memóriaegységbe kerül, ezért a párhuzamos végrehajtáskor az eredmény függne attól is, hogy a két utasítás végrehajtása mennyire párhozamos. Pontosan együtt végezzük őket, vagy az egyik végrehajtása előbb kezdődik, mint a másiké. A párhuzamos végrehajtás körülményeitől függően más és más lenne az eredmény. Ez nyilván nem engedhető meg, tehát nem elég csak a párhuzamos végrehajtásra technikákat kidolgozni és megvalósítani, ügyelni kell arra is, hogy a fenti és a hozzá hasonló problémák se okozzanak hibát a programok futásában.

- Pipeline technológia
  A legegyszerűbb, leggyakrabban használt lehetőség az utasítások párhuzamos végrehajtására a pipeline vagy csővezeték-technika. Pipeline nélkül az utasítás végrehajtását a 6.4. ábrán láthatjuk.



6.4. ábra. *Végrehajtás pipeline nélkül*

Vagyis az utasítás végighalad a beolvasás, dekódolás és végrehajtás fázisain, majd ezután következhet egy újabb utasítás végrehajtása. Bármely időpillanatban vizsgáljuk a processzort, annak csak egyetlen egysége dolgozik, a többi kihasználatlanul áll.
Ezzel szemben a pipeline úgy működik, mint egy szerelőszalag a gyárban. Az első utasítás dekódolása majd végrehajtása közben a felszabadult beolvasó egység már olvassa a memóriában az éppen feldolgozás alatt lévő utasítást követőt, majd ha lehetséges az ezt követőt stb. A bekapcsolás utáni pár ciklustól eltekintve bármely időpillanatot vizsgálva azt tapasztaljuk, hogy a processzor összes egysége dolgozik.
Az előző két ábráról látható, hogy a soros végrehajtás hat időegység alatt két utasítást, a pipeline ugyanennyi időegység alatt négy utasítást fejez be, azaz sokkal nagyobb teljesítményre képes, így rövidebb idő alatt futtatja le ugyanazt a programot.
A pipeline annál hatékonyabb, minél több részre, fokozatra (angolul stage) sikerül szétosztani egy utasítás végrehajtását. A több pipeline fokozat ugyanakkor nagyobb, bonyolultabb processzort jelent, és az utasítások végrehajtására is több órajel-periódusidő kell.



6.5. ábra. *Pipeline-végrehajtás*

További problémát okoznak az elágazó utasítások. Elágazáskor csak a végrehajtás egy későbbi fázisban derül ki, hogy a programot melyik programág utasításainak végrehajtásával kell folytatni, tehát elképzelhető, hogy az addig betöltött, részben végrehajtott utasításokat ki kell dobni, és más utasítások a végrehajtást kell elkezdeni. Az ilyen újrakezdések természetesen jelentős teljesítménycsökkenést okozhatnak. Elkerülésükre további ötletek megvalósítására van szükség. Ilyen ötlet lehet az elágazások előrebecslése.

Ugyancsak meg kell oldani azt is, hogy mi történjen akkor, ha a párhuzamosan végrehajtott utasításokban az operandusok egymástól nem függetlenek. Ebben az esetben a később indított utasítás-végrehajtással meg kell várni annak az utasításnak a befejezését, amelynek eredményére a folytatásban szükség lesz. Ez a késleltetés szintén ronthatja a processzor teljesítményét. A problémákat és a megoldásukat is figyelembe véve a pipeline-technológia hatékonyabb programfuttatást tesz lehetővé, mint az egyszerű soros utasítás-végrehajtás.

- Szuperskalár-processzorok
  Mivel az utasítás-végrehajtáskor a legtöbb időt a művelet-végrehajtás, azaz az ALU működési ideje teszi ki, ezért kézenfekvő megoldás a teljesítménynövekedésre az, hogy a processzorba több ALU-t is beépítenek. Az ilyen több ALU-val rendelkező processzorok a szuperskalár-processzorok. Ezek jelentik párhuzamos végrehajtás következő szintjét. Az ilyen CPU-k az órajel egy periódusidején akár több utasítás végrehajtását is befejezhetik. Az adatfüggőségek itt is késleltetéssel küszöbölhetők ki. A késleltetés okozta késedelmek megszüntetésére az ilyen processzorokban gyakran tartalmaznak soron kívüli utasítás-végrehajtást (angol szaknyelven out-of-order execution-t). Ez azt jelenti, hogy a függőséget okozó utasítás helyett, a processzor egy független utasításban megadott műveletet hajtat végre az éppen szabad ALU-n, a felfüggesztett utasítást pedig akkor folytatja, amikor az adatfüggőség megszűnik.

- Vektorprocesszorok
  Olyan speciális célra kifejlesztett processzorok, amelyeknek adatregiszterei többszörözve vannak, és regisztervektort alkotnak. Ezekbe kerülnek a műveletvégzéshez szükséges adatkódok, innen kerülnek a vektorművelet elvégzésére képes ALU-hoz, ami ugyanazt a műveletet végzi el mindegyik regisztertartalmon. Az eredmény is egy regisztervektorba kerül. A regisztervektorok feltöltését és az eredmény kiolvasását, általában a kódok átvitelét gyors eszközökkel végzik, így a processzor rendkívül gyors. Általános célú processzorral kombinálva, illetve speciális egységekben (például videokártyákban) használják. Nagy mennyiségű kód átkódolása (például titkosítása) nagyon gyorsan elvégezhető ezzel a processzorral.

- Multiprocesszoros rendszerek
  Az előzőekben megismert processzorok a sebességük növelését a processzor bizonyos részeinek megsokszorozásával érték el. A multiprocesszoros rendszerek ezzel szemben több teljes értékű processzort tartalmaznak. Ezek a processzorok lehetnek egyenrangúak, azaz mindegyik processzorra rá lehet bízni bármelyik folyamat végrehajtását. Ez a szimmetrikus multiprocesszoros rendszer, röviden SMP (symmetric multiprocessing). Ha a processzorok nem egyenrangúak, azaz mindegyik más-más feladat elvégzésére alkalmasak, akkor aszimmetrikus multiprocesszor-rendszerről beszélünk, rövidítve AMP (asymmetric multiprocessing). Az AMP rendszerekkel kezdődött a multiprocesszorok korszaka, és a szimmetrikus

rendszerek csak később jelentek meg a számítógépeken. Tulajdonképpen AMP rendszer őseinek tekinthetők azok a második-harmadik generációs gépek is, amelyekben az I/O elvégzésére speciális vezérlőáramköröket alkalmaztak.

- A szoftverek és a szoftvertermékeket előállító eszközök
  Az operációs rendszerek bonyolultsága, szolgáltatásai legalább olyan mértékben fejlődtek, mint a hardver eszközök. A felhasználó-gép kapcsolatot lehetővé tevő karakteres parancs-vezérlést felváltotta a grafikus felület, aminek kezelése sokkal könnyebb, és könnyebben is tanulható. Erről részletesebben a 13. fejezetben olvashatunk.
  Ebben az időszakban fejlesztették ki a strukturált programozást támogató programozási nyelveket. Strukturált programtervezésről akkor beszélünk, ha a programjainkat olyan struktúrákból (programelemekből) építjük fel, amelyeknek végrehajtása az első utasításukkal kezdődik, és az utolsó utasításukkal fejeződik be. Az így megtervezett és leírt programot tekintjük strukturált programnak.
  Ekkor jelent meg a programozásban az objektumszemlélet is, és ezzel együtt az objektumorientált programozási nyelvek és használatuk. Ebben a szemléletben a programot objektumok rendszerének kell elképzelni, és a működés során ezek az objektumok kölcsönhatása, „beszélgetése” szolgáltatja annak a feladatnak a megoldását, amihez a program készült. Az objektumorientált programozási nyelvekhez definiáltak olyan objektum osztályokat is, amelyekből – mintegy mintakollekcióból – a feladatunk megoldásához szükséges objektumok létrehozhatók. Természetesen a programozó saját objektumosztályokat is megalkothat, és ezek alapján is kreálhatók objektumok.
  Teljesen új szemléletű programozás a logikai programozás. Ezt a szemléletet a szakértői rendszerek kifejlesztésének igénye inspirálta. Ezt a filozófiát követve az instrukciókra, utasításokra épülő programozási rendszereket felváltották azok a programfejlesztési eszközök, amik nem utasításokat fogalmaztak meg, hanem következtetési szabályokat definiáltak, és axiómákat írtak le. Ezek gyűjteménye lett a program. Majd a felhasználó által feltett eldöntendő kérdésre, az axiómákra támaszkodva a következtetési szabályokat felhasználva a programot futtató rendszer megpróbálta megkeresni a választ, ami kétféle lehet; az egyik – nincs válasz, a másik megadta, hogy a kérdésre a válasz igen vagy nem. Erre a programozási stílusra próbálták 2000 környékén megépíteni a tervek szerint ötödik generációs számítógépet. A fejlesztők szerint sikerrel, a piac szerint nem.

A programozási nyelvekben megjelentek a párhuzamos működést támogató programozási eszközök. Ezek segítségével a programozó is képes volt olyan programot készíteni, amiben megadhatta, hogy a program mely részeit lehet párhuzamosan végrehajtani, és azt is, hogy ezt a párhuzamos végrehajtást hogyan kell vezérelni. A hagyományosnak mondható, egyetlen munkafolyamot – szálat – meghatározó algoritmusok mellett megjelentek olyan algoritmusok, amik a feladatot párhuzamosan végrehajtható folyamatokkal oldották meg. Ezzel teljessé vált a párhuzamos technológia: párhuzamos algoritmus, program és végrehajtás.

6.5. Az ötödik generáció

Az **ötödik generációs számítógép**ek elveinek kidolgozása, ezek alapján a gép megtervezése és megépítése napjaink feladata. A próbálkozások és a kutatások ígéretesek, de a jegyzet írásakor még nem történt meg az az áttörés, amely lényegesen megváltoztatná a számítógépet, és ezzel megteremtené azt a berendezést, amit az ötödik generációhoz sorolhatnánk.

7. Személyi számítógépek – PC-k

Tágabb értelemben személyi számítógépen vagy PC-n (personal computer) olyan általános célú számítógépet értünk, amelynek mérete, ára és szolgáltatásai lehetővé teszik, hogy bárki számára megvásárolható legyen, különösebb hozzáértés nélkül bárki, a munkahelyen, otthon némelyik típusát akár utazás vagy az utcán séta közben is használhassa.

Szűkebb értelemben PC-nek nevezzük a személyi számítógépek egy családját, az „IBM kompatibilis” PC-ket. Az IBM a világ egyik legnagyobb informatikai vállalata egyaránt foglalkozik hardvergyártással és szoftverfejlesztéssel is. A PC-család alapját képező olcsó, moduláris felépítésű számítógépet az 1981 augusztusában dobta piacra, ami azonnal sikert aratott. Később – látva a kedvező eladási adatokat – az IBM újabb és újabb modellekkel jelentkezett, míg végül más cégek is gyártani kezdték az ilyen gépeket, tovább csökkentve azok árát, és növelve elterjedtségüket. Paradox módon napjainkban épp az IBM nem gyárt személyi számítógépeket, miután az IBM-PC üzletágát 2005-ben eladta a kínai Lenovonak.

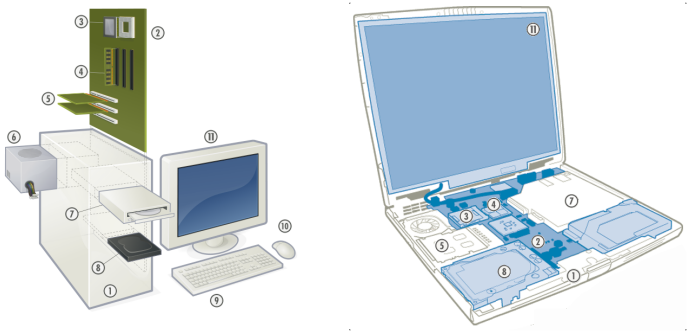
7.1. ábra. *Személyi számítógépek*

7.1. Felépítésük

Az PC asztali változatának fizikai felépítése nem sokat változott a megjelenése óta.

A különböző részegységek az alaplaphoz (2) csatlakoznak. Ezen helyezkedik el a processzor (3) és a memóriamodulok (4) csatlakoztatására szolgáló foglalatok, valamint a gép rugalmas bővíthetőségét biztosító csatlakozók. Bővíthetjük a gépet például szabványos bővítőkártyákkal (5), merevlemezes – HD – (7) vagy optikai – CD, DVD – (8) meghajtókkal. Az előbb említett eszközök a tápellátást biztosító tápegységgel (6) együtt a számítógépházban (1) kaptak helyet. A gépház hátulján további csatlakozók állnak rendelkezésre a külső perifériák, billentyűzet (9), egér (10), monitor (11) csatlakoztatására. Ezek a perifériacsatlakozók vagy közvetlenül az alaplapra vannak építve, vagy bővítőkártyákra szereltek, és ezeken keresztül csatlakoznak az alaplaphoz.

A moduláris felépítésnek köszönhetően a részegységek mindegyike cserélhető akár egy másik gyártó kompatibilis termékére is. A hordozható számítógépek tervezésekor a mobilitást helyezik előtérbe a



7.2. ábra. *Az asztali PC és a laptop felépítése*

bővíthetőséggel szemben. Ezekben a gépekben legfeljebb a operatív memória bővíthető, más nem. Az egyes részegységek cseréje is csak korlátozottan lehetséges, és a legtöbbjük felépítése a gyártóra jellemző, ezért más cégek termékei a cseréhez sem használhatók. Szabványos kivitelezésűek általában a háttértárak, a CD-meghajtók és esetleg a processzor. Ezek szükség esetén más gyártmányú termékekkel is helyettesíthetők.

7.2. Alaplap

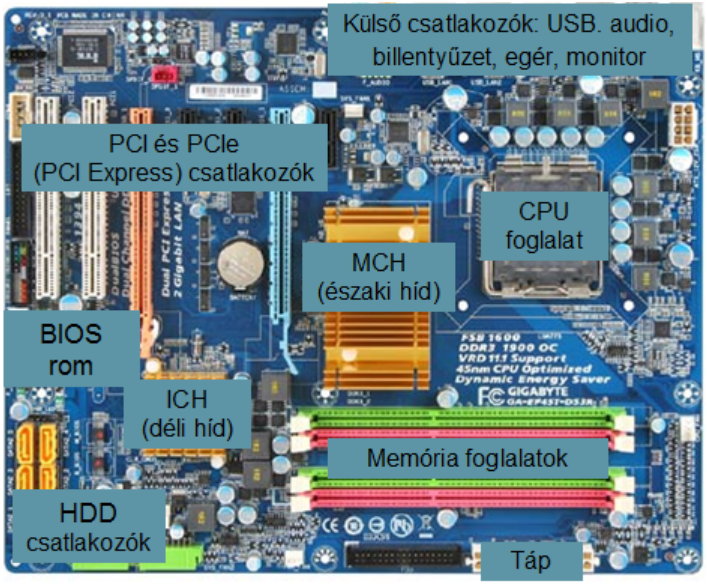
Az alaplap (angol neve mainboard vagy motherboard) egy nyomtatott- és integráltáramkör-rendszert tartalmazó lap. Áramköreinek egy része a tápegységről kapja a működéséhez szükséges elektromos feszültséget, más részüket állandó feszültséggel látja el egy elem (esetleg akkumulátor). A legfontosabb integrált áramkörei a memóriavezérlő (MMU), a megszakításkezelő, a DMA-vezérlő, az órajelgenerátor, a valós idejű óra és a flashmemória. Ezeknek az áramköröknek egy része önálló integrált áramköri lapkán (más szóval chipben), más része összevontan, úgynevezett multichipekben szokták megvalósítani. Az alaplap integrált áramköri lapkáinak összessége a lapkakészlet, más szóval a chipset. Két fontos kapcsolatot megvalósító áramkör is található az alaplapon. A CPU-t a memóriával és a gyors PCI-busszal (Peripheral Component Interconnect) vagy

a PCIe-busszal (Peripheral Component Interconnect express) összekötő egység az északi híd. A PCI-buszt a lassú ISA-busszal (Industry Standard Architecture) a déli híd köti össze. A híd (angolul bridge) feladata a különböző működésű kódtovábbító rendszerek összehangolása, illetve az egyik buszon szállított kódok átalakítása, hogy a másik buszon továbbszállítható legyen. A hidakba manapság már beépítik azokat az áramköröket, amelyek a kommunikációt segítik. Például az északi híd tartalmazhatja az MMU-t, a DMA-t, a megszakítás-vezérlőt is. A PCI buszra általában a háttértárak és a gyors perifériák, míg az ISA buszra a lassú perifériák vezérlői csatlakoznak. A modernebb PC-ken már nem használnak ISA buszt. Ezeken a déli híd egy ICH (I/O Controller Hub) azaz I/O-vezérlő csatlakozójának elosztója.

A processzor, az operatív memória nem része az alaplapnak, szabványos foglalatokba illesztve kapcsolhatók hozzá. Ugyancsak szabványos csatlakozón kapcsolódnak a különböző bővítőkártyák, amelyekkel a PC-hez kapcsolható perifériák körét bővíthetjük. Ezek a kártyák a PCI processzorfüggetlen buszra csatlakoznak. Régebbi személyi számítógépeken szinte az összes periféria ilyen kártyákkal volt csatlakoztatható, a mai modern gépeken a perifériacsatlakozók egy részét az alaplapra integrálták.

Az alaplapon található a PC elindításához szükséges ROM is. Ez a chip a régebbi architektúrájú számítógépen a BIOS-t (Basic Input Output System, magyarul alap beviteli, kiviteli rendszer) tartalmazta. Ez a programrendszer indult el a gép bekacsolásakor. Szolgáltatásaihoz tartozott a hardvereszközök állapotának ellenőrzése, és az operációs rendszer betöltése az operatív memóriába. A legújabb PC-kben az a ROM egy kisméretű flashmemória, ami csak az operációs rendszer betöltését indítja el, az összes többi szolgáltatás már ennek a dolga.

Az alaplap beépített áramkörei meghatározzák azt is, hogy milyen processzor, memória és bővítőkártya használható. A csatlakozók kialakítása is kizárja némelyik típust a bővítésből, de ez még nem elegendő a valóban alkalmazható elemek kiválasztására. Az alaplap kézikönyvében megtalálhatók azok a paraméterek, amelyekkel a felhasználni kívánt processzor, memórialapok és a bővítőkártyák köre meghatározható.



7.3. ábra. *Az asztali PC alaplapja*

7.3. Processzor

A legelső IBM PC az Intel Corporation által 1979-ben megjelentetett 8088-as processzora köré épült. Ez az egy évvel korábban kifejlesztett 8086-os processzor egyszerűsített változata. Mivel a gépcsalád fejlesztése során alapvető fontosságú volt a kompatibilitás megőrzése, ezért a későbbi gépek is a 8086 processzorcsalád tagjaira (286, 386, 486, pentium stb.) – összefoglaló nevükön – az x86 architektúrájú processzorokra épültek. Nemcsak az Intel gyárt x86 architektúrájú processzort. A legnagyobb konkurense az AMD, de több más kisebb gyártó is készít vagy készített ilyen processzorokat például a VIA, a Transmeta vagy a Cyrix.

Egy tipikus modern x86 architektúrájú asztali PC-be szánt processzor 64 bites (x86-64 architektúra). Ez

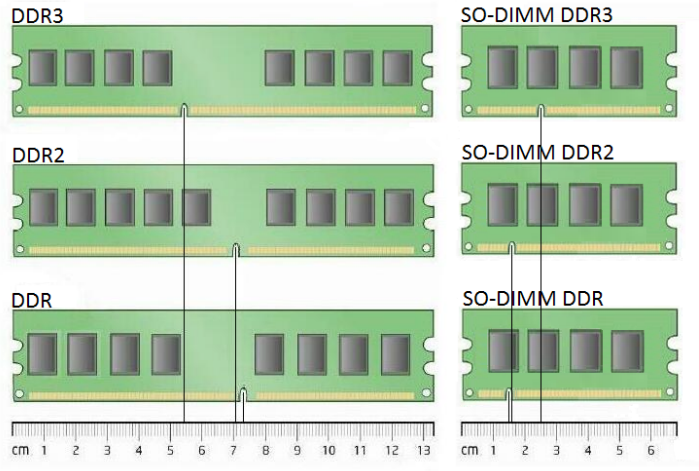
azt jelenti, hogy a regisztereinek mérete 64 bit. 2–3 GHz frekvenciájú órajelen működik, és az ismertetett technikai megoldások szinte mindegyikét (20–30 fokozatú pipeline, szuperskalár-működés, out-of-order utasítás-végrehajtás, vektorutasítás-készlet stb.) alkalmazza. A processzor sebességét az órajel frekvenciájának változtatása minimális mértékben befolyásolhatja.

7.4. Memória

Ahogy a legtöbb általános célú számítógépben – az IBM PC-kben is a DRAM egy típusát az SDRAM-ot használják operatív tárként. Bár az alapvető működési elv változatlan ennek a memóriának az évek során több, egymással nem kompatibilis típusát fejlesztették ki. Napjainkban leginkább a DDR2 és DDR3 SDRAM memóriákkal találkozhatunk a gépekben. (DDR a double data rate, magyarul kétszeres adatátviteli sebesség rövidítése. A szám a változatok sorszáma.) Különböző formátumú modulokat használnak az asztali gépekben és a laptopokban. Ezek a SO-DIMM (Small Outline Dual In-line Memory Module, azaz kisméretű kétoldali oldalanként önálló csatlakozójú memóriamodul) DDR RAM-ok. Hogy a látszólag egyforma kivitel ellenére megkülönböztethető legyen a különböző verziójú RAM, a csatlakozó felület kiképzését is megváltoztatták. a 7.4. ábrán jól látszik a különböző helyen lévő szakadási hely.

7.5. Háttértárak

A merevlemezegység, a HDD (a Hard Disk Drive rövidítése), adathordozója a merevlemez, a HD (Hard Disk). Működése során egy elektromágneses tekercccsel vezérelt mágnesező részt, illetve indukciós részt tartalmazó egység (író/olvasó fej) írja fel, illetve érzékeli, olvassa le a tárolandó – leolvasandó kódokat egy kör alakú, gyorsan forgó lemez mágnesezhető felületére – felületéről. Egy lemezmeghajtón több, egy tengelyre, egymás fölé szerelt lemez is lehet. Általában mindegyikhez két író/olvasó fej tartozik. Az adatok koncentrikus körök, vagy más néven sávok (angolul track-ek) mentén vannak tárolva. Az egymás alatt lévő sávokat cilindernek hívják. A sávok szektorokból állnak. Ezeken a szektorokon lévő kódok egy olvasási illetve művelettel írhatók, olvashatók. A mai HDD-k már több szektort tudnak együtt kezelni. Az író/olvasó fejeket egy mozgatható karrendszerre szerelik, ezáltal érhetők el a sávok a lemezen. A szektor mérete az első HDD-kben általában



7.4. ábra. *Asztali számítógépekbe és laptopokba való DDR memóriatípusok*

256 bájt volt, a mai szektorméret 512 bájt. Az átvitel gyorsítására itt is alkalmaznak cache-t, amit a vezérlő elektronikával és a csatolófelülettel együtt a merevlemez-meghajtóba építenek.

A merevlemezen tárolt kódok logikai egységeket alkotnak. Ezek a logikai egységek a fájlok (file-k). Kialakításukat a felhasználók által használt programok és az operációs rendszerek végzik. A fájlokat a lemezen egy hierarchikus rendszerbe (könyvtárrendszerbe vagy más néven mapparendszerbe) szervezik. Ezt a rendszert alapvetően az operációs rendszer segítségével a felhasználók alakítják ki. Ezekről bővebben az operációs rendszerről szóló fejezetben olvashatunk.

A HDD-k legújabban az SCSI (Small Computer System Interface, mit jelent magyarul) buszra, korábban a SATA-buszra (SATA az angol Serial Advanced Technology Attachment rövidítése) csatlakoznak, ami soros átvitelt biztosít. Régebbi gépekben a párhuzamos átvitelt biztosító PATA-buszon (Parallel Advanced Technology Attachment rövidítése, majd magyarul is jön) keresztül történt a HDD és a számítógép többi része közt az átvitel.

Az SCSI busz vezérlője általában a PCI vagy a PCIe buszok valamelyikére kapcsolódik.

7.5. ábra. *A merevlemez-meghajtó szétszedett állapotban, három merevlemezzel*

A merevlemez-meghajtók általában a számítógép belső egységei, de léteznek külső HDD-k is.

HDD-t használva tartsuk szem előtt, hogy különösen érzékeny a külső behatásokra, könnyen tönkremegy. Bár a modern meghajtók belső jellemzőik folyamatos mérésével képesek előre jelezni a közelgő meghibásodást, a merevlemezről mindig készítsünk biztonsági másolatot. A meghibásodás-jelzését figyelésére és a biztonsági mentés elkészítéséhez segédprogramra van szükség. Több ilyen program létezik.

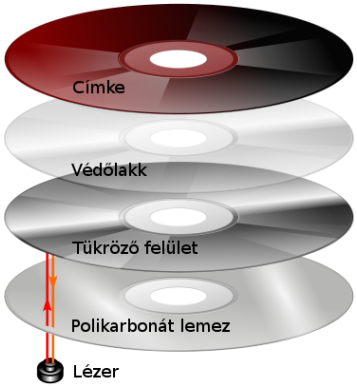
A legújabb fejlesztések következményeképpen a HDD-k mellett megjelentek a flashmemóriás háttértárak is. Ezek a teljesen elektronikus működésű SSD (Solid-State Drive magyarul szilárdtest meghajtó) tárolók. Mivel az ilyen memóriák csak korlátozott számú újraírást viselnek el, kiegészítő áramkörök gondoskodnak a cellák egyforma terheléséről és a hibás cellák helyettesítéséről. Általában a SATA-buszra csatlakoztathatók, de készülhetnek más buszra, például a PCIe-buszra vagy USB-re csatlakoztatható SSD-k is. Nagy előnyük a HDD-kkel szemben, hogy mozgó alkatrészt nem tartalmaznak. Kezelésükhöz nem kell külön meghajtó egység, vezérlőjük a memóriával egy egységet alkot. Hátrányuk, hogy kevésbé megbízhatók, azaz jóval hamarabb tönkremennek, mint a HDD-k. Egyik fajtájuk a külső egységként használható pen drive.

7.6. ábra. *A SATA buszra csatlakoztatható SSD meghajtók és belső felépítésük*

Archíválásra, nagymennyiségű kódok megőrzésére alkalmas háttértárak egy másik csoportját alkothatják az optikai elven működő tárolóegységek. Ide tartoznak a különböző CD (Compact Disc magyarul kompactlemez), DVD (Digital Versatile Disc magyarul digitális sokoldalú lemez) és a BD (Blu-Ray Disc magyarul kék sugár lemez, mivel a Blu-t az angol blue szóból alkották) adathordozó-lemezek írásra és olvasásra egyaránt képes meghajtók. Mindhárom típus rétegzett szerkezetű lemezeket használ a kódok tárolására. Az adathordozó egy fényvisszaverő képességű vékony fémréteg, amin létrehozott gödrök (pit) és púpok (bump) jelenítik meg a tárolandó kódokat. Léteznek többrétegű adathordozók is, ekkor az egyik réteg a fényt féligáteresztő tulajdonságú. Ezek a rétegek egy fényáteresztő lemezre van rögzítve, föléjük lakkréteg, majd arra címke kerül (7.7. ábra).

Leolvasásuk úgy történik, hogy egy lézersugárral átvilágítjuk a fényáteresztő lemez, ami visszaverődik a felette lévő tükröződő felületről. A gödrök és púpok helyén a visszavert lézerfény intenzitása más és más lesz. Ezt az intenzitáskülönbséget érzékeli egy detektor, amelynek kimenetéből erősítés és hibajavítás után megkapható a tárolt kód elektronikus megfelelője.

Az optikai lemezeken az adatok egy spirális pálya mentén helyezkednek el. A tárolható kódmennyiség függ a tároló rétegen lévő púpok, gödrök sűrűségétől, illetve a lemez adathordozó rétegeinek számával. Tipikus



7.7. ábra. *Az optikai lemez rétegei, felépítése*

tárolási kapacitások egyoldalas, egyrétegű 12 cm átmérőjű lemezek esetében:

7.17. táblázat. *Az optikai tárolók tipikus tárolási kapacitása*

CD	700 MB
DVD	4,7 GB
BD	25 GB

Valódi háttértárként való alkalmazásuk még nem elterjedt, mivel a meghajtók lassúak, elsősorban a kódok felírása tart hosszú ideig. Adathordozóik sem mindig újraírhatók. Ezért általában nagy tömegű adatkódok tárolásakor használjuk őket.

7.6. A PC-k bővíthetősége

A személyi számítógépeket kezdetektől fogva a széleskörű bővíthetőség jellemezte. Az alaplapon megvalósított buszokra olyan csatlakozókat szereltek, amiken egy vagy több perifériának, háttértárnak vezérlőegységét építették. Ezekhez a kártyákhoz csatlakoztatható a megfelelő periféria, háttértár. Manapság forgalomba levő PC-kben sok input, output perifériának a csatlakozója az alaplapra van szerelve, ezekhez nem kell külön bővítőkártya. Tipikus PC-buszok, amiket manapság használnak, és amikre valamilyen bővítés vagy periféria csatlakoztatható: PCI, PCIe, SATA, SCSI, USB.

Önellenőrzés

Melyik állítás igaz?

- [ ] A PCI és a PCIe buszokkal a memóriából közvetlenül lehet a perifériákra (például egy nyomtatóra) kódot továbbítani.
- [ ] Az USB-re akár 127 periféria is csatlakoztatható, és ezekre a perifériákra az USB-n sorosan érkeznek a kódok.
- [ ] Az operatív memória is az USB-re csatlakozik.
- [ ] Az USB-re háttértár nem csatlakoztatható.

III. MODUL

Operációs rendszerek és számítógép-hálózatok

Kulcsszavak: operációs rendszer, fájlkezelés, memóriakezelés, folyamatok, virtuális gép, számítógép-hálózat

E modulban a számítógépeket „életre keltő”, legalapvetőbb szoftvercsomaggal, az operációs rendszerrel ismerkedünk meg. Áttekintjük az operációs rendszerek feladatait és megismerkedünk néhány jelentősebb operációs rendszerrel. Ezután a számítógépek összekapcsolását biztosító hálózatokkal foglalkozunk.

6. LECKE

Operációs rendszerek 1.

8. Operációs rendszerek

8.1. definíció: Az operációs rendszer olyan alapvető fontosságú programcsomag, ami a hardvert közvetlenül kezeli, a felhasználók számára pedig egy egységes környezetet biztosít.

Mindannyian használtunk már operációs rendszert, akár úgy is, hogy annak a nevét sem tudtuk.

Egy számítógép megfelelő szoftverek nélkül csak áramkörök és más elektronikai alkatrészek haszontalan halmaza lenne. Bekapcsolás után ugyan áram alá kerülne, de semmi egyéb nem történne. Gondolatban lépjünk egyet és képzeljük el, hogy a gépünk mégis működik már: amikor nyomtatni akarnánk, ahhoz semmilyen segítséget nem kapnánk, hanem el kellene kezdenünk gondolkodni, hogy miféle elektronikus jeleket kellene a lézernyomtató kábelére kiadni ahhoz, hogy az épp a nekünk tetsző mintázatot égesse a papírra.

A hardver eszközeink kezelését az operációs rendszernek nevezett programcsomag végzi, a felhasználók számára pedig egy áttekinthető, egységes környezetet biztosít. Ez az oka annak, hogy ha egy régi notebookon és egy vadonatúj desktop számítógépen ugyanaz az operációs rendszer működik, akkor úgyanúgy tudunk például nyomtatni – noha a két gép hardvere szinte minden részletben különbözik. Mi ugyanazt a felületet látjuk mindkét gépen, és ennek segítségével működtetjük a gépeket. Nincs különbség a két gép kezelésében, és szinte észre sem vesszük, ha az egyik géphez egy színes tintasugaras, a másik géphez pedig egy fekete-fehér lézernyomtató csatlakozik, noha a két perifériának merőben más a működési elve.

Az operációs rendszer használatát nem lehet megkerülni. A gép bekapcsolásakor elindul, és a kikapcsolás (pontosabban a szabályos leállítás) előtt ez az utolsó program, ami más programok leállítása után utolsóként a processzort is megállítja.

A mai modern számítógépek már elképzelhetetlenek operációs rendszer nélkül. A számítógépek hőskorában, több mint fél évszázada azonban a gépkezelő személyzet, az ún. operátorok végezték a mai operációs rendszerek alapfeladatait (8.1. ábra). Az operátornak kellett odaadni a futtatandó programot, a kor technikai színvonalának megfelelő adathordozón (8.2. ábra).

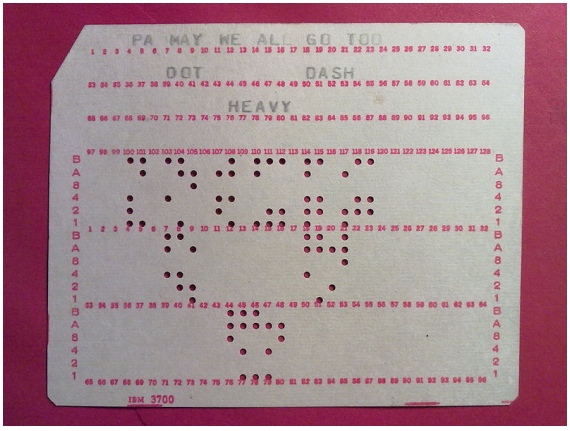
A biztonsági, ütemezési kérdésekért az operátor felelt, ő döntötte el, hogy a programot mikor lehet és kell

8.1. ábra. *Operátorok. Feladataik egy részét fokozatosan átvette az operációs rendszer (©Corbis, ismeretlen szerző)*

futtatni, a futáshoz szükséges erőforrásokat (lásd a 10. fejezetben) is ő biztosította a program számára. Amikor pedig lefutott, az eredmények olvashatóvá tételében, például nyomtatásában segédkezett és jelezte a felhasználónak, hogy készen van a program végrehajtása. Ilyen munkastílusban természetesen elsősorban nagyobb méretű számítási feladatok elvégzését lehetett számítógépre bízni, talán ez lehet az oka annak, hogy a programot végrehajtó gépet *számítógépnek* nevezték el.

9. Az operációs rendszer mint virtuális gép

Az operációs rendszer a hardver eszközök kezelése közben elrejti azok felhasználó számára érdektelen részleteit. Ehelyett egy magasabb szintű absztrakciós szintet biztosít a felhasználó számára, aki így már nem a valódi hardver eszközöket, hanem egy egységes, kényelmesebb virtuális környezetet lát. Ezen tulajdonsága alapján az operációs rendszert egy virtuális gépnek tekinthetjük. Az eszközök és a fájlok (lásd később) könnyen megjegyezhető, szimbolikus nevekkel kezelhetők, a programok kényelmesen, technikai részletek ismerete nélkül indíthatók, leállíthatók, és sok-sok magas szintű parancs segíti a munkát, például adatok másolása, vagy a hangerő változtatása. Ez a virtuális felület olyan áttekinthető és egyszerű, hogy milliók tudják kezelni, holott a gépek valódi működését csak nagyon kevesen értik közülük. A belső működési részletek elrejtésével, és



8.2. ábra. *IBM lyukkártya (fotó: SimonTrew)*

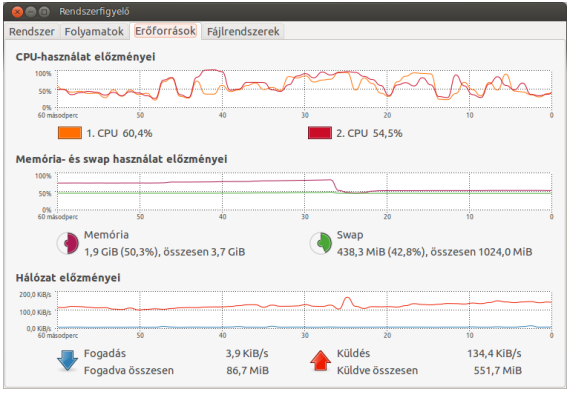
a virtuális felület alkalmas kialakításával a biztonság is megteremthető illetve fokozható. Itt nemcsak az adatok biztonságára gondolunk, hanem akár a gép védelmére is. Amíg egy autót ügyetlen kezelője könnyűszerrel tönkre tud tenni, addig egy számítógép (a fizikai megsemmisítés esetét kivéve) gyakorlatilag elronthatatlan. A számítógépen a felhasználó leginkább csak a saját adatait tudja tönkretenni, hiszen ahhoz – és tipikusan csak ahhoz – az operációs rendszer teljes felügyeleti jogot biztosít számára.

A számítógép kezelője tehát lassan elfeledi (vagy nem is sejti) hogy valójában nem a *számítógépnek* ad utasításokat, hanem a rajta futó operációs rendszer által biztosított virtuális gépnek, ami sokkal egyszerűbben, kényelmesebben kezelhető, mint a programrendszer által elfedett hardverelemek.

10. Az operációs rendszer mint erőforrás-menedzser

10.1. definíció: A számítógép erőforrásainak azokat a szolgáltatásokat nevezzük, amelyeket az operációs rendszer használatakor a felhasználó igénybe vehet.

Ha az operációs rendszer feladatát „alulról felfelé”, a gép fizikai alkotórészeinek, valamint a különböző feladatok megoldására rendelkezésünkre álló programoknak ismeretében igyekszünk megközelíteni, akkor azt mondhatjuk, hogy az operációs rendszer célja, hogy ezt az összetett rendszert menedzselje. Ennek a rendszernek részegységei a hardver és szoftver erőforrások. Egy számítógépen több tucatnyi hardver és még több szoftver eszközt kell kezelni, és ezen erőforrásokat megfelelően elosztani a gépen futó, erőforrásokért versengő programok között. A megfelelő elosztás bonyolult feladat, mert sok, sokszor ellentétes szempontnak kell megfelelnie: például legyen igazságos és optimális. Optimális, de kinek a szempontjai alapján?



10.1. ábra. *Az operációs rendszer felügyeli az erőforrásokat*

Aktivitás: Nevezze meg a a 10.1. ábrán az operációs rendszer által felügyelt erőforrásokat!

A számítógépen egyszerre több programot is futtathatunk. A számítógépünkön folyamatban lévő filmletöltés közben gondtalanul emailezhetünk, közben zenét hallgathatunk. Ezek a folyamatok a gép használója számára még akkor is párhuzamos működésűnek tűnnek, ha nem így van (lásd időosztásos működés). Van olyan feladat, ami csak sorban egymásután végezhető el. Például nyomtatásnál nem keverhetjük össze két feladat lapjait (pláne nem sorait...), így a programok csak szigorúan egymás után használhatják ugyanazt a nyomtatót.

A hardver és szoftver erőforrások mellett az operációs rendszernek az emberi erőforrásokat is kezelnie kell. Gondoljunk csak arra, hogy egy-egy számítógépet több a felhasználó is használhat, ezeket úgy kell kiszolgálni, mintha minden felhasználó önállóan dolgozna a gépen. Természetesen ezt a feladatot sem könnyű megoldani.

11. Az operációs rendszer indítása

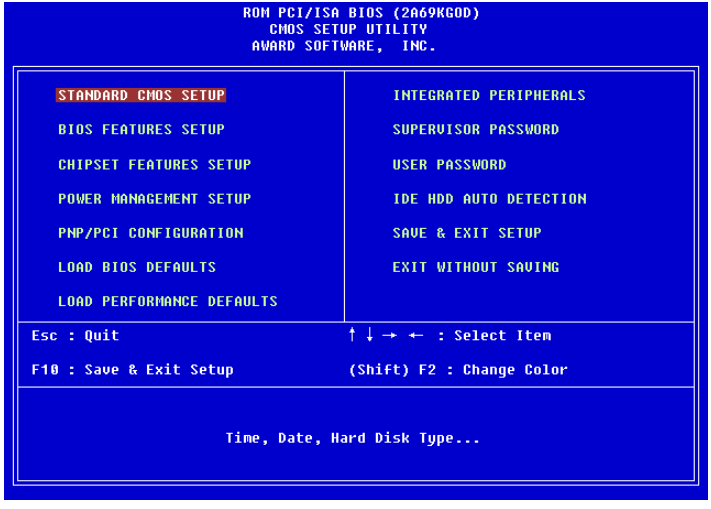
A számítógép kikapcsolva nem több, mint milliónyi elektronikus alkatrész működésképtelen halmaza. Az élettelen „vas” a különböző szoftverek által kel életre. A szoftvereket azonban szoftverek indítják, a felhasználói programok indítását egy alapvető szoftver, az operációs rendszer végzi. Néhány szót kell arról szólni, hogy miképpen tud az operációs rendszer elindulni.

11.1. BIOS

Bizonyos programok bele vannak építve a számítógép hardver elemeibe, így ezeket a hardver és szoftver analógiájára firmware-nek hívjuk (ennek a szónak nem alakult ki magyarosított alakja). Ezek az apró programok képesek többek között arra, hogy önmaguk épségét leellenőrizzék és felismerjék a saját környezetüket. Több eszköznek is vannak ilyen programjai, közülük a legfontosabb az alaplap és a videokártya BIOS-a – ezeknél általában így hívják ezt a „gyárilag” beépített programot. A BIOS a Basic Input/Output System rövidítése, tipikusan néhány megabájt méretű és flash RAM-on tárolódik, hogy szükség esetén ki lehessen cserélni egy újabb verzióra.

Alapvető feladata, hogy kapcsolatot biztosítson a hardver és szoftver részek között. A gép bekapcsolásakor, illetve némi késleltetéssel újraindítás után is a BIOS lép működésbe, és különböző alapvető tesztekkel

feltérképezi a rendelkezésre álló hardverelemeket. Az egyes elemektől kapott válaszok alapján azt is igyekszik eldönteni, hogy azok működőképesek-e. Nem teljes mélységű tesztelést végez, csak azt ellenőrzi, hogy az egyes eszközök szabályosan reagálnak-e a megszólításra. Ha ennek során komolyabb problémát talál, akkor a kiírt hibaüzenet mellett különböző ütemű csipogásokkal jelzi az egyes hibákat – ebben a fázisban még az sem biztos, hogy a videokártya elindult, és a képernyőn ezért nem a működés eredményét látjuk (esetleg semmit sem), de az alaplapra szerelt csipogó biztosan megszólal és jelzi, hogy számítógép valamelyik részegysége nem megfelelően működik. Hogy éppen melyik egység az, a hozzáértő szakember a csipogás stílusából is felismerheti.



11.1. ábra. *Egy tipikus BIOS-képernyő*

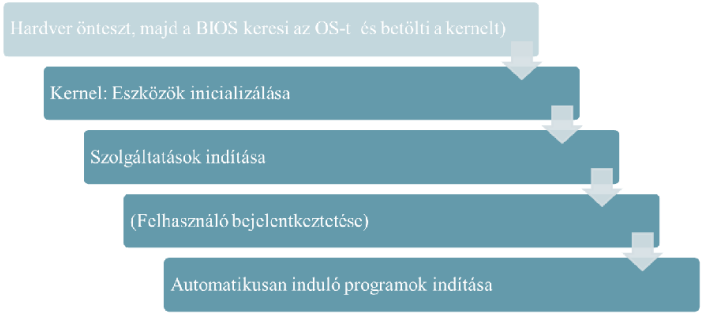
A hardverkörnyezet beállítását a 11.1. ábrán látható egyszerű felületen némileg befolyásolni tudjuk – letilthatunk egyes eszközöket, például USB portot, DVD-t vagy az alaplapi videokártyát, cserélgethetjük a háttértárak hivatkozási sorrendjét, módosíthatjuk a processzor működési frekvenciáját, beállíthatjuk a pontos időt, befolyásolni tudjuk a gép energiagazdálkodási elveit, stb. Valójában csupa olyan dolgot, amit a mai

operációs rendszerekkel is be lehet állítani. Ezértazt kell, hogy mondjuk, a BIOS szerepe egyre csökken, kiváltása folyamatban van az eredetileg az Intel által fejlesztett UEFI (Unified [vagy Universal] Extensible Firmware Interface, magyarul Egységes Bővíthető Firmware Interfész) rendszerrel (eredetileg EFI-nek hívták). A BIOS egy több évtizedes technológia terméke, folyamatos foltozgatással tartják életben. Lassú, kezelése nehézkes, az új eszközök hozzáillesztése egyre nehezebb.

Aktivitás: Ha lehetőségünk van rá, lépjünk be a gépünk BIOS-ába és tekintsük meg, milyen beállítási lehetőségeket kínál. Indokolatlanul semmit ne módosítsunk.

A BIOS leváltása a sok meglévő eszköz miatt nem egyszerű, ezért lassan halad. Így még ma is a BIOS indul el először a mai gépen döntő többségén (11.2. ábra). Ha a hardverek gyorstesztjén túl van, akkor a beállításai ban megadott hely(ek)en elkezdi keresni az operációs rendszert. Ha indíthatót talál, nem keres tovább, hanem elkezdi azt betölteni. Maga az operációs rendszer a rendszermag, az úgynevezett kernel betöltésével indul: ez az operációs rendszernak legalapvetőbb szolgáltatásokat nyújtó része. A hardver erőforrások, többek között a processzor, a memória kezeléséért felel. A kernel inicializálja az számítógép részegységeit, majd elindítja a különböző speciális feladatokat elvégző, folyamatosan futó programokat. Több tucatnyi olyan programot elindít, amelyek egy-egy konkrét funkcióval (szolgáltatással) bővítik az operációs rendszer képességeit. Ilyen például a nyomtatási sorok, különböző biztonsági funkciók kezelése, a felhasználók ki-be jelentkeztetése.

A szolgáltatásokat nyújtó programok elindítása után az operációs rendszer gyakorlatilag működőképessé vált, a felhasználó utasítására vár, aki elkezdheti a munkát a számítógépen, vagy a rendszer beállításától függően névvel és jelszóval azonosítva magát vezérelheti az operációs rendszert vagy indíthatja felhasználói programjait. A bejelentkezés után általában további programok is elindulhatnak. Ezek többnyire felhasználóhoz tartozó apró segédprogramok. Ilyen például a képernyőre kitett valós időt mutató óra vagy időjárás-előrejelző piktogram, de lehet olyan program is, amivel a felhasználó legtöbbször dolgozik.



11.2. ábra. *Az operációs rendszer indulása*

12. A számítógépek működésének szoftveres feltételei

Az operációs rendszer kitüntetett szerepet foglal a szoftverek között. Természetesen nem „kötelező” operációs rendszert használni, de akkor annak funkcióit a gépen futtatott szoftvernek magának kell biztosítania – ez legtöbb esetben csak óriási többletterhet jelentene a szoftver készítőjének, és talán semmi előnnyel sem járna. Ezért szoktuk azt mondani, hogy az operációs rendszer a számítógép „alapfelszereléséhez” tartozik, ami nélkül működésképtelen. Most tekintsük át, hogy mik is az operációs rendszer legfontosabb részei!

- Kernel
  A kernel a hardver erőforrások kezeléséért felel. Egy „nem látható” program, nem tartozik hozzá felhasználói felület (lásd a következő pontot), mindvégig fut, de nem látjuk, ha például valamilyen segédprogrammal megnézzük a futó programok listáját. A kernelt szokás magyarul *rendszermagnak* is nevezni, de inkább az angol megnevezés terjedt el, ezért a továbbiakban is ezt használjuk.
  Alapvetően kétféle megközelítésben szokták a kernelt összeállítani, ez alapján beszélünk monolitikus és mikrokernelről. A *monolitikus kernel* egyetlen, hagyományos értelemben vett nagyobb program, amibe minden szükséges funkciót, az összes szükséges hardverelem kezelőprogramját (ún. meghajtóprogram, driver) beépítették. *Mikrokernelen* egy jóval kisebb programot értünk, ami még a kernelfunkciók közül

is csak a legfontosabbakat tartalmazza, és a futás közben betölthető modulok felhasználói programként tölthetők be. Ezzel a megoldással könnyebb biztosítani a (sokkal kisebb) kernel stabilitását, hibamentes működését, ráadásul kényelmes, hogy a kernel csak a valóban szükséges részeket tölti be a memóriába, és a betöltött modulok készlete bármikor változtatható. Az egyes modulok elszeparáltabbak, így a biztonsági és stabilitási problémák könnyebben kezelhetők. A megoldás hátránya egy monolitikus kernelhez képest, hogy – elvileg – némi teljesítményromlást jelent a modulok közötti interfészek kezelése.
A két alapelvet a modern kernelek sokszor ötvözik azzal, hogy választható, mely részek kerüljenek be monolitikus kernel módjára az állandóan futó központi részbe és melyek azok a kevésbé fontos, vagy opcionális részek, amiket akkor kell betölteni, ha erre szükség van. Ez a kerneltípus a *hibrid kernel*.

- UI (User Interface) – felhasználói felület
  A számítógépet emberek használják, ezért mindenképpen szükséges egy olyan szoftverkomponens, ami azt biztosítja, hogy a kimeneti adatok emberi érzékszervekkel és intelligenciával felfogható legyenek, a felhasználó pedig a gépnek szánt utasításait megadhassa.
  A 13. fejezetben részletesen foglalkozunk a felhasználói felületekkel.

- Segédprogramok (utilities)
  Az operációs rendszert sok-sok segédprogram egészíti ki a teljesebb, kényelmesebb működés érdekében.
  A definíció alapján nem teljesen egyértelmű, hogy ezek közül pontosan melyeket kell az operációs rendszer részének tekinteni, melyek azok, amik hasznos segédprogramként támogatják a felhasználó munkáját. Következzék néhány, a sokszáz közül:

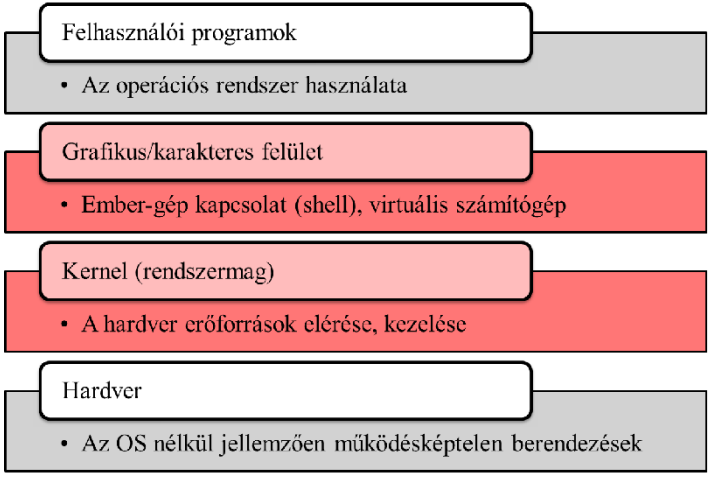
  – pontos idő beállítása, megjelenítése,
  – háttértárak fájlrendszerének menedzselése,
  – szabad hely meghatározása az adattárolókon,
  – tömörítés támogatása,
  – időzítve történő programindítás, például mentés minden éjjel,
  – képek megjelenítése,
  – email küldés parancssorból,

- képernyő háttérképének beállítása,
- konverzió fájltípusok között,
- fájl másolása, törlése, átnevezése.

A listát a végtelenségig folytathatnánk, a szabad operációs rendszerek szinte kimeríthetetlen lehetőséget biztosítanak a munka támogatására, a fizetős rendszerek készítői sokkal szűkmarkúbbak és jelentősen korlátozottabb segédprogram-készletet biztosítanak.

- Fejlesztőeszköz
  Az operációs rendszer biztosítja, hogy a számítógépünk működőképes legyen, de nem ad lehetőséget arra, hogy tetszőleges problémát megoldhassunk vele. Programot kell írnunk vagy íratnunk az operációs rendszerünkhöz a problémáink megoldásához. Ehhez azonban *fejlesztőeszközre* van szükségünk. Egy operációs rendszer attól válik igazán erőssé, hogy fejlesztőrendszert is biztosítanak hozzá. A felhasználó ugyanis az esetek többségében nem a „gépet” szeretné kezelni, hanem saját céljaira készült programot szeretne használni.
  Ezenkívül magát a fejlesztőeszközt és az operációs rendszert is fejleszteni kell, annak összes segédprogramjával együtt. Eljutottunk a tyúk és tojás elsőbbségének a problémájáig, ezért meg kell említeni, hogy alapvetően kétféle környezetre készíthet programot egy fejlesztőrendszer: az *in-platform* fejlesztés olyan programot készít, ami a fejlesztőeszközt futtató operációs rendszeren és hardveren fut, az ún. *cross-platform* fordítók segítségével azonban más operációs rendszerre, akár más hardverre is készíthetünk szoftvereket.
  Egy teljesen új operációs rendszer első példányát mindig egy másik, már létező operációs és fejlesztő rendszer segítségével készítik el. A továbbfejlesztéshez ez a változat már felhasználható lesz.
  Az eltérő hardver- és szoftverkörnyezetben történő fejlesztésre példa lehet egy okostelefonon futó operációs rendszer vagy felhasználói program kifejlesztése, hiszen nehezen képzelhető el – bár elvileg lehetséges lenne –, hogy a pár centiméteres érintőképernyőn egy ujjal gépelve készült. Kényelmesebb egy önálló fejlesztői számítógépen fejleszteni, és a megfelelő hardverre lefordított kódot másolni át a tényleges futtató hardverre.

A 12.1. ábrán látható, hogy a felhasználó illetve a felhasználói programok miképpen érik el a hardvert. A felhasználó vagy valamely felhasználói programmal dolgozik (ez esetben az adott program veszi igénybe az operációs rendszer szolgáltatásait), vagy közvetlenül az operációs rendszert használja annak valamilyen felületén keresztül. A kernel éri el ténylegesen a számítógép hardverelemeit.



12.1. ábra. *Az operációs rendszer a hardver és a felhasználói programok között*

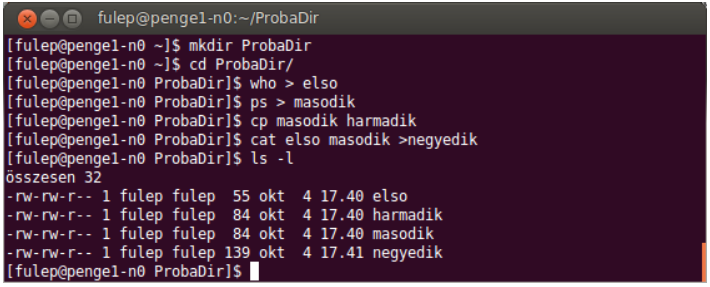
13. Felhasználói felületek

A felhasználói felület az operációs rendszernek nagyon fontos része, ezen keresztül tartható a kapcsolat az ember és az operációs rendszer között. Kidolgozottsága, használhatósága alapjaiban határozza meg magának az operációs rendszernek a használhatóságát, illetve a felhasználónak a használhatóságról alkotott képét. Valójában a felhasználók egy széles tábora kizárólag a felhasználói felület alapján képes megítélni és ítélni meg az operációs rendszert – keveset tud az operációs rendszer memória- és folyamatkezeléséről, biztonsági

kérdéseiről és egyéb fontos részletekről. A használat során azonban megtapasztalja, hogy mennyire intuitív, magától értetődő és szép az adott felhasználói felület, amivel végeredményben folyamatosan kapcsolatban van a képernyőn keresztül.

A szoftvergyártók is felismerték ennek jelentőségét, ezért szebbnél szebb, egyre praktikusabb és kevesebb szakismerettel kezelhető felületekkel igyekeznek felhasználóikat elbűvölni. Régebben egyszerú karakteres képernyőre parancsokat kellett tudni precízen begépelni, később egyre többmindent tettek egérrel irányíthatóvá, manapság pedig azzal próbálkoznak – sikerrel –, hogy természetes ujjmozdulatokkal, rámutatással lehessen irányítani az operációs rendszert. A használat jellege alapján három alapvető felhasználói felülettípust különböztetünk meg.

1. CLI: parancssori interfész (Command Line Interface)
   A parancssori interfész a legrégebbi éppen ezért technikailag a legegyszerűbb típus. A monitoron fix méretű karakterek, és a billentyűzet kizárólagos használata jellemzik. A kommunikáció parancsok begépeléséből és az üzenetek elolvasásából áll. A kisebb erőforrásigény miatt a régebbi rendszereknek csak parancssori felületük volt, ezért manapság többen hajlamosak a karakteres felületet lebecsülni és korszerűtlennek tekinteni. Ez azonban csak a hozzá nem értő felhasználókra jellemző: a karakteres felület ma gyakorlatilag minden korszerű operációs rendszernek is szerves része, mert sokszor hatékonyabb és mindenképpen teljesebb, mintha csak a szebb, grafikus felületen keresztül lehetne az operációs rendszert irányítani. Amíg a grafikus fájlkezeléshez csak néhány állítási lehetőség van (pl. ikonos vagy listás megjelenítés, 3-4 rendezési szempont), addig a Linux mappatartalom-listázásra való ls parancsának kb. 60 olyan kapcsolója van, amivel a megjelenítést szabályozhatjuk.)
   A 13.1. ábrán egy parancssoros munkaablakot láthatunk, néhány egyszerű paranccsal.
   A sorok elején lévő ismétlődő rész (a példában a sor elejétől a $ jelig tart) az ún. prompt, aminek pontos tartalmát a felhasználó szabályozhatja. Különböző információk megjelenítésére használható, és legfőbb célja, hogy az operációs rendszer szöveges parancsokat értelmező programja ennek kiírásával (és a mögötte megjelenített kurzor villogtatásával jelezze, hogy a felhasználó parancsára vár.
   A parancsok a prompt után következnek és a következőképpen épülnek fel: először be kell gépelni a parancs nevét (azonosítóját), példában ezek: mkdir, cd, who, ps, cp, cat, ls. Ezután az egyes parancsokra jellemző

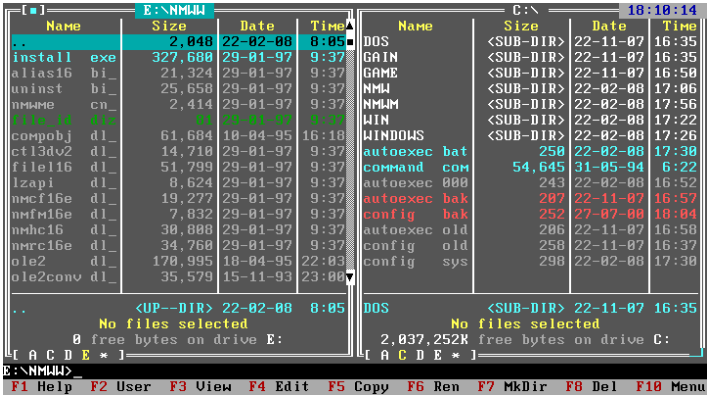


13.1. ábra. *Munka parancssori felületen*

kötelező vagy választható, egy vagy több ún. argumentum vagy paraméter következik, ezek lesznek a parancs végrehajtását végző programrész számára átadott adatok. Például az mkdir parancs új mappát hoz létre, argumentuma a létrehozandó mappa neve (a példában *ProbaDir*); a cp parancs fájlt másol, ehhez logikusan két argumentumra van szüksége: a *masodik* a forrásfájl neve, a *harmadik* pedig a célfájlé. Az egyes parancsok működése ún. kapcsolókkal módosítható. A kapcsolók - jellel kezdődnek. A példában az ls parancs kapcsolója a -l (az angol long, s hosszú jelentésű szó rövidítése), amivel egy bővebb (hosszú) listát kérhetünk az adott mappa fájljairól. A parancs végrehajtásának eredménye a parancs begépelése (és az Enter megnyomása után) közvetlenül a képernyő következő sorától kezdődően jelenik meg a képernyőn.

Érdekesség: Sose becsüljük le a parancssori felületeket, sokszor nagy segítségére van a hozzáértőknek. Tegyük fel, le kell töltenünk az internetről egy tömörített (zip) fájlt, amiben egy szövegfájl található, és ennek első háromszáz sorát emailben el kell küldenünk valakinek. Bárki hamar elvégzi, de unalmassá válik a feladat, ha kiderül, hogy ezentúl óránként kell ezt elvégezni a munkahelyünkön. A kedves olvasó hogyan oldaná ezt meg? Linux alatt a következő parancs gépelhető be:
while true; do wget -q -O - http://itt.hu/ez.zip | gunzip | head -300 | mail ide@cim.hu; sleep 3600; done
Ezzel a feladatot megoldottuk, parancsunk óránként el fogja küldeni emailben az újra és újra letöltött fájl megfelelő részét, anélkül, hogy a felhasználónak bármit is tennie kellene. Biztosan sokkal kényelmesebb és hibamentesebb ezt a parancsot egyetlen alkalommal kigondolni és kiadni, mint óránként, éveken keresztül ugyanazt a feladatot alkalmanként 3 perc alatt „kézzel”, webböngésző, tömörítő, szövegszerkesztő és levelező program segítségével végrehajtani. Nem részletezzük, minimális módosítással az is megoldható, a fenti parancs a számítógép következő újraindítása után, tőlünk függetlenül is folyamatosan fusson, amíg le nem állítjuk.

2. TUI: Szöveges interfész (Text User Interface) A parancssori interfész nyilvánvaló előnyei ellenére egyértelmű, hogy bizonyos gyakran ismétlődő, jól behatárolható feladatokra kényelmesebb megoldásra van szükség.
   A parancssori interfészre jellemző minimális erőforrásigényű, de a képernyőt már nem csak írógép-stílusban használó felhasználói felület az ún. „szöveges interfész”, amit néha „álgrafikus interfésznek” is neveznek, tekintve, hogy karakteres felületen ablakos technikára emlékeztető képernyőképeket valósít meg.
   Kizárólag szöveges interfészt használó operációs rendszerek nem terjedtek el. Azonban a meglévő operációs rendszerekhez készítettek ilyen felületű felhasználói programokat. Fénykorukat talán a PC-s környezetben, a DOS parancssori operációs rendszer idején élték. Első igazán sikeres képviselőjük a Norton Commander volt, ami egész lavinát indított el a fájlkezelő- és mindenféle segédprogramok fejlesztésében. Ezek a programok közvetlenül az operációs rendszer funkcióit bővítették és tették kényelmesebbé. Időnként csupán egy jó ötleten alapuló megjelenítési móddal, máskor az operációs rendszer egyes funkcióinak átvételével. A DOS Navigator nevű segédprogram (13.2. ábra) sokkal gyorsabban tudott fájlokat másolni, mint az operációs rendszer, és megvalósított egy olyan funkciót, ami egy „csupasz” DOS operációs



13.2. ábra. *Szöveges felhasználói interfész*

rendszeren elképzelhetetlen, megoldatlan volt: egyidőben több programot is el tudott indítani, amik között váltogatni lehetett.

3. GUI: grafikus interfész (Graphic User Interface)
   A felhasználói felületek harmadik típusát talán senkinek nem kell bemutatni: grafikus képernyőn jelennek meg az információk, nem különülnek el élesen a grafikus és szöveges elemek. A felhasználó a billentyűzet mellett különböző mutatóeszközöket, leggyakrabban egeret használhat, vagy épp a saját ujjait érintésérzékeny képernyők esetében.
   A grafikus felület jellemző, közismert velejárói az ablakok, ikonok, gombok, gördítősávok és rövid üzeneteket megjelenítő, időnként felpattanó „buborékok”.
   A grafikus felhasználói felület megjelenése ugyan a felhasználók barátságosabb kiszolgálásának igénye miatt jött létre, de ennek feltétele a számítógép hardverképességeinek jelentős növekedése volt. Az erős hardverszolgáltatás megléte viszont felszabadította a felületetkészítők fantáziáját: sok-sok új ötlet, lehetőség segíti a szebb, használhatóbb garfikai felületek kialakítását. Talán egyedül a felhasználói megszokás a gyakori és gyors változások komoly korlátja: a felhasználókat általában nehéz a megszokottól

jelentősen eltérő felületre átszoktatni.
Manapság a grafikus interfész a felhasználók leggyakrabban használt, közkedvelt felülete. Kevesebb szaktudással, élvezetesebben, kényelmesebben használható mint más felületek. Ráadásul „visszafelé kompatibilis”, ami azt jelenti, hogy garfikus felület esetén is biztosított a parancssori interfész azokra a feladatokra, amiknek megoldására a grafikus interfész sok esetben alkalmatlan vagy kevésbé kényelmes.

Az ember-gép kapcsolatában némelyik hardverelem csak arra szolgál, hogy ezt a kapcsolatot támogassa, illetve megvalósításában részt vegyen. Ilyen például a monitor, ami szinte kizárólag arra való, hogy a felhasználói kapcsolattartásért felelős szoftver segítségére legyen azzal, megjelenítse azt a felületet, amin keresztül a felhasználó eljuttathatja a számítógép operációs rendszeréhez akaratát. illetve arra, hogy a programok felhasználónak szánt „üzeneteit” megjelenítse. Hasonlóan, az egér és a billentyűzet szinte egyetlen célja, hogy a felhasználótól adatokat, parancsokat fogadjon. Ezek nélkül a számítógépek kezelése szinte elképzelhetetlen.

Önellenőrzés

1. A következők közül melyek kezelése az operációs rendszer feladata?
   - [ ] Memória kezelése
   - [ ] Hálózat kezelése
   - [ ] Emailek kezelése
   - [ ] Fényképek kezelése
   - [ ] Fájlok kezelése
2. Melyek azok a BIOS-funkciók, paraméterek, amelyeket a mai operációs rendszerek önállóan is el tudnak végezni, be tudnak állítani!
   - [ ] Pontos idő beállítása
   - [ ] Energiagazdálkodási funkciók
   - [ ] Processzor teljesítmény
   - [ ] Különböző eszközök letiltása
   - [ ] Különböző interfészek letiltása
   - [ ] Vírusos emailek blokkolása
3. Saját napi munkája során milyen esetekben használja az operációs rendszere grafikus interfészét, és milyen esetekben részesíti előnyben a parancssort?

7. LECKE

Operációs rendszerek 2.

14. Folyamatok

14.1. definíció: A folyamat (processz) egy – párhuzamosságot nem tartalmazó – program éppen futó, végrehajtás alatt lévő példánya.

A folyamatok kezelése az operációs rendszerek nélkülözhetetlen szolgáltatása.

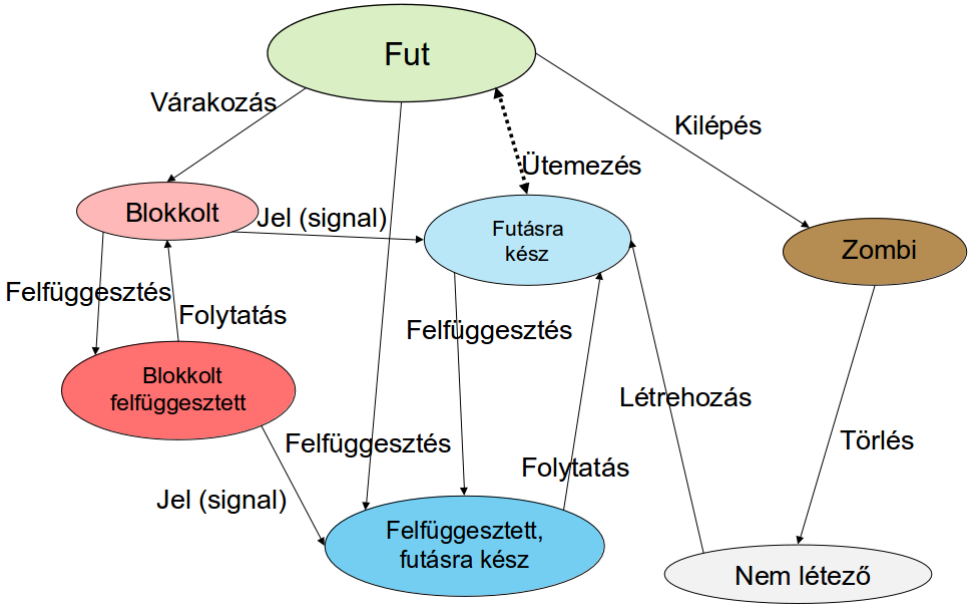
A *programjaink* háttértárakon helyezkednek el, ennek megfelelően statikusnak tekinthetők: sok fájlból állhatnak, van méretük, ennek megfelelően egy konkrét helyet elfoglalnak a merevlemezen vagy egy pendrájvon, egyikről a másikra másolhatjuk őket, stb. A *folyamatok* ezeknek a programoknak a futó példányai. Egy folyamatnak különböző állapotaik vannak: megszületik („épp indul”), fut, várakozik, majd egyszer befejeződik és megszűnik. Az állapotátmenetről a folyamat sokszor nem maga dönt, hanem az operációs rendszer szabályozza (ütemezés, lásd 15. fejezet). A folyamat futás közben használja a processzort (vagy más megfogalmazásban, a processzor „futtatja”), használ egyéb erőforrásokat, memóriát.

A folyamatok menedzselése a modern operációs rendszerek egyik legfontosabb tevékenysége.

Folyamatot csak folyamat indíthat – az operációs rendszer indulásakor elindul egy ún. ősszülő folyamat, ami az összes többi folyamatot elindítja. Ezekből aztán egy hierarchikus rendszer alakul ki.

Az operációs rendszer nyilvántartja a futó folyamatokhoz tartozó információkat (ezt a folyamatosan változó nyilvántartást hívják processz táblának). Ebben táblázatszerűnek képzelhető adathalmazban található minden, amit a folyamatok futásának adminisztrációjához kell: a futtatandó kód és a hozzá tartozó adatterület elhelyezkedése, regisztertartalmak, menedzsmentinformációk (pl. jogosultság) stb. Ezen adatok egyetlen folyamathoz tartozó összességét folyamat tartalomnak (process context) hívjuk.

A 14.1. ábra a különböző folyamat állapotokat mutatja. Egy új folyamat létrejötte a nemlétező (Non-existent) állapotból indul és bekerül a futásra kész, tehát futásra várakozó folyamatok listájára. (Egyszerű, egyprocesszoros esetben ezek közül mindig csak 1 futhat, azt pedig az ütemező választja ki, lásd 15. fejezet.) Az épp futó (Running) folyamatot vagy az ütemező teszi vissza egy idő után a futásra várakozók sorába, vagy „saját akaratából” kerül blokkolt állapotba, mert például egy számára szükséges erőforrásra vár. A folyamatokat



14.1. ábra. *Folyamat állapotok és állapot-átmenetek*

futásukban fel lehet függeszteni (Suspended), ez természetesen a blokkolt, a futó és a futásra váró állapotban is megtörténhet. Futó állapotból a felfüggesztés megszűntével futásra váró állapotba kerül a folyamat, a blokkolt állapotból pedig úgy kerülhet ki, hogy egy jelzés (signal) értesíti arról, hogy a blokkolást kiváltó körülmény megszűnt. Egy folyamat befejezésekor annak futása megszűnik (Zombie), de valamely pozitív idő szükséges a folyamat tartalom törléséhez, majd ezután – ekkor már nem is beszélhetünk folyamatról – a nemlétező állapotba kerül. A futásra képes folyamatok közötti menedzsment feladatokról külön, a 15. fejezetben ejtünk szót.

15. Ütemezés

Az ütemezés célja az azonos erőforrásokra igényt tartó folyamatok közötti választás, az erőforrás kiosztása. Erőforrásként most elsősorban a processzorra gondolunk, így az operációs rendszer ütemezési tevékenységén azt a folyamatot értjük, amikor a processzorra várakozó („futni akaró”) folyamatok közül az operációs rendszer mindig választ egy folyamatot, amelyik a következő időszakban futhat.

Az ütemezésnek több célja is van:

- Minden folyamat lefusson
  Triviális cél annak megszervezése, hogy valamilyen módon minden folyamat kapjon processzoridőt, és tudjon futni.
- Válaszidő csökkentése
  A felhasználó aktivitása és a rá adott első válasz között eltelt idő minimalizálása. Ha egy felhasználó például két ablakban két külön programot futtat, jogosan elvárja, hogy bármelyiket kezeli, úgy érezze, hogy mindkét a program „azonnal” reagál a lenyomott gombokra, holott tudjuk, hogy egyprocesszoros gépen egyszerre valójában csak 1 folyamat futhat.
- A processzor, a rendszer hatásfokának növelése
  - a processzor minél nagyobb kihasználása, ha van futó feladat,
  - átbocsátó képesség növelése (egységnyi idő alatt befejezett feladatok számának növelése, végrehajtási idő csökkentése),
  - átlagos várakozási idő minimalizálása.

Az ütemezés alapesetben a következőképpen működik: az operációs rendszer ütemezésért felelős része, az ütemező (scheduler) rövid időközönként megállítja az éppen futó folyamatot, elment minden adatot, ami majd a futás folytatásához szükséges, és ezek helyére betölti a futásra kiválasztott következő folyamat adatait. (Mint korábban már említettük, ezt az adathalmaz a processz tartalom. A folyamatváltást pedig az angol szakirodalom a process context switch kifejezéssel írja le.)

Az ütemező a fenti feladatot bonyolult, ún. ütemezési algoritmusok segítségével oldja meg. Ezeknek az algoritmusok két alapvető részből állnak:

1. a folyamatváltás időzítésének megtervezése,
2. a következő futtatandó folyamat kiválasztása.

Az ütemező algoritmus a célok összehangolására különböző optimalizációs stratégiákat valósít meg – ezek ugyanis sokszor ellentétesek. Ha egy folyamatot úgy próbálnánk a leggyorsabban lefuttatni, hogy a futása alatt többé nem vennénk el tőle a processzort, ez katasztrofális hatással lenne az operációs rendszer interaktív felületére, és a többi folyamat működésére: az adott program futása alatt azt tapasztalnánk, hogy az összes többi program „lefagyott”, még az egér, vagy a képernyő sarkában az óra se mozdul – ugyanis ezeket is ütemezett folyamatok irányítják.

Az egyik stratégia, amellyel ütemezni lehet, a prioritás figyelése. A folyamatok ugyanis elsőbbségi fontossági sorrend szerint kategórizálhatók. Ez a besorolás a futás során akár dinamikusan is változhat de a felhasználó állandó értékre is beállíthatja. (Például az előtérben futó a felhasználó által éppen kezelt folyamat előnyt élvezhet a háttérben futó, de órák óta nem használt programmal szemben.) Ezeket az ütemező prioritási sorokban, a folyamatok „életkorát” is figyelő ún. process aging algoritmusok segítségével kezeli.

16. Virtuális címzés

A memória egyike a számítógép legfontosabb erőforrásainak, így az operációs rendszer egyik legfontosabb feladata a memória kezelése. Minden folyamat saját memóriaterülettel rendelkezik annak érdekében, hogy egymástól függetlenül futhassanak. A folyamatok a saját futásukhoz szükséges memóriaterületet megigénylik és megkapják, azonban a számítógép véges memóriája így nagyon hamar elfogy. Az is belátható, hogy valójában nincs minden folyamatnak folyamatosan szüksége a teljes saját memóriatartalmára, hiszen jellemzően nem „mindent” használ egyidőben. Az épp nem futó folyamatoknak pedig átmenetileg egyáltalán nincs szükségük a memóriára.

Előfordulhat, hogy a folyamatoknak lényegesen nagyobb címtartományra van szüksége, mint amennyit a fizikai címtartomány biztosít. Ezért, hogy a fizikai memória nagysága ne korlátozza a folyamatokat, a címtartományt függetlenítjük a valós memórianagyságtól. Ezzel elérhető, hogy már a program írójának se kelljen figyelnie arra, hogy a programja milyen memóriával rendelkező számítógépen fut majd. Ezt úgy érjük el, hogy a valós memória mellett egy képzeletbeli, úgynevezett virtuális memória használatát tételezzük fel minden program esetén, és ezt a memóriát képezzük le az 5.3. fejezetben ismertetett módon a valós memóriára. Kétféle memóriáról beszélünk tehát:

- Fizikai (valós) memória
- Virtuális memória

A fizikai memória az, ami az adott számítógépben ténylegesen rendelkezésre áll. Ebben kerülnek végrehajtásra a folyamatok. A virtuális memória pedig rendszerint egy partíció vagy egy fájl egy gyors elérésű háttértáron.

A leképezés irányítását az operációs rendszerre szokás bízni, de a hardverelemek fejlődéseképpen létrejövő MMU jelentős segítséget ad ennek a feladatnak ellátására.

A számítógép tényleges memóriája, a valós memória lapkeretekre van osztva. A virtuális memória e keretekkel megegyező méretű lapokból fog állni. A lapok száma jóval nagyobb is lehet, mint a keretek száma.

A fizikai címtartományt logikailag n lapkeretre (page frame) osztjuk. Egy-egy keret akkora, hogy a virtuális címtartomány egy-egy memórialapja (page) épp beleférjen. Természetesen sokkal több memórialap (N) van, mint lapkeret: $N >> n$.

Természetesen a folyamatok továbbra is csak a fizikai memóriát tudják használni. Futás közben az éppen használatban lévő memórialapoknak a fizikai memóriában kereteiben kell lenniük. Amikor pedig olyan memóriaterületre hivatkoznak, ami nincs a fizikai memória valamely lapkeretében (ez az állapot a laphiba v. page fault, amit az operációs rendszer önállóan, de esetleg hardversegítséggel vehet észre MMU), akkor a rendszer valamely lapkeret tartalmát swap-fájlba menti, és a szükséges lapot betölti a ugyancsak swapből. Az operációs rendszer feladata, hogy eldöntse, mely lapkeret tartalmát szabad, illetve érdemes kicserélni. Ezt a

feladatot a page replacement algoritmus, magyarul lapcserélő algoritmus vezérli, de ebben ugyancsak segíthet az MMU. A lecserélt lapról az operációs rendszer eldönti, hogy történt-e rajta változás annak legutóbbi betöltése óta: ha nem, akkor egyszerűen kidobható, hiszen a virtuális memóriában megvan az adott lap „eredetije”, ha azonban módosult, akkor a csere előtt vissza kell írnia a lap tartalmát a swap-fájlba. Az elindított, de éppen nem futó (pl. valamely erőforrásra, vagy épp felhasználói beavatkozásra várakozó) folyamatok pedig nem foglalják az értékes memóriát a futó folyamatok elől.

Jegyezzük meg azonban, hogy a fizikai memória és a háttértárak elérési ideje között kb. két nagyságrendnyi különbség van, tehát ha egy program futása alatt sokszor van lapcserére szükség, akkor annak működése jelentősen lelassulhat. A fizikai memória megfelelő méretét ez a megoldás nem pótolhatja, de nem is arra való elsődlegesen. A programozó annyit tehet a gyorsabb működés érdekében, hogy természetesen memória-takarékosra írja meg a programját, és ügyel arra, hogy a szükséges memóriahasználat során az egymás után gyakran változó memóriatartalmak lehetőleg egymáshoz közel legyenek – így ha futás közben befférnek ugyanazokba a lapokba, akkor jelentősen kevesebb laphiba fogja lassítani a program futását.

A virtuális memóriakezeléssel a modern számítógépek hardveres bemutatásánál, a 6.4. fejezetben is foglalkoztunk.

Aktivitás: Legalább két különböző operációs rendszeren kérdezze le, használ-e az adott rendszer virtuális memóriát, és ha igen, mekkora méretűt. Ellenőrizze, van-e elegendő szabad memória a programok futtatásához. Az operációs rendszerek súgói a segítségére lesznek. (Windowson a systeminfo, Linuxon a free parancs használatát javasoljuk.)

Windows rendszeren, mint oly sok esetben, a virtuális memória használatával kapcsolatos beállítások után is újra kell indítani az operációs rendszert. Linuxon futás közben módosíthatunk a virtuális memória használatán is.

Aktivitás: A 10.1. ábrán már láthattunk példát arra, miként jeleníti meg egy Linux operációs rendszer grafikus felületen a fizikai és virtuális memória használatát. Keressük meg a hasonló adatok megjelenítésére szolgáló programot valamely Windows rendszeren!

17. Fájlkezelés

17.1. definíció: A fájl (állomány) a számítógép háttértárolóin lévő összetartozó kódok tárolási egysége, ami azonosítóval is rendelkezik.

A fájl a számítógéppel történő adatkezelés alapvető egysége. Adatainkat, programjainkat – megfelelően kódolva – a háttértáron, fájlokba rendezve tároljuk. Egyik fájlban mondjuk egy digitálisan kódolt fényképet, a másikban egy regény szövegét, a harmadikban pedig számítási adatokat stb. találunk. A fájlok használatára az operációs rendszer a lehetőségek széles skáláját nyújtja. Ezáltal óriási terhet levesz a felhasználó válláról: nem kell ugyanis törődnünk azzal, hogy az összetartozó adatok egyben maradjanak, nem kell foglalkoznunk a kódolási, tömörítési módszerekkel, legfeljebb azt kell megadnunk, hogy milyen kódolást, tömörítést alkalmazzon az operációs rendszer. Nem kell törödnünk azzal sem, hogy a fájlunkat az adathordozón hova tegyük, hol van szabad hely a fájl elhelyezésére, és ezernyi más technikai részletkérdéssel sem.

Hogy ez megvalósulhasson, olyan rendszerbe kell foglalni a fájlokat, amelyben könnyen, kényelmesen el lehet igazodni. Ezt a rendszert hívjuk fájlrendszernek. Sokféle fájlrendszer létezik, mivel a különféle szoftver- és hardvergyártok a szabványos megoldások hiányában szinte mindannyian megalkották a saját rendszerüket. Ha egy adathordozót használatba veszünk, meg kell formáznunk. Ez a formázás alakítja ki a tároló fájlrendszerének kereteit, és ettől a pillanatól kezdve a fájlok tárolása a kialakított rendszer szerint történik. A felhasználói igények miatt sok fájlendszer kezelői módszere megegyezik, így a felhasználó fájljaival való műveletvégzéshez nem szükséges ismerni, hogy éppen melyik rendszer szerint alakult ki annak az adathordozónak tartalma, amelyiken a fájljai vannak. Tudni kell viszont azt, hogy az operációs rendszerünk képes-e kezelni az adathordozón lévő fájlrendszert. Ha az operációs rendszer nem ismeri fel az adathordozó fájlrendszerét, akkor a tartalmát nem tudjuk ebben az oprációs rendszerben felhasználni.

A fájlokkal végezhető műveleteket a fájlrendszertől független egységes logikai rendszer tartalmazza. Ennek a rendszernek a használata a fájlkezelés. A leggyakoribb fájlkezelési műveletek: törlés, átnevezés, másolás, tömörítés stb.

Egy fájl tartalma tetszőleges: lehet valamely klasszikus értelemben vett adat kódja, vagy lehet programkód,

17.1. ábra. *Néhány jellemző adathordozó, különböző fájlrendszerekkel*

amely a processzor által végrehajtható utasításokat tartalmazza. Az adatok formájára nézve sincs előírás, nagyon sokféle formátum létezik speciálisan videók, zenék, képek, írásos anyagok, stb, tárolására.

A fájlok olyan adathordozókon helyezkednek el, amelyek tartósan megőrzik a tartalmukat. Ezek némelyike a kezelő egységről, az úgynevezett meghajtóról levehető, és esetleg másik gépre csatlakozó meghajtóra is feltehető, ezzel egyik lehetőségét adva annak, hogy egy számítógépen készült fájljainkat másik számítógépen is használhassuk.

Aktivitás: Gondolkozzunk el azon, hogyan tudnánk megszámolni, hány fájl van a számítógépünkön.

Egy átlagos célra használt, otthoni számítógépen százezres vagy milliós (!) nagyságrendű fájl található. Ennyi fájlt csak megfelelően csoportosítva lehetséges kezelni, ezért a fájlrendszerek teljesen általános, „kötelező” tulajdonsága a *mapparendszer* támogatása. Ez azt jelenti, hogy a fájlok csoportosíthatók (ezeket hívjuk mappáknak), és a mappákban újabb almappák hozhatók létre. A mappa mellett gyakori még a *könyvtár* megnevezés is, de ezt inkább kerüljük, hiszen annak többféle másik jelentése is ismert.

Természetesen az adattárolás konkrét megvalósításából adódóan minden fájlrendszernek vannak korlátai, a fájlok és mappák maximális számára, a mappák maximális egymásba ágyazhatóságára, a fájlok és mappák maximális névhosszúságára vagy a fájlok illetve a teljes fájlrendszer maximális méretére vonatkozóan. Régebbi

tervezésű fájlrendszereknél ezek ma már komoly korlátokat jelenthetnek, azonban a mindenkori modern fájlrendszereket úgy tervezik, hogy a felhasználó ezen a korlátokkal egyáltalán ne vagy csak szélsőséges esetben találkozzon.

A fájlrendszert egy adathordozó használatba vételekor kell kiválasztani, és az operációs rendszer által biztosított segédprogram az adathordozóra írja azokat a segédinformációkat, amik az adott fájlrendszer működéséhez szükségesek. Ezt a folyamatot nevezzük formázásnak. A formázás ritkán végrehajtott művelet, mialatt a háttértár tartalma jellemzően elvész. Új adathordozó beüzemelése után már csak meghibásodások utáni újra beüzemeléskor, vagy a rendszeren történő komolyabb átalakítások alkalmával van rá szükség.

Az operációs rendszer alapfeladatai közé tartozik, így messzemenően támogatja a különböző, fájlokkal kapcsolatos műveleteket:

- a fájlok egyedi elnevezését,
- a sok fájl strukturált tárolását (mappákba szervezését),
- a fájlok visszakereshetőségét, olvashatóságát,
- a fájlok létrehozását,
- a fájlok megkeresését,
- a fájlok másolását,
- a fájlok mozgatását,
- a fájlok módosítását,
- a tartalom megtekintését (esetleg kód szinten is),
- a fájlok törlését, kukába helyezését,
- a fájlok jogosultságainak kezelését (meghatározhatjuk ugyanis, hogy ki végezhet olvasási, írási, törlési, futtatási műveletet a fájlokon),
- a fájltípusok meghatározását, kezelő programokhoz kapcsolását (például beállíthatjuk, hogy a fájlkezelőben

egy zenei fájlra kattintva azt melyik zenelejátszó program elkezdje lejátszani),

- az adathordozókon a megfelelő fájlrendszerek létrehozását.

Aktivitás: Bizonyos esetekben szükséges a kód rövidítése, például helytakarékosság céljából. Ilyenkor a fájl eredeti kódolását változtatjuk meg. A témával részletesen foglalkoztunk a 3. fejezetben. Keressen a gépén egy .bmp képfájlt és tömörítse ZIP eljárással!

17.1. Ismert fájlrendszerek

A következőkben a sok-sok fájlrendszer közül három jellemző típus legfőbb tulajdonságait tekintjük át: egy egyszerű, de a mai apró digitális eszközeink (MP3 lejátszó, fényképezőgép, telefon) által legszélesebb körben használt rendszert, majd a Windows és Linux egy-egy képviselőjét.

- FAT, FAT32:
  Az egyik legrégebbi, leginkább elterjedt, kompatibilitási célokat is teljesítő fájlrendszer. Eredetileg a 80-as években, a DOS operációs rendszerhez készült, a kornak megfelelő színvonalon. Nevét (FAT – File Allocation Table – fájl-helykiosztási tábla) az adathordozóra elhelyezett a táblázatról kapta, amiben nyilvántartja, hogy mely fájlok hol találhatók. A fájlok azonosítására maximálisan nyolc karakterből álló nevet és legfeljebb három karakteres kiterjesztést használt. A kiterjesztés gyakran utalt arra, hogy a fájl milyen módszerrel van kódolva. A korábbi változatuk 12, illetve 16 bites kóddal tartalmazta a fájlok helyének leírását, így viszonylag kisméretű adattároló kezelése volt ezekkel a változatokkal lehetséges.
  A korábbi (kb. az évezredforduló előtti) Microsoft Windows operációs rendszerek a FAT és FAT32 fájlrendszereket használták. Manapság már semmilyen operációs rendszer nem részesíti ezeket előnyben (de mind képes használni), azonban a hordozható digitális eszközökön a mai napig elterjedt: ennek oka talán leginkább a rendszer elterjedtsége (ezáltal kompatibilitása). A mai hordozható merevlemezek döntő többségét FAT32-re formázva szállítják, mint ahogy ez a fájlrendszer kerül a pendrive-okra, MP3 lejátszókra, a fényképezőgépek és okostelefonok flash háttértáraira.

A Windows rendszerek az egyes FAT eszközöket (és ez más fájlrendszerekre is igaz) a sokak által jól ismert meghajtó betűjelek (A:, C:, stb.) segítségével különböztetik meg.

- NTFS
  Az NTFS (New Technology File System, azaz új technológiájú fájlrendszer) a FAT rendszerek leváltására készült. Alapvetően abban különbözik elődjeitől, hogy a fájlok elhelyezkedésének nyilvántartására nem táblázatot, hanem bonyolultabb struktúrát használ. Ezen kívül a fájl kódjai mellett úgynevezett metaadatokat is tárol a fájról. Mivel a rendszert szabadalom védi, részletes felépítését nem ismerjük.
  Elvileg van lehetőség arra, hogy egy mappában egy másik NTFS fájlrendszer tartalmát jelenítsük meg és használjuk, ez azonban a Windows hagyományok alapján nem terjedt el.
  Az NTFS maximális mérete 2 TiB lehet, de az egybefűzhető, ún. *dinamikus kötetek* segítségével ez 16 TiB-re növelhető. A maximális fájlméretet más nem is korlátozza. Ez a méret már kielégíti az általános igényeket. A jogosultságokat egy ACL (Access Control List – hozzáférési lista) segítségével ellenőrzi, lehetővé teszi a fájlrendszer szinten történő (automatikus) fájltömörítést (LZ77-ZIP) és a kvóta rendszer használatát. Ez utóbbi azt jelenti, hogy nyilvántartható az egyes felhasználók helyfoglalása, és beállítható, hogy bizonyos tárhelyméretet senki ne léphessen túl. Ugyanahhoz fájlhoz és mappához a Windows Vista óta több fájlnév, mappanév is rendelhető.

- EXT4
  Az EXT4 (EXTended filesystem version 4 – kiterjesztett fájlrendszer, 4. verzió) egy modern fájlrendszer. A Linuxhoz fejlesztették ki még a 90-es évek elején, azóta pedig a legjobbak között tartják számon. A maximális rendszerméret: 1 EiB (exbibyte, 1 EB = 1048576 TiB), vagyis a mai otthoni gépek TB-méretű háttértárainak az egymilliószorosa is kezelhető lenne ebben a rendszerben. A 16 TiB maximális fájlméret szintén megfelelő a legtöbb feladathoz. Egy mappába 64000 almappa kerülhet. Bevezeti az *extent (kb. terjedelem)* fogalmát a korábbi blokkos megközelítés helyett. Ennek megértéséhez meg kell ismerkednünk az ún. töredezettség problémájával. Ha a fájlok használatakor változtatjuk azok méretét, illetve némelyiket töröljük, akkor elkerülhetetlenné válik, hogy a fájlok egyre több esetben ne folytonosan legyenek tárolva. Ez a fájlok olvasási és írási sebességét is hátrányosan befolyásolhatja. Az extent egy, az adathordozón lefoglalt fizikai terület, amit a rendszer az adott fájl későbbi bővüléséhez tart fenn. Ha az alkalmazott

előfoglaló algoritmus jól becsüli az egyes fájlok jövőbeli méretváltozásait, az a fájlrendszer teljesítményét nagymértékben javíthatja. Linux alatt a töredezettségmentesítéssel gyakorlatilag nem kell foglalkoznia a felhasználónak, mert az EXT4-ben alkalmazott módszer nagyon alacsony szinten képes tartani azt.
Visszafelé kompatibilis a korábbi (ext3, ext2) verziókkal, ami azért érdekes és hasznos, mert például az új helyfoglaló (extent) algoritmusa így használható a régi rendszereken is (ha például ext3 fájlrendszert ext4-ként kívánunk használni).
A fájl-hozzáférések időpontjainak tárolására nanosecundum (egymilliárdomod másodperc) pontosságú időbélyegeket használ.
A Multiblock allocator (több blokkos helyfoglaló) nem egyesével keresi az új blokkokat fájl írásánál, hanem figyelembe veszi, hogy hány blokkra lesz szükség, ezért sokkal „jobb” helyet képes találni – ennek eredménye a kisebb töredezettség. Ezzel összefüggésben a pre-allocation (előre foglalás) lehetővé teszi, hogy egy fájl létrehozáskor ne nulla hosszal, hanem a szükséges terület lefoglalásával jöjjön létre – szintén kiküszöbölve a későbbi töredezettséget.
A késleltetett elhelyezés (Delayed Allocation) a fizikai blokk elhelyezést addig visszatartja, amíg csak lehetséges. Azaz csak akkor keres helyet a letárolandó adatnak, amikor az ténylegesen fizikailag tárolásra kerül az adathordozón: a gyakran írt fájloknál ez komoly teljesítmény-növekedést eredményezhet és a végül letárolt fájl kevésbé lesz töredezett.
Itt mondjuk el, de minden más Linux fájlrendszerre is igaz, hogy a különböző eszközöket egységesen kezeli egyetlen mappaszerkezetben. Ez azt jelenti, hogy egyetlen mappastruktúrában található minden eszköz mappaszerkezete. Például, egy CD tartalma a /cdrom mappában jelenik meg és a többi fájlhoz hasonlóan kezelhető. Akár azt is beállíthatjuk, hogy egy hálózaton elérhető másik számítógép valamely mappájának tartalma látható legyen a gépünk egy mappájában.
A gyökérmappa jele a /, figyeljük meg, Windows alatt épp a fordított törtvonal, a \ használatos a mappák elválasztására, illetve a gyökér jelzésére.

Aktivitás: Gyűjtsük össze a fájlkezeléssel kapcsolatos tapasztalatainkat az általunk eddig használt operációs rendszerekről! Milyen hasonlóságok és különbségek vannak, amiket felhasználóként is észrevehettünk?

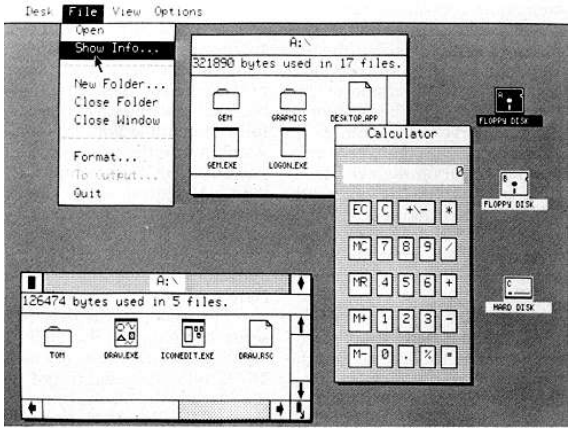
18. Ismert operációs rendszerek

Az informatika néhány évtizedes történetében is több tucatnyi operációs rendszer készült, egy részük a mai napig létezik, más részük pedig már csak a régmúlt emléke. Néhány operációs rendszer az elterjedtsége és közismertsége miatt megérdemli, hogy röviden áttekintsük a történetét és legfontosabb ismérveit.

18.1. Microsoft Windows

A Microsoft Windows nevű operációs rendszere olyan általános ismertségnek örvend, hogy sokan azt hiszik, Windows nélkül a gépük talán nem is lenne működőképes. Az 1980-as években járunk, amikor a Microsoft jó üzleti érzékkel felvásárolta, és fejlesztette a szintén közismert DOS operációs rendszer különböző változatait. Bár a DOS-t később egészen 2000-ig fejlesztették, a Microsoft alapító tulajdonosa, Bill Gates már ezidőben felismerte, hogy a széles körben való elterjesztéshez egyszerű, kényelmes grafikus felületre van szükség. 1983-ban jelentette be egy grafikus felhasználói felületű, többfeladatos rendszer létrehozását. A bejelentett programot Interface Managernek nevezte. Ezt a nevet még a megjelenés előtt Windows-ra változtatták.. Akár a DOS-t, a Windowst is eredetileg kizárólag az IBM számára készítette a Microsoft, a két cég között valódi közös munkáról azonban ritkán esett szó a folyamatos rivalizálás miatt. A Microsoft új operációs rendszerét megelőzve, 1985-ben az IBM megjelentette saját Top View nevű, a DOS-hoz képest sokkal fejlettebb operációs rendszerét, aminek egy komoly hibája volt – grafikus felhasználói felületet még nem tartalmazott. A Top View sosem vált sikeressé, rövid időn belül fel is hagytak a fejlesztésével.

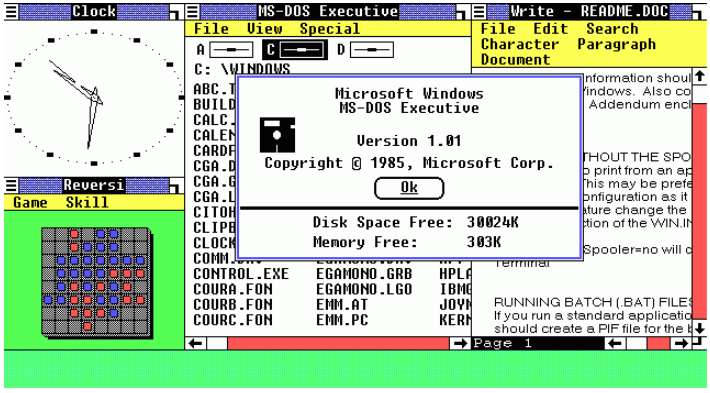
Itt jegyezzük meg, hogy akkoriban is komoly verseny volt már a szoftverpiacokon, sok más cég is nagy erőkkel dolgozott a fejlesztéseken – voltak, akik technológiában messze a Microsoft előtt jártak, de nem volt olyan marketingjük és alakuló felhasználói bázisuk, mint a DOS kapcsán a Microsoftnak: hiába jó egy operációs rendszer, ha kevesen használják, nem írnak rá programokat, ha nem írnak rá sok programot, kevesen fogják használni (lásd a 12. fejezetben a fejlesztőeszközökről szóló részt) – ebből az ördögi körből keveseknek sikerült kilépni.



18.1. ábra. *GEM (forrás: atarimagazines.com)*

A Microsoft új operációs rendszere pedig egyre késett, közben például a Digital Research GEM (Graphics Environment Manager, grafikus környezetkezelő) megvalósította a Microsoft két évvel korábban tett ígéreteit (18.1. ábra).

Végül 1985. november 20-án megjelent a Windows első változata (18.2. ábra, érdemes összehasonlítani pl. a korábban megjelent GEM rendszerrel a 18.1. ábrán). A program lassú volt az akkori hardvereken, és a felhasználók visszajelzései alapján tele volt hibákkal. A hasonló szoftvereket fejlesztő versenytársak ellopott szabadalmak, ipari kémkedés miatt folyamatosan perelték a céget. A nyilvánvalóan igaz állításokat a vállalat nem különösebben tagadta, inkább próbált mindenkivel megegyezni. Szép lassan sok fejlesztést, licenszet kifizetett, „összevásárolt”, ezek későbbi értékéhez képest jelentéktelen árakon, amivel megalapozta a cég elképesztő anyagi sikerét és sok versenytárs végleges kiszorítását.



18.2. ábra. *A Windows egyik legkorábbi verziója*

Aktivitás: Miközben a szoftveres fejlődésről olvasunk, érdekes rácsodálkozni, milyen fejlődésen mentek keresztül a hardverek. Olvassa le a a 18.2. ábráról, mekkora háttértárral rendelkezett a Windowst futtató számítógép!

A sikertelenségek, a nyilvánvaló hibák ellenére – az egyébként viszonylag olcsón árult Windows – egyre jobban terjedni kezdett: ennek különösen nagy lökést adtak egyes komoly szoftverek Windowsra írt változatai. Ilyenek voltak mint például az Aldus Pagemaker kiadványszerkesztő, a Corel Draw rajzoló vagy a Microsoft által „házon belül” készített Word szövegszerkesztő és Excel táblázatkezelő programok.

A Windows 1987-ben megjelenő 2.0 változata szabadalmi viták miatt újabb, hosszú évekig tartó pereskedési hullámot indított, amiből végül a Microsoft nyertesként került ki. Közben 1990-ben megjelent a 3.0, majd 1992-ben a 3.1 változat, ami talán a legismertebb volt Magyarországon is, hiszen ez volt az első, magyar nyelvű verzió. A Microsoft itt vezette be a virtuális memória és a mai napig is közismert True type betűtípusok használatát. Ezek a rendszerek is sok bosszantó hibát tartalmaztak, de az egyre növekvő számú alkalmazás és a Microsoft kétségtelenül ügyes piaci stratégiája miatt milliószámra adtak el belőle. Egyre inkább nyilvánvaló lett,

hogy minél hamarabb piacra tudják dobni az újabb, továbbfejlesztett rendszereket, az annál hamarabb fogja a hasznot is termelni – különösen, ha közben a konkurenciát is sikerül megelőzni. 1995 nyarán nagy kapkodásban megjelent a Windows 95. A Microsoft marketingjére jellemző, hogy voltak, akik tévedésből úgy vásárolták meg a programot dobozban, hogy nem is volt számítógépük. A rendszer megint csak tele volt hibákkal – emlékezetes a magyar nyelven is megjelent *Windows 95 hibák* című könyv, ami kizárólag e verzió hibáival és lehetséges orvoslásukkal, megoldási javaslatokkal foglalkozott... A felhasználók mégis használták, hiszen szerencsés esetben kényelmes volt a használata: a sok beépített eszközmeghajtó automatikusan használatban tudta venni (plug-and-play) az akkori kurrens hardver eszközöket. Korábban például egy egér használatához a floppyn mellékelt egérkezelő programot is telepíteni kellett. A Windows 95 az ilyen munkát számos esetben feleslegessé tette, mert sok eszközt felismert.

A korábbi Windows-ok a működő DOS operációs rendszeren elindított grafikus felhasználói felületek voltak. A Windows 95 volt az első olyan Windows változat, amit már DOS nélkül, „üres” gépre is lehetett telepíteni – a felhasználók szempontjából ez fontos különbség volt, bár a Microsoft nagyon hamar „lebukott”: a hozzáértők előtt nem maradhatott rejtve, hogy valójában ugyanúgy egy beleintegrált DOS fut a gépen, csak épp betöltődés után nem a karakteres felületen villogó promptot adja, hanem betölti a grafikus felületet is.

1998-ban jelent meg a Windows 98 változat, ami az első olyan Windows operációs rendszernek számít, ami már nem a DOS-ra épül, majd 2000-ben a nem túl sikeres Windows ME (Millenium Edition). 2001-ben már piacra dobták a Windows XP változatot, ami már olyan hardvereket támogat és olyan szoftveres technikákkal dolgozik, hogy a mai napig is használatos (a természetesen mindig újat eladni szándékozó Microsoft legnagyobb bánatára és minden igyekezete ellenére...)

Itt említsük meg, hogy az 1990-es években a Microsoft több termékvonalat kezdett el párhuzamosan fejleszteni, a különböző felhasználói igényeknek megfelelően. Az említett Windows 95 termékcsaládot inkább az otthoni felhasználóknak szánták, más szóval elsősorban játékra tervezték. A vállalati felhasználókat egy már évek óta fejlesztett másik termékvonal, a Windows NT (New Technology) célozta meg, 1993-ban a 3.1 verzióval indították útjára (talán azért épp ezzel a számmal, mert az akkor árusított Windows a 3.1 verziónál járt). A különbség jelentős volt, az NT már 32 bites és sokkal megbízhatóbb volt, amit egyrészt az alapok újragondolása, másrészt a multimédiás képességek kihagyása biztosított. Az NT kétféle változatban került forgalomba, a

munkaállomásnak (asztali gépnek) szánt Workstation és a kiszolgálókra tervezett Server. Későbbi jelentős verziók voltak még az 1994-ben megjelent 3.5 és az 1996-os 4.0. Ezen vonal folytatása volt a Windows 2000, majd a Microsoft lassan felhagyott azzal, hogy otthonra és munkára különböző operációs rendszert tervezzen: a Windows XP-ben újra egyesült a két ág, és a különböző felhasználói igényeket azóta inkább ugyanazon a rendszeren belül hangolják (Home és Professional változatok, bár inkább üzleti, mint technológiai szempontból fontos a megkülönböztetés.) Ezután jelent meg 2006-ban a nem túl sikeres Vista, 2009-ben a sokkal jobban sikerült Windows 7, és 2012-ben megjelent a Windows 8.

A Windows 7 megjelenésekor a Microsoft a kényelmes felhasználói felületre, a gyorsindítás eszköztárra, a biztonságra, az otthoni hálózati funkciókra helyezte a hangsúlyt. Maximum 2 processzort támogat, a 32 bites verzió maximum 4 GB, a 64 bites pedig 192 GB memóriát képes kezelni. Az alkalmazások a virtuális memóriát látják, aminek felső része csak a rendszeré, az alsó a felhasználói programoké. A felső részben helyezkedik el többek között az operációs rendszer kernele, és az operációs rendszer és a felhasználói aktivitás pillanatnyi állapotát leíró ún. session space. A Windows régi hibaforrása volt, hogy a meghajtóprogramokat az operációs rendszer részeiként kezelte, így azok rendszerszintű hibákat okozhattak – a *kék halál* fogalom mindenki számára ismert, aki használt régebben Windowst. A Vista óta egyre több meghajtóprogram normál felhasználói programként működik, ami gyakorlatilag megszüntette a rendszerösszeomlások jelentős részét. A fizikai memóriában lefoglalt memóriaterületek vannak a különböző eszközök számára (Memory Mapped I/O, PCI hole). Ez az oka annak, hogy az elvi 4 GB helyett egy 32-bites Windows kb. 3 GB-ot tud kihasználni, több nemigen látszik belőle. A 64 bites változat ügyesebb szervezéssel kevesebb veszteséggel dolgozik (Memory remapping).

Eddig csak a Microsoft személyi számítógépekre tervezett operációs rendszereivel foglalkoztunk. A teljesség kedvéért említsük meg, hogy a Windowsnak van kiszolgáló gépekre és a mobil eszközökre tervezett termékvonala is. Ez utóbbi kategória ismert képviselői a CE (Embedded Compact; magyarul: beágyazott tömörítvény, a mai napig sok GPS használja) és Windows Mobile termékek (18.3. ábra).

Aktivitás: Milyen operációs rendszer, annak pontosan melyik verziója fut a saját számítógépén? Sorakoztasson fel érveket, milyen szempontok alapján választotta ki ezt a sokféle elérhető rendszer közül!

18.3. ábra. *Windows Mobile-t futtató Asus telefon*

18.2. Linux

A Linux megismeréséhez ugorjunk vissza az 1960-as évekbe, a Unix operációs rendszer születéséhez. Egy több kutatóintézet (MIT, Bell, General Electric) által vezetett nagyszabású projektben igyekeztek egy modern operációs rendszert előállítani. A projekt nagyon rosszul haladt, végül különösebb sikerek nélkül abbamaradt. Ken Thompson, az egyik résztvevő nekiállt egymaga folytatni a munkát, komoly egyszerűsítésekkel, hogy esélye legyen a sikerre. Az operációs rendszer fejlesztéséhez, szinte mellékesen kifejlesztett egy B nevű programozási nyelvet, amiben később a tényleges fejlesztőmunkát végezte. A kezdeti sikereken felbuzdulva Dennis M. Ritchie vezetésével néhány újabb fejlesztőt sikerült bevenni a csapatba. Ritchie vezetésével a B nyelvet tökéletesítve, kibővítve megalkották a máig is használt C nyelvet, és ezen a nyelven újraírták, tovább fejlesztették Thompson rendszerét – amit ekkor már Unixnak hívtak. Már az 1970-es években járunk, és a Unix sokkal jobban működik, mint az akkori ismert operációs rendszerek. A forráskód ráadásul szabadon használható volt bárki számára, így kutatóintézetek, egyetemek százain tértek át rá szinte azonnal, és a forráskód birtokában ki-ki fejlesztette, javította, bővítette. Kereskedelmi cégek is fantáziát láttak a Unixban: sok cég alakította át a maga igényei és elképzelései szerint, és árusítani kezdték a saját Unix változatukat: az IBM *AIX*, a Sun *Sun OS, később Solaris*, a Hewlett Packard *HP-UX*, a Silicon Graphics *IRIX* néven. Az első igazán jól működő kereskedelmi Unix változat

az AT&T nevéhez kötődik, *System V* néven. Az egyetemek közül kiemelkedik a University of California at Berkeley munkája: ők különösen sokat tettek hozzá a fejlesztéshez, BSD (Berkeley Software Distribution) néven a mai napig elérhető az általuk fejlesztett és terjesztett rendszer. A részletek megismerése nélkül jegyezzük meg, hogy az egyik fő Unix-fejlesztő csoportban felmerült a számítógépek összekapcsolásának ötlete amiatt, mert egy többszintes épületben a „nagy" számítógéptől több emelet távolságban dolgoztak a programozók, és fárasztó volt a mágnesszalagokat fel-le szállítani. A következő években – bár a sok különböző helyen folytatott, sokszor üzleti érdekek alapján egymás ellen történő fejlesztés bizonyára nem segítette az egységes Unix rendszer terjedését – a Unix (változatokban) mégis töretlenül fejlődött, és érdekes módon a lefektetett szabványok és kváziszabványok miatt nem túlságosan térnek el egymástól, a különbségekkel folyamatosan együtt kellett élni.

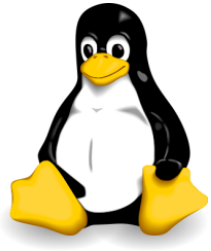
A GNU projekt 1983-ban Richard M. Stallman vezetésével indult azzal a céllal, hogy a Unix-ok logikájával működő, bevált, de szabadon használható operációs rendszert fejlesszenek ki. A GNU a *GNU is Not Unix* rekurzív szójátékból származik, ami arra utal, hogy Unix megoldásokon alapuló, de a kereskedelmi Unix-okból semmit át nem vevő rendszerről van szó. A GNU a mai napig komoly fejlesztés alatt áll, maga az operációs rendszer szűken értelmezve nem működik (például nincs önálló GNU kernel), de hasznos szoftverek, segédprogramok százait adta a projekt a világnak – ezek jelentős része a ma használatos Unix és Linux rendszereknek alapvető részeit jelenti.

Az 1990-es évek elején Linus Torvalds finn egyetemista az Intel 386-os processzorral szerelt gépére írt egy Unix szabványnak megfelelő (POSIX) rendszermagot. (Erről az első nyilvános értesítés 1991. augusztus 25-én írt emailjében található, ezért azóta ezt a napot tartják a nevével fémjelzett rendszer, a Linux születésnapjának.) A rendszermag forráskódját az interneten elérhetővé tette, és akkor még nem sejtette, hogy a GNU projektben elkezdik azt használni: a meglévő GNU elemek az elkészült kezdetleges rendszermagot jól kiegészítették. A rendszer teljesen ingyen, bárki számára elérhető volt, és látványos fejlődésnek indult. A rendszer neve Linus Torvalds keresztnevéből adódott, a Unix-ra jellemző *x* betűvel végződő elnevezések alapján kezdték Linuxnak nevezni. A GNU-hoz való kötődése miatt a Linuxot sokszor GNU/Linux néven említik, és mivel a fejlesztésnek itt is több fő csapásiránya alakult ki, az ún. *disztribúció* nevét a Linux név elé teszik, például: Slackware Linux, RedHat Linux, Ubuntu Linux, Suse Linux. A disztribúciók, mint fejlesztési irányok leginkább úgy képzelhetők el, hogy valamely szervezet, ha saját Linux disztribúció fejlesztésbe kezd, akkor saját maga dönt a használt

felhasználói felületről és a sokezernyi segédprogram összeállításáról. A disztribúcióhoz készíteni kell egy telepítő programot, mintha egy „különálló” operációs rendszer lenne: az is persze, de a disztribúciók mélyén sokszor ugyanazok a szoftverek dolgoznak, és ezek természetesen szabadon változtathatók, így maga a Linux disztribúció kiválasztása inkább személyes szimpátián múlik, illetve egyik disztribúció például otthoni személyes használatra, a másik kiszolgáló gépre lett jobban felkészítve.

Napjainkban a szuperszámítógépek több, mint 90%-a, a 10 legerősebb számítógép mindegyike, az okostelefonok több, mint fele, a webszerverek 70%-a Linux operációs rendszerrel dolgozik. A linuxos személyi számítógépek (asztali, notebook, netbook, ultrabook) és például a tabletek számát nagyon nehéz becsülni, mint ahogy azt is, hogy hány DSL modemben, routerben, DVD lejátszóban, GPS-ben, háztartási gépben, például mosógépben fut Linux. Valójában mindegy is, elég annyit tudni, hogy a Linux napjaink egyik legelterjedtebb operációs rendszere. Az előzőek alapján elmondhatjuk, hogy a kedves olvasó is sokszor használt már Linuxot – még ha nem is vette észre. A kiszolgálókon és az eszközökbe ágyazott rendszereknél ugyanis a felhasználó nem tudja, mit használ – a mögöttes rendszer pedig csendben, hibamentesen végzi a dolgát.

A Linux megörökölte a különböző Unix-ok erényeit: biztonságos, stabil, szabad (ingyenesen használható és módosítható) rendszer. Kiváló támogatás (support) segíti a használatot, legyen szó akár fizetős kereskedelmi támogatásról, vagy akár az ingyenes, online, mindennél gyorsabb és hatékonyabb fórumokról és mindenre kiterjedő dokumentációról. A felhasználóbarát felületeket nagyon kényelmes használni, a tengernyi, jellemzően ingyenes szoftverek között kényelmesen keresgélhet a felhasználó, a neki tetsző programot pedig akár egyetlen egérkattintással telepítheti. (Az alkalmazások telepítésének módja a mai Linuxok egyik legkiemelkedőbb tulajdonsága a személyi számítógépeken elterjedt Windows-okkal szemben: a telepítés egy kattintásból vagy egyetlen parancs begépeléséből áll – magával a program „megszerzésével” sem kell törődni, a kiválasztott szoftver és az esetlegesen annak futásához szükséges egyéb szoftverek az internetről automatikusan letöltődnek. A szoftverek ezután azonnal használatba vehetők (bármely felhasználó által) és az egyéni beállítások sosem keverednek a rendszerszintű beállításokkal. A szükségtelen szoftverek bármikor, a telepítéshez hasonlóan pillanatok alatt eltávolíthatók – ilyenkor a saját beállításainkat megőrizhetjük, de ha akarjuk, egyetlen fájl sem marad a törölt program után.) A rendszer frissítése automatikus, és nem csak a szűken vett operációs rendszer frissül, hanem az összes telepített alkalmazás is!



18.4. ábra. *Tux, a Linux-pingvin*

A Linux kabalaállata Tux, a pingvin, ami a 18.4. ábrán látható. A név a Torvalds UniX betűiből származik.

18.3. Android

Az előzőekben tárgyalt operációs rendszerek személyi számítógépekre, illetve kiszolgálókra készültek, most harmadik példaként megnézünk egy merőben más hardveren futó operációs rendszert is.

A mobiltelefónia fejlődésével, az egyre komolyabb teljesítményű eszközök megjelenésével felmerült, hogy a mobil eszközök a gyártóik egyedi fejlesztései helyett valami egységes platformot, mondjuk így, operációs rendszert kapjanak. Egy régi telefon testre szabhatósága kimerült két-háromféle háttérstílus kiválasztásában és egyedi csengőhangok csereberéjében, az újabb mobil eszközök azonban egyre inkább hasonlítanak egy kompakt, mini számítógépre, ami az eredeti funkció, a telefonálás mellett ezernyi más dologra is alkalmas. Ehhez azonban megfelelő szoftverekre van szükség. Figyelembe kell venni a mobil eszközök eltérő képességeit: az asztali eszközökkel, vagy akár noteszgépekkel ellentétben ezeket a mobil eszközöket kis méretű érintőképernyő, gyenge processzor, kevés memória, kevés tárhely, a billentyűzet és egér hiánya jellemzik, és mindenképpen a tartós vezeték nélküli használatra kell felkészülni.

A több fejlesztési irány közül mi ismerkedjünk meg napjaink egyik legnépszerűbb, előremutató mobil operációs rendszerével, az Androiddal. Fejlesztését egy azonos nevű cég kezdte, amit a Google 2005-ben felvásárolt és

nagy erőkkel továbbfejlesztett.

Az Android egységes, nyílt forráskódú operációs rendszer, Linux alapokon (rendszermagon) fut. A Linux kernel tartalmazza a hardver által kezelendő eszközök meghajtó programjait. Ezeket a speciális eszközmeghajtókat a gyártók írják, a saját hardverükhöz igazítva. Ide tartozik a képernyő, a kamera, a mikrofon, a hangszóró, a WLAN (Wifi), az A-GPS, a gyorsulásmérő, a Bluetooth, az iránytű, a gyorsulásmérő, a flash memória és minden, itt nem felsorolt egyéb eszköz működéséhez szükséges meghajtóprogram. A Linux rendszer az eszközök alapvető kezelését végzi, az operációs rendszer funkciójához híven itt történik a folyamatok ütemezése, és például a szűkös akkumulátor-kapacitás miatt oly fontos energiagazdálkodás is.

A felhasználói felület kezelése és a tényleges alkalmazói szoftverek futtatása átkerült egy virtuális gépre (a virtuális gépekről a következő fejezetben olvashatunk): a Linux alapú Android rendszer egy Java virtuális gépet futtat, a felhasználó már az „igazi” operációs rendszert nem is érzékeli, hanem egy kényelmes felhasználói felületet lát, és a rendszerre írt programok is már mindig ezen a virtuális gépen futnak. A megoldás legfőbb előnye jól megmutatkozik az Android elterjedtségi adatain: a különböző hardverek ellenére a megírt programok más gyártó hardvereszközein is futhatnak (hiszen a virtuális gép azonos lehet, alatta a tényleges, különböző eszközök kezelését pedig a Linux rendszer végzi)

A valódi hardverek eltüntetésével, helyette egy kényelmes programozói interfész biztosításával megnyílt az út a mobil eszközök minél teljesebb kihasználása és a tengernyi alkalmazás elkészítése felé.

Az Android elválaszthatatlan részét képező Java virtuális gép (neve: Dalvik) nem tévesztendő össze a Sun virtuális Java gépével, teljesen más az utasításkészlete, és a generált programkód.

Az Android 2008 őszén jelent meg, és az első évben egyáltalán nem volt sikeres. Ezután azonban folyamatosan jelentek meg az újabb verziók, amik a hardvereket egyre hatékonyabban kihasználva, a felhasználók igényeit egyre jobban eltalálva már 2010-ben elérte az 50%-os részesedést az okostelefonok piacán. A kis fáziskéséssel terjedő táblagépek között is rohamosan terjed, jelenleg közelíti az 50%-ot. Elsősorban e két eszköztípus az Android elsődleges, de nem kizárólagos hardver környezete: a 18.5. ábrán bal oldalon látható Toshiba notebook is Androidot futtat.

18.5. ábra. *Az Android rendszer célhardverei*

19. Virtuális gép koncepció

Az informatikai hardverek és alkalmazások fejlődésével egyre jobban előtérbe kerülnek a különböző virtualizációs technikák. Gondoljunk csak arra, hogy a modern operációs rendszerek egyik funkciója is épp a virtualizáció: a bonyolult, és gépenként különböző hardver közvetlen kezelése helyett egy virtuális gépet használunk, és ha programot készítünk, abban egy *print* paranccsal szeretnénk nyomtatást kezdeményezni, úgy, hogy nem kell foglalkoznunk azzal, hogy a kapható sok ezernyi nyomtató közül a futtató géphez épp milyen típus van csatlakoztatva. Nem csak programozók hallottak már a Java programozási nyelvről. A Java egyik alapötlete, hogy egy egységes Java virtuális gépet biztosít a Java programok számára. Így történhet meg, hogy egy Java program könnyedén futtatható a különböző hardvereken, legyen szó különböző operációs rendszereket futtató személyi számítógépekről vagy különböző gyártmányú mobiltelefonokról.

A virtualizáció többszintű is lehet, azaz egy virtuális környezetben létrehozhastunk újabb virtuális elemeket is. Ilyen eset fordul elő akkor, ha egy operációs rendszerben egy olyan környezetet építünk fel, amely úgy működik, mintha önmaga egy különálló számítógép lenne. Természetesen ehhez a virtuális géphez is tartozik

19.1. ábra. *A virtualbox nevű virtuális gép kezelőfelülete*

operációs rendszer, ami a gép használatát lehetővé teszi. Ennek az operációs rendszernek a felhasználója észre sem veszi, hogy a gépe virtuális, hiszen a rendszer kezelő felülete pontosan úgy működik, mintha valós számítógépen futna. A valós hardver operációs rendszerét gazdának (hostnak), virtuális gépét vendégnek (guestnek) nevezzük.

Megjegyzés: Most arról beszélünk, hogy egy számítógépen több operációs rendszer fusson *egyidőben*. Vegyük észre, ehhez semmi köze nincs annak, ha például az otthoni számítógépünkre több operációs rendszert telepítünk, amik közül a gép indításakor választunk. Azok közül egyszerre mindig csak az egyik futhat, az el nem indítottak nem végeznek semmilyen tevékenységet. A virtuális gépek (akár több is) egyszerre futhatnak ugyanazon a gépen, amelyen a gazda operációs rendszer is működik.

Érdekesség: Az egyetemen is sok virtuális szerver üzemel, a felhasználók természetesen nem veszik észre, hogy ezen gépekhez nem tartozik önálló hardver.

A gazdagépen vendég operációs rendszer futtatásának nagyon sok oka lehet:

- Másik operációs rendszer ideiglenes használata
  Előfordulhat, hogy egy Linuxos notebookon egy Windows program futtatására van szükség. Erre a Linux nem, csak a Windows képes, ezért vendég-Windowst telepítve a program futtathatóvá válik. (Természetesen más megoldás is lehetséges, erre nem térünk ki.)
- Régi operációs rendszer használata
  Például, DOS, C64 GEOS, Windows 3.1 használata lehet szükséges egy valamikor kizárólag azon futó, későbbi rendszereken működésképtelen program futtatásához, vagy abból történő adatmentéshez.
- Hálózat kialakítása és tesztelése
  Egyetlen gépen indított több virtuális gépen egyszerűen és olcsón tesztelhetünk például hálózati funkciókat, vagy a saját programunk hálózati működését. Nagy cégek (pl. Microsoft) a saját rendszergazdai tanfolyamaikon virtuális gépeket biztosítanak a hallgatóknak, amin a különböző feladatokat gyakorolhatják.
- Felhasználói támogatás (helpdesk)
  Nagyon sok cég üzemeltet telefonos ügyfélszolgálatot. Ha betelefonál valaki, hogy nem tudja a hálózatot beállítani az angol nyelvű és kevesek által használt Windows ME rendszerén, akkor nagy segítség lehet a technikai személyzetnek, ha egy virtuális gépen az előretelepített sok-sok rendszer közül gyorsan elindítja ezt a verziót, és akkor pontos programnevekkel, menüpontokkal segítheti a gyakorlatlan felhasználót.
- Pillanatképek, visszaállítás
  A virtuális gépeknek könnyen elmenthetjük különböző állapotait, és ezek között könnyűszerrel válthatunk. Az egyes állapotokhoz rövid magyarázatot rendelve hasznosak lehetnek a következő címkéjű gépállapotok:

  1. Telepítés utáni, de a felhasználói programok nélküli, „üres” gép

2. Irodai alkalmazások telepítve
3. X program telepítése előtti állapot
4. Y vírus tesztelése előtti állapot
5. Y vírus kártétele utáni állapot

Az első két állapot segítségével könnyen „újratelepíthető” egy gép, Windows felhasználók nagyra értékelhetik az így megnyerhető sok-sok munkaóra megspórolását. A 3. pont szerinti állapot akkor lehet hasznos, ha a problémásnak látszó X program valóban komoly zavart okozott a rendszeren és jó lenne a telepítés előtti utolsó állapothoz néhány kattintással visszatérni. A 4. és 5. pont vírusokkal foglalkozó szakemberek munkáját segítheti.

- Infrastruktúra konszolidáció
  A virtuális gépek egyik gyakori használati területe a kiszolgálógépek üzemeltetése. Ahol komolyabban használnak számítógépeket, ott előbb-utóbb akár tucatnyi kisebb-nagyobb szerver is megjelenhet: webszerver, fájlszerver, nyomtatószerver, licensz-szerver, adatbázis-szerver, alkalmazás-szerver, az előzőek biztonsági másolatait őrző tükörszerverek. Ezek a szerverek sokszor különböző operációs rendszert használnak, eltérő hardverigényűek és egyéb (pl. licensz vagy biztonsági) okokból nem könnyű őket összevonni. Így azonban a vállalatnak komoly költség a géppark fenntartása, gondoljunk csak a szükséges géptermi kapacitásra, az áramfelvételre vagy épp a szükséges emberi erőforrásokra. Sok felsorolt szervert helyettesíthetjük egyetlen erősebb gazdagéppel, amin virtuális gépeket futtatva valósítjuk meg az egyes szerverfunkciókat. A megoldásnak nagyon sok előnye van:
  - Hardver, elektromos és fenntartási költségek csökkentése:
    Egyetlen szerver karbantartása, javítása, menedzselése, üzemelési költsége sokkal olcsóbb, mint több kisebbé
  - Egy erősebb szerver sokkal olcsóbb, mint sok kisebbé
  - Az adatmentések és visszaállítások sokkal egyszerűbbek és gyorsabbak
  - A gazdagép hardvere úgy bővíthető, hogy az egyes vendég-gépek működését az nem hátráltatja

  - Hardvercsere után az új gépre kell másolni a virtuális gépeket és azok azonnal újraindíthatók.

- Tesztelés
  Bizonyos programok, különösen saját fejlesztésű alkalmazások telepítésének tesztelésénél nagy segítség lehet, hogy a virtuális gép állapotai menthetők, korábbi állapotra visszaállíthatók. Így például ugyanazt a tesztelési fázist egy gép ugyanolyan állapotáról többször újra lehet kezdeni. Tesztelésnél hasznos lehet, ha egy programot a lehetséges felhasználói kör alapján különböző, például angol, magyar, német, spanyol és orosz nyelvű, 32 és 64 bites rendszereken is rendszeresen kipróbálnak, és mindehhez nem kell tucatnyi valódi számítógépet üzemeltetni.

Nézzük meg, hogyan működik mindez a gyakorlatban. A gazdagépen szükség van egy virtuális gépet megvalósító szoftverre. Ebben először is létre kell hozni a vendég gépet. Be kell állítanunk a hardver legfontosabb paramétereit, például mennyi RAM legyen a gépben, és mekkora háttértárral dolgozhasson. A hálózati kapcsolatait is be kell állítanunk: lehet teljesen elszigetelt is és olyan is, hogy a gazdagép hálózati kapcsolatait lássa. Sokféle tesztelésnél hasznos az a lehetőség, amikor a vendég-gépekből saját elszigetelt hálózatot hozhatunk létre: a vendég gépek egymást, csak egymást fogják látni a hálózaton. Ha ezeken túl vagyunk, be kell állítanunk, honnan tudjuk az új gépünket telepíteni. A szokásos megoldás lehet például egy DVD vagy DVD-képfájl, amit szintén beállíthatunk a virtuális gépünk számára. Ezután az elkészült gépet a gazdagépen el kell indítanunk, és ettől fogva a valós gépekhez hasonlóan kezelhető.

Aktivitás: Hozzon létre egy virtuális gépet a saját (valós) számítógépén és telepítsen rá operációs rendszert. A munkához az Oracle VM Virtualbox nevű szoftvert ajánljuk, amiben egy varázsló segítségével néhány kattintással kialakítható egy virtuális gép. Az operációs rendszer telepítése szakembereknek való feladat, de egyszerű alapesetben bárki által elvégezhető. Virtuális gépen kockázat nélkül tanulhat, kísérletezhet: nem kell adatvesztéstől vagy a számítógépe működésképtelenné válásától tartania.

Önellenőrzés

1. Milyen szempontok alapján választana a hordozható winchesterére fájlrendszert?
2. Mely állítások igazak a következő operációs rendszer családokra?

Linux:

- [ ] Modern, biztonságos és igen rugalmas
- [ ] Általában ingyenesen használható
- [ ] Hálózat kezelésére alkalmatlan
- [ ] Kizárólag Linux számítógépekkel kommunikál

Windows:

- [ ] Minden számítógépen megtalálható
- [ ] Tipikusan szervereken elterjedt
- [ ] Ingyenesen használható
- [ ] Népszerű, elterjedt

3. Melyik állítás igaz az operációs rendszerre?

- [ ] Egy olyan szoftvercsomag, ami elrejti a hardver részleteit, helyette egy virtuális gépet mutat a felhasználó számára
- [ ] Működését elvileg nem lehet megkerülni
- [ ] Minden számítógépen fut
- [ ] Egy olyan szoftverrendszer, ami a hardver és szoftver erőforrásokat kezeli
- [ ] Lehetővé teszi, hogy a hardver erőforrásokat szimbolikus nevekkel kezeljük
- [ ] Magas szintű parancsokat biztosít pl. a fájlkezeléshez

- [ ] Részei: a kernel, eszközmeghajtó programok, segédprogramok.
- [ ] Sok, több tucat operációs rendszer létezik.
- [ ] Együttműködő számítógépek rendszere
- [ ] Az adatok tárolási módja
- [ ] A BIOS új változata
- [ ] Egy olyan programcsomag, ami virtuális hardvert kezel és ezt biztosítja az alkalmazásoknak is
- [ ] Az a programcsomag, amit minden esetben a számítógép gyártója biztosít a géphez
- [ ] Az a program, amit a felhasználó bejelentkezés után mindig automatikusan elindít
- [ ] Kb. tíznél kevesebb operációs rendszer létezik a világon.

8. LECKE

Számítógép-hálózatok 1.

20. Számítógép-hálózatok

A *hálózat* széles körben használt fogalom, a kémfilmektől kezdve a különböző kozmetikumok árusítására szervezkedő marketing-bemutatókig sokfelé találkozhatunk ezzel a kifejezéssel. Mi természetesen kommunikációs hálózatokkal, azon belül is számítógép-hálózatokkal fogunk foglalkozni akkor is, ha az egyszerűség kedvéért egyszerűen a *hálózat* szót használjuk.

20.1. definíció: Számítógép-hálózaton két vagy több egymással összekapcsolt, együttműködő, autonóm számítógép kapcsolatát értjük, ahol az egymással összekapcsolt számítógépek között adatforgalom van.

Másképpen fogalmazva a hálózat egy olyan kapcsolat, (logikus módon minimálisan két számítógép között), ahol adatcsere céljából együttműködnek az egyébként önálló számítógépek. Az önállóság és az adatforgalom a definíció fontos kulcsszavai.

Aktivitás: Magyarázzuk meg, miért nem tekintjük hálózatnak egy számítógép és hozzá kapcsolt két - bár tudjuk, komoly elektronikát tartalmazó - nyomtató kapcsolatát.

A definíció semmit nem mondott a hálózat fizikai megvalósításáról. Az sokféle lehet, és a logikai működés szempontjából lényegtelen.

A hálózatba kapcsolt számítógépeket sokszor csomópontnak (angolul *node* vagy *host*) nevezik.

Nézzük sorba azokat az igényeket, szempontokat, amik életre hívták a hálózatokat.

- Erőforrások összevonása, perifériák közös használata
  Hálózat segítségével összevonhatók és megoszthatók bizonyos erőforrások, például elegendő lehet egyetlen nagyteljesítményű nyomtató néha egy egész adminisztratív rendszerben, nem szükséges minden szobának önálló nyomtatóegység. Ez minden szempontból praktikus és takarékos megoldás.

- Az erőforrások helytől független elérése
  A hálózat használatával az egyes erőforrások és az igénybe vevő csomópontok fizikai távolsága, földrajzi elhelyezkedése alig befolyásolja egyes erőforrások igénybevételi lehetőségét. Például az oktatás adminisztációját támogató Neptun rendszer otthonról éppúgy elérhető, mint az egyetem géptermeiből.
- Fájlok távoli elérése, megosztása
  Megfelelően gyors hálózattal bárhonnan elérhetjük, illetve akár másokkal is megoszthatjuk saját fájljainkat is. Gondoljunk csak a hálózati mappákra, online tárhelyekre, képgyűjteményekre.
- Hálózati kapcsolat megosztása (internet)
  Egy csomópont a saját hálózati kapcsolatát továbbadhatja más csomópontoknak. Ezért van az, hogy otthonra elegendő egyetlen internet előfizetéssel egy hálózati kapcsolódási pontot kérni valamely szolgáltatótól, és ezt megfelelő berendezések segítségével tudja az egész család, gyakorlatilag akárhány számítógéppel és egyéb eszközzel használni. Ilyen berendezés például a *hálózati hub*, ami vezetékes kapcsolati pontra képes több gépet is csatlakoztatni, vagy egy *wlan access point* (vezetéknélküli hozzáférési pont), ami drótnélküli hálózaton képes egyszerre több eszköz kapcsolatát biztosítani.
- Megbízhatóság (redundancia, adatbiztonság) növelése
  Azzal, hogy megfelelő szervezés esetén az adataink redundáns módon, távoli gépeken is tárolódnak, akár több helyen is, az adatbiztonság magas szintre fokozható. Hálózat igénybevétele szinte nélkülözhetetlen a megfelelő adatmentéshez: ha a mentés ugyanazon gép valamely mappájába készül, vagy akár szalagra kerül, amit a gép tetején tartanak, mit sem ér egy meghibásodás vagy tűz esetén. A Világkereskedelmi Központ (World Trade Center, WTC) felrobbantása miatt több cég nem azért ment tönkre, mert nem volt pénze új irodára vagy ne lett volna biztosítása, hanem azért, mert a cég összes adatát fizikailag ugyanott tárolták, így a cég az összes adatát (ügyfelek, rendelések, dokumentációk, stb.) elveszítve szó szerint megsemmisült.
- Gazdaságosság növelés
  Az előzőekből következik, mintegy azok összefoglalása, hogy a hálózat használatának egyik célja a gazdaságosság, a költséghatékonyság növelése.

- Speciális hálózati szolgáltatások megvalósítása
  Végül, de nem utolsósorban a hálózat célja olyan szolgáltatások nyújtása is, ami a hálózat nélkül elképzelhetetlen lenne. Ilyenek a különböző kommunikációs szolgáltatások, köztük a nagy népszerűségnek örvendő ún. közösségi szolgáltatások (pl. Facebook).

Ezek alapján összefoglalhatjuk, kinek van szüksége hálózatra. Mindenkinek...

Aktivitás: Az előzőek alapján gondolja végig, önnek pontosan miért van szüksége hálózatra!

21. A hálózathoz szükséges eszközök

A hálózat definíció szerint több olyan gép olyan kapcsolata, amely valamilyen adatátvitelt, kommunikációt végez. Bármilyen kommunikációhoz viszont szükség van egy közös jelrendszerre, hogy a felek megértsék egymást. Ezt a jelrendszert, nyelvet kell mindenkinek használnia és értenie, aki részt vesz a kommunikációban. Természetesen szükség van olyan a fizikai eszközrendszerre is, ami a kommunikációt lehetővé teszi. A szükséges komponensek tehát:

- Kommunikációs szabvány
- A hálózatot szabvány szerint kezelni tudó eszközök
  - Operációs rendszer
    Természetesen a hálózatba kötött gép operációs rendszerének támogatnia kell a hálózati szabványt, illetve a kapcsolódáshoz szükséges eszközöket
  - A hálózathoz való fizikai kapcsolódást megvalósító eszköz
    Ez a „hálókártya”, pontosabban hálózati csatoló, amelynek nagyon sokféle típusa lehet

– Jeltovábbító átviteli közeg

  Ezek általában valamilyen kábelek, de lehet vezeték nélküli, sugárzás jellegű átvitel is: rádiófrekvencia, mikrohullám. Ilyen esetben speciális átviteli közegre nincs szükség, a rádiójelek levegőben, vagy légüres térben is terjednek.

– A hálózat működtetéséhez szolgáló különböző elektronikus berendezések. A jelek erősítésére a *repeater* (jelismétlő) szolgál, a jelek elosztására a *switch*. A korábban különálló hálózatok illesztésére való berendezés a *bridge* (híd) vagy a *router* (útválasztó), manapság a két eszköz funkciója összeolvadt, és jellemzően router névvel hivatkoznak rá.

21.1. ábra. *A hálózatépítés eszközei*

A 21.1. ábrán láthatjuk a hálózatépítés tipikus eszközeit: kétféle kábelt (a fekete optikai szálas, a kék a közismert csavart érpáros, lásd 22.4. fejezet), és egy hálózati eszközt. Az, hogy pontosan mi az eszköz funkciója, a képről nem megállapítható. Ha megszámoljuk a csatlakozókat, megállapíthatjuk, hogy egy ún. 48 portos eszközről van szó, ami azt jelenti, hogy 48 eszközt csatlakoztathatunk bele.

22. A hálózatok osztályozása

A hálózatok különböző szempontok szerinti osztályozása nagyban segít megérteni a hálózatok működését.

22.1. A hálózatok kiterjedtsége

A hálózat fontos ismérve a hálózat mérete, ahol méreten most azt a területnagyságot értjük, ahol a hálózat elhelyezkedik. Bár az alábbi csoportosításban az egyes kategóriák határai elmosódnak, más technológiát igényel egy néhány szobát vagy néhány kontinenst lefedő hálózat kiépítése, és sok esetben a hálózat létrehozásának célja is különbözik.

- PAN – Személyi hálózat (Personal Area Network)
  Mobiltelefon – notebook
- LAN – Helyi hálózat (Local Area Network)
  Otthon, iroda
- MAN – Városi hálózat (Metropolitan Area Network)
  Például kábeltévé hálózat, vagy a ma divatos Smart City („okos város”) városirányítási koncepció tipikus területe
- WAN – Kiterjedt méretű hálózat (Wide Area Network)
  Internet

Más, talán divatosabb csoportosítás szerint megkülönböztetjük a cégen vagy irodaházon belüli „helyi” hálózatot, és az azon kívüli világot:

- intranet,
- internet.

22.2. A résztvevő kommunikációs partnerek száma

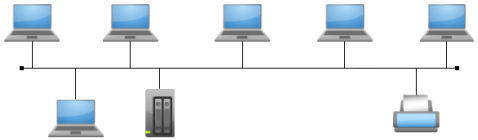
A kommunikációs partnerek számát tekintve két alapvető kommunikációs formát képzelhetünk el: az egyikben egy-egy, azaz pontosan két partner kommunikál, a másikban pedig több. (Hagyományos kommunikációban

előbbire példa lehet egy magánbeszélgetés két fél között, az utóbbira pedig például egy egyetemi előadás, amikor egyetlen beszélőt sokan hallgatnak.)

- Pont-pont kapcsolat
  Pontosan két csomópont kommunikál.
  Ne feledjük, ez csak egy logikai felosztás, nem törődünk azzal, hogy lehetséges-e ezt a kommunikációt harmadik félnek hallania. Arról, hogy a két csomópont közötti kommunikációról valóban senki más ne értesülhessen, azt biztonságtechnikai eszközökkel külön meg kell oldani (dedikált vonal, titkosítás). Például, két fél közötti skype beszélgetést a szoftver titkosít annak érdekében, hogy harmadik fél azt le ne hallgathassa valahol a hálózaton.
- Üzenetszórásos csatornára épülő kapcsolat
  A kommunikációs csatornán minden csomópont osztozik. A feladott üzenetet mindenki veszi, és aki nem címzett, az figyelmen kívül hagyja. Ezt úgy képzeljük el, hogy ha a folyosón valakinek a nevét kiáltva számára valamilyen feladatot közlünk („Bécike, gyere át az irodámba”), azt mindenki hallja, de közülük egyetlen ember kivételével mindenki figyelmen kívül hagyja az üzenetet. Az adó-szerep természetesen felcserélődhet („Nem érek rá”), ezt természetesen megint mindenki hallja. Arra a kommunikáló partnereknek figyelnie kell, hogy egyszerre mindig csak 1 állomás adhat: ha egyszerre adnának (azaz egymás szavába vágva kiabálnak) akkor az üzenet valószínűleg elveszik (nem érteni, mit mondtak) és az üzenetet meg kell ismételni.

22.3. Hálózati topológia

Hálózati topológiának nevezzük a csomópontok közötti kapcsolatok logikai elrendezését, más szóval, a hálózat alakját. Egy adott csomópont egy vagy több másik csomóponthoz kapcsolódhat. Grafikus ábrázolásnál a csomópontok közötti kapcsolatokat vonalak jelzik. A kapcsolatok lehetnek egyirányúak és kétirányúak, ilyen esetben egyszeres vagy dupla vonal, illetve nyíl mutathatja a kapcsolat irányát. Általában azonban a kapcsolatok kétirányúak, ezért a topológiatípusok ábrázolásánál egyszerű vonalakkal szokás a kapcsolatokat ábrázolni, és ezeken is kétirányú kapcsolatot értünk.



22.1. ábra. *Sín topológia*

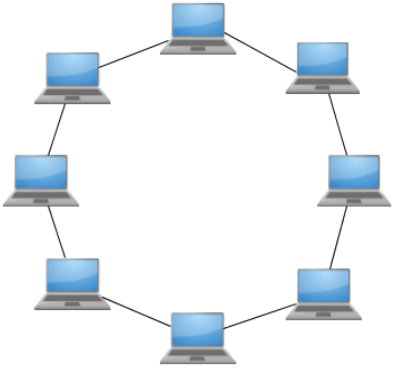
A topológia vizsgálatánál a csomópontok távolsága, adatátviteli sebessége, egyéb paraméterei nem érdekesek, csupán az összeköttetés tényét, illetve ennek mintázatát vesszük figyelembe. A különböző topológiák matematikailag ún. irányítatlan gráfokkal leírhatók.

Sín topológia

A sín topológia logikailag az egyik legegyszerűbb elrendezés. A rendszer a karácsonyfaizzók füzéréhez hasonlít. 1-2 évtizede még gyakori megoldás volt ez a topológia. A kapcsolatot biztosító olyan sínre voltak csatlakoztatva a számítógépek, amelynek két végét ún. lezáróelemmel zárni kellett. A sin szakadásakor (kábelszakadás) ilyen lezáróelemek természetesen nem voltak az újonnan keletkezett végeken, ezért mindkét hálózatdarab működésképtelenné vált. A sín topológiával felépített hálózat gépei a sínre küldött üzeneteket érzékelve kiválasztják a nekik címzettet, a nem nekik szólót figyelmen kívül hagyják. (Úgy működnek, mint az előzőekben említett hétköznapi kommunikáció.) Ez egy sor biztonsági kérdést is felvet, amelynek megoldására többféle módszer is ismert. Manapság ezeknek különösebb jelentősége nincs, mivel ezt a topológiát ritkán és akkor is egy egységen belül használják.

Gyűrű topológia

A sín topológiából a két végpont összekapcsolásával gyűrű hozható létre. Így zárt kör keletkezik, erre kapcsolódnak a számítógépek. Ez a topológia elméletileg már elviselhetne egy kábelszakadást, de a lezáró elemek a szakadás helyén természetesen hiányoznak, így a rendszer sín topológiában sem működhet tovább.



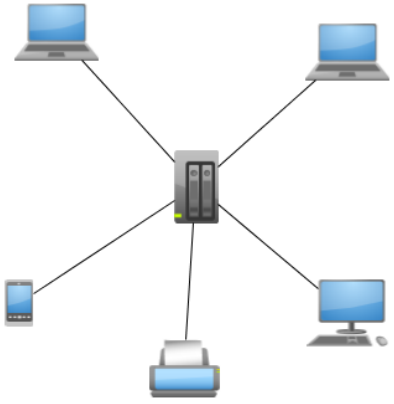
22.2. ábra. *Gyűrű topológia*

Az üzenetek küldése és fogadása a sín topológiánál leírtak szerint történik.

Csillag topológia

A csillag topológiát kisebb hálózatok esetén – legfeljebb néhány tucat csomópontig – manapság is használják. Ilyen méretnél még technikailag könnyen megoldható, hogy minden csomópont egyetlen központhoz csatlakozzon. Ez a topológia – az előzőekben tárgyalt topológiákkal ellentétben – centralizált, hiszen van a hálózatban kiemelt csomópont. Ez a központi csomópont általában egy „normál” számítógép, hanem egy erre a célra készített speciális hálózati eszköz. A csillag topológia csökkenti a hálózati meghibásodásból adódó problémák súlyát azzal, hogy minden csomópont kapcsolatban áll a központi csomóponttal, hiszen hálózati hiba esetén csak az érintett csomópont esik ki, a hálózat többi része működőképes marad. Természetesen, a központi eszköz meghibásodása működésképtelenné teszi a hálózatot, a gyakorlatban azonban ritkán jelent gondot, mert a hálózati eszközök igen megbízhatóak.

A központi elem fizikai korlátai miatt a csillag topológia legfeljebb néhány tucat csomópontig használható,

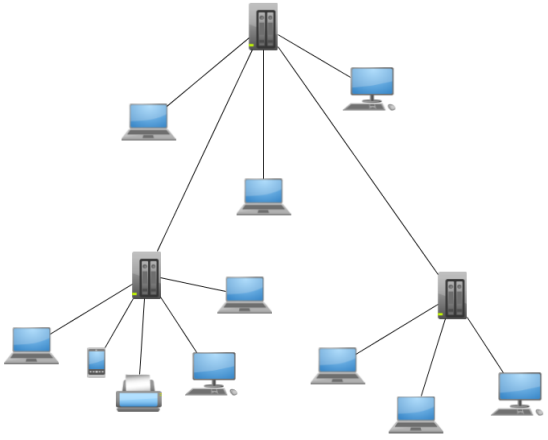


22.3. ábra. *Csillag topológia*

afölött a a 22.3. fejezetben tárgyalt fa topológia veszi át a helyét. A csillag topológia nagy gyakorlati jelentőséggel bír annak ellenére, hogy a nagy mennyiségű kábelezés miatt az eddigi topológiák közül ezt a legdrágább kivitelezni.

Fa topológia

A fa topológia a mai hálózatok tipikus alakja, például egy közepes méretű cég irodaháza esetében találkozhatunk vele. Több csillag topológiás hálózat összekapcsolásával jöhet létre, így több „központ” is van a hálózaton. Itt már lehetnek olyan gépek, amik több csomóponton keresztül érik el egymást (22.4. ábra), de minden gép között pontosan egy útvonal van, ami meghibásodás esetén problémát jelenthet. A fa topológiát a fákhoz hasonló alakjukról nevezték el. A topológia valamelyik csomópontja gyökérpontnak tekinthető, ebből a pontból a hálózat minden elemét el lehet érni. Ezt az elérési útvonalakat ágaknak szokás nevezni, azokat a hálózati eszközöket, amiknek csak egyetlen kapcsolatuk van, leveleknek nevezzük. A fa topológia szerint felépített

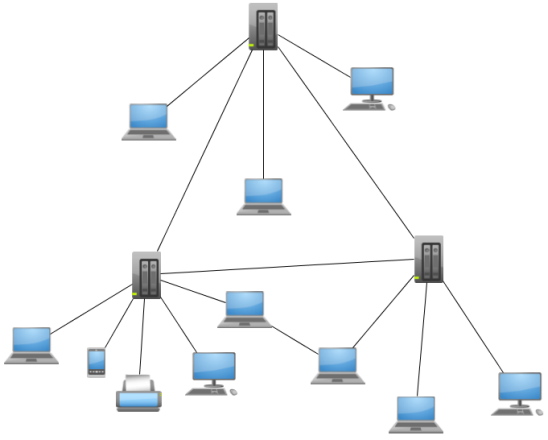


22.4. ábra. *Fa topológia*

hálózat minden számítógépe eléri a másikat, de csak egyetlen úton. Ha ez valamilyen okból megszakad, a két gép képtelen lesz egymással kommunikálni.

Szabálytalan topológia

A fa topológia legfőbb gyengesége, hogy bármely két számítógép között egyetlen, ráadásul akár több hálózati csomóponton átmenő út van, és ha bárhol hiba történik az útvonalon, az a két gép kapcsolata megszűnik. Ezen további, ún. redundáns kapcsolatok létrehozásával lehet javítani (22.5. ábra). Minél több redundáns kapcsolat van, a hálózat annál megbízhatóbb, hiszen valamely csomópont vagy útvonal meghibásodása esetén biztosan található alternatív útvonal. Ez a matematikailag általános gráfként leírható topológia jellemző a nagy hálózatokra, és az internetre.



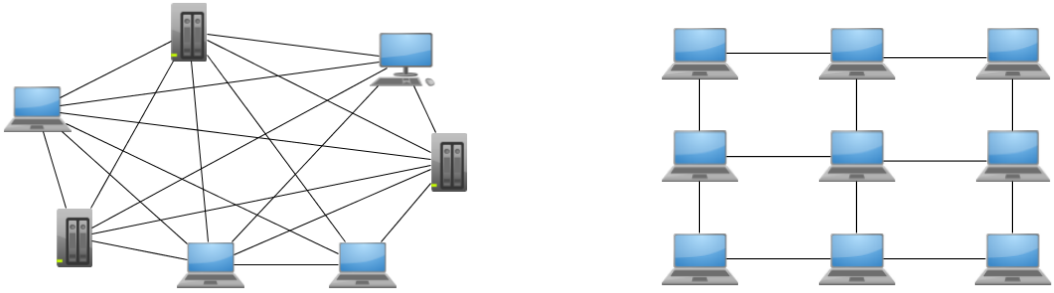
22.5. ábra. *Szabálytalan (gráf) topológia*

Teljes topológia

A teljes topológiát inkább csak elméleti lehetőségként említjük. Ebben a topológiában minden csomópontnak közvetlen kapcsolata van minden más csomóponttal (22.6. ábra). Ez a kommunikáció miatt kedvező lenne, azonban a gyakorlatban már egészen kis hálózatoknál komoly technikai problémákat vet fel, ezért a gyakorlatban nemigen használják.

Egyéb speciális topológia

Végezetül említsük meg, hogy speciális feladatokhoz időnként speciális topológiát terveznek. Egy-egy konkrét feladat kapcsán elképzelhető, hogy olyan jellemző kommunikációs minta figyelhető meg, hogy érdemes a hálózatot is ennek a mintának megfelelően összeállítani. A 22.6. ábra egy teljes és egy lehetséges speciális topológiát is mutat.



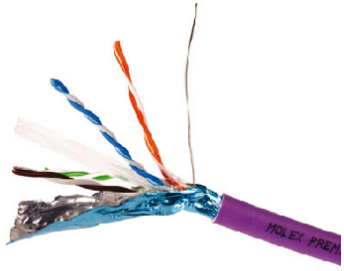
22.6. ábra. *Teljes és egy lehetséges speciális topológia*

22.4. Adatátviteli közeg

A hálózat – technikai szemmel nézve – lényeges tulajdonsága, hogy milyen adatátviteli közeg segítségével kommunikálnak a rá csatlakozó számítógépek. Ez egyébként az átlag felhasználót nem kell, hogy érdekelje. Az otthoni internetelérés esetében teljesen mindegy, hogy a jelek telefonvonalon, vagy tévékábelen, esetleg valamilyen mikrohullámú rádióhullámokon érkeznek, a legtöbb felhasználót a maximális letöltési sebesség érdekli. A részletekben természetesen vannak különbségek.

Alapvetően, két fő csoportra oszthatjuk az adatátviteli közegeket:

1. Vezetékes
   - Elektromos vezeték
     Ezekben a – jellemzően rézből készült – vezetékekben elektromos jelek viszik az adatok kódjait. A nagyon egyszerű működési elv, és az olcsó megvalósítás mellett vannak egyértelmű hátrányok: a kábelek elektromágnesen és más zavarokra érzékenyek, és a maximális átviteli távolság pár száz méter körül van, ennél nagyobb távolságokra jelismétlőket (a jelet befogadó, majd felerősítve továbbküldő) berendezéseket kell telepíteni).



22.7. ábra. *Csavart érpáras kábel*

Leggyakoribb kábeltípus, amivel a gyakorlatban találkozhatunk, az UTP (Unshielded Twisted Pair – árnyékolatlan csavart érpár), amit a 22.7. ábrán is láthatunk. Ennek a kábeltípusnak 4 pár, egyenként szigetelt vezetéke van. A páronkénti összesodrás a különböző zavarokat csökkenti. Az egyes összesodort és árnyékolatlan érpárok közötti áthallást úgy csökkentik, hogy különböző mértékben sodorják az egyes párokat. Egy mechanikai erősítést szolgáló (nyújtást, szakítást akadályozó) erős acélszál is van a 8 adatvezeték mellé szerelve, majd egy alufólia vagy hasonló árnyékoló veszi körbe az összes vezetéket. A különböző felhasználási területekre szánt UTP kábeleket kategorizálják, a CAT5, CAT5e, és CAT6 kábeleket használnak manapság az irodákban, előbbi másodpercenként 100 Mbit, utóbbi kettő 1000Mbit átvitelére képes – legfeljebb kb. 100 méteren.

- Fényátvitelen alapuló vezeték
  A fényvezető kábelek sok előnnyel büszkélkedhetnek az elektronikus átvitelt használó vezetékekkel szemben. Az optikai kábeleken keresztül továbbított jelek nem érzékenyek külső villamos zavarra (például villámlásra) – a fényvezetők nemfémes anyagból készülnek, így nem vesznek fel és nem bocsátanak ki elektromágneses, vagy rádiófrekvenciás zavarokat. Ezek a nemfémes anyagok jellemzően különböző üvegek (innen az üvegszál elnevezés) vagy műanyagok, ezek villamosan szigetelnek, így a csatlakoztatott készülékekben villamos meghibásodásokat nem okozhatnak.

Mivel elektromágneses jellegű lehallgatás optikai kábeleken nem lehetséges, sokkal biztonságosabbak, mint a csavart érpáras rézkábelek (de lehallgatásuk így sem lehetetlen).
Az optikai kábelek nagyon fontos tulajdonsága, hogy a bennük futó jelek akár 100 kilométeres távolságra is továbbíthatók erősítés nélkül.

2. Vezeték nélküli

- Infravörös
  Az infravörös (IR) fény alacsony energiaszintű, közvetlen rálátást igénylő kommunikációs forma, a jelek nem képesek falakon, de még kisebb akadályokon sem áthatolni (lásd: távirányítók)

- Lézer
  Általában néhány száz méter, esetenként néhány kilométer áthidalására alkalmasak. A lézer terjedését, a hétköznapi lámpafényhez hasonlóan, zavarják a légköri szennyeződések és lebegő anyagok (köd, eső, por). A lézerfény különlegessége, hogy nem széttartóan, hanem iránytartóan, minimális szóródással terjed, ezért alkalmas az átvitelre.

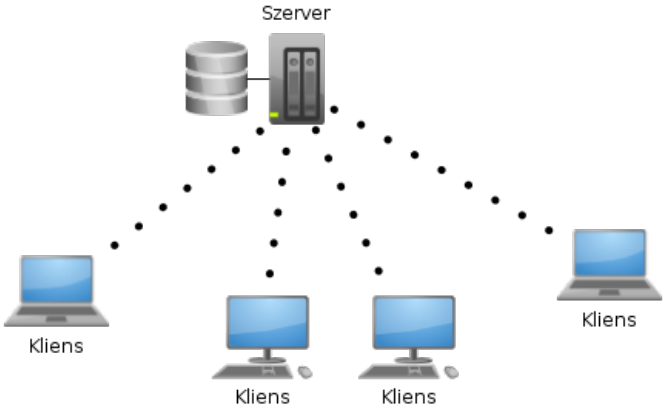
- Rádióhullám
  A rádióhullámok antenna által kisugárzott elektromágneses hullámok, sebességük a fénysebességgel egyezik meg. A terjedési tulajdonságai frekvenciafüggőek: az alacsony frekvenciájúak minden irányban terjednek és sok akadályon (pl. épületek) viszonylag jól áthaladnak. A nagyfrekvenciás hullámok (100 MHz felett) kevésbé szóródva, szinte egyenes vonalban terjednek (olyannyira, hogy nagyobb távolságnál már a Föld görbülete is problémát jelent, ezért is érdemes minél magasabb tornyokat használni), jól fókuszálhatók, a tereptárgyakról pedig visszaverődnek. Az elektromágneses hullámok kényesek más elektromos berendezések (például nagy teljesítményű villanymotorok) közelségére.
  Különösen fontos a gyakorlatban a mikrohullám. A kb. 300 MHz-től 300 GHz-ig terjedő tartományt nevezzük így. A nagyobb sávszélességet kívánó vezeték nélküli kommunikáció tipikus megoldása, régebben igen elterjedt volt, az optikai kábelek megjelenésével és egyre olcsóbbá válásával a szerepe csökkent, de széles körű használata továbbra is megmaradt.

Érdekesség: A GSM telefonhálózat is a mikrohullámú tartományt (Európában 900 MHz a legelterjedtebb) használja.

22.5. A hálózati modellek – a résztvevők rangja

A hálózati együttműködés vizsgált szempontja az is, hogy a hálózat résztvevői milyen viszonyban állnak egymással. Itt hardver és szoftver vonatkozásokat együtt vizsgálunk. Alapvetően mellé- és alárendelő viszonyról beszélhetünk, azaz:

- Egyenrangú résztvevők
  Az ún. Peer to Peer (P2P) hálózat lényege, hogy a hálózat résztvevői egyenrangúak, így közvetlenül egymással kommunikálnak, bármiféle (magasabb rangú) központi hálózati csomópont nélkül. Bárki lehet tehát szolgáltató vagy szolgáltatást igénybe vevő szerepben. A módszer előnye a skálázhatóság, hibatűrés, hátránya azonban a pazarló erőforrás-kihasználás, bonyolultabb adminisztráció és megvalósíthatóság. (A P2P fogalmat nemcsak hálózatokra, hanem önállóan, közvetlenül egymáshoz kapcsolódó szoftverekre is szokás használni.)
- Nem egyenrangú résztvevők
  Az eltérő szerepek miatt dedikált (kitüntetett) gépek is vannak a hálózaton.
  - Kliens – szerver architektúra
    Amint az a 22.8. ábrán látható, (az egymás között egyenrangú) kliens számítógépek kiszolgáló géphez kapcsolódnak. A kiszolgáló (szerver) gép valamely szolgáltatást nyújt, a kliensek pedig igénybe veszik ezt a szolgáltatást. Természetesen a többféle egyidejű szolgáltatáshoz akár egy időben több kiszolgáló is működhet, vagy egy hardver eszköz többféle szolgáltatást is nyújthat. Egy kiszolgáló is lehet kliens is valamely más szerver szolgáltatásának vonatkozásában.
  - Terminálszerver – terminál (vékony kliens)



22.8. ábra. *Kliens - szerver architektúra*

A terminálszerver architektúra nagyban hasonlít a klasszikus kliens-szerver megoldásra. Annyi a (jelentős) különbség, hogy amint említettük, a hagyományos kliens számítógép önmagában is működőképes, több szerverhez is kapcsolódhat, illetve más szolgáltatás vonatkozásában akár maga is lehet kiszolgáló szerepben is. Ezzel szemben a terminálszerver kliensei (amiket terminálnak, illetve vékony kliensnek hívunk) egyetlen szolgáltatóhoz kapcsolódnak és nem lehetnek más szolgáltatás kiszolgálói. A vékony kliensek minden szolgáltatás-igényét egyetlen szerver, a gazdagép szolgálja ki. A működés közben gyakorlatilag mindent (az operációs rendszert, annak szoftvereit és a szükséges hardver erőforrásokat is) a gazdagépről vesznek, helyben csak a technikailag minimális szinten szükséges erőforrások vannak jelen.

A hálózati rendszer működése a következő: a vékony kliens bejelentkezik a gazdagépre, a felhasználó gyakorlatilag a gazdagépen dolgozik. Helyben csak az adatok megjelenítése, adatbevitel stb. zajlik. Ennek megfelelően a vékony kliensben csak egy minimális hardver (processzor, memória

22.9. ábra. *Vékony kliensek*

stb.) és egy olyan, végletekig leegyszerűsített operációs rendszer van, aminek legfőbb képessége a terminálszerverhez (alkalmazás-szerverhez) kapcsolódni. Ebben rejlik a terminálszerveres munkahelyek kialakításának legfőbb előnye: az alkalmazások, adatok mind-mind egy könnyen védhető, biztonságos, megbízható kiszolgálón vannak, a felhasználók pedig olcsó, kicsi, zajtalan, keveset fogyasztó, nagyon üzembiztos, de meghibásodás esetén mégis pillanatok alatt kicserélhető, nagyon kis helyet foglaló eszközt használhatnak (természetesen monitor, billentyűzet, egér ugyanúgy csatlakozik hozzájuk). A 22.9. ábrán két vékony kliens látható, a rajtuk található csatlakozókból, gombokból következtethetünk a készülékek méretére. Használatuk leginkább ott előnyös, ahol sok ember ugyanazzal a jól definiált szoftverkészlettel dolgozik – márpedig nagyon sok olyan munkahely van, ahol a cég egy-két specifikus (például termelésirányító, számlázó, ügyfélszolgálati) szoftverére és mellette 1-2 általános szoftverre (szövegszerkesztő, levelező, internetböngésző) van szükség, semmi más használatát nem engedélyezik és nem is szükséges a használatuk.

Aktivitás: Keressünk 2-3 példát olyan munkakörülményekre, ahol a vékony kliensek előnyösen alkalmazhatók!

Összehasonlítva a kliens-szerver modell kliensét és a terminálszerver vékony kliensével:

- Normál számítógép (PC, munkaállomás, notebook)
  Önálló munkára képes
  Sok saját erőforrás (saját, tetszőlegesen erős hardver, és saját szoftverek)
  Drágább, bonyolultabb
  Általános célra, bárki mástól függetlenül használható
- Vékony kliens / terminál
  Olcsóbb (a beszerzési kisebb az üzemeltetés kisebb költségű).
  Nincs meg minden szükséges erőforrása
  A szerver nélkül nem tudja a feladatait ellátni, csak a szerver által biztosított feladatokra alkalmas
  Minden fontos adat központilag tárolódik (komoly biztonsági és szervezési előny)

Önellenőrzés

1. Mely kijelentések igazak a vezetékes adatátviteli közegekre?
   - ☐ Mindig olcsóbb, mint a vezeték nélküli megoldások.
   - ☐ Jellemzően elektronikus- vagy fényjelek segítségével történik a kommunikáció.
   - ☐ Ebbe a csoportba tartoznak a rézkábelek.
   - ☐ Napjainkban egyre inkább háttérbe szorulnak.
   - ☐ Kizárólag elektronikus jelek átvitelével működhetnek.
   - ☐ Némely változatuk infravörös adatátvitelt alkalmaz.
   - ☐ Némely típusuk elektromágneses zavarokra érzékeny.
2. Miért hasznos a hálózatokat kiterjedés szerint osztályozni?
3. Milyen előnyei vannak egy normál számítógépnek a vékony klienssel szemben?
   - ☐ Olcsóbb
   - ☐ Üzembiztosabb
   - ☐ Karbantartása egyszerűbb
   - ☐ Nagyobb teljesítményű
   - ☐ Kisebb fogyasztású
   - ☐ Általában kisebb helyet foglal
   - ☐ Hálózati kapcsolat nélkül is használható
   - ☐ Általános célra, sokoldalúbban használható
4. Javasolna-e vékony klienst banki ügyintézők, illetve fejlesztőmérnökök asztalára? Mivel indokolná a döntését?

9. LECKE

Számítógép-hálózatok 2.

23. Általános hálózati architektúra

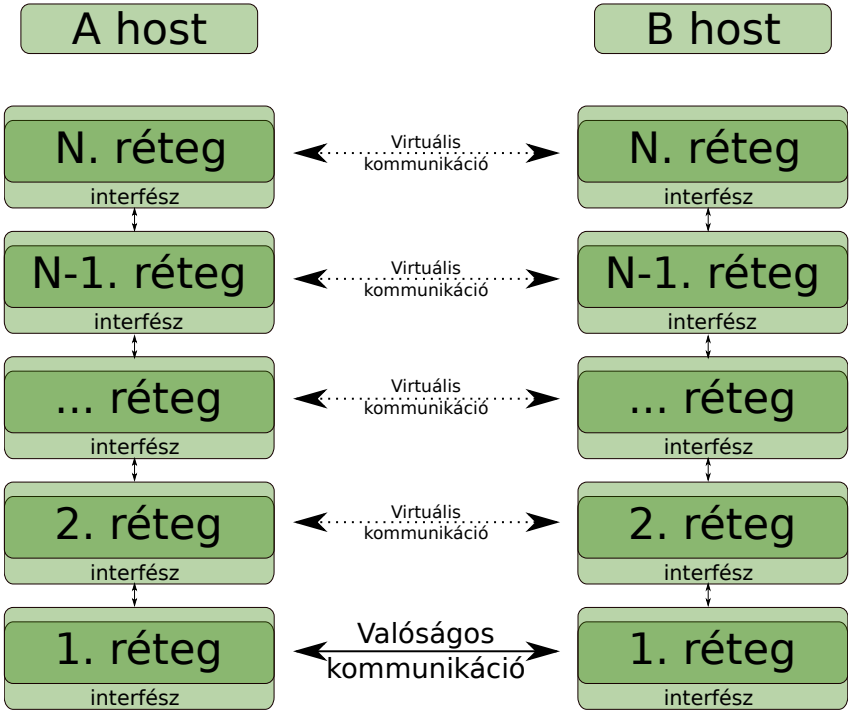
E tananyag olvasóinak már bizonyára furcsán hangzik, de 1-2 évtizede még voltak olyan számítógépek, operációs rendszerek, amiket semmiféle hálózati működésre nem készítettek fel. Gondolkozzunk el azon, mi kellene ahhoz, hogy egy ilyen gépbe egy hálózati csatoló kártyát szerelve azt működésre bírjuk, és mondjuk egy fényképmegosztó programot írjunk rá. Érezhetjük, hogy igen szerteágazó feladatunk lenne. Felhasználóként csak a „Feltöltés” gombot kell megnyomni, de a programozó érzi, sok minden kell ahhoz, hogy a gépbe dugott kábelen elinduló jelsorozat segítségével fotóink feltöltődjenek egy szerverre.

Ezernyi dologra kellene figyelni, vegyünk sorra közülük néhányat: maga az alkalmazás felhasználói felületének elkészítése lenne a „fő” feladat, de emellett a hálózati funkciók elkészítése igen sok energiánkat kötné le. Foglalkoznunk kellene azzal, hogy a gépeket összekötő kábelen végül is milyen jelek mennek a drótban, vagy azzal, hogy a célállomást hogyan találjuk meg a hálózaton. A titkosítási, biztonsági kérdéseket sem hagyhatnánk figyelmen kívül. Sok-sok részfeladatot kellene megoldanunk, mire az alkalmazás működőképes lenne. Ha pedig egy másik alkalmazást is írnánk elavult gépünkre, bizonyos programrészek ismétlődnének abban a programban is.

Régről adódott az ötlet, hogy ha már a hálózat kezelése egyrészt bonyolult, másrészt gyakran ismétlődő feladat, egy egységes koncepciót kellene kialakítani a sok részfeladat ügyes szervezéséhez. A különböző hálózati feladatokat ezért részfeladatokra bontották. A részfeladatokat ún. *rétegekbe* szervezték (23.1. ábra), utalva arra, hogy a részfeladatok között vannak alacsonyabb és magasabb szintű feladatok, illetve ezek meghatározott sorrendben kapcsolódnak egymáshoz.

Egy réteg igénybe veszi az alatta lévő réteg szolgáltatásait, és jól definiált szolgáltatásokat nyújt a felette lévő rétegnek. Az egyes rétegek szolgáltatásainak megvalósításának részletei rejtve maradnak más rétegek elől. A rétegek közötti kommunikáció definiált módját nevezzük protokollnak. Maga a kommunikáció a rétegek csatlakozási felületein, az interfészeken keresztül zajlik. Az interfészeken kódok, és különböző vezérlő információk cserélődnek a két szomszédos réteg között.

A valóságos kommunikáció mindig a legalsó rétegek között zajlik, de ebben az architektúrában a felsőbb rétegek is úgy érzékelik, mintha egy másik, azonos réteggel kommunikálnának. Az egyes rétegek, mivel



23.1. ábra. *Általános, rétegekbe szervezett hálózati architektúra*

virtuálisan a saját szintjükön kommunikálnak más hálózati csomópontokkal, saját kapcsolatfelépítési és -lebontási mechanizmust is megvalósíthatnak.

23.1. definíció: Hálózati architektúrán a rétegek és protokollok (szabályrendszerek) rendezett halmazát értjük.

23.1. ISO-OSI hálózati referencia modell

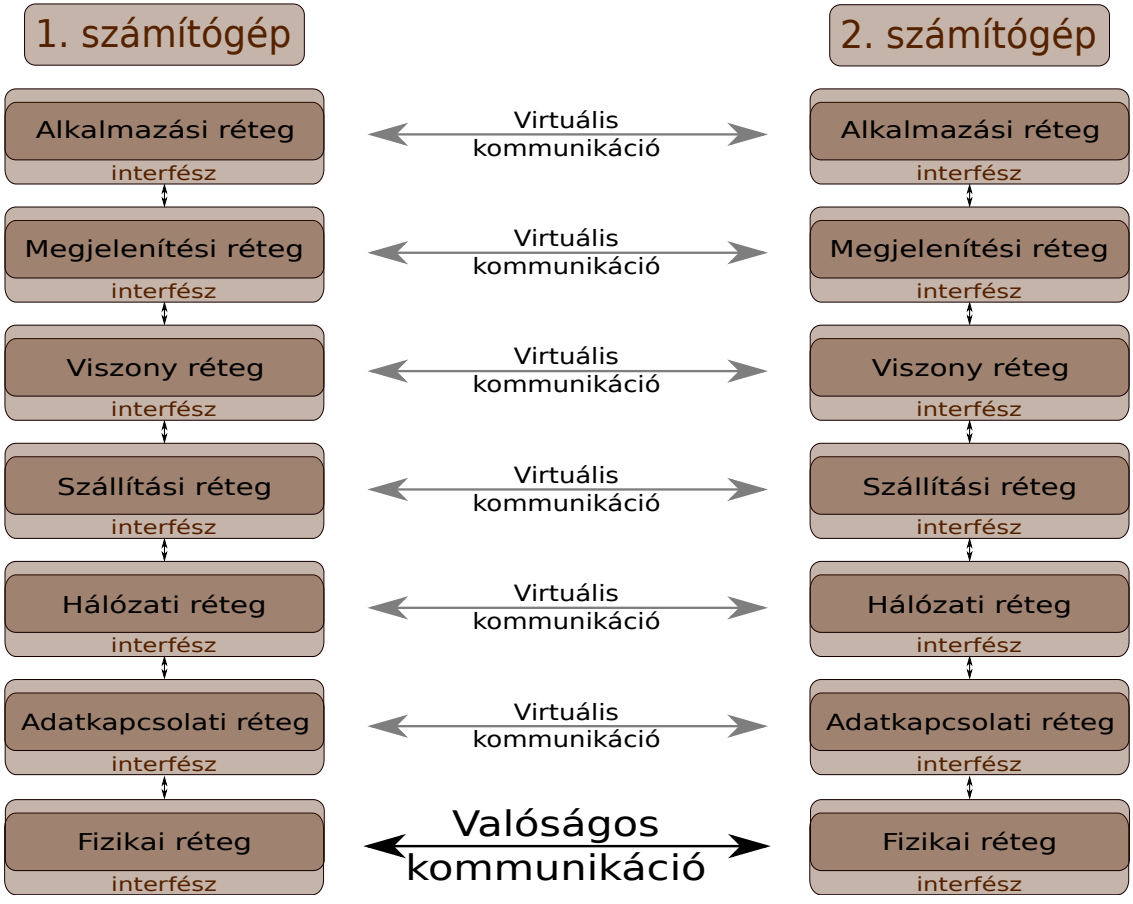
Az International Standards Organisation (ISO) nevű szabványügyi szervezet az általunk a 23. fejezetben megismert elvek alapján készített egy referenciamodellt (Open System Interconnection – nyílt rendszerek összekapcsolása), amivel a hálózatépítés későbbi irányát szerette volna kijelölni és megtartani.

A fentiek rövidítéséből adódó ISO-OSI modell egy absztrakt leírás a a hálózati funkciók rétegekbe szervezésének, gyakorlatilag egy ajánlásnak foghatjuk fel, ahol az egyes rétegek feladata van megfogalmazva. Ebből adódóan a részletes megvalósítást (implementációt) nem tartalmazza a modell, és az interfészek specifikációjára sem tér ki. Úgy is fogalmazhatunk, hogy nem konkrét protokollokat, hanem csak feladatokat, funkciókat határoz meg. Az egyes rétegek kialakításának szempontjai a következők:

- Különböző absztrakciós szinteket képviseljenek,
- Minimális kommunikáció legyen szükséges a rétegek között,
- Egy rétegbe kevés, jól elkülöníthető (elemi) feladat kerüljön,
- Rétegek száma – a többi szempont figyelembe vétele mellett – minimális legyen,
- Egy-egy réteg feladata szabványosítható legyen.

Az ISO-OSI modell, aminek felépítését a 23.2. ábrán láthatjuk, tehát az általa ajánlott 7 réteg funkcióinak meghatározását jelenti. Nézzük sorban az egyes rétegeket, a legalacsonyabb szinttől kiindulva:

1. Fizikai réteg: bitfolyam
   A bitek kommunikációs csatornán történő áthaladásáért felelős. A tényleges átvitel úgy történik, hogy a fizikai átviteli közeg valamely tulajdonságát megváltoztatja, a vevő pedig ezt a változást érzékelve képes abból az eredeti adatokat visszaállítani. Az egyes bitek reprezentációja tehát valamilyen fizikai paraméterrel (például feszültségszinttel) történik.
   A fizikai réteghez tartozik az adatátviteli közeg és csatlakozók kialakítása (például milyen anyagú vezetéket, hány „lábú” és milyen alakú csatlakozót használjunk). Ha az átvitel analóg, akkor a digitális számítógép jeleit át kell alakítani, és az átvitel után visszaalakításnak kell következnie mielőtt a jel a számítógépre



23.2. ábra. *Az ISO-OSI hálózati modell*

jutna. Ezt a feladatot is a fizikai rétegben kell megoldani. A feladatot egy speciális csatoló elem, a modem végzi.
Az adatátvitel kezelése bitszinten történik, és itt kerülnek meghatározásra az adatátviteli irányok. A valódi kétirányúság neve a *full duplex*. Az ún. *half duplex* átvitelt úgy képzeljük el, hogy az adatáramlás kétirányú lehet, de nem egyidőben, hanem felváltva. A *simplex* átvitelnél csak egyirányú átvitel lehetséges.

2. Adatkapcsolati réteg: keretek
   Feladata, hogy a hálózati réteg számára hibamentes átvitelt biztosítson két hálózati eszköz között.
   Három fő feladat:
   - A hálózati rétegtől kapott információkat ún. *keretekbe* rendezi.
   - Hibamentesség biztosítása hibaellenőrzéssel és -javítással.
   - A keretek továbbítása és a továbbítás nyugtázása.
     Adatfolyam vezérlés (például lassú vevő megfelelő kezelése)

   Ebben a rétegben dolgozik a *switch* és a *bridge*.

3. Hálózati réteg: csomagok
   Feladata, hogy a kommunikáló felek közötti, tetszőleges hosszúságú adatokat továbbítsa. Ehhez meg kel határozni a csomagok közlekedési útját (útvonal-választás, angolul routing, innen ered a *router* elnevezés) – a router a hálózati szinten működik. A router útvonal-kiválasztási stratégiája lehet statikus vagy dinamikus, előbbi esetben előre beállított szabályok alapján, utóbbi esetben pedig a forgalom elemzése alapján dönt a csomagok irányításáról. További funkció az adatáramlás ellenőrzése (torlódásvezérlés, ismétlés) és folyamatosan szükséges hibaellenőrzés.

4. Szállítási réteg: datagramm, szegmens
   A szállítási réteg legfőbb feladata, hogy az adatátvitelt a felhasználó számára átlátszóvá tegye. Valódi forrás-cél (end-to-end) réteg. Biztosítja az adott kapcsolat megbízhatóságát, az üzeneteket igény esetén akár sorrendhelyesen állítja vissza.

5. Viszonyréteg: üzenetek
   A különböző gépek közötti „párbeszéd” lehetőségének biztosítása. A párbeszéd egyfajta „felhasználói viszony”, innen az elnevezés. A párbeszédek velejárója a szinkronizáció, lehetőség van szinkronizációs pontok meghatározására, ahonnan hiba esetén meg lehet ismételni a párbeszédet. A réteg figyeli és feloldja az esetleges kölcsönös várakozási (*deadlock*) helyzeteket. A párbeszédek szervezésében kialakítható egy-vagy kétirányú kapcsolat is.
6. Megjelenítési réteg: üzenetek
   Az átadandó információ szintaktikájával és szemantikájával foglalkozik. Ebbe a rétegbe a következő feladatok tartoznak: kódkonverzió (például különböző szabványos kódolások között), titkosítás, tömörítés.
7. Alkalmazási réteg: a felhasználói szintű üzenetek
   A réteg feladata a különböző kommunikációs szolgáltatások biztosítása. Ezek azok a szolgáltatások, amiket a felhasználók ténylegesen igénybe vesznek. Ilyenek lehetnek például a következők:
   - Fájl-szolgáltatások: fájlátvitel (FTP), archiválás, adatmentés
   - Nyomtatószolgáltatások: nyomtatás távoli nyomtatóra
   - Kommunikációs szolgáltatások
   - Elektronikus levelezés: SMTP, MIME, POP3, IMAP protokollokkal
   - Hálózati erőforrások nyilvántartása
   - Emberi erőforrások nyilvántartása (bejelentkezési azonosítók, jogosultságok, telefonszámok)
   - Alkalmazás szolgáltatások
   - Távoli géphasználat (SSH, távoli asztal)
   - Böngészők (HTTP)
   - Adatbázisok távoli elérése

Később az OSI modellben a 4. és 5. (szállítási- és viszonyréteg), illetve a 6. és 7. (megjelenítési és alkalmazási)

rétegeket összevonták, így egy egyszerűbb, ötrétegű modell jött létre. Ennek gyakorlati példáját vizsgáljuk meg a következő fejezetben.

24. TCP/IP

24.1. definíció: A TCP/IP (Transmission Control Protocol/Internet Protocol – átvitelvezérlő protokoll/internet protokoll) az internetet felépítő protokollok összefüggő rendszere.

A TCP/IP sok protokoll összessége, a neve is két fontos protokollból (TCP, IP) adódik.

Kifejlesztése az 1960-as években kezdődött, de csak 1979-ben jelent meg az első igazán sikeres 4. verzió, amit mind a mai napig széles körben használnak (IPv4). Az ezután következő 5. verzióra sosem tértek át, manapság a 6. verzió (IPv6 – lásd később) gyors bevezetése folyik.

A TCP/IP az ISO/OSI modellben leírt réteges felépítésben működik, ahol minden egyes réteg egy jól definiált feladatot lát el, és a rétegek egymás között szintén jól definiált módon kommunikálnak. A modernizált, 5-rétegű ISO/OSI modellnek megfelelően a TCP/IP 5-rétegbe szervezi a hálózati kommunikációval kapcsolatos feladatokat. A rétegek feladatai megfelelnek a modellnek, így nem is szorulnak különösebb részletes magyarázatra.

Az alábbiakban az egyszerűség kedvéért csak a küldő fél feladatait írjuk le, a fogadó fél értelemszerűen az adott feladatokat megfordítva látja el (például nem kereteket képez, hanem a kapott keretekből a fejléc-információk alapján visszaállítja az eredeti adatsort). Magát a kommunikáció folyamatát a korábbiaktól eltérően most a legfelső, alkalmazási rétegtől lefelé tekintjük át, mert így könnyebb a megértése:

- Alkalmazási réteg
  Az alkalmazási réteget az operációs rendszerek segédprogramjai és a különböző felhasználói alkalmazások valósítják meg. Az itt dolgozó protokollokra néhány példa: HTTP, FTP, IMAP, POP3, SMTP, DHCP, DNS, SSH, Telnet, BitTorrent, IRC.

A kapcsolatok azonosítására az ún. *portok* (kapuk) szolgálnak, a szerver mindig 1-1 porton érkező kéréseket vár, hogy kiszolgálhassa.
A szabvány 65535 TCP és ugyanannyi UDP portot engedélyez (e két fogalmat a szállítási rétegnél tisztázni fogjuk), ebből az első 1024 pedig a szabványosított protokollok számára van fenntartva (például, TCP 21: FTP, TCP 80: HTTP. A TCP magyarázatát annak helyén, a szállítási réteg leírásánál olvashatjuk.)

- Szállítási réteg
  Az alkalmazási rétegtől kapott kódok feldolgozását kezdi meg azzal, hogy kiegészítő információkat (fejlécet, idegen szóval headert) csatol hozzá, ami tartalmazza többek között, hogy milyen protokollal történik az üzenetváltás – legjellemzőbb a TCP és az UDP.
  A TCP (Transmission Control Protocol) a fölötte álló rétegnek megbízható adatátvitelt biztosít úgy, hogy alatta valójában megbízhatatlan protokoll található. Kapcsolatorientált, full duplex átvitelt biztosít, kliens-szerver kapcsolatok kialakítását támogatja. Az adatátvitelben nyugtázás történik mindkét oldalon, ezért megbízható az adatátvitel. A megbízhatóság növelésére ún. ellenőrző összeget is számol a protokoll a csomagra. Az ellenőrző összeg egy matematikai algoritmus alapján előállított számsor, aminek segítségével nagy biztonsággal eldönthető egy kapott kódsorozatról, hogy történt-e rajta sérülés az átvitel során.
  Az UDP (User Datagram Protocol) gyorsabb, azonban az adatátvitel szempontjából nem megbízható. Azt is mondhatjuk, hogy ott használják, ahol a gyorsaság fontosabb a megbízhatóságnál. Tipikus felhasználói terület a valós idejű (real-time) adatátvitel: például az online filmnézés, zenehallgatás. Egy online rádió hallgatásánál, ha bármely okból adatátviteli hiba történik, legfeljebb sercen egyet a hang: ez kisebb baj, mintha az épp hallgatott zeneszám lejátszása megállna a háttérben folyó ellenőrzési-újraküldési folyamat miatt, majd az adott ponttól újraindulna a lejátszás.

- Hálózati (internet) réteg
  A szállítási rétegtől megkapott, adatcsomaghoz egy újabb fejléc-információt csatol arról, hogy a szállított adat a hálózat mely végpontjának van címezve. Ide tartozik az IP protokoll, aminek jelenleg két verziója van használatban: az IPv4 és IPv6. Az IPv4 protokollra a későbbiekben részletesebben is kitérünk.

- Adatkapcsolati réteg
  Az adatkapcsolati réteg az átvivendő kódokat egyforma, kezelhető méretű darabokra, ún. keretekre bontja,

ehhez természetesen újabb fejléc hozzáadása szükséges a visszaállíthatóság miatt. Az ide tartozó régebben gyakran használt protokollok a PPP, FDDI, Token-Ring, manapság pedig a népszerű Ethernet vagy az IEEE 802.11a/b/g/n (népszerű nevén: Wi-Fi).

- Fizikai réteg
  A fizikai réteg végzi a tényleges adatátvitelt a rendelkezésre álló fizikai kommunikációs csatornán. Valójában az adatkapcsolati rétegtől kapott kereteket küldi át a csatornán, az ehhez kapcsolódó részfeladatok (pl. hibaellenőrzés, esetleges ismétlés) elvégzésével együtt: a feljebb levő rétegek már hibamentesnek látják a csatornát. Ide tartozó ismert protokollok például: RS-232, 10Base-T, 100Base-TX, 1000Base-TX. A felsőbb rétegek ettől a rétegtől valójában csak annyit várnak el, hogy bájt-folyamokat tudjanak küldeni és fogadni.

A TCP/IP protokollcsalád a fentieken kívül még több protokollt is tartalmaz, itt csak a legfontosabbakat és legismertebbeket említettük meg.

A sok-sok protokollt alaposan megismerni egyenként sem egyszerű, az összeset pedig egészen összetett feladat. Éppen ezért vannak az üzenetváltást megvalósító folyamat rétegekre, és az egyes rétegek különböző funkciói protokollokra bontva. A sok-sok protokoll, amiket precíz szabálydefiníciónak foghatunk fel, így alkalmas arra, hogy *egy internet méretű* hálózatot különösebb meghibásodás nélkül évtizedek óta, non-stop működtessen. Gondoljuk csak meg: az internet, mint világméretű rendszer, évtizedek óta működik, és rohamosan fejlődik úgy, hogy sosem kellett és nem is lehet karbantartási vagy javítási céllal lekapcsolni. A csatlakozók között szabványossági problémák sem merültek fel. Az internet világméretű elterjedését, nagy sikerét és példátlan megbízhatóságát éppen e sok, jól átgondolt, egymást jól kiegészítő protokoll tette lehetővé.

Képzeljünk egy percre bele abba, hogy az internet, mint világméretű műszaki berendezés, évtizedek óta működik, és tudjuk jól, rohamosan fejlődik úgy, hogy sosem lehet karbantartási vagy javítási céllal lekapcsolni, és a sok-sok csatlakozott ország között szabványossági problémák sem merülnek fel. A szabványosságot, mint előnyt nem véletlenül vetettük fel: az ipar és a társadalom más területeken sokkal egyszerűbb szabványosítási törekvéseket sem tudott maradéktalanul megoldani. (Gondolhatunk például az vonatok országonként eltérő nyomtávjára, vagy a bal- és jobboldali közúti közlekedés problémáira, de például mindannyian tudjuk, hogy

egy otthon felejtett telefon- vagy notebook töltő helyett nem könnyű egy megfelelőt kölcsönkérni.) Az internet világméretű elterjedését, nagy sikerét és példátlan megbízhatóságát éppen e sok, jól átgondolt, egymást jól kiegészítő protokoll tette lehetővé.

Később szólunk róla, de itt érdemes megemlíteni, hogy a jelenleg széles körben használt IPv4 éppen az internet óriási elterjedtsége miatt nem használható tovább, a már megtervezett új verziójára történő átállás már megkezdődött és valószínűleg még hosszú évekig tart úgy, hogy a milliárdos létszámú felhasználói tábora ebből gyakorlatilag semmit nem vesz észre.

24.1. IPv4 hálózatok

Tekintsük át röviden, hogyan áll össze a TCP/IP protokolljaival egy hálózat, például az internet! A jelenleg még legáltalánosabban használt IP 4-es verzióra (IPv4) alapozzuk a bemutatást.

Ahhoz, hogy az egyes hálózati eszközök egyértelműen azonosíthatóak legyenek, minden eszköznek (számítógép vagy hálózati berendezés) egyedi azonosítóra van szüksége. Ez a MAC Address (Media Access Control Address, eszközhozzáférés-felügyelet cím): a hálózati csatolót, mint hardvereszközt azonosító 48-bites szám. 48 bitet hexadecimális alakban is 12 számjeggyel (karakterrel) írhatunk le, így a könnyebb olvashatóság miatt 6 számpárral szokták a MAC címet leírni, a számpárokat kettősponttal elválasztva, például:

00:1E:33:7E:60:E2

A hálózati eszközökre, illetve a hálózati csatolót tartalmazó eszközökre (például notebook) a gyártók a legtöbb esetben (a gép alján lévő matricákra) ráírják az eszköz MAC címét.

A 48 biten elvileg $2.8 * 10^{14}$ eszköz különböztethető meg, ez több, mint elég. Az első 6 karakter a gyártót/forgalmazót azonosítja (OUI, Organizational Unique Identifier – szervezeti egyedi azonosító). Az ezután következő második 6 karaktert a gyártó adja ki sorszámként. Komolyabb gyártók hamar elhasználják ezt a 24 bitet, ilyen esetben újabb OUI-t vagy akár OUI tartományt igényelhetnek.

Érdekesség: Az interneten találunk olyan szolgáltatást, ami például a saját gépünk hálózati csatolóiról a MAC cím alapján megmondja, hogy milyen gyártmányúak.

A MAC címek rendszerével egy hálózat összes csomópontja megkülönböztethető – két probléma ezért akad: ezeket a számokat nagyon nehéz észben tartani, illetve a gyártó által beállított statikus számokkal nehéz lenne logikai hálózatokat kialakítani. A probléma megoldásának első lépése az IP-címek használata.

24.2. definíció: Az IP cím (az IPv4 protokoll szerint) nem más, mint egy 32 bites (nemnegatív, egész) szám.

$2^{32} = 4\,294\,967\,296$, azaz bő négymilliárd különböző IP cím képzelhető el ennek alapján (látni fogjuk, az összes szám a speciális jelentésű számok és egyéb okok miatt nem használható). Egy ilyen nagyságrendű szám nagyon nagy ahhoz, hogy így észben tartsuk, ezért az IP címeket 8-bites csoportokba rendezve, ponttal elválasztva, és a megszokott 10-es számrendszerben szokás megadni, például:

173.194.35.184

Ezt azért már könnyebb megjegyezni, különösen, ha értjük a mögöttes logikát is. Ehhez azonban a kettes számrendszerben kell számolnunk. Váltsuk vissza a fenti IP-címet bináris alakra, az áttekinthetőség miatt benne hagyjuk a 8 bitenként elhelyezett pontokat:

10101101.11000010.00100011.10111000

Ez a $2\,915\,181\,496$ decimális szám egyébként, de az IP címeket ebben a formában soha sem használjuk, hiszen megjegyezhetetlen és ráadásul semmilyen plusz információ (ahogy látni fogjuk) nem olvasható ki belőle közvetlenül.

Aktivitás: Elevenítsük fel magunkban a számrendszerek kezelését, különös tekintettel a bináris (kettes) számrendszerre. A következők megértéséhez ugyanis szükséges ezek ismerete.

IP-címmel lehet, és tudni is kell azonosítani egy hálózat (például az internet) összes csomópontját (számítógépét és egyéb eszközét). Az *internetre* általában nem egyedi számítógépek csatlakoznak, hanem számítógép-hálózatok. Ezért is mondják az internetre, hogy *hálózatok hálózata*. Ebben a többszörösen összetett rendszerben az azonosítás logikája, hogy először csatlakozó hálózatot azonosítjuk, és azon belül azonosíthatók a csomópontok.

24.18. táblázat. *IP Címosztályok*

Címosztály	Hálózat cím	Host cím	Első bitek	Címtartomány	Maszk
A	1 bájt	3 byte	0...	1.0.0.0 - 126.255.255.255	255.0.0.0 vagy /8
B	2 bájt	2 byte	10...	128.0.0.0 - 191.255.255.255	255.255.0.0 vagy /16
C	3 bájt	1 byte	110...	192.0.0.0 - 223.255.255.255	255.255.255.0 vagy /24

A *címosztály* azt mondja meg, hogy a rendelkezésre álló 32 bitből hány bit azonosít egy internetre csatlakozó hálózatot, és hány bit használható a hálózaton belüli gépek azonosítására. A 24.18. táblázaton jól látható, miért praktikus az IP-cím 8-as csoportosítása:

Öt címosztály van, de D és E osztályok speciális célokra vannak fenntartva, ezért ezeket kihagytuk a táblázatból. (Néha még ötnél több címosztályt is megkülönböztetnek - gyakorlati jelentősége számunkra az első háromnak van.)

A 24.18. táblázatból leolvasható, hogy a 3 általánosan használt címosztály esetén a 4-ből rendre 1, 2 és 3 bájtot foglal el a hálózat címe. A maradék bitek használhatók a hostok hálózaton belüli azonosítására, ami így 3, 2 illetve 1 bájtnak adódik. Az, hogy egy címről bármikor tudni lehessen, hogy melyik címosztályba tartozik, az egyes címosztályokba tartozó IP címek megállapodás szerinti bitekkel kezdődnek – ezt is leolvashatjuk az ábráról. Ezek alapján a használható címtartományt decimális alakban magunk is könnyen felírhatjuk.

A 24.18. táblázat még az ún. alapértelmezett (default) alhálózati maszkot mutatja az egyes címosztályok esetén. A maszk egy olyan, szintén 32 bites szám, ahol a hálózat azonosítójának bitjein mindenhol 1 áll, a host-bitek helyén pedig 0. (A 8 darab 1-es bitből álló szám a 255, ahogy szokásos módon 4 darab decimális számként írjuk le a maszkot. A maszkot másik módon is szokás megadni: például egy A osztályú IP cím után írt /8 azt mutatja, hogy a hálózatot 8 biten azonosítjuk.

Nézzünk egy példát: a fent már említett 173.194.35.184 cím bináris formában a következő:

10101101.11000010.00100011.10111000

Az első biteket vizsgálva egyértelmű, hogy ez csak B osztályú cím lehet, ami azt jelenti, hogy 2 bájton tároljuk a hálózat azonosítóját és kettőn a hálózaton belül a host azonosítóját:

Hálózat azonosító: 10101101.11000010
Gép azonosító: 00100011.10111000

Bár a hálózat címe csak a 4 bájtos IP cím első valahány (jelen esetben 2) bájtja, hálózat címeként erre helyiérték-helyesen, azaz a host-azonosító biteket nullával feltöltve kell hivatkozni (így a két cím nem is keverhető össze). Példánkban tehát a hálózat címe:

Hálózat címe : 10101101.11000010.00000000.00000000

Ugyanez a szokásos decimális formában:

173.194.0.0

Ha összevetjük ezt a felírási módot az eredeti IP címmel, akkor láthatjuk, miért praktikus az IP cím 4 darab decimális számmal való felírása. Az alhálózati maszk, amint említettük, arra való, hogy megmutassa, hogyan osztjuk el a 32 bitet a hálózat és a host címe között. Segítségével egyszerű bitenkénti logikai művelettel (ÉS – AND) is megkapható a hálózat címe:

10101101.11000010.00100011.10111000 AND
11111111.11111111.00000000.00000000 =
10101101.11000010.00000000.00000000

A B címosztályban a hálózatot azonosító részben a 16 bitből az első kettő mindig ugyanaz. A maradék $2^{14} = 16\,384$ féleképpen tartalmazhat 1-et és 0-t. Ebből a csupa 0 és csupa 1-es biteket tartalmazót leszámítva a maradékkal 16382 hálózat azonosítható. Egy-egy hálózatban a második két bájt, 16 bit azonosítja az egyes gépeket. Az azonosításra $2^{16} = 65\,536$ féle bitsorozat lehetséges, hasonló okokból kettővel kevesebb, azaz 65534 számítógép azonosítható ezért maximum ennyi gép tartozhat egy B osztályú hálózatba. Ugyanígy a többi címosztályhoz is meghatározhatjuk, hány hálózat hozható bennük létre és hány gép tartozhat egy hálózatba, ezeket a 24.19. táblázatban foglaltuk össze.

Az összes címosztályt eleve fel sem soroltuk, és azt is megfigyelhetjük, hogy az említettek sem mindenhol

24.19. táblázat. *Hálózatok és hostok száma az egyes címosztályokban*

Címosztály	Hálózat cím	Host cím	Hálózatok száma	Gépek száma
A	1 bájt	3 bájt	126	16 777 214
B	2 bájt	2 bájt	16 382	65 534
C	3 bájt	1 bájt	2 097 150	254

követik folytonosan egymást. Ennek több oka is van, de vannak különböző célokra fenntartott speciális címek, címtartományok, amik közül néhányat mindenkinek ismernie kell. Ilyen például a 127.0.0.1 cím (angolul loopback), ami minden esetben a gép saját címét jelenti: ha a saját gépünk valamely hálózati funkcióját szeretnénk használni erre a címre hivatkozhatunk, nem kell tudnunk saját gépünk egyébként mindig létező, egyedi IP címét. A csupa egyes bitet tartalmazó gépcímek (pl. 255) nem egyedi gépet jelentenek, hanem csoportos üzenetküldésre (broadcast) valók. Ha bármely okból olyan hálózatot építünk, amely nem a nyilvános internet közvetlen része, erre a célra a 192.168.1.0 – 192.168.254.254 címtartomány használható, ezek a címek csak helyi hálózaton használhatók.

Az IP cím méretéből (32 bit) adódik, hogy eleve kb. 2^{32}, azaz 4 milliárd egyedi gép lehetséges a hálózaton, a szó szerinti értelemben vett számítógépek mellett beleértve a különböző egyéb hálózati eszközöket is. A hálózatok és hálózatonként a gépek maximális számából könnyen belátható, hogy az IPv4 cím készlet véges. Ebben a rendszerben már eleve csak kb. minden második embernek jutna hálózati számítógép, emellett gondoljunk arra, hogy mi magunk hány IP címet használunk (notebook, netbook, asztali gép, okostelefon, otthoni médiaszerver, biztonsági IP kamera...)

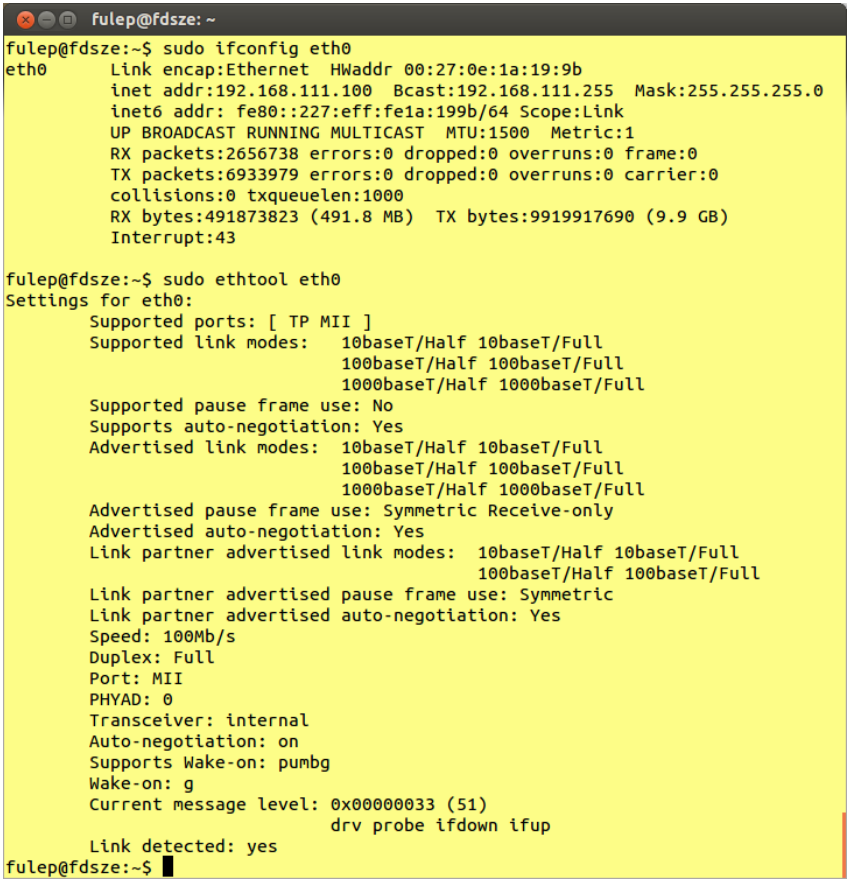
Az IPv4 mai állapotában is vannak még lehetőségek a spórolásra (enélkül már most sem lenne elég a címtartomány). A legfontosabb, hogy a helyi hálózatokon használható a már fentebb említett lokális (általában 192.168.1 hálózati azonosítójú) címtartomány, amivel az egyes háztartások, kisebb cégek akár egyetlen valódi, publikus IP cím birtoklása esetén is tudnak mögöttes hálózatot fenntartani. Ehhez egyetlen eszközre van szükség, ami megkapja ezt a publikus IP címet *és* kap egy másik, belső hálózati IP címet is. Az eszköznek a

gateway vagy másik nevén *router* funkciót kell ellátnia, amit egy hasonló nevű hálózati eszközzel, vagy akár egy két hálózati kártyával rendelkező Linux számítógéppel valósíthatunk meg. Ha a belső hálózatról bárki az internet felé szeretne kommunikálni, ez a két IP címmel rendelkező eszköz kifelé úgy kommunikál, mintha valójában ő volna a kommunikáló fél, az adatok forrása (hiszen az interneten érvényes IP cím birtokában valóban csak ő képes erre), a viszontválaszt pedig továbbítja a belső hálózaton működő eredeti gépnek. Mindkét irányba úgy továbbítja tehát a csomagot, mintha azok forrása ő maga lenne. Ehhez természetesen a csomagok fejléceit menet közben módosítgatni kell. A kapcsolódó technikákat (IP masquerading (álcázás), illetve NAT (Network Address Translation – hálózati címfordítás) néven említik.

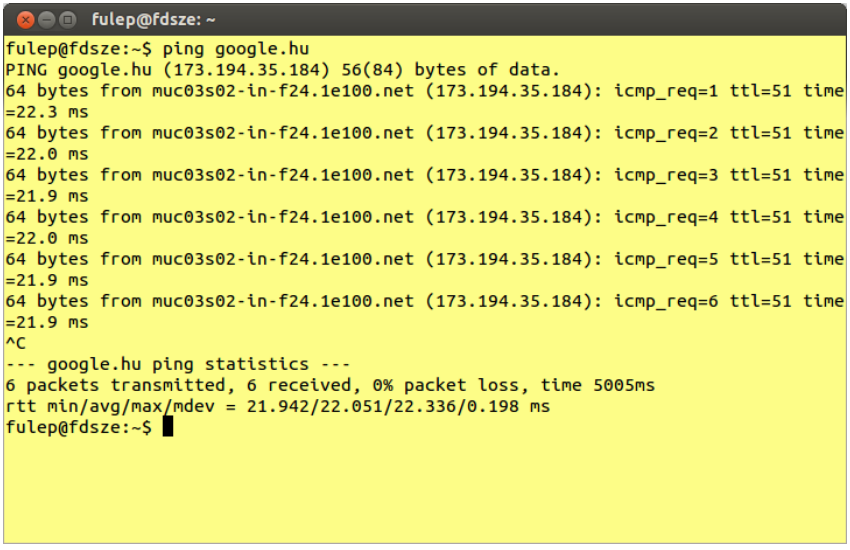
Az interneten tehát egyre problémásabb a jelenlegi IP címrendszer fenntartása. A végleges, és folyamatban lévő megoldás a jelenleg elterjedt IPv4 rendszer cseréje lesz. Az új szabvány neve az IPv6, amiben a 32 bites címtartomány 128 bitesre történő bővítése minden elképzelhető igényt bőven kielégít, és emellett a mai elképzeléseink szerinti hálózati igények számára is elegendő. (A mai eszközök mellett már nincs olyan jelentősége a bitekkel való spórolásnak, mint néhány évtizede volt, és az is sokkal jobban látszik, hogy milyen igényekre kell a jövő internetjét felkészíteni.) Az átállás az új szabványra *hosszú évek óta* elkezdődött, mégpedig az átlag felhasználók legkisebb zavarása nélkül: gyakorlatilag minden új eszköz, új szoftver (operációs rendszer) támogatja az IPv6-ot is, és általában inaktív állapotban várja, hogy egy majdan felgyorsított ütemű átállás esetén könnyen, gyorsan beüzemelhető legyen.

Aktivitás: Kérdezzük le a gépünk hálózati beállításait! Linuxon az ifconfig, Windowson az ipconfig paranccsal tehetjük ezt meg legegyszerűbben. A programok működéséről az operációs rendszer dokumentációjában találunk információt. Figyeljük meg, hogy a gépünk fel van-e már készítve az IPv6 használatára! (Lásd 24.1. ábra)

Aktivitás: Ellenőrizzük hálózati kapcsolatunk működőképességét oly módon, hogy azt ellenőrizzük, elérhető-e a google.hu szerver. Ismerjük meg és használjuk a ping és traceroute parancsokat. Ezek a programok hasonló módon használhatók és elérhetők a közismert PC-s operációs rendszereken. (Lásd 24.2. ábra)



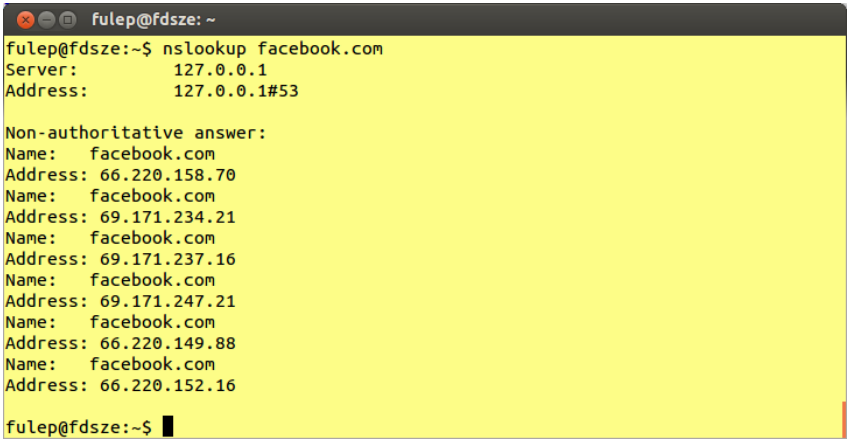
24.1. ábra. *Az IP konfiguráció és az egyik hálózati csatoló beállításainak lekérdezése Linux rendszeren*



24.2. ábra. *A ping parancs kimenete*

A felhasználók a számokból felépített IP-címeket általában nehezen jegyzik meg. Emiatt könnyebben megjegyezhető a „beszédes” neveket is rendelni lehet az IP-címmel azonosított eszközökhöz. A nevek kialakítására egy hierarchikus rendszerben van lehetőség. A név-IP-cím hozzárendelést az úgynevezett DNS (Domain Name Service – névszolgáltatás) kiszolgálók végzik. (A megbízhatóság növelésére egy-egy területen több DNS kiszolgálót is üzemeltethetnek.) A egy adott névszerver egy névterület (domain) neveit kezelik, más területekhez tartozó nevek elérését biztosítandó hierarchikus kapcsolatban állnak egymással. A névszolgáltatás hierarchikus névadást teszt lehetővé, és a névhez tartozó IP-cím megkeresése is ezen hierarchia mentén történik. A név tagjait pontokkal választjuk el egymástól.

Annak kiderítésére, hogy egy adott névhez milyen IP-címek tartoznak, az operációs rendszerekben megfelelő segédprogramokat találunk.

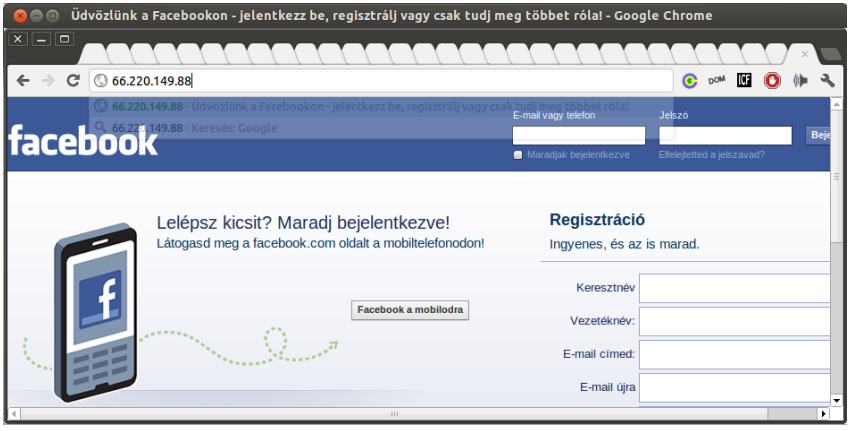


24.3. ábra. *Egy domain név mögött rejlő IP cím kiderítése*

A nevek használata nem kötelező, az IP-címek is bármikor használhatók.

Ha valamely szolgáltatónak nagy ügyfélköre van, vagy valamely szolgáltatást nagy tömeg vesz igénybe, akkor a biztonságos és gyors kiszolgálás érdekében több számítógépszerver látja el a feladatot. Ilyen esetben ugyan egy névvel, de több hozzá tartozó IP-címmel kell dolgozni. Hogy egy felhasználót éppen melyik szerver fogja a sok közül kiszolgálni, a DNS szerver dönti el.

Aktivitás: Derítsük ki, hogy a facebook.com néven elérhető szervernek mi lehet az IP-címe. Egy ilyen forgalmú szolgáltatást az egész világ számára már nem egyetlen számítógép biztosít. A terhelés elosztására emiatt több IP-címet is találhatunk. Próbáljuk ki a böngészőnkben az egyik IP-cím használatát! (Lásd 24.3. és 24.4. ábrák)



24.4. ábra. *A könnyen megjegyezhető domain nevek helyett IP címek is használhatók*

25. Építsünk otthon hálózatot!

Manapság már a háztartások bármelyikének lehet internet előfizetése és néhány számítógépe is vagy hasonló, hálózatra kapcsolható eszköze (például okostelefon, tablet, hálózati háttértár, vezeték nélküli hálózati nyomtató, IP kamera, youtube-képes és böngészővel felszerelt ún. smart (okos) TV stb.).

A hálózatok kialakítására is természetesen megvannak a szakemberek, de aki egyetemen informatikát tanult, annak egyszerűbb eseteket meg kell tudnia egyedül oldania.

Tekintsük át először, mi történik, amikor előfizetünk „az internetre”. Az internet használata alapvetően ingyenes. Nem kell fizetnünk egy weboldal megtekintéséért, vagy egy email elküldéséért. Akkor mégis, miért fizetünk?

A válasz nagyon egyszerűen megadható, ha végiggondoljuk, hogy mi is az internet. (Emlékeztetőül: hálózatok hálózata.) Ha ebbe a rendszerbe a saját eszközeinket – hálózatunkat – be szeretnénk kapcsolni, akkor egy kapcsolódási pontra van szükség. Ez a kapcsolódási pont valamelyik, már az internetre kapcsolódott felhasználó

rendszerében lesz megtalálható, és ezt kell a mi hálózatunk számára valamilyen módszerrel elérhetővé tenni. Hát ezt és ennek a kapcsolatnak az üzemeltetését kell megfizetnünk, és még esetleg annak ránk eső részét is, amit a kapcsolódó hálózat üzemeltetője fizet azért, hogy neki is van valahol egy kapcsolódási pontja.

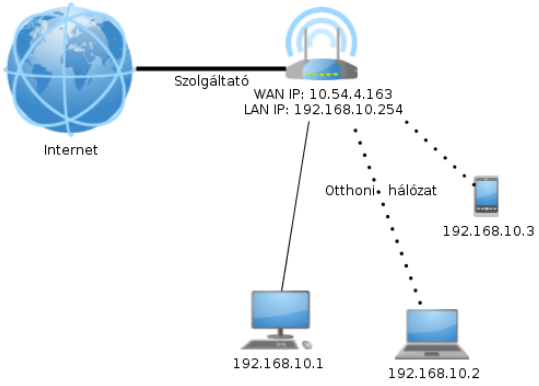
Az internetes kapcsolatért általában egy belépési díjat és havi üzemeltetési díjat kell fizetni. Az internet népszerűvé válása óta számtalan szolgáltató jött létre csak annak biztosítására, hogy a világ bármely részén legyen megvásárolható kapcsolódási pont, és ezeken 24 órás felügyeletet biztosítva biztonsággal szolgálják ki a rendszerükhöz csatlakozókat. Ezeket a cégeket nevezzük internet-szolgáltatóknak.

Kapcsolódási pontot a szolgáltató a csatlakozni kívánó telephelyére telepítve ad bérbe. Ez fizikailag azt jelenti, hogy kapunk egy aktív hálózati elemet, amihez kiépíti a szolgáltató a saját hálózatához való csatlakozást is. (Ez sokféle lehet; a telefonvonaltól a rádióhullámig bármi, ami átvitelre alkalmas.) Ez az aktív elem – attól függően, hogy mennyire vagyunk nagyfelhasználók – vagy fix, vagy dinamikus és publikus IP-címet kap. A dinamikus IP-cím azt jelenti, hogy az internethez történő csatlakozáskor, a kapcsolat idejére kapjuk csak meg az adott IP címet, a kapcsolat bontása után, egy későbbi csatlakozáskor nem biztos, hogy ugyanazt a címet osztja ki nekünk a szolgáltató.

A hálózati elemhez kapcsolhatjuk a saját hálózatunkat, a legegyszerűbb esetben egy számítógépet.

Ahhoz, hogy végül kapcsolódni tudjunk a hálózatra, több mindenre is szükség van: először is, a számítógépünk szabványos hálózati csatolójához kell illeszteni a szolgáltató hálózatát – mivel a hálózat fizikailag sokféle megvalósítású lehet (például rádiójelekkel, telefonvonalon, kábeltévé-hálózaton érkezik) ezért az ehhez való eszközt (modemet) a szolgáltató szokta szállítani és rendelkezésre bocsátani. A modemtől a számítógépünkig terjedő rész kiépítése már az előfizető feladata, de egy gép erejéig a szolgáltatók még ebben is szoktak segíteni és sok esetben olyan eszközt szállítanak, ami nem csak modemként funkcionál hanem egy néhány gépes hálózatot minden tekintetben kiszolgál (több gép csatlakoztatási lehetősége, vezeték nélküli hálózat, biztonsági funkciók stb.). Csatlakoztatás után pedig kapunk egy IP-címet, ami egy időre, csatlakozás idejére a saját rendelkezésünkre áll.

Időzzünk el kicsit azon, hogy milyen lehetőségeink vannak egy saját hálózat kialakításakor. Megtehetnénk, hogy a magánhálózatunk minden csomópontjának szerzünk egy publikus IP címet, ilyenkor minden gép azonnal az



25.1. ábra. *Otthoni hálózat logikai felépítése*

internet teljes értékű tagjává válna – ez nem reális lehetőség, és épp az imént tárgyaltuk, hogy a szolgáltatónktól egyetlen ilyen címet kaptunk. Egygépes „hálózatot” természetesen kialakíthatunk: egyetlen géppel közvetlenül kapcsolódhatunk az internetre. Második lehetőségként nem kapcsolódunk az internetre, attól leválasztott hálózatot tervezünk: ebben az esetben valójában semmihez és senkihez nem kell alkalmazkodnunk, bármilyen címtartományt használhatunk, sőt, valójában az internet protokolljai helyett bármilyen egyéb hálózatot is építhetünk. Ez a lehetőség is inkább elméleti jelentőségű csupán, mivel ebben az esetben nem lesz internet kijárásunk (kapcsolatunk) az internet felé. A harmadik lehetőséggel foglalkozunk a továbbiakban. Saját hálózatunkat úgy alakítjuk ki, hogy – a szolgáltatótól kapott egyetlen IP címen kívül – nincs szükségünk publikus IP címekre, de mégis minden otthoni gépünk az internet részévé fog tudni válni.

A 25.1. ábra bemutatja, miképpen tehetjük eszközeinket az internet részévé. A szolgáltató általa biztosított csatolóelem legalább egy csatlakozó porttal rendelkezik. Ehhez csatlakoztatni leget egy útválasztót, routert. Ha esetleg a szolgáltató csatolóelemének is van útválasztó szolgáltatása, akkor saját routerre egyszerű esetben nincs is szükség.

Az útválasztó eszközhöz egyszerre legalább két hálózat csatlakoztatható. A legfőbb feladata hogy az érkező hálózati csomagokat mindig a megfelelő hálózathoz irányítsa.

Ennek az eszköznek a két hálózathoz való csatlakozás végett két IP címe is van: egy „igazi”, ami az interneten egyedi, segítségével csatlakozik eszközünk az internetre és ezzel érhetnek el bennünket bárhonnan a nagyvilágból, ha úgy akarjuk. Az eszköz képes lesz arra, hogy egy másik, belső hálózatot is kialakítson, amibe a saját eszközeink fognak beletartozni. Ennek a hálózatnak a címtartománya, mint azt a 24. fejezetben láttuk, 192.168.1.0 – 192.168.255.0 lehet.

A 25.2. ábrán látható egy router. Minden útválasztó úgynevezett címfordítóként (NAT) is működik. A belső gépekről érkező csomagokat az internetre továbbítás előtt úgy módosítja, hogy azok feladójaként saját magát, azaz saját külső, nyilvános IP-címét tünteti fel, így az azokra érkező válaszcsomagok is hozzá kerülnek majd. Mielőtt továbbítaná az internetről érkezett válaszcsomagot, a saját címét lecseréli a valódi, a belső címzett IP-címére, és a belső hálózaton lévő eredeti feladó, azaz a válasz címzettjének részére ad át. A belső hálózat bármely gépe is kommunikál a külvilággal, a távoli gép úgy fogja érzékelni – és fizikailag így is van –, mintha a publikus IP-címmel rendelkező routerrel állna kapcsolatban.

Aktivitás: Olvassuk le a 25.1. ábráról az otthoni hálózat IP címét.

A routerünk, ami a 25.2. ábrán láthatóhoz hasonlóan néz ki, így címfordítóként (NAT) is kell, hogy működjön: a készüléknek két IP címe van, az egyik a külső (WAN), a másik a belső (LAN) IP címe. A NAT alapötlete onnan ered, hogy a jelenleg használt IP címtartomány véges, így nagy pazarlás lenne akár az otthoni eszközeink számára is eszközönként egy-egy publikus IP címet elhasználni. A NAT egy olyan, sok otthoni hálózati eszköz által is biztosított szolgáltatás, mely lehetővé teszi a belső hálózatra kapcsolt gépeink közvetlen kommunikációját külső gépekkel anélkül, hogy saját nyilvános IP címmel kellene rendelkezniük. A belső gépekről érkező csomagokat az internetre továbbítás előtt úgy módosítja, hogy azok feladójaként saját magát (azaz saját külső, nyilvános IP címét) tünteti fel, így az azokra érkező válaszcsomagok is hozzá kerülnek majd továbbításra. Ilyen esetben a válaszcsomag címzettjének a címét a belső hálózati megfelelő IP címre módosítja, és a belső hálózaton elhelyezkedő eredeti feladó, azaz a válasz címzettjének részére ad át. A belső hálózat bármely gépe is kommunikál a külvilággal, a távoli gép úgy fogja érzékelni (és fizikailag így is van), mintha a

25.2. ábra. *Linksys gyártmányú router otthoni célokra*

routerünk külső IP címének megfelelő géppel állna kapcsolatban.

Egy komolyabb router igen szerteágazó szolgáltatással, és ehhez kapcsolódóan sok beállítási lehetőséggel rendelkezik. Az itt tárgyalt, kis irodába vagy otthonra való egyszerűbb routerek beállítása egy, a router által szolgáltatott webes felületen történhet. A routeren fut egy webkiszolgáló (webszervert), amire valamelyik böngésző segítségével névvel, jelszóval belépve menürendszerben elvégezhetjük a szükséges beállításokat. Egyszerűbb változtatások az OK gomb lenyomásakor azonnal életbe lépnek, bizonyos esetekben azonban újra kell indítani az útválasztót: ilyenkor az eszköz figyelmezteti a felhasználót, hogy az újraindulás idejére (10-20 másodperc) megszakad a hálózati kapcsolat, ennek letelte után tudunk újra elérni az internetet.

A beállítási folyamat úgy indul, hogy elindítunk egy böngészőt, ennek címsorába megadjuk a router belső IP-címét. A külsőt a szolgáltatótól kapja dinamikusan (azaz változhat), de általában biztonsági okokból „kívülről”, azaz az internet felől nem is elérhető a webes felület. A belső hálózat IP címe pedig tipikusan 192.168.1.0, rajta a router címe 192.168.1.1, vagy valami hasonló szokott lenni. A kezdeti beállítást általában ráírják a router aljára a belépési névvel és jelszóval együtt, de a kézikönyvben biztosan benne van.

A gyártók sok esetben nem egyedi jelszavakat generálnak, hanem minden (!) eszközükbe ugyanaz van alapértelmezetten beállítva. Leggyakoribb az admin-admin név-jelszó páros, a többi változat is hasonlóan bonyolult. Az ilyen jelszavakat haladéktalanul cseréljük le egy megfelelő védelmet nyújtóra. Általunk nem ismert, például dokumentáció nélkül kapott használt eszközök esetén az internet jöhet segítségünkre, és például a http://www.routerpasswords.com címen nagyon sok eszköz hozzáférési adatait összegyűjtve megtaláljuk. Egyes gyártók újabb modelljei egyedi, megfelelő bonyolultságú jelszavakkal érkeznek, ezeket az interneten nem, hanem csak az eszközre írva találjuk meg.

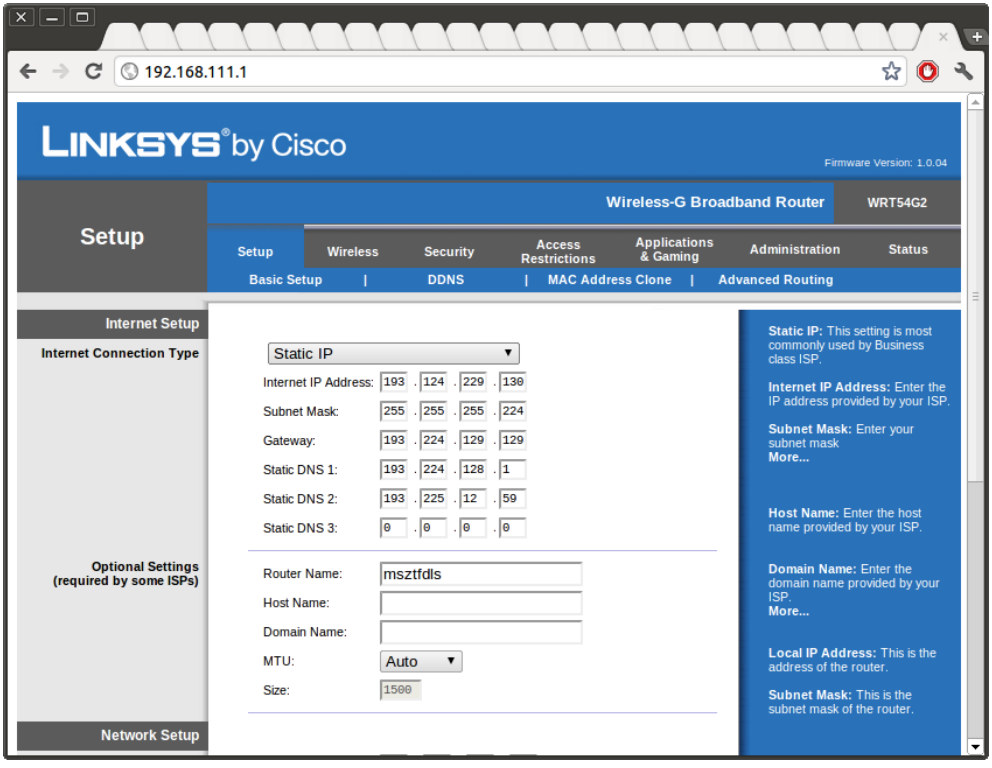
Ha a korábban beállított jelszót végképp elvesztettük, a készülékek rendelkeznek egy (általában csak hegyes tárggyal megnyomható) reset gombbal, amivel minden beállítás a gyári alapértékekre áll vissza.

A webes felületre példát a 25.3. ábrán láthatunk. A felület nyelve az esetek döntő többségében angol, nem jellemző, hogy a nyelvet át lehetne állítani. Ezért is igyekszünk minden kapcsolódó kifejezést angolul is leírni.

Aktivitás: A 25.3. ábráról olvassuk le a router külső és belső IP címét!

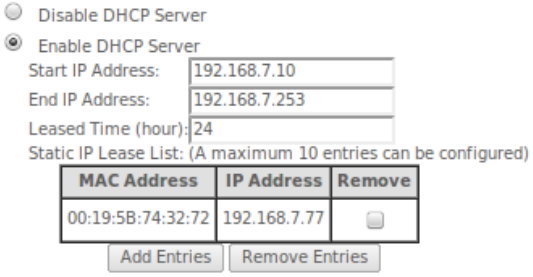
A továbbiakban áttekintjük a routerek legfontosabb funkcióit.

Az otthoni hálózat is csak akkor lesz működőképes, ha minden eszközünk rendelkezni fog egy egyedi IP címmel. Ezt megvalósítani két módon lehetséges. Az első triviális: minden gépre egy egyedi IP címet állítunk be kézzel. Ez minden egyes új eszköz csatlakoztatásakor egyre kényelmetlenebbé válik. A másik, a megfelelő megoldás a DHCP (Dynamic Host Configuration Protocol – dinamikus számítógép-beállító protokoll) használata. A routerünkben egy DHCP szerver is van, ami kérésre mindenkinek biztosít egy egyedi IP címet az előre beállított szabályok alapján. A szabályok nagy hálózaton sokfélék lehetnek, de a mi esetünkben nagyon egyszerűek: egy előre megadott tartományból a DHCP szerver mindig a soron következő szabad címet osztja ki, illetve az is megoldható, ha azt szeretnénk, hogy valamely gép mindig ugyanazt a címet kapja (MAC cím alapján minden hálózati eszköz felismerhető, egy listát állíthatunk össze, hogy az adott MAC című eszköz milyen IP címet kapjon). A 25.4. ábrán erre láthatunk példát egy egyszerű router webes felületén: a DHCP szerver a 192.168.7.10 és 192.168.7.253 közötti címeket osztja, kivéve a megadott MAC című gépnek, ami minden esetben a 192.168.7.77 címet fogja megkapni.



25.3. ábra. *Egy router webes felülete*

A MAC címek használatának biztonsági vonatkozása is van. A routerek lehetőséget biztosítanak arra, hogy ún. *MAC szűrést (MAC filter)* használjunk: ezzel a biztonságot növelhetjük oly módon, hogy a saját eszközeink MAC címeit listába vesszük, és azt mondjuk, hogy a hálózatunkra csak ezen a listán szereplő eszközöknek engedjük a helyi hálózathoz való hozzáférést. Vezetékes hálózatnál ennek nincs jelentősége, de vezeték nélküli



25.4. ábra. *DHCP beállítások a webes felületen*

csatlakozásnál egy bizonyos védelmet jelent (a szomszéd ellen). Kis kényelmetlenség, hogy első alkalommal a saját új eszközeink, vagy hozzánk látogató barátunk okostelefonját is engedélyezni kell, ha használni akarja a hálózatunkat. Mivel – az eredeti koncepcióval ellentétben – sok eszköz szoftveresen meg tudja változtatni a MAC címét, ez a módszer komoly védelmet nem jelent, de a biztonságnál minden kis lépés közelebb visz az ideális állapothoz.

A vezetékes irányból a biztonságot a fizikai hozzáférés korlátozása biztosítja leginkább, illetve az, hogy amint említettük, a külső hálózat (WAN) irányából a router konfigurációs felülete alapesetben nem elérhető. A belső hálózatot emellett ún. *tűzfal (firewall)* is védi: ez azt jelenti, hogy a forgalom elemzésével az eszközünk kizárólag az előre engedélyezett forgalmat engedi át magán. Hiába van valamilyen szoftveres biztonsági rés a gépünkön (például nagyon gyenge jelszót állítottunk be, esetleg teljesen kiiktattuk a jelszavas védelmet), ha az internet irányából bizonyos hozzáférési módokat a tűzfal mindenképpen tilt (blokkol). Emellett hasonló beállításokat tehetünk szoftveres tűzfallal a saját számítógépeinken is, az egy másik védvonalat jelenthet.

A tűzfalak részletes konfigurálása speciális szakértelmet igényel, ezért nem is foglalkozunk vele, de a tűzfalat az alapbeállításokkal mindenképpen kapcsoljuk be: ez mindent meg fog engedni, ami az általános használathoz szükséges (web, email, skype, stb.), de az internet irányából érkező, ezektől eltérő, nem szokványos kéréseket támadásoknak minősíti és egyszerűen elnyeli. Ne keverjük a tűzfal funkcióját a víruskeresőével: a tűzfal

bizonyos szolgáltatásokat engedélyez, de az átküldött adatok vizsgálata nem tartozik a hatáskörébe, például beengedi az emailt, de nem figyeli, hogy vírusos fájlt kaptunk-e. A routerek víruskereséssel nem foglalkoznak.

A biztonsággal a most említettek mellett leginkább a vezeték nélküli hálózatok esetében kell kiemelten foglalkoznunk. A vezeték nélküli (WLAN) hálózat határa ugyanis nem a szoba, iroda falai közé esik: antennatípustól és az épület sajátságaitól függően 10-20-100 méter sugarú *gömbszerű*, térbeli alakzatnak képzeljük inkább, ahol a biztonsági szabályok betartása nélkül biztosak lehetünk benne, hogy valaki – ártó szándékkal, vagy csupán az „ingyen” internetcsatlakozást kihasználva – előbb vagy utóbb rácsatlakozik.

A vezeték nélküli hálózat védelme több elemből áll: a legegyszerűbb (és legkevésbé hatásos) módszer, hogy titokban tartjuk a hálózat nevét. A hálózat nevének ismerete nélkül ugyanis nem lehet az adott hálózathoz csatlakozni. Ha a számítógépen az elérhető vezeték nélküli hálózatokat keressük, azokról a hálózatokról kapunk listát, amik időről időre sugározzák saját azonosítójukat (nevüket), hogy épp ezek a programok megtalálhassák őket. Ha ezt nem akarjuk, a routerben letilthatjuk a név kisugárzását (SSID Broadcast). Ezután a hálózatunk „láthatatlan” lesz, aki tudja a nevét, mert megmondjuk neki, természetesen ugyanúgy csatlakozhat hozzá.

A vezeték nélküli hálózat legfontosabb védelmét a titkosítási algoritmusok jelentik, ami azt jelenti, hogy a kommunikáció titkosítva történik, meghiúsítva, vagy legalábbis megnehezítve a forgalom lehallgatását. A részletek ismertetése nélkül javasoljuk, hogy a WPA2-PSK (Wifi Protected Access Version 2, Pre-Shared Key) protokollt használjuk, minden modern eszköz ismeri – ez sem sebezhetetlen, de esetünkben a legjobb biztonsági szintet jelenti – sokkal jobbat, mint a korábbi rendszerek (például, WEP, WPA korábbi verziója).

A vezeték nélküli hálózat jelszavát nem kell minden csatlakozáskor begépelnünk, azt megjegyzi a számítógép a csatlakozási paraméterek között. Ezért különösen igaz, hogy érdemes *jó* jelszót használni: az 12345 helyett biztosan nehezebb lesz a HJÖÖ45D73&@##67FkKkK3j karaktersorozatot visszafejtenie a betolakodónak.

A vezeték nélküli biztonsági beállítások konfigurálása közben, ha nem figyelünk, megeshet, hogy „kizárjuk” magunkat a saját routerünk által biztosított vezeték nélküli hálózatból – ilyenkor nem kell kétségbe esni, hiszen ezek a beállítások a vezetékes hozzáférésre nem vonatkoznak. Ha vezeték nélküli kapcsolaton konfiguráljuk a routerünket, mindig tartsunk magunknál kábelt is, amivel szükség esetén bármikor elérhetjük a rendszerünket.

A konfiguráció teljes tönkretétele esetén a router reset gombjának segítségével visszaállíthatjuk az eredeti gyári állapotot.

A vezeték nélküli hálózat biztonságát két további módon fokozhatjuk: a fizikai hozzáférés korlátozásával. Egyrészt sok eszközön beállítható, hogy a nap mely időszakában legyen aktív a vezeték nélküli hálózat: például előre tudhatjuk, hogy éjféltől reggel 6-ig biztosan nem használjuk sosem, ezért ebben az időszakban az eszköz automatikusan lekapcsolja a vezeték nélküli hozzáférést. (Ez arra az esetre is megoldást jelenthet, ha csökkenteni akarjuk a sugárzó antennák esetleges egészségkárosító hatását.) Olyan routerek is léteznek, amin egy egyszerű gombbal bármikor ki-be kapcsolható a vezeték nélküli hálózat, ha például kikapcsoltuk a notebook-unkat, vagy épp elindulunk otthonról, egy mozdulattal biztosan kizárhatjuk az illegális felhasználás lehetőségét – miközben egy kevés áramot is megtakarítunk.

A fizikai korlátozás (és energia-takarékosság) másik módja, hogy csak a szükséges sugárzási energiával használjuk az antennáinkat: egyrészt nem vásárolunk „túl erőset”, túl nagy területet besugárzót, másrészt sok eszköz lehetőséget biztosít a sugárzás szabályozására: ha a tízes skálán 10-es értékkel két házzal arrébb is jól fogható a jelünk, akkor feleslegesen erősen sugároz az antennánk. Ezt az értéket csökkentve elérhetjük, hogy sokkal alacsonyabb sugárzási szinten is megbízhatóan működjön a hálózatunk azon az 1-2 szobányi területen, ahol arra szükségünk van.

Önellenőrzés

1. Sorolja fel, milyen feladatokat végez az operációs rendszer!
2. Ön az operációs rendszerek mely szolgáltatásait használja?
3. Az Ön által épp használt operációs rendszeren tekintse meg, milyen folyamatok futnak, és ezek közül melyik folyamat használja legtöbbet a processzort!
4. Soroljon fel néhány olyan tulajdonságot, ami megkülönböztet egy Windows és Linux gépet egymástól!
5. Készítsen listát azon futó folyamatokról, amiket Ön indított, listát tegye el egy feladat.txt állományba.
6. Gondoskodjon arról, hogy az imént készített állomány tartalmát mások ne olvashassák el.
7. Mi jellemző a parancssori interfészre?
   - ☐ Olyan interfész, amit gyakorlatilag minden operációs rendszer biztosít, de jellemzően a haladó felhasználók dolgoznak vele.
   - ☐ A kommunikáció jellemzően parancsok begépeléséből és a rá adott válaszok elolvasásából áll
   - ☐ Jellemzően sokkal alacsonyabb hardvererőforrás-igényű, mint a grafikus felületek
   - ☐ Gyakorlatilag minden operációs rendszer része, mert hatékonyabb és teljesebb, mint a grafikus felhasználói felületek
   - ☐ Elavult, régi interfész, használatát a profik mellőzik
   - ☐ Napjaink legismertebb felhasználói felülete, ami elsősorban a mobil eszközökön jellemző
   - ☐ Részletekbe menő megoldásokra sok esetben alkalmatlan
8. Mi az oka a hálózati funkciók rétegekbe szervezésének?
   - ☐ A funkciók elkülönítésével átláthatóbb a bonyolult rendszer működése.
   - ☐ Az egyes rétegek szoftveres megoldásai elkülönülnek, akár más gyártótól is származhatnak.
   - ☐ Történelmi okokról beszélhetünk, technikai indoka nincsen.

- [ ] A rétegekbe szervezett funkciók minden esetben olcsóbbak.

9. Egyedül élő személynek milyen célból lehet szüksége saját, otthoni hálózatra?
10. Mik a hálózatok osztályozásának legfőbb, megismert szempontjai?
11. Sorolja fel, Ön milyen célokból használ számítógépes hálózatot!
12. Otthoni hálózatán, amely 3 számítógépből áll, az egyik gép nem tud a WLAN hálózathoz csatlakozni, a másik kettő üzemel. Melyik tevékenységeket nem szükséges elvégeznie?
    - [ ] Ellenőrzöm a nem működő gép hálózati csatolókártyáját
    - [ ] Ellenőrzöm a nem működő gép hálózati beállításait
    - [ ] Ellenőrzöm az otthoni internetmegosztó eszköz beállításait
    - [ ] Az ügyfélszolgálaton bejelentem a hibát.
    - [ ] Ellenőrzöm, hogy mindhárom gépen azonos verziójú víruskereső fut-e.
13. Adott egy gép: 192.168.96.101, az alapértelmezett hálózati maszk 255.255.255.0. Határozza meg, mi a hálózat címe! (192.168.96.0) Kiadhatók-e a következő IP címek egy számítógépnek ezen a hálózaton:
    - [ ] 192.168.96.1
    - [ ] 192.168.96.255
    - [ ] 192.168.96.254
    - [ ] 192.168.96.0
14. Válassza ki, mik lehetnek valós IPv4 IP címek:
    - [ ] 133.43.65.87
    - [ ] 192.168.343.88
    - [ ] 1024.43.256.3
    - [ ] 246.512.256.3
    - [ ] 0.0.0.256
    - [ ] 256.256.256.0

15. Jelölje meg, hogy mely funkciók jellemzőek egy otthoni használatra szánt routerre:
- [ ] Tűzfal
- [ ] DHCP szerver
- [ ] DHCP kliens
- [ ] vezetéknélküli kapcsolódási pont
- [ ] email kliens
- [ ] áram nélküli működés
- [ ] Szünetmentes áramellátás biztosítása

16. Melyik az az alkatrész, ami biztosan nem hiányozhat egy hálózatba kötött számítógépből!
- [ ] Hálózati csatoló eszköz
- [ ] Processzor
- [ ] Hangkártya
- [ ] Tűzfal
- [ ] Memóriamodul

17. Ügyfélszolgálati munkahelyen pontosan miképpen tudná elképzelni virtuális gépek használatát?

IV. MODUL

Kiadványszerkesztés

Ez a modul a kiadványszerkesztés számítógépes rendszereinek használatával foglalkozik. Nem egy konkrét kiadványszerkesztő program ismertetése a cél, hanem azt szeretnénk elérni, hogy az olvasó a modult áttanulmányozása után képes legyen egy szakdolgozat, műszaki leírás vagy egy egyszerűbb kiadvány formai megtervezésére és valamilyen kiadványszerkesztőt megismerve képes legyen a megtervezett dokumentumot megszerkeszteni is.

Ezt úgy kívánjuk elérni, hogy megismertetjük az olvasót a kiadványok alapelemivel kapcsolatos fogalmakkal, megpróbálunk megfogalmazni olyan szabályokat, amiket a szerkesztés során célszerű betartani, azért, hogy a végeredmény esztétikailag megfelelő legyen, és egyben olyan dokumentum keletkezzen, ami szakmai – elsősorban nyomdai előkészítést értve ezalatt – szempontból is megfelelőnek lesz tekinthető.

Mindennek indokául azt tekintjük, hogy manapság – éppen a számítógépes térhódításnak köszönhetően – a hétköznapi életben is olyan dokumentumok előállítását követelik meg, amelyek külalakja mindenképpen túlmutat a ma már elavult írógéppel készített dokumentumok formai lehetőségein. Sőt egyre nagyobb az igény arra, hogy a már elkészült dokumentumok könnyen továbbfejleszthetők (bővíthetők, átszerkeszthetők, esetleg nyomdai úton sokszorosíthatók) legyenek.

10. LECKE

A kiadványszerkesztés alapelemei

Ebben a leckében a kiadványszerkesztés alapelemeivel ismerkedhetünk meg.

Az első fejezetet olvasmányként ajánljuk áttanulmányozni. Tartalmát megtanulni csak akkor szükséges, ha valakinek szándékában áll – vagy leendő munkaköre erre kényszeríti – nagyobb dokumentumokat gyakrabban létrehozni, ezek esztétikai kialakítását önállóan megtervezni. (Megjegyezzük, hogy egy szakdolgozat igényes elkészítése elegendő okot szolgáltathat e fejezet gondos áttanulmányozásához.)

A második fejezetben már azokat az ismereteket foglaltuk össze, amely mindenféle dokumentum előállításához szükségesek lehetnek. Nevezetesen itt beszélünk arról, hogy milyen lépéseket, munkafázisokat célszerű végigvinni ahhoz, hogy a munkák végeredménye mindenféle szempontból elfogadható legyen.

Megismerheti az olvasó ebben a fejezetben azt is, hogy mik azok a lehetőségek, amik a dokumentumok kialakítására alapesetben minden szerkesztőprogramban megtalálhatók.

A szolgáltatások használatáról viszonylag kevés szó esik, mivel sokféle programot használhatunk a munkavégzéshez. A konkrét leírásokban a Word különböző verzióinak illetve ritkábban az InDesign és a Writer programokban meglévő lehetőségeit mutatjuk be.

26. Számítógépes kiadványszerkesztési alapismeretek

26.1. Az írás története

Az írás kialakulása az emberi közösségek fejlődése során fellépő közigazgatási, gazdasági és kulturális szükségletek következménye. A társadalmak szervezési és irányítási feladataihoz, valamint a kultusz és ismeretek ápolásához létrehozott (és alkalmazott) írás – amely elsőként Mezopotámiára és Egyiptomra jellemző – az írásbeliséget, és vele párhuzamosan az írástudók rendjét eredményezte.

Az írás tudatosan rögzített, jelentést tartalmazó, olvasható jelekből áll. Sohasem önkényes kitalálás eredménye, általában több nemzedék, gyakran több nép által kialakult konvenció.

A megörökíteni kívánt információt kezdetben kő-, majd agyagtáblákra vésték, később papirusz- és pergamentekercsekre, bőrhártyákra írták. Az íróeszközök ennek megfelelően vésésre alkalmas eszközök, ill. különböző típusú tollak voltak. A papírt Kínában találták fel i. sz. 105-ben (Caj Lun), de Európába csak 1270 körül jutott el.

Az írás kialakulásánál négy fejlődésfokot különböztetünk meg, amelyek egyenlőtlenül fejlődve a különböző korokban egymás mellett és egymást kiegészítve is fellelhetők.

1. Emlékeztető (mnemotechnika) Minden olyan emlékeztető (például az útjelzések és egyéb, régóta használatos jelzések), amely közöl valamit, az írás első lépcsőfokához tartozik, ugyanúgy, mint a perui őslakók gondolatközlési formája: a csomóírás.

2. Képírás (piktogramma) Az írás fejlettebb korai formái festmények és rajzok. A vadászatok eseményeit ábrázoló korai kőkorszaki barlangfestmények azonban még sokértelműek, mert az elbeszélő ábrázolásokból hiányzik a hagyomány, viszont a Pireneusok lejtőjén talált, későbbi kőkorszakból származó kavicsok színes festékkel rajzolt jelei már valamelyest hasonlítanak mai betűinkhez.

3. A gondolat rögzítése (ideogramma) A maya-kódexekben már kifejezetten ideogrammákkal találkozhatunk: természeti tárgyak egyszerűsített képmásainak ábrázolásával adták vissza a szó értelmét. Később megjelennek a rejtvényszerű szótagjelek is. A jelek többnyire kombinációk, többértelműek, csak folyamatos olvasás alkalmával derül ki valódi értelmük.

A képek fokozatosan egyszerűsödnek, kisebbek lesznek, szóképletekké merevednek, eredeti értelmük lassan elvész. A régebbi ékírásban még felismerhetők a képírás jellegzetességei, de ezek idővel eltűnnek. Ennek elsősorban az az oka, hogy a gondolatoknak agyagtáblákon való rögzítésénél a jeleket egy élesre vágott fával nyomkodták a nedves agyagba, amelyet azután a tűző napon szárítottak meg. A rajzos képírás helyét így fokozatosan felváltotta az agyagtáblákon megjelenő függőleges, vízszintes és diagonális (ferde) vonalak összefüggéseiből és kombinációiból kialakuló ékírás.

4. A hangok jelei (fonogramma) A mai európai hangírás *betűi* kétségtelenül egykori képek rövidítései. Kiválasztásuk azon az elven alapul, hogy egy hangnak az ábrázolására olyan fogalom képét alkalmazták, amely a vélt hanggal kezdődik. Az első ilyen fajta ábécét a semita eredetű hikszoszok hozták létre az egyiptomi hieroglifák mintájára.

A föníciaiak a szavakat hangokra bontották és ezeket fonogrammákkal (hangjelekkel) ábrázolták.

A régi görög írásban ezen túlmenően új a magánhangzók hangjelének megjelenése, a soronkénti írás mértani rendje és az, hogy az írást balról jobbra vezették. A görög betűk jellemzője az egyforma betűmagasság és a függőleges vonalak uralma. A jelek formai különbsége meglehetősen egyszerű. Az írásjelek könnyű felismerhetőségét a függőleges, vízszintes és diagonális egyenesek, valamint a kör alakú formák ellentétei és kombinációi biztosították.

A római ábécé a főníciai, az etruszk és a görög ábécéből alakult ki, néhány új betűvel kiegészítve a források betűit. A latinok csak nagybetűket használtak. A csupa nagybetűvel való írás papíron, kézzel nem könnyű, és a vésett formától néha jelentősen eltér. Kerekebb, lágyabb betűket használtak a kéziratok készítői. Ezt az írásmódot unciális írásnak nevezzük. Az Kr. u. 400–500 környékén ebből az írásformából alakult ki a félunciális írás, amelyben a betűk magassága már különböző volt. Ezzel az írásformával megtörtént az első lépés a latin kisbetűs írás kialakulásában.

Az elmúlt ötszáz évben tervezett több ezer betűtípus többségénél a nagybetűk formája és arányai az i. sz. 114. körül emelt Trajanus-oszlop feliratához igazodnak. Ennek köszönhető például az a hagyomány is, hogy a csupa nagybetűvel *(verzállal)* szedett szöveget erősebben megritkítjuk, mivel a Trajanus-feliraton nem rendezték szavakba a betűket, hanem közöket hagytak a betűk között.

A fő ösztönzést a kisbetűk kialakulására a késői római folyóírásnak, a római kurzív írásnak és az ókeresztény kódexek félunciális írásának köszönhetjük. A kisbetűs ábécé a fejlődés további menetében úgy alakult ki, hogy az emberek mind gyorsabban írtak, az egyik betűről a másikra áttérve nem emelték fel a tollat, így a betűk egymásutánjából folyamatos szavak, ritmikus sorok keletkeztek, amelyben döntő része volt a lúdtoll használatának.

Az utolsó fejezet a latin írás kialakulásában a hetedik század végén Nagy Károly rendeletére bevezetett kisbetűs írás (karoling kisbetű), amely leegyszerűsítette és egységesítette az addig használt nagybetűs írásokat. A mai latin betűs írásunk ennek a kisbetűs írásnak válfaja. A gót betű és kézírásunk is ezekre a betűkre vezethetők vissza.

26.2. A könyvnyomtatás kialakulása

A XIII. századig könyvre csak a latinul írni-olvasni tudó keveseknek volt szüksége. Az egyház részére, az iskolai oktatáshoz és a közhatósági teendőkhöz a könyvek kizárólag a kolostorokban készültek. A szöveget a szerzetesek tíz-tizenkét leveles füzetenként másolták. A másolatokat a korrektor felülvizsgálta, ellátta kézjegyével, a miniátor színes kezdőbetűket festett, majd a füzeteket a ligátor (könyvkötő) könyvekké formálta: két fatábla közé erősítette, a fatáblákat bevonta bőrrel, ötvözött sarkokat és kapcsokat erősített rá, majd átadta a könyvtárnoknak, aki azt hosszú láncra verve erősítette a könyvtári olvasópulthoz.

A könyvek iránti keresletnövekedés a XIII. században az Európa nagyvárosaiban sorra alakuló egyetemeknek köszönhető. A másolást ekkor már hallgatók végezték, akik munkájukért fizetést kaptak. A bőrhártyás kötetek mellett olcsó papirosra írt kéziratos könyveket is készítettek. Egyes nagyvárosok egyetemei szabványosították a jegyzetek alakját, írásmódját stb. is.

A XV. században a haladás és a polgárosodás következtében a könyvek iránti igény ugrásszerűen megnövekedett. A könyvkereskedők hivatásos másolókat alkalmaztak, akik egy könyvet körülbelül fél év alatt másoltak le. Ez a teljesítmény azonban nem volt elegendő az igények kielégítéséhez. Az egyre nagyobb minőségi elvárások és a továbbra is magas könyvárak ellenére a színvonal fokozatosan romlott, a másolatról készült másolatokban halmozódtak a hibák. Szükségessé vált tehát egy olyan technológia létrehozása, amellyel gyorsan és olcsón

lehet szöveget közrebocsátani akár több ezer példányban, egymással teljesen azonos formában.

A legelső tömeges könyvsokszorosítás Laurens Janson Coster holland fametszőmester nevéhez fűződik, aki Aelius Donatus latin nyelvtanát tükörfordítottan fatáblákba metszette, és azokról 32 oldalas füzeteket nyomtatott. A nyomat úgy készült, hogy a fatábla eredeti síkjában megmaradó betűket, képelemeket befestették, ráfektették a papírt, és egy préssel megnyomták.

A fatáblák vésése közben elkövetett hibát nem lehetett javítani, újra kellett vésni az egész táblát. A fametszetes tömbnyomtatásnak azonban nemcsak ez volt az egyetlen hátránya. Az azonos betűk esetlegessége, a sorok görbesége, a sortávolságok egyenetlensége a vésőmester ügyességétől és a fa helyi keménységváltozásától függött. Ez utóbbiból eredő egyenetlen kopás a később készülő nyomatok minőségére is rossz hatással volt.

Az ólombetűkkel való szövegsokszorosítást elsőként egy mainzi aranyműves, Johannes Glensfleisch Gutenberg alkalmazta 1450 körül, aki az addigi különböző technikákat összevonta, ésszerűsítette, majd azokhoz hozzáadva saját találmányait létrehozta a könyvnyomtatást.

A különálló betűk szavakká, mondatokká való összeállítását már a rómaiak is ismerték, sőt, az ókori és középkori népek több évszázaddal a könyvnyomtatás feltalálása előtt nyomtattak pecsétnyomókkal.

Gutenberg a kódexekhez hasonlóan szép kivitelű könyvek nagy példányszámban és tökéletesen azonos formában történő olcsó előállítását tűzte ki célul. Rendszere az ábécére épült. A betűkészlet kidolgozásakor mesterábécét tervezett, ennek elemeit kiemelkedő betűként kemény fémpálcába véste *(patrica)*. A domborművű betűket rézötvözetbe préselve *matricát* készített, amelynek segítségével ólomöntéssel tetszőleges számú azonos betűt állíthatott elő. Az azonos magasságúra és vastagságúra, de a betű alakja által megkívánt szélességűre öntött betűk segítségével lehetett összerakni egy-egy kiadvány egy-egy oldalának nyomóformáját, azaz elvégezni az kézi (ólom)*szedést*.

Az egyenes sorokat és az egyenletes sortávolságot az ólombetűk azonos magassága biztosította. A kódexmásolók kézírását utánzó *proporcionális* (arányos) szedés a különböző szélesség miatt volt megvalósítható. (Az „i” betű sokkal keskenyebb, mint az „m”, így kevesebb helyet foglal.)

26.1. ábra. *Gutenberg Bibliája. Forrás: British Library*

Eleinte a nyomtatott könyveket kevésbé értékelték, mint a kézzel írottakat, ennek ellenére a nyomtatás gyorsan teret hódított Európában, segített az írás és a nyelv egységesítésében és a reneszánsz eszmék elterjedésében. Bár a nyomatás hatékonyságának növelése érdekében folyamatosan tökéletesítették a nyomdagépeket és a betűk előállítását, a technológia gyakorlatilag évszázadokon át változatlan maradt.

A kézi szedést a XIX. század végén gépesítették. Olyan gépeket készítettek, mint például a Linotype, amely billentyűzet segítségével igény szerint betűsorokat tudott önteni. Ezekkel a gépekkel való nyomóformakészítés volt a linószedés, ami a kézi szedést elsősorban a nagy példányszámú nyomtatványok (például újság-) nyomtatásának előkészítésében váltotta fel.

Az 1960-as években megjelentek olyan szedőgépek, amelyeken a mai számítógépes szedéshez hasonlóan lehetett előállítani a kiadvány egy mintáját és erről a mintáról készült aztán a különböző nyomástechnikáknak megfelelő nyomóforma. Ezt a szedésformát nevezzük fényszedésnek, ami a linószedést teljesen elavulttá tette.

A fényszedés számítógéppel való kiváltása az 1990-es években történt meg Napjainkban már kizárólag csak digitális szedéstechnológiával találkozhatunk.

26.3. Tipográfiai alapismeretek

A *tipográfia* egy kiadvány megtervezésének, a felhasznált eszközök kiválasztásának tudománya és művészete. A tipográfia a szöveges közlés megformázásával, a szedett szöveg és az illusztrációk együttes elrendezésével foglalkozik. Hagyományosan a nyomtatott szövegek megtervezését tekintik tipográfiának, de mivel a technika fokozott fejlődésével a szöveges közlés egyre inkább képernyőn történik, valamilyen szinten mindenki kapcsolatba kerül vele, aki – akár csak alkalmanként is – szövegek előállításával foglalkozik.

26.3.1. Tipográfiai szakkifejezések

A tipográfia szakszavai a korai technológiát és gyakorlatot tükrözik; a terminológia modernizálására tett kísérletek a legtöbb esetben kudarccal végződtek. A szakszavak apránként alakultak ki, a tipográfia szerves fejlődésének megfelelően.

A betűtípus *(typeface)* a betűk, számok és írásjelek egyedi, azonos grafikai elven megtervezett készlete, amelyeket „nyomtatott” szöveg szedéséhez használnak. Egy betűtípusnak nagyon sok betűváltozata lehet, ezek a változatok együtt alkotják a *betűcsaládot (type family)*. A betűváltozatok több szempont szerint csoportosíthatók. *Funkció* alapján megkülönböztetünk álló *(roman),* kurzív *(italic)* és kiskapitális *(small caps)* betűfajtát. A betűt alkotó vonalak *vastagsága* szerint egy változat lehet világos *(light),* normál, félkövér *(bold)* és kövér *(extrabold)*. Az egyes jelek *szélessége* szerint lehet keskeny *(condensed),* normál, széles *(expanded)* és egészen széles. *Díszítettsége* szerint egy változat lehet kontúros *(outline),* plasztikus, árnyékolt, díszes stb.

Az ólombetűk idején minden egyes betű fémtömbön helyezkedett el, ezt *betűtestnek* nevezték, melynek szélessége a betű szélessége szerint változott. A betűtest magassága *(törzsméret)* állandó, meghatározta a sorok közti minimális távolságot. A sortávolságot ólomcsíkok *(térzők)* közbeiktatásával növelni lehetett, ezt *sorritkításnak (leading)* nevezzük. A betűközt a betűtest szélessége szabja meg, ennek a köznek a kiigazítása az *egalizálás (kerning)*.

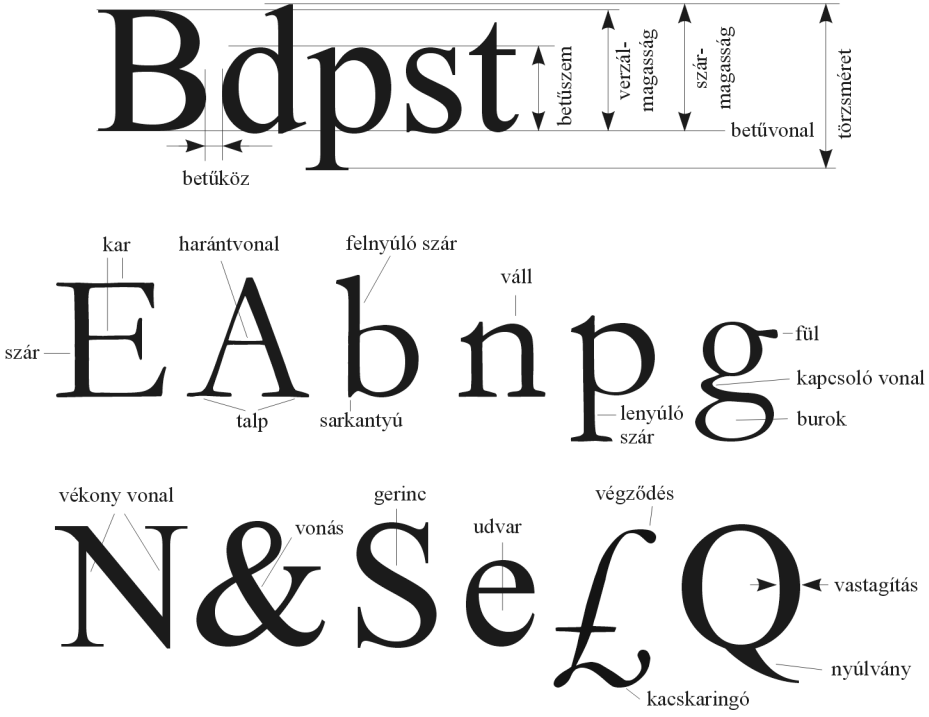
26.3.2. Tipográfiai mértékrendszerek

A nyomdai mértékrendszer egységesítését és rögzítését az üzemek közötti anyagszállítás tette szükségessé. Az európai kontinensen ma is használatos nyomdai pontrendszer kialakítását a francia Pierre Simon Fournier kezdte meg 1737 körül. Rendszerének lényege az, hogy a betűtörzsméretek egy alapegység többszörösei lehetnek csak.

Fournier rendszerét 1770-ben Francois Ambroise Didot módosította, a francia királyi lábmértékre építve fel azt. Az egység – a nyomdai pont – a lábnak a 6×12×12-ed, azaz 864-ed része (≈0,376 mm).

Az Európában használatos metrikus *Didot-féle pontrendszert* egy nemzetközi nyomdászkongresszus Hermann Berthold feldolgozása alapján rögzítette 1881-ben. Olyan, 300 mm-es etalon mércéket adtak közre, amelyek megfeleltek 798 Didot-féle pontnak (jele: p), ebből következően 1 méter 2660 Didot-féle pont.

Minden nyomdai hosszméret a Didot-féle pont egész számú többszöröse. A gyakorlatban használatos értékeket névvel is ellátták (pl. 4 p: gyémánt, 5 p: gyöngy, 12 p: ciceró stb.).



26.2. ábra. *A nyomdabetűk alapfogalmai*

Az *amerikai pontrendszert* az USA betűöntőinek szövetsége rögzítette 1886-ban, melyben 1 pont 0,0138 hüvelyk, azaz 0,35052 mm méretű, így 1 hüvelykben 72,463768116 pont van. Az angolszász országok ezt a rendszert vették át.

Az említett pontrendszerek mellé harmadikként csatlakozik a *számítástechnikai pontrendszer,* amely az amerikai pontrendszerből alakult ki, amit úgy módosítottak, hogy a pica és a pont mérete a hüvelyk egy hatoda illetve egy hetvenketted része legyen: Így

$$1 \text{ hüvelyk} = 25{,}4 \text{ mm} = 6 \text{ pica} = 72 \text{ pont},$$

vagyis 1 pont $\approx$ 0,353 mm. A számítógépes pica pont (jele: pt) és a nyomdászatban használt Didot-pont között a váltószám 1,065.

Mivel az ólomszedést és a fényszedést ma már nem alkalmazzák, egyre többet veszít a jelentőségéből a Didot-féle európai és az amerikai pontrendszer. A számítógépes nyomdai előkészítés egyeduralkodóvá válásával mindenütt egyre inkább a számítógépes pontrendszert használják.

26.3.3. A tipográfia alkotóelemei

A nyomtatott oldalon elhelyezett elemek elsődleges feladata az információközlés. Az elemeket funkciójuk alapján a következő kategóriákba sorolhatjuk: *tipográfiai, grafikai, illusztratív* és *díszítő* elemek. A betűk, nyomdai díszek, vonalak, foltok általában eleve rendelkezésre állnak többszöri felhasználás céljára, így nem teljesen egyedi alkotóelemei egy-egy nyomtatványnak vagy feliratnak. Egyediek viszont az illusztrációk, mivel ezeket célzottan egy-egy kiadvány díszítésére készítik..

A tipográfiai elemekhez tartozik a szöveg, a különböző rangú címek, egyéb mellérendelt szövegek (mint például a képaláírás), az oldalszámozás, és minden más tipográfiai jel, amely a nyelvi kommunikációt szolgálja.

Grafikai elem minden vizuális elem, amely a tipográfiai kommunikáció hatékonyságát növeli. Ilyenek a *léniák* (vonalak), a *piktogramok* (szedhető nem nyelvi szimbólumok), *körzetek* (nyomdai díszek) és minden más nem képi elem, amely elválasztja egymástól, és ezáltal áttekinthetővé teszi a szöveg egyes részeit.

Illusztratív elemekhez sorolható a rajz, az ábra, a fénykép, a térkép, a grafikon, a táblázat és minden olyan elem, amely kiegészíti és magyarázza a szöveget. A díszítőelemek azok, amelyek a mondanivaló megértése szempontjából nem szükségesek és nem is mindig hasznosak. Az illusztratív elemek elrendezése, a szöveges környezetbe való beépítése vagy szöveggel való ellátása napjainkban már egyre inkább a tipográfus munkája.

27. A szöveg szedése és szerkesztése

Egy szöveges dokumentum elkészítése egymást követő munkafázisokból áll. Ezeket a munkafázisokat számítógépes szerkesztésnél még akkor is célszerű betartani, ha az elkészítendő dokumentum nem túl nagy. Ez biztosítja ugyanis azt, hogy amit készíteni fogunk esztétikailag és tartalmilag megfelelő minőségű lesz. Ezek a lépések a következők:

1. A nyers szöveg bevitele minimális formai követelmények betartásával. Ezt a feladatot a szerzők végzik.
2. Első korrektúra. A helyesírási nyelvtani hibák javítása. A korrektorok munkája.
3. A korrektor javítási javaslatainak feldolgozása. A szerző feladata.
4. Lektorálás, a szöveg tartalmi ellenőrzése. A lektor feladata. Egy dokumentumnak több lektora is lehet.
5. A lektor által javasolt javítások elfogadása vagy visszautasítása. Elfogadás esetén a szükséges módosítások elvégzése a nyers szövegen. A szerzők feladata.
6. A nyers szöveg megformázása végleges formai követelmények szerint. A számítógépes szerkesztők feladat. Régen ezt a munkát végezték a szedők.
7. Korrektúrázás, a megformázott szöveg helyesírási és formai hibáinak felderítése. A korrektorok munkája, ideális esetben az első korrektúrát végző korrektorok ebben a munkában a saját első javításaikat nem ellenőrizhetik.
8. A korrektor javaslatainak megfelelő javítás a megformázott dokumentumban. A szövegszerkesztők és a műszaki szerkesztők feladata.

A számítógépes szövegszerkesztők megjelenésével a szerzők gyakran a fenti lépéseket jó részét egyetlen lépésé vonták össze, mivel a programok szolgáltatásai ezt lehetővé tették. Más lépések pénzügyi okokból elmaradtak. Ezek az elmaradó lépések általában a lektori és a korrektori munkák voltak. A megmaradó feladatokat összevonását vagy összemosását pedig az tette lehetővé, hogy a szerkesztőprogramokat be lehet állítani úgy, hogy az egyes lépésekhez tartozó feladatokat automatikusan elvégezzék. például automatikusan formázzák a begépelt szöveget, a helyesírási hibákat jelezzék, sőt javítsák, ha lehet stb.

Ennek ellenére javasolt az összevonást elkerülni, és a munkát a fenti lépésekben célszerű elvégezni. Sokkal szebb, jobb és ami lényeges jobban átszerkeszthető anyagot kapunk, ha így végezzük a szerkesztést. Nézzük tehát, hogyan célszerű a szövegszerkesztőkben ezeket a feladatokat elvégezni!

27.1. A nyers szöveg bevitele

A dokumentumaink készítésének első lépése a *szöveg begépelése*. Erre a célra olyan egyszerű programot válasszunk, amely alkalmas formázásokat és grafikát tartalmazó dokumentumok létrehozására, továbbá képes azt olyan formátumban menteni, amely megkönnyíti más programok használatát. A szerzők a cikkeket például elkészíthetik WordPad segítségével háttértár adathordozójára menthetik Rich Text formátumban, majd a leadott kéziratokból a műszaki szerkesztő megtervezi, a szövegszerkesztő InDesignnal összeállíthatja a kiadványt. Egyszerűbb dokumentumokhoz (diplomamunka, tanulmány) a teljes munkafolyamatra egyetlen irodai alkalmazás is elegendő (például Microsoft Word, LibeOffice Writer).[1]

Begépeléskor ügyeljünk a helyesírási és az egyszerűbb tipográfiai szabályok betartására, formai beállítást lehetőleg ne végezzünk.

Az összefüggő szövegben egymás után következő szavakat szóközzel választjuk el, hogy az írott szöveg világosan áttekinthető legyen. Kötőjellel írjuk viszont a szókettőzéssel keletkezett összetételt (néha-néha), a mellérendelő összetételt (kezét-lábát) és az ikerszavakat (hébe-hóba).

[1] A kiadványkészítésére alkalmas eszközök közül nem részesítjük előnyben egyiket sem, célunk elsősorban a munkafázisok és az általános tipográfiai szabályok bemutatására.

A mondatok szerkezetét, tagolódását, részeik egymáshoz kapcsolódását az írásjelek tükrözik. Az írásjelek használatának szabályai nyelvenként eltérőek. Magyar szövegben szóközt hagyunk az írásjellel lezárt mondatok, az összetett mondatok tagmondatai között, a kezdő zárójelek és idézőjelek előtt, a végzárójel és –idézőjel után, valamint a gondolatjel előtt és után. Nincs viszont szóköz a pont, a kérdőjel, a felkiáltójel, a vessző, a kettőspont, a pontosvessző előtt; a kezdő zárójel és az idézőjel hozzátapad az utána következő, a végzárójel és –idézőjel pedig az előtte álló szóhoz.

Ügyelni kell néhány írásjel begépelésénél is. A kötőjel *(diviz)* általában tapad az előtte és utána álló szó utolsó, illetve első betűjéhez: *használtautó-kereskedés.* A nagykötőjel hosszabb és keskenyebb, mint a kötőjel, hosszúsága megegyezik a gondolatjelével. A nagykötőjel néhány kivételtől eltekintve mindig tapad az előtte és utána álló szóhoz vagy számjegyhez: *Győr–Budapest, Apollo–11.* Szóköz akkor kerül a nagykötőjel elé és mögé, ha több szóból álló, bonyolultabb írásmódú szerkezeteket kapcsol össze: *január 5. – február 9.* A gondolatjel és a nagykötőjel között az a különbség, hogy a gondolatjel mindig szóközzel kapcsolódik az előtte álló szó utolsó betűjéhez, és nem tapad a következő szó első betűjéhez. A párbeszédjel a gondolatjellel azonos hosszúságú és formájú írásjel, amely a bekezdés élén áll, utána pedig nem törhető szóköz következik.

A tulajdonneveket mindig nagybetűvel kezdjük. A többi szót általában kis kezdőbetűvel írjuk, azonban az áttekinthetőség végett a mondatok elején ezeket is nagybetűvel kezdjük.

Az alábbi táblázat néhány gyakori begépelési hibát mutat be:

Hibás	**Helyes**
zárójeles (rész) példa	zárójeles (rész) példa
mondatvégi pont . Új mondat ...	mondatvégi pont. Új mondat...
kettőspont : egy, kettő	kettőspont: egy, kettő

Az írásjelek egy sorba kerülnek azzal a szóval, amelyikhez tapadnak. Ha szóközt tennénk mondjuk egy felkiáltójel elé, az önállóan is átkerülhetne a következő sorba, így az (hibásan) „!”-lel kezdődne.

A folyószöveget – tartalmi szempontok alapján – címekkel, bekezdésekkel és kiemelésekkel tagoljuk. A gépelés

során a programok többsége a szavakat automatikusan sorokba, azokat pedig oldalakra tördeli. A sor végére érve a szövegszerkesztő automatikusan új sorban folytatja a begépelt karakterek elhelyezését. Az ENTER billentyűt csak akkor kell és szabad leütni, ha új bekezdést kezdünk.

Az oldal végén is automatikus az új oldalra való áttérés. Ha azt szeretnénk, hogy valamelyik szövegrész új oldalra kerüljön, akkor szövegbevitelkor néhány vezérlőkarakter – mégpedig az úgynevezett *rögzített oldalhatár* karakterek – valamelyikének beszúrásával kezdhetünk új oldalt.

Előfordul, hogy két szónak feltétlenül egymás mellett kell lennie, mert nem akarjuk, hogy a sor vége széttörje őket (például 30 km). Ilyen esetben a két szó közé ugyancsak egy vezérlő karaktert, a *nem törhető szóközt* vagy a *nem törhető kötőjelet* kell tennünk.

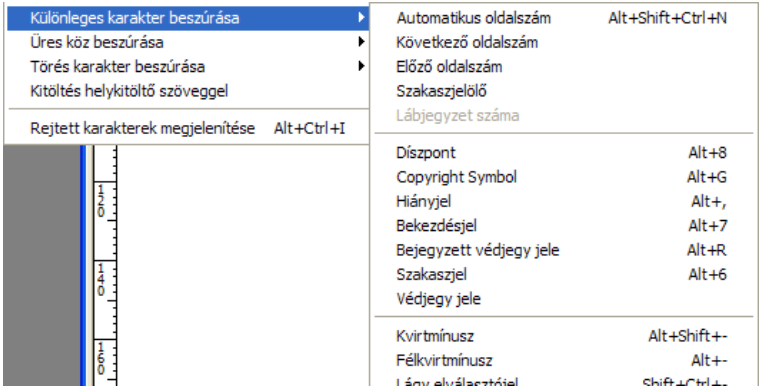
Folyamatos gépelésnél a szavak közé pontosan egy szóközt tegyünk. Szövegrészek pozicionálásához (például listák készítéséhez) szóközök helyett használjunk tabulátorkaraktert. Ezt a TAB billentyű megnyomását vihetjük be a dokumentumba. A tabulátorjel hatására a beviteli pont a következő tabulátorpozícióra ugrik, és a szöveg ehhez a ponthoz igazodik. A tabulátorpozíciók beállíthatók, akár minden tabulátorjelhez külön-külön is. Ha nem állítottunk tabulátorpozíciót, akkor az alapértelmezésben megadott érték alapján igazít a szövegszerkesztő. Nagyobb terjedelmű listáknál alkalmazzunk táblázatokat.

27.1.1. Speciális karakterek és szimbólumok

A dokumentum elkészítésekor a betűkön, számokon és írásjeleken kívül számos olyan karaktert kell használnunk, amely nem található meg a billentyűzeten. Ezeket a karaktereket menüből vagy párbeszédablakból kiválasztva, illetve a hozzájuk rendelt gyorsbillentyűkkel illeszthetjük be a szövegbe. A beszúrás folyamata a különböző szövegszerkesztőkben hasonló. Az InDesign programnak a beszúrást koordináló párbeszédablaka a 27.1. ábrán látható.

27.1.2. Automatikus javítás

Már a begépeléskor törekedjünk arra, hogy minél kevesebb helyesírási hibát tartalmazzon a nyers szövegünk. Elsősorban az irodai munkához készült szövegszerkesztők tartalmaznak olyan szolgáltatást, amellyel már a



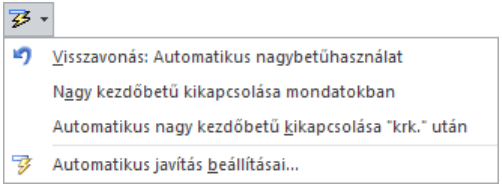
27.1. ábra. *Speciális karakter beszúrása InDesignban*

szöveg beírásakor a gépelési, helyesírási és nyelvtani hibákat automatikusan javítani lehet, Természetesen az automatikus javítás nem minden hibára vonatkozik, csak azokra, amiket a szerkesztőnk ismer, vagy – amikről tudva, hogy el fogjuk őket követni – a megfelelő módszerrel beállítottunk. A beállítás szövegszerkesztőnként más és más módon történhet, de közös tulajdonság, hogy a beállítások párbeszédablak segítségével adhatók meg, illetve módosíthatók, ha szükséges. (Például Word esetén Eszközök/Automatikus javítási beállítások menüpont.)

Az automatikus javítás hibát is okozhat. Ilyen hibák lehetnek például amikor a program az ismeretlen rövidítés után nagykezdőbetűsre váltja a szót, vagy amikor a hüvelyk jelét (") kicseréli nyomdai idézőjelre (”). Erre figyelve a hibás javítás visszavonható (27.2. ábra).

27.1.3. Vezérlőkarakterek

Azokat a jeleket, amelyek a nyomtatásban nem jelennek meg, de a dokumentum szerkezetét, tördelését befolyásolják, *vezérlőkaraktereknek* nevezzük. Ilyenek a már említett oldaltörést generáló rögzített oldalhatár



27.2. ábra. *Automatikus javítás visszavonása*

karakterek, bekezdésjel stb. Ezek halmaza szövegszerkesztőnként ként eltér, de kezelésük a betűkhöz és a speciális karakterekhez hasonlóan történik. A dokumentumba sokféleképpen kerülhetnek. Például a bekezdésvégjelet az ENTER billentyű megnyomásával, más karakterek menüből vagy gyorsbillentyűkkel szúrhatjuk be.

A vezérlőjeleket a nyers szöveg bevitelekor lehetőleg kerüljük, csak akkor használjunk ilyeneket, ha feltétlenül szükséges. Ezek a jelek a szerkesztés utolsó fázisában, a végleges forma kialakításakor kapnak jelentős szerepet.

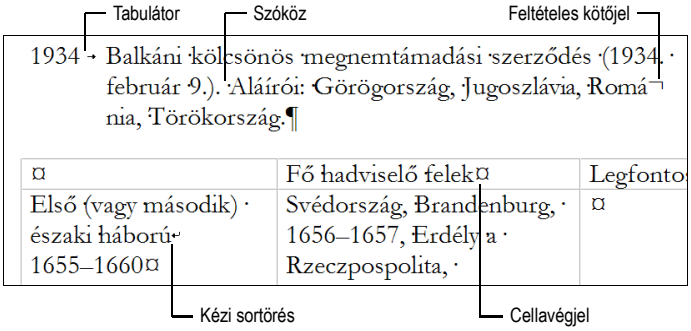
A szöveg begépelése és szerkesztése során vezérlőkarakterek kívánság szerint elrejthetők vagy megmutathatók. A megjelenítéshez vagy elrejtéshez irodai szövegszerkesztőknél kattintsunk a **Szokásos** eszköztár bekezdéskaraktert (¶) ábrázoló gombjára, illetve a beállításoknál kapcsoljuk be a szükséges formázási jeleket. A rejtett karakterek megjelenítése kiadványszerkesztőknél hasonló módon történik.

A 27.3. ábrán néhány vezérlőkaraktert és speciális jelet láthatunk.

Bonyolultabb dokumentumok esetén a szerzők táblázatokat, képleteket, ábrákat, képeket is használ a nyers szövegbe szúrva. Ezeknek az elemeknek elkészítése már bonyolultabb szerkesztési ismereteket követelnek, mint az egyszerű szöveg begépelése, ezért később részletesebben foglalkozunk velük.

A nyers szöveg kialakítása elvileg folyamatosan – a már meglévő szöveg végén folytatva a bevitelt – történhet. Ha készen vagyunk, a munka következő fázisa az első korrektúra és javítás elvégzése. Ez a szöveg- és kiadványszerkesztő programok segítségével a korrektor (esetenként a szerző) végezheti.

Ehhez a munkafázishoz már át kell tudnunk tekinteni a meglévő szöveget, azaz mozogni kell tudnunk



27.3. ábra. *Vezérlőjelek a Word-dokumentumban*

a dokumentumban, javítanunk kell a felfedezett hibákat és megjegyzéseket kell tudnunk fűzni azokhoz a részekhez, amiket valamilyen okból problémásnak tartunk, de javítani nem tudjuk. Ezt tekintjük át a következő fejezetekben.

27.2. Mozgás a dokumentumban, blokkműveletek

A mozgás és a szöveg kijelölése olyan alapvető műveletek, amelyek nélkül a szöveg szerkesztése elképzelhetetlen. Kétféleképpen mozoghatunk a dokumentumban: az egyik, ha a beviteli pont is mozog, a másik, ha nem, csak a dokumentumnak más és más része látszik a képernyőn. Ezek a feladatok egérrel és billentyűzettel egyaránt elvégezhetők. Munkánk hatékonyságát a megfelelő módszer alkalmazása nagymértékben meghatározza.

27.2.1. Mozgás a billentyűzettel

A billentyűzet segítségével való mozgás leggyakrabban használt műveleteit a 27.20. táblázat foglalja össze. A feltüntetett billentyűparancsok egyes szövegszerkesztőknél minimális mértékben eltérhetnek.

27.20. táblázat. *A beviteli pont mozgatása billentyűzettel*

Billentyű	**A mozgás iránya**
←, →	egy karakterrel balra, jobbra
↑, ↓	előző, következő sor
CTRL+←, CTRL+→	előző, következő szó
CTRL+↑, CTRL+↓	előző, következő bekezdés
HOME, END	az aktuális sor eleje, vége
PAGE UP, PAGE DOWN	egy képernyővel fel, le
ALT+CTRL+PAGE UP, ALT+CTRL+PAGE DOWN	ablak elejére, végére
CTRL+PAGE UP, CTRL+PAGE DOWN	az előző, következő oldal tetejére
CTRL+HOME, CTRL+END	dokumentum eleje, vége
TAB, SHIFT+TAB	egy cellával jobbra, balra (táblázatban)
SHIFT+F5	ugrás az előző szerkesztési pontra

Ha a dokumentumban a billentyűzet segítségével mozgunk, a beviteli pont helyzete a mozgásnak megfelelően változik.

27.2.2. Mozgás az egér segítségével

Minden dokumentumablaknak van egy vízszintes és egy függőleges görgetősávja, amikkel a dokumentumokat könnyen átlapozhatjuk. A rajtuk elhelyezkedő csúszkák a képernyőn látható rész függőleges és vízszintes helyét jelzik a dokumentumban. A görgetősávokkal végezhető műveleteket a 27.21. táblázat foglalja össze.

27.21. táblázat. *A görgetősávokkal végezhető műveletek*

Művelet	**A görgetés iránya**
kattintás a gombra	felfelé egy sornyit
kattintás a gombra	lefelé egy sornyit
kattintás a csúszka fölé	felfelé egy képernyőnyit
kattintás a csúszka alá	lefelé egy képernyőnyit
csúszka húzása a megfelelő irányba	ugrás a meghatározott oldalra
kattintás a gombra	balra
kattintás a gombra	jobbra
kattintás a gombra	előző oldal (vagy objektum)
kattintás a gombra	következő oldal (vagy objektum)
kattintás a gombra	adott típusú következő objektum

Ha a görgetősávokkal mozgunk, a beviteli pont nem mozdul el. A beviteli pont áthelyezéséhez az I alakú egérmutatóval kattintsunk a dokumentumban a megfelelő helyre. Ha a gördítősávot alkalmazva

megváltoztatjuk a képernyőn látható dokumentumrészt, majd megnyomjuk valamelyik mozgató billentyűt, akkor a beviteli pont úgy változik, mintha az a képernyőn látható dokumentumrészen lenne, és ahogyan ebből a pozícióból a billentyű lenyomásának megfelelően változnia kell.

27.2.3. Ugrás a dokumentum meghatározott helyére

Hosszú dokumentumoknál a leggyorsabb és egyben leghatékonyabb mozgást az Ugrás (Microsoft Word) vagy a Navigáció (LibeOffice Writer) biztosítja. Aktiválásuk után a megjelenő párbeszédablakban kiválasztható az a dokumentumelem, amire ugrani szeretnénk, majd az **Előző** és **Következő** gombokra kattintva a kijelölt dokumentumelem beviteli pont előtti, illetve utáni példányára léphetünk (például a következő címsorra), de ha pontosan tudjuk a keresett elem nevét vagy számát közvetlenül arra vihetjük a beviteli pontot.

27.2.4. Kijelölés billentyűzettel

A dokumentum egy részét (vagy akár egészét) akkor kell kijelölni, ha az ezután következő műveleteteket a megjelölt dokumentumrészen szeretnénk elvégezni. Sok parancs végrehajtási módja attól függően változik, hogy a dokumentum tartalmaz-e kijelölést. Billentyűzettel csak egyetlen összefüggő rész (blokk) jelölhető ki a következőképpen: nyomjuk le a SHIFT gombot, majd a mozgató billentyűket használva a program kijelöli azt a területet, amin a mozgás során a beviteli pont áthaladt, illetve törli a kijelölést, ha úgy mozgunk, hogy a beviteli pont a már kijelölt területen halad át. A kijelölt blokkot a programok inverz színekkel jelenítik meg.

27.2.5. Kijelölés egérrel

Az egérrel a kijelölést elvégezhetjük közvetlenül a szövegben, vagy – amennyiben a használt szövegszerkesztő lehetővé teszi – a kijelölő sáv segítségével. Kijelölő sáv a szöveg balra található külön meg nem jelölt terület, ahol az egérmutató alakja mindig felfelé mutató, jobbra dőlő nyíllá alakul. Ez a kijelölő sáv a táblázatok összes cellájában is megtalálható. Ezen a területen a nyíl iránya hasonló, de alakja más és színe fekete.

27.22. táblázat. *Kijelölés az egér használatával*

Kijelölendő rész	**Művelet**
tetszőleges szövegrész	az egér bal gombjának nyomva tartása mellett húzzuk végig az egeret a kijelölendő szövegen
objektum	kattintsunk a kijelölendő elemre
egy szó	kattintsunk duplán a szóra
egy mondat	tartsuk lenyomva a CTRL billentyűt, és kattintsunk a mondaton belül
egy bekezdés	kattintsunk triplán a bekezdésen belül vagy duplán a kijelölő sávra a bekezdés mellett
több bekezdés	kattintsunk duplán a kijelölő sávra, majd az egér bal gombjának nyomva tartása közben húzzuk az egeret a sáv mentén
egy sor	kattintsunk a kijelölő sávra a sor bal oldalán
több sor	a kijelölő sávban nyomjuk le az egér bal gombját, és annak nyomva tartása közben húzzuk az egeret a sáv mentén
teljes dokumentum	kattintsunk triplán a kijelölő sávra
szövegoszlop	kijelölés alatt tartsuk lenyomva az ALT billentyűt

27.3. A begépelt szöveg módosítása

Gépelési hibák javítására a legegyszerűbb módszer az, ha a módosítás helyére visszük a beviteli pontot (kurzort), töröljük a helytelen karaktert vagy szövegrészt, majd beírjuk a helyeset.

A beviteli pont előtti karakter a BACKSPACE, a beviteli pont utáni a DELETE billentyűvel törölhető. Az irodai felhasználásra készült szövegszerkesztőknél a beviteli pontot megelőző szót vagy szórészletet a CTRL+BACKSPACE billentyűkombinációval egy lépésben is törölhetjük. Hasonlóan használhatjuk a CTRL+DELETE billentyűkombinációt a beviteli pontot követő szó vagy szórészlet törlésére.

Hosszabb, összefüggő szövegrészt gyorsan és biztonságosan úgy törölhetünk, ha kijelöljük a törlendő szöveget, majd megnyomjuk a DELETE billentyűt. Ha valamilyen szövegrészt át szeretnénk írni, akkor kijelöljük, és elkezdjük az új begépelését. A kijelölt rész automatikusan törlődik.

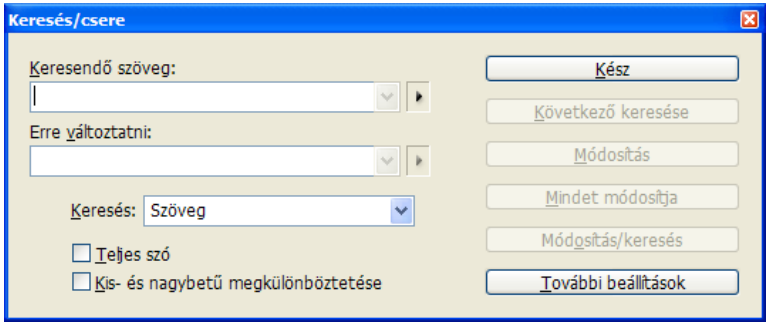
A gépelést kétféle üzemmódban végezhetjük. A *beszúró* üzemmódban a begépelt karakter a beviteli pont helyén megjelenik, az attól jobbra levő szöveg pedig hátrébb tolódik, míg *felülíró* üzemmódban a kurzortól jobbra lévő karakter átíródik. A két üzemmód között az Insert billentyű megnyomásával válthatunk.

27.3.1. Visszavonás, visszaállítás

A szerkesztés során bármikor végezhetünk hibás műveleteket, vagy olyat, amelynek eredményével nem vagyunk megelégedve. Ilyenkor szükségessé válik az előző állapot visszaállítása. Ehhez válasszuk ki a **Szerkesztés** menü **Visszavonás** parancsát, vagy kattintsunk a **Visszavonás** gombra. Ha a gomb rendelkezik lefelé mutató nyíllal, egyszerre több műveletet is visszavonhatunk. A **Mégis** gombra kattintva lehetőségünk van a visszavont műveletek ismételt elvégzésére (vagyis a visszavonás visszavonására) is.

27.3.2. Keresés és csere

Gyakran szükség van arra, hogy megkeressük vajon hol fordul elő egy szövegrész a dokumentumban, illetve ha megtaláltuk módosítsuk is. Erre például akkor lehet szükség, egy helyesírási hibát, amit több helyen is elkövettünk, javítanunk kell. Egy rövid levél esetén ezt megtehetjük úgy is, hogy a szöveget átolvassuk,



27.4. ábra. *Keresés és csere InDesignban*

és javítsuk ahol kell, de hosszabb dokumentumoknál ez a módszer pontatlansága és időigénye miatt nem megfelelő, inkább a keresés és csere funkciót alkalmazzuk.

A cserét a helyesírási és nyelvhelyességi hibák javítása mellett a felesleges karakterek és üres bekezdések törlésére, a hibás karakterek lecserélésére vagy akár formázásra is használhatjuk.

A keresés vagy csere előtt jelöljük ki azt a tartományt, amelyben keresni akarunk, majd adjuk ki a **Keresés** vagy a **Csere** parancsot. (Ha semmit sem jelölünk ki, a program a beviteli ponttól kezdődően a teljes dokumentumban keres vagy cserél, az ablakon lévő mezők állapotának és a kiválasztott gombnak megfelelően.)

A párbeszédablak mezőinek értelemszerű kitöltésével megadható, hogy mit kell keresni, ezt mire kell cserélni, ha szükséges. A keresés vezérelhető néhány kapcsolóval. Ezekkel állítható a kis- és nagybetűnek a keresés során való megkülönböztetése, a teljes szóra való keresésé és még néhány egyéb feltétel is.

Ha minden szükséges mezőt kitöltöttünk, a feltételeket beállítottuk, akkor elindítható a keresés, illetve a csere folyamata. Ezt a folyamatot többféleképpen lehet levezényelni, amelyhez a párbeszédablakban különböző gombokat találunk. Kereshetjük a keresendő szöveg legközelebbi előfordulását, ezt lecserélhetjük, ha akarjuk, de elindíthatjuk a keresést úgy is, hogy a keresendő szöveg minden előfordulása automatikusan legyen lecserélve. A feladatunk csak annyi, hogy a panel gombjai közül arra kattintsunk, ami a kívánságunknak

megfelelő keresést, cserét biztosít. Az összes előfordulás cseréjét körültekintően végezzük, mert a keresett szöveg olyan helyen is előfordulhat – és ott jól –, amire esetleg nem is gondolnánk.

A kereséshez, cseréhez a vezérlőjelek és a speciális karakterek ún. metakarakterek segítségével adhatók meg (például Wordnél a bekezdésjel kódja ^p). A metakarakterek egy része listából történő kiválasztással is betehető a beviteli mezőkbe.

Amennyiben a keresett szövegrész nem adható meg pontosan, de körülírható (pl. bármely egész szám), reguláris – más néven szabályos – kifejezést alkalmazhatunk mintakeresés vagy -csere céljából. A szövegminták literális és speciális karakterekből állnak, és egy vagy több szövegrészt írhatnak le. Literális karakternek azt a jelet nevezzük, ami nem jelent mást, csak önmagát (például az *a* literál csak az *a* betűre, a . [pont karakter] bármelyik karakterre illeszkedik). A LibreOffice Writer reguláris kifejezései a következő alapelemekből építhetők fel:

literális karakter:	betű- vagy számkarakter
számszerűsítő:	? * +
mennyiség:	{n} {n,} {n,m}
illeszkedési pozíció:	^ $ \> \<
vezérlőkarakter:	\
elágazás-szétválasztó:	|
csoport:	(*alkifejezés*)
karaktercsoport:	[*karakterlista* és/vagy *karaktertartomány*]
inverz karaktercsoport:	[^*karakterlista* és/vagy *karaktertartomány*]
karakterosztály:	[:alpha:] [:digit:] [:alnum:] [:lower:] [:upper:]
egyéb metakarakter:	. \n \t

A speciális karakterek részletes leírását lásd a LibreOffice Writer Súgóban.

A kifejezések felépítése egzakt módon az alábbi formában adható meg:

```
regKif :: = ág ('|' ág) *
ág :: = szelet*
szelet : = atom mennyiségjelző?
mennyiségjelző :: = [?*+] |('{' pontos |minimum
|intervallum '}')
atom :: = karakter |karakterosztály |('(' regKif ')')
karakterosztály ::= metakarakter |karaktercsoport
karaktercsoport ::= '[' pozKarakterlista |negKarakterlista ']'
pozKarakterlista ::= ( karakter |karakterintervallum
|metakarakter) +
negKarakterlista ::= '^' pozKarakterlista
```

A fenti formális leírás hétköznapibb megfogalmazásban a következő: a reguláris kifejezések egy vagy több ágból állnak. Az ágak szeletekre oszthatók. Minden szelet atomból és mennyiségjelzőből (számszerűsítő vagy mennyiség) tevődik össze. A mennyiségjelzővel az atom előfordulási gyakoriságát adhatjuk meg explicit módon. Elhagyása esetén az atom pontosan egyszer fordulhat elő. Az atom egyetlen karakter, karakterosztály vagy beágyazott kifejezés lehet.

Példa: Egy-egy gyakorlati reguláris kifejezés felírása a látszólagos bonyolultság ellenére sem nehéz. Például az 1[0-9]{3} reguláris kifejezéssel minden kétezer előtti négyjegyű évszám megtalálható. (Magyarázat: 1-gyel kezdődjön, ez az 1. karakterrel van megadva. Ezt három tetszőleges számjegy követi. A [0-9] rész a tetszőleges számjegyet adja meg, a {3} pedig az 1-et követő számjegyek számát definiálja.)

27.3.3. Kijelölt szövegrész másolása, mozgatása, törlése

A kijelölt szövegrészt törölhetjük a dokumentumból, másolatot készíthetünk róla, vagy éppen áthelyezhetjük máshova. Másolhatunk és mozgathatunk dokumentumon belül, dokumentumok között, vagy akár a kiadványszerkesztő és egy másik alkalmazás között.

A törléshez nyomjuk meg a DELETE vagy a BACKSPACE billentyűt. Kis távolságokra történő mozgatásnál használjuk a „fogd és vidd” technikát. (Egérkurzorral mutassunk rá a kijelölt blokkra, és a bal egérgombot lenyomva mozgassuk el az új helyére.) Ha az egérgombot elengedjük, a kijelölt szövegrész megjelenik az új pozícióban. Másolás esetén ugyanez a teendő, csak az egérgomb felengedése közben tartsuk lenyomva a CTRL billentyűt.

A dokumentumelemek az operációs rendszer által biztosított Vágólap segítségével biztonságosabban és kényelmesebben mozgathatók, másolhatók. A kijelölés után mozgatásnál a **Kivágás**, másolás esetén a **Másolás** nyomógombra kattintsunk. A beillesztéshez helyezzük a beviteli pontot a célpozícióba, majd ezt követően nyomjuk meg a **Beillesztés** gombot.

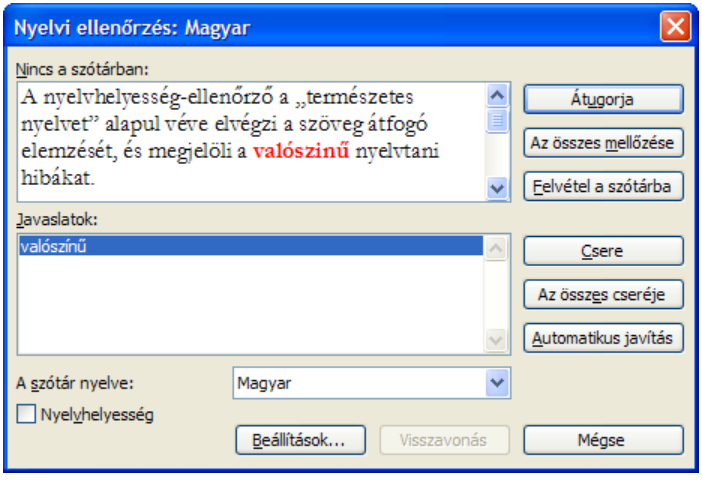
27.4. Nyelvi ellenőrzés

A kiadványszerkesztők nem csak szövegszerkesztésre alkalmasak, a szöveg nyelvi ellenőrzését is el tudják végezni, ha a szükséges programok telepítve vannak. Ezek a segédprogramok kétféle feladatot látnak el: egyrészt a dokumentum helyesírás- és nyelvhelyesség-ellenőrzését végzik, másrészt automatizálják a szavak elválasztását.

A szöveg ellenőrzéséhez és az elválasztáshoz be kell állítani, hogy milyen nyelven írtuk meg a nyers szöveget. ennek a megadott nyelvnek megfelelően történik az ellenőrzés, és az elválasztás. Célszerű a dokumentumban szereplő idegen szavakhoz külön-külön hozzárendelni a nekik megfelelő nyelvet, mert így elkerülhető, hogy a helyesírás-ellenőrzés ezeknél a szavaknál fennakadjon.

A helyesírás-ellenőrző a szavakat külön egységenként kezeli, ezért az elgépelt (már–mar, vízió–víziló), de önmagában értelmes szavakat nem javítja. Az ellenőrzés különböző szótárak alapján történik. Ezek egyrészt a nyelvi ellenőrző programhoz adott szótárak, másrészt a felhasználó által készített szótárak lesznek. A program nemcsak a hibásan leírt szavakat figyeli, hanem pl. a szóismétléseket és a rossz helyen használt nagybetűket is.

A nyelvhelyesség-ellenőrző a a beállított nyelvnek megfelelő nyelvi, nyelvtani szabályokat figyelembe véve elvégzi a szöveg átfogó elemzését, és megjelöli a valószínű nyelvtani hibákat, de nem szűri ki az összes hibát, csak azokat jelzi, amiket az ellenőrző algoritmus felismerni képes. A jelzett hibák sem minden esetben jelentik



27.5. ábra. *A szöveg nyelvi ellenőrzése*

azt, hogy valóban hibás a megjelölt szó.

A helyesírást és a nyelvhelyességet beírás közben, illetve utólag, a teljes dokumentumban vagy egy kijelölt szövegben egyszerre is ellenőrizhetjük. Alapesetben a Word például a helyesírási hibákat és elírásokat piros aláhúzással jelzi, a nyelvhelyességi hibákat pedig zölddel. Teljes körű ellenőrzéshez válasszuk ki a **Nyelvi ellenőrzés...** parancsot.

Hiba észlelésekor a Word a **Nyelvi ellenőrzés** párbeszédablak felső sorában feltünteti a probléma okát, alatta pedig szövegkörnyezetéből pirossal kiemelve a hibás szót láthatjuk. Ha valóban hibás, és javítani kell, válasszunk ki a jót a javaslatok közül, vagy írjuk át a megjelölt hibás szót jóra, majd kattintsunk a **Csere** gombra. Amennyiben a megjelölés téves, ugorhatjuk át, esetleg felvehetjük a szótárba, hogy a jövőben a program ne érzékelje hibásnak.

A beállított nyelvnek megfelelően történik a szöveg kézi vagy automatikus elválasztása is, mely szolgáltatást

Word esetében a teljes dokumentumra, Writerben és InDesignban bekezdésenként kapcsolhatjuk be. Elválasztáskor az irodai szövegszerkesztők egyenként kezelik a sorokat, a kiadványszerkesztők figyelembe veszik a szóközökre, betűközökre, jelek méretezésére és az elválasztásra beállított opciókat, majd úgy tervezik meg az elválasztást, hogy ezeknek az elemeknek az elrendezése a legjobb legyen a bekezdésben.

27.5. Korrektúra

Egy dokumentum helyesírási és nyelvhelyességi hibáit a nyelvi ellenőrző programok teljes egészében nem derítik fel, illetve nem javítják. Sőt előfordulhat – főleg akkor, ha programok javaslatait kritika nélkül elfogadjuk –, hogy a javítás során újabb hibák kerülnek a szövegbe. Ezért a fontosabb munkáknál (tanulmány, könyv) a szöveget korrektorral is javasolt átolvastatni.

A korrektor a nyelvtani, helyesírási és tipográfiai hibákért felel, tartalmi kérdésekben nem dönthet. A véleményezés hagyományosan nyomtatott papíron, a szabványos korrektúrajelekkel történt. A különböző korrektúrafordulókon a problémákat különböző színekkel jelölték: kékkel a szedési hibákat, pirossal az eredeti kéziratban nem szereplő újabb kiegészítéseket, javításokat.

Az elektronikus korrektúra során a program felügyeletével magában a dokumentumban is elvégezhetjük a módosításokat. Ha több korrektor van, ezeket a javításokat szerzőnként eltérő színnel lehet megkülönböztetni. A korrektúrát követően a szerző minden egyes javaslatról eldöntheti, hogy elfogadja azt, vagy az eredeti szöveget változatlanul meghagyja.

Ha azt szeretnénk, hogy látszódjon hol, mit, mikor és ki változtatott, akkor legyen bekapcsolva a változások követése, aminek következményeképpen a módosítások a változások a beállításnak megfelelően láthatók lesznek, sőt az eredeti állapot visszaállítható, azaz a szerzőnek nem kell a javasolt változást törölnie és visszaírnia az eredeti szöveget. A módosított sorok mellett megjelenő többnyire függőleges vonal is felhívja a figyelmet a változásra.

Önellenőrzés

1. Egészítse ki az alábbi mondatot!
A tipográfia a szöveges közlés megformázásával, a ☐ és az ☐ együttes elrendezésével foglalkozik.

2. Párosítsa az alábbi fogalmakat a magyar megfelelőjükkel!
fogalmak: typeface, outline, italic, roman, leading
lehetséges magyar megfelelők: sorritkítás, sarkantyú, betűarc, kurzív, betűtípus, vékony vonal, álló, kontúros, térző, egalizálás, betűköz, betűszem, olasz reneszánsz
Megoldás:

1. typeface – ☐
2. outline – ☐
3. italic – ☐
4. roman – ☐
5. leading – ☐

3. Mi a diviz?

- ☐ gúnyrajz
- ☐ kötőjel
- ☐ vízjel
- ☐ betűtörzs
- ☐ kettős kereszt

- [ ] ikerbetű
- [ ] alapvonal
- [ ] vízszintes vonal
- [ ] nem törhető vezérlőjel

4. Sorolja fel, funkciójuk alapján milyen csoportokba sorolhatók a Word-mezők!

_____, _____, _____

5. Egészítse ki az alábbi mondatot!
Azokat a jeleket, amelyek a nyomtatásban nem jelennek meg, de a dokumentum szerkezetét, tördelését befolyásolják, _____ nevezzük.

6. Az alábbiak közül mi található meg a vár+om$ reguláris kifejezésben?

- [ ] literális karakter
- [ ] számszerűsítő
- [ ] illeszkedési pozíció
- [ ] vezérlőkarakter
- [ ] elágazás-szétválasztó
- [ ] csoport
- [ ] metakarakter

11. LECKE

Dokumentumok formai kialakítása, dokumentumelemek

A második lecke első része – 28. fejezet – a dokumentumok formai kialakításáról szól. Megismerjük a dokumentumok oldalaira ajánlott klasszikus és modern elrendezési formákat, szó lesz arról, hogy az egyes dokumentumelemek hogyan alakíthatók a terveinknek megfelelő formátumú elemekké.

Ebben a leckében elsősorban azt taglaljuk, hogy mit lehet a szerkesztőprogramokkal elérni, és kevésbé foglalkozunk azzal, hogy ezt konkréten hogyan is lehet végrehajtani az egyes programokkal.

Természetesen ebben a részben is arra törekszünk, hogy szisztematikus munkára adjunk receptet, és ilyen munkára ösztönözzük a programok használóit is.

Gyakori hiba ugyanis a szerkesztésben az ad hoc megoldások használat, amelynek az a következménye, hogy a dokumentumban még kismértékű változás is felboríthat minden beállítást, és a munkát szinte elölről kell kezdenünk. Végezzük a munkánkat szisztematikusan kerülve a pillanatnyilag jónak tűnő egyedi megoldásokat!

A második részben olyan dokumentumelemek ismertetünk, amelyek kiegészítői a szerző által elkészített és megformázott szövegnek. Ilyenek a tartalomjegyzék, a bibliográfiai adatok listája, a tárgymutató stb. A legtöbb szerkesztőprogram ezek elkészítését megfelelően szabályozva ugyan, de automatikusan végzi.

Ebben a részben találjuk a képletek megkomponálásának szabályait is. Ugyancsak itt olvashatunk a kiegészítő dokumentumelemekről, mint például a címnegyed, amik a nyomtatott kiadványok állandó részei kell hogy legyenek.

28. Elrendezés, formai kialakítás

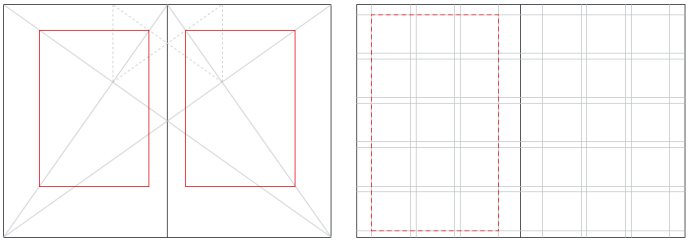
Miután véglegesítettük a dokumentum tartalmát, a helyesírási hibákat kijavítottuk, hozzáfoghatunk a formai kialakításnak. Elsőként meg kell terveznünk a szöveg és egyéb dokumentumelemek elrendezését, az alkalmazott betűk típusát, méretét stb., majd a terv alapján kivitelezhetjük a formázást.

A dokumentumok formázásakor három *formázási egységet* különböztetünk meg. A legkisebb formázható elem a karakter. Ezekből szavak, mondatok épülnek fel. A mondatok bekezdésekbe szerveződnek, és ezek a bekezdések alkotják a formázás következő szintjét. Egy dokumentum további hagyományos egységei, mint a fejezet is, nem formázási egységek, mivel gyakran előfordulhat, hogy egy-egy fejezet nem egységes formátumú kell hogy legyen. Ezért a szövegszerkesztésben megjelent a szakasz, mint a dokumentumok legnagyobb formázási egysége. A bekezdéseket bekezdésvégjel, a szakaszokat szakasztörés-karakter választja el egymástól. Néhány formai beállítást, noha szakaszokra definiálhatók, a teljes dokumentumra is megadható.

Dokumentumaink esztétikus kinézetét a formai tulajdonságok megtervezésével biztosíthatjuk. A terveknek megfelelően kialakítjuk a szakaszok, a bekezdések és a karakterek tulajdonságait leíró beállításokat. A szakaszszintű beállítások szakaszonként, a bekezdés- és karakterszintű beállításokra a formai tulajdonságok leírására szolgáló stíluselemek használhatók. Ezen utóbbi két egység formázását egyedi beállítással lehetőleg kerüljük, csak akkor célszerű az egyedi beállítás, ha a dokumentumunk pár oldalas, és egyszeri felhasználásra készült. A stílusok azon túl, hogy összefoglalják az adott formázási egység beállításait, rugalmas kialakítást tesznek lehetővé.

A formázást a nagyobb egységek beállításától a kisebbekig célszerű végezni. Először adjuk meg a papír és a margók méretét, majd állítsuk be a bekezdések és a kiemelt szövegrészek stílusát. A stílusoktól eltérő egyéni formai beállításokat ezt követően végezzük el.

A szövegszerkesztők és a kiadványszerkesztők tartalmaznak olyan előre definiált formákat, amelyekre a tervezésben építhetünk. Ezeknek a formátumoknak gyűjteményét kiadványtípusonként a szerkesztők is kialakíthatják, ami mintegy dokumentumsablon önállóan is háttértárra menthető. és egy dokumentum elkészítése ennek az elmentett formai tulajdonságokat leíró rendszernek betöltésével indítható.



28.1. ábra. *A hagyományos és modern lapstruktúra*

28.1. Az elrendezés megtervezése

Ahhoz, hogy egy kiadvány megfelelő struktúrájú, valamint esztétikai megjelenésű legyen, mindent meg kell terveznünk, akár egy épületet. A műszaki kivitelezés során a legapróbb részleteket is szigorúan ennek a. *tipográfiai tervnek* megfelelően kell kialakítanunk. Ez a szemlélet fokozottan érvényes egy könyvsorozat vagy időszaki kiadvány – idegen szóval periodikum – esetén.

A dokumentumelemek elrendezését tekintve kétféle stílust különböztetünk meg. A *hagyományos stílus* még a kézzel írt dokumentumok elrendezését alapul véve alakult ki. A hagyományos elrendezést a klasszikus görög matematikában és geometriában megadott arányok jellemzik, esztétikai szépségét az egyszerűség és a nyugodt harmónia biztosítja.

A *modern stílus* azokra a vizuális alapelvekre épül, amelyeket a XX. század elején rögzítettek az alaklélektan kutatói. A modern nyomtatott oldalakon az elemek elrendezését és a közöttük lévő teret modulhálós szerkezet határozza meg. A tervező döntései mögött nincs összefüggő filozófia, a funkció dönti el, hogy mi kerüljön az oldalra: csak azok az elemek szerepelnek, amelyeknek funkciójuk van és csak akkorák, amekkorának a funkció betöltéséhez lenniük kell. A kommunikációs szerep mellett megjelenik az elrendezés grafikai értéke.

A dokumentumok oldalainak, lapjainak elrendezésének megtervezésekor arra kell törekedni, hogy a lapokon rend, egyensúly, harmónia uralkodjon. Ezek a jellemzők egyrészt tipográfiai ismeretekkel, másrészt egészséges esztétikai érzékkel megtervezett oldalakkal lehet biztosítani. Néhány egyszerű szempontot az alábbiakban javasolunk a tervezéshez.

28.1.1. Rend vagy káosz

A hagyományos oldalformátumon a rendet az egymást követő bekezdések monoton ismétlődése adja. Ezt a monotonitást meg-megtöri az egyes bekezdések formai megjelenítésének változása, táblázatok, illusztrációk, képek elhelyezése az oldalon stb. Arra kell a szerkesztőnek törekednie, hogy a monotonitás és az azt megtörő elemek összességében a jobb megértést, a lényegest a kevésbé lényegestől való megkülönböztetést segítsék.

Egy-egy oldal elemei adják az oldal mintázatát. Ennek a mintázatnak segítenie kell az olvasót, ezért úgy kell megtervezni, hogy ne váljon unalmassá – no nem a tartalom, mivel annak érdekességét figyelemlekötő erejét a szerzőnek kell biztosítania. A szerkesztő a nyomtatott szöveg szürkeségét monotonitását – és ezzel az egyhangúságát oldhatja fel olyan elemek használatával, mint a rövidebb bekezdésekre való tördelés, az egyes bekezdések kezdőbetűinek díszes formázása, az úgynevezett iniciálék használata.

Megtöri a monotonitást a kiemelés – javasolt a kurzív betűk használata –, a fejezetcímeknek a többi bekezdéstől való eltérő stílusa is. Nem javasolt az egyhangúság feloldására a betűtípusok keverése még úgy sem, hogy egy bekezdésen belül a betűk típusa nem változik, csak az egymást követő bekezdéseket szedjük más betűtípussal. Nem javasolt a kiemelés bold stílusú betűkkel szedni sem.

Modern stílusú oldalakon nagyobb szabadsággal oldhatók fel az oldalak egyhangúsága. Változatos módon helyezhetjük el a bekezdéseinket, illusztrációinkat. Változatos betűméretet alkalmazhatunk, bizonyos esetekben még a betűtípusok is keverhetők, bár itt sem ajánlott a román betűcsalád (az úgynevezett talpas betűk családja) betűit a helvét betűcsalád (az úgynevezett talp nélküli betűk családja) betűivel keverni.

Nagyobb a szabadság a modern stílusban a bekezdések elhelyezését tekintve is. Főleg a hirdetéseket tartalmazó oldalakon néha még az is előfordulhat, hogy egyes bekezdéseket 180 fokkal elforgatva helyezzük el a többi bekezdés közé.

Szélsőségesen fogalmazva a hagyományos elrendezésben elsősorban a szigorú rendnek kell uralkodnia, a modern szerkezetben ezt a rendet néha jelentősen is felboríthatjuk, és a kaotikus elrendezést választhatjuk a tervezésben.

A bulvárlapok ennek a kaotikus elrendezésnek élő példái. A harsány főcímek rejtik a mondanivaló sivárságát!

28.1.2. Szimmetria

A szimmetria kultúránk integráns része, mindenféle tervezési tevékenységben gyakran alkalmazzák. A csomagolások és a címkék is jobbára szimmetrikusak; a készülékek tervezésében erősen érvényesül a szimmetria, még akkor is, ha a funkció nem kívánja meg. Az kiadványok oldalainak tervezésében is gyakran alkalmazzuk.

A hagyományos nyomtatott oldalon a *tengelyszimmetria* érvényesül, ahol egy láthatatlan függőleges tengely két egyenlő félre osztja az egészet. Az elrendezés a szimmetria által a statikus egyensúlyra törekszik, a stabilitás és a rend érzését kelti. Ez a stílus elsősorban a komoly mondanivalót tartalmazó kiadványokra jellemző.

A modern oldalelrendezést az *aszimmetria* jellemzi, ezáltal dinamikussá téve az összképet. A mozgást az elemek statikus alaphelyzettől eltérő pozíciója, valamint a méretek klasszikus arányoktól való eltérése sugallja. Manapság már mindenhol alkalmazható, szélsőséges megoldásaival a hirdetésekben, a plakátokon, a bulvármédia termékei között találkozhatunk. Olyan kiadványokban célszerű alkalmazni, amiben a figyelemfelhívás a legfontosabb szempont.

28.1.3. Egyensúly és harmónia

A *harmóniát* úgy jellemezhetjük, mint az elemek békés együttléte a nyomtatott oldalon. Harmónia akkor jön létre, ha az alkotóelemek mérete, léptéke, egymáshoz viszonyított helyzete arányban áll egymással, a kontraszt minimális és semmi sem téríti el a szemet lapon az olvasási iránytól.. Az elemek *szürkeértékének,* azaz a világos–sötét arányának és *léptékének* kontrasztja és szimmetriája lényeges szerepet tölt be a hagyományos oldalelrendezés által keltett harmónia- és egyensúlyérzetben.

Sajnos az egyszerűbb szövegszerkesztők ennek kialakításához adnak ugyan eszközöket, de automatikusan nem biztosítják, hogy a megfelelő egyensúly megteremtődjék; ezt a szerkesztőre bízzák. A jobb kiadványszerkesztők már bizonyos automatizmusokat is alkalmaznak, de a szerkesztő szerepe az egyensúly megteremtésében itt is jelentős lehet.

Ha modern stílusú kiadványt akarunk létrehozni a tervezés alapelve az egység és a változatosság közötti egyensúly elérése legyen. Az elemek rendszerint egyszerűbbek, és nagyobb kontrasztot teremtenek az oldalon, mint hagyományos megfelelőik. A szöveget kompakt tömbökben célszerű elhelyezni, és sokkal nagyobb mértékben használhatók a díszítő és illusztratív elemek. Ezek az utóbbiak akár az oldalak területének akár 60-70 százalékát is kitölthetik, ezzel sokkal összetettebb elrendezésre adnak lehetőséget.

28.1.4. Arány

Az arány lényeges eleme a harmóniának és az egyensúlynak. A hagyományos nyomtatott oldalt a görög matematika és geometria által kidolgozott arányok egyszerűsége jellemzi. A klasszikus arányrendszer a nyugati kultúra része maradt századokon át, s ma is ezt tarjuk a szemnek kellemes elrendezésnek.

A legismertebb klasszikus arányt *aranymetszésnek* nevezik. Egy szakaszt akkor osztunk fel az aranymetszés szabályainak megfelelően két részre, ha a rövidebb rész úgy aránylik a hosszabbhoz, mint a hosszabb a teljes szakaszhoz. Az aranymetszést nemcsak az oldal szélességi–magassági arányánál használjuk fel, hanem az oldalon elhelyezett elemek formájánál és elhelyezésénél is.

A modern oldalakon az összetettebb szerkezetben az arányok nem a klasszikus rendszert követik. Egyénileg, a szerkesztő ízlése szerint alakíthatók.

28.1.5. Térköz és textúra

Egy dokumentumelem *szürkeértékét* az határozza meg, mennyire sötétnek vagy világosnak látszik. Ez a tulajdonság viszonylagos, erősen függ attól, hogy az elem milyen környezetben helyezkedik el. A *textúra* a szürkeérték, többnyire szabálytalan eloszlása egy elemen belül.

A hagyományos nyomtatott oldal szürkeértékét a szöveg egyenletessége által létrehozott textúra adja, az egyéb elemeknek ehhez kell harmonizálniuk. A közöket vizuálisan úgy kell elosztani, hogy egy statikus és szimmetrikus szerkezeten belül a textúrák harmonikusan ötvöződjenek, nagy üres területek, egyenetlenségek nem engedhetők meg. Az elemeket úgy kell méretezni, hogy az elosztásuk egyenletes lehessen, nem célszerű nagyméretű kép mellé egy kicsit tervezni stb. Ha az elemek túlságosan eltérő méretűek, egyenetlen textúra jön létre.

A modern stílust a tömörség és a kontraszt különbözteti meg a hagyományostól. A modern betűtípusokkal tervezzünk, így a szedett szöveg egyenletes textúráját a talpak hiánya, a betűk viszonylag nagy szemmagassága, valamint a lehető legkisebb betűköz és sortávolság adja. A kontraszt megteremtésében alapvető szerepet játszik az üres tér.

28.2. Lapelrendezés

A kiadványok alapeleme az oldal, ezért első lépésben az oldalbeállítás az, amit a dokumentum formázásakor el kell végeznünk. Hogy mennyi szöveg fér el egy oldalon, az a papír méretének, tájolásának és a margók méretének a függvénye ezért ezeket kell a lap, illetve oldal alapbeállításakor megadni.

Az oldalak elrendezését szakaszokkal (pl. Word), oldalstílusokkal (pl. Writer) vagy mesteroldalak (pl. InDesign) lehet definiálni. Elsőként a papír méretét állítsuk be, majd ezt követően adjuk meg a tájolást, az oldalkialakítást (egyoldalas, kétoldalas), a margók méretét, az oldalak kezdőpontját (páros, páratlan stb.), függőleges igazítását, valamint az élőfej, élőláb beállításait.

Az élőfej az oldalak tetején, az élőláb az oldalak alján elhelyezkedő, oldaltípusonként ismétlődő jellegű szöveget tartalmaznak (például oldalszámot). A margók és élőfej, élőláb méretek, valamint a papírméret meghatározzák azt a területet, ahol a kiadvány szövege és az illusztrációk általában elhelyezkednek. Ezt a területet szedéstükörnek nevezzük.

A magyar szabványban megtalálható papíralakok és szedéstükörméretek az elrendezés sok variációját teszik lehetővé. A margók méreteit a klasszikus kialakítás szerint a hagyomány határozza meg (például átlagosan kiadványban a belső:felső:külső:alsó margók aránya 2:3:4:6), méretüket a papír és a szedéstükör mérete alapján

számíthatjuk ki. A modern elrendezésben a szerkesztő szabad kezet kap a méretek meghatározásában, csak az a megkötés, hogy ne legyenek a margók túl kicsik vagy nagyok a szedéstükör méretéhez viszonyítva.

Ha B/5-ös méretű álló tájolású papíron, klasszikus elrendezés szerint 27 ciceró szélességű szedéstükröt akarunk elhelyezni, a margóméreteket a fenti arányok mellett az alábbi módon határozhatjuk meg: a 176 mm széles lapon 27×4,511 mm ≈ 122 mm széles szedéstükör mellett a belső és a külső margó összesen 176 – 122 mm = 54 mm. Mivel a külső margó kétszerese a belsőének, a belső margó 54 / 3 mm =18 mm. A felső margó a belső másfélszerese: 27 mm, a külső margó 36 mm, az alsó pedig 54 mm.

28.3. Szakaszszintű formázás

Az oldal, illetve a lapok adatainak megadása szakaszonként is történhet, de a teljes dokumentumra is beállítható. Számos szerkesztési feladat viszont csak szakaszok létrehozásával kivitelezhető, ezért számos szerkesztési feladat csak szakaszok létrehozásával végezhető el. Ilyen beállítások a következők:

1. A szöveg hasábokra való bontása új szakaszt kíván, ha a hasábok száma megváltozik.
2. A lábjegyzetek sorszámozása szakaszonként állítható.

Amennyiben a kiadvány- illetve a szövegszerkesztő program nem ismeri az oldalak formátumát meghatározó oldalstílusokat, ezeket a szakasztulajdonságokkal adhatjuk meg:

1. hogy az adott szakasz páratlan illetve páros oldalon kezdődjön-e,
2. hogy a szakasz milyen papírméretet használjon, ennek milyen legyen a tájolása és margóméretei,
3. milyen legyen az adott szakaszon belül az élőfej- és élőláb-beállítás.

28.3.1. Szakaszok létrehozása

A Word-dokumentum létrehozása után mindig egyetlen szakaszból áll, azonban úgynevezett töréspontok beszúrásával tetszőleges számú szakaszra oszthatjuk. A dokumentum egyes szakaszait az ún. *szakasztörések* határolják. A szakasztörés – akárcsak a bekezdésjel – vezérlőkarakternek számít, ez a jel hordozza az utána következő szakasz tulajdonságait. Ha töröljük – lehet –, akkor az utána lévő szakasz minden eleme az előtte lévő szakaszhoz kapcsolódik átvéve annak minden tulajdonságát.

Többféle szakasztörés típus létezhet, ami meghatározza, hogy a hozzá tartozó szakasz hol kezdődjön. Folyamatos típus esetén az új szakasz folytatólagosan, közvetlenül az előző után kezdődik, új oldal típusnál a következő oldal tetején. Kétoldalas dokumentumoknál a páros vagy páratlan oldalra törő szakasztörésekkel azt állíthatjuk be, hogy a szakaszok bal vagy jobb oldalon kezdődjenek. Ha egy-egy fejezetet önálló szakaszként állítunk be, akkor ezzel a típussal érhetjük el, hogy a fejezetek kezdete a kiadvány melyik oldalára essen.

A Writer-dokumentumban a szakaszok névvel ellátott szövegblokkok, melyek elrendezése (pl. hasábszám) elsőbbséget élveznek az oldalstílusban megadott elrendezéssel szemben, a szakaszokat írásvédetté tehetjük a véletlen módosítások elkerüléséhez, normál szöveggé alakíthatjuk, elrejthetjük, vagy megjelenítésüket logikai kifejezések megadásával vezérelhetjük.

28.3.2. Hasábok

Többhasábos elrendezésnél a hasábok lehetnek azonos, illetve különböző szélességűek. A szövegük folyamatosan olvasható, az egyik hasáb alját elérve a szöveg a tőle jobbra következő tetején folytatódik.

A hasábokkal kapcsolatos jellemzőket – a hasábok számát, a szélességüket, a hasábok közti távolságot a hasábközt – a **Hasábok** párbeszédablakban állíthatjuk be, ugyanitt de akár választhatunk előre elkészített mintákból is. Ha azonos szélességű hasábokat szeretnénk használni, csak egy szélesség- és egy térközméretet kell megadnunk. A hasábok közé a térköz közepére elválasztó vonalat is húzhatunk.

A hasábok és térközök méretei összegének mindig meg kell egyeznie a szedéstükör szélességével. Ha olyan méreteket adunk meg, amiknek összege kisebb vagy nagyobb, mint a szedéstükör szélessége, a szerkesztőprogramok jelzik a hibát, és általában az utolsó hasáb méretét úgy változtatják meg, hogy az

egyenlőség teljesüljön. Például: 12,2 cm tükörszélesség mellett 0,5 cm hasábközt beállítva, a hasábok szélessége 5,85 cm lesz.

28.4. Bekezdés szintű formázás

Kiadványainkban – a könnyebb olvashatóság érdekében – a folyószöveget bekezdésekkel tagoljuk, egy-egy új gondolat általában új bekezdésbe kerül. Ezeknek a bekezdéseknek a külalakja határozza meg elsősorban azt, hogy a dokumentumunk hogyan fog kinézni.

A bekezdésformázó parancsok kiadása előtt meg kell határoznunk a formázandó bekezdések körét. Egyetlen bekezdés formázásához elég, ha a beviteli pontot az adott bekezdésbe helyezzük. Több bekezdés együttes formázásakor készítsünk olyan kijelölést, amely érint minden formázandó bekezdést.

A dokumentumban minden bekezdést bekezdésjel zár le, a bekezdések formázási tulajdonságait ezek hordozzák. Ha egy bekezdésjelet másolunk, mozgatunk vagy törlünk, azzal együtt tulajdonképpen másoljuk, mozgatjuk vagy töröljük az adott bekezdés formázási jellemzőit is. Amikor bekezdésen belül leütjük az ENTER billentyűt, az újonnan keletkező bekezdés általában örökli az előző beállításait.

Az alapbeállítások között a szöveg igazítását, a sorok behúzását, a bekezdés előtti, utáni térközök és a sorköz méretét adhatjuk meg.

28.4.1. Bekezdések igazítása, behúzása, térközök, sortávolság

A sorok igazítása lehet balra vagy jobbra zárt, aszerint, hogy a sorok a bal oldali margó mellett kezdődnek-e egyvonalban egymás alatt, vagy a jobb oldali margó mellett fejeződnek-e be egymás alatt egyvonalban. Középre zárásnál a sorok szimmetrikusan helyezkednek el a hasábban, sorkizárt (tömbös) igazításnál a sorok mindkét margó mellett egyvonalban lesznek. Többhasábos szedés esetén a hasábközhöz történik az igazítás, ha a szöveg nem a margó mellett kezdődik vagy végződik.

A bekezdések másik fontos jellemzője a behúzás, ami a bekezdés szövegének és a margóknak viszonyát szabályozza. Ez beállítható az első sorra, a teljes bekezdés jobb illetve bal oldalára A jobb és bal oldali behúzás a

szöveg jobb és bal oldali margótól való távolságát határozza meg. Ha nem adjuk meg, akkor ez a távolság nulla, azaz a szöveg minden sor – esetleg az első kezdetét és az utolsó végét kivéve – éppen a margónál kezdődik és fejeződik be. Negatív értékekre a sorok kezdete és vége a margóra nyúlik.

Az első sor behúzása a bal oldali behúzáshoz mérten relatívan határozza meg az első sor kezdetének helyét. Pozitív értékre beljebb, negatív érték esetén a bekezdés többi sorához mérten kijjebb kezdődik a sor. Az angolszász hagyomány szerint, amit az európai gyakorlat is átvett,az első sort tompán is hagyhatjuk, azaz nem adunk meg plusz behúzási értéket az első sorra. Ha mégis megadnánk, akkor ennek mértéke a hagyományos forma szerint 24 cicerós (≈10,8 cm) sorig 1 kvirt (egy betű törzsméretével azonos szélességű üres hely; betűnégyzet), hosszabb soroknál 2 kvirt.

Két bekezdés közti üres terület méretét – a térközöket – ugyancsak bekezdés szinten lehet szabályozni. Ha a méret nulla, akkor a bekezdések közti távolság megegyezik a sorok közötti távolsággal, ha nem nulla, akkor általában annyi üres hely marad két bekezdés között, amennyit ez a két érték definiál.

A sortávolság ugyancsak beállítható. Lehet automatikus méretű egyszeres, másfélszeres, kétszeres stb. sortávolságot megadni, de lehet pontos méretű sortávolságot is beállítani.

A szövegbeosztás beállításaival a bekezdés oldalak közti tördelését szabályozhatjuk. Megakadályozhatjuk, hogy a bekezdés első vagy utolsó sora külön lapra, vagy egy címsor önállóan a lap aljára kerüljön. A lap alján egyedülálló első sort fattyú-, a lap tetején egyedül álló utolsó sor árvasornak nevezzük, ezek megjelenése tiltható le. A címsorokat tartalmazó bekezdéseket célszerű „együtt a következővel” tulajdonsággal ellátni, így akadályozva meg azt, hogy a lap alján egy-egy cím egyedül álljon. Beállítható az is, hogy egy-egy bekezdés sorai egy oldalra kerüljenek, vagy megadható az is, hogy a bekezdés új oldalon kezdődjön, még akkor is, ha az előző oldalra bőven elférne.

28.4.2. Tabulátorok

A TAB billentyű lenyomásakor általában egy olyan vezérlőkarakter kerül a dokumentumba, amelynek hatására a beviteli pont a következő tabulátorpozícióra ugrik. A tabulátorpozíciók a szöveg bal bekezdésmargótól mért vízszintes elhelyezkedését, igazítását és az előtte levő hely kitöltésének módját határozzák meg.

A bekezdésekben akkor is vannak tabulátorpozíciók, ha egyet sem adunk meg. Az alapértelmezett pozíciók balra igazítottak, nem tartalmaznak kitöltő karaktert, távolságuk egymáshoz képest azonos. A tabulátorpozíciók helye, igazítási módszere, helykitöltő tulajdonsága általános esetben a szövegszerkesztők egy párbeszédablakából állítható be. Speciális beállítási lehetőségek programonként eltérőek lehetnek.

A tabulátorok segítségével könnyen készíthetünk egyszerűbb listákat és jegyzékeket, egyszerűbb táblázatokat. (Pl. a tartalomjegyzék oldalszámait jobbra igazíthatjuk.) Bonyolultabb táblázatok előállítására használjuk a szerkesztők táblázatkészítő szolgáltatásait.

28.4.3. Felsorolás és számozás

A szövegben gyakran használunk felsorolásokat. A felsorolt elemeket számozhatjuk vagy betűzhetjük, de azonos értékű fogalmak esetén használhatunk díszpontot, illetve egyéb más felsorolásjelet is. A számok után pontot, a betűk után kerek zárójelet teszünk. A magyar kiadói gyakorlat alapján a betűket kurziváljuk, azaz dőlt kiemeléssel látjuk el. A számok illetve a díszpontok helyzete, a felsorolás szövegének elhelyezkedése a bekezdésekre megadható behúzás típusokkal, némely esetben tabulátorral szabályozható.

A felsorolásokat a különböző szerkesztő programokban hasonló módon készíthetjük el. A felsorolás kialakításához általában több, előre beállított ún. *bajuszos listaformátum,* illetve számozott vagy betűzött formaelem közül választhatunk. Ha nem találunk megfelelő formátumú elemet, testreszabással bármelyiket módosíthatjuk, vagy akár újat is készíthetünk.

28.4.4. Szegély és mintázat

A bekezdések köré szegélyvonalat rajzolhatunk, és kitölthetjük a hátterüket. Amennyiben az előre elkészített szegélytípusok nem felelnek meg, a vonalak stílusa a keret egy-egy oldalára külön-külön is megadható. A szöveg és a szegély távolságát a belső margók megadásával állíthatjuk be.

A programok a szegélyezést a bekezdés tartalmazó hasábnak a két oldalra megadott behúzások által meghatározott szélességében végzik el, még akkor is, ha a bekezdés csak egyetlen szót tartalmaz. Ha a bekezdést csak a szöveg szélességében akarjuk szegélyekkel ellátni, akkor módosítsuk a behúzást.

Egymás utáni bekezdések köré csak akkor rajzoltathatunk közös szegélyt, ha a két bekezdés behúzási értékei azonosak. Ha különböznek, a bekezdések önálló szegélyeket kapnak. Ha mégis azt szeretnénk, hogy a bekezdések egyetlen keretbe kerüljenek, készítsünk egyetlen cellából álló táblázatot, tegyük a cellába a bekezdéseket, és a cellát szegélyezzük.

28.4.5. Iniciálé

Iniciálénak nevezzük a könyvfejezetek vagy folyóiratcikkek első, díszes, a többi betűnél általában nagyobb kezdőbetűjét. Az iniciálé alsó talpvonalát mindig egy sorba helyezzük valamelyik szövegsorral, a mellette lévő első sort vagy szót néha kiskapitális vagy nagybetűs formátumúra állítjuk. Gyakran látni újságokban olyan iniciálét, amely alatt nincs szöveg. Első sor behúzásával ez az effektus könnyen megvalósítható.

28.5. Karakterszintű formázás

Az alapvető karakterjellemzők közé a betűtípust, betűstílust és a betűméretet soroljuk. A betűtípuslistában a számítógépünkre telepített betűtípusokat láthatjuk. Az eszköztár betűtípusmezejében annak a szövegnek betűtípusa látható, ami ki van jelölve, vagy azé, amiben a beviteli pont van. Ha a kijelölt szöveg egynél több betűtípust tartalmaz, akkor ez a mező üres. Ha betűtípust akarunk állítani nyissuk le a legördülő listát a betűtípusmező melletti nyíllal, válasszuk ki a megfelelő elemet, vagy a mezőre való kattintás után gépeljük be a típus nevét, és a beállítás hatása a kijelölt szövegen azonnal megjelenik. Ha nem volt szöveg kijelölve, akkor a beviteli ponttól kezdődően a begépeléskor azzal a betűtípussal jelenik meg a szöveg, amit beállítottunk.

A karakterek formázása menüből is elvégezhető. A menüpont aktiválása után egy párbeszédablakban megjelenő típuslistából választható ki a betűtípus. Ezt követően a betűstíluslistában azok a stílusok jelennek meg, amelyek az adott betűtípushoz elérhetők. Ezek közül az egyik meg van jelölve, ha ez nem megfelelő, válasszunk egy másik stílust. Hasonlóan adhatjuk meg a betűméretet is. Választhatunk a listából a javasolt betűméretek közül, de 0,5 pt pontossággal akár be is gépelhetjük a kívánt méretet.

A betűk térközének és pozíciójának megadásával a karakterek egymáshoz viszonyított vízszintes elhelyezkedését (betűköz, betűpárok alávágása) és függőleges pozícióját (elhelyezés) állíthatjuk be. Az alávágás a

meghatározott karakterpárok közötti köz növelésének vagy csökkentésének, a betűköz egy szövegblokk lazábbá vagy tömörebbé tételének eszköze. InDesignban a szöveget automatikusan alávághatjuk metrikus vagy optikai alávágás használatával. A metrikus alávágás a betűtípusban megadott betűpárokhoz rendelt úgynevezett alávágási párokat használja, azaz két egymást követő betű alávágását a betűpárra megadott minta szerint végzi a program. Ha az alkalmazott betűtípus nem tartalmaz alávágásipár-definíciót, azaz az úgynevezett kerningtáblát, vagy ha egy soron belül két különböző betűcsaládot vagy -méretet állítunk be, célszerűbb az optikai alávágást használni, amely a szomszédos karakterek alakja alapján állítja be az alávágást. Az alávágás mértéke is megadható, az esztétikai igényeinknek megfelelő érték kísérletezéssel állítható be.

A verzállal – nagybetűvel – szedett címsoroknál a betűközöket általában ritkítjuk, és alávágást alkalmazunk. A karakterek emelésére, illetve süllyesztésére sorközi képek vagy képletek esetében lehet szükség.

Karakter szintű formátum a nyelv is, bár általában szavakra célszerű beállítani. A nyelvi ellenőrzés és az automatikus elválasztás megfelelő működésének alapfeltétele a nyelv helyes beállítása. És természetesen a jó működéshez elengedhetetlen a beállított nyelvhez tartozó szótárak, elválasztó algoritmusok megléte is, bár a beállítást nem általában nem a formátumbeállításhoz tartozó menüpontok között találjuk.

28.6. Stílusok használata

A dokumentumok egységes megjelenését legkönnyebben stílusok használatával biztosíthatjuk. A stílus betű-, bekezdés-, oldal-, képobjektum- vagy táblázatformázási tulajdonságok gyűjteménye, melyet egyetlen készletként őrzünk és nevezünk el. Egy-egy stílus alkalmazásakor a program a hozzá tartozó összes formázási tulajdonságát egyszerre állítja be azon az elemen, amire a stílust alkalmaztuk.

Az egyéni formázással szemben a stílusok nagyfokú rugalmasságot biztosítanak. Ha egy stílus tulajdonságait módosítjuk, a változás azonnal érinteni fog minden olyan dokumentumelemet, amelyet az adott stílussal formáztunk meg, ezáltal rengeteg időt takaríthatunk meg. Érdemes megjegyezni, hogy általában az egyedi beállítások felülbírálják a stílushoz tartozó beállításokat, így ezek a stílus megváltoztatásával sem változnak. Igaz, hogy a Word újabb verziói a beállítástól függően minden egyedi beállításkor a változásnak megfelelő stílust is generál, és ennek változtatása már az egyedi beállításokat is megváltoztatja.

28.6.1. Stílus alkalmazása

Az irodai szerkesztő programoknál minden új dokumentum tartalmaz előre elkészített stílusokat, melyek számát és beállításait a létrehozáskor felhasznált dokumentumsablon határozza meg. A Word például a bekezdésekre alapesetben a Normál vagy a Szövegtörzs nevű stílust alkalmazza mindaddig, amíg ezen nem változtatunk.

A stílusok alkalmazását megelőzően ki kell jelölnünk formázandó szövegrészt. Ha csak egy bekezdést vagy szót akarunk megformázni, helyezzük a beviteli pontot a megfelelő bekezdésbe vagy a szón belülre. A kívánt stílust – általában típusonként különböző – listából választhatjuk ki. Vegyes lista esetén a stílusok neve mellett betűstílusoknál **a**, bekezdésstílusoknál ¶ jel található.

28.6.2. Új stílus létrehozása

Az előre definiált stílusok mellett magunk is létrehozhatunk saját, úgynevezett *felhasználói stílusokat*. A stílus létrehozásához a létrehozás elindítása után megjelenő párbeszédablakban elsőként adjuk meg a stílus nevét, ha szükséges, a típusát, majd a stílus alapjának válasszuk ki azt a stílust, amelynek formai beállításait örökölni szeretnénk. Ha nem adunk meg alapstílust, *önálló stílust* hozunk létre. Ekkor a stílushoz tartozó minden tulajdonságot definiálni kell, kivéve, ha a felajánlott tulajdonságok megfelelnek. Ellenkező esetben csak az alapstílustól való eltéréseket kell definiálnunk. A következő bekezdés stílusa mezőben az is megadható, hogy milyen legyen annak a bekezdésnek a stílusa, amely az adott stílusú bekezdés ENTER-rel való lezárásakor kezdődik.

A stílusokhoz billentyűparancsot rendelhetünk. Ennek segítségével egy szöveget rövidebb idő alatt formázható meg.

28.6.3. Stílus módosítása és törlése

A stílusok nagyfokú rugalmasságot biztosítanak a formázásban. Ha utólag úgy érezzük, hogy valamelyik stílus nem megfelelő, és emiatt átdefiniáljuk, a változás általában azonnal megjelenik minden szövegrészen, amelyet az adott stílussal formáztunk meg. Az egyedi formázásokra a változás nem történik meg.

Ha valamely stílusra már nincs többé szükségünk, töröljük. Amikor használatban lévő bekezdésstílust törlünk, a programok az alapértelmezett stílust alkalmazzák az érintett bekezdésekre. Betűstílus törlésekor az érintett szövegrészek karakterformátumai a bekezdés stílusában definiált formátumokra változnak.

28.6.4. Stílusok másolása dokumentumok között

A legtöbb alkalmazás lehetővé teszi a stílusok átvitelét a dokumentumok között. Ez történhet szervező segítségével vagy stílusok exportálásával és importálásával. A műveletet körültekintően kell elvégezni akkor, ha olyan stílust importálunk, amely olyan stílust definiál át, amelyik alapja egy vagy több másiknak. Exportnál hasonló probléma merülhet fel abban dokumentumában. ahova az export irányul.

A stílusokat a Vágólap felhasználásával vagy „fogd és vidd” technikával is átvihetjük az egyik dokumentumból a másikba, amennyiben az adott nevű stílus nem létezik a céldokumentumban. Ha a másolt stílus létezik, a szöveg a céldokumentum beállításainak megfelelően formázódik.

28.7. Dokumentumsablonok

Az előzőekben ismertettük, hogy a stílusokat milyen módon vihetjük át egyik dokumentumból a másikba, azonban ennek a módszernek megvannak a maga korlátai. Például egy több dokumentumból álló kiadvány formátumának kialakítása, vagy egy könyvsorozat elkészítése ezzel az eljárással már kényelmetlenné válhat. A megoldást a sablonok adják.

A sablonokban rögzíthetjük azokat a dokumentumelemeket, amelyek rendszeresen ismétlődnek a munkánkban (pl. levelek fejléce, könyvek címnegyedíve stb.), definiálhatjuk a formázáskor használandó betű- és bekezdésstílusokat. A sablonok emellett szövegtárelemeket, makrókat, eszköztárakat, menüket és gyorsbillentyűket is tartalmazhatnak.

A sablonok készítése a dokumentumokhoz hasonlóan történik. Miután befejeztük a munkát, adjuk ki a **Mentés másként...** parancsot, majd fájltípusnak válasszuk ki a dokumentumsablont.

29. A kiadványok felépítése és elemei

29.1. A kiadványok felépítése

A kiadványok az esetek többségében három fő részből állnak: *főszövegből, járulékos részekből* és *borítóból.* A főszöveg a dokumentumok legfontosabb, tartalmi része, amely a szerző érdemi mondanivalóját tartalmazza. A járulékos részek kiegészítik, magyarázzák a főszöveget, illetve segédletekkel látják el a kiadványt. A járulékos részek vagy a főszöveget megelőző, vagy a követő oldalakra kerülhetnek.

A járulékos részek sorrendje nem szigorúan kötött, azonban a könyvkiadás története során kialakultak bizonyos konvenciók. Főszöveget könyvek, periodikumok esetén megelőzi a címnegyedív, az ajánlás, a mottó, az előszó és a köszönetnyilvánítás; főszöveget követő járulékos rész az irodalomjegyzék, a bibliográfia, a függelék, az utószó, a kronológia, a szakkifejezések jegyzéke, a névmagyarázatok, a rövidítések jegyzéke, az illusztrációk jegyzéke, a kötetben közölt írások jegyzéke, a mutatók, az idegen nyelvű összefoglaló, valamint az információs és kereskedelmi oldalak. A tartalomjegyzéknek a magyar hagyományban nincs szigorúan rögzített helye, tudományos művekben általában a címnegyedív után helyezzük el, szépirodalmi művek esetén általában a kolofonoldal elé szúrjuk be. A mellékletek kerülhetnek a könyv különböző oldalaira elszórtan, vagy együtt a könyv végére.

Időszaki kiadványoknak nevezzük azokat a sorszámozással vagy keltezéssel ellátott, időközönként megjelenő nyomdatermékeket, amelyek egymás után következő számokból, évfolyamokból, kötetekből, illetve füzetekből állnak. Az időszaki kiadványoknak állandó, ismétlődő részegységeik vannak (pl. rovat), jellegben és tartalomban hasonlítanak egymáshoz. Bár a napi- és hetilapok, valamint folyóiratok elrendezése, formai kialakítása a könyvekétől eltérő, a korábban bemutatott elvek ezekre a kiadványokra is érvényesek.

29.1.1. Címnegyedív

A címnegyedív a könyv első ívének egynegyede (első négy oldala): *szennycímoldal, sorozatcímoldal, címoldal* és *copyrightoldal.* A címnegyedívet nem paginázzuk (vagyis nem látjuk el oldalszámmal), de ezek az oldalak is beleszámítanak a könyv terjedelmébe.

A szennycímoldalra a könyv *szerzője, főcíme* és többkötetes mű esetén a *kötet száma* kerül. Egyszerzős mű esetén a teljes nevet kiírjuk, két-három szerzőnél elegendő a szerzők vezetékneve nagykötőjellel elválasztva. Ha a könyvnek háromnál több szerzője van, a szennycímoldalon a főcím önállóan, a szerzők neve nélkül is szerepelhet.

A sorozatcímoldal a sorozatba illeszkedő művek esetén a sorozat adatait tartalmazza, azaz itt tüntetjük fel a *sorozatcímet,* a *sorozatszerkesztő nevét,* az *alsorozatcímet,* a *sorozatszámot,* illetve a *sorozatot gondozó intézmény nevét,* ha az nem azonos a kiadóval.

A címoldalon helyezzük el a bibliográfiai hivatkozás öt alapadatát, vagyis a *szerző nevét,* a *könyv címét,* a *kiadó nevét,* a *kiadás évét* és a *megjelenés helyét,* továbbá a *kötetszámot* és a *kiadásjelzést.* A magyar hagyomány szerint a szerzőket ábécérendben, tudományos fokozat, illetve betöltött pozíció nélkül, gondolatjellel (azaz a nagykötőjel előtt és után szóközzel) elválasztva szokás feltüntetni. Amennyiben a szerzők helyett a szerkesztőt adjuk meg, nevét a cím alatt helyezzük el. A kiadásjelzés a kiadás számát és minőségét *(változatlan, bővített, javított)* közli. Az első kiadás tényét sohasem tüntetjük fel a könyvön.

A copyrightoldalra kerülnek a könyv létrejöttében szerepet játszó személyek és testületek (például szerkesztő, fordító, kontrollszerkesztő, illusztrátor, lektor stb.), a szerzői jogok jelölése és azok az adatok, amelyeket nem tudunk a címoldalra elhelyezni (például háromnál több szerző felsorolása, fordítás esetén a mű eredeti címe stb.).

29.1.2. Tartalomjegyzék

A tartalomjegyzék a könyvek belső címrendszerét, a járulékos részek címeit (kivéve címnegyedív, ajánlás, mottó, kolofonoldal, információs és kereskedelmi oldalak) és kezdő oldalszámait tartalmazza. Egyszerűbb szerkezetű könyveknél a főszöveg címeit antikvából szokás szedni, a járulékos részekét kurzívból, azonban a bonyolultabbaknál, ahol a kurziválásnak más szerepe van, egyéb tipográfiai megoldást kell alkalmazni.

A tartalomjegyzékben lehetőleg minden címet tüntessünk fel, de ügyeljünk arra, hogy a túlságosan terjedelmes tartalomjegyzék nehezíti a tájékozódást. Amennyiben meglehetősen sok a belső cím, a negyedrangú és annál alacsonyabb címeket elhagyhatjuk.

A tartalomjegyzék-összeállítás a kiadvány- és szövegszerkesztőkben hasonló módon, stílusok, fejezetszintek és tartalomjegyzék-bejegyzés mezők segítségével történik. Az első lépés mindig a címsorok megjelölése, majd ezt a jegyzék beszúrása és formázása követi. Az összeállított tartalomjegyzékben a különböző címfokozatokat számok használatával vagy tipográfiai eszközökkel (behúzás, betűfajták és -fokozatok) szokás megkülönböztetni, ez utóbbi esetben két-három betűfajtánál ne használjunk többet.

29.1.3. Irodalomjegyzék

Az irodalomjegyzék, amely a mű írása során áttekintett és felhasznált irodalmat tartalmazza, elengedhetetlen része a tudományos munkáknak, szakkönyveknek, tankönyveknek. A jegyzék alapegysége, a *bibliográfiai tétel,* minimálisan öt adatból áll: a szerző neve, a könyv címe, a kiadó neve, a megjelenés helye és a megjelenés éve. Ezekhez az adatokhoz többkötetes műveknél hozzájön a kötetszám, illetve nem első kiadású könyveknél a kiadásszám. Ha a könyvnek csak egy meghatározott részére utalunk, az oldalszámokat is fel kell tüntetnünk. A bibliográfiai tételbe a kötet terjedelmét és a sorozat címét is felvehetjük.

A leírásban a szerző neve után kettőspontot teszünk. A címet és az alcímet – amit a könyv címoldaláról olvassunk le – kurziváljuk, utána vesszőt vagy pontot teszünk. Ezután következnek vesszővel elválasztva a megjelenésre vonatkozó adatok: a megjelenés helye és éve, valamint a kiadó neve.

Gál Tibor: *A web programozása*, Műegyetemi Kiadó, Budapest, 2006.

Terjedelmi vagy hivatkozási adatok közlésekor a kiadó neve után vessző kerül. Ha a teljes könyvre utalunk, az utolsó számozott oldalt vegyük figyelembe. Az oldalszám után nincs pont, a pagina (p.), illetve az oldal (o.) rövidítése egyaránt helyes.

Gál Tibor: *A web programozása,* Műegyetemi Kiadó, Budapest, 2006, 636 p.

Ha egy könyv adott oldalára vagy meghatározott részére hivatkozunk, az oldalszám után pontot teszünk.

Gál Tibor: *A web programozása,* Műegyetemi Kiadó, Budapest, 2006, 156–165. p.

A periodikumok cikkeinek bibliográfiai leírásában általában a szerző és a tanulmány címe után kurziválva a kiadvány neve szerepel, majd ezután következnek az egyértelmű azonosításhoz szükséges adatok. Ettől az

egyes periodikumok eltérhetnek, és egyéni szabályok szerint kérhetik a leírást.

Bodri Gabriella: Pelletfűtésű kandallók. *Szép házak,* XIV. évf. (2005) 5. sz. 104–105. p.

Az irodalomjegyzék formailag többféle lehet. A *sorszámozott* bibliográfia arab számokkal ellátott tételeire a folyószövegben a szögletes zárójelbe tett sorszámmal hivatkozhatunk (például [25]). Az *alfabetikus* bibliográfiában a tételek a szerzők neve alapján ábécérendben szerepelnek. Ha egy szerzőnek több munkája is megtalálható a jegyzékben, a másodlagos rendezési szempont a megjelenés éve. A szerző nélküli könyveket a névelő nélküli címük szerint soroljuk be. A *kronologikus* bibliográfia a megjelenés éve szerint rendezett. Az *évszámkiemelő* bibliográfiát akkor készítünk, ha a főszövegben szövegközi zárójeles hivatkozásokat használunk (Gál 2006). A bibliográfiai tételekben a kiadás éve közvetlenül a szerző neve után szerepel, sorrendjük alfabetikus.

Gál Tibor: *A web programozása.* Budapest, Műegyetemi Kiadó.

A *rendszerező* bibliográfia tételei forrástípusok, országok, nyelvek vagy témák szerint vannak csoportosítva. A csoportokban a sorrend alfabetikus.

29.1.4. Mutatók

A mutatók a főszövegben és a függelékben szereplő fontosabb neveket, fogalmakat, kulcsszavakat és -kifejezéseket, illetve évszámokat sorolják fel az oldalszám feltüntetésével. A mutató típusa szerint lehet személynévmutató, tárgymutató, földrajzi nevek mutatója, évmutató stb., de kisebb munkákban össze is vonhatjuk őket (például név- és tárgymutató).

Az egyes tételek besorolásánál az általános magyar betűrendet használjuk. A névelőket csak az irodalmi művek mutatójában vesszük figyelembe a sorrend kialakításakor. Az oszlopos mutatókban az egyes főtételek alatti altételeket is ábécésorrendbe állítjuk. Az altételekben a főtételek szavait a sor elején nem ismételjük, hanem ˜ jellel helyettesítjük. sorba rendezésnél a sorrend kialakításában a tilde (˜) jellel jelölt tárgyszó is beleszámít.

A mutatókban az összevont tételek kétféle formában helyezhetők el. Az *oszlopos mutatóban* a főtétel és az altételek külön sorokban helyezkednek el. Az ún. *bokrosított mutatóban* a tételeket csoportosítva adjuk meg,

az egyes csoportok a bokrok, egy-egy bokor egy-egy bekezdést alkot. Az egy bekezdésbe kerülő tételeket pontosvesszővel választjuk el egymástól.

A tételekben a címszó és az utolsó oldalszám után nem teszünk írásjelet, azonban az oldalszámokat vesszővel választjuk el egymástól. Ha kettőnél több egymást követő oldalra hivatkozunk, a kötőjeles formát alkalmazzuk (például 41–43). A mutatón belül utalhatunk egy másik címszóra is a *lásd, lásd még, vesd össze* kifejezések használatával. Az utalószót mindig kurziváljuk, és pontosvesszővel választjuk el az előtte álló oldalszámtól.

A mutató elkészítése a tárgyszavak összeállításával kezdődik. Lehetőleg ezt a feladatot a szerző végezze el. Miután a dokumentumban a szerkesztőprogram szabályainak megfelelően megjelöltük a megfelelő szavakat és kifejezéseket, a tartalomjegyzékhez hasonló módon beszúrjuk a megfelelő formátumú összeállítást. A program összegyűjti a tárgymutató-bejegyzéseket, sorba rendezi őket, beírja a megfelelő oldalszámokat, megkeresi és törli az azonos oldalra mutató ismétlődő bejegyzéseket, végül megjeleníti a mutatót.

29.2. Dokumentumelemek

29.2.1. Élőfej, élőláb

A könyvek részeit különböző technikai kellékek egészíthetik ki, ezek közé tartozik az élőfej és az élőláb, amelyek a tájékozódást könnyítik. Ez az elem szakaszokhoz rendelve definiálható, a páros és páratlan oldalra is eltérő formátummal. A kiadványokban a szokásos sorszámozás szerint a páros oldal a bal, a páratlan oldal a jobb oldalra kerül a kiadványban. A bal és a jobb oldali élőfej kialakítása általában a következő: ha a kötetben különböző szerzők művei szerepelnek, akkor bal oldalt a szerző, jobb oldalt a cím; ha egy szerző van, bal oldalt a mű címe, jobb oldalt a fejezet címe, illetve bal oldalon a fejezetcím, jobb oldalon a fejezeten belüli alcím – ennek hiányában az oldalon érintett téma megjelölése – szerepel.

Az élőfej hagyományosan beleszámít a szedéstükörbe, de az oldalszám, ha az élőfejben helyeztük el – nem. Érdemes megjegyezni, hogy a szöveg- és kiadványszerkesztők egy része a szedéstükörhöz számolja az élőfejet, más része nem. Az oldalszám az élőfejben kívülre kerüljön, a szöveg középre vagy belülre. A szótárakban és lexikonokban az élőfej a címszó szerinti keresést szolgálja, itt az oldalszám van belül és a szöveg kívül. Kéthasábos szedésnél az élőfej mindkét szélén más címszó szerepelhet.

Az oldalszám legtöbbször az élőlábba kerül középre vagy kívülre igazítva. Belülre soha ne tervezzünk oldalszámot!

Az élőfej és élőláb megtervezésénél a legfontosabb szempont a folyószövegtől való távolság megfelelő megválasztása, amely körülbelül félsornyi, vagy annál kicsivel több, de egy sornál kevesebb legyen. Ha léniát is használunk, akkor a lénia alatti és fölötti távolságot lehetőleg ugyanakkorára állítsuk. Az élőfej szövegét lénia hiányában antikva helyett szedjük a folyószövegnél két betűfokozattal kisebb verzállal, vagy legyen kiskapitális.

29.2.2. Idézetek

Másoktól „kölcsönzött” gondolatokat, mondatokat szó szerint csak idézőjelek között, a forrásmegjelölésével írjunk le. A magyar helyesírási szabályzat háromféle idézőjelet javasol. A szövegbe ékelt, szó szerinti idézetnél *„macskakörmöt”* használunk. Az idézeten belüli idézeteket *»lúdlábbal«* jelöljük. A felső állású, ún. *’félidézőjelet’* nyelvészeti, filológiai munkákban kifejezések, mondatok jelentésének megadására, illetve egyéb szövegekben hármas idézés esetén alkalmazzuk.

Az idézett szövegnek sem a szórendjét, sem a helyesírását nem változtathatjuk meg. Az idézetben a kihagyásokat három ponttal jelöljük, de ha utólag írunk be valamit, azt tegyük szögletes zárójelbe. Az idézetekben talált hibát a szögletes zárójelbe foglalt latin [sic!] (így!) szócskával jelezzük.

29.2.3. Utalások

A folyószövegben gyakran utalunk a szöveg más helyére vagy más szövegrészekre. A sorközi utalásokat a megfelelő helyen kerek zárójelben közöljük. Az utalásban a *lásd* kifejezés azt jelenti, hogy az ott található szöveg, illusztráció kiegészíti, tovább részletezi a leírtakat, a *vesd össze* hasonló gondolatra vagy problémára hívja fel figyelmünket. Ha a szöveghelyhez közeli ábrára, táblázatra utalunk, elegendő – a lásd és a vesd össze nélküli – rövid forma is.

(4. ábra)
(lásd 15. fej.)

(lásd alább a 3. táblázatot)
(vö. 7. lábj.)

29.2.4. Illusztrációk

A táblázatok mint illusztratív elemek vonalakkal határolt, oszlopokba rendezett adatsorok, amelyek kiegészítik és szemléltetik a szöveget. A táblázatok rovatokból (cellákból) állnak, amelyek közül egyesek feliratokat, míg mások adatokat tartalmaznak.

A *táblázatfejben* találhatók általában az oszlopok elnevezései, ezeket nagy kezdőbetűvel kezdjük, utánuk nem teszünk pontot. Ha a fej összetett (azaz a fejben további osztások vannak), csak az első rovat szókezdő betűje nagybetű. Az oldalrovatokba kerülnek a sorok nevei, ezek írásmódja a fej feliratainak megfelelő legyen.

A mennyiség neve	A származtatott egység	
	neve	jele

A fej által megjelölt adatok a fejlezáró lénia alatti táblázatrészben, a *lábban* találhatók. A láb sorai a megnevezéseket tartalmazó *oldalrovatból* és az *adatrovatokból* állnak. A sorok és oszlopok numerikus adatainak összege – ha ez szükséges – az *összesítőoszlopba* és az *összesítősorba* kerülnek.

Az ábrák a folyószöveg állításait illusztrálják: bemutatják egy gép szerkezetét, szemléltetik egy folyamat lezajlását stb. A szemléletes ábrák akár többoldalas leírásnál is pontosabban kifejezhetik a mondandót.

Az illusztrációkat a könnyebb azonosítás érdekében sorszámmal látjuk el. A számozás történhet fejezetenként is.

3. táblázat
2.5. ábra

Amennyiben az illusztrációnak címe is van, akkor az elnevezés után pontot kell tenni. Az ábra, táblázat stb. szavakat félkövér betűből is szedhetjük.

9. ábra. A nyersolaj világpiaci árának változása 2010-ben

Gyakran előfordul, hogy az ábra több részből áll, vagy egyes részeit betűkkel, számokkal jelöljük, ilyenkor a magyarázó szövegek közé vessző kerül.

3. ábra. A betűformák és a technológia
a) Magasnyomás, *b)* számítógépes szedés

29.2.5. Képletek

A matematikai, fizikai és kémiai képletek szedésére speciális szabvány vonatkozik. A szabvány betartása a nagyobb kiadókban a műszaki szerkesztők feladata. Az alábbiakban csak a legfontosabb szabályokat ismertetjük.

1. „Egyenleteket tartalmazó könyvek alapszövegét olyan betűtípusból szedjük, amely a legteljesebb matematikai jelekkel rendelkezik (pl. Times). Erre a célra az antikva (álló) betűfajta a legmegfelelőbb. A képletek betűtípusa mindenképpen, a fokozata lehetőség szerint egyezzen meg a szöveg betűjével.

2. A számokat négy jegyig egybe, e fölött hármas csoportokba kell írni.

$$1986$$
$$2\,345\,749$$

3. Antikvából szedjük a számokat, az írásjeleket (zárójeleket, egyenlőségjeleket stb.), a mértékegységeket (kg, cm, m^2 stb.), a kémiai vegyjeleket, valamint a rövidítéseket (sin, cos, tg stb.).

$$(5+7)^2 = 144$$

4. Kurzívból szedjük a számokat helyettesítő betűket.

$$(a+b)+c = a+(b+c)$$

5. A kitevők, indexek, zárójelek előtt nincs betűköz. A műveleti jelek előtt és után azonban kell (ennek értéke 2 pont). Egyenlőségjel előtt és után fél kvirt (fél négyzet) betűközt tegyünk.

$$a^2 = (b + c)^2$$

$$f_1(y),\ f_2(y)$$

6. A zárójelek általánosan elfogadott sorrendje: gömbölyű, szögletes, kapcsos, horpadt. A zárójelek méretét a közéje zárt képlet határozza meg.

$$\langle\{[()]\}\rangle$$

$$\left(\frac{a}{b+c}\right) - y$$

7. Összeadáskor, kivonáskor az azonos helyi értékű számok kerüljenek egymás alá, továbbá ügyeljünk arra, hogy a lénia a számoszlop hosszával egyezzen meg.

$$\begin{array}{r} 235\ 728 \\ +\ 337\ 219 \\ \hline 572\ 947 \end{array}$$

8. A törtvonalakat és a műveleti jeleket az egész számok középvonalába kell szedni.

$$1\frac{1}{2} + 2\frac{1}{2} + 3\frac{1}{2} = 7\frac{1}{2}$$

9. A képletek elválasztásakor a műveleti jeleket nemcsak a sor végén, hanem a sor elején is el kell helyezni. Egymás alatt álló, logikailag összetartozó képletek esetén az egyenlőségjelek egymás alatt álljanak.

10. Tördeléskor ügyeljünk arra, hogy az oldal élére ne kerüljön képlet, azaz a képletet megelőző magyarázó szöveg egy része mindenképpen kerüljön át az új oldalra.” (Gyurgyák 2005)

A szövegszerkesztőkhöz, kiadványszerkesztőkhöz többnyire hozzárendelhetők olyan segédprogramok (például Mathtype Eqution, Microsoft Eqution stb.), amik a képletek szerkesztésére készültek, Ezek a programok

automatikusan gondoskodnak a fenti követelmények jó részének teljesüléséről. Más részük pedig a programban beállítható. Használjuk ezeket a segédprogramokat! Ha az egyenletszerkesztő segédprogramok használata mellett döntöttünk, akkor ne készítsünk a dokumentumunkban olyan képleteket, sőt egyedül álló matematikai szimbólumokat se, amik nem ezzel készültek, mivel a kétféle módszerrel szerkesztett képletek stílusa a legritkább esetben egyezik meg. Ez az olvasóban bizonytalanságot okozhat, és esztétikai szempontból sem megfelelő.

29.2.6. Jegyzetek

A jegyzetek a folyószöveget kiegészítő rövid megjegyzések, irodalmi hivatkozások, amelyek anélkül közölnek a témával kapcsolatos plusz információt, hogy a főszöveg gondolatmenetét megtörnék. A jegyzetek szoros kapcsolatban állnak az irodalomjegyzékkel, azzal összhangban kell lenniük.

A jegyzetelési formát a könyv funkciója és típusa határozza meg. Tudományos művekben *láb-, fejezet vagy szöveg végi jegyzetet* alkalmazunk, esszékben, szépirodalmi és ismeretterjesztő művekben – néhány bibliográfiai hivatkozás esetén – *szövegközi jegyzeteket*. Ha egy-két tucatnál több jegyzet van, de számuk jóval száz alatt marad, és ezek kizárólag irodalmi hivatkozások, akkor használhatunk szövegközi, *sorszámozott bibliográfiai jegyzeteket,* melyeknél a hivatkozott mű bibliográfiai sorszámát szögletes zárójelben adjuk meg. A *zárójeles jegyzetek,* ahol a szerző vezetéknevét és a mű megjelenési évszámát, szükség esetén az oldalszámot a folyószövegben kerek zárójelben adjuk meg, szintén csak irodalmi hivatkozásokhoz használhatók. Alkalmazásukkor az irodalomjegyzék csak alfabetikus lehet, és a főszöveghez fűzött kiegészítéseket csak lábjegyzetben helyezhetjük el. Ne keverjük a különféle megoldásokat, egy dokumentumban egyféle módszerrel dolgozzunk.

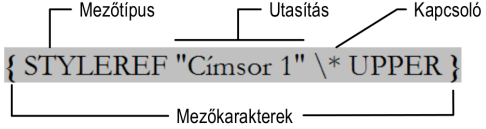
A jegyzetszámokat és a nem szövegközi jegyzeteket mindig kisebb betűfokozattal szedjük a folyószövegnél. Ha a könyvben a jegyzetek száma száz alatt marad, számozásuk a hagyomány szerint – a teljes kötetre – folyamatos lehet, egyébként fejezetenként újrakezdődő legyen. A folyószövegben jegyzetszámokat általában az írásjel után tesszük, de ha a jegyzet csak egy bizonyos szóra, illetve zárójelek vagy gondolatjelek közé zárt szövegre vonatkozik, a szám az írásjel elé kerül. A jegyzetszám mindig tapadjon az előtte álló karakterhez.

29.2.7. Mezők

A mezők a dokumentum változó adatainak helyőrzői. Tartalmuk utasítás a szerkesztőprogramnak, amit az értelmezve végrehajtja a megfelelő feladatot, amely sokféle lehet: az oldalszám vagy dátum beillesztésétől egészen bonyolult dolgokig, mint például egy tartalomjegyzék vagy tárgymutató összeállítása.

A mezők kezelését az egyes programok különbözőképpen valósítják meg. Míg Writerben a mezőtulajdonságok csak párbeszédablakok segítségével módosíthatók, a Word-mezők közvetlenül szerkeszthetők a dokumentumban, ennek következtében akár egymásba is ágyazhatók.

A Word mezőit az ún. *mezőkarakterek*, valamint az ezek közé írt *mezőtípus* és az *utasítások* alkotják. Ezeken kívül számos mező eredményét szabályozhatjuk *mezőkapcsolók* megadásával. Amennyiben egy mező utasításában egy vagy több másik mező is szerepel, a program a mezők kiértékelését a legbelsővel kezdi.



A Word mezőkarakterei külsőleg a kapcsos zárójelekhez hasonlítanak, de közvetlenül csak a CTRL+F9 gyorsbillentyűvel vagy a mezőbeszúró paranccsal vihetők a dokumentumba.

A mezők funkciójuk alapján három csoportba sorolhatók:

1. *Eredménymező* – Valamilyen eredményt jelenítenek meg a dokumentumban. Például a NUMPAGES mező az oldalak számát adja.
2. *Jelölőmező* – Ezeknek a mezőknek nincs látható eredménye, különböző szolgáltatásokhoz szükséges adatokat tárolnak vagy állítanak elő.
3. *Akciómező* – Az ilyen típusú mezők hatására a szerkesztőprogram valamilyen műveletet hajt végre. Pl. a GOTOBUTTON mező segítségével a dokumentum egy adott részére ugorhatunk. A mezőtől függően a műveletet a program automatikusan vagy csak kérésre hajtja végre.

Az eredménymezőket kétféleképpen jeleníthetjük meg: az egyik üzemmódban a mezőkódot láthatjuk, a másikban az eredményt. (Wordben a megfelelő üzemmód a Beállítások párbeszédablak Megjelenítés oldalán állítható.) A mezőeredmények nyomtatáskor automatikusan aktualizálódhatnak, de a kijelölt mezők szerkesztéskor kérésre is frissíthetők. Amennyiben egy mező eredményét meg szeretnénk őrizni, zárolásával akadályozzuk meg a frissítését. Később a mezőfrissítést ismét engedélyezhetjük. Irányított beillesztéssel vagy a mező hivatkozásának megszüntetésével a mezőeredmény normál szöveggé alakítható, ekkor csak az eredmény marad, a mezőben tárolt parancs eltűnik!

A mezők szinte minden dokumentumban előfordulnak. Gondoljunk csak arra, hogy az automatikus oldalszámozás is mezők segítségével történik. Szerkesztésük, átalakításuk viszont komolyabb szaktudást és némi programozási ismeretek igényel.

Önellenőrzés

1. Jelölje meg a formázási egységeket!
   - ☐ folyószöveg
   - ☐ címsor
   - ☐ bekezdés
   - ☐ hasáb
   - ☐ táblázat
   - ☐ karakter
   - ☐ élőfej
   - ☐ oldalszám
   - ☐ képlet
2. Számítsa ki a leírás alapján az oldalbeállításhoz szükséges hiányzó adatokat! A papírméret $14{,}8 \times 21$ cm^2, a szedéstükör 100×140 mm^2, a külső margó kétszerese a belsőnek, az alsó pedig 2 cm-rel nagyobb a felsőnél.
   A belő margó mérete: ☐ mm
   A felső margó mérete: ☐ mm
3. Egészítse ki az alábbi mondatot!
   A ☐ egy bekezdés utolsó sora, amely egymagában kerül az oldal tetejére.

4. A felsorolt lehetőségek közül válassza ki azt, amelyik csak bekezdésformátumokat tartalmaz!

- ☐ szegély távolsága a szövegtől, sűrített betűköz, térköz utána 6 pt, fattyú- és árvasorok engedélyezése, igazítás
- ☐ bal behúzás, sorköz, felsorolás, elválasztás nélkül, számozás
- ☐ nagybetűs, térköz előtte 1 sor, nincs elválasztás, új oldalra, sorkizárt igazítás
- ☐ aláhúzás, betűpárok alávágása, emelt elhelyezés, rejtett
- ☐ behúzás balról, sorköz, tabulátor, oldalszegély, mintázat
- ☐ egy oldalra, szegély színe, jobb behúzás, középre zárt igazítás, iniciálé elhelyezése
- ☐ vázlatszint, tabulátorpozíció, dupla sorköz, új oldalra, függő behúzás
- ☐ aláhúzás, behúzás balról, sorköz, szegély, hasábok száma
- ☐ együtt a következővel, térköz, első sor behúzása, jobb margó, kitöltés

5. A címnegyedív melyik oldalára kerülnek a megadott adatok? Írja az oldalak sorszámát a mezőkbe! (Több lehetséges oldal esetén a számokat növekvő sorrendben, közvetlenül egymás után adja meg!)

1. szennycímoldal
2. sorozatcímoldal
3. címoldal
4. copyrightoldal

könyv szerzője: ☐

kiadó neve: ☐

illusztrátor neve: ☐

kötetszám: ☐

fordító neve: ☐

sorozatszerkesztő neve: ☐

szerzői jogok jelölése: ☐

kiadásjelzést: ☐

lektor neve: ☐

6. Egészítse ki az alábbi mondatot!
A tartalomjegyzékben egyszerűbb szerkezetű könyveknél a főszöveg címeit ☐ szedjük, a járulékos részekét kurzívból.

7. Hol választjuk el egymástól pontosvesszővel a bekezdésbe kerülő tételeket?

- ☐ személynévmutató
- ☐ tárgymutató
- ☐ oszlopos mutató
- ☐ földrajzi nevek mutatója
- ☐ bokrosított mutató
- ☐ évmutató
- ☐ tartalomjegyzék
- ☐ kisebb mutatók

8. Egészítse ki az alábbi mondatot!
A sorközi utalásokat a megfelelő helyen ☐ közöljük.

Modulzáró feladatok

9. Az alábbiak közül mi található meg a vá(r|g)ok reguláris kifejezésben?

- [ ] literális karakter
- [ ] számszerűsítő
- [ ] illeszkedési pozíció
- [ ] vezérlőkarakter
- [ ] elágazás-szétválasztó
- [ ] csoport
- [ ] karaktercsoport
- [ ] metakarakter

10. Számítsa ki a leírás alapján az oldalbeállításhoz szükséges hiányzó adatokat! A papírméret 176×250 mm^2, a szedéstükör 122 mm széles. Könyvfűzés után a vágáskor három oldalon leesik 8 mm. A vágás után azonos méretű margókat kell kapnunk.

A belő margó mérete vágás előtt: _____ mm

A felső margó mérete vágás után: _____ mm

11. Csoportosítsa az alábbi elemeket! elemek: lénia, táblázat, címsor, fénykép, két bekezdést elválasztó nyomdai dísz, margóra elhelyezett izzót ábrázoló kép, térkép egy regényben, képaláírás, létrát ábrázoló iniciálé mint A betű, margóra festett kalocsai motívum, oldalszám, térkép a történelemkönyvben, körzet
csoportok: tipográfiai elem (1), grafikai elem (2), illusztratív elem (3), díszítő elem (4)

Megoldás:

	képaláírás
	oldalszám
	címsor
	lénia
	körzet
	táblázat
	két bekezdést elválasztó nyomdai dísz
	margóra elhelyezett izzót ábrázoló kép
	létrát ábrázoló iniciálé mint A betű
	fénykép
	térkép a történelemkönyvben
	térkép egy regényben
	margóra festett kalocsai motívum

12. Tegye ki a relációjelet a méretek közé!

betűszem ☐ törzsméret ☐ származás

13. Az alábbiak közül jelölje meg a szakasz szintű formátumokat!

- ☐ papírméret
- ☐ sorköz
- ☐ a belső margó mérete
- ☐ betűstílus
- ☐ sorkizárás
- ☐ a helyesírás és a nyelvtan ellenőrzésének mellőzése
- ☐ automatikus elválasztás tiltása/engedélyezése
- ☐ vázlatszint

14. Tegye megfelelő sorrendbe a bibliográfiai tétel alapadatait! (A betűket egymás után, vessző és szóköz nélkül adja meg!)

a) megjelenés helye
b) szerző neve
c) kiadó neve
d) megjelenés éve
e) könyvcím

Megoldás: ☐

15. Válaszoljon az alábbi kérdésekre!

A dokumentumok formázásakor hányféle formázási egységet különböztetünk meg?

☐

Mi a legkisebb formázható elem?

☐

16. Döntse el az alábbi állításokról, hogy igaz vagy hamis! (I/H)

☐	A hasábok szélességének megadásánál figyelembe kell venni a lap méretét.
☐	A szöveg lap szélétől való távolságát csak a margók mérete határozza meg.
☐	A bekezdések között lévő térköz méretét az aranymetszés szabálya szerint kell beállítani.
☐	Jobbra igazított tabulátorpozíciónál a sorok vége a jobb margóhoz kerül.

17. Mivel jelöljük a szövegekben a hármas idézést?

- ☐ hiányjel
- ☐ kvirtmínusz
- ☐ ligatúra
- ☐ felső állású félidézőjel
- ☐ macskaköröm
- ☐ lúdláb
- ☐ alsó állású félidézőjel
- ☐ díszpont

18. Egészítse ki az alábbi mondatot!

A zárójeles jegyzetek, ahol a szerző ☐ és a mű megjelenési évszámát, szükség esetén az ☐ a folyószövegben kerek zárójelben adjuk meg, szintén csak irodalmi hivatkozásokhoz használhatók.

V. MODUL

Táblázatkezelés modul

Az Informatika I. jegyzet táblázatkezelés modulja egy negyedéves kurzus oktatási anyagát tekinti át. A felépítés során az alapoktól indultunk, hiszen fontos célunk volt az ismeretek korrekt elméleti megalapozása, tisztázása. Emellett már a kezdeteknél elterveztük azt is, hogy több szoftvereszközön fogjuk bemutatni a megoldásokat, ezzel is hangsúlyozva azt, hogy a táblázatkezelés „filozófiája” eszközfüggetlen.

A projekt célkitűzésének megfelelően a jegyzetet önálló tanulásra szántuk. A leckék elején rövid tájékoztatást/segítséget adunk a feldolgozáshoz, a leckék végén pedig hasonló módon kérdések, feladatok segítik a tanulót abban, hogy visszajelzést kapjon a tanulási folyamat sikeréről. Fontos azonban hangsúlyozni, hogy mindez nem garantálja a teljes megértést! Ez csak úgy lehet teljes, ha a mellékelt/órákon szereplő gyakorlati feladatokat is végigcsináljuk!

A modulban leggyakrabban két táblázatkezelő-családra hivatkozunk. Ezek a következők: Microsoft Excel (továbbiakban csak Excel) különböző verziói és a LibreOffice Calc (röviden csak Calc).

12. LECKE

A táblázatkezelés általános áttekintése

A lecke a táblázatkezelés történeti és általános elméleti áttekintését, majd az egyszerű táblázatkezelés lehetőségeinek, problémáinak bemutatását tartalmazza. Ez a lecke – a bevezető témák miatt – kimondottan olvasmányos jellegű, azaz könnyen és gyorsan feldolgozható alfejezeteket is tartalmaz, emellett viszont már folyamatosan egyre több olyan részt tárgyalunk, amelyek intenzívebb elmélyülést (és gyakorlati kipróbálást) követelnek meg.

A lecke adatbázis-kezelés és a táblázatkezelés kapcsolatával foglalkozó részében feltételezzük, hogy az olvasó ismeri az előbbi szakterület alapfogalmait, ugyanis itt nem célunk ismertetni ezeket.

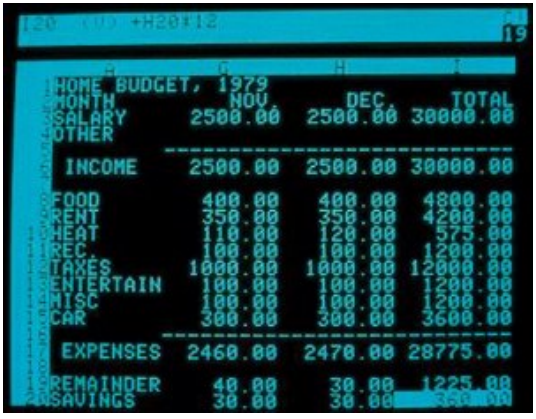
30. A táblázatkezelésről általában

30.1. A táblázatkezelés története

A számítógép előtti időszakban a gazdasági, pénzügyi és számítási feladatokat alapvetően papír–ceruza módszerrel oldották meg. Segítségként függvénytáblázatokat, illetve az akkori kor számolást támogató eszközeit használták (pl. logarléc, később számológép).

A modern számítógépes táblázatkezelés gyökerei az 1970-es évek végére nyúlnak vissza, ekkor jelentek meg a személyi számítógépek a piacon. Miután az emberek megvették ezeket az akkor még eléggé drága gépeket, meglepetéssel tapasztalták, hogy a számítógépekkel még a matematikai alapműveleteket sem lehet elvégezni. Ha erre is akartuk használni őket, akkor programot kellett írni hozzá.

Ebből következően a különböző pénzügyi és számítási feladatok megoldására is külön programot kellett készíteni. Egy harvardi egyetemi hallgató, Dan Bricklin a probléma felismerése és alapos elemzése után olyan általános programon kezdett el gondolkodni, amelynek segítségével minden ilyen jellegű feladat viszonylag egyszerűen modellezhető és megoldható. 1978-79 telén kidolgozta a táblázatkezelés alapelvét: egy nagy táblázatot kell létrehozni, és a táblázat sorait és oszlopait a sakk mezőinek azonosításához hasonlóan betű- és számkoordinátákkal kell ellátni, és ezekkel a táblázat bármelyik cellája kiválaszthatóvá válik. A cellák adatokat és adatok közti összefüggéseket tartalmazhatnak. Az akkor legnépszerűbb géptípusra, az Apple-re alapozva elkészítette az első táblázatkezelő programot, a VisiCalc-ot.



30.1. ábra. *A Visicalc képernyője*

A program igen nagy sikerrel mutatkozott be a piacon, hiszen segítségével minden felhasználó viszonylag egyszerűen oldhatta meg a maga speciális problémáját. Az egyik kortárs elemző szerint: „A VisiCalc hatására az üzletemberek számára hirtelen nyilvánvalóvá vált, hogy szükségük van egy személyi számítógépre”. Ez a program mai szemmel nézve is sok mindent tud abból, amit a modern táblázatkezelők, de természetesen jóval kezdetlegesebb.

Később a Microsoft cég is elkészítette saját PC-s táblázatkezelő programját (Multiplan). Az 1982-ben megjelent első változat szerény tudású volt; csak 256 sort és 64 oszlopot tartalmazott a táblázata . Ez azonban nem lett nagyon sikeres, jóval népszerűbb lett nála a Lotus cégnek a vele közel egy időben megjelent Lotus 1-2-3. táblázatkezelője. Ez a program ugyanis sokkal többet tudott, és egyszerűbben is lehetett használni. Nagyon sok eleme, funkciója és kezelési sajátossága táblázatkezelési szabványként terjedt el, és még napjainkban is szinte az összes táblázatkezelőben lényegében ugyanúgy használható. A program nevében szereplő számsor jelentéséről valamikor egy könyvben tréfásan azt írták, hogy azért van ott, mert a program kezelése olyan egyszerű, mint az 1-2-3.

Valójában az 1-2-3 a táblázatkezelő programok hármas funkciójára utal. Eszerint az ilyen programoknak tudniuk kell:

1. az adatokkal való műveletvégzést, számolást (alap táblázatkezelés);
2. a táblázat adatainak grafikus ábrázolását;
3. bizonyos adatbázis-kezelési funkciókat.

Ezek a funkciók a Lotus1-2-3 óta minden táblázatkezelőben megtalálhatók.

Nagyon sikeres volt ugyanebben az időben a Quattro táblázatkezelő is. Kis erőforrásigénye miatt is kedvelték, de Magyarországon fontos szempont volt az is, hogy az általában csak angolul beszélő időszakban nagyon kellemes volt egy magyar programmal dolgozni (a program legelső verzióját magyar fejlesztőgárda készítette el).

Táblázatkezelő programok nemcsak PC-s környezetben készültek, hanem lényegében minden akkori számítógépre. A CalcResult nevű programocskát például a Commodore 64 gépre fejlesztették ki.

A Microsoft a Multiplan után egy olyan új program kifejlesztésébe kezdett, amely ugyanazt tudta, mint az összes addigi népszerű táblázatkezelő együtt, csak még szebbre és még könnyebben használhatóra tervezte. A programot Excelnek nevezték el. Először Apple Macintosh gépekre készítették el, abból a környezetből hozták át a PC-kre, már a Windows 3.1 környezet alá. Ez volt az első olyan PC-s táblázatkezelő, amely nem DOS-os környezetben dolgozott. A Quattro még jó pár évig sikeresebb maradt, mint az Excel korai verziói, de a 90-es évek elejétől lassan az Excel lett a legnépszerűbb táblázatkezelő program.

Magyarországon elsőként a program 3.0-ás, illetve 4.0-ás verzióit használták, de igazán széles körben csak az 5.0-ás változat terjedt el az 1990-es évek második felében, ami több újdonságot is tartalmazott a korábbiakhoz képest. Ezt a verziót már az Office integrált programcsomagba építették be, amely tartalmazta még a Word, az Access és a PowerPoint programokat is.

Az újabb Office csomagokról (97, 2000, XP, 2003, 2007, 2010) – és ezen belül speciálisan az Excelről is – általánosan azt mondhatjuk el, hogy a változtatásokban két fő trend érvényesült. A programok felületének a kialakításával egyrészt az alapul szolgáló operációs rendszer környezetéhez igazodtak (pl. 2007 és 2010 –

30.2. ábra. *Az 5.0-ás Excel induló képernyője*

Vista, ill. Windows 7), másrészt a felhasználók kényelmének a kiszolgálása érdekében – általában sikeresen – változtattak kisebb-nagyobb mértékben (pl. diagramkészítés a 97-es verzióban; véletlenszámok kezelése a 2010-es verzióban). Ugyanakkor összefoglalva azt is kijelenthetjük, hogy a főbb Excel lehetőségek átlagos felhasználói nézőpontból nagyjából most is ugyanazok, mint korábban.

A közeljövőben számolhatunk a 2013-as Excel elterjedésével is, amely olyan új lehetőségeket valósít meg mint például az érintőképernyő használata.

Érdekes új színfolt a piacon a szabad szoftverek megjelenése és az utóbbi években már ténylegesen is mérhető térhódítása. Míg néhány éve elég könnyen el lehetett intézni az ilyen programokat annyival, hogy: „fapados”, „úgysem tud közel annyit sem, mint az Excel vagy a Word”; napjainkban már közel sem ilyen egyértelmű a helyzet. A szabad szoftverek tudásukban és a felhasználók kiszolgálásában meglepően gyorsan zárkóznak fel az Office programokhoz. Hangsúlyozva, hogy persze problémák azért adódnak velük (főként az átjárhatóság

miatt, lásd még később is), mégis, oktatásba való bevezetésük legalábbis megfontolandó, vagy akár: javasolt. Ezért hangsúlyozottan foglalkozunk a táblázatkezelő Calc programmal is.

30.2. A táblázatkezelő programok szolgáltatásai

Az elsősorban irodai munka során használt táblázatkezelő programok adatok gyors és sokoldalú kezelését teszik lehetővé.

Ezen programok segítségével összefüggéseket tudunk meghatározni a sorokba és oszlopokba rendezett adatok között; az alapadatok felhasználásával számított értékeket állíthatunk elő. A számítások elvégzéséhez képleteket és függvényeket használhatunk. Lehetőség van kiválogatás, kigyűjtés típusú feladatok elvégzésére is. Az alapadatok változása esetén a program – a beállítás függvényében – magától frissíti az eredményeket, amik grafikusan szemléltethetők.

A fontosabb elvárások egy ilyen programmal szemben a következők:

1. könnyű programkezelés;
2. kényelmes és gyors adatbevitel;
3. kapcsolatok létrehozásának lehetősége az adatok között;
4. új adatok származtatása feltételek, valamint matematikai és egyéb eszközök felhasználásával;
5. adatok rendezése;
6. bizonyos tulajdonságú adatok kiválogatása;
7. diagramkészítés;
8. képek, ábrák megjelenítése;
9. nyomtatás;
10. barátságos súgórendszer.

30.3. Adatbázis-kezelők és táblázatkezelők

Sokan összekeverik az adatbázis-kezelő és a táblázatkezelő rendszereket, pedig fontos megkülönböztetni ezt a két programtípust.

Az általuk kezelt adatok első közelítésben hasonlóak ugyan, de a megoldható problémák mélységükben már különbözőek. A megoldandó feladat jellegéből határozható meg, hogy melyik rendszer használata célszerű. Az adatbázis-kezelők összetettebb és mélyebb struktúrájú problémák megoldására használhatók fel.

A legjobban elterjedt relációs adatbázis-kezelőkben az adatmodellezés után elkészített rendszer az adattáblákat hierarchikusan kezeli. A táblák között kapcsolatok létesíthetők, némelyik rendszer ezeket a kapcsolatokat az adattáblákhoz rendelve tárolni tudja. Itt jóval összetettebb módon lehet leképezni az adott problémát, mint a táblázatkezelőknél.

A tervezés során három alapelemet definiálunk. Ezek a következők: egyed és egyedtípus, tulajdonság és kapcsolat. Az alapelemek terve adja a modell logikai szintjét. A terv megvalósítása a fizikai szint, itt keletkeznek az egyedtípusokat reprezentáló táblázatok, az egyed-előfordulásokhoz tartozó tulajdonságok értékei (állandó adatok), és valósulnak meg a tervezett kapcsolatok.

A táblázatkezelők is rendelkeznek adatbázis-kezelés jellegű funkciókkal (kereszttáblázat készítése, rendezés, lekérdezés vagy szűrés), de némelyik adatbázis jellegű műveletnél csak egy táblát tudunk használni (pl. szűrés), bár van olyan funkció is, ahol több táblából is vehetjük az adatokat (kimutatás).

Megjegyezzük, hogy a táblázatkezelőben sokkal kötetlenebb adatkezelés valósítható meg, mint az adatbázis-kezelésben. Például a cellákba írt adatok nem alkotnak szükségképpen valódi adatbázis-táblázatot.

Adatbázis-kezelő esetén az összekapcsolt táblákból minden nehézség nélkül lehetséges a lekérdezés, jelentés készítése. Sokkal rugalmasabb lehetőségei vannak, mint a táblázatkezelőnek (könnyen definiálható lenyíló listák, feltételek...). Adatbevitel maszkoltan, feltételekkel korlátozva – azaz ellenőrizve történhet. Igaz makrót – apró programot – írhatunk táblázatkezelőhöz is az adatbevitel ellenőrizésére, de ez jóval nehézkesebb itt, mint az adatbázis-kezelők esetén. Hasonlóan, pl. az Excel is képes lenyíló listák és választó gombcsoport kezelésére, de ugyanez sokkal rugalmasabban valósítható meg az adatbázis-kezelő rendszerekben.

Adatbázis-kezelőt célszerűbb használni nagyobb mennyiségű egyed-előfordulás (sor) esetén, illetve ha az egyed sok tulajdonsággal rendelkezik (oszlop).

Kicsit leegyszerűsítve mondhatjuk, hogy az adatbázis-kezelő az adatok egy megfelelően rendszerezett halmazát és a köztük levő komplex kapcsolatok előállítását, kezelését teszi lehetővé, míg a táblázatkezelő alapfunkcióját tekintve elsősorban az alap- és származtatott adatok valamilyen formában történő megjelenítésével foglalkozik.

30.4. Problémamegoldás táblázatkezelő programok segítségével

A táblázatkezelő programokkal megoldható problémák nagyon változatosak. A megoldás előállítására nincsen olyan általánosan megadható részletes recept, amelynek segítségével biztosan célhoz érünk, de a főbb lépések mindig a következők:

Figyelmesen olvassuk el, elemezzük és értsük meg a feladatot!
Hogyan oldanánk meg mi a feladatot? (Építsük fel a megoldási módszert, mintha számítógép nélkül, papír–ceruzával dolgoznánk.)
Milyen segítséget tud adni az általunk elképzelt módszerhez a táblázatkezelő program? (Gyűjtsük össze a szükséges apparátust: függvények, képletek stb.)
Állítsunk elő egy (rész)megoldást egy cellában vagy egy példányban! (Előfordulhat, hogy a teljes megoldás több részből áll.)
Ha szükséges: másoljuk le a (rész)megoldást előállító képletet ahova kell!

A lépésekről részletesebben:

1. lépés

A megoldás elkészítése előtt pontosan értelmeznünk kell a feladatot. Jó esetben a feladat írásos formában adott (például a vizsgán, vagy a munkahelyi vezetőtől kapjuk meg a szöveget), máskor esetleg a pontos megfogalmazást is nekünk kell elkészíteni valamilyen szóbeli információ alapján. (Ez a nehezebb.) Fontos, hogy lássuk, milyen adatokból kell kiindulnunk, és azokból milyen eredményt kell meghatároznunk.

2. lépés

Meg kell terveznünk a számítás menetét, és azt, hogy melyik adatot hogyan tudjuk elhelyezni a papíron készített táblázatban ahhoz, hogy áttekinthető megoldást kapjunk.

A feladat megértése és a papíron történő megoldás sok esetben nagyon egyszerű (akár triviális), máskor jóval nehezebb lehet. Célszerű az első két lépéshez szükség esetén jegyzeteket is készíteni. Fontos, hogy megfelelő időt szánjunk ezekre a lépésekre, mert az itt „megspórolt” idő később hiányosságokhoz, esetleg teljesen rossz számítógépes megvalósításhoz vezethet!

3. lépés

Meg kell vizsgálnunk, hogy melyik adatot hogyan érjük el, az milyen formában adott, és hogyan lehet az adatokból a szükséges kifejezéseket felépíteni. A táblázatkezelő program sokféle számítás elvégzéséhez tartalmaz beépített függvényeket. El kell döntenünk, hogy mely részfeladatokhoz használhatók fel ezek. Ha nem függvény vagy képlet állítja elő a megoldást, akkor elemezzük az adott eszköz használatát (pl. szűrő). (Tájékozódjunk a súgóban, szakkönyvekben vagy az interneten is!)

4. lépés

Az adott feladat vagy részfeladat megoldását előállítjuk a szükséges képlet alkalmazásával egy cellában, ill. a megfelelő eszköz alkalmazásával megoldjuk a feladatot. Ha erre lehetőség van, érdemes a bemeneti adatok változtatásával ellenőrizni, hogy a képlet valóban jó eredményt ad-e.

Megjegyezzük, hogy az utóbbi két lépés lehet esetleg iteratív. Tudásunk fejlődhet, a táblázatkezelő program eszköztára bővülhet – mindezek következtében előfordulhat, hogy esetleg hónapokkal később egy adott problémára fejlettebb megoldás állítható elő. Ugyanezen okok miatt a fontos és hosszú ideig aktuális (pl. ciklikusan ismétlődő céges) feladatok esetében célszerű időnként később is visszatérni a problémára, és önvizsgálatot tartani: Tényleg ez a lehetséges legjobb megoldás? Tényleg a legjobb eszközt használtuk?

5. lépés

Amennyiben több cella értéke is hasonló módon számolható, akkor a képletet másolással kell a többi cellába bevinni. Ehhez először megállapítjuk, hogy a másolás után a várt eredményt kapjuk-e. Ha nem, akkor alkalmasan át kell alakítani a képletet, és utána elvégezni a másolást.

Ezekkel a lépésekkel mindig célhoz érünk, azonban természetesen ismerni kell az alkalmazandó eszközöket (a táblázatkezelő program apparátusát), és rendelkeznünk kell bizonyos általános problémamegoldó képességgel. Ez utóbbi tanulással, gyakorlással elsajátítható.

30.5. Melyik táblázatkezelő programot válasszuk?

Noha rengeteg könyv, jegyzet, oktatási segédanyag foglalkozik táblázatkezelő programok bemutatásával, mégis meglepően ritkán találkozhatunk olyan (igényes) elemzéssel, amely táblázatkezelő programok összehasonlító értékelését végezné el. Ebben az alfejezetben bemutatunk néhány olyan fontos szempontot, amelyek segítségével válasz adható a következő – egyáltalán nem könnyű – kérdésekre: „A táblázatkezeléshez melyik az igazán jó szoftvereszköz?” „Mi melyik táblázatkezelő programot válasszuk?”

Nem állítjuk, hogy a most következő felsorolás lépése adják az „egyedül üdvözítő utat”, de mindenképpen egy jó pár éves gyakorlati használat, oktatási tapasztalat, szakmai konzultáció stb. eredményeként megfogalmazott megalapozott szakmai vélemény. Tehát az a jó táblázatkezelő, . . .

1. . . . amelyik számunkra elérhető, illetve legálisan elérhető (!);
2. . . . amelyik tudja azt, amit egy jó táblázatkezelőnek tudni kell (lásd fent) illetve amelyik tudja azt, ami számunkra a munkánk során szükséges;
3. . . . amelynek a használata számunkra kényelmes, barátságos;
4. . . . amelyet a környezetünkben mások is széles körben használnak (átjárhatóság).

Mindenki pontosan tudja – bár önmaguktól csak nagyon kevesen gondolnak rá… –, hogy az első pont a legfontosabb. Számolnunk kell azzal a tendenciával is, hogy állami intézményeknél, cégeknél, vállalatoknál is egyre hangsúlyosabb szempont lehet az ingyenes, illetve az eddiginél olcsóbb megoldások keresése.

A második pont lehet egyénfüggő. Például egy szerényebb igényű felhasználó akár jóval kisebb tudású programmal is beérné – már amennyiben valóban szabadon választhatna…

A harmadik pont szintén szubjektív, ha valaki Linux rendszert használ, akkor valószínűleg a Microsoft termékek helyett eleve a szabad szoftverek felé fordul. A negyedik pont azért nagyon fontos, mert munkánk során általában együtt dolgozunk másokkal. Ha azt tapasztaljuk, hogy környezetünkben egyre többen áttérnek a 2003-as Office-ről a 2010-esre, akkor célszerű nekünk is követni a példát.

Mint azt a fentiekben is láttuk, a választás sokszor nemcsak az egyéntől függ, Az is lehet, hogy a főnök vagy a munkahely kötelezően elírja, hogy melyik táblázatkezelő programot kell használni. Ezért törekszünk az általános tárgyalásra, sokszor egyszerűen azt fogjuk mondani, hogy a „táblázatkezelő tudja ezt és ezt”. Ugyanezen okból több programot is használunk, *igazi „fő” eszköz megjelölése nélkül!* Az általunk támogatott táblázatkezelők következők:

1. MS Excel 2003 (pár éve még szinte egyeduralkodó volt, napjainkban is sokan használják);
2. MS Excel 2010 (sokan áttértek rá a 2003-as felhasználói közül, sokan eleve ezt tanulták);
3. LibreOffice Calc (szabad szoftver, terjedőben; mi a 3.5-ös és 3.6-os verziókat használjuk, a 2012-es állapot szerint).

Fontos hangsúlyozni, hogy az összes feladatot nem tudjuk bemutatni mindhárom környezetben, ez meg is haladná lehetőségeinket. Ugyanakkor ez nem is feltétlenül szükséges, mert – mint látni fogjuk – a hasonlóság magas fokú. Az Olvasó számára érdekes házi feladat lehet, hogy egy adott programban elkészített megoldást „átvisz” (esetleg átalakítással) egy másik táblázatkezelőbe. A konvertálásokkal (esetleges veszteségekkel) még később foglalkozunk.

31. Egyszerű táblázatkezelés

Bármelyik táblázatkezelőt is választjuk saját eszközünknek a fentiek közül, általánosan igaz, hogy a program lehetőségeinek teljes bemutatására a jegyzet keretein belül nincs mód. Az itt megszerezhető ismeretekkel azonban már sok érdekes feladat és gyakorlati probléma megoldható. A táblázatkezelés „filozófiájának” megismerése pedig szilárd alapot nyújt az érdeklődőknek a további – akár önálló – tanuláshoz is.

Az alapvető használattal foglalkozó részben eltekintünk egyes általánosabb (nem táblázatkezelő-specifikus) lehetőségek tárgyalásától – például a fájlkeresés részletes ismertetése –, és néhány esetben ésszerű rövidítéseket alkalmazunk. Mindezt természetesen úgy tesszük, hogy a fejezet önállóan is tanulható és feldolgozható legyen.

A táblázatkezelő programok más programrendszerekhez hasonlóan indíthatók és állíthatók le.

31.1. Képernyőelemek

A program elindítása után a képernyőn megjelenik a táblázatkezelő ablak, amely a következő részekből áll (példánkban most: 2003-as és 2010-es Excel):

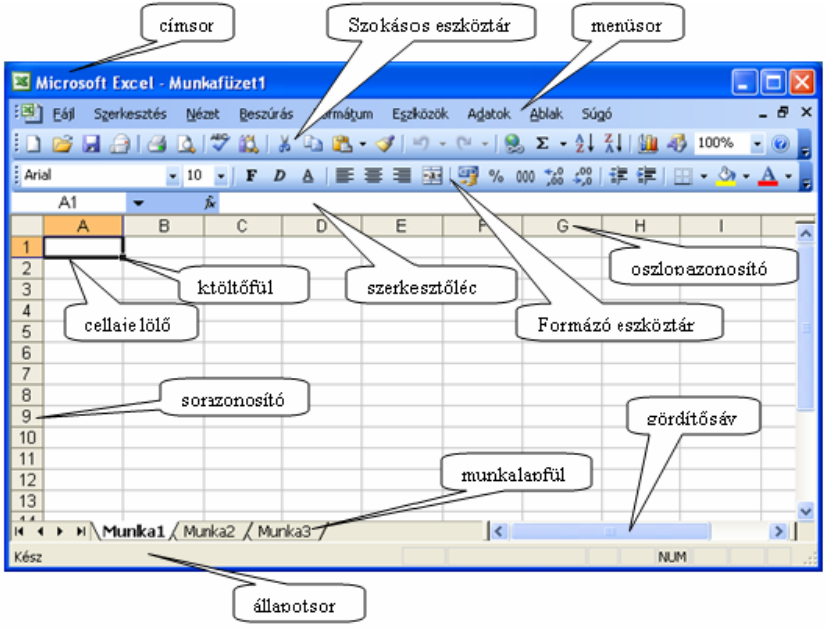
A képernyő felső sorában (ez a címsor) a Microsoft Excel programnév mellett az éppen szerkesztett munkafüzet (vagy másképpen az Excel dokumentum) neve látható.

Alapértelmezett beállítást feltételezve a címsor alatt egy menüsor helyezkedik el, amely 9 főmenüpontot tartalmaz. Közülük néhány nagyon hasonló más Windows rendszerek menüpontjához (például a **Fájl**), de vannak egyediek is (például az **Adatok**).

A címsor és a menüsor szélein található kapcsológombok az ablakok vezérmenüjének aktivizálására illetve az ablakok átméretezésére szolgálnak.

A 2003-as Excelben a menüsor alatt eszköztárak, gyorsítósávok helyezkednek el. Ezek egyes gyakran használatos, menüből is hívható funkciók egérrel történő gyors elérését teszik lehetővé.

A 2010-es Excelben a menük alatti úgynevezett menüszalagon az elemek csoportokba rendezve – kicsit másképpen mit a 2003-asban –, helyezkednek el. A menüszalag kialakításánál fontos szempont volt, hogy

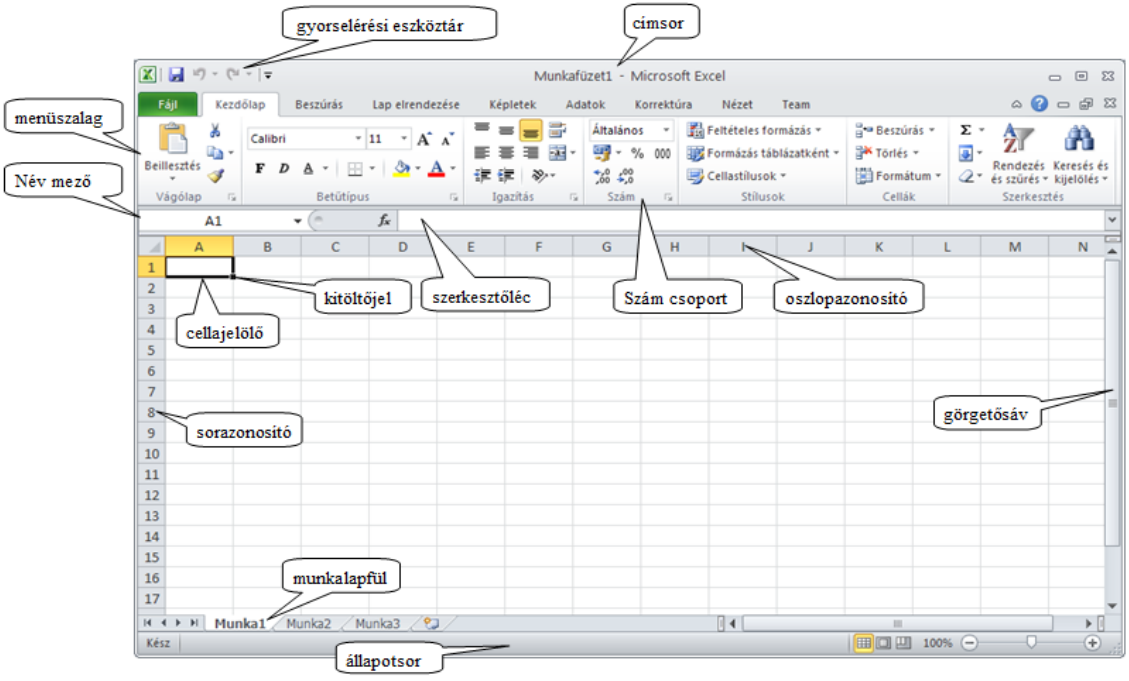


31.1. ábra. *A 2003-as Excel ablakainak a felépítése*

ebbe beleépítsék az eszköztár funkciót is.

Speciális új elem a 2010-es Excelben a gyorselérési eszköztár, amely egyes kiemelt funkciók azonnali hívását teszi lehetővé.

A menük és a párbeszédablakok, illetve a menüsávok/gyorsítósávok a Windows környezetben megszokott módon használhatók, a használat részleteit nem tárgyaljuk.



31.2. ábra. *A 2010-es Excel ablakainak a felépítése*

A gyorsítósávok alatt található a szerkesztőléc és a munkalap. A munkalap oszlopokba és sorokba rendezve cellákat tartalmaz.

Megkülönböztetjük a sablont (másképpen munkalap vagy üres tábla) az általunk elkészített táblázatoktól. Az üres táblákba építjük fel a táblázatokat. Hasonlóan megkülönböztetjük a cellaterületet (kis téglalap a képernyőn) magától a cellától. A cellákba írjuk be az adatokat, és ezt a táblázatkezelő valamilyen módon

Program	Sorok száma	Oszlopok száma
Excel 2003	65536	256
Excel 2010	1048576	16384
Calc	1048576	1024

31.3. ábra. *A táblázatkezelők munkalapjának méretei*

megjeleníti a cellaterületen.

A cellákat (és a cellaterületet is) oszlop- és a sorkoordinátával azonosítjuk. *Az alapértelmezett azonosítás A1 stílusú*, a sorokat számokkal, az oszlopokat az angol ABC betűivel jelöli a program. A „Z” oszlopot követően az oszlopok két betűből álló nevet kapnak (például: AA, AB stb.).

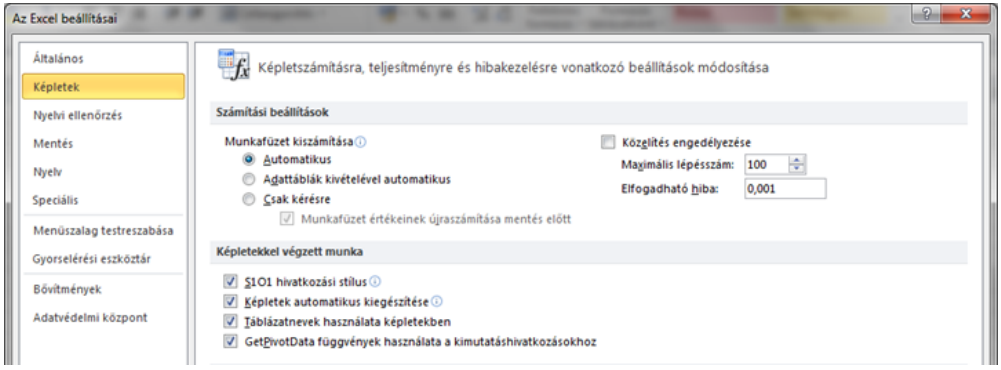
A jóval ritkábban használt S1O1 (az angol nyelvű változatban R1C1) stílusnál mind a sorokat, mind az oszlopokat számmal jelöli a program, ez a hivatkozási stílus makrók rögzítésénél vagy VisualBasic programok készítésénél lehet hasznos. Az S1O1 hivatkozási stílust be- és kikapcsolni Excel 2003-ban és Calc-ban az **Eszközök/Beállítások...**, Excel 2010-ben a **Fájl/Beállítások** menüpont alatt lehet.

Az S1O1 (R1C1) bekapcsolása után a névmezőben (másképpen névdobozban) a következőképpen tünteti fel a program a hivatkozást: "S" + a sor száma + "O" + az oszlop száma ("R" + a sor száma + "C" + az oszlop száma).

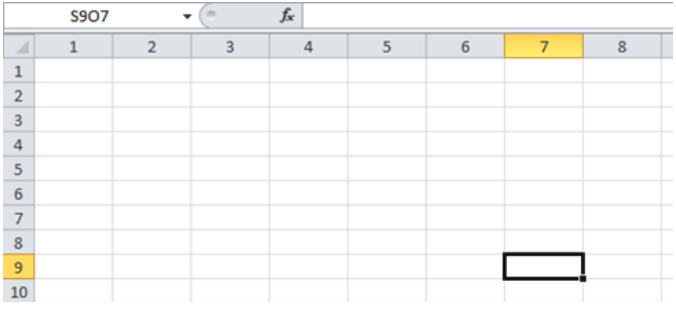
Az aktuális cella helyét egy téglalap alakú mutató jelzi. Ezen cella címét a szerkesztőléc bal szélén, tartalmát a jobb oldali fehér mezőben láthatjuk. Kijelölt és elnevezett blokk esetében a bal szélső névmezőben a blokk neve található.

A munkalap logikailag egy munkafüzet (Excel dokumentum) része.

A munkalap jobb szélén és alján gördítősávok vannak, amik az egér segítségével gyors mozgást tesznek lehetővé. A sávon látható csúszka pozíciója a képernyőn látható munkalaprészlet helyzetét jelzi a munkaterületen vagy a munkaterület bejárt részén belül.



31.4. ábra. *Az S1O1 hivatkozási stílus beállítása a 2010-es Excelben*



31.5. ábra. *Az S1O1 hivatkozási stílus alkalmazása (bekapcsolva)*

Az alsó gördítősáv bal szélén a munkalapokat azonosító kis fülek vannak. Ha sok a munkalap, akkor a füleknek csak egy része látszik. A láthatóság a mellettük lévő nyilakkal értelemszerűen szabályozható.

A képernyő legalsó részén az állapotsor a szerkesztést megkönnyítő adatokat tartalmaz. Mutatja az egyes kapcsolók állapotát, valamint a kiadott paranccsal kapcsolatos megjegyzéseket.

Az Excel ablakban nemcsak egy, hanem több dokumentum is elhelyezhető, illetve az aktuális munkafüzet lekicsinyíthető. Ilyenkor az ismertetett képernyő-elrendezés annyiban módosul, hogy a szerkesztőléc és az állapotsor közötti rész külön dokumentumablakba kerül, amelynek címsora az aktuális munkafüzet neve, az Excel ablak címsorából pedig ez a név eltűnik.

A LibreOffice Calc ablakának felépítését részletezően külön nem tárgyaljuk, mert nagyon hasonlít a 2003-as és kisebb mértékben 2010-es Excelhez. A dokumentumokra, munkalapokra fentiekben tett általános kijelentések a Calc-ban ugyanígy érvényesek.

31.2. A munkakörnyezet beállítása

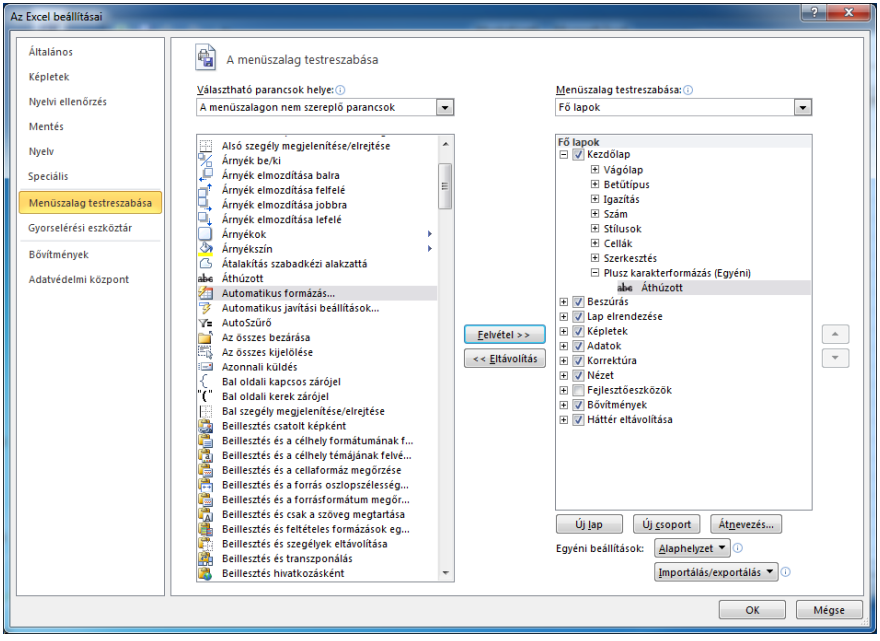
A táblázatkezelő programok lehetőséget adnak a munkakörnyezet bizonyos mértékű testreszabására. Tipikusan ilyen jellegű feladatok lehetnek a következők:

1. Munkánkhoz egyes funkciókat gyakran használnánk, és ezek nincsenek a meglévő eszköztáron vagy menüszalagon illetve a menüben;
2. Új eszköztárat, menüt, menülapot akarunk létrehozni;
3. Billentyűparancsot akarunk hozzárendelni egyes gyakran használt funkciókhoz.

A – meglehetősen bőséges – lehetőségek többségét Excel 2003-ban és Calc-ban az **Eszközök/Testreszabás...** menüpont alatt, Excel 2010-ben pedig a **Fájl/Beállítások/Menüszalag testreszabása** ill. **Fájl/Beállítások/Gyorselérési eszköztár** pont alatt találjuk meg. A részleteket a használt program menüpontjainak a súgó segítségével való tanulmányozásával ismerhetjük meg.

Ugyancsak bőséges választék áll rendelkezésünkre a táblázatkezelő programok környezeti beállításaihoz. Az általános lehetőségeken felül módunk van többek között számolási, megjelenítéssel, szerkesztéssel és hibaellenőrzéssel kapcsolatos opciók finomhangolására.

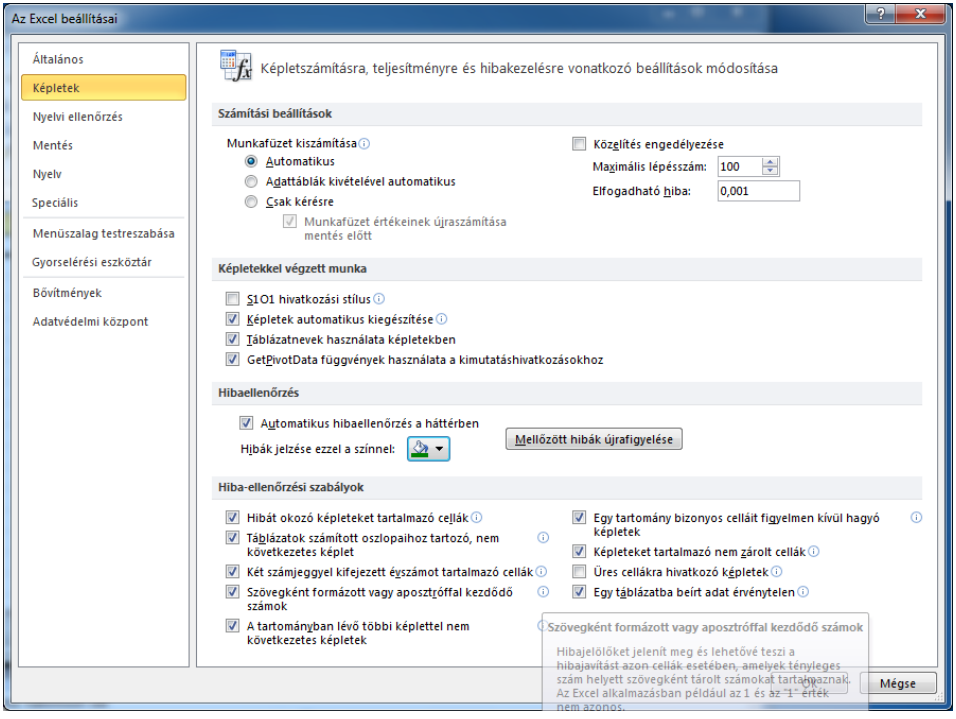
A beállítási lehetőségek Excel 2003-ban és Calc-ban az **Eszközök/Beállítások...**, Excel 2010-ben a **Fájl/Beállítások** menüpont alatt érhetők el.



31.6. ábra. *Menüszalag testreszabása (új lap felvétele, új paranccsal) Excel 2010-ben*

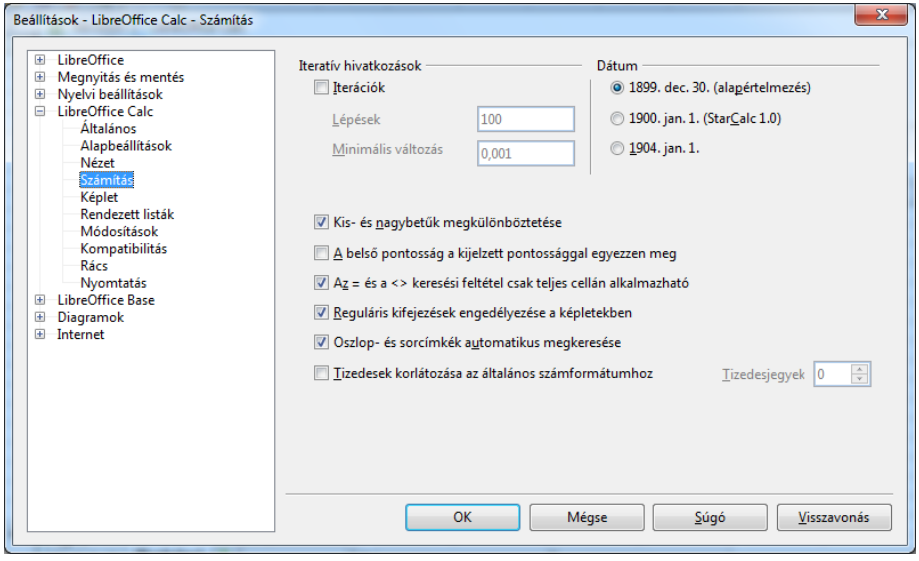
31.3. Fájlműveletek

Új munkafüzet létrehozása a táblázatkezelőkben a **Fájl** menü (2010-es Excelben Backstage) **Új dokumentum...** vagy **Új** parancsával történik, A megjelenő munkaablakban/fülön válasszuk ki az **Üres munkafüzet** vagy **Munkafüzet** elemet. A 2003-as Excelben és a Calcban azt is megtehetjük, hogy egyszerűen a **Szokásos** eszköztár **Új dokumentum** gombjára kattintunk. A táblázatkezelő elindításával automatikusan új dokumentum készítése kezdődik.



31.7. ábra. *Képletek, teljesítmény és hibakezelési beállítások (Excel 2010)*

Az eszköztár gombjaira most és a továbbiakban a táblázatkezelők alapértelmezése szerint hivatkozunk. Mivel a testreszabás funkcióval a gombok helye megváltoztatható, sőt az eszköztárról el is távolíthatók, ezért nem garantálható, hogy a használt táblázatkezelő képernyőjén is az általunk leírtaknak megfelelően helyezkednek el.

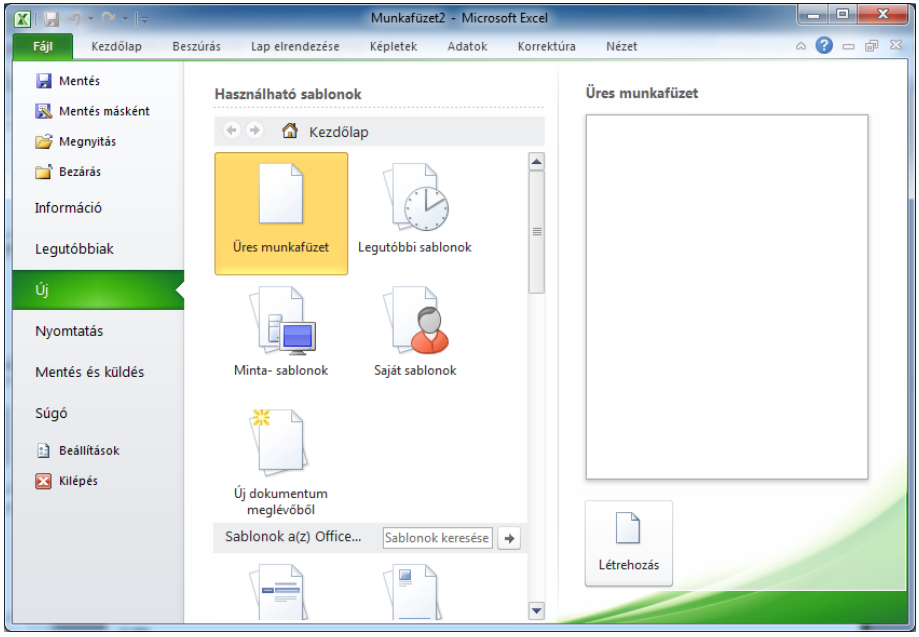


31.8. ábra. *Számítási beállítások Calc-ban*

Ha egy korábban elkezdett munkát szeretnénk folytatni, akkor be kell töltenünk a kívánt munkafüzetet. Erre a **Fájl** menü (Backstage) **Megnyitás** parancsa vagy a 2003-as Excelben és a Calcban a **Szokásos** eszköztár megfelelő gombja szolgál. A megjelenő párbeszédablakban meg kell adni, vagy listából ki kell választani a fájl elérési útvonalát és nevét, hasonlóan ahhoz, mint más Windows alkalmazásokban.

Ha olyan fájlt szeretnénk megnyitni, amelyik nem azzal a táblázatkezelővel készült, amelyikben éppen dolgozunk, akkor a megfelelő konverzió elvégzéséhez általában át kell állítani a **Fájltípus** mezőt.

Az aktuális munkalapon végzett változtatásokat a táblázatkezelő a memóriában tárolja. Mivel az írható memória tartalma kikapcsoláskor általában elveszik, ezért a későbbi felhasználáshoz a változásokat el kell mentenünk. A táblázatkezelők a mentésre általában többféle lehetőséget kínálnak, valamennyi a **Fájl** menüben



31.9. ábra. *Új dokumentum létrehozása (Fájl Backstage, Excel 2010)*

(Backstage-ben) található meg. A **Mentés** funkciót választva a gép a mentést azonnal végrehajtja (ez a funkció a 2003-as Excel és a Calc esetében a gyorsítósávon is elérhető). Ha a munkafüzetet más névvel vagy először szeretnénk elmenteni, akkor a **Mentés másként** funkciót kell választani, illetve ha nem ezt választottuk, akkor is az ehhez tartozó ablak jelenik meg. A megjelenő párbeszédablakban a fájl nevét és elérési útvonalát kell megadni, de a fájltípus is megváltoztatható. Az alapértelmezés szerint a mentés mindig abban a fájltípusban történik, amit a használatban lévő táblázatkezelő használ.

Ezért a megnyitáshoz és a mentéshez fontos tudnunk, hogy melyik az adott táblázatkezelő alapértelmezett formátuma, ill. hogy a táblázatkezelőnk milyen egyéb fájltípusokat képes előállítani. A 31.23. táblázatban összefoglaljuk a jegyzetben tárgyalt programok által ismert fájltípusokat.

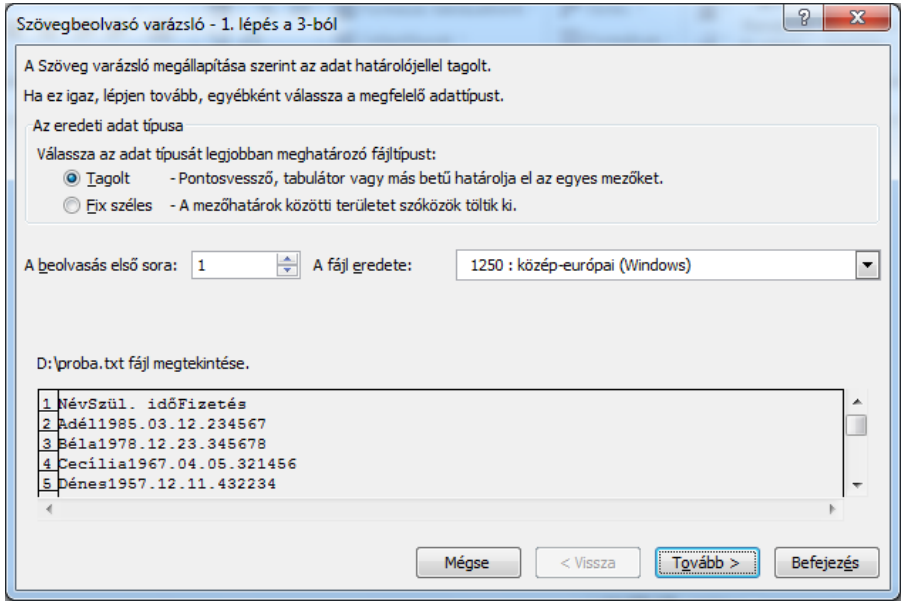
31.23. táblázat. *Átjárhatóság a táblázatkezelők alapértelmezett formátumai között*

Program	Alapértelmezett formátum	Egyéb támogatott formátum (csak az előzőek közül)
Excel 2003	**xls**	**mentésnél nincs más lehetőség; betöltésnél (csak egy kiegészítő ún. kompatibilitási programcsomaggal): xlsx, ods**
Excel 2010	**xlsx**	**Mentésnél, betöltésnél: xls, ods**
Calc	**ods**	**Mentésnél, betöltésnél: xls, xlsx**

Ha nem az alapértelmezett fájlformátumot használjuk mentésnél és megnyitásnál egyaránt csekély veszteség előfordulhat.

A táblázatkezelők ezen felül sok egyéb fájltípust is tudnak kezelni. Néhány fontosabb ezek közül: XML formátum, tagolt szöveg (leggyakrabban tabulátorral, szóközzel vagy. pontosvesszővel), korábbi Excel verziók munkafüzetei. Ezek közül különösen fontosak a szövegfájlok, hiszen sok esetben ezekben kapjuk meg a feldolgozandó adatokat. Az ilyen fájlok betöltésénél általában a táblázatkezelőbe beépített varázsló vezet bennünket végig a szükséges lépéseken. Ehhez hasonló a csv formátum is.

Ha befejeztük egy munkafüzet használatát, akkor célszerű menteni és bezárni a fájlt.



31.10. ábra. *Szöveges fájl beolvasása (varázslóval) 2010-es Excelben*

31.4. Mozgás a táblázatban

Az aktuális cellát jelző mutatót a táblázat tetszőleges helyére elmozgathatjuk. Erre a billentyűzettel és az egérrel egyaránt sok lehetőség van. A fontosabbakat a 31.24. táblázatban összefoglaltuk.

Nem megszokott mozgásra használhatók a CTRL+←, CTRL+→, ... és az END ←, END →, ...

Ilyenkor a cellakijelölő mindig teli cellára szeretne ugrani a megadott irányban. Ha éppen teli cellán áll, akkor az utolsó teli cellára ugrik, ha több ilyet nem talál, akkor a táblázat végére pozícionál. Ha üres cellán áll, akkor a legközelebbi teli cellára ugrik, ha ilyet nem talál, akkor pedig a munkalap végén áll meg.

31.24. táblázat. *Mozgás a táblázatban*

Cél	**Billentyűkombináció**
Egy cellával balra, jobbra, fel, le	←, →, ↑, ↓
Ugrás a sor elejére	HOME
Ugrás egy képernyőt fel/le	PAGE UP / PAGE DOWN
Ugrás egy képernyőt jobbra/balra	ALT+PAGE DOWN / ALT+PAGE UP
Ugrás a bejárt tartomány elejére/végére	CTRL+HOME / CTRL+END
Ugrás a következő/előző munkaterületre	CTRL+PAGE DOWN / CTRL+PAGE UP

Egérrel a képernyő tetszőleges cellájára kattintva az lesz az aktuális, továbbá a gyorsabb mozgáshoz a gördítősávok használhatók.

Az Excel változatok a fentiek mellett még lehetőséget biztosítanak a direkt ugrással történő pozícionálásra is (CTRL+G; Excel 2003: **Szerkesztés/Ugrás...**; Excel 2010: **Kezdőlap/Szerkesztés/Keresés és kijelölés**).

Még a bizonyos gyakorlattal rendelkező, átlagos – nem kezdő – felhasználók sincsenek sokszor tisztában azzal, hogy nagyobb méretű táblázatok bejárásakor mennyire fontos készség szinten ismerni a gyors mozgás lehetőségeit. Egy több ezer adatot tartalmazó táblázat esetén egy nem teljesen egyszerű, de nem is túl bonyolult feladat megoldása során akár tíz-húsz perces különbségek is jelentkezhetnek a hatékony technikai megoldások alkalmazásával a hétköznapi, megszokott, ezért könnyen megjegyezhető módszerekkel szemben.

31.5. Adatok

31.5.1. Beírás a cellákba, javítás, törlés

Az aktuális cellába tetszőleges adatot beírhatunk. A beírás végét az ENTER leütése vagy – amennyiben nem a már meglévő cellatartalmat változtatjuk éppen – valamely kurzormozgató billentyű, illetve a TAB lenyomása jelzi. Ezután a cella tartalma a begépelt adat lesz.

Gyakran a beírt adat olyan hosszú, hogy nem fér el egy adott cellához tartozó területen. Ilyenkor a gép szöveges adatok esetén a szomszédos cellák helyét is – amennyiben azok üresek – felhasználhatja a megjelenítésre. Más típusú adatnál a hosszú adat bekerül ugyan a cellába, de nem lesz látható teljes hosszában. Előfordulhat az is, hogy a megjelenítendő adat helyett # jelek sorozatát látjuk a cellában. Általában akkor, ha az adat numerikus és nem fér el, vagy hibaüzenet jelenne meg, de ez sem fér ki. A szerkesztőléc jobb oldali mezőjében azonban ekkor is megjelenik a cella teljes tartalma.

Előfordulhat, hogy egy cella tartalmát utólag módosítani szeretnénk. Ilyenkor álljunk rá a cellára, és nyomjuk le az F2 billentyűt, vagy egérrel kattintsunk a szerkesztőléc jobb oldali mezőjére. Ezzel szerkesztő módba kerülünk, és a cella tartalma részben vagy teljesen átírható. Hosszabb cellatartalom esetén a mozgáshoz használhatjuk a nyilakon kívül a HOME és az END billentyűket, illetve egérrel kattintsunk a megfelelő helyre. Az Ins billentyűvel váltogathatunk a beszúró és a felülíró üzemmód között. Az első az alapértelmezett, ami azt jelenti, hogy a szerkesztés megkezdésekor ez az állapot lesz aktív. A javítást az ENTER lenyomásával véglegesíthetjük. Ha esetleg javítás közben meggondoljuk magunkat, akkor az Esc billentyűvel kiléphetünk a szerkesztésből. Ilyenkor megmarad az eredeti cellatartalom.

Az aktuális cella tartalma a Del billentyűvel törölhető.

31.5.2. Adattípusok

Az adatok műveleti tulajdonságaik alapján különböző típusokba sorolhatók. A táblázatkezelők adattípusai a következők: szöveg, szám, dátum, idő és logikai (ez utóbbi be lehet sorolva a számtípusok közé (például a Calc program esetében).

A cellaterületen ezekbe a kategóriákba látszólag nem sorolható üzeneteket, hibaüzeneteket (például #HIÁNYZIK, #ÉRTÉK!, #HIV!, Hiba:511) is láthatunk. Itt azonban vegyük figyelembe, hogy a cella eredeti tartalma valamilyen hibás kifejezés, és erre a hibára utal az üzenet. Egyes kategorizálások szerint a táblázatkezelőkben léteznek hiba és tömb adattípusok is.

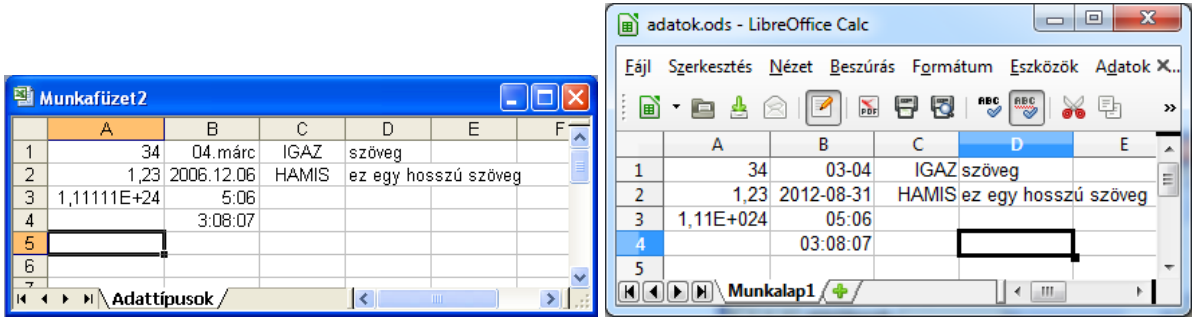
Az adatok típusát a begépelés formátuma alapján a táblázatkezelő felismeri, és azokat ennek megfelelően jeleníti meg. Alapértelmezésben a szöveget balra, a számot, valamint az idő és dátum adatokat jobbra igazítja. A logikai adatok és az egyéb üzenetek csupa nagybetűvel jelennek meg, vagy középen (Excel) vagy jobbra igazítva (Calc). Természetesen az alapértelmezett beállítások felülbírálhatók! Ilyen esetekben gondosan tájékozódjunk, hogy mi a cella tartalmának illetve az ott lévő képlet értékének adattípusa!

A számadatok alapesetben csak számjegyeket, tizedesvesszőt, előjelet és – tudományos megjelenítés esetén – exponens jelet (E) tartalmazhatnak.

A dátum típusú adatok megadására többféle mód is van. A legcélszerűbb az, ha év, hó, nap sorrendben, 2 számjeggyel – az év esetében 2 és 4 jegy egyaránt használható – egymástól ponttal elválasztva adjuk meg az adatokat. A nap után nem szabad pontot tenni. A Calc programban alapértelmezésként használható az éééé-hh-nn formátumú megadás is.

Az idő típusú adatok hasonlóan többféle módon, de a legegyszerűbben óra:perc formátumban adhatók meg, ahol az óra 0 és 23 között, a perc 0 és 59 között érték lehet. A perc után egy kettősponttal elválasztva másodperc is definiálható. A táblázatkezelők ezeket az adatokat számként tárolják és kezelik. A dátumok logikailag sorszámok; a legkorábbi dátum Excelben 1900.01.01 (a „nulladik nap”, fiktív dátumként 1900.01.00), az utolsó – a mostani Excel és Calc verziókban – 9999.12.31, aminek sorszámértéke Excelben 2958465. Érdekesség, hogy a Calc programnál egy kis listából kiválasztható a „nulladik nap” (az alapértelmezés 1899. dec. 30.). Az időt a táblázatkezelőkben 0 és 1 közötti törtszámok jelentik, ahol a 0 még értelmezhető időpont, az 1 nem. Egy cellában lévő törtrészt is tartalmazó valós szám dátumként és időként, sőt mindkettőként egyszerre is értelmezhető. Azt, hogy a táblázatkezelő minek mutatja formátumbeállítással határozhatjuk meg.

A logikai adatok az IGAZ és a HAMIS értékek. Beíráskor kisbetűvel is leírhatók, a megjelenítéskor alapértelmezés szerint automatikusan nagybetűre változnak).



31.11. ábra. *Adattípusok az Excelben (2003-as verzió) és a Calc-ban*

Az ezektől különböző összes többi adat mind szöveges. A táblázatkezelő üzenetei szintén a cellaterületen jelennek meg, nem cellatartalmak!

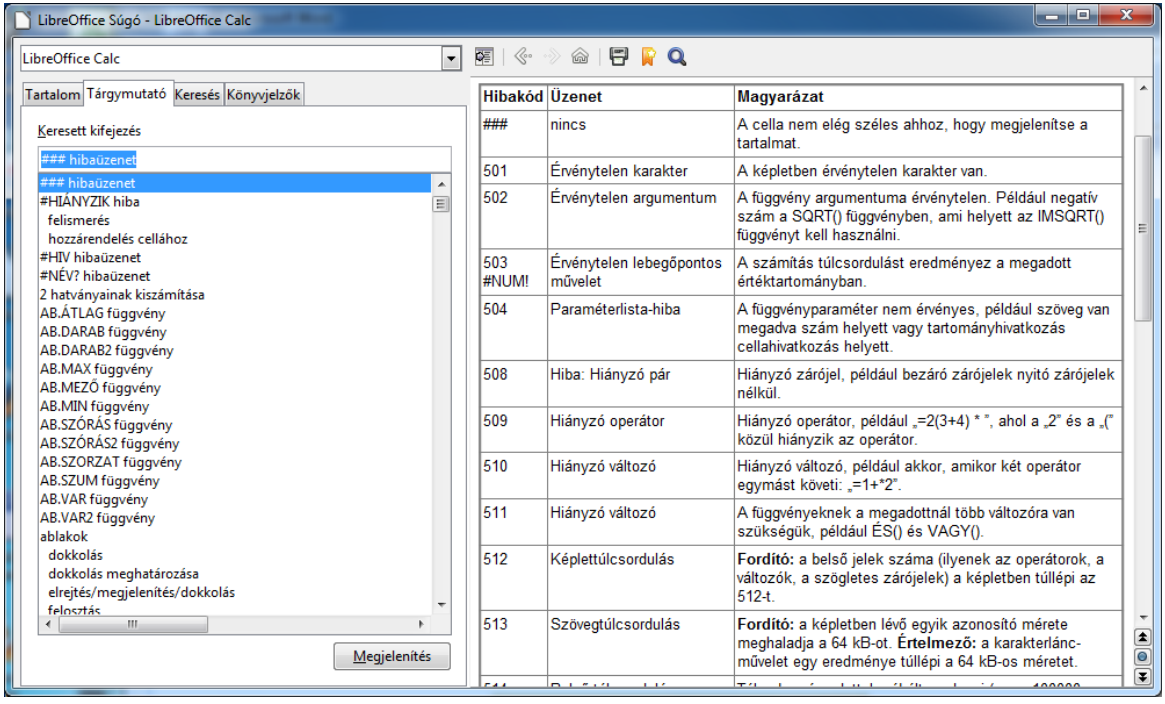
Ugyanígy szöveg lesz a dátumnak szánt, de elrontott formátumban megadott idő- és dátumadat is, pl. 1999.2.29.; 23:62 illetve 2:4:5:6.

Ha egy adat formailag szám vagy logikai típusú, de a használata szövegként célszerű (pl. személyi szám része), akkor írjunk elé ’ (aposztróf) jelet. Az aposztróf hiányában a táblázatkezelő az alapértelmezésnek megfelelő típusba sorolja ezeket az adatokat is.

Az adatokkal kapcsolatban még három fontos fogalmat kell tisztázni: mi a cella tényleges értéke, mi a tartalma és mi a megjelenített értéke, azaz: mit és hogyan látunk a cellaterületen. Formailag ezek gyakran egyformák, de lehetnek akár mind különbözők is!

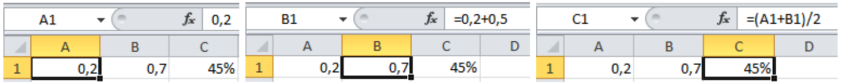
Nézzük a 31.13. ábrát! Az A1-es cellában a tartalom, a megjelenített érték és a tényleges érték egyaránt 0,2. A B1-es cellánál a megjelenített és a tényleges érték 0,7; de a tartalom az =0,2+0,5 képlet. A C1-es cellában a tartalom az =(A1+B1)/2 képlet. A tényleges érték ennek a képletnek az eredménye, a 0,45. A megjelenített érték pedig 45%.

A cella tartalma és tényleges értéke mindig összefügg egymással, de a megjelenített érték lehet az első kettőtől

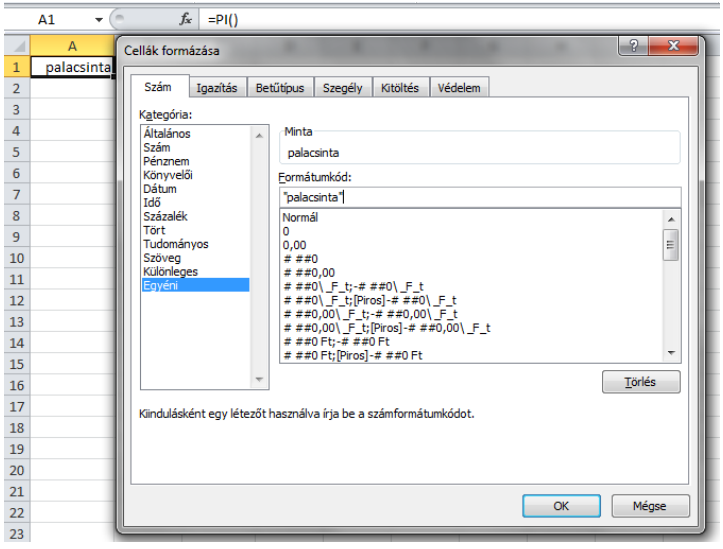


31.12. ábra. *A Calc súgója a hibaüzenetekről*

teljesen független is. Ez annak a következménye, hogy a megjelenítést a felhasználó szabályozhatja. A 31.14. ábrán az A1 cella tartalma az =PI() képlet, tényleges értéke a pi szám 15 jegy pontossággal, a megjelenített érték pedig az egyéni számformátum miatt a palacsinta szó.



31.13. ábra. *Cellák valódi tartalma, tényleges értéke és megjelenített értéke*



31.14. ábra. *Eltérés a cella tartalma és megjelenített értéke között*

Az adattípusok megismerése és hibátlan használata nem csak az informatikusoknak, de mindenkinek fontos és hasznos, hiszen ennek a tudásnak birtokában vagyunk képesek megoldani a következő táblázatkezelési problémákat:

1. Mi egy adattípus értékkészlete, azaz mi az az érték, ami az adott adattípussal még megadható, mi az, ami nem?
2. Milyen problémák adódhatnak abból, ha túllépjük ezeket a határokat? (Például, ha egy cellában lévő képlet értéke túl nagy lesz, akkor úgynevezett túlcsorduláshiba keletkezik, amit a táblázatkezelő hibaüzenettel jelez is.)
3. Milyen pontosságot várhatunk a táblázatkezelőtől egész, illetve törtszámok használata esetén?
4. Milyen problémák adódhatnak kerekítési hibákból?

A táblázatkezelő a valós számokat lebegőpontos kódolással tárolja. Amint már tudjuk, ebben a kódolásban létezik a gépi epszilon. A 31.16. látható Excel-táblázat ennek a becsült értékét számítja ki és mutatja meg.

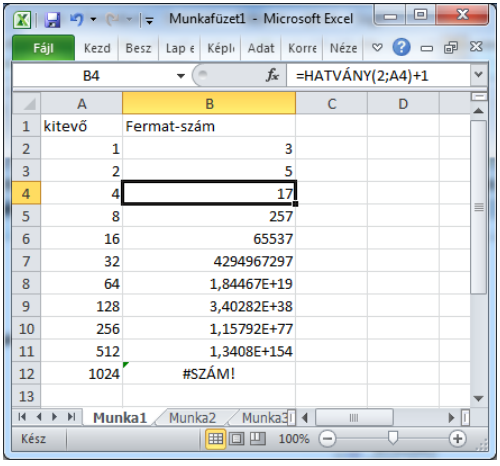
Jól megfigyelhető, hogy a **B** oszlopban szereplő kis törtérték a **C** oszlopban már kerekítve látszik, és az 52. sorban a törtrész már „elveszett”, így természetesen C52 értéke már 1 lesz, mint ahogyan a C oszlop minden, az 52. sor utáni értéke is! Lásd valós számok kódolása!

Az, hogy a táblázatkezelőben kiszámított értékek nem mindig pontosak, gazdasági és mérnöki számítások jó részében biztosan elegendő (pl. egy híd terhelésvizsgálatánál is). Űrszondák navigálására vagy komoly matematikai problémák megoldására pedig nem a táblázatkezelő a megfelelő eszköz.

📖 *Első feladat (önálló gyakorlásra)*

Gondolkodjunk el azon, hogy vajon mekkora szám hozzáadása nem változtatja meg az egymilliót!

📖 *Második feladat (önálló gyakorlásra)*



31.15. ábra. *Túlcsordulás bemutatása Fermat-féle számokkal Excel 2010-ben*

Határozzuk meg az Excel 2003, Excel 2010 és Calc táblázatkezelőkben a legnagyobb, még pontosan ábrázolható egész számot; a legnagyobb, tudományos megjelenítéssel még értelmes valós (lebegőpontos) számot!

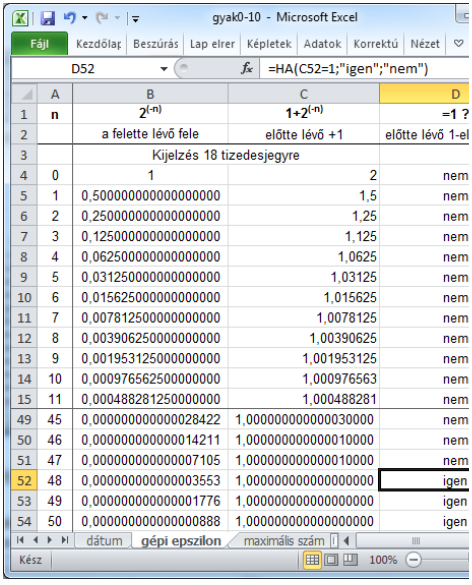
Harmadik feladat (önálló gyakorlásra)

A gépi epszilonos példa hatására gondoljuk át, hogy hogyan célszerű megoldani táblázatkezelőkben azt a feladatot, amikor olyan számsort kell összeadni, amiben vegyesen szerepelnek nagyon kicsi és nagyon nagy számok!

31.5.3. Kifejezések

Adatok, cellacímek, műveleti jelek és függvények felhasználásával táblázatainkban kifejezések készíthetők. A kifejezéseket cellákban helyezhetjük el.

A használható műveleti jelek szám típusú adatokra az összeadás, kivonás, szorzás, osztás és hatványozás. A hatványozás a gyökvonást is jelenti, hiszen amint tudjuk egy szám a k-adik gyöke az 1/k-adik hatványával



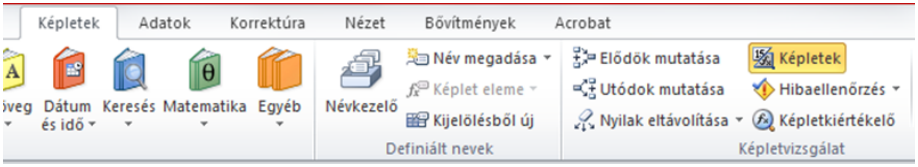
	A	B	C	D
1	n	$2^{(-n)}$	$1+2^{(-n)}$	=1 ?
2		a felette lévő fele	előtte lévő +1	előtte lévő 1-el
3		Kijelzés 18 tizedesjegyre		
4	0	1	2	nem
5	1	0,500000000000000000	1,5	nem
6	2	0,250000000000000000	1,25	nem
7	3	0,125000000000000000	1,125	nem
8	4	0,062500000000000000	1,0625	nem
9	5	0,031250000000000000	1,03125	nem
10	6	0,015625000000000000	1,015625	nem
11	7	0,007812500000000000	1,0078125	nem
12	8	0,003906250000000000	1,00390625	nem
13	9	0,001953125000000000	1,001953125	nem
14	10	0,000976562500000000	1,000976563	nem
15	11	0,000488281250000000	1,000488281	nem
49	45	0,000000000000028422	1,000000000000030000	nem
50	46	0,000000000000014211	1,000000000000010000	nem
51	47	0,000000000000007105	1,000000000000010000	nem
52	48	0,000000000000003553	1,000000000000000000	igen
53	49	0,000000000000001776	1,000000000000000000	igen
54	50	0,000000000000000888	1,000000000000000000	igen

31.16. ábra. *Gépi epszilon meghatározása (Excel 2010)*

egyezik meg, pl. negyedik gyököt az 1/4 hatványra emeléssel számíthatjuk. A kifejezések felírásakor a felhasznált műveleteknek a matematikában megismert tulajdonságait kell használnunk. Ezek a tulajdonságok: asszociativitás, disztributivitás, kommutativitás és a végrehajtási sorrend, a precedencia.

Ugyanezek a műveletek végrehajthatók a dátum és idő típusú adatokkal is, mivel ezeket a táblázatkezelők számként használják, de az összeadás és a kivonás kivételével használatuk nem célszerű.

A szöveges adatokon elvégezhető művelet az összefűzés, ennek jele az & jel. A művelet végrehajtása nagyon egyszerű: az eredményt úgy kapjuk meg, hogy az első operandus értéke után odaírjuk a második operandus értékét.



31.17. ábra. *Képletek megjelenítésének beállítása Excel 2010-ben*

Használhatjuk még a hasonlítást (akár különböző típusok esetében is, vállalva azt, hogy az „eredmény” értelmetlen vagy haszontalan), amelynek eredménye IGAZ vagy HAMIS.

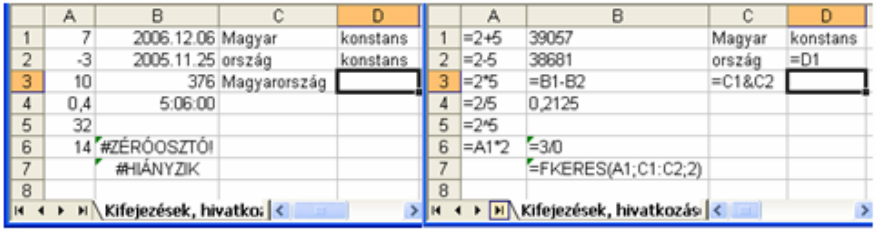
A kifejezésekben az adatok azonosítására cellahivatkozások is elhelyezhetők, ill. az adatok kezelésére típustól függően függvények is használhatóak, ezekkel később még foglalkozunk.

A kifejezéseket a szöveges adatoktól való megkülönböztetés végett egy megkülönböztető jellel kell kezdeni. Ez a jel az = (esetleg a + vagy -).

Ha sikerült formailag helyesen megadni a kifejezést, akkor a táblázatkezelő alapértelmezésben rögtön a számított eredményt jeleníti meg a cellatartományon. Ha mégis a képletet szeretnénk látni, akkor álljunk rá a cellára, illetve a megfelelő lapon a **Képletek** kapcsoló aktivizálásával kérhetjük az általános képlet-megjelenítést. Ilyenkor az esetleges hibaüzenet helyett is az eredeti kifejezést láthatjuk a munkalapon.

A képlet megjelenítő kapcsoló a 2003-as Excelben az **Eszközök/Beállítások...** parancs **Megjelenítés** lapján, a 2010-es Excelben a **Képletek** menüszalag **Képletvizsgálat** csoportjában, a Calc-ban pedig az **Eszközök/Beállítások...** menüpont **LibreOffice Calc/Nézet** lapján érhető el.

Érdemes átgondolni, hogy a kezdetben üres cellába beírt adat egyben meghatározza a végrehajtható műveleteket is. Ez az elv hasonló ahhoz, ahogy a modern objektum-orientált programozási nyelveknél csak futási időben derül ki egyes műveletek pontos végrehajtási módja (késői kötés).



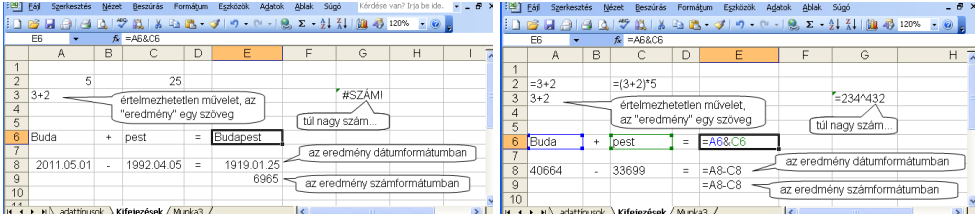
	A	B	C	D
1	7	2006.12.06	Magyar	konstans
2	-3	2005.11.25	ország	konstans
3	10	376	Magyarország	
4	0,4	5:06:00		
5	32			
6	14	#ZÉRÓOSZTÓ!		
7		#HIÁNYZIK		
8				

Kifejezések, hivatko

	A	B	C	D
1	=2+5	39057	Magyar	konstans
2	=2-5	38681	ország	=D1
3	=2*5	=B1-B2	=C1&C2	
4	=2/5	0,2125		
5	=2^5			
6	=A1*2	=3/0		
7		=FKERES(A1;C1:C2;2)		
8				

Kifejezések, hivatkozás

31.18. ábra. *Kifejezések, hivatkozások 1. – eredmény- és képletes megjelenítés*



31.19. ábra. *Kifejezések, hivatkozások 2. – eredmény- és képletes megjelenítés*

31.6. Blokkműveletek

Munkánk hatékonyságát jelentősen megnöveli, ha nemcsak egyes cellákat, hanem cellákat tartalmazó téglalap alakú tartományokat (blokkokat) is tudunk kezelni.

31.6.1. Megadás, kijelölés, törlés

Képletekben a blokkokat általában bal felső és jobb alsó celláik egymástól kettősponttal elválasztott koordinátáival adjuk meg (például A2:G5).

A blokkok elvileg bármely átellenes cellacímmel azonosíthatók, de a táblázatkezelő minden más megadási formát lecseréli a bal felső, jobb alsó koordinátákra.

Blokkot kijelöléssel is megadhatunk, amit billentyűzettel vagy egérrel, többféle módon is végrehajthatunk. A kijelölési módszerek a 31.25. táblázatban találhatók. A kijelölt blokkot a táblázatkezelő inverz színekkel jeleníti meg.

31.25. táblázat. *Kijelölések*

Egy blokk kijelölése (egyszerű kijelölés)	
billentyűzettel	1. Álljunk a blokk egyik sarkába, nyomjuk le és tartsuk lenyomva a SHIFT billentyűt, majd a nyilakkal mozogjunk a blokk ellentétes sarkára. 2. Álljunk a blokk egyik sarkába, nyomjuk le az F8 billentyűt, majd a nyilakkal mozogjunk a blokk ellentétes sarkára.
egérrel	A bal gombbal kattintsunk rá a blokk egyik sarkára, majd a gomb nyomva tartása mellett húzzuk az egérkurzort a blokk másik sarkába.
Több blokk kijelölése	
billentyűzettel	A SHIFT és a nyilak segítségével jelöljük ki az első blokkot. A SHIFT+F8 billentyűk lenyomása után átléphetünk a következő blokkra, aminek kijelölése után újra a SHIFT+F8 használatával a továbbiakra.
egérrel	1. Nyomjuk le a SHIFT+F8 billentyűket, majd egyesével jelöljük ki a blokkokat. 2. Az első blokk kijelölése után tartsuk lenyomva folyamatosan a CTRL billentyűt a további blokkok kijelöléséhez.
Kijelölés megszűntetése	
billentyűzettel	Nyomjuk le valamelyik kurzormozgató billentyűt.
egérrel	Kattintsunk a táblázat egy semleges részére.

Nagy méretű blokkok kijelölésére alkalmazhatjuk a SHIFT billentyűt együtt a gyors mozgás lehetőségeinél tárgyalt CTRL+←, CTRL+→, ... billentyűkombinációkkal; illetve használhatjuk a kijelöléshez a név mezőt (beírjuk a megfelelő blokk koordinátáit; egymástól pontosvesszővel elválasztva több blokk is megadható).

A táblázatkezelőkben kétféleképpen lehet törölni. Az egyik a cellák tartalmának és egyéb jellemzőinek törlése, amelynek hatása más cellák tartalmára nem terjed ki. A másik az úgy nevezett teljes törlés. Ennek a törlésnek nemcsak a törölt cellára vagy blokkra van hatása, hanem a környezetre is.

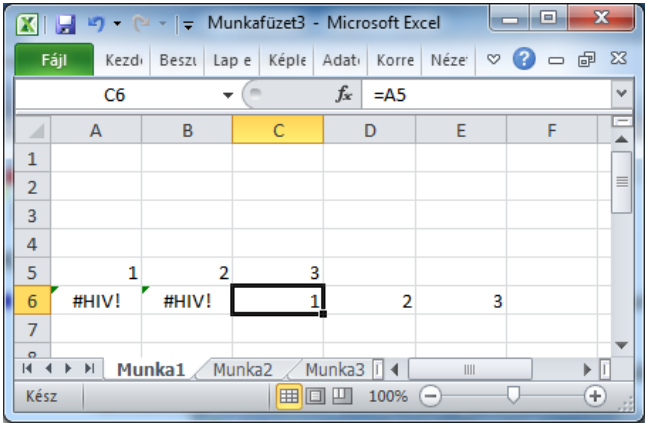
Az első szerint a kijelölt blokkok tartalma törölhető a DELETE billentyűvel. Törlésre a menüből is van lehetőség, Excel 2003-ban és Calc-ban a **Szerkesztés/Tartalom törlése**, Excel 2010-ben pedig a **Kezdőlap/Szerkesztés/Törlés** parancs választásával. Ezt alkalmazva a törölt területen a tartalom, a forma és még néhány tulajdonság tűnik el, de minden más változatlanul megmarad. Megmarad például minden olyan hivatkozás is, amely a törölt blokkra vagy valamelyik cellájára vonatkozik.

A másikfajta törlés a cellát, illetve a blokkot úgy törli, hogy a mellette lévő cellákat a törölt területre húzza. Ebben az esetben minden olyan hivatkozás, amely a törölt terület celláira vonatkozik hibás lesz, az átmozgatott cellákra való hivatkozás pedig az új helyüknek megfelelőre változik. Ennél a törlésnél meg kell adnunk a törlés módját is, amely azt határozza meg, hogy a törlendő blokk helyét a program a cellák balra vagy felfelé tolásával töltse fel. A táblázatkezelőkben ezzel a módszerrel lehetőség van azon oszlopok, illetve sorok teljes törlésére is, amelyekben a kijelölt blokk elhelyezkedik. A törlés végrehajtása a **Szerkesztés** menü **Törlés...** pontjával (2003-as Excel), **Szerkesztés** menü **Cellák törlése...** pontjával (Calc), és **Kezdőlap** menüszalag **Cellák** csoportjával (2010-es Excel) lehetséges.

31.6.2. Másolás, mozgatás

A kijelölt blokk a táblázat tetszőleges részére másolható vagy átmozgatható.

Ehhez jelöljük ki a másolni vagy mozgatni kívánt blokkot, majd válasszuk a **Szerkesztés** menü **Másolás** vagy **Kivágás** parancsát (Excel 2003 és Calc), illetve a **Másolás** vagy **Kivágás** ikont a **Kezdőlap** menüszalagról (Excel 2010). Jelöljük ki az új blokk helyét, illetve álljunk annak leendő bal felső sarkára, majd válasszuk a **Szerkesztés** menü **Beillesztés** parancsát (Excel 2003 és Calc), illetve a Beillesztés megfelelő lehetőségét a



31.20. ábra. *Hivatkozás a munkalapról lefuttatva (Excel 2010)*

Kezdőlap menüszalagról (Excel 2010).

Ha a másoláskor a másolt cellákban képletek vannak, akkor ügyelni kell arra, hogy a képletben szereplő cellahivatkozások csak úgy módosulhatnak, hogy ezzel kerüljünk a „táblázaton kívülre”. A logika működését jól szemléltetik következő egyszerű feladatok: mi lesz az eredmény, ha a 31.20. ábra szerint a C6 cella tartalmát a B6 vagy A6 cellába másoljuk (#HIV!).

Aktivitás: Töröljük ki a 31.20. ábra B5-ös celláját a cellák balra mozgatásával. Figyeljük és magyarázzuk meg a történteket!

Ezek a funkciók közvetlenül is hívhatók gyorsbillentyűkkel (CTRL+C, CTRL+X, CTRL+V), továbbá a **Szokásos** eszköztár (2003-as Excel és Calc esetén) megfelelő ikonjaival.

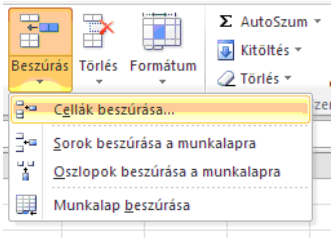
Egérrel a „fogd és vidd” módszert alkalmazva mozgathatunk, másolhatunk (lásd az operációs rendszerek és a szövegszerkesztés fejezeteket). A kijelölt cella vagy blokk jobb alsó sarkán lévő kitöltőfülre vagy kitöltőjelre

állva az egér bal oldali gombját lenyomva a szomszédos cellákba másolhatjuk a kijelölt blokk, cella tartalmát úgy, hogy a egeret abba az irányba mozgatjuk, ahova a másolatot el szeretnénk helyezni. Vigyázat, a blokk tartalmából csak annyit másol át a program, amennyi az egérrel bejárt területre elfér. Ha a blokk belseje felé mozgunk, akkor a bejárt terület tartalmát törli a program! Ezeket a műveleteket az egér jobb gombjával történő kattintás után megjelenő helyi menü segítségével is végrehajthatjuk.

Nagyméretű blokkoknak egyforma adatokkal való feltöltésére hatékonyabb módszer lehet a következő:. Vigyük vágólapra a beillesztendő tartalmat, majd jelöljük ki a blokkot, szerkesztő módban szúrjuk be a vágólap tartalmát, és nyomjunk CTRL+ENTER-t. Egy másik lehetséges eljárás szintén a kitöltőfület használja fel. Először ki kell tölteni egy oszlopot. A mellette lévő oszlop kitöltéséhez elegendő az első cellába beírni a tartalmat, és ezt lehet másolni azokba a cellákba, amelyek üresek, de a mellettük lévő cellákban van tartalom. A másoláshoz csak a kitöltőfülre történő kell duplán kattintani és a táblázatkezelő automatikusan kitölti az oszlopot üres celláit a mellette lévő oszlop kitöltött cellái mellett vagy addig, amíg egy tartalommal rendelkező cellát el nem ér.

31.6.3. Beszúrás

Sokszor szükség lehet arra, hogy egy táblázatba utólag üres sorokat vagy oszlopokat szúrjunk be. Ekkor jelöljünk ki egy blokkot úgy, hogy bal felső sarka ott legyen, ahol a beszúrást el szeretnénk végezni, és annyi sort vagy oszlopot tartalmazzon, ahányat be szeretnénk szúrni. Végül válasszuk a **Beszúrás** menü **Sorok** vagy **Oszlopok** parancsát (Excel 2003 és Calc), illetve a **Sorok** vagy **Oszlopok beszúrása** opciót a **Kezdőlap** menüszalag **Cellák** csoportjában (Excel 2010). Az eredmény: a megfelelő számú üres sor vagy oszlop bekerül a táblázatunkba.



31.21. ábra. *Beszúrási lehetőségek (Excel 2010)*

Ha nem teljes sort/oszlopot szeretnénk beszúrni, hanem csak néhány cellát, akkor a fentivel azonos kijelölés után a **Beszúrás/Cellák...** paranccsal dolgozunk (Excel 2003 és Calc), illetve a **Cellák beszúrása...** opciót választjuk a **Kezdőlap** menüszalag **Cellák** csoportjában (Excel 2010). Ilyenkor meg kell adni, hogy a gép a beszúrandó blokk helyét milyen módon alakítsa ki (lehetséges a cellák eltolása lefelé ill. jobbra).

Önellenőrzés

1. Jelöljük meg a következő funkciók közül azokat, amelyeket a táblázatkezelő programok támogatnak!

- [ ] Nyomtatás
- [ ] Adatok rendezése
- [ ] Diagramkészítés
- [ ] Keresés és csere
- [ ] Helyesírás-ellenőrzés
- [ ] Spamszűrés

2. Adjuk meg a táblázatkezelőkkel történő feladatmegoldás lépéseinek helyes sorrendjét! (válaszok bekeverve)

- [ ] – A megoldás felépítése papír-ceruza módszerrel, számítógép nélkül
- [ ] – Ha szükséges: a (rész)megoldás másolása
- [ ] – A feladat megértése/elemezése
- [ ] – A táblázatkezelős megoldáshoz szükséges apparátus összegyűjtése
- [ ] – A (rész)megoldás előállítása egy cellában/egy példányban

3. Jelöljük meg a következő funkciók közül azokat, amelyeket tipikusan máshogy hajtunk végre a táblázatkezelőben kicsi, illetve nagy méretű adathalmaz esetén:

- ☐ Fájl megnyitása
- ☐ Fájl mentése
- ☐ A táblázatkezelő felületének testreszabása
- ☐ Kijelölés
- ☐ Gyors mozgás

4. Adjuk meg, hogy hogyan igazítja a táblázatkezelő a következő adatokat! (Az alapértelmezett beállítások szerint) (b = balra, j = jobbra, k = középre, n = a megadottak alapján nem lehet eldönteni)

☐ – =3+2
☐ – 3+2
☐ – ez egy szöveges adat
☐ – 3.4.5
☐ – 3.4.5.6
☐ – 3:4:5
☐ – 3:4:5:6
☐ – 2012.12.21
☐ – 2012.12.21
☐ – 1899.12.31
☐ – IGAZ
☐ – IGEN

5. Az alábbiak közül melyik operátor alkalmazható hiba nélkül két (tetszőleges értékű) szöveges operandus esetén?

☐ – (törtjel) mint osztás

☐ – > (nagyobb jel) mint összehasonlítás

☐ – = (egyenlő jel) mint összehasonlítás

☐ – & (és jel) mint összefűzés

☐ – , (vessző jel) mint összefűzés

13. LECKE

Címzési módok, függvények

A lecke először a címzési módokat tárgyalja, majd a függvények használatának első részét tekinti át. Az anyag elsajátítása – a téma rendkívül szerteágazó volta miatt – csak akkor tekinthető sikeresnek, ha az órákon szerepelt gyakorlati feladatokat is bizonyos jártassággal meg tudja oldani a tanuló. Ennek megfelelően a lecke önálló feldolgozására fordítandó idő erősen függ az előképzettségtől, diákonként akár többszörös szorzók is adódhatnak.

31.7. Relatív, vegyes és abszolút címek

Ha olyan cellatartalmat másolunk, amely hivatkozást is tartalmaz, akkor azt tapasztaljuk, hogy a hivatkozás a másolás során bizonyos esetekben módosul. Írjunk be például az A1 cellába egy egyest, az A2-be pedig a következő képletet:

=A1+1.

Ennek eredményeképpen A2 cellaterületén eggyel nagyobb értéket látunk (a cella tartalma nem a 2 szám, hanem egy képlet, ami hivatkozást tartalmaz). Másoljuk ezt a képletet lefelé néhány sorba! Az első néhány természetes számot fogjuk látni az oszlopban.

Ennek az az oka, hogy a hivatkozás a cellák között egy logikai kapcsolatot határoz meg, ami a másolás során öröklődik. Így A2 cella ugyanolyan „kapcsolatban” áll az A1 cellával, mint A3 az A2-vel, A4 az A3-mal és így tovább. Az A3 cella értéke A2+1=3 lesz, az A4 értéke 4 stb.

Természetesen ez a feladat a táblázatkezelőkben más egyszerű eszközökkel is megoldható:

Beírjuk a sorozat első két elemét két cellába, kijelöljük őket, és az egér bal gombjával lehúzzuk a kitöltőfület.

Írjuk be a sorozat első elemét egy cellába, jelöljük ki a feltölteni kívánt blokkot, majd válasszuk Calc-ban és Excel 2003-ban a **Szerkesztés/Kitöltés/Sorozatok...** menüpontot, Excel 2010-ben pedig a **Kezdőlap/Szerkesztés/Kitöltés** menüpontot, és adjuk meg sorozat paramétereit (számtani vagy mértani, lépésköz, stb.).

A cellák közötti logikai kapcsolatmegadás ezen – nagyon gyakran alkalmazott – módját *relatív hivatkozás*nak nevezzük. A hivatkozás természetesen nemcsak szomszédos, hanem egymástól távoli cellákra is használható, erre a későbbiekben látunk még példákat.

A relatív hivatkozás másoláskor mindig megváltozik. Függőleges másoláskor csak a sorkoordináta, vízszintes másoláskor csak az oszlopkoordináta, mindkét irányú másoláskor mindkettő.

Sok feladat megoldásához a relatív hivatkozás a megfelelő eszköz, más esetekben azonban szükség lehet arra, hogy ezt a logikai kapcsolatot felülbíráljuk.

Negyedik feladat

Egy autó különböző utakat tesz meg (ezek az adatok kilométerben egy táblázatoszlopban adottak), és ki szeretnénk számolni az utakhoz tartozó benzinköltséget. Egy-egy cellában található a benzinár és az autó fogyasztása 100 km-en, amelyeket a megoldást előállító képlet használ majd.

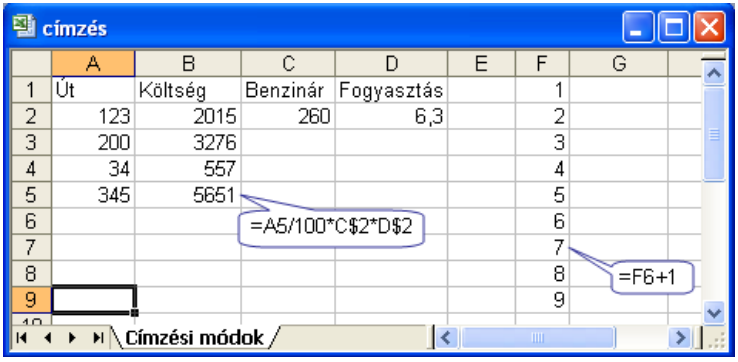
A képlet a B2-es cellára a következő: **=A2/100*C2*D2** (lásd 31.22. ábra.)

Ha lefelé lemásoljuk a képletet a többi adat mellé, akkor meglepetésünkre az eredmény az összes másolással feltöltött cellában 0 lesz. A problémát elemezve láthatjuk, hogy a relatív hivatkozás miatt a harmadik sorban **=A3/100*C3*D3-ra**, a negyedikben **=A4/100*C4*D4-re**, . . . változik a képlet, de ezek a C és D cellák már üresek, így értékük 0. Ez az oka annak, hogy rossz eredményt kaptunk.

Első közelítésben gondolhatunk arra, hogy lemásoljuk a benzinár és a fogyasztás adatokat a megfelelő C és D cellákba. Ez azonban úgynevezett adatredundanciát okozna, és. így például benzinár változása nehezen lenne nyomon követhető: egyesével kellene megkeresni és átírni azon cellákat, ahol ezt az adatot újra elhelyeztük. Ez komoly hibalehetőséget rejt magában.

Így a következő fontos szabályt fogalmazzuk meg:

Egy adott tulajdonságot leíró adat a táblázatban csak egyszer helyezhető el. Más előfordulásait csak hivatkozással állíthatjuk elő.



31.22. ábra. *Különböző címzési módok*

Vigyázat, ez nem azt jelenti, hogy ugyanaz az érték nem szerepelhet többször is a táblázatban! Ez természetesen előfordulhat akkor, ha az érték minden előfordulását más tulajdonság leírására használjuk (pl. van két diák, akik ugyanazon a napon születtek, ekkor a születési dátumuk azonos, mégis mindkettőt külön tároljuk).

Az előző probléma megoldásához tehát szükség lenne arra, hogy a C és a D oszlop megfelelő celláira való hivatkozás ne „vándoroljon” lejjebb, hanem mindig a második sorra mutasson, azaz rögzíteni kell a hivatkozásban a második sort. Ez a táblázatkezelőkben úgy valósítható meg, hogy a rögzíteni kívánt koordináta elé egy $ jelet írunk. A lefelé is szabadon másolható megoldó képletet tehát a következő: **=A2/100*C$2*D$2**.

Most ez volt a megoldás, de természetesen a feladat jellegétől függően szükség lehet az oszlophivatkozás rögzítésére is. Ekkor az oszlopkoordináta elé írunk $ jelet. A hivatkozásnak ezt a módját *abszolút* sor- vagy oszlophivatkozásnak nevezzük. Ha a sor- és az oszlophivatkozás egyaránt rögzített, akkor abszolút hivatkozásról beszélünk, ha csak az egyik koordinátára vonatkozik a rögzítés, akkor azt *vegyes hivatkozás*nak nevezzük.

Ha egy hivatkozást abszolúttá tettünk (sor- és oszlopkoordinátáját egyaránt rögzítettük), akkor már nem a cellák egymáshoz viszonyított (relatív) elhelyezkedését adjuk meg, hanem egy olyan általánosan érvényes képletet/képletrészt kaptunk, amely tetszőleges másolás után is pontosan ugyanarra a cellára vonatkozik.

Mikor milyen hivatkozást használjunk?

A lehetséges hivatkozástípusok a következők:

A1, A$1, $A1, $A$1.

Egy megoldásban alkalmazandó rögzítést a feladat jellege határozza meg. Át kell gondolni, hogy egy cellára már jó képlet relatív hivatkozásai a másolás során hogyan változnak meg, és a változások közül melyek jók számunkra, ill. melyek okoznak hibát. Ha egy koordináta változása hibát okoz, akkor azt rögzíteni kell alkalmazni. Ezt *szükséges rögzítésnek* nevezzük.

Ha egy koordináta rögzítése nem okoz hibát, de a feladat megoldása szempontjából nem indokolt, akkor *felesleges rögzítésnek* minősül. Például: ha egy képletet csak függőlegesen fogunk másolni, akkor a hivatkozott cella oszlopazonosítóját felesleges rögzíteni, mert az a másoláskor úgysem fog változni.

Ha a rögzítés hibát okoz, akkor *hibás rögzítésről* beszélünk.

Ötödik feladat (önálló gyakorlásra)

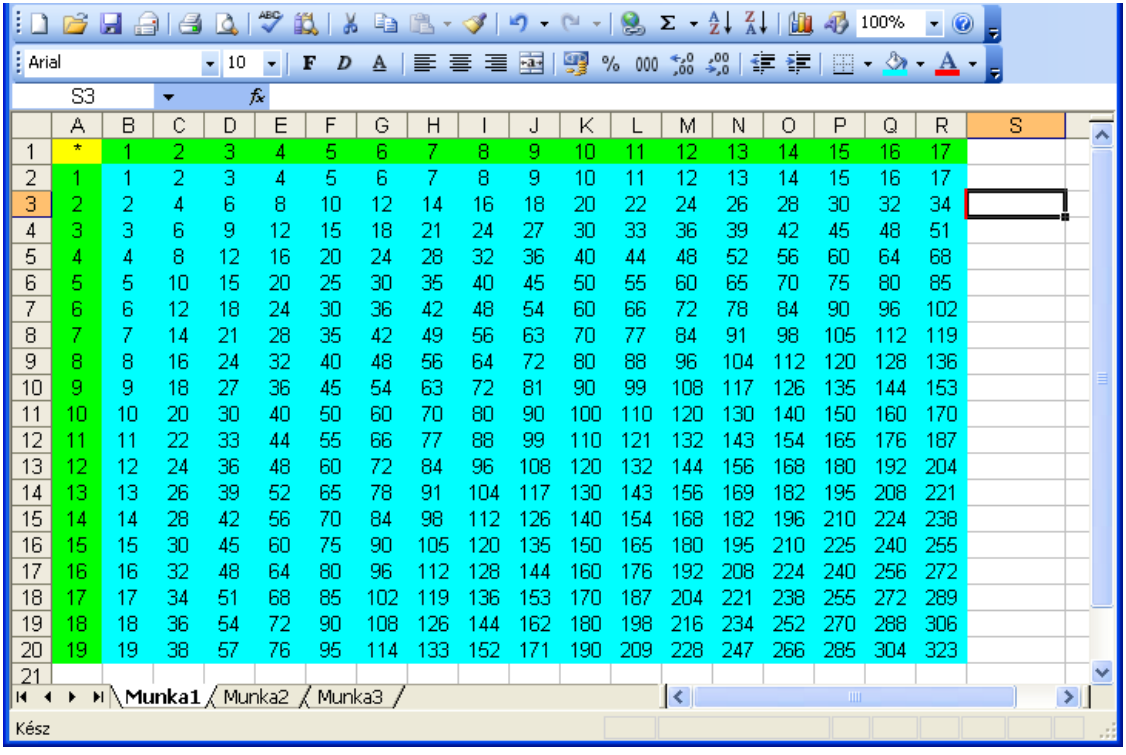
Készítsünk faktoriális táblázatot! Az A oszlopot töltsük fel pozitív egész számokkal (pl. 1-től 15-ig). A B1-es cellába írjuk be az 1-es számot (ez az 1!). A B2-es cellába készítsünk olyan képletet, amely az A2-ben található szám (a 2) faktoriálisát állítja elő. A képlet legyen lefelé másolható!

Segítség: az n szám faktoriálisát most rekurzív képlettel határozzuk meg, így n! = n(n – 1)!*

Milyen hibajelenséget tapasztalunk, ha túl nagy egész szám faktoriálisát akarjuk kiszámoltatni?

Hatodik feladat (önálló gyakorlásra)

*Készítsünk szorzótáblát! Az A1 cellába írjunk be egy * jelet, majd az A2-től A11-ig és a B1-től K1-ig terjedő tartományt töltsük fel növekedő számsorozattal egytől kezdve. A táblázat további celláit töltsük ki egy darab (!) másolható képlet előállításával! 31.23. ábra.*

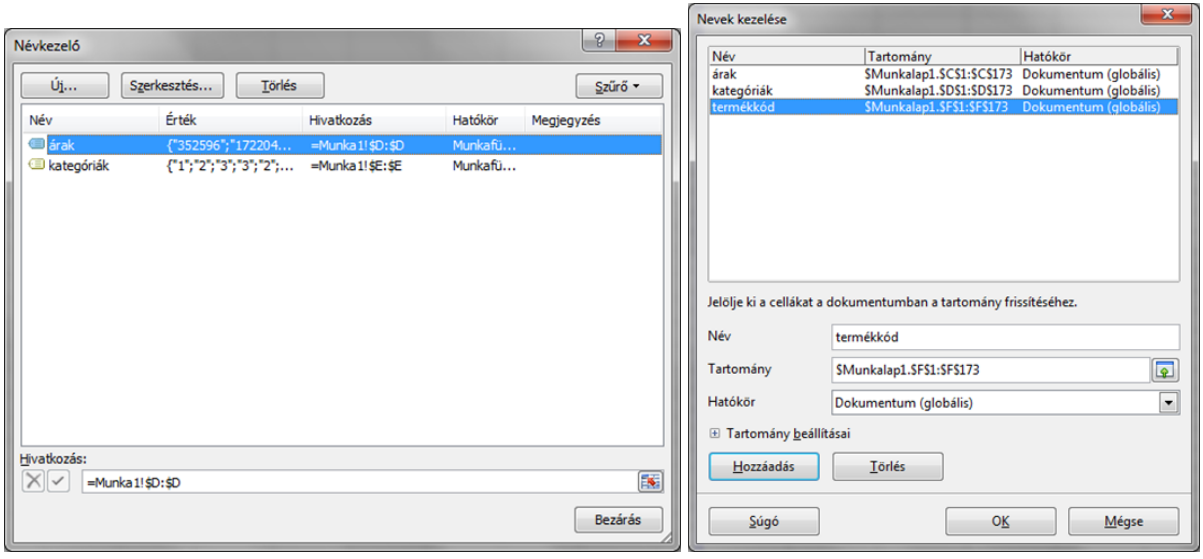


*	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68
5	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85
6	6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96	102
7	7	14	21	28	35	42	49	56	63	70	77	84	91	98	105	112	119
8	8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128	136
9	9	18	27	36	45	54	63	72	81	90	99	108	117	126	135	144	153
10	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170
11	11	22	33	44	55	66	77	88	99	110	121	132	143	154	165	176	187
12	12	24	36	48	60	72	84	96	108	120	132	144	156	168	180	192	204
13	13	26	39	52	65	78	91	104	117	130	143	156	169	182	195	208	221
14	14	28	42	56	70	84	98	112	126	140	154	168	182	196	210	224	238
15	15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255
16	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272
17	17	34	51	68	85	102	119	136	153	170	187	204	221	238	255	272	289
18	18	36	54	72	90	108	126	144	162	180	198	216	234	252	270	288	306
19	19	38	57	76	95	114	133	152	171	190	209	228	247	266	285	304	323

31.23. ábra. *Szorzótábla*

Nevek használata

Cellahivatkozások/blokkok esetében azért használunk neveket, mert így a hivatkozások egyszerűbbé válnak (különösen kényelmes lehet ez másolás során, esetleg a táblázatkezeléshez kevésbé értő felhasználók számára). A kijelölt blokk úgy nevezhető el, hogy a szerkesztőléc bal oldali név mezőjére kattintunk, majd beírjuk a nevet.



31.24. ábra. *Nevek kezelése az Excel 2010 és Calc programokkal*

A nevekhez tartozó blokkok címe módosítható Excel 2003-ban a **Beszúrás** menü **Név** pontján belüli **Név megadása...** ablakban, Excel 2010-ben a **Képletek** menüszalagon, a **Definiált nevek** csoportban a **Névkezelő** nyomógombra kattintva, Calc-ban pedig a **Beszúrás/Nevek/Kezelés** paranccsal. A Névkezelő Calc-ban a **Névdoboz** (**Képlet** eszköztár) lenyíló listájából is elérhető a **Nevek kezelése...** pont alatt. A nevek törlése is ugyanezen pontok alatt lehetséges.

Ha a név mező segítségével nevezünk el egy cellát vagy blokkot, akkor ezzel olyan nevet hozunk létre, ami abszolút módon hivatkozik az adott cellára vagy blokkra. Ha a Névkezelőt használjuk az új név létrehozásához, akkor nem csak abszolút, hanem relatív és vegyes hivatkozással is hozhatunk létre nevet. Ebben az esetben a program a hivatkozást ahhoz a cellához viszonyítja, amelyiken állva a Névkezelőt elindítottuk.

A cella/blokkhivatkozásokon kívül képleteket és adatokat is tudunk nevesíteni.

A névadási szabadság nem teljes. Természetes, hogy már kiadott név még egyszer pontosan ugyanúgy más blokkra nem szerepelhet, emellett egyes táblázatkezelőkben lehetnek olyan speciális védett, beépített nevek, amelyeket nem használhatunk.

Excelben és Calc-ban egyaránt érvényes, hogy név csak számot, betűt és aláhúzást tartalmazhat, az első karaktere csak betű vagy aláhúzás lehet.

32. Függvények használata

32.1. A függvények megadása

A táblázatkezelők függvényfogalma lényegében megfelel a matematikai függvényfogalomnak. (De ez az általános számítástechnikai értelmezés is.) Eszerint nulla, egy vagy több bemenő adat – más néven paraméter – felhasználásával a gép előállítja a függvényértéket, azaz a visszaadott értéket.

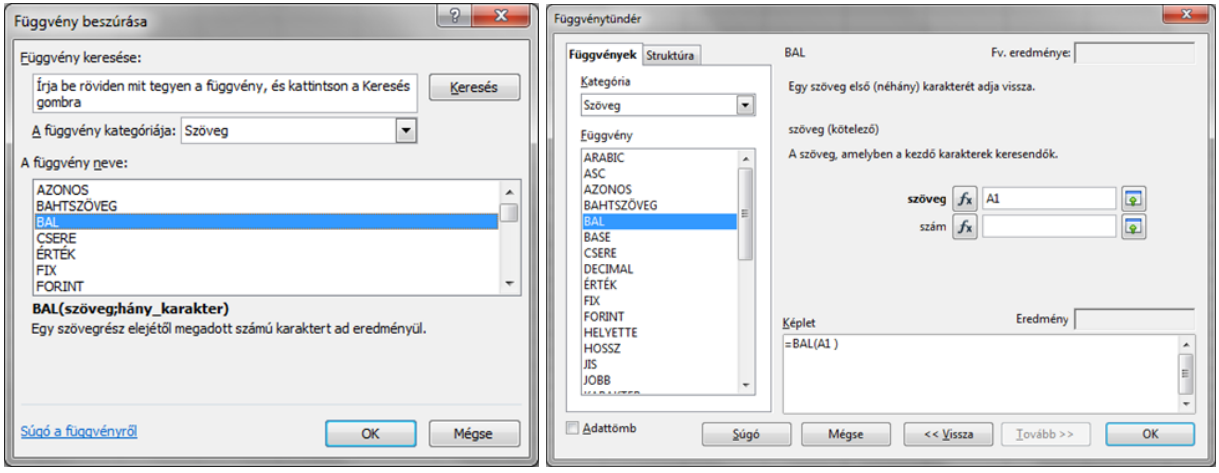
A függvények általános alakja a következő (a paramétereket argumentumoknak is nevezzük):

függvénynév(paraméter1; paraméter2; . . . paraméterN).

A függvények akár többszörösen is egymásba ágyazhatók, ilyenkor a belső függvény általában a külső paramétere.

függvénynek némelyikének 0, másik csoportjuknak 1, a harmadik csoporthoz tartozóknak több paramétere lehet. Ha nincs paraméter, akkor egy üres zárójelpárt kell írni a függvény neve után – pl. **Pi()**. Előfordulhat az is, hogy egy függvény egyes paramétereit nem kötelező minden esetben megadni.

Ugyan a speciálisabb használati esetek közé tartozik, de a hibás működés elkerülése érdekében érdemes külön figyelnünk arra, hogy egyes (általában a több paraméterrel rendelkező) függvényeknél a paraméterek megadása elmaradhat, de az elválasztó pontosvessző nem. A táblázatkezelő az elmaradt értéket ilyenkor egy alapértelmezettnek tekinthetővel helyettesíti, és így számol. Pl. az =Ha(A2>0;1;) képlet értéke 1 lesz, ha az A2 értéke nagyobb, mint nulla, de 0 (ez a 3. paraméter alapértéke) ha nem!

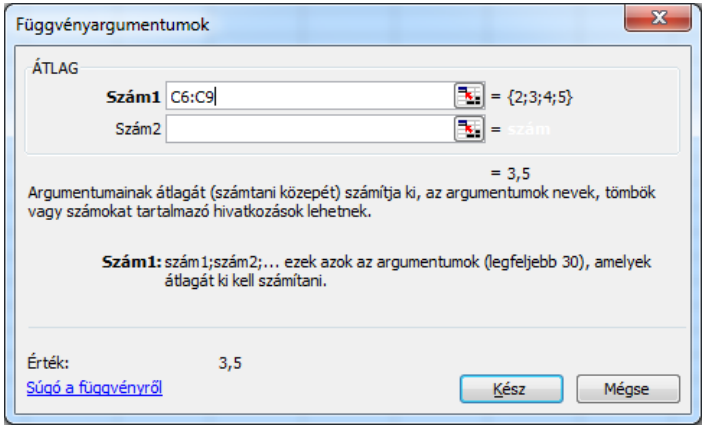


32.1. ábra. *Függvényvarázsló vagy tündér indítása az Excelben és a Calc-ban*

A függvények használatához az általunk tárgyalt táblázatkezelők jelentős segítséget nyújtanak azzal, hogy beépített Függvényvarázslóval, tündérrel rendelkeznek. Ez a hasznos segédprogram mindhárom táblázatkezelőben a szerkesztőlécen elhelyezkedő f_x ikonnal indítható legegyszerűbben. Menüből az aktiválás a következő módon történik: **Beszúrás** menü **Függvény...** pontja (Excel 2003 és Calc), illetve **Képletek/Függvénytár/Függvény beszúrása** pont (Excel 2010). Excelben – beállítástól függően – a varázsló sok esetben indítható még a **Szokásos** eszköztárról (2003-as változat), illetve a **Képletek/Függvénytár** csoportból (2010-es változat); mindkét programnál az **AutoSzum** ikonhoz tartozó lenyíló listából a **További függvények...** pontot kell választani.

Az ablak felépítése Calc-nál eltér az Excel változatokétól. Ez az eltérés a használatban is megjelenik.

Indítás után – bármely táblázatkezelő használata esetén – először ki kell választani azt a függvényt, amellyel dolgozni szeretnénk. Egymásba ágyazott függvények esetén először a legkülső függvényt.



32.2. ábra. *A Függvényvarázsló Excelben*

A függvények kategóriánként csoportosítva találhatók meg. Ezek az ablak felső vagy bal felső részében láthatók. Az első kettő a legutóbb használt függvények illetve az összes függvény csoportja, utána logikailag összetartozó függvények kisebb-nagyobb csoportjai következnek. Az ablakban látható a kiválasztott kategória összes függvénye névsor szerint. Ezek közül az egyik ki van jelölve. Erről függvényről az ablakban rövid tájékoztatást is olvashatunk.

A megfelelő függvényt Excelben egy, Calc-ban dupla kattintással választhatjuk ki. Ezután meg kell adnunk a paramétereit, ehhez Excelben kattintsunk az **OK** gombra, ami után új ablakot kapunk; Calc-ban paraméterek megadása is a Tündér első ablakában történik. Ebben a lépésben az egyes paramétereket a megfelelő mezőkbe beírhatjuk, illetve – cellák vagy blokkok esetén – rákattintva vagy a kijelölést elvégezve a program beírja a megfelelő hivatkozásokat. Ha befejeztük a függvény megadását, akkor a **Kész** illetve az **OK** gombra kattintva az aktuális cellában megjelenik a megkomponált függvény, és befejeződik a Függvényvarázsló, illetve tündér működése.

Ha varázslóval, tündérrel dolgozunk, akkor a mezők kitöltésén kívül nem kell a paraméterek elválasztásával és

a külső zárójelek megadásával foglalkozni, mert ezt a varázsló, tündér megoldja helyettünk. Ha a varázslónak, tündérnek a paraméterek megadására szolgáló mezőjében újabb függvényt akarunk megadni, akkor a mező kézi kitöltésekor a belső függvény paramétereinek elválasztására már be kell gépelni a pontosvesszőket és meg kell adni a függvény zárójeleit is.

Ha a függvénynek nincs paramétere, akkor a kiválasztás után már nincs más teendőnk, mint a **Kész**, **OK** gombra kattintani.

Az egyik legegyszerűbb függvény az összegzést végző **Szum**, amely a paraméterként megadott blokk(ok) ill. cellák számtípusú celláit összegzi. Használata közvetlenül az eszköztárról (Excel 2003 és Calcesetében) illetve a menüszalagról (Excel 2010-nél) is lehetséges.

Megjegyezzük még, hogy a Függvényvarázsló ablakának a felépítése Excel 2003-ban és 2010-ben teljesen megegyezik, kis különbséget csak a megjelenő kategóriáknál és bennük levő függvényeknél találunk.

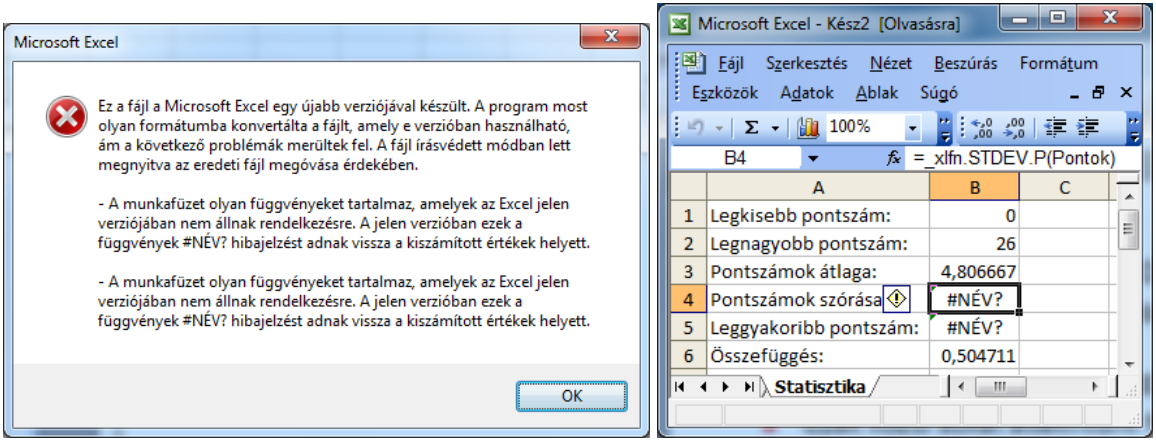
Speciálisan az Excel 2010 lehetősége még, hogy a menüszalagról (**Függvénytár** csoport) közvetlenül is hívhatunk az egyes függvénykategóriákat.

A két legfontosabb eltérés az, hogy a 2003-as Excelben még meglévő, de 2010-es verzióban már nem használatos függvények az utóbbi programban a Kompatibilitási kategóriában találhatóak; másrészt az Excel 2010-es változata sok új függvényt tartalmaz, amelyek a korábbi verziókba még nem voltak beépítve.

Azért nem törölték ki az elavultnak vélt függvényeket, hogy a 2003-as változatban készült munkafüzeteket is további átalakítás nélkül meg lehessen nyitni 2010-es programban.

A változások miatt problémák, veszteségek adódhatnak olyan esetekben, ha új Excel munkafüzetet szeretnénk menteni egy régebbi verzió formátumába, illetve hasonlóan, egy újabb verzióval készült munkafüzetnek egy régebbi verzióban történő megnyitási kísérleténél. (Az ilyen megnyitáshoz esetleg szükség lehet egy konverziós programbővítésre is.)

A továbbiakban részletesebben is megismerkedünk az összetartozó függvények csoportjaival (külön kitérve az Excel verziókra és Calc-ra). Néhány fontosabb tagot részletesen is bemutatunk, feladatok és példák megoldásával, elsősorban gyakorlati alkalmazásukra helyezve a hangsúlyt. Emlékeztetünk arra, hogy mindig



32.3. ábra. *Kompatibilitási probléma Excel 2010-ből 2003-ba való áttéréskor (Az eredeti táblázat B4 cellájában a Szórás függvényt használtuk.)*

olyan másolható képletet kell előállítani, ami átalakítás nélkül az ugyanúgy számítandó összes cellatartalom kiszámítására alkalmas.

A táblázatkezelőkbe beépített összes függvény áttekintésére itt nincs mód, de ez talán nem is szükséges, hiszen az azonos csoportba tartozó függvények között sok esetben nagy a hasonlóság. Egy-egy jellemző függvény megismerésével (a Súgó vagy a Függvényvarázsló vagy tündér segítségével) a hasonlóak kezelése is könnyen elsajátítható lesz.

32.2. Matematikai, logikai és statisztikai függvények

A matematikai függvények a következő részcsoportokba sorolhatók:

1. trigonometriai függvények és inverzeik;

2. alapvető nem trigonometriai függvények (például abszolút érték, előjel, faktoriális, négyzetgyök stb.);
3. kerekítéseket elvégző függvények;
4. összegző függvények;
5. néhány más speciális függvény.

Ezen függvények túlnyomó többségének kezelése nem jelenthet problémát, mert működésük megfelel matematikai elvárásainknak. Az utolsó csoport néhány tagja okozhat csak meglepetést.

A logikai függvényeket feltételek megfogalmazására és kiértékelésére használjuk (Nem, És, Vagy, Igaz, Hamis, Ha).

A statisztikai függvényekkel összefoglaló jelleggel foglalkozunk.

32.2.1. Véletlenszámok használata

📖 *Hetedik feladat*

Egy játékos kockadobási statisztikát készít. Határozzuk meg a dobások átlagát, döntsük el, hogy az átlag nagyobb-e π-nél, és határozzuk meg, hogy szerepelt-e egyes vagy hatos a dobások között. A dobások 1 és 6 közötti véletlen egész számok legyenek!

Megoldás Excel 2003-ban és Calcban:

Véletlenszámokat a **Vél** függvénnyel tudunk előállítani. Ennek a matematikai függvénynek nincs paramétere, tehát a név mögött kötelező megadni egy üres zárójelpárt, ez jelzi az üres paraméterlistát. Ez a függvény egy 0 és 1 közötti (0-val lehet egyenlő, 1-nél azonban kisebb) véletlen valós számot állít elő, ebből kell az 1..6 intervallumba eső véletlen egész számot készítenünk. Ehhez az eredeti véletlenszámot először megszorozzuk hattal (így olyan x számot kapunk, amelyre $0 \leq x < 6$), hozzáadunk egyet ($1 \leq x < 7$), majd az így kapott érték egészrészét vesszük például az **Int** függvénnyel. Az **Int** helyett használható más kerekítést végző függvény is, pl. **Kerek.le**. A végeredmény tehát

=Int((Vél()*6)+1).

Ezt a kifejezést írjuk be a B2-es cellába, majd az eredményt másoljuk végig az oszlopban, ameddig kell, esetünkben a B9-es cella az utolsó; lásd az ábrán lent.

Megoldás Excel 2010-ben:

Használható az előző módszer, de beépítettek egy új függvényt is, amellyel a feladat egyszerűbben megoldható. Ez a **Véletlen.között** függvény. Két paramétere van, az alsó és felső határ, amelyek között a véletlenszámokat generálja. Az egyenlőség az alsó és felső határnál egyaránt lehetséges. Így a megoldás:

=Véletlen.között(1;6).

A képletet hasonló módon másoljuk végig az oszlopban.

Véletlenszámok használata esetén vigyáznunk kell arra is, hogy alapértelmezésben a gép a kifejezéseket, így a **Vél** vagy **Véletlen.között** függvényt tartalmazókat is minden egyes ENTER leütés után újraszámolja. Ez, mint jelen esetben is, zavaró lehet. Ha akarjuk, az újraszámoltatási mód menü segítségével kikapcsolható. Ezután az F9 billentyűvel kérhetünk újraszámolást.

Az újraszámolási mód ki- ill. visszakapcsolása a táblázatkezelőkben a következő helyeken lehetséges: **Eszközök/Beállítások.../Számolás** (Excel 2003), **Eszközök/Cellatartalom** (Calc), **Képletek/Számolási beállítások** (Excel 2010). Az alapértelmezés az **Automatikus** illetve az **Automatikus számolás**, választható a **Csak kérésre** illetve **Újraszámolás** vagy **Manuális** opció.

32.2.2. Feltételek

A dobások átlagát az **Átlag** függvénnyel határozzuk meg, amelynek használata lényegében megegyezik a **Szum** függvényével.

A második kérdéshez először el kell döntenünk, hogy az átlag nagyobb volt-e π-nél, majd ettől függően szöveggel beírni a választ. Ezt a választást a táblázatkezelő a **Ha** függvénnyel valósítja meg.

A függvény általános alakja a következő:

Ha(logikai kifejezés; érték, ha a kif. igaz; érték, ha a kif. hamis).

A logikai kifejezésben az **Átlag** függvény eredményét a π-vel hasonlítjuk, amit a **Pi** paraméter nélküli függvény ad meg. A HA visszaadott értéke igaz illetve hamis logikai feltétel esetén egyaránt szöveg lesz, amit mindig idézőjelek közé kell tenni. A D2 cellába tehát a következő képlet kerül:

=Ha(B10>Pi();"igen";"nem").

32.2.3. Összetett feltételek

Az utolsó kérdésre szintén a **Ha** függvény felhasználásával válaszolunk. Az elvégzendő tevékenységek ugyanazok, mint az előző részfeladatban.

A nehezebb rész a feltétel megfogalmazása. Úgy dolgozunk, hogy meghatározzuk a legkisebb dobás értékét, és megnézzük, hogy az egyes-e, majd a legnagyobb dobás értékét a hatossal hasonlítjuk.

Ehhez a **Min** és a **Max** függvényt használjuk, amelyek a paraméterként megadott blokk(ok) illetve cellasorozat legkisebb vagy legnagyobb elemét adják vissza. Egyszerre két feltételt kell vizsgálni. Az a számunkra megfelelő, ha valamelyik feltétel IGAZ értékű. Olyan képletet kell előállítanunk, amelyik akkor és csak akkor IGAZ, ha a két feltétel valamelyike IGAZ. Ezt a logikai vagy műveletet alkalmazva készíthetjük el. Így:

Min(B2:B9)=1 $\vee$ **Max(B2:B9)=6**.

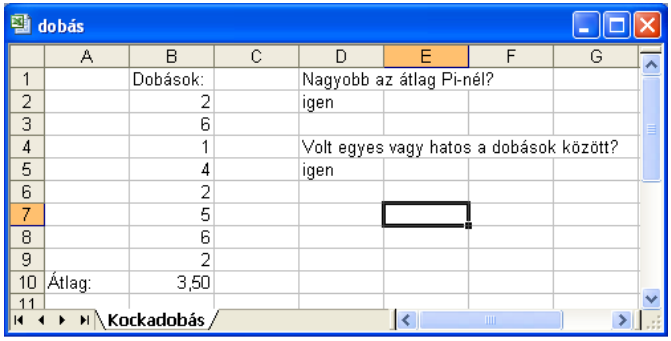
Az Excel a vagy műveletet a **Vagy** függvény használatával valósítja meg, ennek általános alakja:

=Vagy(logikai kif1; logikai kif2; ... logikai kifN).

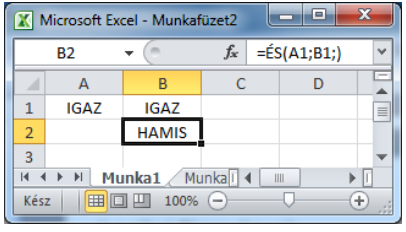
A feltételeket tehát a paraméterlistában felsorolva kell megadni. Az eredmény IGAZ lesz, ha valamelyik paraméter-kifejezés értéke IGAZ (logikai és művelethez az **És** függvényt ugyanígy használjuk, ilyenkor az eredmény csak abban az esetben IGAZ, ha az összes paraméter-kifejezés értéke IGAZ).

Az utolsó kérdésre választ adó képlet tehát:

=Ha(Vagy(Min(B2:B9)=1;Max(B2:B9)=6);"igen";"nem").



32.4. ábra. *Kockadobási statisztika*



32.5. ábra. *Logikai függvény hibás használata*

Az És függvény használatánál mindig HAMIS lesz a függvényérték, ha egy „üres paramétert” (több ;, mint amennyi a paraméterek elválasztásához kellene) írunk be az argumentumok közé. Ez a paraméter nyilván HAMIS értékű, így az eredmény is HAMIS lesz (a Vagy függvény paramétereinél elkövetett ilyen elírás nem hiba, mivel a vagy művelet eredménye nem függ a HAMIS értékű operandustól!).

Aktivitás: Járjunk utána, hogy miért nem!

Nyolcadik feladat (önálló gyakorlásra)

Két játékos, Adél és Béla kockát dobálva azon versenyez, hogy 10 dobásból ki ér el több találatot. Készítsünk egy másolható képletet a kockadobások szimulálására, majd függvény segítségével válaszoljunk arra a kérdésre, hogy ki nyert. A képlet jelezze azt is, ha döntetlen az állás.

Segítség: A döntetlen állás vizsgálatát úgy tudjuk megoldani, hogy egy Ha függvénybe egy másik Ha függvényt ágyazunk.

32.2.4. A matematikai és logikai függvénykategóriák további elemezése

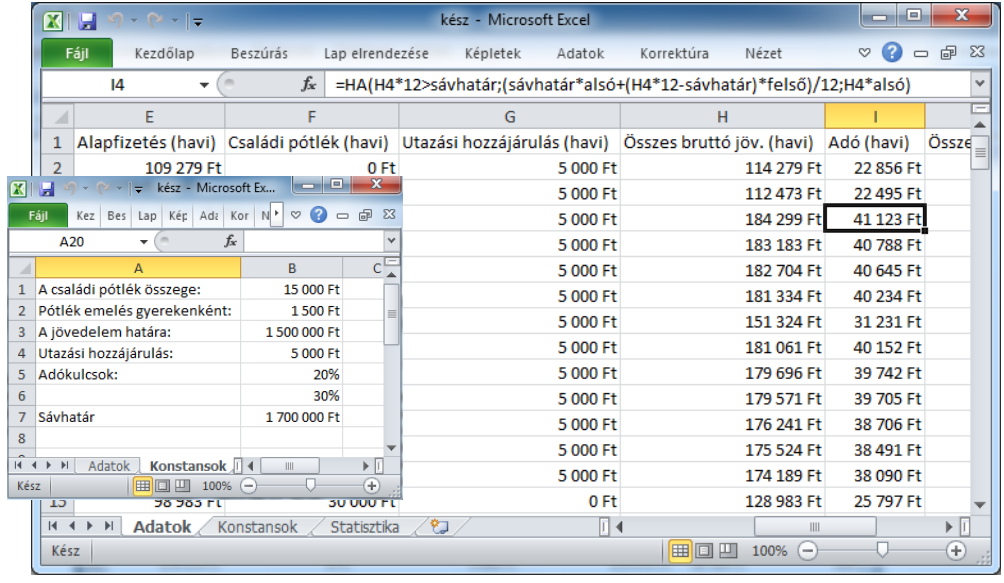
A logikai függvényeket, ezen belül is a Ha függvényt nagyon gyakran használjuk a táblázatkezelőkkel való problémamegoldás során, az egyszerű „igen-nem” típusú válaszokon át egészen a bonyolult feltételek megfogalmazásáig. Sok esetben az is előfordul, hogy a megoldás viszonylag egyszerű elemekből építkezik ugyan, de összességében mégsem igazán könnyű felépíteni.

Kilencedik feladat (önálló gyakorlásra)

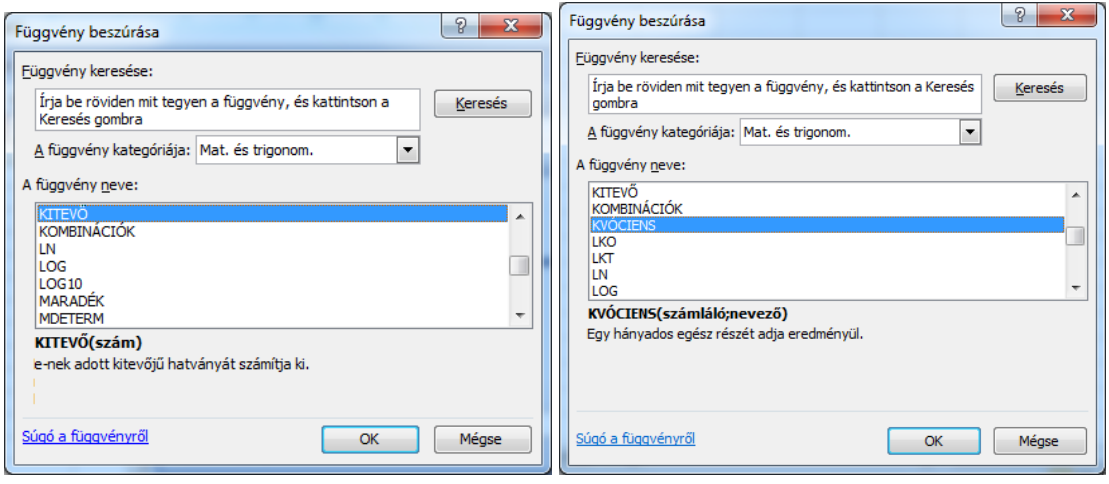
Adjunk megoldást a két kulcsot alkalmazó egyszerű adószámítási feladatra a következők szerint: Rögzített egy (éves) jövedelemhatár, amely felett a magasabb kulccsal, alatta pedig az alacsonyabb kulccsal adóznak a dolgozók. A kulcsok szintén adottak. Meghatározandó az egyes jövedelmek után fizetendő adó (havi szinten).

Az Excel 2003-as változatához képest a Calc és a 2010-es Excel több új függvényt is tartalmaz ezekben a kategóriákban.

Aktivitás: Járjunk utána, hogy milyen új függvények vannak az Excel 2010-es verziójában és a Calc táblázatkezelőben az Excel 2003-as verziójához képest. Milyen különbséget tapasztalhatunk a csoportbeosztásban? Szorítkozzunk csak a matematikai függvényekre. (Lásd 32.7. ábra.)



32.6. ábra. *Adó meghatározása kétkulcsos rendszer szerint*



32.7. ábra. *A matematikai és trigonometriai kategória bővülése (Excel 2003 és 2010)*

Önellenőrzés

1. Jelöljük meg a következő kifejezések közül azokat, amelyek használhatók névként a táblázatkezelőben:

- ☐ A1
- ☐ alma
- ☐ körte
- ☐ név
- ☐ A1:B1
- ☐ _C
- ☐ a_b
- ☐ 2A

2. Jelöljük meg a következő kifejezések közül azokat, amelyek értéke egyenlő az =A1+B1 kifejezéssel:

- ☐ =SZUM(A1:B1)
- ☐ =SZUM(A1;B1)
- ☐ =SZUM(B1:A1)
- ☐ =SZUM(A1;A1:B1;B1)
- ☐ =SZUM(A1)+B1

3. Adjuk meg a táblázatkezelő válaszát a következő képlet esetén:
=HA(ABS(PI())>4;IGAZ;3)

Válasz: ☐

4. Jelöljük meg a következő kifejezések közül azokat, amelyek alkalmasak véletlenszámok előállítására:

- ☐ =VÉL()
- ☐ =VÉL(3)
- ☐ =VÉL(3;4)
- ☐ =VÉL*()
- ☐ =VÉLETLEN.KÖZÖTT(0;2)
- ☐ =VÉLETLEN.KÖZÖTT(2*10;5)

5. Az A2 cellában az =A1+B1 képlet szerepel. Adjuk meg, hogy mi lesz az A2 cella tartalma, ha

- ... a B oszlop elé beszúrunk egy új oszlopot! ☐
- ... az első sor elé beszúrunk egy sort! ☐
- ... töröljük a munkalapról B oszlopot! ☐
- ... töröljük a B1-es cella tartalmát! ☐

6. Az A2 cellában az =$A1+B1 képlet szerepel, az A1 cella tartalma 3, a B1 celláé 2, a B2 celláé 1, az A3 celláé pedig 0. Adjuk meg, hogy mi lesz az A2 cella értéke, ha

- ... a B oszlop elé beszúrunk egy új oszlopot! ☐
- ... az első sor elé beszúrunk egy sort! ☐
- ... töröljük a munkalapról B oszlopot! ☐
- ... töröljük a munkalapról az A oszlopot! ☐
- ... töröljük a munkalapról az első sort! ☐
- ... töröljük a B1-es cella tartalmát! ☐

14. LECKE

Függvények

A lecke a függvények használatának második, nagyobb részét tekinti át (a keresőfüggvényekig). Hangsúlyosan felhívjuk hallgatóink figyelmét arra, hogy ez a rész erősen gyakorlatias! Az anyag elsajátítása itt – a téma rendkívül szerteágazó volta miatt – csak akkor tekinthető sikeresnek, ha a jegyzetben bemutatott és az órákon szerepelt gyakorlati feladatokat is bizonyos jártassággal meg tudja oldani a tanuló. Ennek megfelelően – a korábbiakhoz hasonlóan – a lecke önálló feldolgozására fordítandó idő erősen függ az előképzettségtől, diákonként akár többszörös szorzók is adódhatnak.

A statisztikai függvények áttekintése

Gyakran előfordul, hogy mérési adatokat szeretnénk elemezni a táblázatkezelő segítségével. Ilyen elemzésekben leggyakrabban a következő kérdések fordulnak elő: mi az adatok átlaga, mekkora az adatok szórása, mekkora a középső, mekkora a k-adik legkisebb és legnagyobb elem, hány adat kerül egy intervallumrendszer egyes intervallumaiba, melyik adat fordul elő leggyakrabban, stb. Ezekre a kérdésekre megfelelő beépített függvények segítségével kaphatunk választ.

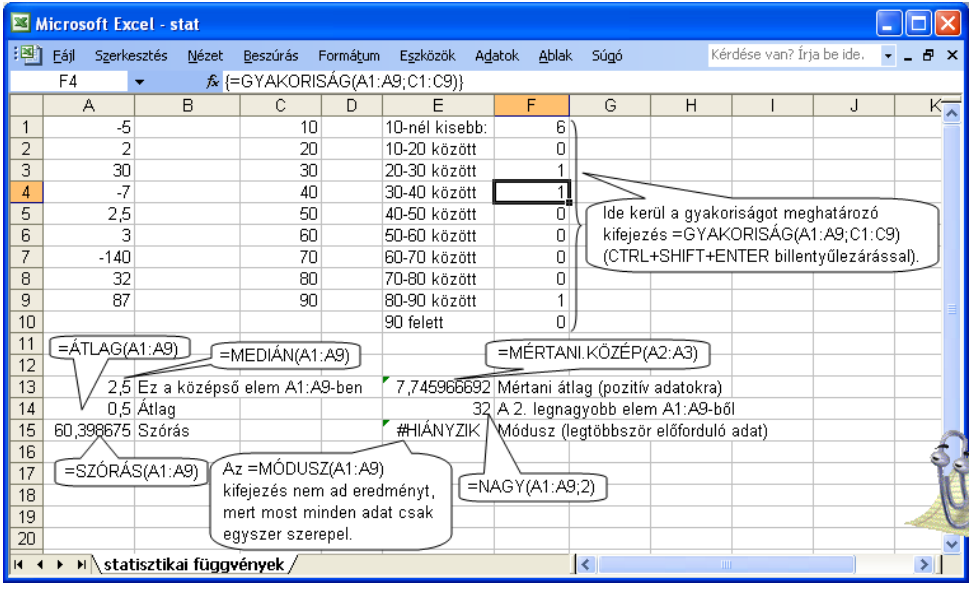
1. Átlagszámítás: **Átlag(átlagolandó adatsor)**. A függvény értéke az argumentumban megadott számok átlaga lesz.
2. Mértani átlag: **Mértani.Közép(adatok)**. Az adatok mértani közepe lesz a függvényérték (csak pozitív adatokra működik!).
3. Középső elem keresése: **Medián(számok)**. A függvény értéke az argumentumban specifikált számok közül a középső szám, ha a számok páratlan sokan vannak. Párosan sok elem esetén bontsuk kétfelé a számhalmazt. Mindkét félbe ugyanannyi szám kerüljön. Az egyik fél a kisebb a másik fél a nagyobb számokat tartalmazza. A Medián a kisebb számok legnagyobbikának és a nagyobb számok legkisebbikének átlaga lesz.
4. K-adik legkisebb elem keresése: **Kicsi(Adatsor;k)**. A függvényérték az adatsor elemeinek növekvően rendezett sorából a k-adik elem lesz.

5. K-adik legnagyobb elem keresése: **Nagy(adatsor;k)**. A függvény értéke az adatsor csökkenően rendezett sorából a k-adik érték.
6. Adatrendszer szórása: **Szórás(adatok)**. Az adatok szórásértékét adja. (A kiszámító algoritmus közelítő módszerrel számol.)
7. Legtöbbször előforduló adat meghatározása: **Módusz(adatok)**. A függvényérték az adatok között legtöbbször előforduló érték lesz. Ha minden elem csak egyszer szerepel, akkor válaszként a #HIÁNYZIK vagy az #ÉRTÉK üzenetet kapjuk.

A **Gyakoriság(adatsor;intervallumok)** függvénnyel a megadott adatsor elemeinek előfordulási gyakoriságát határozhatjuk meg. Ez egy tömbfüggvény (bevitelét a CTRL+SHIFT+ENTER-rel jelezzük), eggyel több értéket ad válaszul, mint ahány intervallumot a második paraméterével megadtunk. A tömböt oszlopba rendezve adja meg. Az intervallumok jobbról zárt, egymáshoz illeszkedő, egymást követő intervallumok lesznek, amiket a második paraméterben megadott végpontok definiálnak. Az első intervallum bal végpontja a mínusz végtelen, az utolsó intervallum jobb végpontja a plusz végtelen. Ezeket nem kell megadni.

Aktivitás: A 32.8. ábra alapján készítsük el a táblázatot a függvények megismerésének elmélyítésére!

A statisztikai függvényeknél is tapasztalhatunk változásokat az Excel 2010-es verziójában a korábbi állapothoz képest. Pl. a Szórás függvényt meghagyták a régebbi programokkal való kompatibilitás miatt, az új Szór.s függvény a szórás pontos megállapítására szolgáló algoritmussal számol. A Módusz függvényt is felerősítették, felkészítve arra, hogy több leggyakoribb elem is lehet (Módusz.egy, Módusz.több).



32.8. ábra. *Statisztikai függvények*

32.3. Szöveg-, idő- és dátumkezelő függvények

A szövegkezelő függvények a táblázatkezelőkben a következő részcsoportokba oszthatók:

1. a szöveg valamely részét kivágó függvények (**Bal**, **Jobb**, **Közép**),
2. konverziós műveleteket végrehajtó függvények (szöveg és más típusú adatok között),
3. szövegdarabok helyettesítését, cseréjét vagy összeillesztését elvégző függvények.

Az idő- és dátumkezelő függvények ezen adatok részeinek kivágását és konverzióját hajtják végre (pl. **Perc** – a 2010-es Excelben **Percek –**, **Hét.napja**), illetve egy-egy speciális függvény a mai dátum és a mostani idő előállítására alkalmas (a **Ma** és a **Most**). Ezeknél függvényeknél figyelni kell arra, hogy a táblázatkezelők a dátumot és az időt számként tárolják. Itt csak a dátumkezelőkkel foglalkozunk részletesebben, de ezzel lényegében az időkezelők használatát is megismerjük.

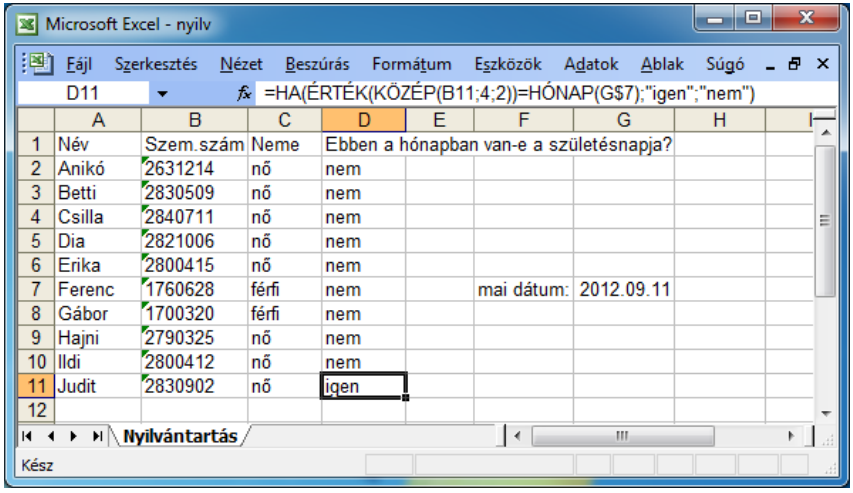
Tizedik feladat

Egy vállalat nyilvántartást készít dolgozóiról a személyi számuk első hét jegye alapján. Ezek felhasználásával válaszoljuk meg, hogy a dolgozó férfi vagy nő, és ebben a hónapban van-e a születésnapja (ezt mindig a mai dátumra vonatkoztatjuk).

A személyi szám első hét jegyét szöveges adatként tároljuk.

A dolgozó nemét a személyi szám első jegyének vizsgálatával állapíthatjuk meg. Ehhez a **Bal** függvényt használjuk. Ennek a függvénynek az értéke az első paraméterben megadott sztring első karakterétől számított, a második paraméterben megadott hosszúságú része lesz. (a **Jobb** függvény egy karakterlánc végéből ugyanígy állítja elő a függvényértéket). A válasz megfogalmazásához a Ha függvényt használjuk. A C2 cellára a megoldás tehát (egy leegyszerűsített helyzetet feltételezve):

=Ha(Bal(B2)="1";"férfi";"nő").



	A	B	C	D	E	F	G	H
1	Név	Szem.szám	Neme	Ebben a hónapban van-e a születésnapja?				
2	Anikó	2631214	nő	nem				
3	Betti	2830509	nő	nem				
4	Csilla	2840711	nő	nem				
5	Dia	2821006	nő	nem				
6	Erika	2800415	nő	nem				
7	Ferenc	1760628	férfi	nem		mai dátum:	2012.09.11	
8	Gábor	1700320	férfi	nem				
9	Hajni	2790325	nő	nem				
10	Ildi	2800412	nő	nem				
11	Judit	2830902	nő	igen				
12								

32.9. ábra. *Vállalati nyilvántartás*

A második kérdéshez a születési hónapokra van szükség. A személyi számból a negyedik, ötödik jel jelenti ezt. amit összehasonlítunk a mai dátumból kapott hónappal. A különböző típusú adatok kezelése miatt konverzióra is szükség lesz.

A mai dátumot a **Ma** függvény állítja elő. Értékét a számítógép belső órája határozza meg. Ebből a **Hónap** függvénnyel kapjuk meg az aktuális hónapot (a hasonló **Év** és **Nap** függvény értelemszerűen egy dátum év és nap részének a meghatározására szolgál). A **Ma** függvényt beleírhatjuk direkt módon a képletbe, vagy egy tetszőleges cellába írhatjuk a táblázaton kívül, és erre ezután hivatkozhatunk.

A személyi szám hónap részét a **Közép** függvénnyel vágjuk ki, amelynek három paramétere a következő: az adott szöveg, hányadik jeltől kezdődjön a kimásolandó rész és ez hány jelből álljon. A kivágott hónaprészt az **Érték** függvénnyel számmá alakítjuk, és ezt hasonlítjuk a mai dátumból már ismert módon meghatározott hónappal.

Végül egy **Ha** függvénybe beépítjük a megfelelő függvényeket, és a képletet lefelé másolhatóvá tesszük:

=Ha(Érték(Közép(B2;4;2))=Hónap(C$10);"igen";"nem").

A 2000 után született fiatalok személyi száma már 3-mal, ill. 4-gyel kezdődik. Ha képletünket fel akarjuk készíteni arra, hogy pár év múlva ilyen fiatal dolgozókat is alkalmaz majd a cég, akkor a **Maradék** függvényt is be kell építeni a kifejezésbe (a kettővel való osztási maradékot határozzuk meg). Noha a **Bal** függvény karakteres visszaadott értéke miatt elméletileg konverzióra is szükség lenne, a gyakorlatban ez elhagyható, a táblázatkezelő ezt az átalakítást automatikusan elvégzi. Így a módosított megoldás a C2 cellára a következő:

=Ha(Maradék(Bal(B2);2)=1;"férfi";"nő").

A 2010-es Excelben a **Párose** információs függvény felhasználásával egy megoldás is léehetséges (**=Ha(Párose(Bal(B2));"nő";"férfi")**).

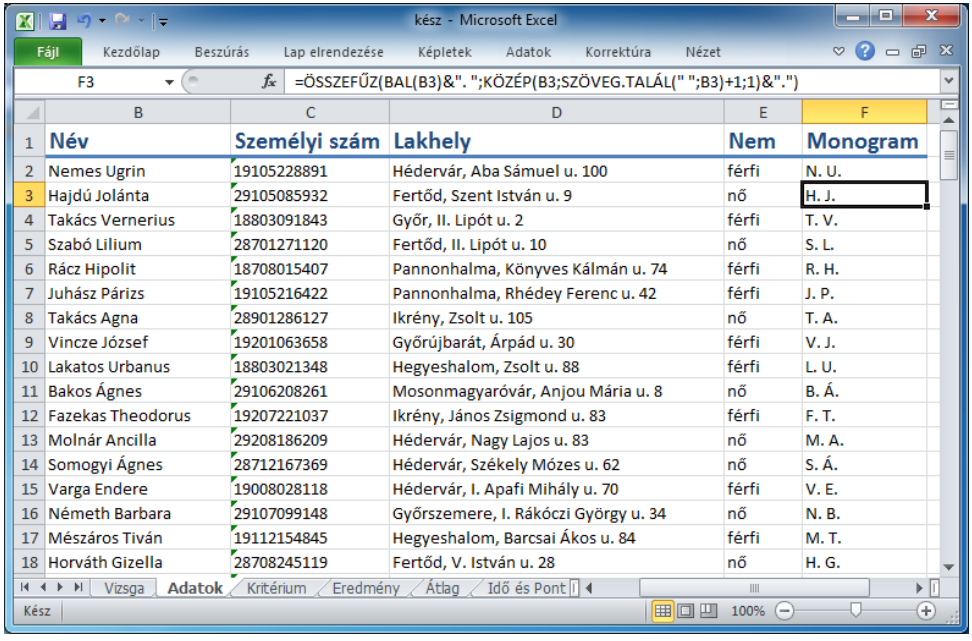
A feladatban a személyi szám jegyeit szöveges adatként tároltuk. Mivel szám típusú adatok szövegként való tárolása egyes esetekben logikai hiba (most természetesen nem), ezért a táblázatkezelő (csak az Excel verziók) – beállítástól függően – figyelmeztet bennünket (jelzés: kis zöld háromszögek a cellák bal felső sarkában). Ez a jelzés kikapcsolható (Excel 2003: **Eszközök/Beállítások/Hibaellenőrzés** ablak, **Szövegként tárolt szám** mező; Excel 2010: **Képletek** menüszalag, **Képletvizsgálat** csoport, **Hibaellenőrzés**).

Tizenegyedik feladat

Határozzuk meghallgatók neve alapján a monogramjukat. Az egyszerűség kedvéért feltesszük, hogy a vezetéknév és a keresztnév nem kezdődhet dupla betűvel, és nincs kettős keresztnév sem.

A feladatot több lépésben oldjuk meg (32.10):

1. A vezetéknévből kimásoljuk a kezdőbetűt.
2. Megkeressük a névben a(z első) szóközt, és a következő részből is leválasztjuk a kezdőbetűt.
3. Összefűzzük a kezdőbetűket, a pontokat és a szóközt.



	B	C	D	E	F
1	Név	Személyi szám	Lakhely	Nem	Monogram
2	Nemes Ugrin	19105228891	Hédervár, Aba Sámuel u. 100	férfi	N. U.
3	Hajdú Jolánta	29105085932	Fertőd, Szent István u. 9	nő	H. J.
4	Takács Vernerius	18803091843	Győr, II. Lipót u. 2	férfi	T. V.
5	Szabó Lilium	28701271120	Fertőd, II. Lipót u. 10	nő	S. L.
6	Rácz Hipolit	18708015407	Pannonhalma, Könyves Kálmán u. 74	férfi	R. H.
7	Juhász Párizs	19105216422	Pannonhalma, Rhédey Ferenc u. 42	férfi	J. P.
8	Takács Agna	28901286127	Ikrény, Zsolt u. 105	nő	T. A.
9	Vincze József	19201063658	Győrújbarát, Árpád u. 30	férfi	V. J.
10	Lakatos Urbanus	18803021348	Hegyeshalom, Zsolt u. 88	férfi	L. U.
11	Bakos Ágnes	29106208261	Mosonmagyaróvár, Anjou Mária u. 8	nő	B. Á.
12	Fazekas Theodorus	19207221037	Ikrény, János Zsigmond u. 83	férfi	F. T.
13	Molnár Ancilla	29208186209	Hédervár, Nagy Lajos u. 83	nő	M. A.
14	Somogyi Ágnes	28712167369	Hédervár, Székely Mózes u. 62	nő	S. Á.
15	Varga Endere	19008028118	Hédervár, I. Apafi Mihály u. 70	férfi	V. E.
16	Németh Barbara	29107099148	Győrszemere, I. Rákóczi György u. 34	nő	N. B.
17	Mészáros Tiván	19112154845	Hegyeshalom, Barcsai Ákos u. 84	férfi	M. T.
18	Horváth Gizella	28708245119	Fertőd, V. István u. 28	nő	H. G.

32.10. ábra. *Monogramok meghatározása*

A vezetéknév kezdőbetűjének leválasztására most is a **Bal** függvényt használjuk. A szóközt a vezeték- és a keresztnév között a **Szöveg.talál** függvénnyel keressük meg, amelynek két kötelező paramétere a következő: a keresett szövegrész, és az a szöveg, ahol keresünk. A megtalált szóköz utáni karaktert a **Közép** függvénnyel másoljuk ki, végül a kezdőbetűket, a pontokat és a szóközt az **Összefűz** függvénnyel állítjuk össze egy karaktersorozattá.

=Összefűz(Bal(B2);".";Közép(B2;Szöveg.talál(" ";B2)+1;1);".").

Tizenkettedik feladat (önálló gyakorlásra)

Bővítsük az előző megoldást oly módon, hogy a vezetéknévben és a keresztnévben megengedett a dupla kezdőbetű, és előfordulhat kettős keresztnév is.

Segítségként megadjuk a megoldás javasolt lépéseit:

1. A vezetéknévből leválasztjuk a kezdőbetűt.
2. Megkeressük a névben az első szóközt.
3. Előállítjuk a név maradékát (ez már csak a keresztnév).
4. Az első lépéssel azonos módon a név maradékából leválasztjuk a kezdőbetűt.
5. Megnézzük, hogy a név maradékában van-e még szóköz.
6. Ha igen, akkor leválasztjuk a név utolsó részét, és meghatározzuk a kezdőbetűt.
7. Összefűzzük a kezdőbetűket.

Az útmutató alapján készült megoldást a 32.11. ábra mutatja.

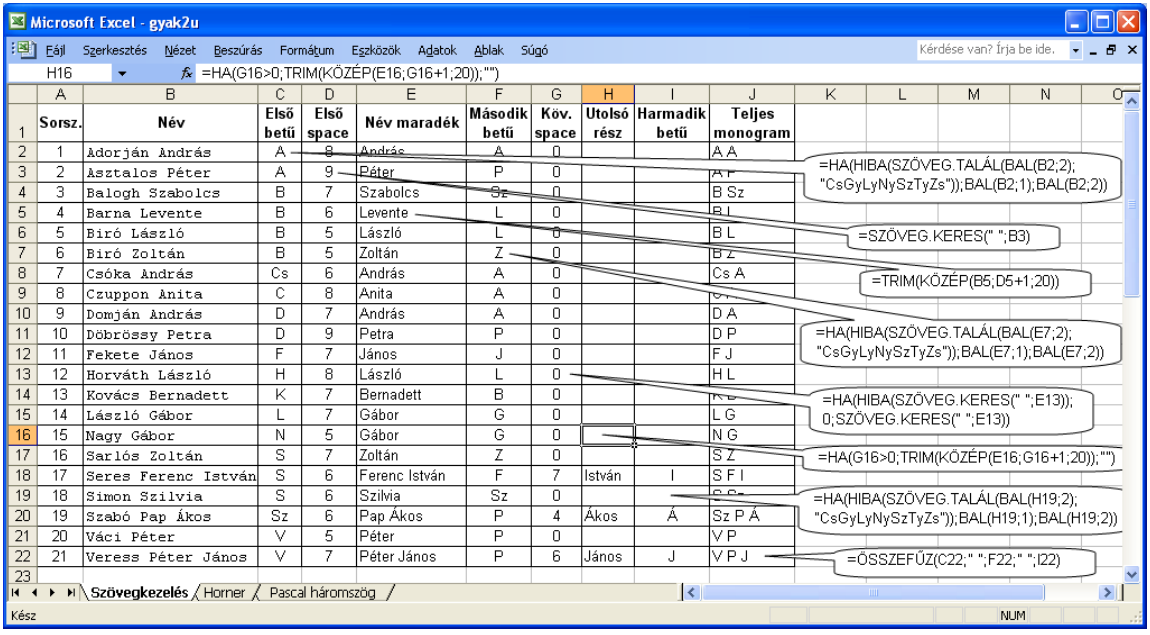
A kezdőbetűk vizsgálatánál az algoritmus néha tévedhet, ha a kettősnek tűnő betű valójában mégsem kettős (pl. Lyka László). Három tagnál hosszabb nevekre az algoritmus már nem használható jól.

Tizenharmadik feladat (részben önálló gyakorlásra)

Páciensek születési adatai (személyi számmal adottak) és a mai dátum felhasználásával állapítsuk meg a betegek korát. A vizsgálatnál figyeljünk arra, hogy a páciensnek volt-e már az adott évben születésnapja!

Segítségként megadjuk a javasolt felépítés egyes lépéseit:

1. A mai dátum év értékéből kivonjuk a születési dátumból kapott évtized és év értéket úgy, hogy ez utóbbiakat hozzáfűzzük a "19" szövegkonstanshoz. Ezzel megkapjuk a páciens korát úgy, hogy még nem

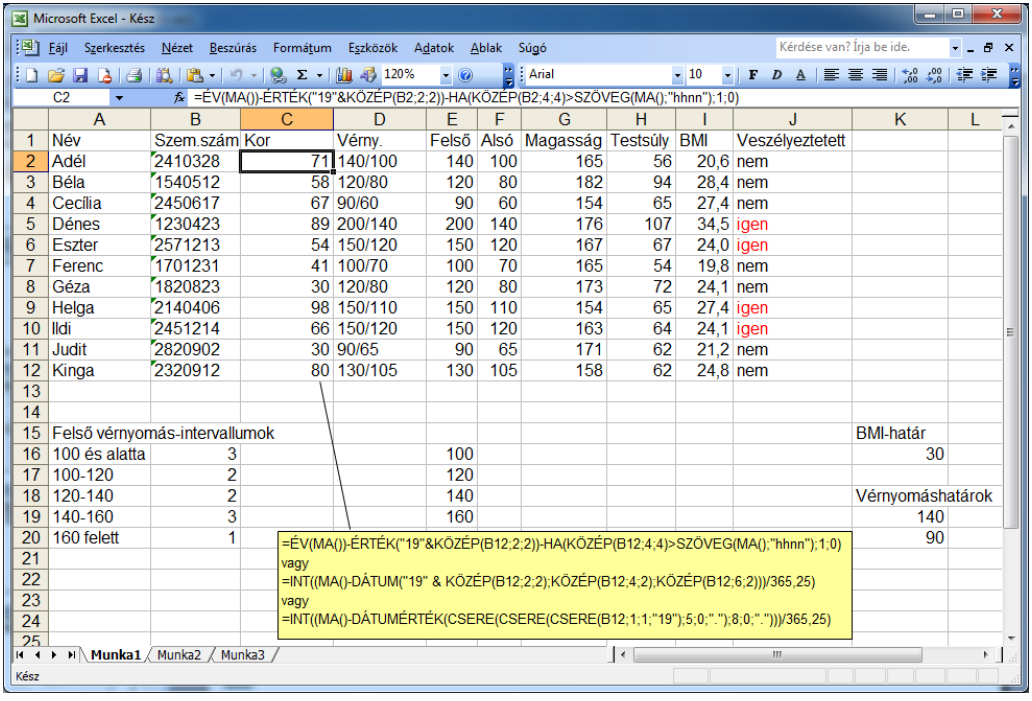


32.11. ábra. *Monogramok előállítása 2.*

vizsgáltuk azt a lehetőséget, hogy az idén volt-e születésnapja (esetleg: nem töltötte be még az idén az adott évet).

2. A mostani dátum hónap és nap részét ("hhnn" egyéni formázással) szöveggé alakítjuk, és összehasonlítjuk a születési dátum hónap és nap részével. Ha az illetőnek az idén még nem volt születésnapja, akkor levonunk a korból még egyet.

A megoldáshoz az **Év**, **Ma**, **Érték**, **Közép**, **Ha** és **Szöveg** függvényeket használtuk fel.



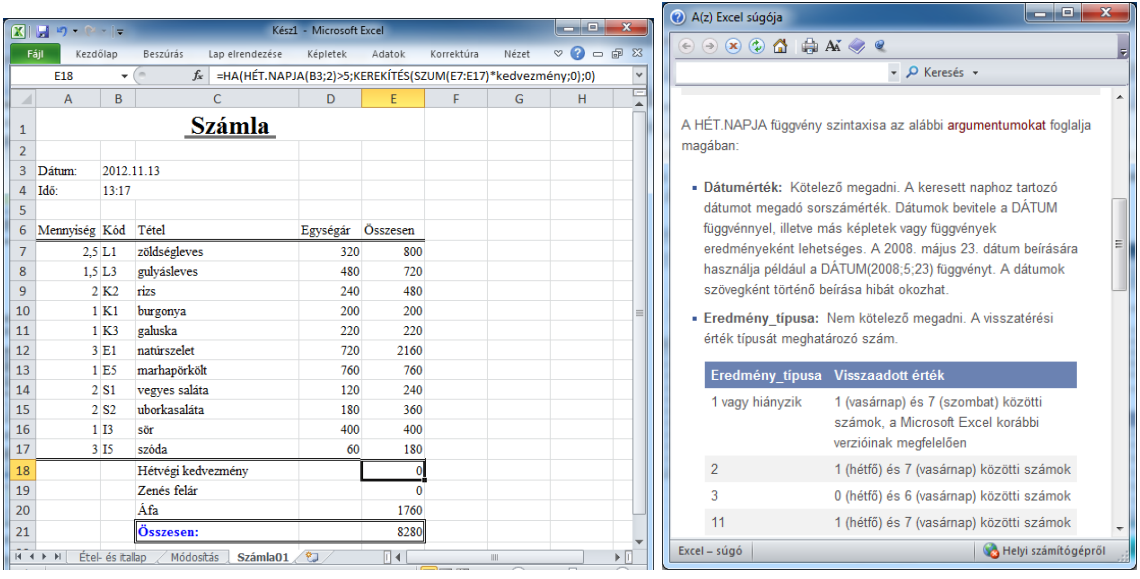
	A	B	C	D	E	F	G	H	I	J	K	L
1	Név	Szem.szám	Kor	Vérny.	Felső	Alsó	Magasság	Testsúly	BMI	Veszélyeztetett		
2	Adél	2410328	71	140/100	140	100	165	56	20,6	nem		
3	Béla	1540512	58	120/80	120	80	182	94	28,4	nem		
4	Cecília	2450617	67	90/60	90	60	154	65	27,4	nem		
5	Dénes	1230423	89	200/140	200	140	176	107	34,5	igen		
6	Eszter	2571213	54	150/120	150	120	167	67	24,0	igen		
7	Ferenc	1701231	41	100/70	100	70	165	54	19,8	nem		
8	Géza	1820823	30	120/80	120	80	173	72	24,1	nem		
9	Helga	2140406	98	150/110	150	110	154	65	27,4	igen		
10	Ildi	2451214	66	150/120	150	120	163	64	24,1	igen		
11	Judit	2820902	30	90/65	90	65	171	62	21,2	nem		
12	Kinga	2320912	80	130/105	130	105	158	62	24,8	nem		
13												
14												
15	Felső vérnyomás-intervallumok										BMI-határ	
16	100 és alatta	3			100						30	
17	100-120	2			120							
18	120-140	2			140						Vérnyomáshatárok	
19	140-160	3			160						140	
20	160 felett	1									90	
21												
22												
23												
24												
25												

```
=ÉV(MA())-ÉRTÉK("19"&KÖZÉP(B12;2;2))-HA(KÖZÉP(B12;4;4)>SZÖVEG(MA();"hhnn");1;0)
vagy
=INT((MA()-DÁTUM("19" & KÖZÉP(B12;2;2);KÖZÉP(B12;4;2);KÖZÉP(B12;6;2)))/365,25)
vagy
=INT((MA()-DÁTUMÉRTÉK(CSERE(CSERE(CSERE(B12;1;1;"19");5;0;".");8;0;".")))/365,25)
```

32.12. ábra. *Életkor meghatározása születési dátum alapján*

Oldjuk meg a feladatot más módon is (lásd: 32.12. ábra)!

A következő példában egy speciális dátumkezelő függvény érdekes alkalmazását mutatjuk be. A feladat: számlakészítés egy család éttermi fogyasztásáról. Ennek része, hogy a számla összegét módosítani kell aszerint, hogy hétvégi napra esett-e az éttermi látogatás (hétvégi kedvezmény figyelembe vétele). A **Hét.napja** függvény



32.13. ábra. *A Hét.napja függvény alkalmazása feladatmegoldáshoz*

alkalmas ilyen típusú vizsgálatra; a kötelező argumentum egy dátum, a visszaadott érték egy sorszám, 1-től 7-ig. Az alapértelmezés az, hogy a függvény a hét napjait vasárnaptól sorszámozza, lehetőség van azonban arra, hogy ezt módosítsuk (amennyiben a második, opcionális paraméter értéke 2, akkor a sorszámozás hétfőtől indul). (32.13. ábra)

32.4. Egyéb fontos függvények

Egyes speciális esetekben különleges adatkezelésre van szükség.

Előfordulhat például, hogy valamely cella tartalma üres, és külön vizsgálat nélkül ez hibához vezet. Ilyen

esetekben használjuk az **Üres** függvényt. Ez megvizsgálja, hogy a paraméteréül megadott cella tartalmaz-e adatot, és ettől függően IGAZ vagy HAMIS logikai értéket ad vissza.

A **Hiba** vagy **Hibás** függvény megvizsgálja, hogy a hivatkozott kifejezés/paraméterül megadott érték hibaüzenetet állít-e elő. A válasz logikai IGAZ vagy HAMIS. A 2010-es Excelben a Hiba függvényt a **Hiba.e** váltotta fel, a működés azonban megegyezik.

Szám, **Szöveg.e**, **Logikai**: a függvények megvizsgálják, hogy a hivatkozott kifejezés/paraméterül megadott érték a megfelelő típusba tartozik-e.

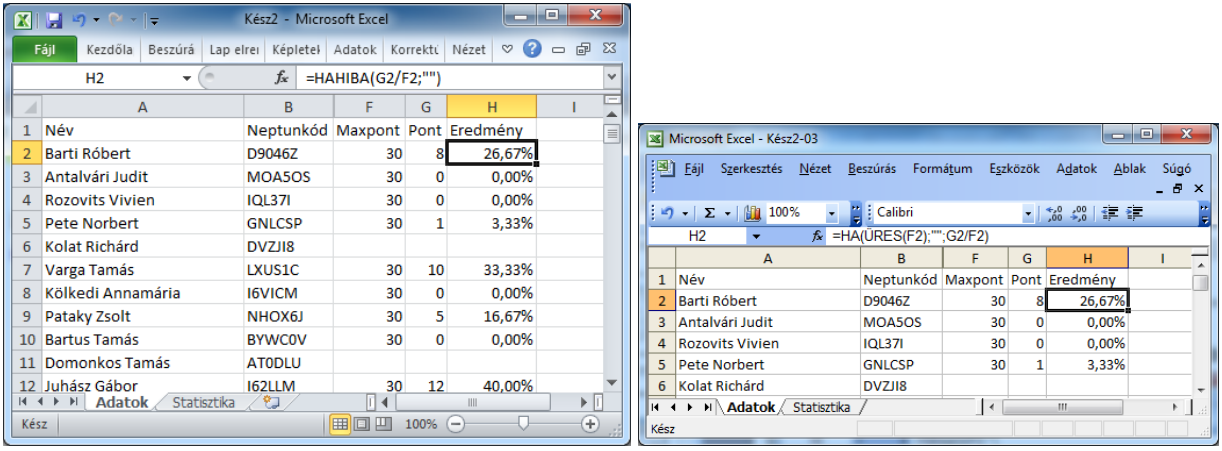
Ezeknek a függvényeknek a segítségével is kimutatható, hogy a Calc (az Exceltől eltérően) nem használ logikai típust. A numerikus típusból formátummegadással lehet logikai értéket megjeleníteni. Ha Calc-ban a **Szám** függvénynek egy logikai értékkel megjelenített cellát jelölünk meg paraméterként, a függvény értéke IGAZ lesz. A **Logikai** függvény csak akkor ad IGAZ értéket, ha a paramétere olyan cella, aminek értéke numerikus, a kijelzés formátuma pedig logikai. Ha valamelyik kifejezésben logikai értéket kell használni, akkor a logikai érték helyett numerikus értéket kell írni. A 0 a HAMIS, az összes többi szám az IGAZ értéknek felel meg.

A fenti függvények mind az információs függvénykategóriába tartoznak.

Mint ismert, a műveletek elvégzése csak azokkal a típusokkal történhet, amelyekre az adott műveletek értelmezve vannak (például összeszorozni csak számokat lehet). Ezért típus-átalakításra, konverzióra van szükség akkor, ha az adat formája alapján egy művelet elvégezhető lenne, de a típus ezt a műveletelvégzést nem teszi lehetővé. A konverziók egy része automatikus vagy figyelmeztető jelzéssel „javasolt” (például, ha egy számformájú adatot szöveg típusúként adunk meg, akkor az Excel figyelmeztet, és átalakítást javasol), más része függvénnyel, függvényekkel végezhető el. Ilyen konverziós függvények például a **Szöveg**, **Érték**, **Római**, **Dátumérték**, stb. Ezeket a függvényeket több különböző kategóriába sorolják a táblázatkezelők.

Itt is külön kitérünk arra, hogy a 2010-es Excel (és a Calc is) több új függvényt is tartalmaz a 2003-as változathoz képest. Érdekes példa a **Hahiba** függvény, amely lehetőséget biztosít alternatív válasz megadására abban az esetben, ha a beírt képlet hibás (a függvény a 2010-es Excel logikai kategóriájában szerepel).

Említést érdemel még a 2010-es Excel új Tervezés kategóriája, ami a Calc-ban is megtalálható nagyon hasonló repertoárral Kiegészítő kategórianévvel. Itt sok egyéb mellett konverziós függvényeket találhatunk, amelyek



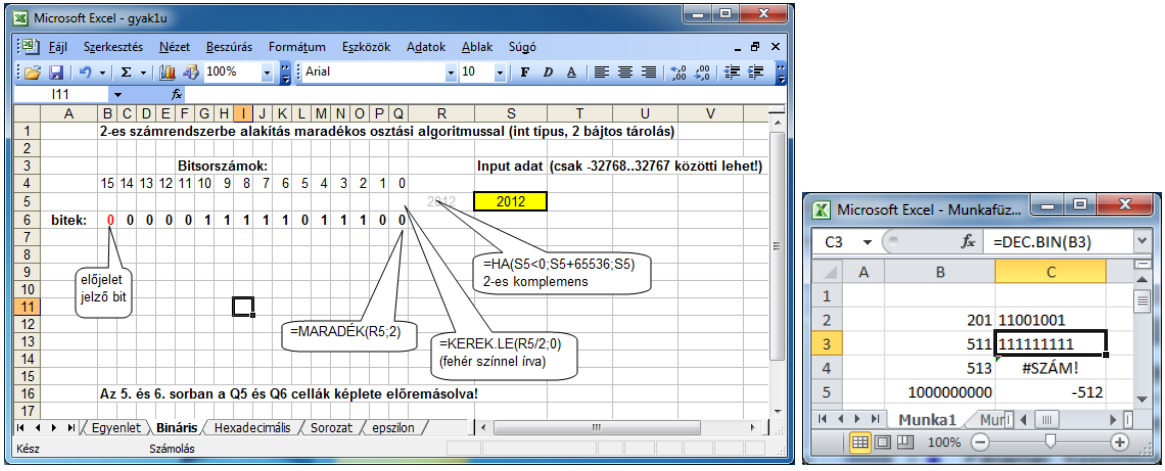
32.14. ábra. *A **Hahiba** függvény kiváltása az **Üres** függvénnyel a 2003-as Excelben*

segítségével a tízes, kettes, nyolcas és tizenhatos számrendszer között végezhetünk átalakításokat. A **Dec.bin** függvény (Calc: **Dec2bin**) például tízes-kettes átalakítást biztosít, hátránya azonban, hogy csak a [–512, 511] intervallumba eső számokat tudja átalakítani. Párja a **Bin.dec** függvény (Calc: **Bin2dec**), hasonlóan működik: a negatív számok az [1000000000, 1111111111], a nemnegatívak a [0, 0111111111] intervallum számaiból képződnek. Ez az átalakítás a komplemens kód szabályai szerint történik.

A 2010-es Excel és a Calc tartalmaz függvénypárost a tizenhatos–tízes átalakításra is.

A problémára természetesen adható általánosabb megoldás is a táblázatkezelőkben (a 32.15. ábra bal oldalán).

A megvalósításhoz beírjuk az átváltani kívánt számot egy adott cellába (az ábra szerint: S5). Mellette megvizsgáljuk, hogy a szám negatív-e, ez utóbbi esetben 65536-ot hozzáadunk (nemnegatív esetben a túlcsordulás miatt a hozzáadás felesleges, de nem okozna hibát). Egy segédsorban felvesszük a bitsorszámokat (4. sor).



32.15. ábra. *Decimális-bináris átalakítás*

A szám átváltását maradékos osztással végezzük. Kiszámítjuk a hányadost és a maradékot (5. és 6. sor). Végeredményben a maradékok sorozata a szám fixpontos alakját adja. A legmagasabb helyi értékű bit az előjelet mutatja.

Más fixpontos számábrázolási módszerek is ismeretesek, és ilyen tárolási forma nem csak egész számokra lehetséges.

Önellenőrzés

1. Hova igazítja a táblázatkezelő alapértelmezés szerint az alábbi függvény eredményét (Excel 2010)? (b = balra, j = jobbra, k = középre, n = a megadottak alapján nem lehet eldönteni)

   =Ha(Percek(Most())>-1;7;A2)

   Válasz: ☐

2. Adjuk meg azt a legegyszerűbb képletet, amely a B2-es cellába írt személyi számból (szöveges adat) lecsípi a legelső karaktert! A képlet a cellára vegyes hivatkozással hivatkozzon, oszloprögzítéssel.

   Vá: =Bal($B2)

3. Melyek igazak a következő állítások közül az =Most()-Ma() képletre?

   ☐ Értéke mindig egész.
   ☐ Értékét a Kerek.le függvény biztosan nem változtatja meg.
   ☐ Értéke nagyobb a pi számnál.
   ☐ Értéke 0 és 1 közé esik (0-nál nagyobb egyenlő, 1-nél kisebb).
   ☐ Értéke dátumformában nem jeleníthető meg.
   ☐ Értékét célszerű időformátumban megjeleníteni.

4. Csoportosítsuk a következő függvényeket!/Jelöljük meg, hogy a következő függvények beletartoznak-e az alábbi kategóriákba! Egy függvény több csoportba is tartozhat, de az is lehet, hogy egyikbe sem.

Párose

- ☐ Második paramétere elhagyható
- ☐ Értéke mindig egész szám
- ☐ Első paramétere logikai típusú
- ☐ Egyik kategória sem

Szum

- ☐ Második paramétere elhagyható
- ☐ Értéke mindig egész szám
- ☐ Első paramétere logikai típusú
- ☐ Egyik kategória sem

Abs

- ☐ Második paramétere elhagyható
- ☐ Értéke mindig egész szám
- ☐ Első paramétere logikai típusú
- ☐ Egyik kategória sem

Int

- ☐ Második paramétere elhagyható
- ☐ Értéke mindig egész szám
- ☐ Első paramétere logikai típusú
- ☐ Egyik kategória sem

Ha

- [ ] Második paramétere elhagyható
- [ ] Értéke mindig egész szám
- [ ] Első paramétere logikai típusú
- [ ] Egyik kategória sem

Vagy

- [ ] Második paramétere elhagyható
- [ ] Értéke mindig egész szám
- [ ] Első paramétere logikai típusú
- [ ] Egyik kategória sem

Hahiba

- [ ] Második paramétere elhagyható
- [ ] Értéke mindig egész szám
- [ ] Első paramétere logikai típusú
- [ ] Egyik kategória sem

Percek

- [ ] Második paramétere elhagyható
- [ ] Értéke mindig egész szám
- [ ] Első paramétere logikai típusú
- [ ] Egyik kategória sem

Hét.napja

- [ ] Második paramétere elhagyható
- [ ] Értéke mindig egész szám
- [ ] Első paramétere logikai típusú
- [ ] Egyik kategória sem

5. Jelöljük meg, hogy a következő függvények beletartoznak-e az alábbi kategóriákba! (Az is lehet, hogy egy függvény egyik kategóriába sem sorolható be.)

Szum

- [ ] Matematikai és trigonometriai
- [ ] Szöveg
- [ ] Idő- és dátumkezelő
- [ ] Logikai

Abs

- [ ] Matematikai és trigonometriai
- [ ] Szöveg
- [ ] Idő- és dátumkezelő
- [ ] Logikai

Pi

- [ ] Matematikai és trigonometriai
- [ ] Szöveg
- [ ] Idő- és dátumkezelő
- [ ] Logikai

Int

- [ ] Matematikai és trigonometriai
- [ ] Szöveg
- [ ] Idő- és dátumkezelő
- [ ] Logikai

Ha

- [ ] Matematikai és trigonometriai
- [ ] Szöveg
- [ ] Idő- és dátumkezelő
- [ ] Logikai

Vagy

- [ ] Matematikai és trigonometriai
- [ ] Szöveg
- [ ] Idő- és dátumkezelő
- [ ] Logikai

Szöveg.talál

- [ ] Matematikai és trigonometriai
- [ ] Szöveg
- [ ] Idő- és dátumkezelő
- [ ] Logikai

Most

- [ ] Matematikai és trigonometriai
- [ ] Szöveg
- [ ] Idő- és dátumkezelő
- [ ] Logikai

Percek

- [ ] Matematikai és trigonometriai
- [ ] Szöveg
- [ ] Idő- és dátumkezelő
- [ ] Logikai

Hét.napja

- [ ] Matematikai és trigonometriai
- [ ] Szöveg
- [ ] Idő- és dátumkezelő
- [ ] Logikai

15. LECKE

Keresőfüggvények

A lecke elsősorban a keresőfüggvények használatát tárgyalja, de ide csatoltuk a több munkalap használatát tárgyaló rövidebb fejezetet is. Az anyag elsajátítása – hasonlóan a korábbiakhoz – itt is csak akkor tekinthető sikeresnek, ha a jegyzetben bemutatott és az órákon szerepelt gyakorlati feladatokat is bizonyos jártassággal meg tudja oldani a tanuló.

32.5. Keresőfüggvények

A keresőfüggvényeket olyan típusú feladatok megoldására használjuk, amikor ismerjük egy logikailag összetartozó adatsor egy elemét, és ehhez keresünk valamely másik elemet ugyanabból az adatsorból. A feladat megfogalmazása táblázatos formában: keresünk egy adattáblázatban, a sorok első celláiban egy ismert értéket, és ha megtaláltuk, szükségünk lenne a sor további mezőinek tartalmára is. Ugyanez a feladat oszlopokra is előfordulhat.

A keresőfüggvényeket a **Mátrixfüggvények** csoportban találjuk meg.

📖 *Tizennegyedik feladat*

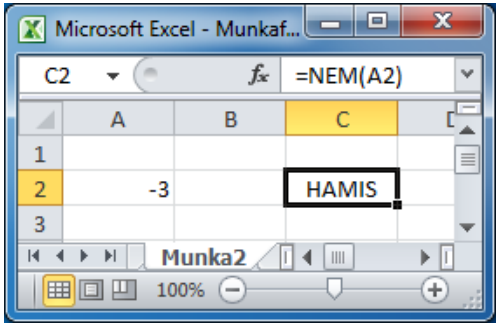
Egy cég dolgozóinak nevét és fizetését tárolja egy táblázatban. Meg kell határozni egy adott nevű dolgozó fizetését (feltehetjük, hogy csak egy ilyen nevű dolgozó van).

Táblázat első sorában való keresésre a **VKeres** illetve első oszlopában történő keresésre az **FKeres** függvény használható.

Az **Fkeres** függvény általános alakja a következő (a **Vkeres** teljesen hasonló):

FKeres(mit keresünk; hol; oszlopszám; [tartományban keres])

A függvények első paramétere a keresett érték (ez általában egy cella értéke, de lehet akár egy kifejezés értéke is). Ezt keresünk a második paraméterként megadott tábla, az ún. keresési tábla első oszlopában vagy sorában. Vigyázzunk arra, hogy a második paraméterrel definiált tartomány ne tartalmazzon fejlécet, mert ez hibát okozhat. A harmadik paraméter egy sorszám, amely megadja azt, hogy keresett értéket tartalmazó oszlop vagy sor hányadik sorából illetve oszlopából kell kivenni a visszaadott értéket. A negyedik paraméter egy logikai érték, ezt azonban nem kell minden esetben megadni.



32.16. ábra. *Szemléltetés: a 0-tól különböző számértékeket a táblázatkezelő logikai igaznak tekinti*

Ha harmadik paraméterként olyan sor-, illetve oszlopsorszámot adunk meg, ami kívül esik a második paraméterben definiált területen akkor a #HIV! hibaüzenetet kapjuk.

A keresőfüggvényeket kétféle módon használhatjuk: az egyik lehetőség a tartományban keresés, a másik a nem tartományban történő, pontos keresés. Ezt a választást a negyedik paraméter értéke szabályozza.

Ezen paraméter értéke alapértelmezésben 1 vagy IGAZ, ami azt jelenti, hogy ha nem adjuk meg, akkor ez az érték lesz a negyedik paraméter. Ha ez nem megfelelő, akkor a 0 vagy HAMIS értéket kell megadni.

A táblázatkezelő a 0 számértéket HAMIS logikai értéknek tekinti, az összes többi szám IGAZ logikai értéket jelent.

Ha ez a paraméter nincs megadva, vagy meg van adva, és értéke igaz, akkor a keresés típusa tartományban keresés lesz. Ami azt jelenti, hogy a keresendő értéket a függvény a keresési táblázat első oszlopában megadott értékekkel határolt, balról zárt intervallumokban keresi. Egy-egy intervallum határait a táblázat egymást követő sorainak első oszlopban lévő értéke adja, kivéve az utolsó intervallumot, amelynek jobboldali határa az adott adattípus lehető legnagyobb értéke lesz. Ennek következményeképpen az intervallumoknak nem lehet közös része, és nem lehet közöttük kimaradó rész sem. Természetesen ezt csak úgy tudjuk elérni, ha a táblázat az első oszlop szerint növekvően rendezett. Ha a keresett értéket nem tartalmazza egyetlen – az első oszlop által

meghatározott – intervallum se, akkor a #HIÁNYZIK hibaüzenetet kapjuk. Ugyancsak ezt az üzenetet kapjuk gyakran (de nem minden esetben) akkor is, ha az első oszlop adatai nem rendezettek, azaz nem adnak megfelelő intervallumokat!

Ha diszkrét értékeket keresünk, akkor ezeket az értékeket intervallum-határolóként is megadhatjuk, és pontos egyezésre keresünk. Ez a keresőfüggvények használatának logikailag legegyszerűbb esete (mint később látni fogjuk, ez a megközelítés „bolondbiztosság tekintetében” támadható).

Ha a negyedik értéke hamis, akkor a keresés pontosan az első oszlop értékei között történik. Rendezettség nem szükséges. Ha a keresett érték nincs az első oszlopban, akkor a #HIÁNYZIK hibaüzenetet kapjuk.

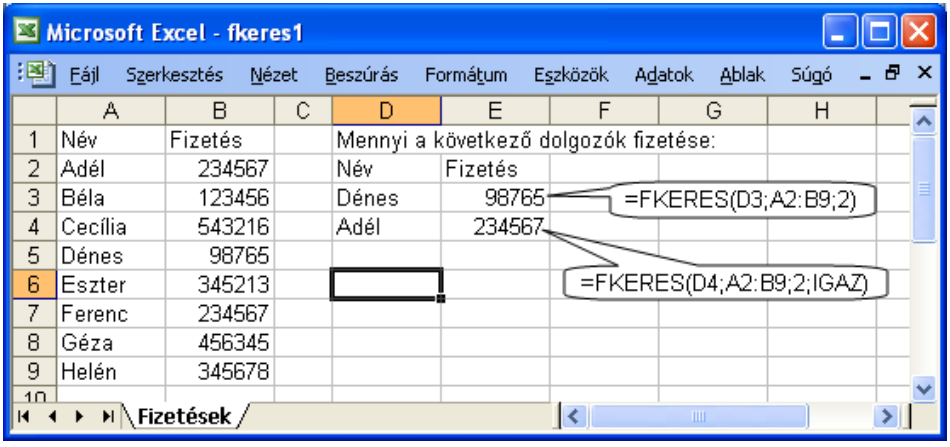
32.5.1. Példák

1. A legegyszerűbb esetben az adatok a keresési tábla első oszlopában, sorában névsorba rendezettek, és a keresett érték pontosan megtalálható közöttük. Ha a tizenharmadik feladatban a dolgozók neveit névsorba rendezve adták meg, akkor ezt a keresési módszert használjuk.

2. A tizennegyedik feladatban azt vizsgáljuk, hogy egy érték melyik megadott intervallumba esik. Ez a tartományban keresés tipikus esete. Ilyenkor a keresési tábla első oszlopában, sorában az adatok rendezettek, és pontos egyezést általában nem fogunk találni.

A keresés folyamatát ekkor logikailag úgy képzelhetjük el, hogy ha a függvény a keresett értéknél nagyobbat talál (magyar táblázatkezelőnél magyar ábécé szerinti rendezést feltételezve – szöveges adatok esetén), akkor a keresés megszakad, és a függvény az utolsó még nem nagyobb értékhez tartozó, a harmadik paraméter által meghatározott adatot adja vissza. Az utolsó intervallum esetén az adott adattípushoz tartozó lehető legnagyobb értékét „képzelhetjük” az első oszlop után következő első üres cellába. A korrekt működéshez tehát az intervallumok alsó határát kell megadni.

3. Ha a tizenharmadik feladatban az adatok a keresési tábla első oszlopában nincsenek névsorba rendezve, akkor nem szabad a tartományban keresés módszerét alkalmazni, mivel az eredmény ezzel a módszerrel hibás lehet.



32.17. ábra. *Fizetések meghatározása rendezett táblázatból*

A tizenharmadik feladat megoldása névsorba rendezett táblázat esetében:

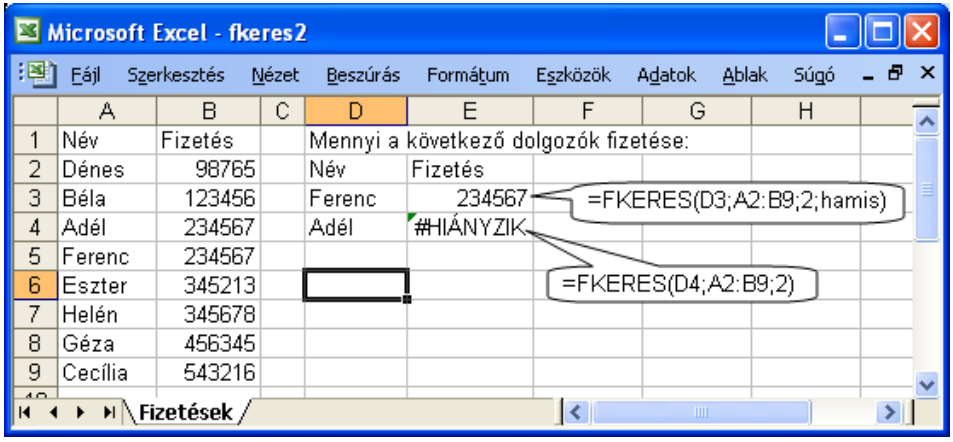
=FKeres(D3;A2:B9;2).

Ha nem névsorban követik egymást az A oszlopban a nevek, akkor át is rendezhetjük a keresési tábla adatait (és ekkor a logikailag legegyszerűbb esethez jutunk), ha erre jogunk van. Jegyezzük meg, ha a keresett név nem szerepelne a névsorban, szintén hibás eredményt kapnánk.

De névsorba való rendezés nélkül is megoldható a feladat a következő képlettel:

=FKeres(D3;A2:B9;2;hamis).

Mi történik akkor, ha a keresendő érték megadása hibás? Ha a „tartományban keresés” típussal dolgozunk, akkor a függvény úgy dolgozik, mintha rendezett lenne az első oszlop szerint a táblázat. Eredmény is születhet, ami persze nem mindig helyes, de a felhasználó ezt nem feltétlenül veszi észre –hibaüzenetet csak akkor kapunk, ha a keresendő érték az első névnél is kisebb. Ha a „nem tartományban történő keresés”-t alkalmazzuk, akkor



32.18. ábra. *Fizetések meghatározása nem rendezett táblázat esetén. Adél fizetésének megkeresésére adott függvény hibásan van paraméterezve.*

viszont minden hibásan megadott paraméterérték hibaüzenethez vezet. Ez az utóbbi megközelítés a korrekt.

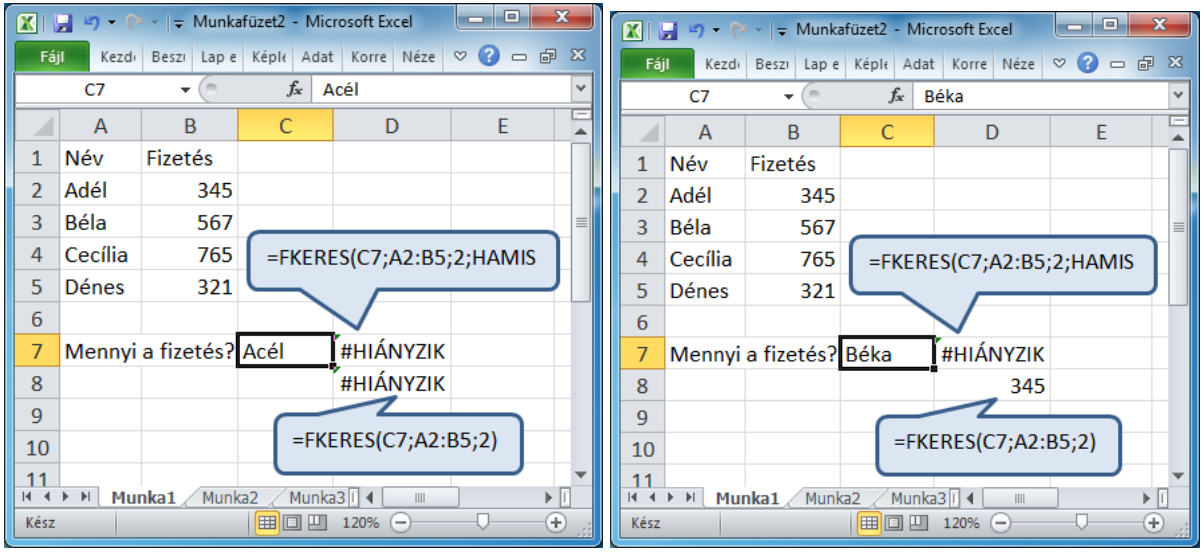
Tizenötödik feladat

Egy dolgozat értékelésekor adottak a jegyekhez tartozó sávhatárok. Határozzuk meg a hallgatók jegyeit a százalékban megadott teljesítmény alapján.

Ez a feladat példa a fenti keresési problémák kezelésére numerikus környezetben is.

A megoldásához a hagyományosan megadott intervallumokat át kell írni oly módon, hogy az alsó sávhatár legyen a sáv azonosítója. Így garantálható az, hogy az alsó és felső határok közé eső értéket az Excel meg fogja találni a megfelelő helyen.

Ha a keresési érték nagyobb, mint tábla legnagyobb értéke (például ha 95-öt keresünk), akkor a legutolsó sorból veszi az adatot (itt a 85 sorából, tehát ötöst ad eredményül).

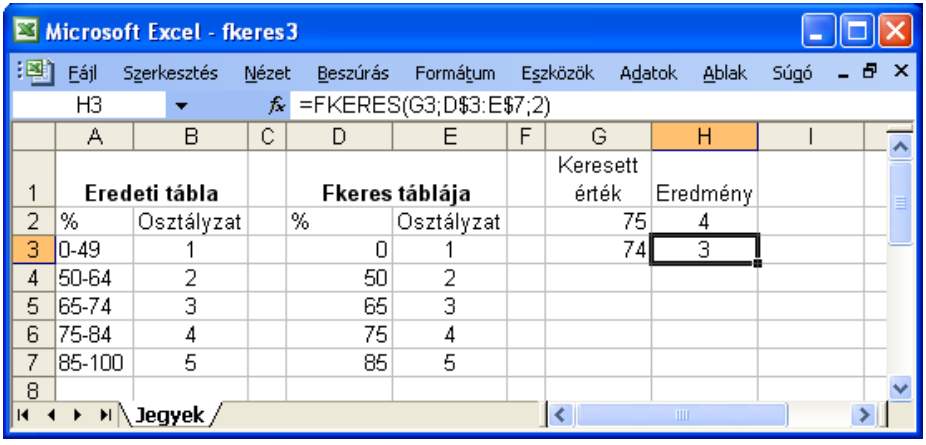


32.19. ábra. *Rendezett táblázatban különböző keresési típusok és válaszok felhasználói hiba esetén*

Visszautalva a korábbiakra, blokkhivatkozást tartalmazó kifejezés másolásakor érdemes a hivatkozott blokkot elnevezni, mert így nem kell rögzítéseket alkalmazni.

Vannak olyan feladatok is, ahol alaphelyzet szerint a keresőtábla valamely magasabb sorszámú oszlopában kell keresni, mint ahonnan az eredményt várnák. Ilyenkor az eredmény oszlopának értékeit a keresőtáblába közvetlenül az utolsó oszlop mögé, megfelelő hivatkozással elhelyezzük, és a keresőtáblát úgy adjuk meg, hogy a hivatkozásban az legyen az első oszlop, ahol keresni akarunk. Ezután a feladat a szokásos módon megoldható. A segédoszlopot, sort célszerű elrejteni.

Ilyen típusú feladatok keresőfüggvények nélkül is megoldhatók. A használható eszközök különböző indexelő és helymeghatározó függvények (például: **Hol.van**, **Index**, **Oszlop**, **Sor**).



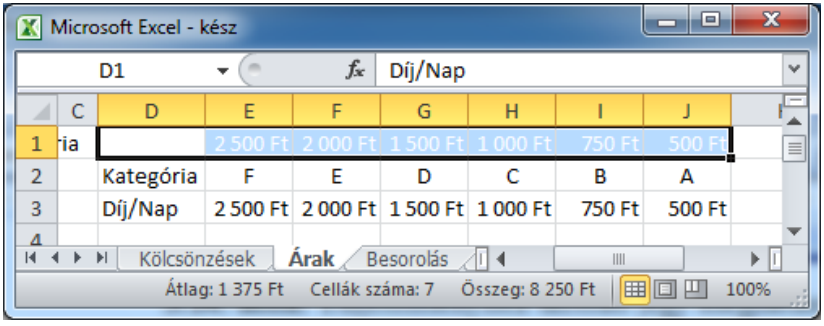
32.20. ábra. *Teljesítményhez tartozó jegy meghatározása*

Aktivitás: Próbáljuk megoldani a tizennegyedik feladatot keresőfüggvények nélkül!

Tizenhatodik feladat (önálló gyakorlásra)

Egy klasszikus többkulcsos (3 vagy több kulcs) adórendszert áttanulmányozva és elemezve készítsünk táblázatot a következő adatokkal: sávhatár, adó a sávhatárig, a sávhatár feletti rész adója százalékosan (éves adatok). Az adó a sávhatárig oszlop elemeit számítással határozzuk meg a százalék és a sávhatár oszlop alapján. Vegyünk fel néhány dolgozót havi fizetés adatokkal, és határozzuk meg a havi szinten általuk megfizetendő adót az előző táblázat szerint.

Segítség: a megoldáshoz 3 darab keresőfüggvényt kell használni.



32.21. ábra. *Segédtáblázat végére másolt első sor Vkeres függvény alkalmazásához*

33. Műveletek munkalapokkal

33.1. Több munkalap használata, kapcsolt táblázatok

Egy munkafüzet általában több munkalapból áll, ezek füleit a képernyő bal alsó részén címkékkel jelezve láthatjuk. A munkalapok közötti váltás a címkékre kattintással, illetve a CTRL+PGUP illetve CTRL+PGDN billentyűkkel hajtható végre. A nem látható füleket a fülcsoport mellett elhelyezkedő nyilakkal jeleníthetjük meg.

A munkalapokkal a következő műveletek végezhetők el: átnevezés, törlés, beszúrás, másolás és áthelyezés. Ezeket a funkciókat legegyszerűbben a helyi menüből választhatjuk ki, amelyet úgy aktivizálhatunk, hogy a megfelelő munkalap címkéjére az egér jobb gombjával rákattintunk, de a táblázatkezelők menüjéből is elérhetők (33.1. ábra).

A gyakorlatban sokszor előfordul, hogy valamely munkalapon elhelyezkedő táblázatra vagy adatra egy másik munkalapról kell hivatkozni. Egy-egy bonyolultabb probléma megoldása ugyanis sok – különböző jellegű

adatokat tartalmazó – táblázatot igényelhet, és ezeket nem mindig célszerű – vagy nem lehet –, egy munkaterületen elhelyezni.

Egy másik munkalap cellájára vagy blokkjára úgy tudunk hivatkozni, hogy először megadjuk a munkalap nevét, majd egy **!** illetve **.** jel (Excel illetve Calc) után következhet a cellahivatkozás. Ha például a hetedik feladatban a dolgozók nevét és fizetését tartalmazó táblázat egy másik – Adat nevű – munkalapon helyezkedik el, akkor Excelben és Calc-ban így módosul a hivatkozás:

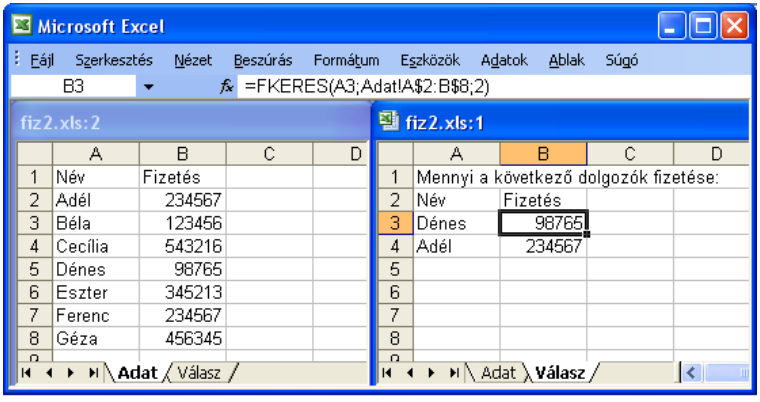
=FKeres(A3;Adat!A$2:B$8;2) illetve **=FKeres(A3;Adat.A$2:B$8;2)**.

Amennyiben elnevezett blokkokat használunk, és csak egy ilyen nevű blokk van a munkafüzeten, akkor a program megtalálja a hivatkozást úgy is, ha másik munkalapra vonatkozik.

Egyes feladatoknál arra is szükség lehet, hogy különböző munkafüzetek között teremtsünk kapcsolatot. Ilyenkor Excelben a fájl nevét szögletes zárójelben adjuk meg a hivatkozásban, például: **[A.xls]Munka1!A2**. Calc-nál a fájlnevet aposztrófok közé írjuk, és elválasztójelként a füzet és a lap neve közé #-et rakunk, például: **'A.ods'#Munkalap1.A1**.

Program	Menüpontok (munkalapok kezelése menüből)
Excel 2003	Formátum/Lap/Átnevezés…, Szerkesztés/Lap törlése, Beszúrás/Munkalap, Szerkesztés/Lap áthelyezése vagy másolása…
Excel 2010	Kezdőlap/Cellák csoporton belül Formátum/Munkalap átnevezése, Törlés/Munkalap törlése, Beszúrás/Munkalap beszúrása, Formátum/Lap áthelyezése vagy másolása…
Calc	Formátum/Munkalap/Átnevezés…, Szerkesztés/Munkalap/Törlés…, Beszúrás/Munkalap…, Szerkesztés/Munkalap/Áthelyezés/másolás…

33.1. ábra. *Munkalapok kezelésére használható menüpontok*



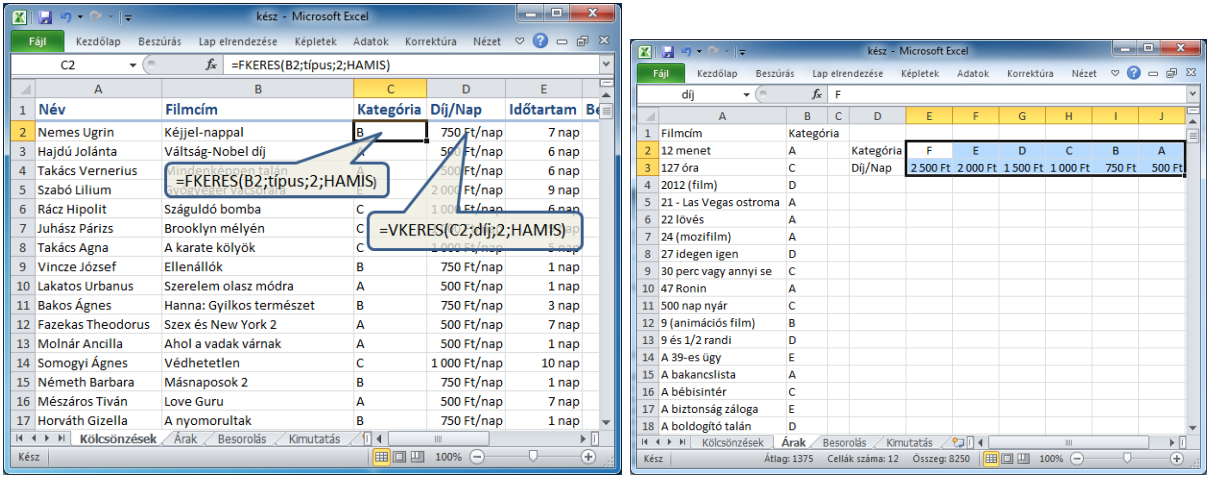
33.2. ábra. *Hivatkozás másik munkalap celláira*

33.2. Láthatóság és védelem

A táblázatkezelő ablakában több munkafüzet megjelenítésére is lehetőség van. Azt, hogy hogyan mutassa a munkafüzeteket szabályozható (menüvel, gyorsbillentyűvel stb.). A beállítási változatokból a 33.4 összefoglalt menüpontokkal lehet választani.

Lehetőségünk van arra is, hogy a táblázat egyes részeit – például a feliratokat – a képernyőn rögzítsük. Így nagy táblázat esetén ezek több képernyőoldalt ellapozva is láthatóak maradnak Rögzítést kérve az aktuális cella feletti sorok és a balra levő oszlopok a munkalapon való mozgáskor mindig láthatóak maradnak a képernyőn – már ha elférnek. A rögzített terület határát vékony fekete vonalak jelzik. A rögzítés feloldható. Ezekkel kapcsolatos menüből elérhető lehetőségeket az egyes programokban a 33.5. ábrán ismertetjük.

Nagy táblázat különböző részeit egyszerre áttekinthetjük a táblázat felosztásával (**Ablak/Felosztás** ill.



33.3. ábra. *Hivatkozás másik munkalap celláira elnevezett blokkokkal*

Program	Lehetőségek
Excel 2003	**Ablak/Elrendezés**
	Ablak/Összehasonlítás egymás mellett a következővel: …
Excel 2010	**Nézet/Ablak/Mozaik**
	Nézet/Ablak/Párhuzamos megjelenítés
Calc	(Az új dokumentum új ablakban jelenik meg.)

33.4. ábra. *Ablakok együttes megjelenítésének lehetőségei táblázatkezelőkben*

Nézet/Ablak/Felosztás). Használata hasonló a rögzítéshez.

Időnként szükség lehet arra, hogy a táblázat egyes részeit illetéktelenek ne tudják módosítani. Ez a védelem beállítható a cellákra, a munkalapra és a teljes munkafüzetre is. A beállítás részleteit a 33.6. ábra mutatja.

Program	Táblarögzítés	Rögzítés feloldása
Excel 2003	**Ablak/Ablaktábla rögzítése**	**Ablak/Ablaktábla feloldása**
Excel 2010	**Nézet/Ablak/Panelek rögzítése/Ablaktábla rögzítése**	**Nézet/Ablak/Panelek rögzítése/Ablaktábla feloldása**
Calc	**Ablak/Rögzítés** (kapcsoló be)	**Ablak/Rögzítés** (kapcsoló ki)

33.5. ábra. *Ablaktábla rögzítésének és feloldásának lehetőségei*

Munkalapvédelem esetén megadhatjuk, hogy milyen műveletek engedélyezettek a védett területen (pl. kijelölés, formázás, törlés, szűrés). A védelem feloldását célszerű jelszóhoz kötni.

Program	Cellákra	Munkalapra/munkafüzetre
Excel 2003	**Formátum/Cellák…**	**Eszközök/Védelem**
Excel 2010	**Kezdőlap/Cellák/Formátum/Védelem**	**Kezdőlap/Cellák/Formátum/Védelem**
Calc	**Formátum/Cellák…**	**Eszközök/Dokumentum**

33.6. ábra. *Védelem beállításának lehetőségei*

Önellenőrzés

1. Mi lesz a keresőfüggvény értéke az alábbi táblázat esetén, ha a képletet az A7 cellába írjuk? (Próbáljuk fejben megadni a választ, úgy, hogy nem használjuk a táblázatkezelőt.)
Képlet: =FKERES("Szilva";$A$1:F6;2;HAMIS)

	A	B	C	D	E	F
1	Alma	1	Gyümölcs	30	40	50
2	Ananász	2	Befőtt	25	400	40
3	Barack	3	Lekvár	20	4000	30
4	Körte	4	Üdítő	15	40000	20
5	Meggy	5	Szörp	10	400000	10
6	Szilva	6	Pálinka	5	4000000	0

Válasz: ☐

2. Mi lesz a keresőfüggvény értéke az alábbi táblázat esetén, ha a képletet a D8 cellába írjuk?
Képlet: =VKERES("szilva";$A$2:E$4;2;HAMIS)

	A	B	C	D	E
1	alma	körte	barack	eper	ananász
2	körte	barack	eper	ananász	meggy
3	barack	eper	ananász	meggy	szilva
4	eper	ananász	meggy	szilva	mangó

Válasz: ☐

3. Tudjuk, hogy egy Vkeres függvényt használó képletet írtak be a táblázatba, amelynek második paramétere az ábrán látható A1:E3 blokk. Melyek igazak az alábbi állítások közül?

	A	B	C	D	E
1	alma	körte	barack	eper	ananász
2	körte	barack	eper	ananász	meggy
3	barack	eper	ananász	meggy	szilva
4	eper	ananász	meggy	szilva	mangó

- ☐ A függvény válasza lehet „eper”.
- ☐ A függvény válasza lehet „mangó”.
- ☐ A függvény válasza biztosan nem lehet „alma”.
- ☐ A függvényt biztosan a „tartományban keres” opcióval kell megadni.
- ☐ A függvény csak akkor találja meg a „barack” szót az első sorban, ha a „nem tartományban történő keresést” adjuk meg.

4. Megnyitottunk egy olyan táblázatot, amelynek első oszlopát korábban elrejtették. A következő műveletek közül melyik alkalmas az oszlop megjelenítésére?

- ☐ A hozzáférési jelszó ismeretében oldjuk fel a lapvédelmet.
- ☐ A B oszlop kijelölésekor – a gomb lenyomva tartásával – húzzuk az egeret balra, a táblázaton kívülre, ezután az oszlop a helyi menüből felfedhető.
- ☐ Szúrjunk be egy új oszlopot a B oszlop elé, így az adatok ismét láthatóvá válnak.
- ☐ Egyik művelet sem, mert a feladat nem oldható meg.

16. LECKE

A táblázat, mint adatbázis

A lecke a táblázatkezelők által nyújtott adatbázis szintű szolgáltatásokat tárgyalja. Az anyag elsajátítása – a korábbiakhoz hasonlóan, az intenzív gyakorlási igény miatt – csak akkor tekinthető sikeresnek, ha a jegyzetben bemutatott és az órákon szerepelt gyakorlati feladatokat is bizonyos jártassággal meg tudja oldani a tanuló.

34. A táblázat, mint adatbázis

A táblázatkezelőkben egy adatoszlopokat tartalmazó, fejléccel rendelkező táblázat sok esetben úgy is tekinthető, mint egy adatbázis. Ilyenkor a táblázat sorait – ahol logikailag összetartozó, különböző típusú adatok találhatók – rekordoknak, egyes celláit pedig egy-egy rekord mezőinek nevezzük. A mezőneveket az első sor tartalmazza.

Az adatbázisblokkra szigorúbb szabályok vonatkoznak, mint egy általános táblázatra. Egy általános Excel vagy Calc táblázatban lehet két teljesen azonos sor vagy oszlop, egy adatbázisblokkban nem. Az a szigorúbb megkötés is érvényes, hogy adatbázis blokkban nem szerepelhetnek egyforma azonosítójú oszlopok. Nem lehet benne teljesen üres sor vagy oszlop sem, ugyanakkor az adatbázisblokkban is lehet egy vagy több üres cella.

Az adatbázisblokknak célszerű nevet adni.

Adatbázis szinten logikailag a következő műveletek hajthatók végre:

1. Új adat (rekord vagy mező) beszúrása;
2. Rekord(ok) törlése;
3. Rekord illetve mező módosítása;
4. Rekordok sorba rendezése;
5. Bizonyos tulajdonságú rekordok leválogatása, szűrése;
6. Speciális műveletek a szűrt rekordokkal.

Új adatok beszúrása illetve rekordok, vagy mezők törlése a blokkműveleteknél korábban ismertetett módon lehetséges, rekordok vagy mezők módosítását pedig egyszerűen átírással végezhetjük el. Mező vagy rekord

beszúrása természetesen új oszlop vagy sor beszúrásával indítható. Az új oszlop és sor megfelelően kitöltendő!

A rendezés és a szűrés lehetőségei csoportosítva – mindhárom általunk tárgyalt táblázatkezelőben – az **Adatok** menüben találhatók meg.

34.1. Rendezés

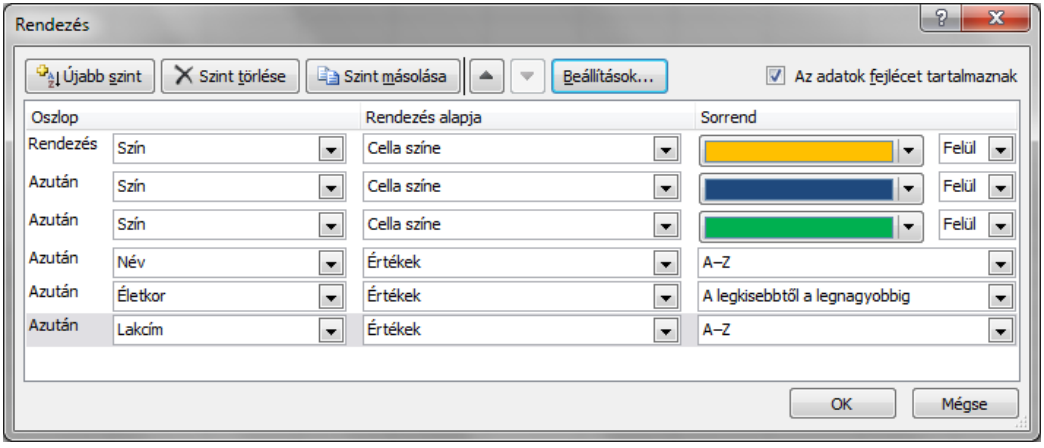
A rendezés előtt célszerű kijelölni az egész táblázatot. Ha rendezendő táblázat minden cellája ki van töltve, akkor elegendő a táblázat belsejébe állni, így a program felismeri azt. A rendezést Excel 2003-ban az **Adatok/Sorba rendezés...**, Excel 2010-ben **Adatok/Rendezés és szűrés/Rendezés**, Calc-ban az **Adatok/Rendezés...**, menüponttal indíthatjuk el.

A továbblépés előtt ellenőrizzük, hogy a táblázatkezelő helyesen ismerte-e fel a mezőazonosítók sorát vagy oszlopát. (Általában sorok szerint rendezünk, de előfordulhat az is, hogy a rendezész oszlopok szerint kell elvégezni, mivel az adatbázisunk transzponálva tartalmazza az adatokat. A mezőnevek egyik esetben se kerüljenek bele a rendezendő adatok közé.) Amennyiben szükséges, Excel 2003-ban a **Van rovatfej** illetve **Nincs rovatfej** rádiógombokkal, Excel 2010-ben **Az adatok fejlécet tartalmaznak**, Calc-ban pedig **A tartomány oszlopcímkéket tartalmaz** jelölőnégyzettel módosíthatjuk az automatikus felismerés eredményét.

Ezután be kell állítani, hogy melyik adatsor vagy oszlop szerint történjék a rendezés. Ehhez a **Rendezze** (Excel 2003) vagy **Rendezés** (Excel 2010) ill. **Rendezési szempont** (Calc) részablak bal oldali legördülő listájából ki kell választani a megfelelő mezőnevet. A rendezés jellege emelkedő vagy csökkenő lehet.

Érdekesség, hogy a 2010-es Excelben már nemcsak a cella értéke, hanem a cellaszín, betűszín és cellaikon szerint is lehet rendezni.

Az így beállított rendezés a **Szokásos** eszköztárról (Excel 2003, Calc-ban **Standard** eszköztár, nyilak ikonok) ill. menüszalagról (Excel 2010) is végrehajtható. A táblázat kijelölése után az **A→Z**, **Z→A** gombokra kattintva emelkedő illetve csökkenő rendezést kapunk. Itt figyelnünk kell arra, hogy a mezőazonosítók sora/oszlopa ne kerüljön bele a kijelölésbe, mert ekkor a mezőnevek az adatok közé keveredhetnek, ami súlyos hiba. A rendezés azon adatsor/oszlop szerint történik, amelyikben a kurzor a kijelölés végén állt.



34.1. ábra. *Rendezési beállítások Excel 2010-ben*

Előfordulhat az is, hogy nemcsak egy rendezési szempont szerint kell rendezni, hanem további szempontokat is figyelembe kell venni (például a dolgozókat rendeztük név szerint, de több azonos nevű dolgozót találtunk). Ekkor a menüből – fenti pont(ok) – a ***Rendezze*** (Excel 2003) ill. ***Rendezés*** (Excel 2010) vagy ***Rendezési szempont*** (Calc) ablakrész után a ***Majd/Azután*** ablakrész beállításával egy második rendezési szabályt is megadhatunk, sőt az (újabb) ***Azután*** részablakban még egy harmadikat is (Excel 2010-ben az ***Azután*** mező az **Újabb szint** gombra kattintva aktiválódik). A második rendezési szabály akkor dönt, ha az első rendezés szempontjából azonos adatokat talált.

Calcban és Excel 2003-ban maximum három szempont szerint tudunk rendezni, Excel 2010-ben pedig akár az összes oszlop/sor szerint is lehet.

34.2. Szűrés

Az adatbázisban a rekordok megjelenése alkalmas feltételek szerint beállítható, illetve a rekordok ezen szempontok szerint kilistázhatók. Ezt a műveletet kiválogatásnak vagy szűrésnek nevezzük.

A táblázatkezelők erre általában két különböző lehetőséget kínálnak, az AutoSzűrő helyben, csak az adatbázis és a szűrő menüjének a felhasználásával dolgozik, míg az irányított szűrő egy kimeneti blokkba vagy az eredeti táblázat helyére készíti el a végeredményt.

34.2.1. AutoSzűrő

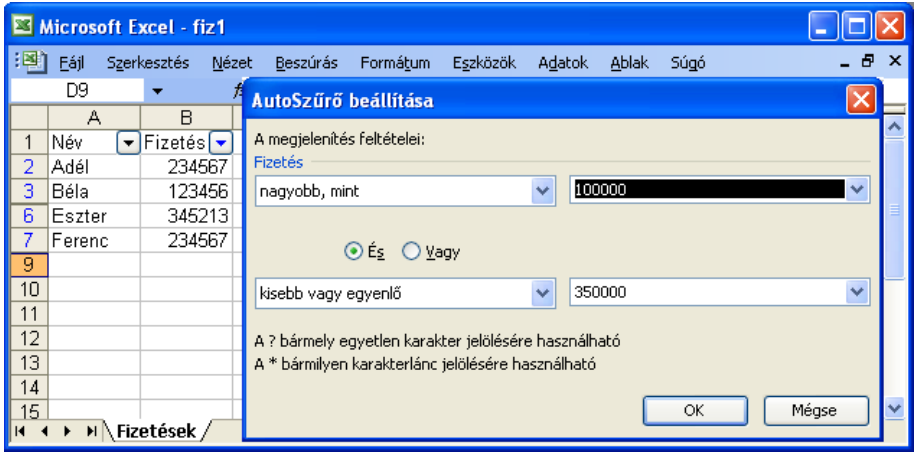
Az AutoSzűrő használatát először Excel 2003-ban mutatjuk be, majd külön ismertetjük az eltéréseket Excel 2010-ben és Calc-ban.

Az aktiváláshoz először ki kell jelölni az adatbázist (vagy legalább kattintsunk bele, hogy a program felismerje), majd az **Adatok** menü **Szűrő** pontján belül be kell kapcsolni az AutoSzűrőt. A bekapcsolt állapotot a menüben egy kis pipa jelzi. A bekapcsolás után a mezőnevek sorában a gép legördíthető listákat jelenít meg. A listákat egyenként lenyitva megadhatjuk az egyes mezőnevekre vonatkozó szűrési feltételeket.

A lehetőségek a következők:

1. **mind**: az összes rekord megjelenik;
2. **Üres**: azok a rekordok jelennek meg, amelyeknél ez a mező üres;
3. **NemÜres:** azok a rekordok jelennek meg, amelyeknél ez a mező nem üres;
4. **érték**: azok a rekordok jelennek meg, amelyeknél a mező értéke megegyezik a kiválasztott értékkel.
5. **Egyéni...**: saját szűrőfeltételt állíthatunk be. Ehhez relációk és – szöveges adatok esetén – az operációs rendszerek használatánál már ismert dzsóker jelek (*, ?) használhatók.

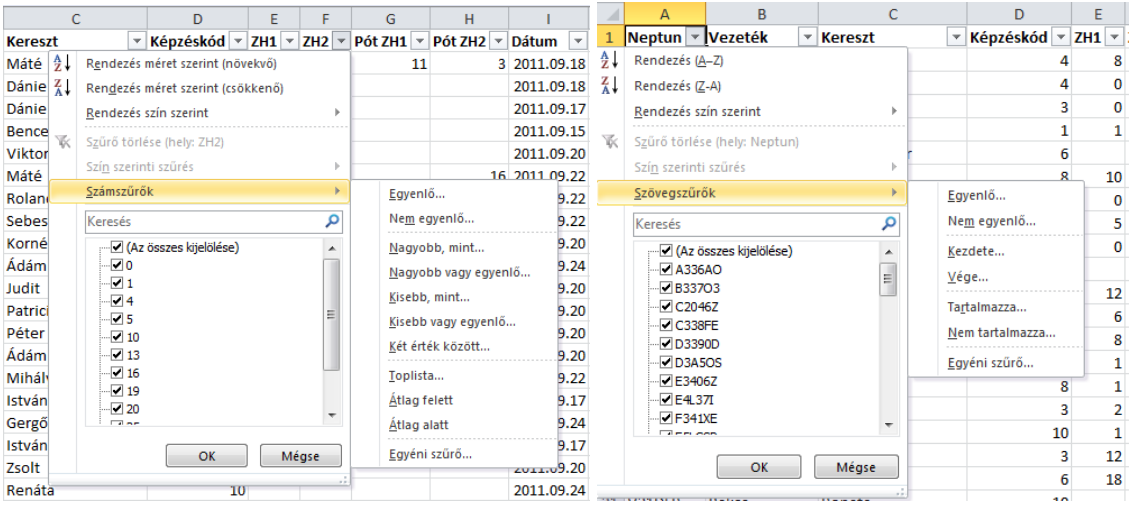
Példák: >**1500** (számtípusú cellákra), **?A*** (szöveges típusú cellákra, irányított szűrésnél ehelyett **?A** is használható). Összetett feltételeket az **És** illetve a **Vagy** kapcsológombokkal állíthatunk be (az ablak két feltétel megadására ad lehetőséget).



34.2. ábra. *Az AutoSzűrő beállítása Excel 2003-ban*

Több különböző mezőre együtt beállított feltételek között logikai és kapcsolat van.

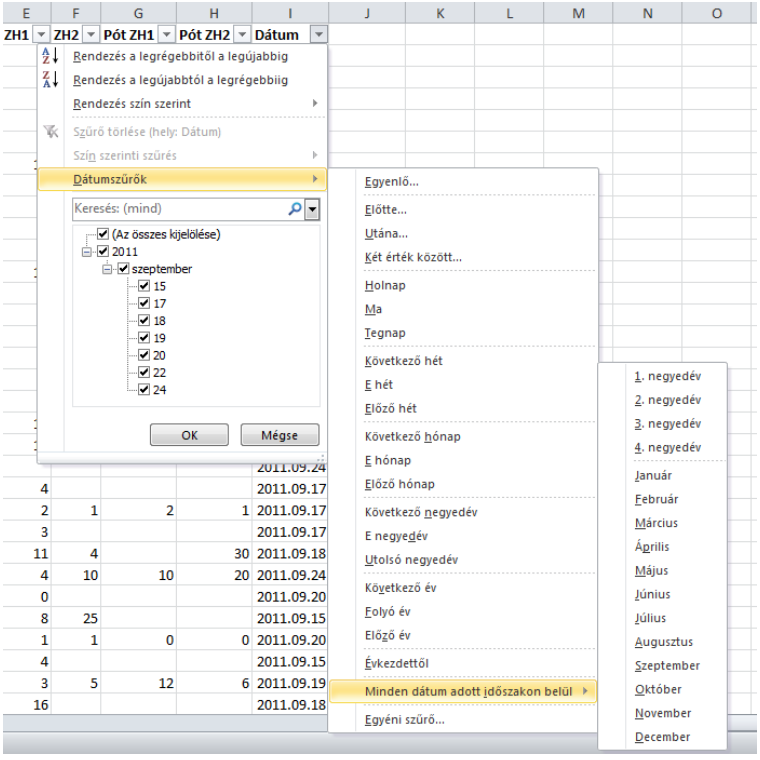
Excel 2010-ben az AutoSzűrő az **Adatok/Rendezés és szűrés/Szűrő** paranccsal érhető el. Az alapvető logika megegyezik az Excel 2003-as programban megismerttel, de a megvalósítás itt eltér abban, hogy az AutoSzűrő felismeri a szűrni kívánt adatok – aktuális oszlop – típusát (szám, szöveg, dátum), ezért speciálisabb (és azokon részben túlmutató) lehetőségeket kínál a megfelelő szám-, szöveg- és dátumszűrő segítségével.



34.3. ábra. *Szám- és szövegszűrő Excel 2010-ben*

A számszűrő érdekes új lehetősége a **Toplista** előállítása (legjobb *n* elem) és az **Átlag feletti** ill. az **Átlag alatti** elemek megjelenítése.

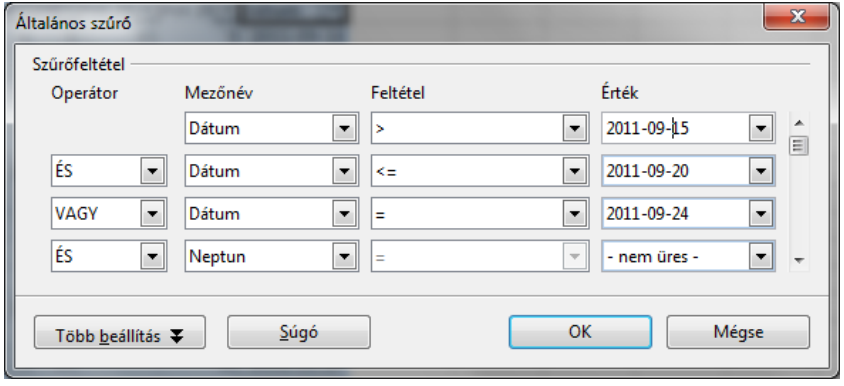
A dátumszűrő tartalmazza a legtöbb újdonságot a korábbiakhoz képest, de a beállítások itt egyszerűek. Pl. napokra, hetekre, hónapokra, negyedévekre, évekre lebontva lehetőségünk van az előző, az aktuális és a következő időegységbe eső adatok direkt megjelenítésére (maszkok – dzsókerjelek – megadása nélkül).



34.4. ábra. *Dátumszűrő Excel 2010-ben*

Calc-ban az AutoSzűrő az **Adatok/Szűrő/Automatikus szűrű** menüpont segítségével érhető el. A megfelelő listát lenyitva az Excel 2003-as felületéhez hasonló pontokat láthatunk (pl. **Üres**, **Nem üres**). Eltérés, hogy az Egyéni szűrőbeállítást (Excel 2003) itt **Általános szűrőnek** nevezik, és ez a lehetőség nemcsak az Automatikus szűrő alól, hanem az **Adatok/Szűrő/Általános szűrő...** menüponttal is elérhető.

Az Általános szűrő abban is különbözik az Excel 2003-ban és 2010-ben megismert AutoSzűrő beállításaitól, hogy itt egy oszlopot nemcsak két feltétel szerint lehet szűrni, hanem jóval több ilyen feltétel is megadható. További eltérés, hogy az Általános szűrőben az aktuálistól különböző további oszlopokra is be lehet állítani szűrőfeltételt, nem kell ehhez újra elindítani az AutoSzűrő egyéni beállítását.



34.5. ábra. *Általános szűrő Calc-ban*

A legérdekesebb plusz lehetőség a Calc általános szűrőjében azonban mindenképpen az, hogy *reguláris kifejezések* felhasználásával is lehet szűrni. Ehhez a **Több beállítás** nyomógombra kell kattintanunk. A reguláris kifejezések segítségével keresési maszkok adhatók meg, amelyek a szöveg- illetve -számtípusú adatokra való illeszkedés szerint eredményeznek találatot. Példák Calc környezetben: [:alpha:]{3} – pontosan három betű; C[:digit:]+.* – C-vel kezdődő kódok, amelyek 2. karaktere számjegy, a továbbiak tetszőlegesek.

A reguláris kifejezések felépítéséhez használható elemek részletesen áttekinthetők a Calc súgójában. Ilyen típusú kifejezéseket használni fogunk a Calc irányított szűrésénél is és további gyakorlati alkalmazásukkal a „Kiadványszerkesztés” című fejezetben (27. fejezet) mélyebben is foglalkoztunk.

Az Általános szűrő lehetőséget kínál arra is, hogy a szűrés eredményét a munkalap üres helyére tegyük – ez a pont már az irányított szűrés felé vezet.

34.2.2. Irányított szűrő

Az irányított szűrő használatához a táblázatkezelőkben általában három blokkra van szükségünk.

Az *adatbázis* mellett szükséges egy *szűrőtartomány*, amelyet a szűrés előtt nekünk kell létrehoznunk. Ennek felső sora azokat a mezőneveket tartalmazza, amely mezők tartalmára feltételt szeretnénk szabni. Ezek alatt következnek a feltételek. Megadásuk az egyes táblázatkezelőkben lényegében ugyanolyan szintaktikával történik, mint az AutoSzűrő esetén. (Excelben: dzsókerjelek és relációk; Calcban: reguláris kifejezések és relációk használata). Ha összetett logikai feltételre van szükségünk, akkor az **és** kapcsolatot egymás mellé írással, a **vagy** kapcsolatot egymás alá írással jelezzük. Ha az **És** kapcsolat ugyanarra a mezőre megadott több feltételre vonatkozik, akkor az adott mezőnév többször fog szerepelni a szűrőtartományban. A szűrőtartomány tartalmazhat felesleges mezőneveket is (amelyekre nem adunk meg szűrési feltételt), de ilyeneket nem célszerű felvenni, törekedjünk a minimális szűrőtartomány használatára. A szűrőtartomány üres cellája azt jelenti, hogy arra a mezőre amiben ez a cella van, nem adtunk meg feltételt. Ez a vagy kapcsolatra hibás eredményt adhat!

Aktivitás: Gondolja át a Boole-algebra műveleteit felidézve, hogy ez miért igaz!

Megjegyezzük, hogy a szűrőtartomány megtervezése és korrekt felépítése általában a szűrés legnehezebb része. Komolyabb gyakorlati problémák esetén ez a fázis viszonylag hosszú gondolkodást, időráfordítást is igényelhet, és sok hibát ejthetünk, ha a feladatot nem gondoljuk át kellőképpen! (Lásd még: feladatok lent.)

Ha nem helyben szűrünk, akkor Excelben létre kell hozni egy *kimeneti blokkot* is, ahol az eredmények megjelennek. Ez a blokk a megjeleníteni kívánt mezők neveit tartalmazza, alatta megfelelő üres területtel, ahová a gép a listát elhelyezi.

Fontos, hogy a mezőnevek mindhárom használt blokkban pontosan ugyanazok legyenek – eltérő szóköz vagy ékezet sem lehet, ezért az a legjobb, ha a blokkok létrehozásakor a mezőneveket másolással készítjük el az új blokkokban.

Új szűrésnél mindig új kimeneti blokkot kell készítenünk, ha a régit használjuk, akkor a régebben szűrt adatok elvesznek.

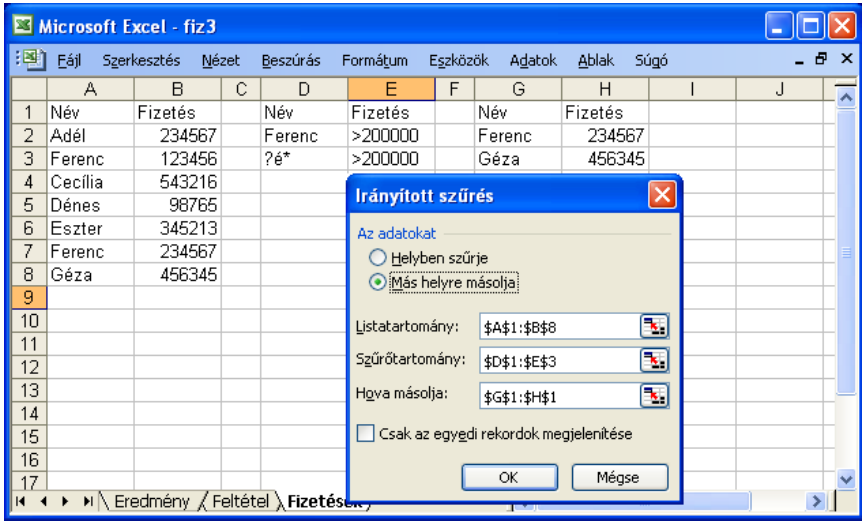
A szűrés technikai végrehajtása Excelben úgy történik, hogy a blokkok gondos elkészítése után az **Adatok/Szűrő** ponton belül az **Irányított szűrő...** alpontot (Excel 2003) ill. az **Adatok/Rendezés és szűrés/Speciális** pontot (Excel 2010) választjuk; megadjuk vagy kijelöljük a listatartományt – azaz az adatbázist –, a szűrőtartományt, majd ha nem helyben szűrünk, akkor a **Más helyre másolja** gomb bekapcsolása után megadhatjuk a kimeneti blokk címét, vagy a számára biztosított területet fejléccel együtt. Az **OK** gombra kattintva a táblázatkezelő elkészíti a listát.

A logikai menet Calc-ban is teljesen azonos, és a használat is nagyjából megegyezik az Excel 2003-as verzióéval. Eltérés, hogy a kimeneti blokk megadásához itt a **Részletek** gombra kell kattintani, és ezután az **Eredmény másolási helye** kapcsolót kell aktiválni. Fontos különbség még az Excelben végrehajtott szűréshez képest, hogy a kimeneti blokk fejlécét nem kell előre elkészíteni, mivel automatikusan a kiindulási táblázat fejléce lesz. Elég, ha megadunk egy megfelelő területet. Ennek következménye, hogy Calc-ban mindig minden oszlop megjelenik a kimeneti blokkban.

Ha a kimeneti blokk megadásánál nemcsak a fejlécet, hanem a teljes output területet kijelöljük, akkor a táblázatkezelő csak akkora részt használ a lista elkészítésére, amennyi a kijelölésben rendelkezésre áll. Ha erre a területre nem fér rá minden rekord, amely a feltételnek eleget tesz, akkor a többlet elveszik, nem jelenik meg a kigyűjtésben.

Ha a Helyben szűrést választottuk, az adatbázisunk soraiból csak a szűrőfeltételeknek megfelelő sorok maradnak láthatóak a szűrés befejezése után. Az **Adatok/Szűrő/Minden látszik** (Excel 2003) ill. **Adatok/Rendezés és szűrés/Szűrők törlése** (Excel 2010) vagy az **Adatok/Szűrő/Szűrő eltávolítása** (Calc) alponttal lehet újra látni a teljes adatbázist.

A 34.6 ábrán bemutatott egyszerű példával ellentétben, valódi gyakorlati problémáknál az irányított szűrést célszerű úgy végrehajtani, hogy a használt blokkokat különböző munkalapokon legyenek Nem elegáns pl. a szűrési feltételeket az adatbázisblokk mellé felépíteni. Ilyenkor a szűrés technikailag bonyolultabbá válik.

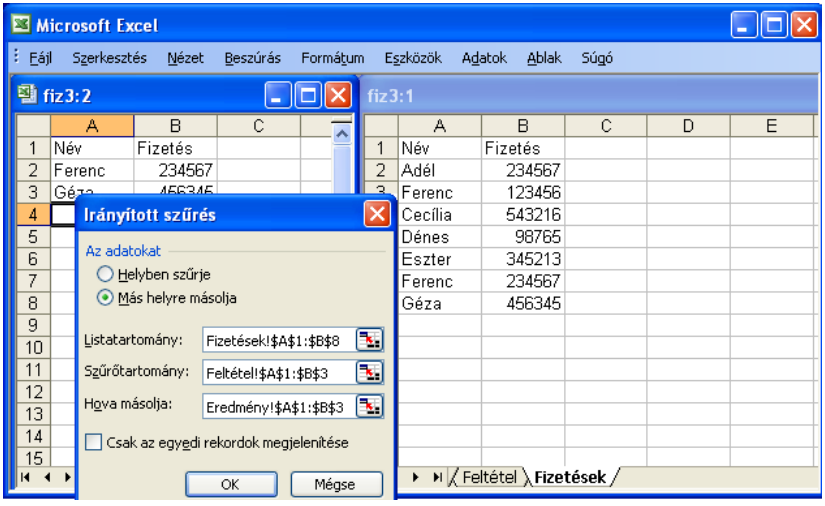


	A	B	C	D	E	F	G	H	I	J
1	Név	Fizetés		Név	Fizetés		Név	Fizetés		
2	Adél	234567		Ferenc	>200000		Ferenc	234567		
3	Ferenc	123456		?é*	>200000		Géza	456345		
4	Cecília	543216								
5	Dénes	98765								
6	Eszter	345213								
7	Ferenc	234567								
8	Géza	456345								
9										
10										
11										
12										
13										
14										
15										
16										
17										

34.6. ábra. *Egyszerű irányított szűrés Excel 2003-ban*

Ekkor mindig arról a munkalapról kell indítani, ahol a kimeneti blokkot várjuk. Ha nem így teszünk, akkor a szűrés végén hibaüzenetet kapunk, ugyanis a szűrt adatok csak az aktív munkalapra kerülhetnek. De a munkalapon belül sem mindegy, hogy hol állunk. Ha a szűrést a kimeneti fejlécéből indítjuk (Excel), akkor a gép azt akarja adatbázisnak tekinteni, és így szintén hibaüzenetet kapunk. Legjobb, ha egy vagy két üres sort kihagyunk a fejléc alatt, és onnan kezdjük a szűrést.

Ha az eredeti táblázat adatai a szűrés után megváltoznak, akkor az Excel a már kiszűrt listát nem változtatja meg. A frissítés itt nem is indokolt, hiszen a szűrés eredménye mindig egy kigyűjtés, a szűrés időpontjának megfelelő állapot szerint.



34.7. ábra. *Irányított szűrő használata több munkalap esetén Excel 2003-ban*

Tizenhetedik feladat (részben önálló gyakorlásra)

Egy DVD-kölcsönző a kölcsönzésekről a következő adatokat tárolja: kölcsönzött film, ügyfél neve, ügyfél lakcíme (város, utca, házszám), kölcsönzési idő, kölcsönzési díj. Listáztassuk ki

1. *a győri lakosok által kikölcsönzött filmek közül azokat, amelyek 3 napnál tovább vannak kint;*
2. *azon Eszter vagy Erika keresztnevű ügyfelek által kikölcsönzött filmeket, akik a Liszt Ferenc utcában laknak.*

Készítsünk szűrőtartományt a feladat megoldására Excelben és Calc-ban, majd hajtsuk is végre a szűrést!

Tizennyolcadik feladat (részben önálló gyakorlásra)

Egy autókereskedés az autókról a következő adatokat tárolja: név (azonosító), szín, évjárat és ár (millió Ft-ban). Ki kell listáztatnunk a zöld vagy piros színű, 1990-es években gyártott, 1 és 2 millió Ft közötti áru autókat. Készítsünk szűrőtartományt a feladat megoldására Excelben és Calc-ban, majd hajtsuk is végre a szűrést!

Segítség: rövid magyarázattal bemutatjuk a feladatok megoldásához használható szűrőtartományokat (Excel).

ügyfél lakcíme	kölcsönzési idő
Győr, *	>3

Vagy – mivel a * karakter a maszk végéről irányított szűrésnél elhagyható:

ügyfél lakcíme	kölcsönzési idő
Győr,	>3

Felhasználjuk, hogy az Excelben a * a kerektersorozat bármelyik részének a helyettesítésére is használható:

ügyfél neve	ügyfél lakcíme
* Erika	* Liszt Ferenc utca
* Eszter	* Liszt Ferenc utca

Aktivitás: Járjon utána, mi a különbség az excel és operációs rendszerek * használata között!

Az évjárat megadásánál az adat számtípusa miatt nem használhatjuk a * karaktert:

szín	évjárat	évjárat	ár (millió)	ár (millió)
piros	>=1990	<=1999	>=1	<=2
zöld	>=1990	<=1999	>=1	<=2

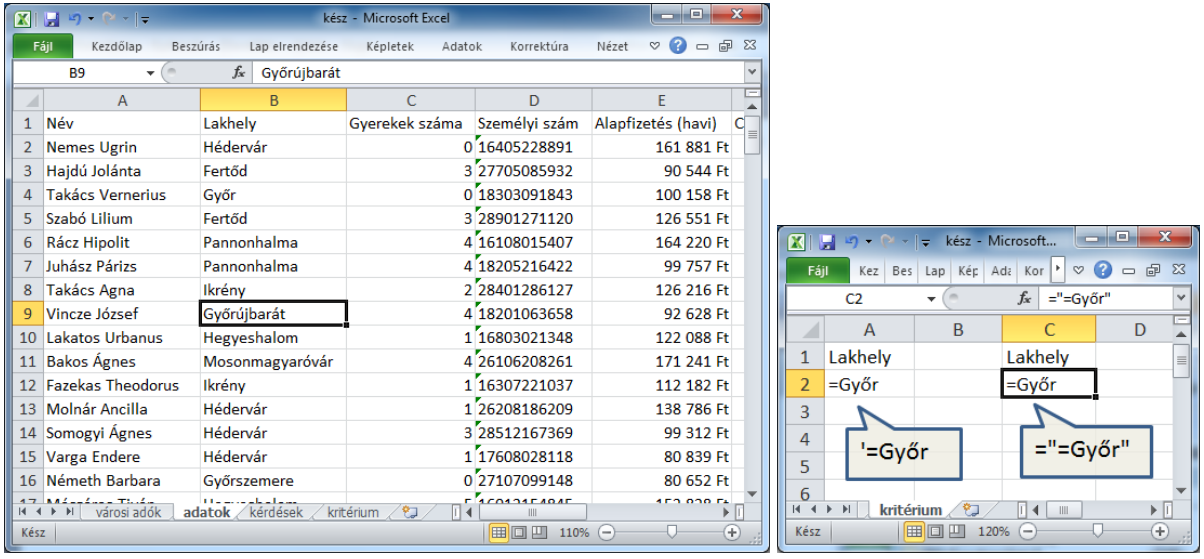
Tizenkilencedik feladat (önálló gyakorlásra)

Értelmezzük az alábbi szűrőtartományokat! Mi lesz a szűrés eredménye?

ügyfél neve	ügyfél lakcíme
Tamás	* Liszt Ferenc utca
	* Bartók Béla út

szín	szín	ár (millió)
piros	zöld	>=1

A győri lakosok kiszűrésére fent bemutatott szűrőtartomány felvet egy érdekes problémát. Mit tudunk tenni akkor, ha a lakcím mezőben csak a település neve szerepel, és nincs utána a „segítő” vessző jel, amivel könnyen le tudtuk választani a számunkra most nem szükséges győrújbaráti, győrszemerei stb. lakosokat? Ilyen esetekben a szűrőtartomány megfelelő cellájában az =Győr karakterláncnak kell szerepelnie, ekkor kapjuk meg a megoldást. A két lehetséges jó beírást a következő ábra mutatja be.

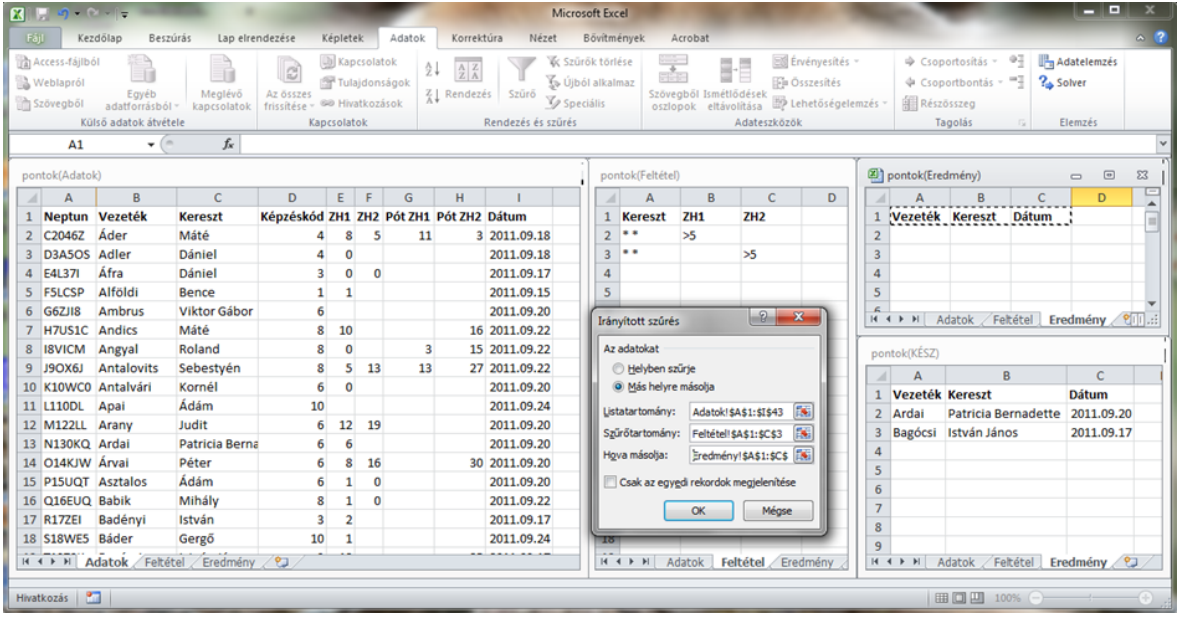


34.8. ábra. *Győri lakosok szűrése – bonyolultabb eset Excel 2010-ben*

Huszadik feladat (önálló gyakorlásra)

Oldjuk meg az előző ábrán bemutatott feladatot Calc-ban is!

Segítség: a szűrőtartomány feltételeit Calc-ban – a fentiekben már röviden ismertetett – reguláris kifejezésekből építjük fel.



34.9. ábra. *Az irányított szűrő használata Excel 2010-ben – olyan dupla keresztnevű hallgatók kiválogatása, akik valamelyik zh-t legalább 6 pontra megírták*

34.3. Adatbázis-kezelő függvények

Az adatbázisblokk bizonyos feltételeknek eleget tevő rekordjait nemcsak listáztatni lehet, hanem a mezőik adataival műveleteket is végezhetünk. Erre az adatbázis-kezelő függvények szolgálnak.

A függvények általános alakja a következő:

AB.függvénynév(adatbázisblokk; mező; feltételtábla).

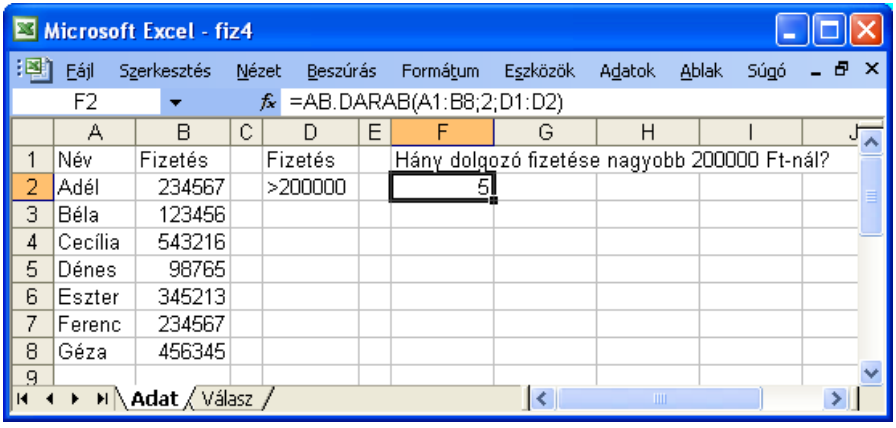
Az adatbázisblokk és a feltételtábla ugyanúgy adható meg, illetve építhető fel, mint az irányított szűrésnél. A mezőparamétert a mező nevével (idézőjelek közé téve, például a 34.10. ábrán: "Fizetés") vagy a mezőnévre való hivatkozással vagy oszlopsorszámmal megadhatjuk meg. Más mezőmegadás hibás (pl. B2; az „eredmény”: #ÉRTÉK!).

A szűrési feltételnek eleget tevő rekordok mezőin illetve mezőinek adatain a leggyakrabban a következő műveleteket végezzük el: számlálás, összegzés, maximum- illetve minimumkeresés, átlagszámítás. A megfelelő függvények: **AB.Darab, AB.Darab2**, **AB.Szum**, **AB.Max**, **AB.Min**, **AB.Átlag**. A **Darab** és a **Darab2** függvények közötti különbség az, hogy az egyik a függvény második paraméterével megadott oszlop számértékű, a másik pedig a nem üres mezőit számolja össze.

Az adatbázis-kezelő függvények fontos tulajdonsága, hogy – hasonlóan más függvényekhez – az adatbázis változása esetén – amennyiben az automatikus számolási opció nincs kikapcsolva –, automatikusan frissítik az értéküket.

Az **AB.Szum** és az **AB.Darab** függvényekhez hasonlóan működnek, és hasonló, de egyszerűbb feladatok megoldására használhatók a **Szumha** és a **Darabteli** függvények. Ezeket a függvényeket nem számítjuk az adatbázis-kezelő függvények közé.

Az adatbázis-kezelő függvények használata a tárgyalt táblázatkezelőkben megegyezik.



	A	B	C	D	E	F	G	H	I	J
1	Név	Fizetés		Fizetés		Hány dolgozó fizetése nagyobb 200000 Ft-nál?				
2	Adél	234567		>200000		5				
3	Béla	123456								
4	Cecília	543216								
5	Dénes	98765								
6	Eszter	345213								
7	Ferenc	234567								
8	Géza	456345								
9										

34.10. ábra. *Adatbázis-kezelő függvény használata*

Huszonegyedik feladat (önálló gyakorlásra)

Az autókereskedéses adatbázist használva függvény segítségével válaszoljunk az alábbi kérdésekre:

1. *Hány darab zöld autónk van?*
2. *Mennyi a piros autók átlagos ára?*
3. *Mennyibe kerül a legdrágább piros autó?*
4. *Hány darab olyan zöld autónk van, amit az 1990-es években gyártottak?*
5. *Hány darab olyan piros vagy zöld autónk van, amit az 1990-es években gyártottak?*

Az adatbázis-kezelő függvényeket ügyes, fejlett – csoportos – módon tudjuk alkalmazni olyan esetekben, amikor a feladatban egymás után több kérdést kell megválaszolni ezekkel az eszközökkel (pl. mennyi a januári, februári, márciusi stb. filmkölcsönzések összes bevétele). Ekkor elkészítjük a megfelelő szűrőtartományokat egymás mellett, és alatta egy másolható (!) képlettel előállítjuk a megoldást.

34.4. Kimutatások

Gyakori feladat, hogy ismétlődő adatelemeket tartalmazó listákból olyan táblázatot készítsünk, amely bizonyos szempontok szerint csoportosítja és összegzi az adatokat. Ezek a táblázatok a kimutatástáblák. Elkészítésükhöz a kimutatás-varázslót illetve a kimutatástündért használhatjuk. A varázsló vagy tündér a következő módokon indítható: **Adatok/Kimutatás vagy kimutatásdiagram...** menüpont (Excel 2003); **Beszúrás/Táblázatok/Kimutatás/Kimutatás** ill. **Kimutatásdiagram** (Excel 2010), **Adatok/Kimutatástábla/Létrehozás...** (Calc).

A kimutatástáblát a Calc 3.5. előtti verziói adattündérnek nevezik.

Kimutatást általában egy listából vagy adatbázisból, illetve több különálló tartományból készítünk. A varázsló vagy tündér indítása után először meg kell adnunk a kimutatás forrását. Ez egy táblázatblokk, nem feltétlenül egyedi adatokat tartalmaz, tehát nem mindig adatbázis. Ezután definiálnunk kell, hogy a kigyűjtés hova kerüljön. Ha nem új munkalapra dolgozunk, akkor itt elegendő egyetlen cellacímet megadni, de ügyelni kell arra, hogy a kigyűjtést tartalmazó táblázat elférjen a megadott cellától lefelé és jobbra.

Vigyázat, a kigyűjtés táblázata esetenként jóval nagyobb területet foglalhat el, mint a forrásadatokat tartalmazó blokk!

Ezt követően meg kell határozni a kigyűjtés szempontjait, meg kell mondani, hogy az egyes szempontok alapján kigyűjtött adatok sor vagy oszlop szerint kerüljenek-e a táblázatba, és meg kell adni azt is, hogy a kigyűjtés melyik adatok közül történjen. A kész kimutatás a varázsló vagy a tündér segítsége nélkül is átrendezhető, átalakítható.

A kimutatástáblázat a forrásadatok változtatásával nem frissül automatikusan, és az átrendezés vagy átalakítás is a régi adatok alapján történik. A frissítést manuálisan kell elindítani (menüből, eszköztárról vagy gyorsbillentyű segítségével), a lehetőségek a következők:

1. Excel 2003: **Adatok/Adatfrissítés** vagy **Adatfrissítés** gomb a Kimutatás eszköztáron
2. Excel 2010: **Elemzés/Adatok/Frissítés** vagy ALT+F5
3. Calc: **Adatok/Kimutatástábla/Frissítés**.

Huszonkettedik feladat

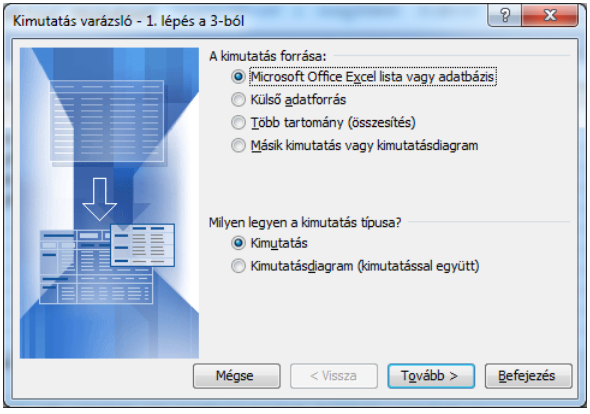
Egy nagykereskedő a megrendeléseiről a következő adatokat tarja nyilván: a megrendelő neve, a megrendelés tárgya, mennyisége, ideje (hónap), valamint a megrendelés teljesítését végző alkalmazott neve. Készítsünk kimutatást és összesítést a megrendelésekről a következő szempontok szerint. Legyen leolvasható a kimutatásról, hogy ki volt a megrendelést elintéző munkatárs, milyen terméket és mikorra rendelt meg a megrendelő.

Első lépésben elkészítjük az alaptáblázatot. Ezt egyszerű begépeléssel a megfelelő fejlécet megadva megtehetjük.

Második lépésben a kigyűjtésvarázslót vagy -tündért használva elkészítjük a kigyűjtést. Ehhez kijelöljük az előbb elkészített táblázatot, és a menüből elindítjuk a varázslót vagy tündért. A kigyűjtés végrehajtása az egyes táblázatkezelőkben eltérő, ezért ezeket programonként külön bemutatjuk.

Excel 2003-ban az első ablakban az automatikusan bekapcsolt **Microsoft Office Excel lista vagy adatbázis**, valamint a **Kimutatás** rádiógombot bekapcsolva hagyjuk, majd a **Tovább** > gombot megnyomva a második ablakhoz lépünk. A varázsló indítása előtt kijelölt táblázat blokkcíme megjelenik a **Tartomány** nevű input mezőben. Ha elfelejtettük volna a forrásokat tartalmazó táblázat kijelölését vagy rosszul jelöltük volna ki, a mező kitöltésével a hiányt pótolhatjuk, vagy javíthatjuk. A **Tovább** > gombbal a harmadik lépésre mehetünk.

A harmadik ablakban megadhatjuk, hogy a kimutatástáblázat hova kerüljön. Kerülhet a forrásadatokat tartalmazó munkalapra, de máshova is. A megfelelő rádiógombot bekapcsoljuk, majd az ablakon lévő inputmezőben megadjuk a kimutatás-táblázat bal felső sarkát kijelölő cella címét.

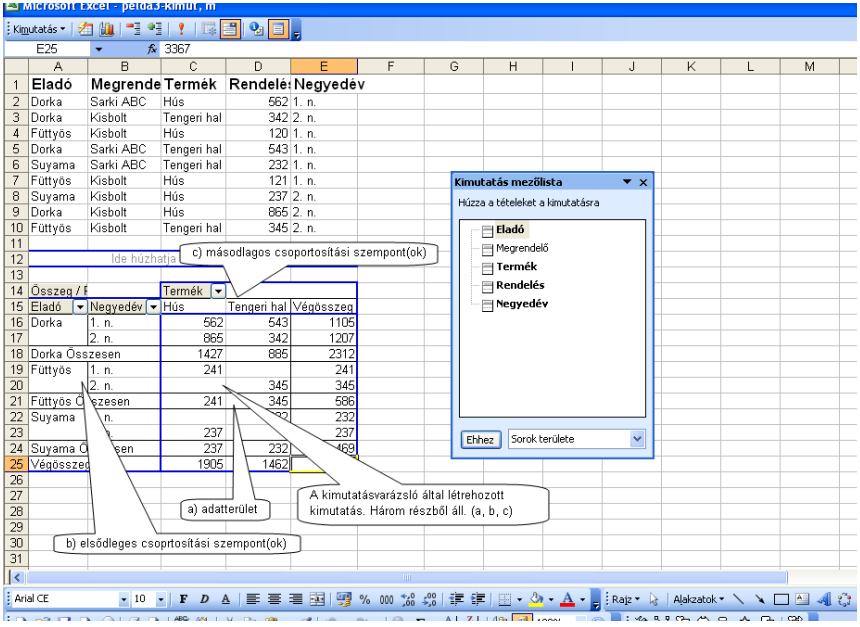


34.11. ábra. *A kimutatásvarázsló indítása Excel 2003-ban*

Ezután az **Elrendezés...** gombra kattintunk. A megjelenő ablakban megtervezhetjük a kimutatás szempontjait, adatait A jobb oldali listából (itt azokat az oszlopneveket találjuk, amiket a forrásadatok oszlopaihoz rendeltünk), egérrel megfogjuk a megfelelőt, és a bal oldali kimutatássémának megfelelő részére húzzuk. Példánkban a sor mezőre a Munkatársunk (Eladó) és a Megrendelő, az oszlopra a Termék és a Megrendelés ideje, az adatterületre a Megrendelés mennyisége kerül. (ha a kigyűjtés áttekinthetően tartalmazza azt, amit a feladatmegoldástól várunk, akkor más megoldás is elfogadható). Ha a kialakítással készen vagyunk, az **OK** gombbal bezárjuk az elrendezés-tervezést, majd a **Befejezés** gomb megnyomása után elkészül az általunk definiált séma szerint a kimutatás, összesítés.

Ha a kimutatás hierarchiája esetleg nem megfelelő, akkor ezt megváltoztathatjuk úgy, hogy egérrel megfogjuk azt a címkét, amelyik nem jó helyen van és a jobbnak vélt pozícióba mozgatjuk. A mozgatható címkék mellett egy lefelé mutató nyilat tartalmazó gomb van. Erre kattintva egy ablak nyílik, amelyben beállíthatjuk azt is, hogy az adott címkéhez tartozó adatok közül melyeket jelenítse meg a kimutatástáblában a táblázatkezelő.

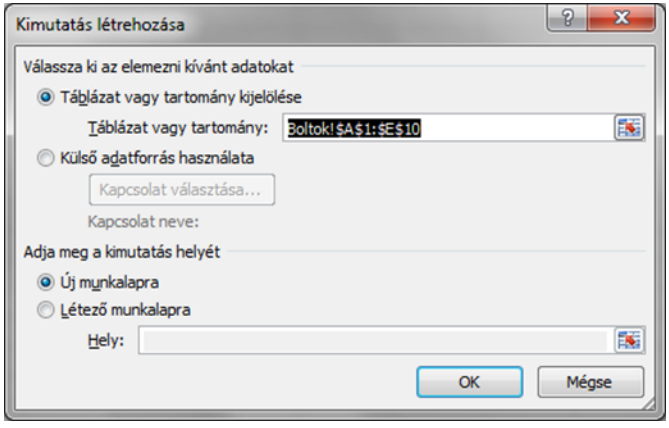
Lehetőség van még a kimutatástáblázat formai megjelenítésének módosítására is.



34.12. ábra. *Összesítések kigyűjtéssel Excel 2003-ban*

Excel 2010-ben a varázsló első három ablakát kissé leegyszerűsítve egybevonták.

A megfelelő adatok megadása után az **OK** gombra kattintva máris eljutunk a kialakításhoz. A 2003-as változattól eltérően itt a jobb oldali ablakrészben a *sor-* és *oszlopcímkéknél* ill. a Σ *értékeknél* lehet megadni, hogy melyik mezőnév szerepeljen az elrendezés az adott helyén. A felhasználni kívánt mezők a kimutatásablak jobb felső mezőlistájában kijelölendők. A *jelentésszűrőbe* felvett mezőnevek segítségével AutoSzűrő funkciót tudunk hozzárakni a jelentéshez, így az egyébként már kész kimutatás adatainak megjelenését korlátozhatjuk. Ilyen lehetőségek a 2003-as Excelben is vannak, de jóval szűkebben.

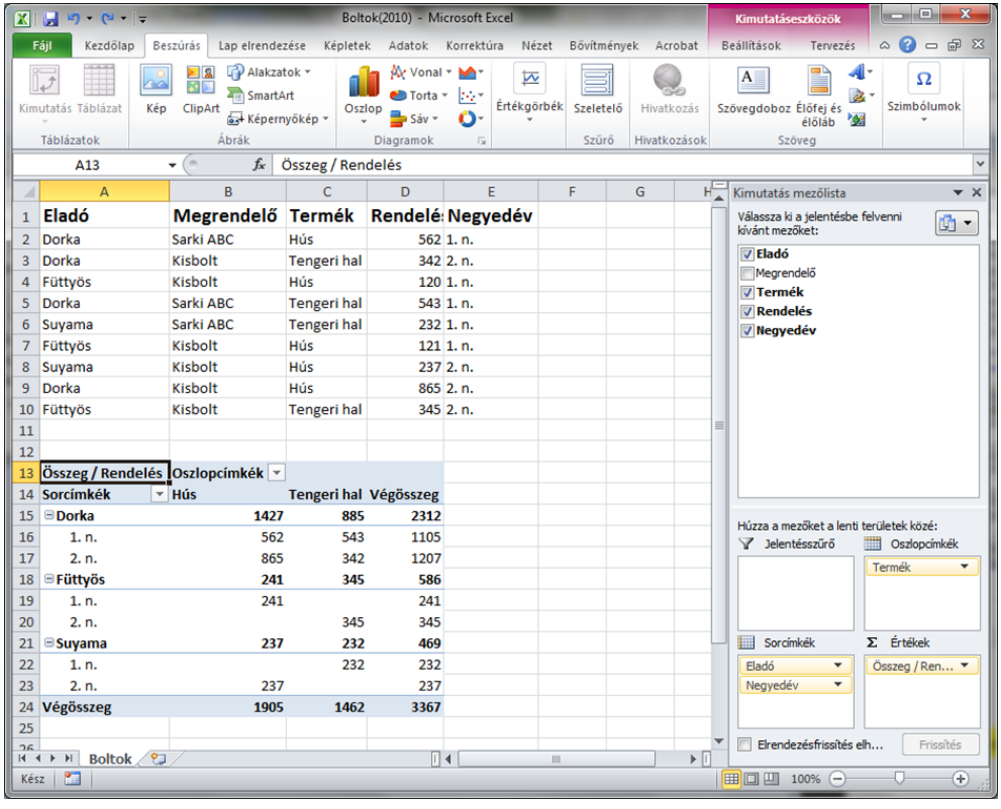


34.13. ábra. *Kimutatásvarázsló Excel 2010-ben*

A Σ értékekhez felvett mezőkön a kimutatás alapértelmezés szerint szöveges adatok esetén a kitöltött mezők darabszámát szám típusú adatok esetén a számok összegét számolja. A számolás módját lehet változtatni, így átlagot, minimumot, maximumot... is számoltathatunk.

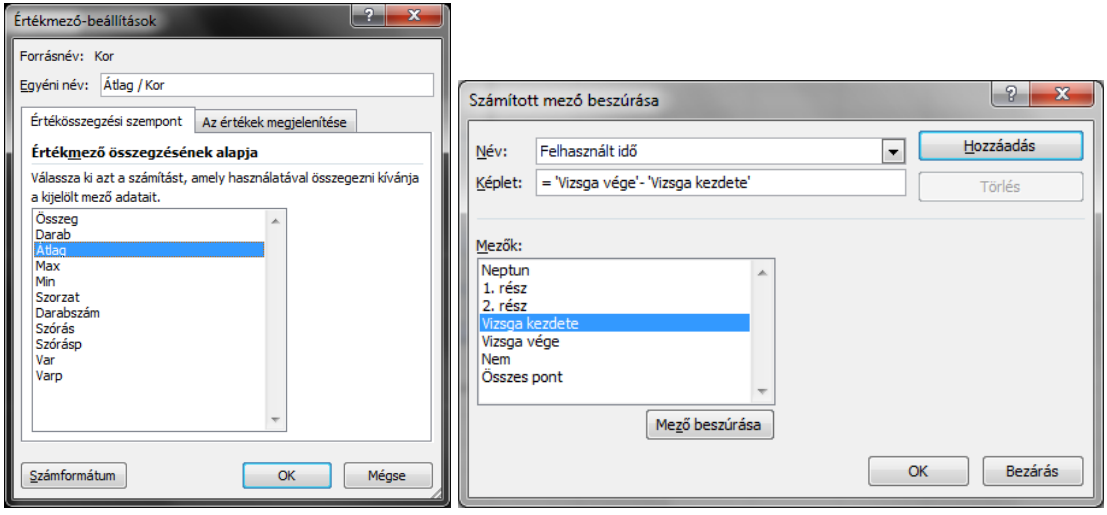
A 2010-es Excelben nemcsak a táblázatban lévő mezőkkel lehet kalkulálni a kimutatás során, hanem új úgynevezett számított mezőt is be lehet szúrni. Ennek a funkciónak a használata azért kényelmes, mert így nem kell a táblázatunkba új segédoszlopot felvenni a különböző számításokhoz. Számított mező készítésénél függvényeket is lehet használni, de nem megengedettek a változó eredményt adó típusok, mint pl. a véletlenszám generátorok, a MA() vagy a MOST() függvény.

Az Excel 2003-hoz hasonlóan itt is módosíthatjuk a már elkészített kimutatás-struktúrát a mezőnevek átmozgatásával. A kis lefelé mutató nyilat tartalmazó gombokkal szintén az adott címkéhez tartozó adatok megjelenítésére adhatunk meg feltételt, itt azonban bővebb lehetőségeink vannak: rendezhetünk, ill. AutoSzűrő-szerű módon szűrhetünk (feliratra és értékre egyaránt).



34.14. ábra. *Összesítések kigyűjtéssel Excel 2010-ben*

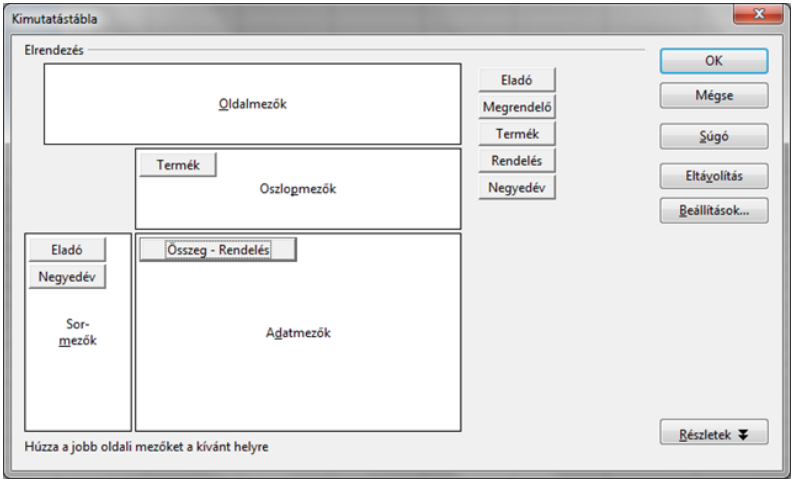
A 2010-es Excelben megváltoztathatjuk még az egész kimutatás formai megjelenését (ha az alapértelmezett elrendezéssel elégedetlenek vagyunk), a jobb oldali ablakrész felső sarkában található kis lenyíló lista segítségével (Mezők szakasz/területek szakasz megjelenítése).



34.15. ábra. *Értékmező-beállítások és számított mező beszúrása Excel 2010-ben*

A kimutatásdiagram ugyanúgy készíthető, mint a kimutatás, csak más csak más beállításokkal kell dolgozni a varázslóval Excel 2003 és 2010-ben egyaránt. Ekkor – a megadás befejeztével – az elemzés grafikus formában is megjelenik. A szemléltetéshez az alapértelmezett diagramtípus az oszlopdiagram, ezen belül sokféle altípus közül választhatunk.

Calc-ban a kimutatáskészítéshez csak egy ablakot kell kezelni a mezők elhelyezése előtt (ezzel adjuk meg forrást). Ha az elemezni kívánt adatokat tartalmazó blokkot a tündér indítása előtt kijelöltük, akkor ebben az ablakban nem kell semmit beállítani, csak az **OK** gombbal tovább kell lépni. Ezután az elrendezés, a kialakítás következik, amely nagyon hasonlít ahhoz, amit a 2003-as Excelben követtünk. A sor-, oszlop- és adatmezők megadása úgy történik, hogy a megfelelő fejléc címkék egérrel áthúzzuk a kimutatássémára. Az oldalmezők szerepe ugyanaz, mint a 2010-es változatban a jelentésszűrőké, segítségükkel kizárhatók azok az adatok, amiknek felhasználása az adott kimutatásban nem szükséges.



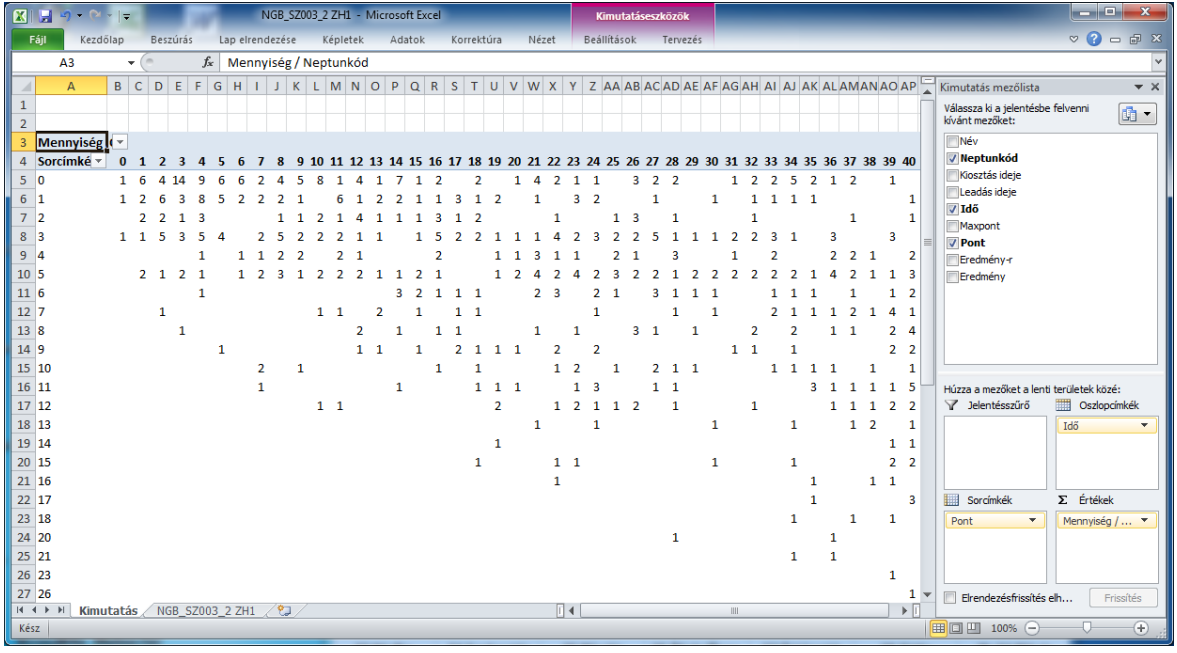
34.16. ábra. *Kimutatás kialakítása Calc-ban*

Az elkészült kimutatás a Calc-ban is módosítható, struktúrájában és formailag egyaránt.

Az összesítéssel kigyűjtött adatok egyes speciális esetekben felhasználhatók statisztikai elemzésekre is. (Erre egyébként a táblázatkezelők általában nyújtanak fejlett céleszközt – ezt a második félévben tanuljuk külön is.) Zh eredményeket tartalmazó számítógépes adatbázis használatával kimutatással kereshetjük a választ pl. olyan típusú kérdésekre, hogy:

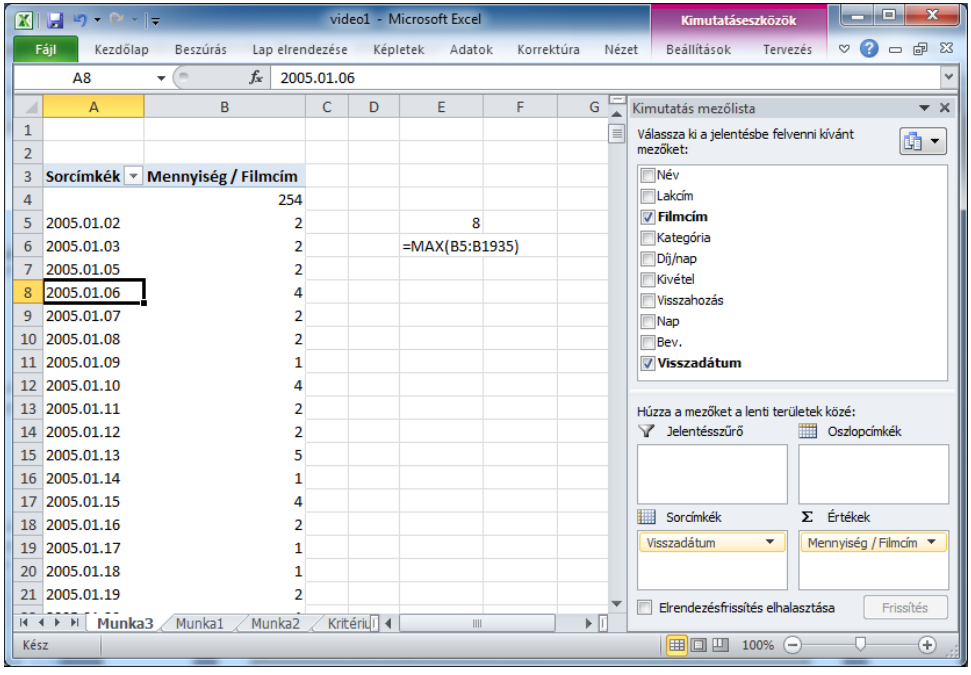
1. Van-e korreláció a két félévközi zh eredménye között? (Remélhetőleg igen, azaz aki jobb eredményt ért el az első zh-n, vélhetőleg a másodikon is sikeresebb.)
2. Van-e korreláció az elektronikus teszt kitöltésére ráfordított idő és az eredmény között?

3. Kerülik-e a hallgatók kimutathatóan valamelyik témakör (feladatcsoport) megtanulását?
4. Van-e olyan feladatcsoport, amely „nem mér” (100%-os teljesítés), illetve olyan, ami túl nehéznek bizonyult (egy hallgató sem senki sem tudta hibátlanul megcsinálni)?



34.17. ábra. *Korrelációelemzés kimutatással (eredmény/idő, Excel 2010)*

A kimutatás képlettel kombinálva gyakran erősebb eszköz a feladatok megoldására. A következő ábrán egy ilyen esetet mutatunk be. (A videókölcsönzős példában meg kellett határozni, hogy melyik volt az a dátum, amikor a legtöbb filmet hozták vissza).



34.18. ábra. *Kimutatás és képlet (Excel 2010)*

34.5. Érvényességellenőrzés

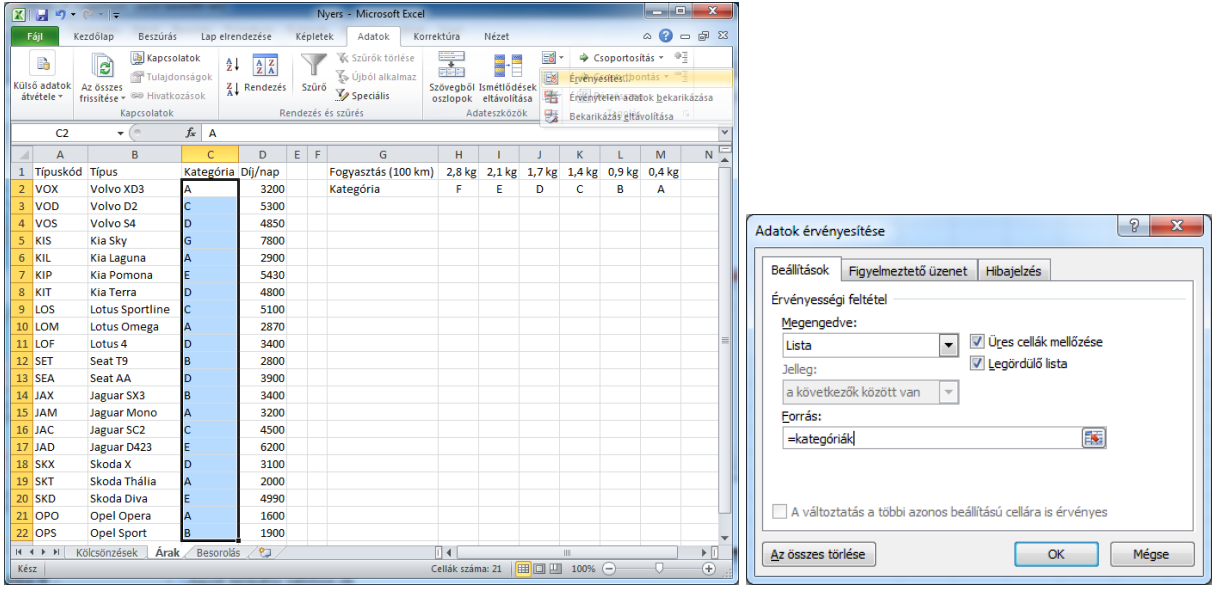
Az adatbázis-kezelők klasszikus szolgáltatása, hogy a felhasználó az adatbázis egy-egy mezőjébe csak olyan adatokat vihet be, amelyek megfelelnek azoknak a korlátoknak, amiket az adott mező adataihoz az adatbázis létrehozásakor rendeltek. Ennek az a következménye, hogy a bevitt adat formailag hibátlan lesz. Táblázatkezelőkben adatok ellenőrzése utólag is és bevitel közben is elvégezhető. Erre szolgál az érvényességellenőrzés és az érvényesítés.

A táblázatban lévő adatok ellenőrzését a követendő lépésekkel végezhetjük el:

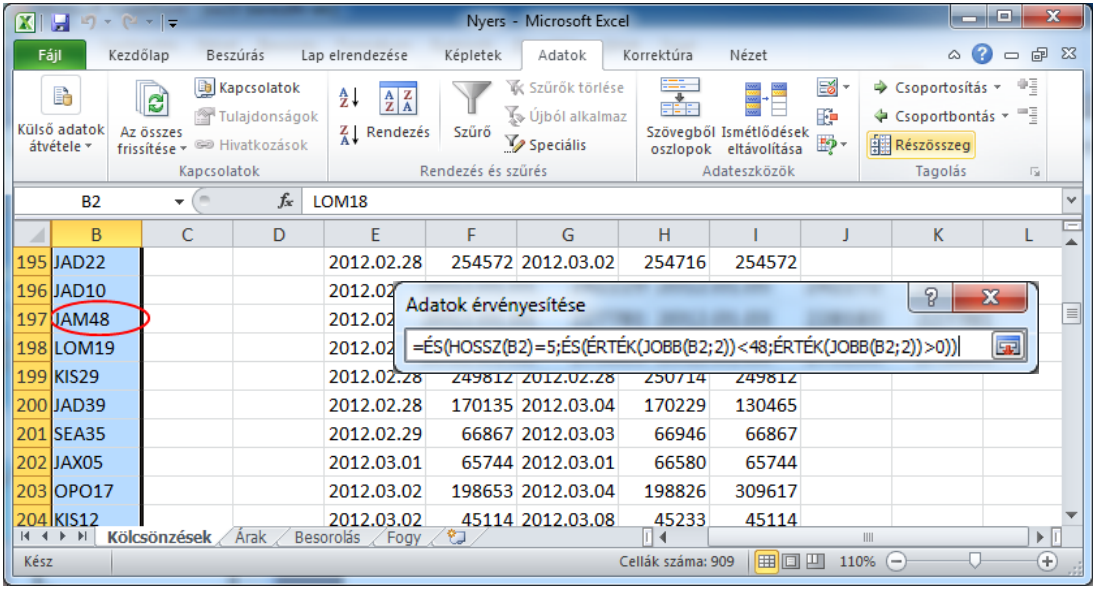
1. Elkészítjük azt a forrástartományt, amely a megengedett adatokat tartalmazza. Törekedjünk arra, hogy ebben ne legyen hiba!
2. Kijelöljük azt a tartományt, ahova beírták az érvényesítendő adatokat. Ezek között lehetnek hibásak is.
3. Végrehajtjuk az érvényességellenőrzést. (Az ellenőrzött celláknál valamilyen módon jelölve lesz a hiba.).
4. Kijavítjuk a hibás adatokat. (Ha a javítás is hibás lenne, nem engedi az átírást a program.)

Az is előfordulhat, hogy a feltételek olyan jellegűek, hogy a megengedhető értékek listában nem sorolhatóak fel, illetve ez túl nehézkes vagy hosszadalmas lenne. Ilyenkor egyéni feltételt kell megfogalmaznunk, és ez alapján végezhető el az érvényesítés. Az egyéni feltételek megadásában minden olyan eszköz felhasználható amivel feltételt lehet megfogalmazni.

Az érvényesítés funkció Excel 2003-ban és Calc-ban az **Adatok/Érvényesítés...**, illetve az **Adatok/Érvényesség...** pont alatt érhető el, Excel 2010-ben pedig az **Adatok** menü **Adateszközök** csoportjában.



34.19. ábra. *Adatérvényesítés-példa – listából Excel 2010-ben*



34.20. ábra. *Adatérvényesítés-példa – egyéni feltétellel Excel 2010-ben*

Önellenőrzés

1. Egy táblázat termékek adatait tartalmazza. Készítsünk szűrőfeltételt a terméknév oszlophoz, aminek a segítségével azokat a termékeket lehet megjeleníteni, ahol a terméknév utolsó előtti betűje „b”!
   Válasz: ☐

2. Az előző feladathoz készítsünk újabb szűrőfeltételt (terméknév oszlop), aminek a segítségével azokat a termékeket lehet megjeleníteni, ahol a terméknév utolsó betűje „b”!
   Válasz: ☐

3. Egy táblázat személyek adatait tartalmazza. Készítsünk szűrőfeltételt a név oszlophoz, aminek a segítségével azokat a személyneveket lehet megjeleníteni, ahol a vezetéknév hossza legalább három karakter, és a keresztnév A-val kezdődik! (Feltehetjük, hogy nincsenek dupla vezetéknevek.)
   Válasz: ☐

4. Válasszuk ki a következő lehetőségek közül azt/azokat, amely(ek) megfelel(nek) egy videotéka kölcsönzési táblázatában az alábbi kritériumnak!

Lakcím	Lakcím	Nap
Szombathely		
Sopron *		
	Tatabánya	>2

- ☐ A több mint kétnapos tatabányai lakcímű kölcsönzések, valamint a szombathelyi lakcímű kölcsönzések.
- ☐ Az összes szombathelyi, soproni vagy tatabányai kölcsönzés.
- ☐ A minimum kétnapos tatabányai lakcímű kölcsönzések, valamint a szombathelyi vagy soproni lakcímű kölcsönzések.

☐ A kevesebb mint kétnapos tatabányai lakcímű kölcsönzések, valamint a szombathelyi vagy soproni lakcímű kölcsönzések.

☐ A több mint kétnapos tatabányai lakcímű kölcsönzések, valamint a szombathelyi vagy soproni lakcímű kölcsönzések.

5. Kimutatást szeretnénk készíteni az alábbi táblázat alapján.
Arra vagyunk kíváncsiak, hogy egy-egy osztályzat hányszor fordul elő közgazdaságtanból. (A kimutatást az alapbeállítások szerint végezzük el, az értékmező összegzésének alapját nem állítjuk át.)
Adjuk meg, hogy hova húzzuk az egyes mezőket, ha az osztályzatelemeket egymás mellett szeretnénk látni?
Lehetőségek: jelentésszűrő, oszlopcímkék, sorcímkék, értékek (Excel 2010 alapon).

	A	B	C	D
1	Neptun	Matematika	Informatika	Közgazdaságtan
2	A2X0AJ	3	2	3
3	B6426B	5	2	5
4	C64S52	4	1	4
5	DLM5VV	4	4	4
6	ENMJ8G	5	4	3
7	FTX2YX	5	2	1
8	FUNU3I	3	5	2
9	G28GLO	2	5	4
10	G44OT4	3	3	2
11	HJSB91	1	2	5

Válasz:
Közgazdaságtan – ☐
Neptun – ☐

17. LECKE

Formázások, diagramok, nyomtatás

A lecke néhány rövidebb témakört tárgyal (formázások, diagramok, nyomtatás). A végén két összetett gyakorló feladat található. Az első megoldással, a második csak válaszokkal.

35. Táblázatok formázása

Egy-egy feladat befejezése után célszerű a kész táblázatot áttekinthetővé és barátságossá tenni. Ezt a célt úgy érhetjük el, hogy a fontosnak tartott részeket kiemeljük, a felesleges vagy zavaró cellákat elrejtjük, esetleg az adatok megjelenítési módját megváltoztatjuk. Ezeknek a feladatoknak az elvégzésére a táblázatkezelőkben megtaláljuk az eszközöket.

35.1. Elrejtés és felfedés

Ha a táblázat egyes részeire csak a számítások végrehajtásához volt szükség, de a felhasználó számára érdektelen információt tartalmaznak, akkor elrejtendők. Ez elvégezhető az Excel 2003- és Calc-ban a: **Formátum** menü **Oszlop** illetve **Sor** pontja, **Elrejtés** parancs; Excel 2010: **Kezdőlap/Cellák/Formátum/Láthatóság/Elrejtés és felfedés** pont, és itt **Sorok elrejtése**, ill. **Oszlopok elrejtése** menüpontokkal. Ezáltal a kijelölt cellákat – ha nincs kijelölés, akkor az aktuális cellát – tartalmazó oszlopok illetve sorok láthatatlanná válnak. Ha utólag mégis láthatóvá kell tenni a értéküket, akkor ez a **Felfedés** sor vagy oszlop; Calc-ban: **Megjelenítés** paranccsal hajtható végre. A felfedés előtt ki kell jelölni egy kisebb blokkot, amely legalább egy-egy cellát tartalmaz mindkét szomszédos– nem elrejtett – oszlopból illetve sorból.

Több munkalap használata esetén előfordulhat, hogy egész munkalapo(ka)t célszerű elrejteni. Ezt is támogatják a táblázatkezelők. Excel 2003: **Formátum/Lap/Elrejtés**; Calc: **Formátum/Munkalap/Elrejtés**; Excel 2010: **Kezdőlap/Cellák/Formátum/Láthatóság/Elrejtés és felfedés/Munkalap elrejtése**.

Ha a lapot újra láthatóvá akarjuk tenni, akkor a parancs végrehajtásakor a táblázatkezelő a megjelenő listában megmutatja az elrejtett lapokat, és ezek közül kell kiválasztani a megfelelőt (a funkció az egyes táblázatkezelőkben az előbbi helyen indítható, a felfedő parancsokkal; Excel 2003: **Lap/Felfedés**; Calc: **Munkalap/Megjelenítés**; Excel 2010: **Elrejtés és felfedés/Munkalap megjelenítése**). Ha nincs elrejtett lap, akkor ez a funkció nem hívható.

35.2. Az adatok megjelenésének formátuma

A táblázatkezelők alapértelmezés szerinti megjelenítésben is megkülönböztetik ugyan a különböző típusú adatokat (például igazítással), de sok esetben az egységesen megjelenített azonos típusú adatok további kategorizálása is szükséges lehet (például a „normál” szám típusú adatoktól Ft végződéssel szeretnénk megkülönböztetni a szintén számokból álló pénzügyi adatokat, vagy „darab” végződést állítunk be egyes számokat tartalmazó cellákra). A kívánt beállításokat a megfelelő blokk kijelölése után Excel 2003-ban és Calc-ban a **Formátum** menü **Cellák...** pontja hatására megjelenő ablak **Szám(ok)** részablakában végezhetjük el, Excel 2010-ben pedig a **Kezdőlap** menüszalag **Szám** csoportjából aktiválható **Szám** részablakban.

Az ablak bal oldalán találhatók a kategóriák, a jobb oldalán pedig az aktuális kategórián belül választható alesetek.

A fontosabb beállítások néhány kategória esetén a következők:

1. **szám**: tizedes nélkül; tizedessel; ezres csoportokra bontva; negatív számok pirosak;
2. **pénznem**: ugyanazok a lehetőségek, mint az előbb, csak Ft végződéssel;
3. **dátum**: év, hónap és nap több különböző sorrendben; a három közül csak kettő látszik; évek csak utolsó két jeggyel; hónapok szöveggel (Calc-ban a szeparátorjel is beállítható);
4. **idő**: óra, perc és másodperc közül csak kettő látszik; az óra 0 és 23 között; az óra 1 és 12 között; délelőtt/délután jelzéssel;
5. **tudományos**: a szám alap*10^kitevő alakban adott, az alap tizedeseinek száma beállítható;
6. **különleges**: egyes speciális formátumok választhatók (pl. adószám, telefonszám stb.);
7. **százalék**: tizedes nélkül; tizedessel.

Az **egyéni** beállításnál a táblázatkezelő szabványos formátumkódjainak felhasználásával saját formátumokat készíthetünk. A lehetőségeket érdemes alaposan áttanulmányozni (súgó), a jellegzetes típusokat (helyiértékek formázása, szövegek megjelenítése, színbeállítás, ezres megjelenítés, valódi tört, tudományos számforma, feltételes formázás, dátum és idő formátumok) mi is bemutatjuk (figyeljük meg, hogy az alapadat, a cella tartalma ugyanaz, a megjelenített érték mégis meglepően más és más lehet).

Formátumkód	Szám	Formázott szám
0	12,34	12
0,00	12,34	12,34
0%	12,34	1234%
0,00E+00	12,34	1,23E+01
0,00	0,1234	0,12
#,00	0,1234	,12
#" "?/? (Excel); # ?/? (Calc)	12,34	12 1/3
# ##0,00 Ft;[Piros]-# ##0,00 Ft (Excel); # ##0,00 [$Ft-40E];[RED]-# ##0,00 [$Ft-40E] (Calc)	1234,56	-1 234,56 Ft
éééé.hh.nn (Excel); YYYY-MM-DD (Calc)	12,34	1900.01.12 1900-01-11
nn.hhh.éé (Excel); DD-MMM-YY (Calc)	12,34	12.jan.00 11-I.-00
hhhh (Excel); MMMM (Calc)	12,34	január
nnnn (Excel); DDDD (Calc)	12,34	csütörtök
ó:pp:mm AM/PM (Excel); H:MM:SS (Calc)	12,34	8:09:36 de. 8:09:36 DE
[ó]:pp:mm (Excel); [H]:MM:SS (Calc)	12,34	296:09:36
0" db"	12,34	12 db
0 " db"	12,34	0 db

35.1. ábra. *Az egyéniformátum-beállítás néhány lehetősége Excel és Calc használatakor*

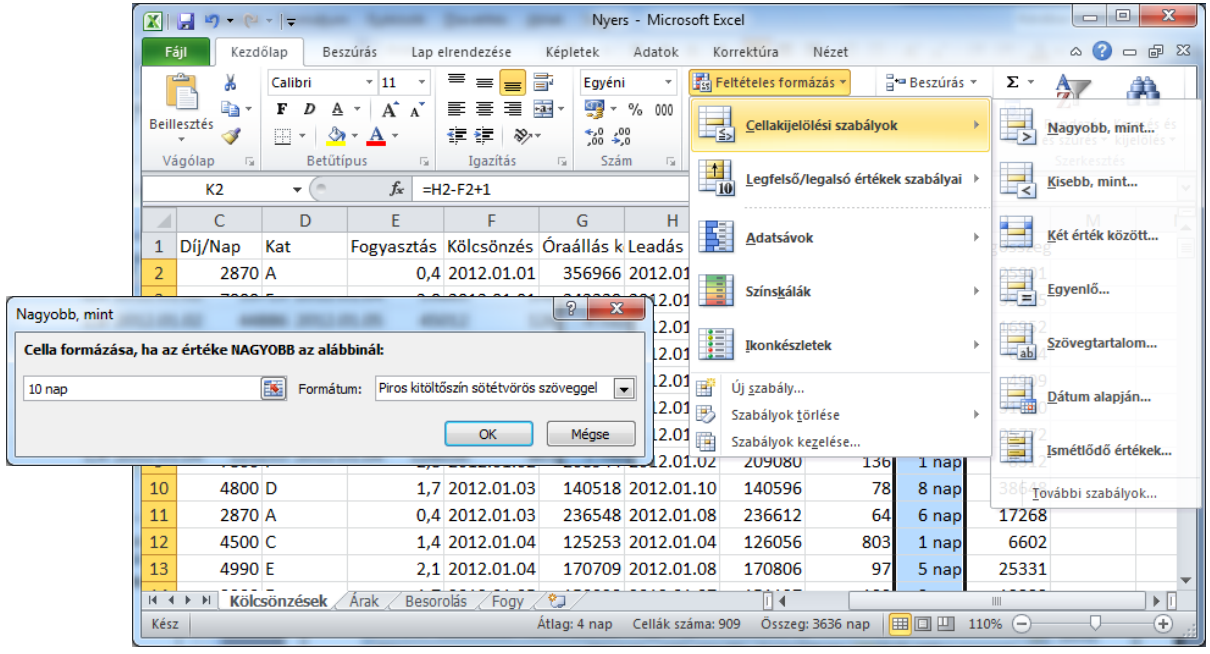
Külön figyelnünk kell arra, hogy míg az Excel magyar rövidítéseket használ a formátumkódoknál, a Calc-ban meghagyták az angol változatokat!

A formátumot – a táblázatkezelő beállításától függően – általában a **Formázó/Formázás** eszköztár megfelelő gombjaival is átállíthatjuk (csak Excel 2003 és Calc). Ezek a gombok a következők lehetnek: **pénznem** (Ft végződés két tizedessel), **százalék** (két tizedessel), **ezres csoportok**, **tizedeshelyek számának növelése** és **csökkentése**.

Az üres cella formátuma is beállítható, és később, ha kitöltjük, akkor az új tartalom értéke automatikusan a beállításnak megfelelő módon jelenik meg.

A dátum és idő típusú adatok formátumát alapértelmezésben a területi beállítások alapján kezeli a táblázatkezelő. Ezek a beállítások – operációs rendszer szinten – a Vezérlőpulton belül a **Területi és nyelvi beállítások**, ill. **Óra, nyelv és terület** kiválasztásával tekinthetők meg. Ha szükséges, a megfelelő pontok módosításával változtathatunk a beállításokon.

Speciális, a gyakorlatban sokszor igen hasznos beállításokat tesz lehetővé a feltételes formázás alkalmazása. Itt eltérő színnel vagy háttérrel emelhetjük ki az adatok közül azokat, amelyek bizonyos feltételeknek eleget tesznek. A megvalósítás az autoszűréshez hasonló, részletesen nem tárgyaljuk. A lehetőségek Excel 2003-ban és Calc-ban a **Formátum** menü **Feltételes formázás** pontjában, Excel 2010-ben pedig a **Kezdőlap** menüszalag **Stílusok** csoportjának **Feltételes formázás** pontjában érhetők el.



35.2. ábra. *Feltételes formázás Excel 2010-ben*

35.3. Méretváltoztatások

A sorok/oszlopok méretét Excel 2003-ban és Calc-ban a **Formátum** menü **Sor/Oszlop**, Excel 2010-ben a **Kezdőlap/Cellák/Formátum/Cellaméret** pontjával állíthatjuk be. A **Magasság. . . /Szélesség. . .** ill. **Sormagasság. . . /Oszlopszélesség. . .** ablakában kell megadni az új méretet. A változtatások a kijelölt cellákat, ha nincs kijelölés, akkor az aktuális cellát tartalmazó sorokra/oszlopokra lesznek érvényesek. Az alapértelmezett méret visszaállítható a **Normál magasság/Normál szélesség. . .** (Excel 2003), ill.

Automatikus sormagasság (Excel 2010) választásával. Calc-ban optimális magasság és szélesség állítható be.

Egy-egy sor/oszlop mérete egérrel menü nélkül is átállítható. Ehhez vigyük az egérkurzort a megfelelő sort/oszlopot a következő sortól/oszloptól elválasztó osztásvonalra a sor- vagy oszlopazonosítón. Ekkor az egérkurzor kétfelé mutató nyíllá változik. Fogjuk meg az egérrel az osztásvonalat és kezdjük el húzni az a megfelelő irányba. A szerkesztőléc név mezőjében a gép kijelzi az aktuális méretet, amely így pontosan beállítható.

Ha az osztásvonalra kétszer kattintunk rá, akkor a táblázatkezelő automatikusan úgy állítja be a sormagasságot vagy oszlopszélességet, hogy az aktuális kijelzés szerint a legtöbb helyet elfoglaló érték éppen elférjen.

35.4. Igazítás a cellaterületen belül

Az adatok a kijelölt cellák területén belül többféle módon igazíthatók, a fontosabb lehetőségek:

1. vízszintesen: alapértelmezett; balra; jobbra; középre; sorkizárt módon; a cellaterületet teljesen kitöltve;
2. függőlegesen: fent; lent; középen.

Ezen kívül beállíthatjuk az írás irányát is, pl. akár felülről lefelé. Ez különösen táblázat fejlécénél vagy kimutatás oldalmezőjében lehet hasznos. A szöveg elhelyezését kérhetjük a **Sortöréssel több sorba** (Calc-ban: Elosztott). A **Cellák egyesítésével** egy-egy cellát képezhetünk több cellából úgy, hogy a cella mérete az összevont cellák összméretével lesz egyenlő. Az egyesített cellára hivatkozni az egyesített cellablokk bal felső cellájának címével lehet.

Az összes említett beállítás a **Formátum/Cellák...** (Excel 2003 és Calc) ill. a **Kezdőlap/Cellák/Formátum/Cellák formázása** (Excel 2010) ponton belüli **Igazítás** részablakban végezhető el. Néhány fontosabb igazítást – az táblázatkezelők beállításától függően – általában a **Formázó** eszköztár megfelelő gombjaival is végrehajthatunk (csak Excel 2003 és Calc). A megfelelő gombok a következők lehetnek: **igazítás balra**, **középre**, **jobbra**, **cellaegyesítés**.

35.5. Karakterformázás

Az adatokat leíró karakterek színe, mérete, betűtípusa és betűstílusa a kijelölt cellákon belül karakterformázással állítható át, Excel 2003-ban és Calc-ban a **Formátum** menü **Cellák...**, Excel 2010-ben pedig a **Kezdőlap/Cellák/Formátum/Cellák formázása** ponton belüli **Betűtípus** (Calc: **Betűkészlet**) részablakban. A részablak lényegében megegyezik a szövegszerkesztők hasonló ablakával, kezelni is hasonlóan lehet.

megjegyzés: Bizonyos beállítások csak a cella értékének egészére, mások az érték részeire is megadhatók, sőt, bizonyos beállítások megadhatósága függ attól is, hogy mi a cella tartalma! (Például felsőindex.)

A formázás fontosabb funkciói – a táblázatkezelő beállításától függően – általában a **Formázó** eszköztáron is megtalálhatók (csak Excel 2003 és Calc): betűtípus illetve betűméret váltás, félkövér, dőlt és aláhúzott megjelenítés (az utóbbi háromból egyszerre több is bekapcsolható), betűszín.

35.6. Cellák színezése, mintázata és bekeretezése

Nemcsak az adatok megjelenési formája, hanem a celláké is változtatható. A cellákhoz háttérszínt és háttérmintázatot adhatunk meg a **Formátum/Cellák...** ill. **Kezdőlap/Cellák/Formátum/Cellák formázása** ponton belüli megfelelő részablakban (Excel 2003: **Mintázat**; Excel 2010: **Kitöltés**; Calc: **Háttér**). A cellák színe – a táblázatkezelő beállításától függően – a **Formázó** eszköztárról is átállítható, a **Betűszín** mellett elhelyezkedő **Kitöltőszín** (**Háttérszín**) gombbal (csak Excel 2003 és Calc). Az átállítás vagy beállítás az aktuális cellára vagy a kijelölt blokkra lesz érvényes.

A táblázat egyes részeit ki lehet emelni rácsozással, bekeretezéssel is. A **Formátum/Cellák...** illetve **Kezdőlap/Cellák/Formátum/Cellák formázása** menüponthoz tartozó **Szegély** (Calc: **Szegélyek**) részablakban adhatjuk meg a keret elhelyezkedését, vonalstílusát és színét. A keretezés – valamivel szűkösebb lehetőségekkel – a **Formázó** eszköztárról is végrehajtható (**Szegélyek** ikon; csak Excel 2003- és Calc-ban).

35.7. Rajzok és szövegdobozok

A munkalapokra rajzokat, képeket, magyarázó szövegdobozokat is beszúrhatunk, amiket aztán arra a területre mozgathatunk át, amit a legmegfelelőbbnek tartunk. A rajzokat egyrészt a táblázatkezelők saját rajzolóprogramjával készíthetjük el. Indítás vagy aktiválás: Excel 2003-ban és Calc-ban a **Szokásos** eszköztár **Rajz**, illetve **Rajz eszköztár megjelenítése** ikonnal; Excel 2010-ben pedig a **Beszúrás** menüszalag **Ábrák** csoportjából; a kezelés megegyezik pl. a Word rajzolójáéval.

Más részüket objektumként szúrhatjuk be, illetve készíthetjük el, lehetőségek: **Beszúrás/Objektum**.

Harmadrészt a kész képeket, rajzokat fájlból Excel 2003-ban és Calcban a **Beszúrás/Kép** menüponttal tehetjük a táblázat munkalapjára. Ugyanezt Excel 2010-ben a **Beszúrás** menüszalag **Ábrák** csoportjával végezhetjük el.

35.3. ábra. *Egyszerű formázott táblázat Excel 2003-mal*

A magyarázó szövegdobozokat, amikkel már az előző ábrákon is találkozhattunk, szintén beszúrással tehetjük a munkalapra. Ezeket Excel 2003-ban és Calcban a **Beszúrás/Kép/Alakzatok** menüponttal lehet beszúrni.

36. Diagramok

A táblázatok adatait grafikusan is megjeleníthetjük, szemléletesen bemutatva ezzel egymáshoz viszonyított nagyságukat és esetleg a változásukat is. Erre a táblázatkezelőkben a diagramkészítőt használjuk.

Először célszerű kijelölni az ábrázolni kívánt adatokat. Ehhez olyan táblázat vagy táblázatrész szükséges, amely legalább egy – numerikus értékekből álló – adatoszlopot vagy adatsort tartalmaz. Nem összefüggő tartomány esetén a szükséges résztartományokat (ezek általában a táblázat egy-egy sorában vagy oszlopában találhatók) egymás után kell kijelölni. Figyeljünk arra, hogyha az szükséges, a kijelölés tartalmazza a mezőneveket is (ha vannak ilyenek). Az adatok neveit tartalmazó sor vagy oszlop a megadásnál előzze meg a többit.

A kijelölés után a diagram elkészítéséhez a Diagramvarázslót/Diagramtündért kell elindítani. Ezt megtehetjük menüből (Excel 2003 és Calc: **Beszúrás/Diagram...**; Excel 2010: **Beszúrás** menü **Diagramok** csoport), valamint a **Szokásos** eszköztár **Diagramvarázsló,** ill. **Diagram** gombjával (csak Excel 2003 és Calc).

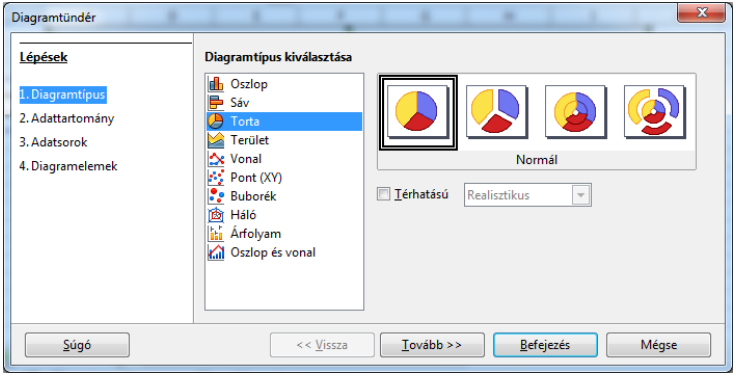
Ezután megjelenik a Diagramvarázsló/Diagramtündér ablaka. Bármelyik táblázatkezelőben is dolgozunk, az első lépésben egy grafikus listát láthatunk a használható grafikonokról. Válasszuk ki a megfelelő diagramtípust a feladat, illetve az adatok jellegének megfelelően:

1. Több numerikus adatsor/oszlop esetén nem célszerű a kör- és tortadiagram használata, mert ezek csak egy számsorozatot képesek megjeleníteni (a hasonló perecdiagrammal több számsorozat is ábrázolható).
2. Viszonylag kevés adat esetén ne olyan diagramtípust válasszunk, amely az egymás utáni adatokat összekötve jeleníti meg, illetve amelynél az egyes adatok önmagukban kevéssé feltűnőek. Nem ajánlott például a terület- és a pontdiagram, ajánlott viszont a sáv-, a kör-, a perec- (Calc-ban: fánk), a torta és az oszlopdiagram. Ez utóbbiakkal az adatok egymáshoz való viszonyát is jól lehet érzékeltetni. (Például egy szavazás végeredménye kördiagramon nagyon szemléletesen bemutatható.)
3. Ugyancsak nem célszerű összekötéses diagramot választani akkor, ha az adatokat csak össze szeretnénk hasonlítani, azaz az adatok nem változást jelenítenek meg.
4. Fordított esetekben (nagy adatmennyiség, változások bemutatása) esetén célszerű a terület- vagy pontdiagramot választani.

5. Szemléltetési célokra különösen illusztratív a 3D diagramok használata.

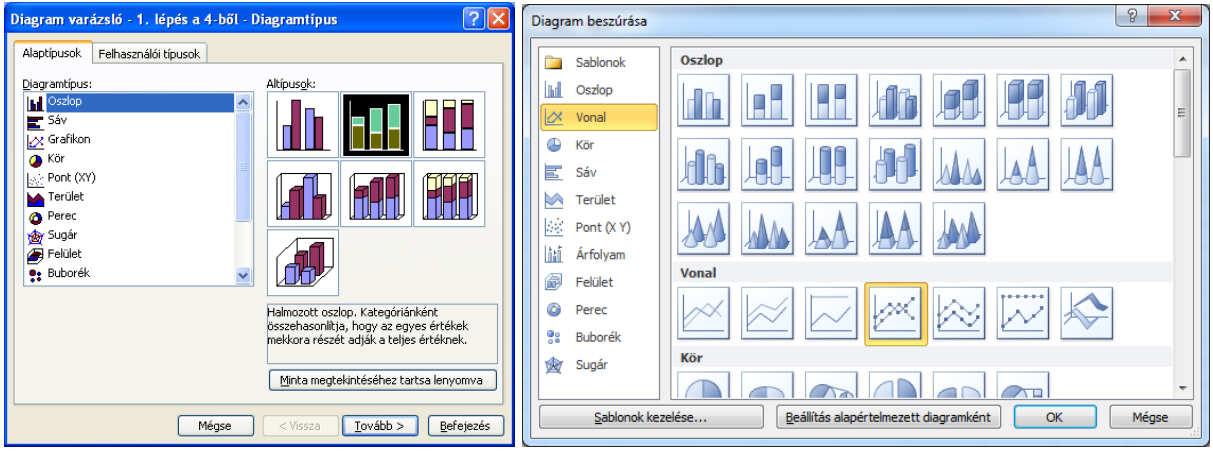
A kívánt típus kiválasztása után még az altípust is meg kell adni (az ablak jobb oldali részében). Az egyes táblázatkezelőkben itt már apróbb eltérések vannak a megjeleníthető altípusokban. Néhány fontosabb beállítási lehetőség az altípusoknál a következő:

1. Kör-, perec- (fánk) és tortadiagram esetén az egyes cikkelyek mellett kérhetjük egy másik adatsorozat (célszerűen nevek) feltüntetését, az egyes cikkelyek elmozdíthatók, százalékos részesedésük megjeleníthető.
2. Kör-, perec- (fánk) és tortadiagram kivételével az összes többi esetben kérhetjük tengelyek és osztásvonalak megjelenítését.
3. Sáv- és oszlopdiagram esetén az adatsorozatok nemcsak egymás mellé, hanem egymásra rakva is ábrázolhatók.
4. Grafikon vagy pont típus esetén kérhetjük az adatok összekötését.



36.1. ábra. *Diagramtípus kiválasztása Calc-ban*

A táblázatkezelők már itt lehetőséget biztosítanak arra, hogy megnézhessük, hogy hogyan fog kinézni a kész diagram a típus/altípus választásunknak megfelelően. Excel 2003-ban ehhez a **Minta megtekintéséhez tartsa lenyomva** gombot használjuk, a Calc a választásunk után a háttérben megjeleníti a diagramot, Excel 2010-ben pedig az **OK** gomb lenyomásával kilépve a diagram beszúrása ablakból a program beszúrja a diagramot).



36.2. ábra. *Diagramvarázsló az Excelben (2003 és 2010)*

Ezután az adatok megadása pontosítható, illetve a diagram formázási beállításai módosíthatók. Excel 2010-ben mindezt a már beszúrt diagramra kattintva tehetjük meg.

A **Tovább** > gombra kattintva – Excel 2003-ban a második, Calc-ban a második és harmadik lépésben – a forrásadatok megadása pontosítható. Ellenőrizzük, hogy az adatok megadása pontos-e, és a táblázatkezelő megfelelően ismerte-e fel a kijelölt adatok logikáját (adatsorok vagy oszlopok). Ha esetleg valamit rosszul adtunk meg (vagy elfelejtettük beírni), akkor most még javíthatunk. A gép a kijelölt terület hivatkozásait abszolút címekkel írja be a megfelelő mezőkbe. Az adatsorok vagy oszlopok elválasztására a pontosvessző szolgál.

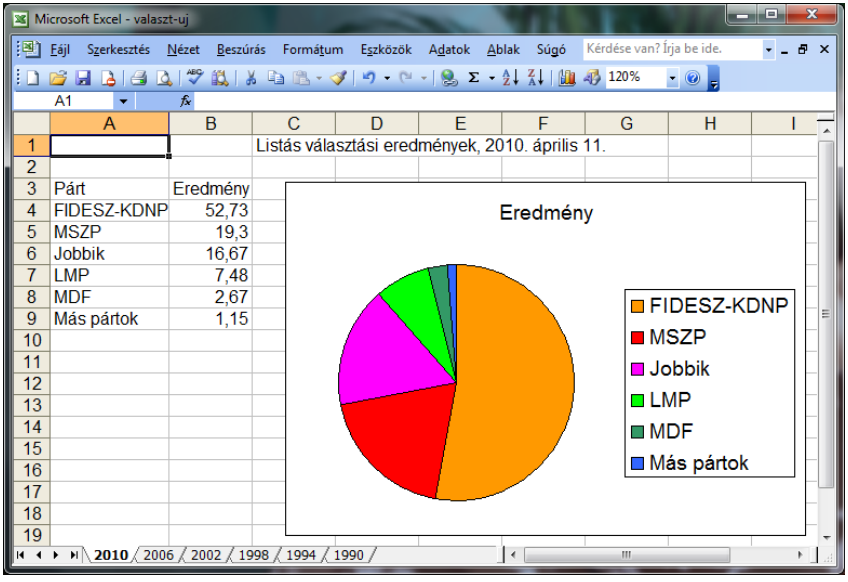
Excel 2003-ban a harmadik, Calc-ban a negyedik lépésben címek és feliratok helyezhetők el az ábrán, ill. a jelmagyarázat beállításai pontosíthatók. Adhatunk címet vagy nevet a diagramnak és a tengelyeknek, címkeként és feliratként megjeleníthetünk értékeket, kategóriákat. A tengelyeknél rácsvonalak adhatók meg, és kérhetjük az ábrán az adattábla megjelenítését is. Természetesen a párbeszédablak elemei típustól függően változnak, hiszen nem minden lehetőség kiválasztásának van értelme minden esetben. A jelmagyarázatnál beállíthatjuk még a láthatóságot és az elhelyezést is.

Excel 2003-ban a negyedik (utolsó) lépés a diagram helyének a meghatározására szolgál. A diagram új lapként és az aktuális lapra egyaránt beszúrható. Calc-ban ez a művelet úgy hajtható végre, hogy a táblázatkezelő által elhelyezett diagramot arrébb „vonszoljuk”.

A Diagramvarázslóval vagy Diagramtündérrel a lépéseken nemcsak előre, hanem – hiba vagy pontatlanság esetén – vissza is léphetünk.

A táblázatkezelőkben diagram a beszúrás után is módosítható (most: Excel 2003 és Calc). A módosítást kezdeményezhetjük Excel 2003-ban és Calc-ban a diagramelemre jobb egérgombbal kattintva, Excel 2003-ban menüből vagy a **Diagram** eszköztár segítségével. Az eszköztár legördülő listájából kiválaszthatjuk a módosítani kívánt diagramelemet.

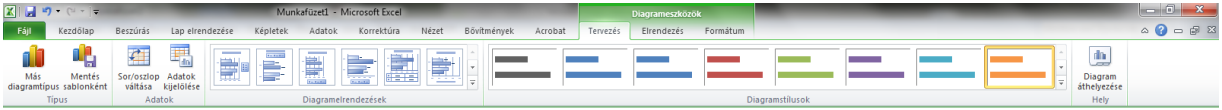
Ha a diagramra kattintunk, akkor az **Adatok** menü **Diagram** menüre változik.amiből a diagramkészítés mind a négy lépése újra hívható, és minden diagramjellemzőt módosíthatunk.



36.3. ábra. *Választási eredmények bemutatása diagrammal*

Excel 2010-ben a használat technikailag eltér az Excel 2003-tól és a Calc-tól. A már beszúrt diagramra jobb egérgombbal kattintva változtatható a forrásadatok megadása (**Adatok kijelölése...** pont – aktiválásával megjelenik az **Adatforrás kiválasztása** ablak), és a címek/feliratok pontosítása, ill. a jelmagyarázat beállítása (szintén az **Adatforrás kiválasztása** ablakban). Ebben a programban a kész diagramot áthelyezhetjük másik munkalapra, a beszúrás után megjelenő **Diagrameszközök/Tervezés** menüszalag **Diagram áthelyezése** pontjával.

A beszúrás utáni módosításokat hasonló módon végezhetjük el mint a másik két programban (változatos lehetőségek a **Diagrameszközök/Tervezés**, **Elrendezés** és **Formátum** menüszalagokon).



36.4. ábra. *A **Diagrameszközök/Tervezés** menüszalag (Excel 2010)*

A diagramok csinosítása, megfelelő külleműre alakítása sokszor igen fontos – és nem feltétlenül egyszerű – része a feladatnak. Ehhez a diagramok elemeinek az elrendezésének gondos megtervezésére van szükség sőt, ez a tervezés sokszor speciális gyakorlatot, tapasztalatot igényel.

Aktivitás: Próbálja ki a következő formai változtatásokat! Ha a vízszintes tengelyen nem jelenik meg minden felirat, akkor kiszélesíthetjük a rajzterületet, vagy a tengely feliratait 90 fokkal elforgathatjuk. Utólag értékeket rakhatunk az adatsorokra vagy oszlopokra, megváltoztathatjuk a színüket, a feliratok színét és betűméretét is. Szükség lehet trendvonalak felvételére és utólagos formázására is.

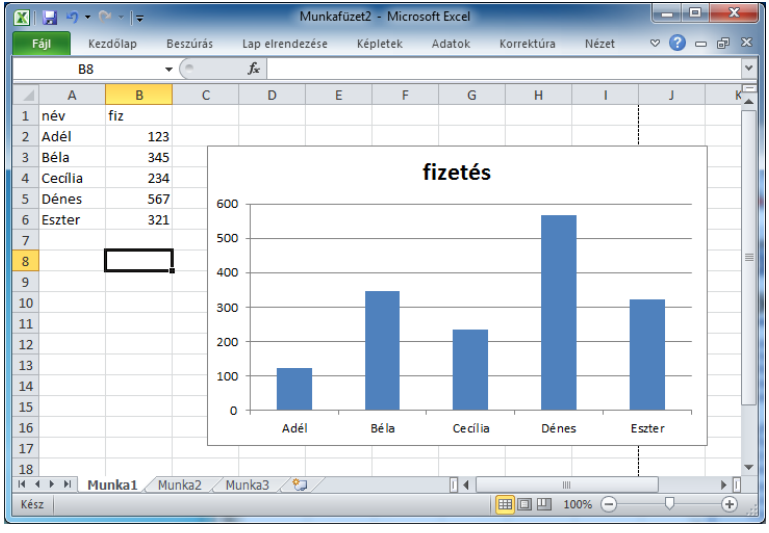
37. Nyomtatás

A táblázat tartalma mindig kinyomtatható. A képernyőn lévő tartalom viszont sokszor nem pontosan ugyanúgy jelenik majd meg a papíron, mint ahogyan a képernyőn látjuk. Előfordulhat például, hogy a nyomtatott lapon a numerikus cellatartalom értéke helyett # jelek sorozatát látjuk. Ezért nyomtatás előtt célszerű megtekinteni, hogyan fog kinézni a munkalap nyomtatott formában. A megtekintés Excel 2003-ban és Calc-ban a **Fájl/Nyomtatási kép** illetve Excel 2010-ben a **Fájl/Nyomtatás** menüponttal vagy (Excel 2003-ban és Calc-ban) a **Szokásos** eszköztár **Nyomtatási kép** ikonjával lehetséges.

Nyomtatás előtt fontos pontosan tudnunk azt is, hogy a táblázat mely része fér rá a papírra vagy lesz látható egy oldalon – nagyon bosszantó lehet például, ha. egy oszlop egy része új oldalra kerül. Hogy ezt megtudjuk, Excel 2003 és 2010-ben aktiváljuk a **Nézet/Oldaltörés megtekintése** illetve Calc-ban a **Nézet/Oldaltörés előnézete** menüpontot. A megjelenő ablakban a dokumentumon kék színű vonallal megjelölve látjuk az alapértelmezett

oldaltöréseket. Az egérrel változtathatunk az oldaltörések helyzetén. Ugyanezt Excel 2003-ban a **Nyomtatási kép** pont alatt is megtehetjük.

Az oldaltörés megtekintése nézetből mindhárom táblázatkezelőben a **Nézet** menü **Normál** menüpontjával térhetünk vissza a normál képernyőnézetre, vagy a **Fájl** menü **Nyomtatási kép** ill. **Nyomtatás** parancsával a nyomtatási kép-nézetre. A nyomtatási képről a normál nézetre visszatérve a táblázatkezelő vonallal határolja körbe azt a területet, ami egy nyomtatott lapra kerül. Ezek a vonalak mindig cellahatárokra esnek.



37.1. ábra. *Laphatár nyomtatásnál Excel 2010-nél*

A nyomtatási kép-nézetben lehetőségünk van az oldal néhány jellemzőjének beállítására is. Módosíthatjuk, megadhatjuk:

1. a papírméretet és az elrendezést;
2. a margók illetve élőfej/élőláb számára biztosított távolságot;
3. az élőfejen/élőlábon belül megjelenő információt, feltüntethetjük az oldalszámot, a dátumot, a fájl nevét, a készítő nevét – ehhez baloldalon, középen és jobbra igazítva összesen 3 pozíciót használhatunk;
4. a cellarácsok, ill. sor- és oszlopazonosítók megjelentetését, a nyomtatás irányát (felülről lefelé vagy balról jobbra), továbbá többoldalas táblázat esetén kérhetjük, hogy az azonosítóként szolgáló mezőnévsorok és rekordnévoszlopok minden kinyomtatott lapokon jelenjenek meg.

Ezek az opciók az egyes táblázatkezelőkön belül különféle módokon aktiválhatók. Excel 2003-ban: **Beállít...** gomb a nyomtatási kép ablakban vagy **Fájl/Oldalbeállítás...**; Excel 2010-ben: **Oldalbeállítás** funkció a Nyomtatási ablakban vagy **Lap elrendezése/Oldalbeállítás** csoport; Calc-ban: **Oldal formátuma** gomb a nyomtatási kép ablakban vagy **Formátum/Oldal...**).

A nyomtatásablakban a fentieken túl megadható még több speciálisan a nyomtatóra jellemző, illetve a nyomtatással kapcsolatos adat, így például a nyomtatás minősége, a papír típusa és az egy lapra nyomtatandó oldalak száma.

38. Mintafeladat

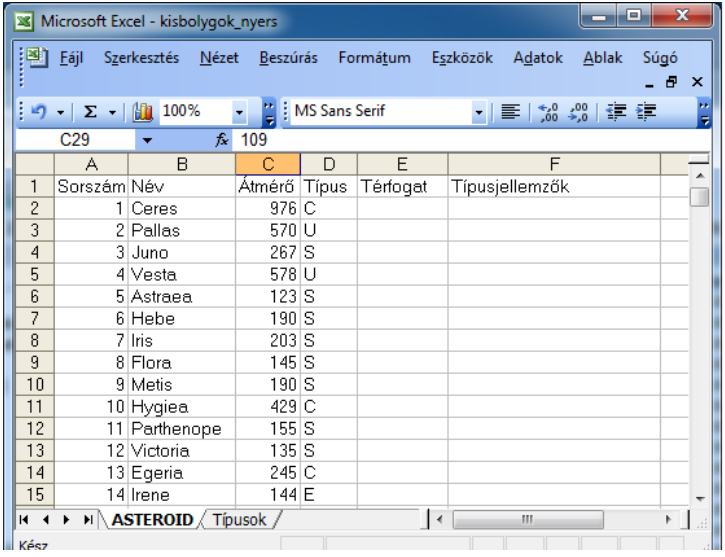
38.1. A feladat leírása (kisbolygók)

A megoldás során a Naprendszer első 100 kisbolygójának néhány adatával dolgozunk, amit az „ASTEROID” munkalapon találunk. Ezek alapján válaszolunk a feltett kérdésekre.

A munkához szükséges kiindulási vagy más szóval nyers xls fájl elérhető ide kattintva.

Az Exceles megoldásokhoz adunk útmutatót, a Calc-ban a szűréses példák megoldása eltér, ez önálló gyakorlásra szánt feladat.

Kövessük a táblázatkezelési feladatok általános megoldásával kapcsolatban tanult lépéseket, és töltsük fel a táblázat hiányzó oszlopait!



	A	B	C	D	E	F
1	Sorszám	Név	Átmérő	Típus	Térfogat	Típusjellemzők
2	1	Ceres	976	C		
3	2	Pallas	570	U		
4	3	Juno	267	S		
5	4	Vesta	578	U		
6	5	Astraea	123	S		
7	6	Hebe	190	S		
8	7	Iris	203	S		
9	8	Flora	145	S		
10	9	Metis	190	S		
11	10	Hygiea	429	C		
12	11	Parthenope	155	S		
13	12	Victoria	135	S		
14	13	Egeria	245	C		
15	14	Irene	144	E		

38.1. ábra. *Kisbolygós feladat – nyers táblázat*

38.2. Útmutató az önálló megoldáshoz

Kiindulásként hozzunk létre egy „Feltétel” nevű munkalapot, és egy másikat a saját monogramunkra átnevezve.

38.2.1. A térfogat meghatározása

A kisbolygók térfogatát az átmérőjük alapján határozzuk meg. A megoldás során az **E** oszlop megfelelő celláit töltjük fel adatokkal. Ha az átmérő nem ismert, akkor ? kerüljön a térfogat cellájába.

Útmutató: A gömb térfogatának képlete: V = 4/3*sugár^3*Pi. Most feltehetjük, hogy a kisbolygók gömb alakúak (bár ez általában nem így van). A nem ismert átmérőjű kisbolygók kezelésére használjuk a megfelelő információs függvényt (Üres).

38.2.2. Típusjellemzők

A „Típusok” munkalap segítségével töltsük fel az „ASTEROID” munkalap „Típusjellemzők” oszlopát. Amennyiben egy adott kisbolygónak nincs típusa megadva, a „nincs adat” szöveget jelenítsük meg a cellatartományban, az előző lépéshez hasonló módszerrel. A feltételesen osztályozott (kérdőjeles pl.: S?) típusú kisbolygókat tekintsük biztosan az adott csoportba tartozónak!

38.2.3. Szűrés

Készítsünk irányított szűrővel egy listát azokról a kisbolygókról, amelyek biztosan vagy valószínűleg C típusúak (ez utóbbiak típusa C?). A listában csak az égitestek neve és típusa jelenjen meg. Az eredmény a monogramos lapra, a szükséges szűrőfeltétel pedig a „Feltétel” munkalapra kerüljön.

38.2.4. Típusstatisztika 1.

Függvénnyel számoltassuk meg, hogy típusonként hány darab kisbolygót találunk a rendelkezésre álló adatok szerint a listában! A kérdőjeles típusú kisbolygókat itt is tekintsük az adott csoportba tartozónak! A szükséges feltételt a „Feltétel” munkalapon adjuk meg, az eredmény tetszőleges munkalapra kerülhet.

Útmutató: Célszerű olyan szűrőtartományokat létrehoznunk, amelyek egymás mellett helyezkednek el, így tudjuk ugyanis majd megfelelő másolással elkészíteni a végleges megoldást.

38.2.5. Típusstatisztika 2.

Készítsünk kimutatást, amely megadja, hogy típusonként hány darab kisbolygót találunk a rendelkezésre álló adatok szerint a listában! A kérdőjeles típusú kisbolygókat itt tekintsük külön kategóriába tartozónak! A nem ismert típusú kisbolygókat vegyük le a listáról (csak Excel 2010-ben végezhető el)!

38.2.6. Diagramkészítés

Ábrázoljuk az előző feladatban kapott eredményeket tortadiagramon, külön munkalapként beszúrva. A grafikon neve legyen: „Kisbolygó statisztika”, a jelmagyarázatban adjuk meg a típusokat.

38.2.7. Formázások

Formázzuk meg a megfelelő oszlopok adatait oly módon, hogy az átmérő km-ben, a térfogat pedig „km^3”-ben jelenjen meg.

38.3. Megoldások

Fontos, hogy ne mechanikusan másoljuk, gépeljük be a megoldást az útmutatóból, hanem törekedjünk arra, hogy megértsük a képletek logikáját. Javasolt, hogy később egyedül is próbáljuk előállítani a helyes képletet, és csak akkor nézzük meg a megadott megoldást, ha úgy gondoljuk, hogy a sajátunk kész!

1. Megoldás (E2 cella, másolható lefelé az E3:E101 tartományban):

```
=HA(ÜRES(C2);"?";4/3*(C2/2)^3*PI())
```

Vagy:

```
=HA(C2;4/3*(C2/2)^3*PI();"?")
```

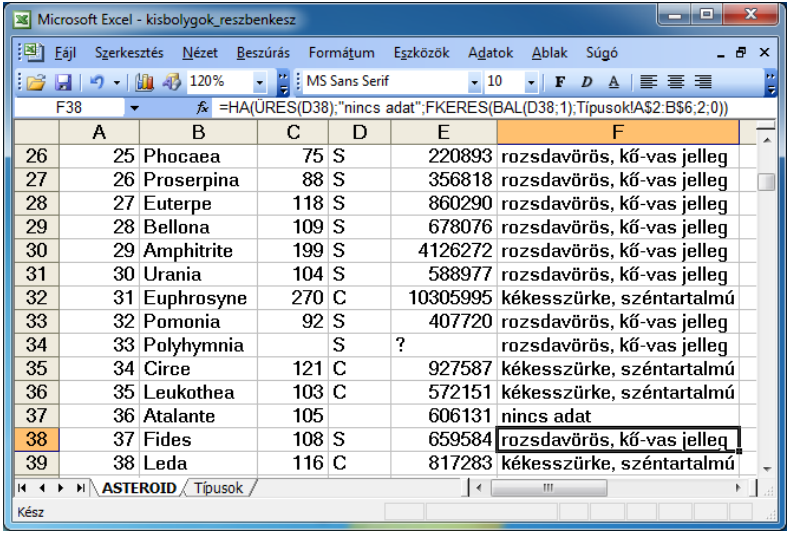
Ennél a feladatnál az Üres függvény alkalmazása helyett a C2=0 logikai képlettel is jó megoldást kapunk, bár ez a megoldás nem használható olyan általánosan, mint amit az Üres függvénnyel kapunk.

2. Megoldás (F2 cella, másolható lefelé az F3:F101 tartományban):

```
=HA(ÜRES(D2);"nincs adat";FKERES(BAL(D2;1);Típusok!A$2:B$6;2;0))
```

Vagy:

```
=HA(NEM(ÜRES(D2));FKERES(BAL(D2;1);Típusok!A$2:B$6;2;0);"nincs adat")
```



	A	B	C	D	E	F
26	25	Phocaea	75	S	220893	rozsdavörös, kő-vas jelleg
27	26	Proserpina	88	S	356818	rozsdavörös, kő-vas jelleg
28	27	Euterpe	118	S	860290	rozsdavörös, kő-vas jelleg
29	28	Bellona	109	S	678076	rozsdavörös, kő-vas jelleg
30	29	Amphitrite	199	S	4126272	rozsdavörös, kő-vas jelleg
31	30	Urania	104	S	588977	rozsdavörös, kő-vas jelleg
32	31	Euphrosyne	270	C	10305995	kékesszürke, széntartalmú
33	32	Pomonia	92	S	407720	rozsdavörös, kő-vas jelleg
34	33	Polyhymnia		S	?	rozsdavörös, kő-vas jelleg
35	34	Circe	121	C	927587	kékesszürke, széntartalmú
36	35	Leukothea	103	C	572151	kékesszürke, széntartalmú
37	36	Atalante	105		606131	nincs adat
38	37	Fides	108	S	659584	rozsdavörös, kő-vas jelleg
39	38	Leda	116	C	817283	kékesszürke, széntartalmú

38.2. ábra. *Kisbolygós feladat – megoldás közben*

3. A megfelelő szűrőtartomány:

Típus
C

4. Másolással létrehozzuk a következő szűrőtartományt pl. a H10:L11 blokkban:

Típus	Típus	Típus	Típus	Típus
C	M	S	U	E

Megoldás a H12 cellára:

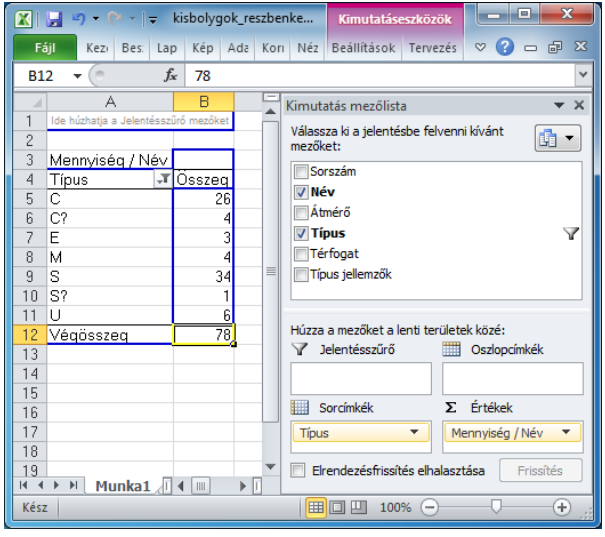
=AB.DARAB2($A1:$D101;4;H10:H11)

A képlet jobbra másolható. Ezt a megoldási ötletet alaposan gondoljuk át!

Az adatbázis megadásánál elég lenne csak a D oszlop is.

Házi feladat, önálló gyakorlásra: az AB.DARAB függvénnyel is elő tudnánk állítani a megoldást?

6. Az alkalmazandó formátumkód: 0” km” (az első beállításhoz).



38.3. ábra. *Kisbolygós feladat – kimutatás Excel 2010-ben*

Önellenőrzés

1. Jelöljük meg az alábbiak közül azokat a számformátumokat, amelyek helytelenek (szintaktikailag hibásak, vagy biztosan hibás választ adnak)!
   - ☐ 0;Normál
   - ☐ -0
   - ☐ 0 "db"
   - ☐ 0" db"
   - ☐ #" db"
2. Egy cellára a 0;[kék]-Normál;"" egyéni formátumkódot alkalmaztuk. Mi igaz a cellára az alábbiak közül?
   - ☐ A cella tartalma nem lehet képlet.
   - ☐ A cella tartalma mindig üres marad.
   - ☐ A pozitív számok helyett 0 jelenik meg.
   - ☐ A pozitív számok egészre kerekítve jelennek meg.
   - ☐ A negatív számok kék betűszínnel jelennek meg.
   - ☐ Ha a cellába írt képlet értéke pozitív szám is lehet, a képlet nem tartalmazhatja a Kerekítés függvényt.

Modulzáró kérdések

Nyissa meg a konyvtar_nyers.xlsx munkafüzetet, és oldja meg az alábbi feladatokat!

3. Határozza meg a dolgozók munkalap alapján, hogy hol lakik Szűcs Enikő!
   Megoldás:

4. Rendezze a Dolgozók munkalap adatait fizetés szerint csökkenő sorrendbe! Kinek van a második legtöbb fizetése?
   Megoldás:

5. Állítsa át az Érvényes munkalap E3:N5 blokkját úgy, hogy csak az Időtartam munkalap első sorában szereplő értékek szerepelhessenek benne! Hány hibás adatot vittek fel eredetileg az Érvényes munkalap adott blokkjába?
   Megoldás:

6. Törölje ki a megadott sorrendben a kódok munkalap I, G oszlopát és az E2:J4 blokkjának a tartalmát! Mennyi a D1:K5 blokk összege?
   Megoldás:

7. Mi annak a cellának a tartalma, aminek a neve banán?

8. Hányszor vettek ki olyan könyvet, melynek az azonosítójában a „WR” karaktersorozat szerepel?
   Megoldás:

9. Hányszor hozták vissza ugyanabban a hónapban a kihozott könyvet, mint amelyikben kivették?
   Megoldás:

10. Hány kölcsönzőnek kezdődik ugyanazzal a karakterrel a keresztneve, mint amivel a vezetékneve végződik?
    Megoldás:

11. Számolja ki, hogy az egyes embereknél ténylegesen hány napig volt a könyv!
    A számolásnál vegye figyelembe, hogyha valaki még aznap visszavitte a könyvet, mint amelyiken kivette,

akkor egy napig volt nála!
Hány napig volt átlagosan azoknál a könyv, akik Csurgón születtek? (egészre kerekítve)?
Megoldás:

12. Számítsa ki a tervezett napok száma oszlop és a dátumok alapján, hogy az egyes emberek hány napot késtek a könyv visszavételével! (Azoknál az embereknél, akik nem késtek a 0 jelenjen meg!)
A számolásnál vegye figyelembe, hogyha valaki még aznap visszavitte a könyvet, mint amelyiken kivette, akkor egy napig volt nála!
Ha az egynapos késéseknél 21 Ft, egyéb esetekben pedig 27 Ft/nap a büntetés, akkor mennyi büntetést fizettek be összesen a könyvet kölcsönzők?
Megoldás:

13. Keresőfüggvény segítségével határozza meg az Időtartam munkalap és a tervezett napok száma alapján, hogy milyen típusúak voltak az egyes kölcsönzések!
A meghatározásnál a következő táblázat szerint járjon el!

Tervezett napok száma		Kölcsönzés típusa
Tól	Ig	
0	1	napos
2	15	félhavi
16	30	havi
31	45	másfél havi
46	60	kéthavi
61	90	háromhavi
91	n	extra

Hány kölcsönzés volt kéthavi vagy háromhavi?
Megoldás:

14. Készítsen kimutatást a születési helyekből! Hol születtek azok, akik a legtöbb napra tervezték a könyvek kivételét összesen?

15. A Kritérium munkalap segítségével készítsen olyan szűrési feltételt, ami azokat a köl-csönzőket listázza ki, akik 75 napnál többre tervezték a könyvek kivételét és a nevük „H” vagy „V” betűvel kezdődik! Mennyi a megjelent ellenőrzőkód értéke?
Megoldás:

39. Fogalomtár

abszolút cellahivatkozás Képletben a cella pontos címét adja meg, függetlenül a képletet tartalmazó cella helyzetétől. Az abszolút cellahivatkozás formája: $A$1.

adattípus Ahhoz, hogy egy megoldandó feladat adatait a számítógép belső világában modellezni tudjuk, a következő tulajdonságok alapján kategorizálást kell végeznünk: *kódolás* (hogyan kell az értékeket a memóriában tárolni), *méret* (mennyi helyet foglal el az adat tárolása), *szerkezet* (összetett típusoknál hivatkozni lehessen az összetevőkre) és *jelentés* (értékhalmaz és az elvégezhető műveletek).

aktív cella Beíráskor az adatok az aktív cellába kerülnek. Ha nincs kijelölve tartomány, az aktív cellát vastag szegély határolja, a kijelölés esetén eltérő háttérszínéről ismerhetjük fel. Egyszerre csak egy cella lehet aktív.

aktív lap A munkafüzet azon lapja, amelyen dolgozunk. Az aktív lap neve félkövéren jelenik meg a hozzá tartozó fülön.

állandó (konstans) Olyan érték, amely nem számítás eredménye, így nem változik meg. Állandó például az 5 számérték és az "alma" érték.

argumentumok A függvény neve után zárójelben felsorolt kifejezések, amelyek értékét a függvény a számítások végrehajtásához használja fel. A függvénnyel használt argumentumok száma és típusa a függvénytől függ.

asszociativitás Az azonos precedenciaszintű operátorok kiértékelési sorrendjét (balról jobbra vagy jobbról balra) határozza meg.

beillesztési terület Az a terület, amelyre a kivágással vagy másolással a vágólapra helyezett adatok be lesznek illesztve.

cella A munkalap egy adott sorának és oszlopának eleme, amely képletet (vagy állandót) és annak értékét, megjelenítési formátumot, jegyzetet valamint adatérvényesítési szabályokat tárol.

cella megjelenített értéke A képernyőn megjelenő érték, amely különböző számformátumok alkalmazásakor eltérhet a tényleges értéktől.

cella tartalma A cellába begépelt karaktersorozat, amely lehet állandó vagy képlet. Az aktív cella tartalma a szerkesztőlécről olvasható le.

cella tényleges értéke Az az érték, amelyet a cella felhasználásakor a táblázatkezelő a számításokhoz alapul vesz.

cellahivatkozás A munkalapcella helyét kijelölő koordinátapár. A B oszlop és a 3. sor találkozásánál lévő cella címe például B3. A munkalap különböző részein elhelyezett adatok értékét a képletekben cellahivatkozások megadásával használhatjuk fel.

cellatartomány Két vagy több azonos lapon lévő cella halmaza. Az adott tartományban lehetnek szomszédos és nem szomszédos cellák is.

cellaterület A cella értékének megjelenítésére felhasznált képernyőterület.

egyesített cella Két vagy több kijelölt cella egyesítésével létrehozott cella. Az egyesített cella hivatkozása megegyezik az eredetileg kijelölt tartomány bal felső cellájának hivatkozásával.

függő cella A képletet tartalmazó cellák közül azokat, amelyeknek az értéke más cellák értékétől függ, függő cellának nevezzük.

függvény Alprogram, amely a paraméterként átvett értékeken elvégezve a benne megfogalmazott számítást valamilyen értéket állít elő.

képlet A táblázatkezelőkben a számításokat többnyire képletek segítségével végezzük el. Az Excelben és a Calc-ban a képletek mindig egyenlőségjellel (=) kezdődnek (bár beírhatók más módon is), amely után egy kifejezés található.

kifejezés Az elvégzendő számítások leírása. A táblázatkezelőkben a kifejezések felépítéséhez függvények, hivatkozások, operátorok (műveleti jelek), állandók, nevek, kifejezések és zárójelek használhatók. A kifejezés kiértékelése precedencia és asszociativitás alapján történik.

kitöltőjel A kijelölés jobb alsó sarkában lévő fekete négyzet, amely adatsorok bevitelére vagy cellák másolására használható. A kitöltőjel felett az egérmutató fekete szálkeresztre változik.

külső hivatkozás Hivatkozás egy másik munkafüzet munkalapján lévő cellára vagy tartományra, illetve egy másik munkafüzetben lévő definiált névre.

másolási terület Azok a cellák, amelyeket az adatok más helyre való beillesztése céljából a vágólapra helyezünk. Másolás után a cellák körül mozgó keret jelenik meg, amely azt jelzi, hogy a vágólapra lettek másolva.

munkafüzet Táblázatkezelő programban készült fájl, amely különböző típusú lapokból (munkalap, diagramlap, makrólap) állhat. A munkafüzetbe adatok vehetők fel, azokon pedig számítások végezhetők vagy diagramon ábrázolhatók.

munkalap (számolótábla) A munkafüzet olyan lapja, amelyet adatok tárolására és kezelésére használunk. A munkalap sorokba és oszlopokba rendezett cellákból áll.

Név mező A szerkesztőléc bal oldalán lévő mező, amely a kijelölt cellát, diagramelemet vagy rajzobjektumot azonosítja. A Név mező segítségével a kijelölt cella vagy tartomány elnevezhető, illetve a korábban elnevezett cellák könnyen megkereshetők és kijelölhetők.

név Olyan karaktersorozat, amely egy cellát, cellatartományt, képletet vagy egy állandó értéket jelöl. A nehezen megjegyezhető tartomány-meghatározások (például Eladások!C20:C30) helyett könnyen érthető neveket célszerű használni (például Termékek).

operandus Az operátorok argumentumait operandusoknak nevezik.

operátor A függvények speciális változatai, melyeket a táblázatkezelőben nem nevekkel, hanem szimbólumokkal – például +, / stb. – azonosítunk. Az Excel operátorai egy- vagy kétargumentumúak. Típus szerint lehetnek számtani, összehasonlító, szöveg- és hivatkozási operátorok.

oszlopazonosító Betűjelzés, amely egy adott oszlopban lévő cellák halmazát jelöli ki.

precedencia (prioritás) Meghatározza, hogy a kifejezésben az operátorokat milyen sorrendben kell végrehajtani. A műveletek végrehajtási sorrendje zárójelek segítségével megváltoztatható.

relatív hivatkozás Valamely cella képletben megadott relatív helyzete a képletet tartalmazó cellához viszonyítva. A képlet másolásakor a relatív hivatkozás automatikusan megváltozik. A relatív hivatkozás formája: A1.

sorazonosító Sorszám, amely egy adott sorban lévő cellák halmazát jelöli ki.

számolótábla lásd munkalap.

szerkesztőléc A menüszalag alatt található sáv, melynek segítségével a cellák tartalmát, illetve a diagramok adatsorainak paramétereit szerkeszthetjük.

típusátalakítás Az adat típusának megváltoztatása, amely történhet implicit (automatikusan) vagy explicit (konverziós függvénnyel) módon.

tömb Adatok csoportja, amely n×m elemű. Akkor használatos, amikor egy képletnek több eredményt kell visszaadnia, illetve ha a függvény sorokban vagy oszlopokban elrendezett argumentumokkal dolgozik.

tömbállandó Állandók olyan csoportja, amely a képletekben argumentumként használható.

tömbképlet Olyan képlet, amely több eredményt képes visszaadni. A tömbképletek kapcsos zárójelek { } között találhatók, bevitelükhöz a CTRL+SHIFT+ENTER billentyűkombinációt kell használni.

40. Irodalomjegyzék

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest: *Algoritmusok*, Műszaki Kiadó, Budapest, 1997.

[2] M. D. Godfrey, D. F. Hendry: *The Computer as von Neumann Planned It, (A számítógép, ahogyan Neumann megtervezte)*, IEEE Annals of the History of Computing, vol. 15, no. 1, 1993, pp. 11–21.

[3] H. H. Goldstine: *A számítógép Pascaltól Neumannig* Műszaki Könyvkiadó, Budapest,1987.

[4] Iványi Antal: *Informatikai algoritmusok*, ELTE Eötvös Kiadó, Budapest, 2004.

[5] Jeff Miller: *Matematikusok postai bélyegeken*, Ponticulus Hungaricus, XVI. évf., 12. szám, 2012. (members.iif.hu/visontay/ponticulus/rovatok/hidverok/matematikusok-belyegeken.html)

[6] John von Neumann: *First Draft of a Report on the EDVAC*, 1945.

[7] Raffai Mária: *Az informatika fél évszázada*, Springer Hungarica Kiadó, Budapest, 1997.

[8] Szíjártó Miklós: *Az informatika alapjai*, Novadat, Győr, 2001.

[9] Wikipedia.org/wiki magyar, angol és más nyelvek oldalai. (Felhívjuk a figyelmet, hogy a wikipédián sokszor egymásnak ellentmondó és változó tartalmú szócikkeket találhatunk. Kritikával olvassuk az ott lévő ismertetőket!)

[10] Tanenbaum, Andrew S.: *Operating systems (Operációs rendszerek)*, ford. Dévényi Károly et al., Budapest, Panem, 2006.

[11] Sikos László: *Bevezetés a Linux használatába*, BBS-Info Kiadó, Budapest, 2005.

[12] Bálint Dezső: *Hálózati ismeretek : alapok, operációs rendszerek, Internet*, 3. átdolg. kiad., Budapest, Talentum Kft., 2000.

[13] Bártfai Barnabás: *Operációs rendszerek : a számítógép használata és a fájlkezelés*, BBS-Info, Budapest, 2010.

[14] Tanenbaum, Andrew S.: *Computer networks (Számítógép-hálózatok)* Budapest, Novotrade Rt., 1995

[15] Drótos László: *Beilleszkedés a hálózat virtuális világába*, Budapest, Nemzeti Információs Infrastruktúra Fejlesztési Program. Koordinációs Iroda, 1995

[16] Bokor Árpád (szerk.): *Hálózatok és rendszerek. Számítógépes gyakorlatok*, Budapest, Műegyetemi K., 2008.

[17] Barbier, Frédéric: *A könyv története.* Budapest, 2005, Osiris Kiadó.

[18] Barbier, Frédéric – Lavenir, Catherine Bertho: *A média története.* Budapest, 2004, Osiris Kiadó.

[19] Funke, Fritz: *Könyvismeret.* Budapest, 2005, Osiris Kiadó.

[20] Gyurgyák János: *Szerzők és szerkesztők kézikönyve.* Budapest, 2005, Osiris Kiadó.

[21] Laczkó Krisztina – Mártonfi Attila: *Helyesírás.* Budapest, 2006, Osiris Kiadó.

[22] Szántó Tibor: *A betűszedés könyve.* Budapest, 1951, Könnyűipari Könyvkiadó.

[23] Virágvölgyi Péter: *A tipográfia mestersége számítógéppel.* Budapest, 1999, Osiris Kiadó.

[24] West, Suzanne: *Stílusgyakorlatok.* Budapest, 1998, Ur Kft.

[25] Kovalcsik Géza: *Az Excel programozása.* Computerbooks, Budapest, 2005.

[26] Kovács András – Kiss Csaba: *Táblázatkezelési ismeretek*, Műszaki könyvkiadó, Budapest, 2000.

[27] Fehérvári Arnold – Kallós Gábor – Kuti József: *Informatika II. – Irodai programcsomag*, Novadat, Győr, 2007.

[28] Pétery Kristóf: *Excel 2010 – Alapok*, Mercator Stúdió, 2011.

[29] Pétery Kristóf: *Excel 2010 – Biblia*, Mercator Stúdió, 2011.

[30] John Walkenbach: *Excel 2010 Formulas*, Wiley Publishing, 2010.

[31] John Walkenbach: *Excel 2010 Bible*, Wiley Publishing, 2010.