# LaTeX kézikönyv

# Tartalomjegyzék

# Táblázatok jegyzéke

# Előszó

A LaTeX egy általános célú dokumentumkészítő rendszer, mely a TeX nevű szedőprogramra épül. Segítségével a legjobb nyomdai minőségben készíthető a rövid feljegyzéstől kezdve a levélen, diplomamunkán, internetes oktatási anyagon, tudományos cikken át egészen a könyvig szinte tetszőleges írásos anyag.

A TeX, s vele együtt a LaTeX különleges program. Különlegessé teszi a számítástechnikában szokatlan idős kora, archaikus felépítése, s ezzel együtt fiatalos alkalmazkodóképessége; minimális hardverigénye, mindemellett nagy tudása; minden számítógéptípuson és minden elterjedt operációs rendszeren való hozzáférhetősége (DOS, Windows 3.11, Windows 9X, NT, XP, OS/2, MacOS, Linux, UNIX, VMS...), ezek szinte tökéletes kompatibilitása; ingyenessége, szabadon elérhető és felhasználható nyílt forráskódja, furcsán írt neve, vagy e név szokatlan kiejtése (tech, latech).

A program leginkább a műszaki és természettudományi egyetemek és kutatóintézetek világában terjedt el. Évente többezer LaTeX-ben írt könyv jelenik meg, a tudományos folyóiratokban, interneten megjelent cikkek, írások, oktatási anyagok és vele írt diplomamunkák száma milliós nagyságrendű. Egyes tudományokban használata de facto szabvánnyá vált, így sokaknak nélkülözhetetlen segédeszköz.

Magyarországon először 1998-ban jelent meg könyv a LaTeX-ről, Wettl Ferenc, Mayer Gyula és Sudár Csaba LaTeX kezdőknek és haladóknak című műve [41]. Az Olvasó ennek, egyes részeiben teljesen átdolgozott, új részeket tartalmazó kiadását tartja most kezében. Leegyszerűsítettük, világosabb szerkezetűvé tettük a LaTeX-hel való első ismerkedésről szóló fejezetet, és feladatokkal bővítettük ki, melyek megoldása megtalálható a mellékelt CD-n. A könyv új részei között több összefoglaló jellegű is van, ilyen például az állománykiterjesztésekről szóló függelék. Ez megerősíti a könyv kézikönyv jellegét – ezért a címváltoztatás.

E könyv széles olvasóközönség számára készült, és sokféle igénynek próbál megfelelni. Reméljük, hogy a kezdő számítógép-felhasználó, a Wordhöz szokott tanszéki titkárnő, a projektfeladatát vagy diplomamunkáját készítő hallgató egyaránt haszonnal forgatja majd. Reményeink szerint e könyv segíteni fog az Olvasónak abban, hogy kevesebb időt kelljen fordítania a számítógépes szerkesztés és szedés problémáinak megoldására, és több figyelmet szentelhessen készülő művére.

Köszönjük családtagjainknak türelmüket és támogatásukat, kollégáinknak, barátainknak és nem utolsósorban a Panem Kiadónak mindazt a segítséget, amellyel hozzájárultak könyvünk elkészültéhez, Maczó Péternek a könyv tipográfiájának megalkotásában végzett munkáját. Külön köszönjük Bujdosó Gyöngyi hasznos szakmai tanácsait, gondos lektori munkáját!

Budapest, 2004. május 1.

Wettl Ferenc (BME) wettl@math.bme.hu
Mayer Gyula (MTA TKI) gam@cs.elte.hu
Szabó Péter (BME) pts@math.bme.hu
lakk@math.bme.hu

1. FEJEZET

# Bevezetés

## 1.1. TEX és LATEX

**TEX, plain TEX** ♦ A TEX dokumentumkészítésre, szedésre szolgáló programrendszer. Megalkotásába 1977 májusában kezdett Donald E. Knuth stanfordi matematikus, miután sokat bajlódott „A számítógép-programozás művészete'' című művének kiadásával [25, 22]. Olyan programot alkotott, mellyel vágyának megfelelően szép megjelenésű művek nyomtathatók, és mellyel az olyan bonyolult feladatok is megoldhatók, mint a matematikai képletek szedése [24].

A TEX program a nyomdászat több évszázados tudását viszi számítógépre úgy, hogy az arra jellemző gondolkodásmód egy részét is megőrzi, és kibővíti azt a számítástechnika adta új lehetőségekkel. A program lényegében az ólomszedést modellezve működik.

Maga a TEX szó a művészet jelentésű görög τέχνη – nagybetűkkel írva TEXNH – szó első három betűjéből áll. E szó kiolvasva „techné'', így a TEX nem „teksz''-nek, hanem „tech''-nek ejtendő. A TEX szó gyönyörű találmány, hisz megbújik benne a *text* (szöveg) és a *technika* szó is. Egy szép külalakot adó, műszaki-matematikai szövegek szedésére különösen alkalmas programnak keresve sem lehet jobb nevet találni. A TEX márkajel középső, ejtett ‚E' betűje a TEX tipográfiai képességeit jelzi. Írógépen vagy egy egyszerű szövegfájlba írva a márkajel `TeX`.

A TEX-rendszernek része a METAFONT nevű program, mely a TEX saját betűkészleteinek létrehozására és újak alkotására is alkalmas [23].

A TEX mára nagyon stabil programmá vált, amelyen már csak ritkán változtatnak. A legutolsó verzió száma 3,14159.[1] A TEX-ben szinte minden tipográfiai feladat megoldható, de néha csak igen nagy fáradsággal. Számítástechnikai hasonlattal élve a TEX a nyomdászat assemblere, melyben több száz elemi parancs segítségével bármely szöveg a kívánt formába önthető. A TEX-ben makrókat is írhatunk, használatukkal a bonyolultabb tipográfiai problémák megoldására is könnyen használható parancsokat készíthetünk. Egy-egy jól összeállított makrócsomag a dokumentumszerkesztés egész folyamatát leegyszerűsítheti.

Több ilyen makrócsomag is készült. Az elsőt maga Knuth készítette el, aminek a „plain TEX'' nevet adta. Ennek alapkönyve ma is a TEXBook [24]. Magyarul Bujdosó Gyöngyi és Fazekas Attila könyvében olvashatunk róla [7] (további magyar nyelvű könyvek: az első magyar TEX-disztribúció kézikönyve [10] és Doob egy művének fordítása [9]). Az első időkben még két további megoldás terjedt el széles körben: az egyik az American Mathematical Society ($\mathcal{AMS}$) által támogatott, Michael Spivak által fejlesztett $\mathcal{AMS}$-TEX [35, 7], a másik Leslie Lamport LATEX programcsomagja [27, 31]. Gyakran találkozni azzal a szóhasználattal, mely a plain TEX helyett TEX-et mond. TEX-en az egész rendszert értjük, melynek a plain TEX és a LATEX is része!

[1] A TEX verziószáma $\pi$-hez konvergál, a METAFONT-é $e$-hez, az utóbbi verziószáma jelenleg 2,7182.

**LaTeX** ♦ Lamport az 1980-as évek elején kezdett a TeX-re épített dokumentumkezelő rendszerének megalkotásába. Ha a TeX-et a nyomdászat assemblerének neveztük, a LaTeX-et egy magas szintű dokumentum-leíró nyelvnek tekinthetjük. A magas szintű azt jelenti, hogy egy-egy LaTeX-parancs mögött igen sok – főként tipográfiai – tudás van, ami lehetővé teszi bonyolult struktúrájú művek egyszerű módon való megírását. A szerzőnek csak saját műve *lejegyzésével* kell törődnie, s a létrejött eredmény nyomdai minőségű lesz.

Elkészült rendszerének Lamport a LaTeX nevet adta, első hivatalos verziójának száma 2.09 lett [27]. A LaTeX szó „latech''-nek vagy „létech''-nek ejtendő. A „lateksz''-nek ejtendő „latex'' szó jelentése kaucsuktej. A LaTeX-et ettől sajátos márkajele különbözteti meg. Egyszerű szövegfájlban a márkajel `LaTeX`.

A LaTeX könnyen megtanulható, sokat tud, ezért igen gyorsan népszerűvé vált. Ma már a legtöbb nagy kiadó foglalkozik LaTeX-hel szedett művek kiadásával. Használata a műszaki és tudományos élet bizonyos területein, a műszaki, tudományos szövegszerkesztésben szabvánnyá vált, ismerete ma már az e területen dolgozók számára nehezen nélkülözhető.

Miután Leslie Lamport visszavonult a további fejlesztéstől, egy 1989-es stanfordi TeX-találkozó után Frank Mittelbach, Chris Rowley és Rainer Schöpf megalkották a LaTeX3 munkacsoportot, és belekezdtek a LaTeX újraírásába és kiterjesztésébe, egy bővített LaTeX, a LaTeX3 létrehozásába. Később a csoport munkájába sokan bekapcsolódtak, így Johannes Braams, David Carlisle, Michael Downes, Denis Duchier, Alan Jeffrey. A kitűzött cél többek között az volt, hogy az új változat a dokumentumtípusok sokkal szélesebb körét támogassa, egyszerű parancsokkal segítse a tipográfusi és a szerkesztői munkát mind a kinézeti terv elkészítésében, mind a végső finom igazítások elvégzésében, támogassa az SGML-szabványnak megfelelő dokumentumok lefordíthatóságát[2], kereteket nyújtson az addigra létrejött különféle LaTeX-változatok és az $\mathcal{AMS}$-TeX matematikai tudásának beolvasztására, támogassa a nemzeti nyelvek használatát, és persze változatlanul értse a LaTeX 2.09-ben írt dokumentumokat is.

Miközben a LaTeX3-csapat az egységes LaTeX megalkotásán dolgozott, és már igen sok kitűzött feladatot megoldott, a TeX/LaTeX-közösség tovább oszlott, elkészült például az $\mathcal{AMS}$-LaTeX, az Amerikai Matematikai Társaság LaTeX-változata [16]. Egyúttal az is világossá vált, hogy a LaTeX3 nem lesz kész a közeli jövőben. A csapat ezért 1994-ben kibocsátotta a LaTeX új verzióját, a LaTeX $2_\varepsilon$-t, mely a LaTeX3 előzetes verziójának tekinthető [27, 14, 13]. Ez gyors változást hozott a LaTeX fejlődésében, segítségével újra egységessé vált a LaTeX-felhasználók tábora. Az új verzió valóban képes volt magába olvasztani a legtöbb jó és elterjedt megoldást, így az az $\mathcal{AMS}$-LaTeX-et is. A LaTeX $2_\varepsilon$ „latech-kettő-e''-nek ejtendő, írógép stílusú márkajele `LaTeX2e`.

Könyvünk a LaTeX $2_\varepsilon$-t tárgyalja, LaTeX rendszeren a LaTeX $2_\varepsilon$-t, a LaTeX jelenlegi hivatalos változatát értjük. A LaTeX $2_\varepsilon$ a régi, LaTeX 2.09-ben írt dokumentumokat is felismeri[3], és automatikusan az ún. *compatibility mode*-ra váltva le is tudja fordítani. Ennek ellenére ma már nem érdemes új dokumentumot LaTeX 2.09-ben írni.

[2] Az SGML – Standard Generalized Markup Language – a dokumentumleírás nemzetközi szabványos nyelve, száma ISO-8879.

[3] Egy LaTeX 2.09-ben készült mű onnan ismerhető meg, hogy a forráskód első sora a `\documentstyle` paranccsal, míg egy LaTeX $2_\varepsilon$-ben írt állomány kódja a `\documentclass` paranccsal kezdődik.

## 1.2. Szerző, tipográfus, szedő, szerkesztő

Egy könyv megírásának legfontosabb fázisai már évszázadok óta a következők: a szerző, miután megírta művét, a kéziratot (ellenőrzés, lektorálás után) átadja a kiadónak. Ott egy tipográfus a könyv tartalmát is figyelembe véve minden apró részletre kiterjedően megtervezi a könyvet, azaz elkészíti a könyv oldalainak kinézeti tervét, megtervezi a borítóját stb. Ezután a kézirat eljut a szedőhöz, aki a kinézeti terv alapján kiszedi a művet. Végül a kiszedett mű a nyomdába kerül, ahol kinyomtatják és bekötik. Az egész folyamatot a szerkesztő irányítja.

A számítógépek megjelenése egyrészt megkönnyítette e folyamat több lépését, másrészt azonban kicsit össze is kuszálta a szálakat. Ma már a legtöbb szerző számítógéppel írja művét, számítógéppel végzik a szedést, és a nyomdában is számítógép vezérelte gépek nyomtatnak. A kiváló számítógépprogramok révén a szerző maga is elő tudja állítani a nyomdakész (*camera ready*) eredményt, amit a nyomdában már csak sokszorosítani kell. Ez azt jelenti, hogy a szerző a mű megírásától egészen a szedésig minden munkát kézben tud tartani, el tud végezni, a tipográfusét és a szedőét is. Csakhogy ezekhez a szerző általában nem ért. Így aztán a számítástechnika gyors fejlődése ellenére is sok olyan dokumentum születik (hivatalos levelektől kezdve a cikkeken át egészen a könyvekig), amelyek a tipográfia legelemibb szabályait sem tartják be.

Az okok egyike az, hogy ugyan a mai nagy tudású dokumentumszerkesztő programokkal nemcsak a szöveget lehet begépelni, de a tipográfiai munka egy része is elvégezhető, a programokba beépített tipográfiai tudás pedig könnyen mellőzhető.[4] A rossz dokumentumszerkesztői gyakorlatot gyakran a szerkesztő program felhasználói felülete inspirálja, mert azon a szerzői és a tipográfusi parancsok ömlesztve szerepelnek, sőt az utóbbiak vannak előtérbe helyezve.

Az igazi megoldás a vizuális megjelenés megtervezésének és a mű megírásának, azaz a tipográfusi és a szerzői munkának világos szétválasztása. Például a tipográfus dolga a cím betűtípusának, a körülötte hagyandó térköz nagyságának meghatározása, míg a szerző dolga csak annyi, hogy megmondja, mi a cím. A LaTeX koncepciójának alapját épp ez az elv képezi: a LaTeX úgynevezett stílusállományaiba beépített tipográfiai tudásnak kell a mű megjelenéséről gondoskodnia. A szerző csak igen ritkán állítgatja a betűméretet, betűtípust, nem méreget térközöket, sorkihagyásokat stb., csak közli a LaTeX-hel a szövegrészek funkcióit: „ez itt a cím'', „ezt a szót ki kell emelni''. Az, hogy mindez vizuálisan hogy jelenik meg, a LaTeX dolga. Tehát a szerző a tipográfusi munka nagy részét a LaTeX-re bízza, amely a szedői munkát a TeX-hel végezteti el. A szerző LaTeX-hel írva mentesül(het) a dokumentum vizuális megjelenésének tervezése, szerkesztése alól, de nyitva áll előtte a lehetőség, hogy szükség esetén a dokumentum végső alakjának legapróbb részleteit is befolyásolja.

[4] E programok felhasználóinak gyakori típusa, miután írt néhány sort, tipográfusi munkába kezd. Például veszi a címnek szánt szöveget, betűit megvastagítja, betűméretét megnöveli, beszúr egy-két üres sort, hunyorít, értékel, majd megismétli a fenti lépéseket. A felhasználók másik típusát a különböző fontok túlburjánzó használata, a *„fontitis"* jellemzi. Ez nemcsak a kezdő honlaptervezők betegsége.

## 1.3. A LaTeX jelene és jövője

**TeX-társaságok, TeX Live** ♦ A TeX felhasználói és fejlesztői több országban társaságokat hoztak létre. A központi TeX-társaság az amerikai székhelyű TUG (TeX Users Group). Honlapjának címe `http://www.tug.org`. A `http://www.tug.org/lugs.html` címen megtalálható a világ összes TeX-szervezete, köztük a MATEX, a Magyar TeX Egyesület is. Honlapjának címe: `http://www.inf.unideb.hu/~matex/`.

E társaságok szerteágazó tevékenységének egyike, hogy minden évben kibocsátják a TeX Live disztribúciót, amelynek 2003-as demo kiadását e könyvhöz is mellékeltük. A TeX Live jelenleg három változatban készül el, egy *demo* CD, egy *install* CD és egy DVD változatban. A *demo* változatban a TeX CD-ről is futtatható Windows, Linux és MacOS X alatt; az *install* változaton több anyag van, de onnan nem futtatható, csak telepíthető a TeX; a DVD változat minden fontosnak mondható anyagot tartalmaz, és *demo* módban is fut. A TeX Live anyaga a `http://www.tug.org/texlive/` címről ingyen letölthető, másolható, de meg is rendelhető a TeX-társaságoktól.

**LaTeX az Interneten** ♦ A TeX-hel és így a LaTeX-hel kapcsolatos anyagok gyűjtőhelye a *Comprehensive TeX Archive Network* (magyarul Átfogó TeX Archívum Hálózat, rövidítve CTAN). Legfontosabb címe: `http://www.ctan.org`, de számtalan más címen is elérhető. A TeX-anyagok leggyakrabban a `tex-archive`, esetleg a `pub/tex` könyvtárban vannak. Az archívum számunkra legfontosabb címei:

```
http://www.ctan.org/tex-archive
ftp://ftp.ctan.org/tex-archive
ftp://ftp.sztaki.hu/pub/tex
http://dante.ctan.org/CTAN
```

Amikor a továbbiakban egy CTAN-on megtalálható állomány vagy könyvtár helyét megadjuk, nem fogjuk leírni az egész címet, hanem a `CTAN` szót a fenti négy, vagy azokkal ekvivalens valamely cím helyettesítésére fogjuk használni. Például a

```
http://www.ctan.org/tex-archive/help/Catalogue/
```

cím helyett csak annyit fogunk írni, hogy `CTAN/help/Catalogue`.

Ezeken a szervereken több gigabyte-nyi anyag van felhalmozva. Ha dokumentációt, program(csomag)ot keresünk, használjuk a `http://www.ctan.org/` fő oldalán felkínált keresőt (*Search*), vagy Graham Williams TeX Catalogue-ját: a `CTAN/help/Catalogue` címen. Ha FTP-vel keresünk egy állományt, akkor használjuk a

```
quote site index string
```

`ftp`-parancsot, ahol `string` a fájlnév egy része. A LaTeX honlapjának címe:

```
http://www.latex-project.org/
```

A CTAN FTP-szerverek erről a címről is elérhetők. A LaTeX magyarításának honlapja:

```
http://www.math.bme.hu/latex/
```

Ezen a címen megtalálhatók lesznek a könyvünkkel kapcsolatos információk is.

**A LaTeX kapcsolata más programokkal** ♦ A LaTeX-re, illetve LaTeX-ről más nyelvre vagy formátumra konvertálni sokféleképpen lehet. Ezek közül kiemeljük a HTML nyelvre való konverziót. Az Eitan M. Gurari által készített, e témában jelenleg a legnagyobb tudású TeX4ht nemcsak HTML-re, hanem XML-re és MathML-re is fordít.

Fontos megemlíteni az MS Wordben írt fájlok LaTeX-re való konvertálhatóságát. Ezt például a Word `rtf`-formátumán keresztül lehet megvalósítani. A fájlt az MS Wordből nem `doc`, hanem `rtf` formátumban kell menteni, amit azután az `rtf2latex2e` programmal lehet LaTeX formátumúvá konvertálni. E konverzió megfelelően átalakítja az MS Word Equation Editorával készült képleteket is.

A legfontosabb matematikai programok, például a Maple, a Mathematica, a MuPAD vagy a MatLab képesek arra, hogy a képleteket LaTeX formátumban írják ki, esetleg arra is, hogy az egész munkalapot LaTeX-be konvertálják.

Több olyan irodai program is létezik, amelynek *export filter*ei közt a LaTeX is szerepel, ami azt jelenti, hogy a dokumentum LaTeX formátumban is elmenthető. Ilyen a Windows és Linux alatt is futó AbiWord dokumentumszerkesztő, vagy a Linux Gnome környezetének táblázatkezelője, a gnumeric. A KDE egyenesen egy LaTeX szerkesztőt kínál, a kile nevű programot, Office-ának pedig része a képletszerkesztő Kformula. Az OpenOffice *export filter*ként és külön konverterként is használható programja a Writer2LaTeX. A Word2TeX egy MS Wordbe beépülő shareware program, mely lehetővé teszi MS Word dokumentumok LaTeX-be való mentését a képletekkel együtt. A LaTeX és más formátumú szöveges állományok közti konverzió lehetőségei a `http://www.tug.org/utilities/texconv/` oldalon vannak összefoglalva.

A LaTeX-hel elkészült dokumentum PostScript és a PDF formátumba való konvertálásáról külön fejezetben szólunk.

**A LaTeX előnyei** ♦ Ahhoz, hogy a LaTeX-et az Olvasó elhelyezhesse a dokumentumkészítő (szövegszerkesztő, dokumentumszerkesztő, kiadványszerkesztő, szedő) programok között, felsoroljuk néhány jellegzetességét. A LaTeX

- képes nyomdai minőségű dokumentum előállítására;
- nyelve egyszerű, bármely szövegszerkesztővel szerkeszthető;
- sok szövegszerkesztő támogatja, többük grafikus felhasználói felülettel rendelkezik (WinEdt, TeXnicCenter, kile), van köztük WYSIWYG-szerű[5] program is (Textures, LyX, SciWord).
- nyelvének alapelemei a dokumentum logikai struktúrájának leírását szolgálják, ami mentesíti a szerzőt a vizuális szerkesztés gondjától, ugyanakkor – ellentétben a HTML-típusú nyelvekkel – a vizuális megjelenés teljesen szabályozható;
- segítségével a dokumentum olyan összetett részei, mint például az irodalomjegyzék, a tartalomjegyzék, a lábjegyzet, automatikusan készíthetők;
- nyelvével a matematikai formulák könnyen írhatók le, ugyanakkor ezeket a legmagasabb tipográfiai szinten jeleníti meg;

[5] WYSIWYG, azaz What-You-See-Is-What-You-Get, magyarul „amit látsz, azt kapod''. E betűszót azokra a programokra használják, amelyek már szerkesztés közben is a dokumentum végső alakját mutatják. A mai WYSIWYG-programok rajongóiétól különböző számítástechnikai kultúrán érlelt nézet szerint e programokra jobban illene a What-You-See-Is-All-You-Get kifejezés: „csak amit látsz, azt kapod''. A LaTeX-re építő, és nem pontosan a végső eredményt mutató programokra (pl. LyX), inkább a What-You-See-Is-What-You-Mean kifejezés illik: „amire gondolsz, azt látod''.

- maga is ingyenes, és ingyenesek a legkülönfélébb tipográfiai feladatok megoldására használható kiegészítő programcsomagok is;
- forráskódja nyílt, korlátozások csak az üzleti célú felhasználásban és a márkanevek használatában vannak (bővebben lásd: LaTeX Project Public License – `http://www.latex-project.org/lppl.txt`);
- rendkívül megbízható, szinte minden számítógéptípus minden fontosabb operációs rendszerén fut, mégis a legmagasabb hordozhatóságot biztosítja;
- kódja más programmal könnyen generálható;
- a vele írt dokumentum változatkezelővel (pl. CVS) tárolható;
- sok nemzeti nyelvet támogat, képes többnyelvű dokumentumok előállítására;
- a magyar nyelvet másutt nem tapasztalt kimagasló módon támogatja (ennek nagy része e könyvben jelenik meg először).

**A LaTeX hátrányai** ♦ A LaTeX-nek megemlíthetők bizonyos hátrányai is:

- bizonyos feladatok megoldása körülményesebb, nehezebben áttekinthető, mint a szokásos WYSIWYG rendszerekben;
- egy dokumentum kinézeti tervének megváltoztatása vagy új terv készítése általában nehéz munka, nemcsak a LaTeX, hanem néha a TeX mély ismeretét is igényli;
- a szükséges ismeretek megszerzése az ingyen rendelkezésre álló dokumentációk ellenére, néha azok nagy száma miatt is nehéz;
- a hibaüzenetek megértése, a hibák kijavítása időnként komoly feladatot jelent.

## 1.4. A könyv jelölései és tagolása

Minden Olvasónknak feltétlenül ajánljuk, hogy mindjárt az elején olvassa el ezt a szakaszt (az 1.4. sorszámút). Itt írjuk le mindazokat az egyezményes jelöléseket, amelyek a könnyebb olvasást és a jobb megértést szolgálják.

**Konvenciók** ♦ Először a könyvben használt konvenciókat ismertetjük.

`Írógép típusú betűk:` Mindazt, amit a számítógép előtt ülve be kell gépelni, továbbá az egyszerű programlistákat *írógép típusú* betűkkel szedjük. Használatukat egy példán szemléltetjük: ahhoz, hogy egy szöveg, például ez itt be legyen keretezve, azt kellett begépelnünk, hogy `\framebox{ez}`.

Groteszk betűk: A programok, programcsomagok neveit úgynevezett groteszk betűkkel szedjük. *Groteszk betű* alatt az egyforma vastagságú vonalakból álló, talpatlan betűt értjük, mely a könyvünkben így néz ki.

*Dőlt betűk:* A LaTeX-parancsok leírásakor gyakori, hogy egy paraméternek csak a típusát adjuk meg, majd részletesen leírjuk, hogy mit lehet helyettesíteni a helyébe. Ezt szemléltetjük a `\framebox` paranccsal: a `\framebox[`*sz*`][`*pozíció*`]{`*szöveg*`}` parancs egy *sz* szélességű keretet tesz a *szöveg* köré, és abban a szöveget balra, középre vagy jobbra zárja aszerint, hogy a *pozíció* helyén `l`, `c` vagy `r` betű áll. Például a `\framebox[18mm][r]{ez}` parancsok hatására egy 18 mm széles keretet kapunk, melyben a szöveg jobbra lesz igazítva: ez.

¬*:* A TEX parancsszavait *nem lehet elválasztani* a forráskódban a sor végén. A `\clear¬doublepage` parancs viszont itt nagyon túlfutna a margón, ezért bevezettünk egy mással össze nem téveszthető, a billentyűzetről be nem vihető elválasztójelet parancsszavaknak csak a könyv szövegében való elválasztására, ez a ¬ jel.

*A látható szóköz (␣):* Ha a forráskódban fontos, hogy egy adott helyen szóköz legyen, ezt a *látható szóköz* jellel jelezzük: ␣. Például ahhoz, hogy a „TEX vagy LATEX'' szöveget megkapjuk, azt kell begépelni, hogy `\TeX\␣vagy␣\LaTeX`.

*Programlisták a könyvben és a CD-n:* Sokat segíthet az Olvasónak, hogy a könyv számtalan példát ad a bemutatott parancsok vagy jelenségek megvilágítására. Ha lehet, a könyvben látszik a példa forráskódja, és mellette jobb oldalon a futtatásának eredménye, továbbá a CD-n megtalálható a példa egy teljes, azonnal futtatható LATEX-állomány formájában is. A példa kódjának sorai sorszámozva vannak, a sorszámok azonban nem részei a kódnak! Ezt azzal jelezzük, hogy a sorszámokat jóval kisebb karakterekkel szedjük. Tekintsük az alábbi példát:

```
01 Ez egy \emph{egyszerű} példa:
02 $(\sqrt{5}-1)^2=6-2\sqrt5$.
```

Ez egy *egyszerű* példa: $(\sqrt{5}-1)^2=6-2\sqrt{5}$.

Ez a példa a CD-n `1_7.tex` néven található meg, mivel ez a példa az 1. fejezetben a könyv 7. oldalán van. A CD-n található állomány nemcsak ezt a két sort tartalmazza, hanem az összes sort, ami ahhoz kell, hogy azonnal lefordítható legyen. Az ezen az oldalon szereplő esetleges következő példa, az `1_7ii.tex` nevet kapja.

`%^` : Egy programrészlet futási eredményét befolyásolhatják a kód elején, a bevezetőjében (preambulumában) kiadott parancsok. Ezeket a kódban úgy fogjuk jelezni, hogy az ilyen sorok elejére a `%^` karaktereket írjuk. A `%` azt jelenti, hogy e sor itt csak megjegyzéssor, a `^` arra utal, hogy a kódsort a program elejére, a preambulumba kell írni. Az `\euro` parancs például csak akkor írja ki az € jelet, ha a preambulumba kerül a `\usepackage{eurosym}` parancs. Ezt így fogjuk kifejezni:

```
01 %^\usepackage{eurosym}
02 A leves csak 1\,\euro\ volt.
```

A leves csak 1 € volt.

*Bekeretezett parancsok:* A 3. fejezettől kezdve a jobb áttekinthetőség érdekében minden parancsot ismertetése során egy kerettel emelünk ki a szövegből így:

`\framebox`[*sz*][*pozíció*]{*szöveg*}

Az ilyen kiemelést követi majd a parancs részletes leírása. A környezeteket a környezetet nyitó paranccsal adjuk meg. Például a `center` környezet esetén így:

`\begin{center}`

A környezetet záró `\end{center}` parancsot általában elhagyjuk, használata ugyanis magától értetődik.

∈ *(eleme jel):* A LATEX olyan nyílt rendszer, amelyhez bárki illeszthet programcsomagokat. Ezek közül több fontosabb programcsomagot részletesen is ismertetünk. Egy ilyen programcsomagot csak úgy használhatunk, ha a `\usepackage` paranccsal betöltjük. Hogy a zavart elkerüljük, a programcsomagok parancsait úgy különböztetjük meg a LATEX egyszerű parancsaitól, hogy a keretben a sor végére, egy ∈ jel után megadjuk, hogy a parancs melyik programcsomaghoz tar-

tozik. Például az \ifthenelse parancs csak az ifthen programcsomag betöltése, azaz a \usepackage{ifthen} parancs kiadása után működik. Ezt így jelezzük:

\ifthenelse{*feltétel*}{*akkor*}{*egyébként*} ∈ ifthen

Előfordulhat, hogy egy parancs a LaTeX parancsai között is szerepel, és valamelyik programcsomagban is. Ilyenkor a programcsomag az eredeti parancs képességeit továbbiakkal bővíti. Ennek jelölésére az ∈ jel után nemcsak a programcsomag nevét, hanem a LaTeX szót is megadjuk.

\begin{tabular}[*poz*]{*oszlopok*} ∈ LaTeX, array

A táblázatokat a \begin{tabular} paranccsal kell kezdeni. Ha azonban betöltjük az array csomagot, kényelmesebben tudjuk használni.

**A könyv fejezetei** ♦ A 2. fejezet olyan bevezető ismertető, amit a kezdő, vagy a LaTeX-ben nem eléggé járatos olvasónak javaslunk teljesen és alaposan átolvasni. Tartalmazza mindazt, amit egy egyszerű felhasználónak tudnia érdemes. Reményeink szerint e fejezet a teljesen kezdők számára is könnyen olvasható és érthető. A feldolgozást segíti, hogy a fejezet gyakorlatokat is tartalmaz. Ezek megoldása is megtalálható a CD-n. A fejezetben leírt ismeret elég a munka megkezdéséhez. Egy-egy rész, mint a LaTeX rajzoló és táblázatkészítő parancsairól szóló (2.6.2, 2.6.3) még ebben a fejezetben is átugorható, ha valaki ezeket a lehetőségeket nem kívánja használni.

A 3. fejezet a LaTeX programnyelvével ismerteti meg az Olvasót. Mindenkinek érdemes elolvasnia a 3.1. alfejezetig; a további alfejezeteket a mélyebb ismeretekre vágyóknak ajánljuk.

Könyvünk 4. fejezettől kezdődő részei referenciakönyvként szolgálnak. Felsoroljuk a LaTeX parancsait, áttekintjük a témához kapcsolódó fontosabb programcsomagokat, ugyanakkor a nehezebb feladatokra is mutatunk megoldást. Először a szöveg formai elemeit tekintjük át (4. fejezet), majd az egész dokumentum kinézetét befolyásoló paramétereket, parancsokat foglaljuk össze (5. fejezet). A 6. fejezetben az ún. úszó objektumokkal, azaz a táblázatokkal és ábrákkal foglalkozunk, és ennek kapcsán kiemelten foglalkozunk a (La)TeX saját eszközeivel megvalósítható grafikájával, így a METAPOST rendszerrel. A 7. fejezetet a TeX fő erősségének, a matematikai formulák szerkesztésének szenteljük. A 8. fejezetben a dokumentumon belüli utalások, valamint a külső dokumentumokra való hivatkozások kezelését tekintjük át. A 9. fejezetben a LaTeX nyelvfüggő elemeiről és a babel következő változatában megjelenő magyar nyelvi támogatásról szólunk. A 10. fejezet a LaTeX fontkezelését ismerteti. Ezután következik a függelék, melyben először a TeX és a LaTeX hibaüzeneteit soroljuk fel (A. függelék), majd a telepítéshez adunk segítséget (B. függelék). Ezután két szótárszerű fejezet következik, a C. függelékben a dokumentumszerkesztés LaTeX-ben is fontos angol szakszavait soroljuk fel fordítással és tömör magyarázattal, majd egy igen részletes állományvégződés-tár következik, amely szinte minden, a (La)TeX használata kapcsán előforduló fájltípust áttekint (D. függelék). A függeléket táblázatok zárják (E. függelék).

Azt reméljük, hogy könyvünk ott lesz a LaTeX-et használók számítógépe mellett, megkönnyítve munkájukat.

2. FEJEZET

# Az első ismerkedés

## 2.1. Bevezetés

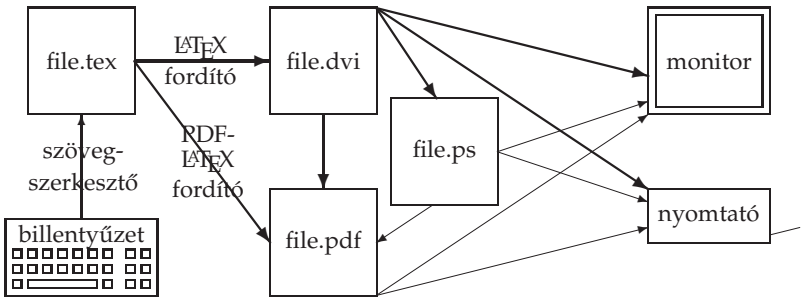
Ebben a fejezetben egy áttekintést kívánunk adni az Olvasónak a LaTeX-ről. Igyekeztünk e fejezetet úgy megírni, hogy a lehető legkevesebb előismeret is elég legyen a követéséhez, hogy használható legyen néhány órás tanfolyamok tananyagaként, a dokumentumkészítésről szóló egyetemi, főiskolai kurzusok segédanyagaként és hogy alkalmas legyen önálló feldolgozásra is. E célok elérése érdekében az anyagot igen rövid, könnyen emészthető részekre osztottuk, és a legtöbb témához önállóan megoldandó feladatokat is kitűztünk.

Először a szükséges programok telepítéséről írunk, majd nekikezdünk az első mű megírásának. Nehézkesnek fog tűnni az indulás, de egy-két óra után meglepően bonyolult, nyomdai minőségű dokumentumokat fogunk tudni szerkeszteni.

Szükséges, hogy az Olvasó alapszinten ismerje a számítógépet, a fájlműveleteket, le tudjon tölteni egy állományt az Internetről vagy egy CD-ről, és hogy ki tudja bontani, ha tömörítve van. Az Olvasónak esetleg telepítenie kell néhány egyéb programot is, ebben jól jöhet a segítség.

## 2.2. Telepítés

Mielőtt a LaTeX ismertetésébe kezdenénk, áttekintjük a használandó programokat és az azok telepítésével kapcsolatos legfontosabb információkat. Szükségünk lesz szövegszerkesztőre és olyan megjelenítő programokra, melyekkel a különböző formátumban (PDF, PostScript) előállított dokumentumokat megtekinthetjük. Végül szükségünk lesz egy TeX rendszerre. Csak a két legelterjedtebb operációs rendszeren (Windows, Linux) futó programokat tekintjük át, a többiről (Macintosh, DOS, OS/2, Amiga stb.) tájékoztatást kapunk a TeX FAQ-ból [11]. Feltételezzük az Olvasóról, hogy a telepítés nehézségeit meg tudja oldani a programokhoz tartozó telepítési utasítások segítségével, vagy ehhez segítséget tud kérni valakitől.



Ahhoz, hogy értsük, mit miért telepítünk, vázlatosan összefoglaljuk, és a fenti sematikus ábrán szemléltetjük a LaTeX működését. Először létre kell hozni egy egyszerű szöveges állományt. Leegyszerűsítve, *szöveges állományon* olyan fájlt értünk, amiben nincs más, csak azok a karakterek, amiket begépeltünk. Adjunk a fájlnak `tex` kiterjesztésű nevet, most legyen ez `file.tex`. Amit mi begépelünk, az valójában egy számítógépprogram. Ez a program azt is tartalmazza, hogy mi a végső dokumentum szövege, de azt is leírja, hogy pontosan hogy nézzen az ki. A LaTeX fordítóprogramja (neve általában `latex`) érti ezt a programot, elkészíti a tördelt dokumentumot és azt egy saját, géptől és eszköztől független (angolul ***d**evice **i**ndependent*, rövidítve dvi) formátumú fájlba írja ki. Ennek a fájlnak a neve megegyezik a forrásállományéval, csak a kiterjesztése `tex` helyett `dvi` lesz, esetünkben `file.dvi`. Ezt azután megtekinthetjük a képernyőn, kinyomtathatjuk, vagy átkonvertálhatjuk egy másik formátumba, pl. PostScript, illetve PDF formátumba. Ekkor keletkezik a `file.ps`, illetve a `file.pdf` állomány. Ezek megtekintéséhez olyan programok kellenek, amelyek nem részei egy TeX-disztribúciónak, de ingyen megszerezhetők (pl. gv ill. Adobe Reader; ez utóbbit a 6.0-s verzió előtt Acrobat Reader-nek nevezték). Van olyan változata a LaTeX-fordítónak, mely nem `dvi`-, hanem `pdf`-formátumú output állományt állít elő. Ennek a pdfLaTeX-fordítónak is egy nyíl jelöli a hatását a fenti ábrán. A nyilak közül 6 vastagabb, ezek azokat a lépéseket jelölik, amelyekhez általában egy ingyenes TeX-disztribúcióban is megtalálható a szükséges program. A vékony nyilakhoz tartozó programok a TeX-nek nem részei, ezért disztribúciótól függően előfordulhat, hogy ezek telepítéséről magunknak kell gondoskodni: tehát kell egy szövegszerkesztő, egy program a PostScript formátumú és egy a PDF formátumú állományok megtekintéséhez és kinyomtatásához A mellékelt TeX Live CD ezeket mind tartalmazza!

**Demo CD ♦** Mielőtt a telepítés nehézségeitől megrémülne az Olvasó, figyelmébe ajánljuk a mellékelt TeX Live CD-t, melyről Windows, Linux és MacOS X alól is lehet futtatni a TeX-et és a csatlakozó programokat! Persze, ha komolyabb munkát akarunk végezni, akkor helyesebb, ha telepítjük is a programokat, de induláshoz, ismerkedéshez, egyszerűbb dokumentumok szerkesztéséhez mindenképp elég!

Windows alatt a CD behelyezése után magától megjelenik egy bejelentkező ablak a TeX Live logójával (ha nem, indítsuk el a `bin/win32/TeXLive.exe` programot). Az Explore CD-Rom > Run TeX off CD-Rom menüpontot választva elindul az XEmacs szövegszerkesztő, ahol már minden lényeges feladat elvégezhető: szövegszerkesztés, fordítás, megtekintés.

Linux alatt az első feladat, hogy *mount*oljuk a CD-t, majd belépünk a `cdrom` könyvtárba, végül elindítjuk a demo változat futásához szükséges minimális beállításokat elvégző shell programot (a promptot > jelöli):

```
> mount -t iso9660 /dev/cdrom /mnt/cdrom
> cd /mnt/cdrom
> sh install-tl.sh
```

Ezután a feltett kérdésekre `d`, `1`, `r`, `r` válaszokat adunk (`d` a könyvtárak létrehozásához, `1` a `TEXDIR` változó értékének megválasztásához, ha itt az automatikusan felajánlott könyvtár nem felel meg, akkor azt is meg kell adnunk, `r` az innen való

visszalépéshez és `r`, hogy *ne* telepítse a TEX-et). Részlet a párbeszédből (az `Enter␣command:` után álló válaszokat adtuk mi):

```
Enter command: d
Current directories setup:
<1> TEXDIR: /usr/TeX
...
Enter command: 1
New value for TEXDIR [/usr/TeX]:
...
Enter command: r
...
Enter command: r
Preparing destination directories...
...
Welcome to the TeX Live system!
>
```

Ezután ki kell adni az alábbi parancsokat, illetve be kell őket írni a `.profile` állományba:

```
VARTEXMF=/usr/TeX/texmf-var
PATH=/mnt/cdrom/bin/i386-linux:$PATH
export PATH VARTEXMF
```

Ha az `/usr/TeX` helyett más könyvtárat választunk, itt is azt kell megadni. Ezután indítható az XEmacs szövegszerkesztő, ahonnan minden feladat elvégezhető!

MacOS X alatt az eljárás nagyon hasonló a Linuxéhoz, csak az elején nem kell *mount*olni, a CD a `Volume` könyvtárból olvasható.

A TEX Live DVD változata további kilenc UNIX operációs rendszeren fut demomódban. A DVD-változat is letölthető az `www.tug.org/texlive` címről.

**A TEX Live telepítése ♦** A telepítés részletes leírása megtalálható a CD `/texmf/doc/tldoc/` könyvtárában több nyelven is, valamint a `http://www.math.bme.hu/latex/` oldalon.

A Win9x felhasználóinak elsőként arról kell gondoskodniuk, hogy legyen elég memória a környezeti változóik számára. Ez a `config.sys` állományba írt `SHELL=C:\WINDOWS\COMMAND.COM /E:4096 /P` sorral elérhető.

A CD behelyezése után megjelenő TEX Live logo menüsorán a TeXLive Software > Install on Hard Disk menüválasztással indítjuk a telepítést, amire elindul a `TeXSetup.exe`. Az első üdvözlő oldalon három kérdésre kell válaszolnunk. A *Quick Install* lehetőséget kiválasztva további kérdések nélkül, az alapértelmezett beállításokkal fut le a telepítés. Kezdőknek ezt ajánljuk! A második kérdés – *Install for all Users* – arra vonatkozik, hogy a gépen minden felhasználó használhassa-e a TEX Live-ot. Ezt természetesen csak adminisztrátorként válasszuk ki. Az utolsó kérdés – *Install XEmTeX support* – melletti négyzetet is érdemes bejelölni. Ekkor több nagyon hasznos program is telepítve lesz: XEmacs, ImageMagick egy része, ISpell, Ghostscript, Perl; ha nem választjuk ki, csak az utolsó kettő, mert ezek szinte nélkülözhetetlenek.

Miután a demo CD leírásánál említett módon *mount*oltuk a CD-t és elindítottuk a `sh install-tl.sh` paranccsal a telepítést, következő parancsokat kell kiadnunk: `s b r d 1 /usr/local/texlive2003 2 /usr/local/texmf-local r i`. Az utolsó `i` parancs elindítja a telepítést. Ezután a `\texconfig` programmal még változtathatunk a beállításokon (lásd a CD `/texmf/doc/tldoc/` könyvtárában lévő dokumentációt és a B. függeléket). Végül beállítjuk a PATH változót:

```
PATH=/usr/local/texlive2003/bin/i386-linux:$PATH; export PATH
```

A telepítés MacOS X alatti módját itt nem részletezzük.

# 2.3. Az első mű

## 2.3.1. Az első mű megírása

Egy egyszerű LaTeX-dokumentumot könnyen létre tudunk hozni. Némi túlzással az írógépelésnél csak annyival kell többet tennünk, hogy a begépelendő szöveg elé írunk néhány sort, és még egyet utána. Hogyan lehet például LaTeX-hel azt a szöveget leírni, hogy „Nem is olyan bonyolult!'’. Baloldalt látjuk annak a szövegállománynak a tartalmát, amit be kell gépelnünk, és jobboldalt azt, amit majd kinyomtatás után kapunk. Tájékoztatásul megrajzoltuk az eredmény bal szélének sarkait, e jelek természetesen nem tartoznak a végeredményhez. Ismételten hangsúlyozzuk, hogy a könyv programkódjai, így a következő is megtalálható a mellékelt CD-n, továbbá hogy a kód előtti sorszámok csak a tájékozódást segítik, nem tartoznak a kódhoz, nem szabad őket begépelni.

```
01 \documentclass{article}
02 \begin{document}
03   Nem is olyan bonyolult!
04 \end{document}
```

Nem is olyan bonyolult!

Ez valóban nem bonyolult!

**A dokumentumosztály kiválasztása** ♦ A szövegállomány a `\documentclass` paranccsal kezdődik. Utána kapcsos zárójelek közé azt kell beírnunk, hogy milyen dokumentumosztályba tartozik művünk. A LaTeX öt *standard* dokumentumosztályt ismer.

`article`: A kisebb dokumentumokat, feljegyzéseket, üzeneteket, a néhány oldalas cikkeket, tanulmányokat, a rövidebb írásokat mind ebbe az osztályba soroljuk. Az esetek nagy részében tehát a `\documentclass{article}` kezdő parancsot használjuk.

`book`: A könyvek a `book` osztályba kerülnek. Ekkor a szövegállomány első sorába azt írjuk, hogy `\documentclass{book}`.

`report`: Ez a beszámolók, hosszabb tanulmányok készítéséhez használatos dokumentumosztály.

`slides`: Fóliaíráshoz való osztály.

`letter`: Ezt használhatjuk levelek írásához.

Ezen az öt alaposztályon kívül még számtalan osztály, többek között a fentiek továbbfejlesztett változatai érhetőek el, ezekről az 5.1.3. pontban szólunk.

**A preambulum és a dokumentum teste** ♦ Első művünk második sorában szerepel a `\begin{document}`, az utolsóban az `\end{document}` parancs. Dokumentumunk teljes szövege e két parancs közé kerül. Ezt a részt nevezzük a *dokumentum testé*nek.

A `\documentclass` és a `\begin{document}` sorok közötti részt, ahová olyan parancsok kerülnek, melyek az egész dokumentumra vonatkoznak, a dokumentum *preambulum*ának nevezzük. Egy LaTeX-dokumentum forrásállománya tehát mindig a következő szerkezetű:

```
\documentclass{a dokumentumosztály neve}
  preambulum
\begin{document}
  a dokumentum teste
\end{document}
```

**Csomagok** ♦ A LaTeX képességei úgynevezett *programcsomag*okkal, röviden csomagokkal (angolul *package*-ekkel) bővíthetők. Van például egy txfonts nevű csomag, amellyel elérhetjük, hogy a LaTeX a Times betűket használja a Computer Modern fontok helyett. Egy csomagot a preambulumba írandó `\usepackage` paranccsal töltünk be, melynek argumentumába írjuk a csomag nevét, esetünkben a txfonts nevet. Ahhoz, hogy első művünket Times betűkkel írjuk ki, a forrásállományba mindössze egy sort kell beszúrni:

```
01 \documentclass{article}
02 \usepackage{txfonts}
03 \begin{document}
04   Nem is olyan bonyolult!
05 \end{document}
```

## 2.3.2. Fordítás, megtekintés, nyomtatás

Elkészítettük első művünket, és azt egy `tex` kiterjesztésű állományba mentettük. A példa kedvéért a neve legyen `elsomu.tex` (a CD-n a fájlnév a fejezet- és oldalszámot is tartalmazza).

Ezután következik a *fordítás*. Ha az XEmacs-et használjuk szerkesztésre, akkor a Command > LaTeX Interactive menüpontot választva lefordíthatjuk az épp szerkesztett állományt. Az XEmacs alsó felében fognak megjelenni a LaTeX üzenetei. Minden hibaüzenetnél Enter-t kell ütni, hogy folytatódjon a fordítás. Végül az `Output written on ....dvi` üzenet fog megjelenni. Az egérrel vigyük vissza a kurzort a felső félablakba, majd a Ctrl-X, 1 kombinációval tüntessük el a fordítási ablakot. A keletkező DVI-fájlt a Command > View DVI menüpontban nézhetjük meg. Ekkor alul, az *echo area*-ban megjelenik egy kérdés, hogy mely DVI-nézőt szeretnénk használni. Fogadjuk el a WinDVI-t az Enter megnyomásával. Külön ablakban elindul a WinDVI, és mutatja a lefordított TeX dokumentumot. A WinDVI File > Print menüpontjából

nyomtatni is lehet. Ha TeXnicCenter, WinShell, WinEdt vagy valamelyik hasonló, grafikus felületű programot használjuk, akkor egy gomb, leggyakrabban egy `LaTeX` feliratú gomb megnyomásával lehet fordítani, és egy `View` feliratúval megtekinteni az eredményt.

Megtehetjük azt is, és ez szinte minden disztribúcióra vonatkozik, hogy a Windows Parancssor ablakából fordítunk, miután beléptünk abba könyvtárba, ahol az állományunk van, vagy olyan könyvtárba tettük, mely bent van a PATH-ban.

```
c:\> cd könyvtár
c:\könyvtár> latex elsomu
```

Tehát miután beléptünk a megfelelő könyvtárba, kiadjuk a `latex` parancsot, utána az állomány nevével. A kiterjesztést nem kell megadni, de lehet. Ha a fordítás sikerült, megtekinthetjük a fordítás eredményét a

```
c:\könyvtár> windvi elsomu
```

paranccsal. Ha a MiKTEXet használjuk, `windvi elsomu` helyett `yap elsomu` a megfelelő parancs!

Linux és MacOS X alatt is nagyon hasonló módon fordíthatunk. Vagy az XEmacs leomló menüit használjuk, a fent leírt módon, vagy egy terminálablakot:

```
> cd konyvtar
konyvtar> latex elsomu
konyvtar> xdvi elsomu
```

Linux alatt a nyomtatás kicsit körülményesebb lehet, ezért az

```
konyvtar> dvips elsomu
konyvtar> gv elsomu.ps
```

parancsokkal először PostScript formátumba konvertálhatjuk, majd a `gv` programmal megnézhetjük és ki is nyomtathatjuk.

További információkat találunk a függelékben és a LaTeX magyarításának honlapján: `www.math.bme.hu/latex`.

## 2.3.3. Az első nehézségek

Mivel a (La)TeX dokumentumleíró programnyelv, a programban hibát is követhet el a program szerzője. Erről fordítás közben hibaüzenetet kapunk. A hibaüzenetek tanulmányozására tekintsük az alábbi rövid LaTeX-állományt:

```
01 \documentclass{article}
02 \begin{document}
03
04 Ezzel baj lesz: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.
05
06 \begin{yyy}
07   Ez mi?
08 \end{yyy}
```

```
09
10 Itt egy ismeretlen \parancs van.
11
12 \end{document}
```

A hibaüzenet lehet csak egy figyelmeztetés, amely olyan hibát jelez, ami nem akadályozza a TEX-et a továbbfordításban. Ilyen hibaüzenetet kapunk pl. ha a LATEX nem tud megfelelő helyen eltörni egy sort, ezért a sor utolsó szava valamennyire kilóg a szedéstükörből. A fenti programot lefordítva a következő üzenetet kapjuk a képernyőre (több egyéb üzenet után):

```
Overfull \hbox (0.47346pt too wide) in paragraph at lines 4--5
[]\OT1/cmr/m/n/10 Ezzel baj lesz: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxx.
```

Ez azt jelenti, hogy a forrásszövegben a negyedik sortól az ötödikig tartó bekezdés egyik (jelen esetben az egyetlen) sora 0.47346 ponttal túltöltötte (*overfull*) a sort, azaz a sor ennyivel túlcsordult (*overflow*) a megengedett sorszélességen. A LATEX azonban itt nem áll meg, tovább fordít.

A LATEX a forráskód hatodik sorában olyan hibát talál, amivel nem tud mit kezdeni, a `\begin` parancs után olyan szó szerepel, amit nem ismer. Hamarosan megtanuljuk, hogy ide egy létező LATEX-környezet nevét kell írni, de az yyy nem az. A hiba súlyossága miatt a LATEX megáll, egy „`! LaTeX Error:`'' szöveggel kezdődő üzenetet ad, majd az üzenet végén egy „?'' jelenik meg:

```
! LaTeX Error: Environment yyy undefined.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.6 \begin{yyy}

?
```

A „?'' kiírása után egy egybetűs parancsra vár, hogy mit tegyen. A következő lehetőségeink vannak:

Return folytassa a feldolgozást;
S (*scroll*) írja ki a további hibaüzeneteket, de ne álljon meg;
R (*run*) fusson tovább a fordítás;
Q (*quietly*) hibaüzenetek képernyőre írása nélkül fusson tovább;
I (*insert*) az „i'' betű után írt szöveget illessze be a hiba helyére;
E (*edit*) szerkessze a forrásállományt;
1 . . . 9 a következő 1 vagy. . . vagy 9 tokent (l. a 101. oldalt) ugorja át;
H (*help*) írjon ki egy rövid tájékoztatót a hiba mibenlétéről;
X (*exit*) álljon le és lépjen ki;
? az itt szereplő betűparancsokat sorolja fel.

A fenti állománnyal próbáljuk végig mindegyik lehetőséget! Olyan hibaüzenetet is kaphatunk, mely nem a LaTeX, hanem az alatta futó TeX hibaüzenete. Ezek is !-jellel kezdődnek, de a `LaTeX Error` üzenet hiányzik. Ilyen üzenetet kapunk a fenti példa lefordítása végén is:

```
! Undefined control sequence.
l.10 Itt egy ismeretlen \parancs
                                  van.
?
```

Az üzenet azt jelenti, hogy a `\parancs` ismeretlen parancs. Itt jól látszik az is, hogy a (La)TeX a hibaüzenetbe beírja azt a szöveget, ahol a hibát találta, és a hiba után megtöri ezt a sort.

Néha nehéz megfejteni a hibaüzenetet, különösképp akkor, ha az nem az általunk írt szövegben elkövetett szintaktikai hiba történik. Az ilyen – az egyszerű felhasználó számára – érthetetlen hiba (*weird error*) szerencsére ritka.

A LaTeX által adott hibaüzeneteket az A. függelékben a hibaüzenetek magyarázataival együtt felsoroljuk.

# 2.4. A felhasználható karakterkészlet

## 2.4.1. Alapkészlet

**A 10 speciális jel** ♦ A forrásállomány begépelésekor alaphelyzetben csak az úgynevezett ASCII karaktereket használhatjuk (l. a szótárat a 609. oldalon). Ide tartoznak az angol ábécé kis- és nagybetűi, a számok, a szóköz, a sorvége jel, a tabulátor vagy röviden tab karakter, valamint az alábbi írásjelek és speciális karakterek:

```
. , : ; ! ? ’ ‘ " @ - + = * / ( ) [ ]
```

Ezek a jelek úgy fognak a kész dokumentumban megjelenni, ahogy begépeltük őket. Matematikai formulákban és bizonyos később részletezendő körülmények között[1] ugyanígy használható három további jel:

```
< > |
```

Alapértelmezésben e három jel helyett a (La)TeX a ¡, ¿, — karaktereket írja ki, ezért megjelenítésükre a `\textless`, `\textgreater`, `\textbar` parancsokat használjuk.

A fent felsorolt karaktereken kívül van még 10, a TeX számára speciális nyomtatható ASCII karakter:

```
\ { } % $ & # _ ^ ~
```

Mivel ezeknek (La)TeX forrásállományokban speciális jelentésük van, megjelenítésük csak parancs segítségével történhet. E tíz jelből hét, úgynevezett *escape* (ejtsd: eszkép) paranccsal – azaz a \ eléírásával – jeleníthető meg. Először a jelet, utána a kiíratót parancsát adjuk meg:

[1] Pl. a T1 kódolás használata esetén – lásd a 23. oldalon.

```
{ \{    } \}    % \%    $ \$    & \&    # \#    _ \_
```

A maradék három jel másként jeleníthető meg, mivel a \\, \^, \~ parancsoknak más jelentésük van. Több megoldást is adunk:

```
\   $\backslash$   \   \textbackslash    \   \char\string‘\\
^   \^{}           ^   \textasciicircum  ^   \char\string‘\^
~   \~{}           ~   \textasciitilde   ∼   $\sim$
```

A `\char\string‘\<` konstrukció közvetlenül azt a karaktert kéri a betűtípusból, melynek kódja < kódjával egyezik meg. A fenti táblázatban a \ és a ^ karakterre használtuk ezt a megoldást.

› **Gyakorlat:** *A (LA)TEX speciális karaktereinek megjelenítése*
Írjuk le LaTeX-ben az alábbi szöveget: 10$-t visszaadott, mert a W&C boltjaiban 50%-kal kevesebbet kellett fizetnie! ‹

A \, {, }, %, $ jelek különösen fontosak, mondhatjuk, hogy ezek a LaTeX nyelvének alapjelei.

**A \ jel (rep jel): parancskezdő karakter** ♦ A \ jelentését az eddigiek alapján is sejthetjük: a LaTeX parancsai ezzel a jellel kezdődnek. A ferde törtvonalat *per*nek olvassuk, innen az ötlet, a fordított törtvonalat – pl. egy LaTeX-parancs kiolvasásakor – *rep*nek mondjuk.

A LaTeX-parancsoknak több típusa van. Az *alfabetikus parancsok* \ jelből és alfabetikus jelekből állnak. E parancsokban megkülönböztetjük a kis- és nagybetűket, tehát `\ibolya` és `\Ibolya` két különböző parancsnév. Az alfabetikus parancsok az első nem-alfabetikus jelig tartanak, pl. egy szám, egy kapcsos zárójel, vagy egy szóköz a parancsszó végét jelzi. A következő három példa mindegyikében `\H` a parancs: `\H{o}`, `\H␣o`, `\H\i`. Nem lehetnek parancsszavak az alábbiak: `\alma2korte`, `\alma␣korte`, `\alma_korte`, `\alma2` (A ␣ jel a begépelt szóközt jelöli).

A *kétjeles parancsok* két jelből, a \ jelből és egy nem-alfabetikus jelből állnak. Pl. a `\'` parancs egy vesszőt tesz az utána következő betűre (`\'o` eredménye ó), míg pl. a \\ (ejtsd rep-rep) paranccsal kényszeríthetjük a LaTeX-et arra, hogy valahol új sort kezdjen. A `\H` parancs alfabetikus, nem számít kétjelesnek. A kétjeles parancsok a második jellel véget érnek, utánuk bármilyen karakter következhet.

A parancsok némelyikéhez hozzá lehet írni egy *-ot. Ezek a *csillagos parancsok*, melyek a csillag nélküli parancs működését kicsit módosítják. Például a \\ új sort kezd, a ** ugyancsak, de nem engedi, hogy ott a LaTeX új oldalt kezdjen. A `\chapter` új fejezetet kezd egy könyvben, a `\chapter*` ugyancsak, de nem ad a fejezetnek sorszámot.

**A % jel: megjegyzések** ♦ A % jellel *megjegyzések* iktathatók a forrásállományba. Mindaz, ami egy sorban a % jel és a sor vége között van, csak megjegyzés, beleértve még a sorvége jelet is. Így, ha egy szó végét szóköz kihagyása nélkül követi a % jel, akkor a szó összeolvad a következő sor első szavával. Erre normál szöveg beírásakor szinte soha sincs szükség, de programok írásakor igen. Az alábbi példában a „homo'' szó után szóköz kihagyása nélkül következik a %, így az egybeíródik a következő

sor első szavával. (Az egyszerűség kedvéért további példáink nagy részének forrásszövegéből kihagyjuk a `\documentclass{article}`, a `\begin{document}` és az `\end{document}` sorokat.)

```
% Az ember tragédiája hetedik szín
A homo%i
usiont hirdette.
```

A homousiont hirdette.

**A { és a } jel: blokkok létrehozása** ♦ A *kapcsos zárójelek* olyan *blokkok*at hoznak létre a forrásállományon belül, melyeket a LaTeX egyetlen egységként tud kezelni. Lássunk egy példát: a szövegbe tetszőleges szélességű és magasságú vonalat húzhatunk a `\rule` paranccsal. E parancsnak két argumentuma van, az első a vonal szélességét, a második a magasságát határozza meg. A `\rule{5mm}{0.25mm}` parancs egy 5 mm széles, 0,25 mm magas vonalat rajzol ki: ▬. Megjegyezzük, hogy itt `0.25mm` helyett `0,25mm` is írható a forrásállományban, mivel itt a (La)TeX számára egyértelmű a vessző jelentése.

A LaTeX által egységként kezelt blokkokat más módon is kijelölhetjük. A parancsok egy részének például vannak ún. *opcionális*, elhagyható argumentumai, ezeket nem kapcsos, hanem szögletes zárójelek közé kell tenni. Például a `\rule` parancsban rögtön a parancsszó után megadható, hogy a megrajzolt vonal mennyivel kerüljön az alapvonal fölé. Például az előző vonalat 1 mm-rel megemeli a `\rule[1mm]{5mm}{0,25mm}` parancs: ▬.

Blokkot jelöl ki az úgynevezett *környezet* is, mely kicsit leegyszerűsítve egy névvel ellátott blokknak tekinthető. A környezet elejét egy `\begin{`*név*`}`, végét egy `\end{`*név*`}` parancs jelöli ki a kapcsos zárójelek helyett. Ezekről a 3.4. alszakaszban szólunk részletesebben.

A blokkok tartalmazhatják egymást, de egymásból nem lóghatnak ki. Ez azt jelenti, hogy blokkok így nem kapcsolódhatnak:

`...{ ...\begin{`*név*`} ...} ...\end{`*név*`} ...`

A karakterek és a parancsszavak a LaTeX tovább már nem bontható blokkjai. Például a `\H` parancs[2] két vesszőt (*hosszú kettős* ékezetet) tesz az argumentumában szereplő betűre. Az „ő'' betűt a `\H{o}`, `\H␣{o}`, `\H␣o` parancsok mindegyikével megkaphatjuk. A harmadik esetben a `\H` és az `o` betű közé ékelt szóközre szükség van, mert ez mutatja meg, meddig tart a parancs. Szóköz nélkül a LaTeX azt hinné, hogy a parancs `\Ho`, amit nem találna parancsainak listáján, ezért hibaüzenetet adna. Ha kapcsos zárójelet használunk az argumentum körül, nincs szükség a szóközre a parancs és az argumentum között, de tehetünk is, mint azt a `\H{o}` és a `\H␣{o}` alak mutatja.

A kétjeles parancsoknál nem kell a parancs után szóközt tenni, hisz a LaTeX tudja, hol a parancs vége. Például a `\'` eredményül vesszőt (*hosszú* ékezetet) tesz az argumentumában szereplő betűre. A `\'o`, `\'{o}`, `\'␣o`, `\'␣{o}` parancsok eredménye egyaránt egy ó betű.

[2] H, mint Hungarian umlaut (rövidebb nevén hungarumlaut).

› **Gyakorlat:** *Blokk kijelölése*
A `\textsuperscript` parancs az argumentumába írt szöveget a felső kitevőbe teszi. Vizsgáljuk meg az alábbi parancsokat.

```
12\textsuperscript{h}-kor   1\textsuperscript st
12\textsuperscript h-kor    1\textsuperscript {st}
12\textsuperscripth-kor     1\textsuperscript{st}
```

Ezek közül melyik eredményezi a $12^{\text{h}}$-kor és a $1^{\text{st}}$ kifejezéseket, és melyik hibás? ‹

**A $ jel: matematikai képletek** ♦ A $ jel hatására a TEX ún. matematikai módba vált, ami azt jelenti, hogy a következő $ jelig mindent képletnek tekint, az így megkapott képletet pedig egyetlen szóként kezeli. A matematikai képletek kezeléséről szól a 7. fejezet, most csak annyit mutatunk meg, amennyi matematikai képletekből egy köznapi szöveg megírásához elég.

Matematikai módban a szóközt csak a parancsszó végének jelzésére használjuk, egyébként nem befolyásolja a képlet megjelenését. A +, −, / és = jeleket értelemszerűen használjuk. Szorzójelként használható a `\cdot` és a `\times` parancs: `$a\cdot␣b$` eredménye $a \cdot b$, `$a\times b$` eredménye pedig $a \times b$. Törtet a kétargumentumos `\frac` paranccsal jeleníthetünk meg, ennek első argumentuma a számláló, a második a nevező, például az `$\frac{1}{1+x}$` parancs ezt adja: $\frac{1}{1+x}$. Kitevő a „^'', alsó index az „_'' karakterrel képezhető, például `$a^2$` képe $a^2$, `$f_1$` képe $f_1$. Ezzel két további speciális karakter jelentését is megismertük. Gyökvonásra az `\sqrt` parancs használható: `$\sqrt{2}$` képe $\sqrt{2}$.

```
01 A bagolyköpetek átlagos
02 biomasssza-tartalma $\frac{t}{k}$,
03 ahol $t$ a zsákmányállatok össztömege
04 és $k$ a bagolyköpetek száma.
```

⌜A bagolyköpetek átlagos biomasssza-
tartalma $\frac{t}{k}$, ahol $t$ a zsákmányállatok
össztömege és $k$ a bagolyköpetek szá-
⌞ma.

## 2.4.2. Betűk bevitele paranccsal

**Ékezetes betűk, repülőékezetek** ♦ A `\'`-höz hasonló parancsokkal leírhatók a latin betűket használó nyelvek egyéb ékezetes betűi is. Az ékezeteknek e parancsokkal való megadását szokták a TEX *repülőékezet*einek nevezni. Repülőékezetekről akkor beszélünk, ha az ékezeteket a betűkre nem tudván föltenni, azokat a betű elé vagy után rakjuk, mintha így akarnánk őket ráröptetni.[3] Az alábbi táblázatban felsoroljuk a TEX ékezetesítő parancsait, mindegyik ékezetet az „o'' betűre téve. Az ékezetes betűket a függelék táblázatai között is felsoroljuk (l. E.5. táblázat, 688. oldal).

[3] A magyar ékezetes betűk repülőékezetes jelölésére Magyarországon kialakult egyik konvenció szerint az á, é, í, ó, ö, ő, ú, ü, ű betűket az `a'`, `e'`, `i'`, `o'`, `o:`, `o"`, `u'`, `u:`, `u"` karakterpárokkal kell megadni. Például a „víztükörből'' szó repülőékezetekkel leírva így néz ki: `vi'ztu:ko:rbo"l`. A szokások nyelvenként változnak, pl. német nyelvterületen a repülőékezetet a betű elé teszik.

| | | | | | |
|---|---|---|---|---|---|
| ó \'o | õ \~o | ō \=o | o̲ \b{o} | ŏ \u{o} | ı \i |
| ò \`o | ô \^o | ȯ \.o | o̧ \c{o} | ǒ \v{o} | ȷ \j |
| ö \"o | ő \H{o} | o̊ \r{o} | ọ \d{o} | o͡o \t{oo} | |

Az „i'' és „j'' betűn már van egy pont, így ahhoz, hogy oda más ékezetet tehessünk, előbb azt le kell onnan venni. Erre szolgál az előbbi táblázat utolsó oszlopának `\i` és `\j` parancsa. A `\i` parancs eredményeként kapott „ı'' betű egyébként több nyelvben, így a törökben is önálló betűként szerepel. A TEX-ben a „hosszú í'' betű előállításához is a `\i` parancsot kell használni. Pl. a sík szót a következő megoldások valamelyikével lehet leírni: `s\'{\i}k`, `s{\'\i}k`, a `s\'\i{}k`, `s\'\i␣k`. A kapcsos zárójelekre, illetve a `\i` utáni szóközre azért van szükség, hogy jelezzük a `\i` parancs végét. Ez a körülményesség sok hiba forrása, ezért a LATEX egy ideje elfogadja a `\'i` alakot is, tehát írhatjuk, hogy `s\'ik`! (Ez a LATEX korábbi változataiban vagy a TEX-ben ezt eredményezné: sík.) Hasonlóképp a franciában előforduló „ï'' betű megkapható a `\"i` paranccsal is! (Így persze LATEX-ben a „í'' kiírása kicsit nehézkesebb: `\'{\.\i}`, de ez szerencsére senkit nem zavar.)

› **Tipp:** *Melyik repülőékezetes alakot használjuk?*
Ne használjuk az ékezetesítő parancsokat szóközt igénylő alakjukban, mert azok a szövegszerkesztők, amelyek automatikus sortörést végeznek, derékba törhetik szavainkat, ami elrontja a forrásszöveg olvashatóságát! Használjuk a `\H␣o` helyett a `\H{o}`, a `\'\i␣` helyett a `\'i` alakot (régebbi disztribúciókban a `\'{\i}`, `{\'\i}`, vagy a `\'\i{}` alakok valamelyikét). Megoldható, hogy se szóközre, se kapcsos zárójelekre ne legyen szükség magyar ékezetes betűk repülőékezetes megadásához ha egy általunk nem használt kétjeles parancsra definiáljuk a `\H` parancsot! Például `\=o` írható `\H␣o` vagy `\H{o}` helyett a `\renewcommand{\=}{\H}` parancs kiadása után. ‹

A fenti parancsokkal tehát csak ASCII karaktereket használva is tudunk – ha nehézkesen is – magyarul írni! Lássunk egy „katasztrofális'' példát! Írjuk le (LA)TEX-ben az alábbi mondatot: „Dúlt árvíz, tűzvész, jött gümőkór''![4] Íme a megoldás három változatban:

```
D\'ult \'arv\'iz, t\H{u}zv\'esz, j\"ott g\"um\H{o}k\'or
D\'ult \'arv\'{\i}z, t\H{u}zv\'esz, j\"ott g\"um\H{o}k\'or
D\'{u}lt \'{a}rv\'{\i}z, t\H{u}zv\'{e}sz, j\"{o}tt g\"{u}m\H{o}k\'{o}r
```

› **Gyakorlat:** *Ékezetes betűk*
A TEX repülőékezeteit használva írjuk le az alábbi földrajzi neveket: Bakı, Csíkszentmihály, La Coruña, Korçë, Nîmes, Plzeň, Rīga! ‹

**Különleges betűk** ♦ Vannak még olyan betűk is, amelyek nem kaphatók meg a fenti ékezetesítő parancsokkal, mert testük is különleges – az angol ábécéhez képest. Ezeket *különleges betűk*nek nevezik. E betűk és az őket előállító parancsok (lásd még az összefoglaló E.5. táblázatot):

[4] E mondat az összes magyar ékezetes magánhangzót tartalmazza, mindegyiket csak egyszer, és más magánhangzó nincs benne. A legismertebb ilyen kifejezés: „árvíztűrő tükörfúrógép''. Aki pedig katasztrófa helyett inkább írói ábrándra vágyik, annak kontrasztként íme Váncsa István mondata: „Öt szép szűzlány őrült írót nyúz''.

| | | |
|---|---|---|
| Ø ø \O\o | SS ß \SS\ss | Å å \AA\aa |
| Ł ł \L\l | Æ æ \AE\ae | Œ œ \OE\oe |

› **Gyakorlat:** *Különleges betűk*
Írjuk le az alábbi földrajzi neveket: Helsingør, Gießen, Årdal, Łódż, Grœnland! ‹

## 2.4.3. Forrásállomány a magyar ábécé betűivel

Azt szeretnénk, ha magyar szöveg gépelésekor a forrásállományban `\H{u}` helyett ű betűt írhatnánk. Csakhogy a betűkódok operációs rendszerenként, sőt azon belül programonként is változhatnak: más az „ű'' betű kódja DOS, Windows vagy MacOS alatt.

A legelterjedtebb kódolások e pillanatban az ISO-8859 (másik nevén ISO-Latin) szabvány szerintiek, a Windows korábban használt ehhez közeli[5] kódrendszere, valamint a gyorsan terjedő Unicode kódrendszer. Az ISO-8859-1, vagy más néven Latin-1 kódkészlet a nyugat-európai, az ISO-8859-2, vagy más néven Latin-2 kódkészlet a közép-európai nyelvek, így a magyar nyelv betűkészletét tartalmazza. A Unicode olyan egymással szoros kapcsolatban lévő kódrendszerek közös neve, melyek célja, hogy tartalmazzák a világ minden elterjedtebb nyelvének minden írásjelét.

Igyekezzünk olyan környezetet kialakítani magunknak, amelyben nem kell a különféle kódrendszerek közt konvertálni. Például könnyen megoldható, hogy Linux és Windows alatt ugyanazzal az – akár angol, akár magyar nyelvű – szövegállománnyal dolgozhassunk konvertálás nélkül.

Az esetek többségében nem lehet eldönteni, hogy egy szövegállomány milyen kódolást használ, így azt a LaTeX-hel tudatni kell. Erre szolgál az inputenc nevű csomag, amellyel meghatározhatjuk, hogy a forrásszöveg milyen kódolásúnak tekintendő. A kódrendszer nevét a `\usepackage` parancs opcionális argumentumában adjuk meg. Ha például a forrás magyar nyelvű, és Latin-2 (vagyis ISO-8859-2) kódolású, akkor adjuk a preambulumhoz a `\usepackage[latin2]{inputenc}` parancsot. Ha egy nyugat-európai nyelven írunk, és a kódolás latin-1, akkor természetesen `\usepackage[latin1]{inputenc}` a parancs. Ha a forrásállományt DOS alatt írtuk a 852-es kódlapot használva (*code page* 852), akkor a `\usepackage[cp852]{inputenc}`, MacOS esetén a `\usepackage[applemac]{inputenc}` parancsot használjuk. Ezután mindegy, hogy az elkészült forrásállományt milyen operációs rendszer alatt írtuk, a LaTeX a másikon is jól fog futni vele, ugyanazt az eredményt fogja adni, bár lehet, hogy a forrásállományt nem fogjuk tudni könnyen elolvasni.

Ha a Unicode UTF-8 nevű kódrendszerét használjuk, akkor az inputenc csomag mellett be kell tölteni az ucs nevű csomagot is:

```
01 \usepackage{ucs}
02 \usepackage[utf8]{inputenc}
```

[5] A Latin-1 és a Windows kódkészlet a 32–126, 161–172, 174–255 kódhelyeken megegyezik, azaz a betűk és a fontosabb írásjelek ugyanott vannak. Részletesen lásd a 10. fejezet táblázatát.

Ilyenkor a fájlnak tilos a 3 UTF-8 fejléc-karakterrel kezdődnie. További információk olvashatók az 527. oldalon.

A fenti kódrendszerek valamelyikét használva már az előző katasztrófa-mondatot is sokkal könnyebb lesz leírni:

```
\documentclass{article}
\usepackage[latin2]{inputenc}
\begin{document}
Dúlt árvíz, tűzvész, jött gümőkór.
\end{document}
```

Dúlt árvíz, tűzvész, jött gümőkór.

› **Tipp:** *Hogyan írjunk magyarul külföldön?*
Magyarul akkor is egyszerűen írhatunk LaTeX-állományt, ha a UNIX vagy a Windows rendszer alatt nincs latin-2, illetve CE, azaz közép-európai fontkészlet telepítve, és nem áll rendelkezésünkre megfelelő Unicode-ot használó program sem. Nyugodtan dolgozhatunk a nyugat-európai fontokkal is. A magyar ábécé betűi a kis és nagy „ő'' és „ű'' betűt kivéve mind szerepelnek a latin-1 készletben is. Azon a kódhelyen, ahol a latin-2-ben az „ő'', illetve az „ű'' betűk vannak, a latin-1-ben az „õ'', illetve az „û'' betűt találjuk, azaz a „hullámos o'' és a „kalapos u'' betűt. Ez azt jelenti, hogy egy latin-1 fontot használó számítógépen szerkesztve állományunkat e két utóbbi betűt kell használnunk az „ő'' és „ű'' helyett. A preambulumba ilyen esetben is a `\usepackage[latin2]{inputenc}` parancsot kell írni!

```
\documentclass{article}
\usepackage[latin2]{inputenc}
\begin{document}
  Dúlt árvíz, tûzvész, jött gümõkór.
\end{document}
```

Dúlt árvíz, tűzvész, jött gümőkór.

Tehát egy nyugat-európai készletet használó gép előtt ülve a fenti szöveg begépelésével is a jó eredményt kapjuk! ‹

A kódkészletekről további információk találhatók a 10. fejezetben.

› **Gyakorlat:** *Az inputenc csomag használata*
Írjuk le LaTeX-ben repülőékezetek használata nélkül csupa nagy-, majd csupa kisbetűvel, hogy „árvíztűrő tükörfúrógép''! Ha operációs rendszerünkkel lehetséges, adjunk latin-2-es és UTF-8-as megoldást is! ‹

## 2.4.4. A TeX belső kódkészlete

**OT1** ♦ A TeX által kezdetben használt belső kódolás csak az angol ábécé betűit tekintette betűnek, ami például megnehezítette más nyelvű szövegek szavainak elválasztását. Az inputenc csomag nem erre a problémára ad megoldást, hisz általa az 'ú' betű csak lefordítódik a `\'u` parancsra, nem válik betűvé. Ahhoz, hogy a (La)TeX „értsen'' egy nyelvet, az kell, hogy annak minden betűje önálló belső kóddal rendelkezzék. Ez különbözhet az input szövegben használt kódétól! A TeX által eredetileg használt kódrendszer neve OT1 (old text 1).

**Európai betűkód: T1** ♦ A LaTeX *belső kódkészlet*e a fontenc nevű csomaggal változtatható meg. A kódkészlet nevét a `\usepackage` parancs opcionális argumentumaként kell megadni. Például a magyar nyelvhez a T1 (text 1) belső kódkészletet kell használni, aminek betöltéséhez a `\usepackage[T1]{fontenc}` parancsot kell a preambulumba írni. (A LaTeX régebbi változatával való kompatibilitás céljából használható a `t1enc` csomag is: `\usepackage{t1enc}`.)

Az európai kódkészletre a T1 mellett a Cork[6] elnevezés is használatos.

Az úgynevezett ec (Extended Computer Modern) fontok már a T1 kódkészlet szerint készültek. A LaTeX alapértelmezésben (OT1 kódolás esetén is) ezeket a fontokat használja. Erről részletesen az 10. fejezetben írunk.

A fontenc csomag `T1` opcióval való betöltése után akár azt írjuk, hogy ő, akár azt, hogy `\H{o}`, a LaTeX mindenképpen ő betűként fogja azt reprezentálni a maga számára. Persze ő betűt változatlanul csak akkor írjunk a forrásállományba, ha betöltöttük az inputenc csomagot. Ha nem töltjük be, akkor a LaTeX a forrásállomány betűinek kódjait a T1 kódolás szerint értelmezi.

A T1 kódolást használva néhány olyan betűt is meg lehet jeleníteni, amelyiket az OT1 kódkészlettel még nem lehetett[7]. A használható karaktereket a könyv végén lévő fonttáblákban is megtalálhatók, itt csak az OT1-ben nem szereplő betűket adjuk meg (lásd még az E.5. táblázatot, 688. oldal):

```
Đ \DH  Đ \DJ  Ŋ \NG  Þ \TH  L̛ \v{L}  t̛ \v{t}  ǫ \k{o}
ð \dh  đ \dj  ŋ \ng  þ \th  l̛ \v{l}  d̛ \v{d}
```

› **Gyakorlat:** *Az OT1-ben nem, de a T1-ben már szereplő betűk*
Írjuk le az alábbi földrajzi neveket: Częstochowa, Piešťanyi, Seyðisfjörður, Þórshöfn, Đurđevac! ‹

A T1 kódolást használva nemcsak több betű, de több írásjel is elérhetővé válik. Ezekről is szó lesz a következőkben.

## 2.4.5. További jelek

**Ligatúrák** ♦ Ligatúrán a nyomdászatban betűknek a szokásosnál szorosabb összekötését értik. Legismertebb példa ligatúrára az ún. f-ligatúra, melynek lényege, hogy az 'f' betű utáni 'i' vagy 'l' az 'f' betűvel összeolvad. A TeX alapértelmezett fontja öt f-ligatúrát ismer: 'fi', 'fl', 'ff', 'ffi', 'ffl'. Vizsgáljuk meg ezeket az alábbi példán:

```
01 \LARGE\noindent
02 ff fi fl ffi ffl\\[2mm]
03 f{}f f{}i f{}l f{f}i f{f}l
```

ff fi fl ffi ffl
ff fi fl ffi ffl

Látható, hogy kapcsos zárójelekkel megakadályozható a ligatúra megjelenése. Erre használható a `\textcompwordmark` parancs is, melyet egyszerűen a két elválasztandó betű közé kell írni. Az angol és a magyar nyelvben nincs olyan szabály, mely tiltaná

[6] Cork egy ír város neve, itt volt az a konferencia 1990-ben, ahol e kódkészlet bevezetése mellett döntöttek.

[7] Új ékezetesítő parancs az ogonyeket kitevő `\k` parancs, és bizonyos szláv nyelvek gyakorlatának megfelelően négy betű mellett megváltozik a `\v` jelentése: vesszőt tesz az L, l, d és t betűk mellé.

a ligatúra megjelenését, de a német nyelvben igen. Németül írva ügyelni kell, hogy a szóösszetétel tagjai közti, valamint a szótő és toldalék közti határt a ligatúra ne lépje át.

› **Gyakorlat:** *Ligatúrák*
Tanulmányozzuk a ligatúrákat az alábbi szövegben, majd összehasonlításul akadályozzuk meg megjelenésüket: „Puff! Oda a maffia mafla fia”! Ezután írjuk le helyesen – ligatúrák megjelenése nélkül – a német Auflage és Briefinhalt szavakat kapcsos zárójelek, illetve a `\textcompwordmark` parancsnak a két összetevő közé ékelésével! ‹

A ligatúrák megjelenítésére használt technikát a (LA)TEX felhasználja más karakterek esetén is: pl. ha két szimpla felső idézőjelet írunk egymás mellé a forrásállományban, akkor az outputban egy dupla idézőjel lesz belőle, vagyis egyetlen karakter. Bár a technikai megoldás azonos, természetesen a dupla idézőjel ettől nem lesz ligatúra.

**Idézőjelek** ♦ Az angol nyelvben kétféle *idézőjel*et használnak, a szimpla és a dupla idézőjelet, és mindegyikből van bal (nyitó) és jobb (záró) változat. A számítógép billentyűzetén általában megtalálható mindkét szimpla idézőjel. A betűtípustól függően a nyitó idézőjel egy felső vessző, ami 6-osra emlékeztet, alul szélesebb, ‘, ’, esetleg balra dől: `. A záró idézőjel egy 9-esre hasonlít, felül szélesebb, egy kicsit jobbra dől: ’, ’, esetleg függőleges. Magyar billentyűzeten a nyitó az AltGr-7, a záró a Shift-1 billentyűkombinációval kapható meg, amerikai billentyűzeten a nyitó az 1 mellett, a záró pedig az Enter billentyű mellett található.

LATEX-ben dupla idézőjelet a megfelelő szimpla idézőjelből úgy kapunk, hogy azt kétszer egymás után begépeljük. A forrásállományban a nyitó ‘ kétszeri megnyomásával kapott karaktersorozatból az outputban “ lesz, míg két ’ billentyűütésből egy ”. Az ASCII karakterkészlet *egyetlen* dupla idézőjeléből a LATEX 99 formájú dupla záró idézőjelet csinál, de az, hogy a billentyűzeten lévő *egyetlen* ” gomb megnyomására mi történik, az a szövegszerkesztőtől is függ. Például az emacs szövegszerkesztőben beállítható, hogy e billentyű megnyomására felváltva két szimpla nyitó, majd két szimpla záró idézőjel jelenjen meg.

Ha egymás mellé kerülne több idézőjel, például egy dupla és egy szimpla, akkor összesen három szimplát kellene a forrásállományba leírnunk, amiről nem látszana, hogy mi is valójában. A zavar elkerülésére a `\,` paranccsal hagyjunk ki köztük egy kis helyet.

Az amerikai nyitó idézőjel 66, a záró 99 formájú, a belső nyitó idézőjel 6, míg a záró 9 formájú. Az angolban épp fordítva, 6 és 9 formájú a külső, 66 és 99 formájú a belső idézőjel. Jó tudni, hogy amerikai szokás szerint – ellentétben a magyarral vagy az angollal – az írásjelek becsúsznak az idézőjelen belülre.

```
Answer ‘‘yes,’’ or ‘‘no.’’\\
‘‘\,‘Yes,’ he said.’’  (amerikai)\\
‘\,‘‘No’’, she said.’  (angol)
```

```
Answer “yes,” or “no.”
“‘Yes,’ he said.” (amerikai)
‘“No”, she said.’ (angol)
```

A többi európai nyelvben használatos idézőjel elérhetővé válik a fontenc csomag betöltése után a T1 kódkészletet használva! A magyar szövegben az idézet egy

alsó 99 formájú jellel kezdődik, amit két vessző egymás után írásával kapunk meg. A záró idézőjel olyan, mint az amerikai, felső 99 alakú. Ezt nevezik „macskaköröm''-nek. A belső idézőjel a „lúdláb'' (»xxx«), amelyet a forrásállományban két > jellel nyitunk, és két < jellel zárunk. Használható még – főként belső – idézőjelként a „félidézőjel'' ('xxx'), melynél a nyitó és a záró is felső 9-es formájú. Ezt a jelet használjuk aposztrófként is (az aposztrófot hiányjelnek is nevezik). A német nyitó megegyezik a magyarral, a záró felső 66 formájú („xxx“), de használják a lúdlábat is (»xxx«). A francia idézőjel is a lúdláb, de fordítva áll, mint a magyarban, és nem tapad a szöveghez (« xxx »). Ezeket foglalja össze a következő példa:

```
%^\usepackage[T1]{fontenc}
magyar: ,,xxx >>yy 'zzz' yy<< xxx''\\
német: ,,xx ,yy‘ xx‘‘; >>xx ,yy‘ xx<<\\
francia: << xxx << yyy >> xxx >>
```

```
„magyar: „xxx »yy ’zzz’ yy« xxx''
 német: „xx ‚yy‘ xx“; »xx ‚yy‘ xx«
 francia: « xxx « yyy » xxx »
```

Ha a LaTeX tudja, hogy magyar szövegről van szó, ki is találhatja, hogy mikor milyen idézőjelet használjon. Valóban, a babel csomag épp a nyelv azonosítására és a nyelvi sajátosságok megvalósítására szolgál. Legújabb változatának betöltése, vagyis a

```
\usepackage[magyar]{babel}
```

parancs kiadása után a \textqq parancs argumentuma köré kiteszi a megfelelő idézőjelet. A fenti példa így írható le:

```
\textqq{xxx \textqq{yy
\textqq{zzz} yy} xxx}
```

```
„xxx »yy ’zzz’ yy« xxx''
```

› **Gyakorlat:** *Idézőjelek, hiányjel*
Írjuk le az alábbi idézetet a babel használata nélkül, majd a babel segítségével is: „Úgy köszöntöm: »Szia Enzo bá'!«'' ‹

**Kötőjel** ♦ Más néven elválasztójel, diviz, angol nevén *hyphen*. Ez jelöli a sorvégi elválasztást.

```
A~leghosszabb magyar szó a Guinness'92
szerint: legeslegmegszentségteleníttet%
hetetlenebbeiteknek.
```

```
A leghosszabb magyar szó a Guin-
ness'92 szerint: legeslegmegszentség-
teleníttethetetlenebbeiteknek.
```

Használatának további esetei: összetett szavakban (levegő-mintavétel), szóösszetételekben (orrán-száján), azonos elő- vagy utótagú felsorolásokban (gép- és gyorsíró, betűtípus és -méret), kétezer feletti számoknál (kétezer-három), kettős családnevekben (Konkoly-Thege), bizonyos többelemű földrajzi nevekben (Arany-patak-völgy), nem tartományt, hanem vagylagosságot, bizonytalanságot kifejező számkapcsolatokban (egy-két ember, 5-6 éves), az -e kérdőszócska előtt (tudod-e), tulajdonnévhez kapcsolt főnév előtt (József Attila-díj), mássalhangzó háromszorozódás esetén (sakkkör), a néma betűre és az írásrendszerünkben szokatlan betűcsoportokra végződő szavak toldalékolásakor (Edinburgh-ból).

```
01 hébe-hóba, Árpád-ház, tv-t, egy-kettőre,
02 10-12 ember, húsz-huszonöt év börtön,
03 Eötvös-féle, Konkoly-Thege.
```

hébe-hóba, Árpád-ház, tv-t, egy-kettőre, 10-12 ember, húsz-huszonöt év börtön, Eötvös-féle, Konkoly-Thege.

**Nagykötőjel** ♦ Nagykötőjel, más néven félkvirtmínusz, angol nevén *en dash*. A kvirt a teljes betűnagyság mérete, a félkvirtmínusz tehát ennek felével egyenlő hosszúságú mínuszt jelent. A forrásállományban két egymás után írt kötőjellel kapjuk meg, azaz a kódbeli `--` eredménye a dokumentumban '–' lesz. Használatának leggyakoribb esetei: számtartományok megadásakor, vagy valamettől valameddig viszonyt érzékeltető kifejezésekben (1848–49-es, Párizs–Dakar, 15–21. oldal, kelet–nyugati), géptípusok betű- és számjelzése között (TU–154), nemzetek neveinek összekapcsolásakor (brazil–magyar meccs), szerzőpáros neveinek összekapcsolásakor (Cauchy – Peano-tétel). Ez utóbbi esetben előtte és utána szokás egy kis térközt hagyni a `\,` parancs beékelésével.

```
01 10--12.~oldal, június--júliusban,\\
02 TU--154-es, észak--déli,\\
03 Bolzano\,--\,Weierstrass-tétel.
```

10–12. oldal, június–júliusban, TU–154-es, észak–déli, Bolzano – Weierstrass-tétel.

**Gondolatjel** ♦ Magyarban a gondolatjel rajzolata megegyezik a nagykötőjelével. Előtte és utána szóköz van, kivéve, ha írásjel követi. Leggyakrabban közbevetések esetén és párbeszédeknél használjuk. Gondolatjelként az angol is használja ugyanezt a jelet, de használja a kvirtmínusz jelet is, mely még hosszabb (angol neve *em dash*). (A kvirtmínusz egy betűfokozatnyi széles mínuszjel.) Három '-' jellel kell megadni, azaz `---` eredménye —. Angolban előtte és utána nincs szóköz! Magyarban e jelet nem használjuk!

```
01 Em dash---like this---is the longest.
02 Magyarban -- mondta -- ez a gondolatjel.
```

Em dash—like this—is the longest. Magyarban – mondta – ez a gondolatjel.

A kivonás jele a (LA)TEX-ben formájában is különbözik mindegyik fenti jeltől: `-`, `--`, `---`, `$-$` képe rendre -, –, —, −.

**Három pont** ♦ Más néven hármaspont, angolul *ellipses*. Megjelenítésére a `\dots` parancs használható. Helytelen három egymás mellé tett ponttal helyettesíteni! Ha egy szövegrész kihagyását jelezzük vele, előtte és utána is van szóköz, ha egy gondolat befejezetlenségének érzékeltetésére használjuk, tapad az előtte lévő szóhoz.

```
01 ,,\dots várom a párom \dots\ üres
02 a polc\dots\\
03 Mit is akartam mondani\dots
```

„... várom a párom ... üres a polc... Mit is akartam mondani...

**Szóköz** ♦ A forrásállomány szavai közti szóköz az outputban is szóközként jelenik meg. Sőt, a forrásállományba egymás mellé írt több szóközből is egy lesz! Ez akkor okozhat gondot, ha egy argumentum nélküli, nem kétjeles parancsot írunk a szövegbe, azt ugyanis az őt követő szóköz zárja le, így a szóköz eltűnik. Például a `\LaTeX`

parancs kiírja a LaTeX márkanevet. Ha azt írjuk: `\LaTeX␣könyv`, azt kapjuk, hogy LaTeXkönyv, ami rossz. Két dolgot tehetünk: kapcsos zárójelekkel lezárjuk a `\LaTeX` parancs hatását, vagy a `\␣` paranccsal kényszerítjük ki egy szóköz megjelenítését:

```
01 \LaTeX könyv, \LaTeX     könyv,\\
02 {\LaTeX} könyv, \LaTeX{} könyv,
03 \LaTeX\ könyv, \LaTeX \ könyv
```

LaTeXkönyv, LaTeXkönyv,
LaTeX könyv, LaTeX könyv, LaTeX könyv,
LaTeX könyv

**Törhetetlen szóköz** ♦ Korábbi példáinkból látjuk, a LaTeX maga dönt, hogy hol töri el a sort. Általában igyekszik szavak határán eltörni. Ha nem akarjuk, hogy egy adott szóköznél eltörjön a sor, a törhetetlen szóközt kell használnunk, melyet a ~ karakter jelöl. Többek között nevek rövidítésénél, mennyiség és a mértékegység között, uralkodók sorszáma és neve között, mondatkezdő 'A' betű után használjuk:

```
01 M.~S.~mester, 9~cm, IV.~Béla, Arany~J.
```

M. S. mester, 9 cm, IV. Béla, Arany J.

Különböző méretű, törhetetlen és merev szóközökről szól a 3.1.3. pont.

**Ami nincs a billentyűzeten** ♦ Egy szövegben az alap jelkészletbe tartozó írásjeleken (. , : ; ? ! stb.) kívül további jelekre is szükségünk lehet. Néhány jelet a (LA)TeX a ligatúráknál látott technikával jelenít meg (pl. «, ", –), azonban többségüket paranccsal kell megadni. Ezek gyűjteménye a CTAN-on a [33] könyvben van, mi a 2.1. táblázatban csak a leggyakrabban előfordulókat tekintjük át. A parancs mellett, a felső kitevőben szereplő név – ha van – annak a csomagnak a neve, melyet az adott karakter használatához be kell tölteni. Ahol ez a név zárójelben van, ott a csomagot nem szükséges betölteni, de ha betöltjük, jobban megtervezett karaktert kapunk!

2.1. táblázat. Néhány különleges karakter

| | | | |
|---|---|---|---|
| € | `\euro`[eurosym], `\EUR`[marvosym] | € | `\EURtm`[marvosym] |
| $ | `\$`, `\textdollar` | £ | `\pounds` |
| ¢ | `\textcent`[textcomp] | ¢ | `\cent`[wasysym] |
| Ⓟ | `\textcircledP` | ® | `\textregistered`(textcomp) |
| © | `\textcopyright`(textcomp) | ↄ | `\textcopyleft`[textcomp] |
| SM | `\textservicemark`[textcomp] | TM | `\texttrademark`(textcomp) |
| † | `\dag` | ‡ | `\ddag` |
| * | * | * | `\textasteriskcentered` |
| • | `\textbullet` | ○ | `\textopenbullet`[textcomp] |
| · | `\textperiodcentered` | ␣ | `\textvisiblespace` |
| ' | `\textquotesingle`[textcomp] | " | `\textquotedbl`[textcomp] |
| ¡ | !‘, `\textexclamdown` | ¿ | ?‘, `\textquestiondown` |
| ‰ | `\permil`[wasysym] | ‰ | `\textperthousand`[textcomp] |
| § | `\S`, `\textsection` | ¶ | `\P` |
| ª | `\textordfeminine`(textcomp) | º | `\textordmasculine`(textcomp) |
| № | `\textnumero`[textcomp] | ✓ | `\checkmark`[amssymb] |
| ✠ | `\maltese`[amssymb] | ※ | `\textreferencemark`[textcomp] |

## 2.5. A főszöveg és járulékos részei

A főszöveg, mely a szerző mondanivalóját tartalmazza, a folyószövegből, az azt kísérő jegyzetekből, valamint illusztrációkból áll. E szakaszban a folyószöveg elemeit és a csatlakozó járulékos részeket tekintjük át, melyeket a LaTeX generál a szerző által a folyószövegbe írt utasítások alapján. Az illusztrációkkal, pl. táblázatok készítésével, ábrák beillesztésével külön szakaszban foglalkozunk.

### 2.5.1. A folyószöveg tagolása

Egy dokumentum betűi szavakká, a szavak mondatokká, a mondatok bekezdésekké, végül ezek a fejezetek többszintű hierarchiájába rendeződnek.

**Szavak, mondatok, bekezdések felismerése** ♦ A (La)TeX a forrásfájlban lévő jelekből találja ki a szavak, mondatok és bekezdések határait. A szabályok egyszerűek. Két *szó*t szóközzel, tabulátor karakterrel vagy egy sorvége jellel lehet elválasztani egymástól. Két szó közé tetszőleges számú szóköz vagy tabulátor karakter kerülhet, sorvége jelből viszont legfeljebb csak egy. A *bekezdések* közé legalább egy üres sort kell szúrni, azaz legalább két sorvége jelet (legalább kétszer meg kell nyomnunk az Enter vagy Return billentyűt). A bekezdés végét a `\par` parancs is jelezheti. E parancs a (La)TeX programozásakor nélkülözhetetlen, szöveg bevitelekor azonban nem használjuk. *Mondat* végét írásjel jelzi, de a nagybetű utáni pontot a (La)TeX nem tekinti mondatvégi pontnak. Precízebben fogalmazva a szabályok a következők:

1. A sor eleji szóközöket és tabulátor karaktereket a (La)TeX figyelmen kívül hagyja.
2. Szóközök, tabulátorkarakterek és sorvége jelek egy sorozatát a sorvége jelek számától függően a LaTeX vagy egy szóközre vagy egy `\par` parancsra cseréli: ha a sorvége jelek száma a sorozatban 0 vagy 1, akkor szóközre, ha több, mint 1, akkor `\par` parancsra. Ez utóbbi parancs hatására új bekezdés indul.
3. Mondat végét pont, felkiáltójel, vagy kérdőjel jelzi, ha előtte nem nagybetű áll, és utána van legalább egy szóköz, tab vagy sorvége jel.

A fenti szabályokat tanulmányozhatjuk az alábbi példán:

```
Mindegy,     hogy hány szóköz kerül két szó
 közé. A~sorvége jel is szóközzé válik
, ezért ez a vessző nincs jó helyen%
   , de ez igen, mert    a százalékjel
törli a sorvége jelet, a sor eleji
 szóközök pedig nem számítanak.

Ez egy új bekezdés, mert előtte kétszer is
megnyomtuk az ENTER billentyűt. \par Ez
egy újabb bekezdés. NB. e rövidítés
után nincs vége a mondatnak, mert a pont
előtt nagybetű áll.
```

Mindegy, hogy hány szóköz kerül két szó közé. A sorvége jel is szóközzé válik , ezért ez a vessző nincs jó helyen, de ez igen, mert a százalékjel törli a sorvége jelet, a sor eleji szóközök pedig nem számítanak.

Ez egy új bekezdés, mert előtte kétszer is megnyomtuk az ENTER billentyűt.

Ez egy újabb bekezdés. NB. e rövidítés után nincs vége a mondatnak, mert a pont előtt nagybetű áll.

A *mondat végét* azért kell megismernie a (LA)TEX-nek, mert van olyan tipográfiai gyakorlat, mely mondatok között kicsit nagyobb térközt hagy, mint a szavak közt. A (LA)TEX-ben is ez az alapértelmezés. A `\frenchspacing` parancs beírásával e különbségtétel megszüntethető, majd visszaállítható a `\nonfrenchspacing` paranccsal. A magyar tipográfiai hagyományok sem tesznek különbséget e két térköz között, ezért az új `babel` csomag a `magyar` opció meghívása esetén automatikusan kiad egy `\frenchspacing` parancsot.

Előfordulhat, hogy a mondatot záró írásjel elé egy nagybetűt kell írnunk. Ekkor a (LA)TEX-hel tudatni kell, hogy ez most mégis mondatvég. Ezt úgy tehetjük meg, hogy az írásjel elé egy `\@` parancsot írunk.

Ugyanígy az is megeshet, hogy egy rövidítés kisbetűkből áll, ekkor pedig azt kell tudatnunk a LATEX-hel, hogy a pont nem a mondat végét jelöli. Ekkor a pont után vagy a `\␣` parancsot vagy a törhetetlen szóköz `~` jelét kell írnunk. Íme egy szemléltető példa, a második sor a jó megoldás:

```
01 \nonfrenchspacing
02 Leállt a BKV. Én pl. gyalog mentem.\\
03 Leállt a BKV\@. Én pl.\ gyalog mentem.
```

Leállt a BKV. Én pl. gyalog mentem.
Leállt a BKV. Én pl. gyalog mentem.

**Címrendszer** ♦ Egy hosszabb dolgozat vagy egy könyv tartalmának áttekinthetővé tételére a folyószöveg bekezdésekre osztása általában már kevés. Ekkor a dokumentumot a szöveg tartalmától függően bekezdésekből álló részekre kell osztani, és e részeket címmel ellátni. E felosztás finomítható kisebb részekre osztással akár négy-öt szint mélységig is.

A hierarchikus felosztás szintjeinek neve is van. Ezeket a LATEX angol elnevezéseit követve adjuk meg: rész (*part*), fejezet (*chapter*), szakasz (*section*), alszakasz vagy pont (*subsection*), al-alszakasz vagy alpont (*subsubsection*), paragrafus (*paragraph*), alparagrafus (*subparagraph*). A hierarchia általában úgy mutatkozik meg egy kész művön, hogy a felsőbb szintek címei jobban ki vannak emelve (például vastagabb, nagyobb méretű betűkkel vannak szedve, és a térköz is nagyobb előttük és utánuk). A sorszámozás – ha van – a hierarchiát úgy tükrözi, hogy az alsóbb szintű viseli a felette lévők sorszámát, például a 3.1.4. alszakasz a 3.1. szakaszban, az pedig a 3. fejezetben van. A szöveg felosztásának legmagasabb szintje, a rész, nem vesz részt a sorszámozás rendjében. Például lehet, hogy az 1–5. fejezetek tartoznak az I. részbe, majd a 6–12. fejezetek a II. részbe.

Egy hierarchiaszinten egy új egység megnyitásához a megfelelő hierarchiaszint nevét kell parancs formájában megadni, majd kapcsos zárójelek között a címet. Tehát sorrendben a parancsok: `\part`, `\chapter`, `\section`, `\subsection`, `\subsubsection`, `\paragraph`, `\subparagraph`.

A `book` és `report` dokumentumosztályokkal az összes fenti hierarchiaszint használható, a `letter` osztályban egyik sem – levélben nincsenek fejezetek. Az `article` dokumentumosztályban a `\chapter` parancs nem használható. Ennek az a kellemes következménye, hogy már megírt cikkekből a címrendszer megváltoztatása nélkül könnyen össze lehet állítani egy beszámolót, disszertációt vagy könyvet úgy, hogy a cikkek mindegyike egy-egy fejezet (chapter) legyen a nagyobb műben.

› **Tipp:** *Áttekinthető forrásállomány*
A fejezetkezdő parancsok előtt és után a forrásfájl jobb olvashatósága érdekében érdemes egy-egy üres sort hagyni. Ennek nincs hatása az eredményre. ‹

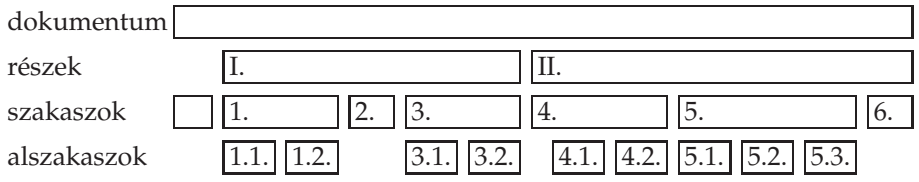
**A címrendszer szintszámai** ♦ A hierarchia egyes szintjei meg vannak számozva. A szakasz ún. szintszáma 1. Ettől lefelé haladva az alszakasz szintszáma 2, az al-alszakaszé 3, a paragrafusé 4 és az alparagrafusé 5. Fölfelé haladva cikk esetén a rész száma 0, míg könyv és beszámoló esetén a fejezet száma 0, a rész szintszáma -1. Összefoglaló táblázat erről a 695. oldalon található.

**A címek sorszáma** ♦ A LaTeX-ben a címrendszer címei automatikusan sorszámot kapnak. Azt, hogy milyen mélységig legyenek a fejezetek címei sorszámozva, a szintszámok ismeretében megváltoztathatjuk. Például a cikkekben alapértelmezésben a szakasz, alszakasz és az al-alszakasz kap sorszámot, a paragrafus és az alparagrafus nem. Ha azt akarjuk, hogy csak a szakasz és az alszakasz kapjon sorszámot, akkor, mivel az alszakasz szintszáma 2, az alábbi parancsot kell megadnunk a preambulumban:

```
\setcounter{secnumdepth}{2}
```

Ha csak egyetlen fejezetnek nem akarunk sorszámot adni, akkor a fejezetkezdő parancs csillagos alakját kell használni. Például az előszó egy mű elején általában nem kap sorszámot, ugyanakkor a rákövetkező fejezet az 1. sorszámot kapja. Ha ezt a megoldást választjuk, akkor az előszó megírását a `\section*{Előszó}` paranccsal kezdjük. A csillagos alak használatának van két mellékhatása: a cím nem kerül be a tartalomjegyzékbe és az élőfejbe sem. Hogy hogyan lehet e mellékhatásokat megszüntetni, azt a 2.5.8 paragrafusban írjuk le.

Egy cikk fejezetekre osztása tehát sematikusan pl. a következő lehet:




› **Gyakorlat:** *Címrendszer*
Készítsünk egy LaTeX-dokumentumot, mely megfelel a fenti sematikus ábrának. A dokumentumban mindenütt legyen valami (akár halandzsa szavakból álló) szöveg, ahol annak kell lenni! [E feladat megoldása 2 percnél ne vegyen több időt igénybe. Használjuk a szövegszerkesztő másolás–beillesztés (*copy–paste*) funkcióit. A szakaszok szintjén az első téglalap egy sorszám nélküli bevezető szakaszt jelöl.] Változtassuk meg a művet úgy, hogy az alszakaszok ne kapjanak sorszámot. ‹

**A főcím** ♦ Magának a dokumentumnak a címét a `\title` parancs argumentumaként adhatjuk meg. Ehhez társul az `\author` parancs (argumentumában a szerző(k) nevével) és a dátum megadását lehetővé tevő `\date` parancs. Ha `\date{}` alakban

hívjuk meg, a forrásállomány fordításának dátuma kerül a címbe. Arra, hogy egy adott helyen a megadott adatok meg is jelenjenek, a `\maketitle` parancs szolgál. Például így kezdhető egy könyv:

```
\title{Ezerkilencszáznyolcvannégy}
\author{George Orwell}
\date{1949}
\maketitle
```

A `\title`, `\author` és `\date` parancsok a preambulumba is írhatók, de a `\maketitle` csak a `\begin{document}` után!

› **Gyakorlat:** *Főcím*
Az előző dokumentumot egészítsük ki úgy, hogy legyen címe, szerzője és a címben szerepeljen a készítés dátuma! ‹

**Kivonat** ♦ Cikkek és beszámolók elején egy rövid kivonatban szokás tömören összefoglalni a dolgozat lényegét. Ezt a kivonat szövegének az `abstract` környezetbe zárásával tehetjük meg. E környezetet mindjárt a `\maketitle` parancs után tegyük. A kivonat `report` osztály esetén külön oldalon, cikk (`article` osztály) esetén a címlapon a cím alatt fog megjelenni.

**Függelék** ♦ Egy dolgozatba függeléket az `\appendix` paranccsal tehetünk, mely a fejezetszámlálót (cikknél szakaszszámlálót) lenullázza, és megjelenésének stílusát számról betűre változtatja. Így az első fejezet a függelékben az „A” jelzést kapja.

**Egy könyv részei** ♦ Egy könyv szerkezetének megvalósításához a `book` osztály három parancsot kínál fel: `\frontmatter`, `\mainmatter`, `\endmatter`. Ezek elhelyezkedése egy dokumentumban a következő:

```
\begin{document}
  cím
\frontmatter
  előszó, bevezetés, tartalomjegyzék
\mainmatter
  a dokumentum szövegének fő része a függelékkel
\backmatter
  irodalomjegyzék, mutatók
\end{document}
```

A `\fronmatter` római számokkal írja ki az oldalszámot, és nem ír sorszámot a fejezetek elé, `\mainmatter` arab számokkal írja ki az oldalszámot, és a számozást 1-től indítja, a `\backmatter` ismét eltünteti a sorszámot a fejezet címe elől.

## 2.5.2. Szavak

**A nyelv kiválasztása** ♦ Ahhoz, hogy a LaTeX tudja, hogyan kezeljen egy szót, például hogyan válassza el, tudnia kell, hogy az milyen nyelvhez tartozik. A használan-

dó nyelv(ek)et a babel csomag betöltésekor opcionális paraméterként soroljuk fel. Például egy angol, magyar és új helyesírás szerinti német szöveget is tartalmazó dokumentum preambulumába a következő írandó:

```
\usepackage[english,ngerman,magyar]{babel}
```

Az utolsónak felsorolt nyelv lesz az alapértelmezett, példánkban a magyar. Ha több nyelvet is megadtunk, egyikről a másikra a `\selectlanguage` paranccsal válthatunk, pl. a `\selectlanguage{english}` paranccsal az angol nyelvre, míg a `\selectlanguage{magyar}` paranccsal magyarra lehet váltani. Az `otherlanguage` környezettel is megadható egy szövegrész nyelve: a nyelv nevét a környezet argumentumában kell megadni. Csak egyetlen szó, vagy rövid kifejezés nyelvének átváltására való a kétargumentumos `\foreignlanguage` parancs. Például az angol „language'' szót magyar szövegben a `\foreignlanguage{english}{language}` paranccsal íratjuk ki. További információ a könyv 87. oldalán található.

**Szavak elválasztása** ♦ A szavakat a LaTeX automatikusan választja el a kiválasztott nyelv elválasztási szabályainak megfelelően.

A TeX-ben és a LaTeX-ben egyaránt használható magyar elválasztásfájl (huhyph.tex) nemcsak az általános elválasztási szabályokat ismeri, hanem sok olyan összetett szót is, mely kivételt képez e szabályok alól (tehát pl. tudja, hogy nem kö-té-li-deg, hanem kö-tél-i-deg a helyes elválasztás).

Előfordulhat, hogy egy szót a TeX rosszul választ el. Ez elkerülhetetlen, hisz mindig születnek újabb összetett szavak, és vannak olyan többjelentésű szavak, amelyek elválasztása a jelentéstől függ (pl. a „karóra'' helyes elválasztásai: kar-ó-ra, ka-ró-ra). A *puha elválasztójel* olyan jel, amely a szóban láthatatlan marad, csak azt a helyet mutatja, ahol a szót el szabad választani. Puha elválasztójelet a `\-` paranccsal tehetünk egy szóba. Például a

```
Már nem volt a szarkánál a kar\-ó\-ra, mikor felröpült a ka\-ró\-ra.
```

mondatban a „karóra'' szó mindkétszer jól lesz elválasztva, ha épp a sor végére kerül.

Ha egy szóban kötőjelek vannak, vagy az input fájlban puha elválasztójeleket is hozzáírunk, akkor a TeX csak a kötő-, illetve puha elválasztójeleknél fogja a szót elválasztani. Ezért a hosszú, kötőjeles kifejezésekbe érdemes puha elválasztójeleket is tenni, illetve érdemes egy szó minden lehetséges elválasztási helyét kijelölni. Az új magyar babelben megtalálható a `'-` rövidítés, amely lehetővé teszi az elválasztást mindkét oldalon. Például az `egyszer'-kétszer` szókapcsolat két szava között mindenképpen lesz elválasztójel, és a TeX szükség esetén elválaszthatja a szavakat külön-külön is.

Egy szó elválasztását az egész dokumentumra vonatkozó érvényességgel, a fájl elején is meg lehet adni a `\hyphenation` paranccsal. E parancs argumentumában szóközökkel elválasztva fel kell sorolni azokat a szavakat, amelyek elválasztását meg akarjuk változtatni. Itt egyszerű kötőjelet kell használni az elválasztás jelölésére. Például a

```
\hyphenation{ke-rék-pár-út LaTeX}
```

parancs hatására az egész dolgozatban jó lesz a „kerékpárút'' elválasztása, a „LaTeX'' szó pedig sehol sem lesz elválasztva.

Ha egy egész kifejezést szeretnénk egyben tartani úgy, hogy nemcsak a szavakon belül tiltjuk meg az elválasztást, de a szavak között sem engedjük meg a sortörést, akkor használjuk az `\mbox` parancsot. Az alábbi kód hatására a telefonszám nem fog a sor végén eltörni.

```
Telefonszámunk \mbox{(36)-(1)-551 3129}.
```

**Szavak és néhány szavas kifejezések kiemelése** ♦ Szavak, rövidebb kifejezések kiemelésére az `\emph` parancs szolgál. Azt, hogy e parancs hatására a kiemelendő szöveg milyen módon (pl. kurzív betűkkel) fog megjelenni, a dokumentumosztály, illetve a preambulumban megadott stílus dönti el. Az `\emph` parancsok egymásba ágyazhatók:

```
01 Ez a szó itt \emph{fontos}.
02 Egy hosszabb \emph{kiemelt
03 szövegből is \emph{ki lehet}
04 emelni egy részt}.
```

Ez a szó itt *fontos*. Egy hosszabb *kiemelt szövegből is* ki lehet *emelni egy részt*.

**Generált szavak** ♦ A LaTeX-ben több olyan parancs létezik, amelyiknek hatására egy-egy szó vagy rövid kifejezés íródik a dokumentum egy meghatározott helyére. Pl. a `\TeX`, a `\LaTeX`, illetve a `\LaTeXe` parancsokkal a TeX, a LaTeX és a LaTeX $2_\varepsilon$ szavakat kaphatjuk meg.

Vannak olyan szavak is, melyek a dokumentum egy adott helyén automatikusan jelennek meg, pl. a `\tableofcontents` parancs hatására nemcsak a forrásállományból összegyűjtött tartalomjegyzék lesz kiírva, de fölé kerül a cím is, hogy „Tartalomjegyzék''. Ezt a szót egy `\contentsname` nevű parancs tartalmazza.

Ha a `babel` csomagot használjuk a `magyar` opcióval, a LaTeX által automatikusan generált szövegeket is magyarul kapjuk meg (Tartalomjegyzék, fejezet, Függelék...).

Alapértelmezésben angol, ill. a `babel` csomaggal magyar nyelven a 2.2. táblázatban felsorolt kifejezéseket tudja automatikusan generálni a LaTeX. A táblázatban felsorolt nevek a `\renewcommand` paranccsal megváltoztathatóak, pl. „Tartalomjegyzék'' helyett „Tartalom'' lesz a tartalomjegyzék címe, ha a forrásállományban a `\tableofcontents` parancs előtt kiadjuk a következő parancsot:

```
\renewcommand{\contentsname}{Tartalom}
```

› **Gyakorlat:** *Kiemelt és generált szavak*
Az előző gyakorlatban készített dokumentumot változtassuk meg úgy, hogy *a)* legyen a dokumentum kétnyelvű, mondjuk magyar és angol, és legyen a magyar az alapértelmezett nyelv; *b)* egy szó és két bekezdés nyelvét változtassuk magyarról angolra; *c)* emeljük ki a dokumentum néhány szavát; *d)* készítsünk a dokumentumhoz tartalomjegyzéket, melyet tegyünk közvetlenül a főcím után, és amelynek ne „Tartalomjegyzék'', hanem „Tartalom'' legyen a címe! ‹

2.2. táblázat. A LaTeX által automatikusan generált szavak

| parancs | Angol | Magyar |
|---|---|---|
| \abstractname | Abstract | Kivonat |
| \alsoname | see also | lásd még |
| \appendixname | Appendix | függelék |
| \bibname | Bibliography | Irodalomjegyzék |
| \ccname | cc | Körlevél–címzettek |
| \chaptername | Chapter | fejezet |
| \contentsname | Contents | Tartalomjegyzék |
| \enclname | encl | Melléklet |
| \figurename | Figure | ábra |
| \glossaryname | Glossary | Szójegyzék |
| \headtoname | To | Címzett |
| \indexname | Index | Tárgymutató |
| \listfigurename | List of Figures | Ábrák jegyzéke |
| \listtablename | List of Tables | Táblázatok jegyzéke |
| \pagename | Page | oldal |
| \partname | Part | rész |
| \prefacename | Preface | Előszó |
| \proofname | Proof | Bizonyítás |
| \refname | References | Hivatkozások |
| \seename | see | lásd |
| \tablename | Table | táblázat |

### 2.5.3. Többmondatos részek kiemelése

**Idézetek** ♦ Idézetek kiemelésére a LaTeX két környezetet ajánl fel, a `quote`, valamint a `quotation` nevűeket. Az előbbit rövid – többnyire egymondatos – szövegekre, vagy ilyenek felsorolására, az utóbbit hosszabb, akár több bekezdésből álló szövegrészek kiemelésére ajánlatos használni. Angol szövegben nem, de magyarban ilyenkor is ki szokás tenni az idézőjelet:

```
Idézetek az idézetekről:
\begin{quote}
  ,,Utálom az idézeteket. Azt mondd, amit
  tudsz!'' \emph{Ralph Waldo Emerson}

  ,,Gyakran idézek magamtól, ez
  megfűszerezi beszélgetéseimet.''
  \emph{George Bernard Shaw}
\end{quote}   További\dots
```

Idézetek az idézetekről:

> „Utálom az idézeteket. Azt mondd, amit tudsz!” *Ralph Waldo Emerson*
>
> „Gyakran idézek magamtól, ez megfűszerezi beszélgetéseimet.” *George Bernard Shaw*

További…

**Versek** ♦ Versek szedésére használható a `verse` környezet. A vers sorait a \\ parancs választja el egymástól. A versszakok közé üres sort kell tenni.

```
\begin{verse}
  Megy a juhász szamáron,\\
  Földig ér a lába;\\
  Nagy a legény, de nagyobb\\
  Boldogtalansága.

  Gyepes hanton furulyált,\\
  Legelészett nyája.\\
  \dots
\end{verse}
```

Megy a juhász szamáron,
Földig ér a lába;
Nagy a legény, de nagyobb
Boldogtalansága.

Gyepes hanton furulyált,
Legelészett nyája.
...

**Tételszerű szövegrészek** ♦ A tételszerű szövegrészek olyan néhány mondatos – tartalmuk miatt tipográfiailag is kiemelt – részei a dokumentumnak, amelyek önálló címmel és legtöbbször sorszámmal is rendelkeznek. Ilyenek pl. a matematikai tételek, lemmák, definíciók vagy a jogi paragrafusok.

LaTeX-ben könnyű szövegrészeket tételszerűen kiemelni. Két dolgot kell tenni:

1. A dokumentumban (pl. a preambulumban) definiáljunk egy új környezetet a `\newtheorem` paranccsal. Legyen a következő példában ennek az új környezetnek a neve `tetel`.
2. A dokumentum szövegében a kiemelni szánt szöveget zárjuk az előzőleg definiált nevű környezetbe, tehát példánkban a `\begin{tetel}` és az `\end{tetel}` parancsok közé.

A `\newtheorem` parancsnak két argumentuma van, az első a környezet neve, példánkban `tetel`, a második a tétel címe. Ez az, ami a tételszerű környezet fejrészében meg fog jelenni. Ha angol nyelvű cikket írunk, akkor a matematikai tételek címe leggyakrabban „Theorem'', magyarul „tétel''. Az első `\begin{tetel}` sor előtt – pl. a preambulumban – ki kell adni a `\newtheorem{tetel}{tétel}` parancsot:

```
\newtheorem{tetel}{tétel}Pythagoras tétele:
\begin{tetel}
  A~derékszögű háromszög befogóira
  emelt négyzetek területösszege az
  átfogóra emelt négyzet területét adja.
\end{tetel}
Igen sok bizonyítását ismerjük\dots
```

Pythagoras tétele:

**1. tétel.** *A derékszögű háromszög befogóira emelt négyzetek területösszege az átfogóra emelt négyzet területét adja.*

Igen sok bizonyítását ismerjük...

Ha egy tételt külső forrásból idézünk, így kell feltüntetni a forrást:

```
%^\newtheorem{tetel}{tétel}
\begin{tetel}[{\cite
  [Pythagoras tétele]{geomk}}]
  $a^2+\allowbreak b^2= c^2$.
\end{tetel}
```

**1. tétel ([42, Pythagoras tétele])** $a^2 + b^2 = c^2$.

Egy dokumentumon belül több tételszerű környezetet is létrehozhatunk. Hozzunk létre például egy új „tétel'' és egy „definíció'' című tételszerű környezetet, és írjunk le két tételt és egy definíciót:

```
%^\newtheorem{tet}{tétel}
%^\newtheorem{dfc}{definíció}
\begin{tet}Ez az első tétel.\end{tet}
\begin{dfc}Ez az első definíció.\end{dfc}
\begin{tet}Ez a második tétel.\end{tet}
```

**1. tétel.** *Ez az első tétel.*

**1. definíció.** *Ez az első definíció.*

**2. tétel.** *Ez a második tétel.*

Látjuk, a tételek és a definíciók egymástól függetlenül sorszámozódnak. Ha több tételszerű környezetünk van, a sokféle számozás nem segíti, hanem inkább nehezíti a gyors tájékozódást, ezért gyakran sokkal jobb, ha csak egyetlen sorszámozás halad végig a művön. A `\newtheorem` parancs első argumentuma két dolgot jelöl: már tudjuk, hogy a tételszerű környezetnek ez lesz a neve, de egyúttal ez lesz a neve annak a számlálónak is, ami az adott tétel sorszámát szolgáltatja. Ha a `\newtheorem` parancs első argumentuma után egy opcionális argumentumba beírjuk egy másik tételszerű környezet számlálójának nevét, akkor az lesz e környezet számlálója is. Az alábbi példában a „tétel'' környezetekek neve és számlálója is `ttl`. A „definíció''-t létrehozó környezet neve `dfo`, de számlálója nem `dfo`, henem `ttl`, vagyis egyetlen számláló fog sorszámot adni a „tétel''-eknek és a „definíció''-knak is.

```
%^\newtheorem{ttl}{tétel}
%^\newtheorem{dfo}[ttl]{definíció}
\begin{ttl} Ez az első tétel.\end{ttl}
\begin{dfo} Ez az első definíció.\end{dfo}
\begin{ttl} Ez a második tétel.\end{ttl}
```

**1. tétel.** *Ez az első tétel.*

**2. definíció.** *Ez az első definíció.*

**3. tétel.** *Ez a második tétel.*

Van olyan gyakorlat, mely a tételek számozását fejezetenként újrakezdi, de a sorszám elé írja az adott fejezet sorszámát is! Ekkor úgy tekinthetjük, hogy környezetünk számlálója a fejezet számlálójának „alszámlálója'', hisz a fejezetszámláló értékének változásakor 1-től kezdve újraindítja a számlálást. Ilyen számlálójú környezetet úgy hozhatunk létre, hogy a `\newtheorem` parancsban a második argumentum után opcionális argumentumként megadjuk annak a fejezetszámlálónak a nevét, amelynek „alszámlálója'' lesz környezetünk saját számlálója. Az `article` dokumentumosztály esetén általában a `section`, könyvekben, azaz a `book` dokumentumosztályban a `chapter` vagy a `section` nevet szokás megadni. A következő példákat úgy szerkesztettük meg, mintha cikkről lenne szó. Példaként definiáljunk egy `Tet` nevű új környezetet! Ha valaki a következő példákat ki akarja próbálni, akkor az első tétel előtt adjon ki néhány `\section{Fejezet címe}` parancsot.

```
%^\newtheorem{Tet}{tétel}[section]
%^\newtheorem{Dfn}{definíció}
\begin{Tet}Ez az első tétel.\end{Tet}
\begin{Dfn}Ez az első definíció.\end{Dfn}
\begin{Tet}Ez a második tétel.\end{Tet}
```

**2.1. tétel.** *Ez az első tétel.*

**1. definíció.** *Ez az első definíció.*

**2.2. tétel.** *Ez a második tétel.*

Természetesen az így definiált számláló is átörökíthető úgy, hogy ezt az alábbi forráskód második sora is mutatja:

```
%^\newtheorem{Ttl}{tétel}[section]
%^\newtheorem{Dfo}[Ttl]{definíció}
\begin{Ttl}Ez az első tétel.\end{Ttl}
\begin{Dfo}Ez az első definíció.\end{Dfo}
\begin{Ttl}Ez a második tétel.\end{Ttl}
```

**2.1. tétel.** *Ez az első tétel.*

**2.2. definíció.** *Ez az első definíció.*

**2.3. tétel.** *Ez a második tétel.*

Egy `\newtheorem` parancsnak legfeljebb csak egy opcionális paramétere lehet. Ez logikus, hisz ha egy tételszerű környezet számlálója helyett egy másikét használjuk, akkor azt, hogy e másik melyik fejezetszinthez tartozik, ennek a másik környezetnek a definíciójában kell megadni.

› **Gyakorlat:** *Több tételszerű környezet sorszámozásának összehangolása*
Két tételszerű környezet létrehozásakor az opcionális paramétereknek milyen egyéb, fel nem sorolt variációja létezik, és az milyen eredményt ad a sorszámozásban? ‹

A tételszerű környezetek kezelésével kapcsolatos további kérdéseket a könyv 4.4. szakaszában vizsgáljuk.

## 2.5.4. Felsorolások, listák

Felsoroláson vagy listán a dokumentumunk egy olyan részét értjük, melyet több rövid, néhány szavas, de legfeljebb két-három rövid bekezdésnyi szakaszra osztunk, és ezt a felosztást tipográfiailag is megjelenítjük. Három legfontosabb típusa:

**Felsorolás,** melyben nincs jelentősége a felsorolt szakaszok sorrendjének, és minden szakaszt egy ún. felsorolásjel (pl. gondolatjel, díszpont, vagy valamilyen egyéb szimbólum) vezet be.

**Sorszámozott lista,** melyben szerepe van a sorrendnek (pl. utasítások felsorolásában, rangsorolásban), ezért ezt jelöljük is a szakaszok elé tett sorszámmal. E sorszám lehet arab szám, kis vagy nagy római szám, de kis- vagy nagybetű is, sőt lehet valami összetettebb címke is (pl. 1. szabály, 321. §).

**Szótárszerű felsorolás vagy leíró lista,** melyben minden szakaszt egy tipográfiailag is kiemelt kifejezés vezet be, mely után annak leírása következik. (Ez a lista például épp ilyen.)

A LaTeX alapértelmezésben úgy tekint a listára, mint amelynek szakaszai több bekezdésből is állhatnak, így a lista szakaszai a bekezdésénél is erősebb kiemelést kapnak. Előfordul, hogy listánkra úgy tekintünk, mint ami egyetlen bekezdés részeként jelenik meg. Ezeket *bekezdésen belüli* listáknak fogjuk nevezni. Abban az esetben, ha a bekezdésen belüli lista számozott lista, a listát sorfolytonosan is meg lehet jeleníteni. Lássunk ezekre példát!

**Felsorolás** ♦ Felsorolásokhoz a LaTeX `itemize` környezete használható. A bekezdésen belüli változatot a paralist csomag `compactitem` környezete nyújtja. Ha be van töltve a paralist csomag, akkor nemcsak az `compactitem`, hanem az `itemize` környezetnek is megadható egy opcionális paraméter, mellyel átállíthatjuk az alapértelmezés szerinti felsorolásjelet. A következő példában díszpont helyett gondolatjelet alkalmazunk, amit az opcionális `[--]` argumentummal érünk el.

```
A~\LaTeX\ a listatípusok mindegyikére
felkínál egy környezetet.
\begin{itemize}
\item Felsorolásokhoz az itemize
  környezet használható.
\item Számozott listákhoz az
  enumerate környezet való.
\item A~description környezettel
  leíró listákat készíthetünk.
\end{itemize}
```

A LaTeX a listatípusok mindegyikére felkínál egy környezetet.

- Felsorolásokhoz az itemize környezet használható.
- Számozott listákhoz az enumerate környezet való.
- A description környezettel leíró listákat készíthetünk.

```
%^\usepackage{paralist}
A~paralist csomag bekezdésen
belüli listát adó környezetei:
\begin{compactitem}[--]
\item compactitem a felsorolásokhoz,
\item compactenum a számozott
      listákhoz,
\item compactdesc a leíró listákhoz.
\end{compactitem}
```

A paralist csomag bekezdésen belüli listát adó környezetei:

–compactitem a felsorolásokhoz,
–compactenum a számozott listákhoz,
–compactdesc a leíró listákhoz.

**Számozott lista** ♦ *Sorszámozott listák* készítéséhez az `enumerate` környezet használható. Itt is csak az `\item` parancsot kell minden szakasz elejére beírni, a sorszámokról a LaTeX gondoskodik. Példán keresztül bemutatjuk, hogy a listakörnyezetek egymásba ágyazhatók. Alapértelmezésben az első szinten arab számok (1., 2.,...), a másodikon zárójelbe zárt kisbetűk ((a), (b),...), a harmadikon kis római számok (i., ii.,...), a negyediken nagybetűk (A., B.,...) jelennek meg.

```
\begin{enumerate}
\item Felsorolásokhoz használható az
itemize környezet.
\item Sorszámozott listákhoz való az
enumerate környezet. A~sorszámai lehetnek
  \begin{enumerate}
  \item arab számok,
  \item \begin{enumerate}
      \item kis római számok,
      \item nagy római számok,
        \end{enumerate}
  \item kis- és nagybetűk.
  \end{enumerate}
\item Leíró listák képzésére való a
description környezet.
\end{enumerate}
```

1. Felsorolásokhoz használható az itemize környezet.
2. Sorszámozott listákhoz való az enumerate környezet. A sorszámai lehetnek
   (a) arab számok,
   (b) i. kis római számok,
       ii. nagy római számok,
   (c) kis- és nagybetűk.
3. Leíró listák képzésére való a description környezet.

A bekezdésen belüli változatot a paralist csomag `compactenum` környezete nyújtja.

Ennek opcionális argumentumában a sorszámozás módja adható meg. Ha az argumentumban van egy jel (de csak egy) az 1, i, I, a, A jelek közül, akkor aszerint, hogy melyik az, rendre arab szám, kis római szám, nagy római szám, kisbetű, illetve nagybetű lesz a „sorszám''.

```
%^\usepackage{paralist}
Sorszámozott lista sorszáma
\begin{compactenum}[I.]
\item arab szám,
\item római szám vagy
\item betű
\end{compactenum}
egyaránt lehet.
```

Sorszámozott lista sorszáma
I. arab szám,
II. római szám vagy
III. betű
egyaránt lehet.

Ugyanez a példa sorfolytonos változatban is megjeleníthető az `inparaenum` környezet segítségével. E példában a sorszám egy kurzívan kiemelt kisbetű egy kerek zárójellel. Az `\itshape` paranccsal később ismerkedünk meg.

```
%^\usepackage{paralist}
Sorszámozott lista sorszáma
\begin{inparaenum}[\itshape a)]
\item arab szám,
\item római szám vagy
\item betű
\end{inparaenum}
egyaránt lehet.
```

Sorszámozott lista sorszáma *a)* arab szám, *b)* római szám vagy *c)* betű egyaránt lehet.

Egy hasonló példa szerepel a 157. oldalon, ahol megmutatjuk, hogyan hivatkozhatunk a számozott lista elemeire.

Ha olyan szöveget írunk a fenti opcionális paraméterbe, amelyben több is szerepel az 1, i, I, a, A jelek közül, akkor zárjuk kapcsos zárójelek közé azokat, amelyeket nem akarunk a lista számlálójává tenni.

› **Gyakorlat:** *Számozott listák készítése*
Készítsünk egy három feladatból álló sorszámozott feladatsort! Legyen a feladatok sorszáma

*a)* „(1)'', „(2)'', „(3)'', illetve
*b)* „1. feladat:'', „2. feladat:'', „3. feladat:''.

Ezután, megtartva az *a)* pontban megadott sorszámozást, adjuk meg a feladatsor eredményeit egy bekezdésen belüli és egy sorfolytonos változatban is! Vizsgáljuk meg, hogy az `\item` parancsnak e környezetekben adható-e opcionális paraméter, és az mire használható! ‹

**Szótárszerű felsorolás, leíró lista** ♦ Leíró listában (más néven szótárszerű vagy lexikonszerű felsorolásban) a listaelemek címe az `\item` parancs opcionális paramétereként adható meg:

```
\begin{description}
\item[Felsorolás] készítésére szolgál az
  itemize környezet.
\item[Sorszámozott lista] készíthető
  az enumerate környezettel.
\item[Leíró lista] képzésére való a
  description környezet.
\end{description}
```

**Felsorolás** készítésére szolgál az itemize környezet.
**Sorszámozott lista** készíthető az enumerate környezettel.
**Leíró lista** képzésére való a description környezet.

Hasonlóképpen a listák többi típusánál tárgyaltakhoz, a paralist csomag compactdesc környezete bekezdésen belüli leíró listát ad.

A listák testreszabásáról részletesen a 4.3. fejezetben írunk.

## 2.5.5. Jegyzetek

**Lábjegyzet** ♦ Lábjegyzetet a `\footnote` paranccsal készíthetünk. Argumentumába kerül a lábjegyzet szövege. A `\footnote` parancsot közvetlenül az után a szó után írjuk szóköz kihagyása nélkül, amelyre a lábjegyzet vonatkozik.

```
Egy farmer elad egy zsák búzát
10\$-ért\footnote{1965-ös árfolyamon 130
forintért.}. Költségei az eladási ár
$\frac45$-ét teszik ki. Mennyi a
haszna?\footnote{E feladat nem Laricsev
példatárából való.}
```

Egy farmer elad egy zsák búzát 10$-ért[1]. Költségei az eladási ár $\frac{4}{5}$-ét teszik ki. Mennyi a haszna?[2]

[1] 1965-ös árfolyamon 130 forintért.
[2] E feladat nem Laricsev példatárából való.

Alapértelmezésben a lábjegyzet helyét a jegyzettel ellátott szó után felső indexbe tett kis jel jelzi. Az `article` osztályban e jelek számok, és folyamatosan növekednek, míg a `book` és `report` osztályban fejezetenként 1-től kezdődnek. Egyéb jeleket is használhatunk a lábjegyzetek megjelölésére (lásd a könyv 4.5.1. szakaszában).

Ha a lábjegyzet egy egész mondatra vagy mondatok sorára vonatkozik, akkor a lábjegyzetet jelző számot a mondatot záró írásjel után írjuk. Ha a lábjegyzet csak egy szóra vagy kifejezésre vonatkozik, a szám az írásjel elé kerül. Ezt mutatja a fenti példa is.

A címrész parancsainak, azaz a `\title`, `\author` vagy a `\date` parancsok valamelyikének argumentumába a `\thanks` paranccsal tehetünk lábjegyzetet.

› **Tipp:** *Lábjegyzetek használata*
Kezdő LaTeX-felhasználóként kerüljük el, hogy lábjegyzetbe különleges parancsokat írjunk. Lábjegyzetet csak normál szövegbe helyezzünk, parancsok argumentumába, matematikai módú szövegbe ne! ‹

**Széljegyzet** ♦ Széljegyzet elhelyezésére a `\marginpar` parancs szolgál, melynek argumentuma lesz a széljegyzet szövege. A `\marginpar{Margó!}` parancs hatása a margón látható. A széljegyzet a `\documentclass` parancs `oneside` opciójának használatakor, azaz egyoldalas szedés esetén mindig a jobb margóra kerül, a `twoside` opció használatakor, azaz kétoldalas szedés esetén páratlan oldalon a jobb, páros oldalon a bal margóra kerül.

Margó!

Ha kétoldalas szedésnél mást akarunk írni a páratlan oldal jobb, és mást a páros oldal bal margójára, akkor a bal margó szövege opcionálisan megadható a `\marginpar[`*bal margó*`]{`*jobb margó*`}` formában.

Például a

```
\marginpar[\raggedleft$\triangleright$]{$\triangleleft$}
```

parancs mindig a külső margóra tesz egy, a szöveg felé mutató háromszöget, azaz egy ◃ jelet a páratlan oldal jobb margójára, és egy ▹ jelet a páros oldal bal margójára. A széljegyzet mindig egy adott szélességű sávban jelenik meg (lásd a `\marginparwidth` parancsot a 4.5.2. alszakaszban). A `\raggedleft` parancs a fenti példában arról gondoskodik, hogy a háromszög a bal oldalon is a szöveg mellett jelenjen meg, azaz a bal margó jobb szélén.

## 2.5.6. Utalások

Gyakran szoktunk a folyó szövegben a dokumentum egy másik helyére *utalni*, vagy egy másik műre, vagy annak valamely részére *hivatkozni*.

Először a művön belüli utalásokkal foglalkozunk. Az utalásokban leggyakrabban az adott részt egyértelműen jellemző valamely sorszámot használjuk (pl. a 16. oldalon, a 3. fejezetben, 4. ábra, 2.1. tétel, a fenti lista 3. pontja szerint…). A LaTeX képes az utalásban használt számot meghatározni, ha azt a részt, amire utalunk, megjelöljük. Erre szolgál a `\label` parancs, melynek egyetlen argumentuma egy címke, amelyet majd az utalásban használunk, s amelyből a LaTeX meghatározza az adott részt jellemző számot. Egy címkével megjelölt hely oldalszámára a dokumentum bármely helyéről a `\pageref` paranccsal hivatkozhatunk. A folyószöveg minden egyéb – a LaTeX által automatikusan megsorszámozott – részére a `\ref` paranccsal utalunk.

```
01 Ide\label{cimke} tettünk egy címkét.
02 Ennek helye: \pageref{cimke}.\ oldal,
03 \ref{cimke}.\ alszakasz.
```

Ide tettünk egy címkét. Ennek helye: 41. oldal, 2.5.6. alszakasz.

Ha magyar szöveget írunk, akkor ügyelnünk kell az utalás előtt szereplő határozott névelőre. A dokumentum átszerkesztése nyomán egy utalás „*a*” 4. oldalról átcsúszhat „*az*” 5. oldalra. Szerencsére a magyar `babel` jó megoldást kínál: `\aref` és `\apageref` parancsai a megfelelő határozott névelőt is kiteszik a szám elé. Mondat elején e parancsok `\Aref` és `\Apageref` változatait használjuk, melyek nagybetűvel kezdik a határozott névelőt.

```
01 Ezt a helyet\label{c1} megjelöltük egy
02 címkével, ami \apageref{c1}.\ oldalon,
03 \aref{c1}.\ sorszámú alszakaszban van.
```

Ezt a helyet megjelöltük egy címkével, ami a 41. oldalon, a 2.5.6. sorszámú alszakaszban van.

Hierarchikusan sorszámozott objektumoknál a LaTeX mindig a legalsó szint sorszámát írja ki. Hogy például a 2.5.6. alszakasz helyett a 2.5. szakaszra utalhassunk, ahhoz az kell, hogy legyen egy címke az alszakasz egy olyan helyén, mely minden paragrafuson kívül van. Erre a célra legjobb, ha a címkét mindjárt a cím után tesszük. Ennek a szakasznak például így adtuk meg a címét:

```
\section{A főszöveg és járulékos részei}\label{sec:jarulekos}
```

Ábrára, táblázatra, listára, tételre… való hivatkozáshoz az objektumot megadó környezeten belülre írjuk a címkét.

```
\begin{tetel}\label{thm:eloszlas}
  Az eloszlásfüggvény
  \begin{enumerate}
  \item balról folytonos,\label{lst:bal}
  \item monoton növekvő. \label{lst:mon}
  \end{enumerate}
\end{tetel}
\Aref{thm:eloszlas}.~tétel
\ref{lst:bal}.~pontja szerint\dots
```

**1. tétel.** *Az eloszlásfüggvény*

*1. balról folytonos,*
*2. monoton növekvő.*

Az 1. tétel 1. pontja szerint…

Angol szövegben az oldal, fejezet, táblázat, ábra vagy listaelem sorszáma után általában nem kell pontot tenni, magyarban azonban igen, hacsak nem teszünk a sorszámok közé nagykötőjelet (például „a 2–4. fejezetben'', „a 3.1–4. szakaszban'').

› **Tipp:** *Milyen legyen a címke*
A címkéknek adjunk könnyen megjegyezhető nevet. Érdemes a címke nevébe olyan információt is beírni, amiből tudható, hogy a címke milyen típusú objektumra hivatkozik. Kialakult szokás a helyet jellemző parancs vagy környezet kezdőbetűit prefixként a címke elejére rakni, amit egy kettőspont választ el a címke további részétől. Fejezetcím mellett a címke kezdete `cha`, szakaszcímnél `sec`, táblázatnál `tab`, ábránál `fig`, egyenletnél `eq`, matematikai tételeknél `thm`. E szokásnak megfelelő címkék például a következők: `sec:Bevezetes`, `tab:kiadas`, `eq:Euler`, `thm:Cauchy`. Bár szabály nem tiltja, címkékben kerüljük az ékezetes betűk használatát! ‹

## 2.5.7. Irodalmi, szakirodalmi hivatkozások

Egy másik műre hivatkozhatunk úgy, hogy a hivatkozás helyén megadjuk a mű bibliográfiai adatait, melyet zárójelbe vagy lábjegyzetbe teszünk, és úgy is, hogy a hivatkozott mű adatai a dokumentum egy elkülönített részében, a többi hivatkozott mű adataival együtt, az ún. irodalomjegyzékben van megadva. Mi csak ez utóbbi esettel foglalkozunk.

**Irodalomjegyzék** ♦ Az irodalomjegyzék a dokumentum egy listaszerű része. Általában e lista minden tagja előtt egy címke áll, ami leggyakrabban egy szám, vagy egy szögletes zárójelbe tett szám (pl. 5 vagy [5]), de lehet a szerző neve (pl. [Knuth]), vagy a szerző nevéből és a cikk évszámából képzett kifejezés (pl. [Knuth84]), vagy bármi más, az adott irodalmat (cikket, könyvet stb.) egyértelműen azonosító karakterlánc (pl. [DEK84]). Az imént felsorolt lehetőségek közül való választás nem a *szerző* feladata, az a stílus része.

A LaTeX az irodalmi hivatkozások kezelésére két módszert kínál fel, az első a `thebibliography` környezettel készített listára, a másik egy – BibTeX nevű – egyszerű bibliográfiai adatbázis-kezelőre épül. Először az első módszert tanulmányozzuk.

**A thebibliography környezet használata** ♦ Irodalomjegyzéket a `thebibliography` környezettel hozhatunk létre, amelyben minden irodalmi hivatkozást egy-egy `\bibitem` parancs vezet be. Mint a következő példából is látható, a bibliográfiai adatok vizuális megjelenését itt a szerző szabályozza, ami nagy hátránya e megoldásnak:

```
\begin{thebibliography}{9}
\bibitem{texbook} Donald E. Knuth,
\textit{The \TeX book},
Addison-Wesley, Reading, 1984.
\bibitem{latexbook} Leslie Lamport,
\textit{\LaTeX\ A Document Preparation
System}, 2nd edn., Addison-Wesley, 1994.
\end{thebibliography}
```

[1] Donald E. Knuth, *The TEXbook*, Addison-Wesley, Reading, 1984.
[2] Leslie Lamport, *LATEX A Document Preparation System*, 2nd edn., Addison-Wesley, 1994.

A `thebibliography` környezetnek egy argumentuma van. Ebben egy olyan karakterláncot kell megadni, amelynél hosszabb az irodalomjegyzék tagjainak címkéjében nem fordul elő. Ha például a jegyzék címkéiben csak egyjegyű számok szerepelnek, akkor elég egy számjegyet, pl. a 9-est ide írni, ha azonban 9-nél több tétel van felsorolva, azaz a címkék kétjegyű számok is lehetnek, akkor egy kétjegyű számot, pl. a 99-est kell az argumentumba írni.

Mint a fenti példán látható, a `\bibitem` parancsnak egy argumentuma van. Ebbe egy tetszőleges karakterlánc írható, ez lesz az a kulcs, amellyel egy adott műre hivatkozni lehet a dokumentumon belül. Ha például Knuth könyvére szeretnénk hivatkozni, akkor ezt a `\cite{texbook}` paranccsal tehetjük meg (hisz ehhez a könyvhöz a `\bibitem{texbook}` parancs tartozik, tehát `texbook` a kulcs), ennek eredményeként a következő jelenik meg a szövegben: [1]. Általában is igaz, hogy a `\cite` parancs hatására az irodalomnak az a címkéje jelenik meg a szövegben, ami az irodalomjegyzékben mellette szerepel. Ez alapesetben a LaTeX által automatikusan generált szám, körülötte szögletes zárójellel vagy utána ponttal. A `\bibitem` opcionális argumentumával saját címkét használhatunk az automatikusan generált helyett. Ha például egy `thebibliography` környezetben a

```
\bibitem[Knuth]{texbook} Donald E. Knuth, \textit{The \TeX book},
Addison-Wesley, 1984.
```

hivatkozás szerepel, akkor a szövegben kiadott `\cite{texbook}` parancs helyén nem [1], hanem [Knuth] jelenik meg, és ugyanez jelenik meg az irodalomjegyzékben is. Ha ilyen címkét használunk, akkor a `thebibliography` környezet argumentumában is ennek megfelelő széles karakterláncot érdemes megadni, például így:

```
\begin{thebibliography}{Knuth}
```

Sajnos itt a szerzőnek vizuális döntést kell hoznia! Javasoljuk, hogy a `\thebibliography` parancs argumentumába legfeljebb 4-5 hosszú karakterláncot írjunk. Ha a hivatkozáshoz más információt is hozzá akarunk adni, akkor azt a `\cite` parancs opcionális argumentumával tehetjük meg. Például azt, hogy „lásd Knuth könyvében [1, 160. oldal]” a következő paranccsal kaptuk meg:

```
lásd Knuth könyvében \cite[160.~oldal]{texbook}
```

A magyar babel-ben a \cite parancsnak is van \acite és \Acite változata, mely automatikusan kiválasztja a hivatkozás elé a megfelelő határozott névelőt.

**A BibTEX használata ♦** Egy téma elmélyült kutatása közben komoly irodalmi lista gyűlhet össze. Egy ilyen bibliográfiai adatbázis létrehozására és kezelésére készült a BibTEX, melyet Oren Patashnik írt, és amely része minden TEX-disztribúciónak. A BibTEX-hel nemcsak az adatbázis adatai kezelhetők, de az irodalomjegyzék megjelenésének stílusa is. Számtalan kiadó és szakfolyóirat stílusállománya áll a szerző rendelkezésére, így – ellentétben a bibliography környezettel – itt nem kell a vizuális megjelenést programozni.

A BibTEX lényege, hogy a bibliográfiai adatok egy adatbázisba kerülnek, ami valójában egy egyszerű, kézzel is szerkeszthető ASCII állomány. Kiterjesztése .bib. A LATEX innen olvassa ki azokat az adatokat, amikre a dokumentumban hivatkozunk. Ezt az adatbázist bárki elkészítheti magának, de az interneten sok téma sokezernyi adatot tartalmazó adatbázisa megtalálható és bárki által használható.

Lássunk egy példát! Legyen az alábbi állomány neve b_konyv.bib:

```
@String{pub-AW = "Ad{\-d}i{\-s}on-Wes{\-l}ey"}

@Book{Knuth:ct-a,
  author       = "Donald E. Knuth",
  title        = "The {\TeX}book",
  publisher    = pub-AW,
  year         = "1986",
}

@Book{Lamport:LDP86,
  author       = "Leslie Lamport",
  title        = "{\LaTeX}---{A} Document Preparation System---%
                  User's Guide and Reference Manual",
  publisher    = pub-AW,
  year         = "1985",
}
```

Ebben a bib-fájlban két könyv bibliográfiai adatai szerepelnek. Mindkettőt a @book parancs vezeti be (itt nem számít különbözőnek a kis- és nagybetű, írható @BOOK vagy @Book is). Számtalan további dokumentumfajta adatai írhatóak le hasonló parancsokkal (teljes listájuk és részletes magyarázat a könyv 432. oldalán): @article (cikk), @book (könyv), @inbook (könyvrészlet), @inproceedings (konferenciaközleményben megjelent cikk), @masterthesis (diplomamunka), @phdthesis (doktori disszertáció), @proceedings (konferenciaközlemény), @unpublished (még kiadatlan). E speciális formátumú parancsok első argumentuma egy címke, erre lehet hivatkozni a cikkből a \cite paranccsal. A további argumentumok *mezőnév* = ''valami'', esetleg *mezőnév* = {valami} vagy *mezőnév* = rövidítés alakúak. A *mezőnév* lehetséges változatainak teljes listája a könyv 433. oldalán található, itt csak a legfontosabbakat soroljuk fel: author (szerző), editor (szerkesztő), title (cím), publisher (kiadó), address (a kiadó címe), year (kiadás éve), journal (folyóirat neve), volume

(kötetszám pl. folyóiratnál), pages (oldalak), series (sorozat neve). A használható mezőnevek függnek a dokumentum típusától, pl. a journal mezőnevet `@article`, azaz cikk esetén használjuk, de `@book` esetén nem.

A fenti mintaállomány mutatja a rövidítések megadására szolgáló `@string` parancs használatát is. Ennek argumentumába *rövidítés* = "*szöveg*" alakú kifejezések írhatók.

**A BibTEX futtatása ♦** Készítsük el az alábbi LATEX-állományt, legyen `b_latex.tex` a neve:

```
\documentclass{article}
\begin{document}

Books of Knuth \cite{Knuth:ct-a} and Lamport \cite{Lamport:LDP86} are useful.

\bibliography{b_konyv}
\bibliographystyle{plain} %%% abbrv, alpha, apalike, unstr

\end{document}
```

Két új parancs szerepel a forrásban: a `\bibliography` parancs argumentumába azt, vagy vesszővel elválasztva azokat a fájlnev(ek)et írjuk, mely(ek) a bibliográfiai adatokat tartalmazzák (esetünkben `b_konyv.bib`). A másik parancs a `\bibliographystyle`, melynek argumentumába a bibliográfiai stílust leíró állomány nevét írjuk (a kiterjesztés nélkül). E stílusállomány gondoskodik a tipográfiai megjelenítés milyenségéről, amibe nemcsak olyasmik tartoznak, mint a betűtípus meghatározása, hanem pl. az is, hogy csupa nagybetűvel kezdődjön a cím minden szava, vagy hogy a szerző nevében a keresztnévnek csak a kezdőbetűi szerepeljenek vagy a teljes név, vagy hogy a vezetéknév kerüljön előre vagy a keresztnév stb. Mi a `plain` stílust használjuk. A legfontosabb stílusok és tulajdonságaik:

`plain` Ez az alapstílus. A bibliográfiai adatok előtt szögletes zárójelpárba zárt számok jelennek meg, az adatok a szerzők vezetékneve szerint vannak rendezve, azonos név esetén a publikáció éve szerint.

`huplain` A `plain` stílus magyar tipográfiát követő változata (lásd még 8.3.10. alszakaszban).

`abbrv` A `plain` stílustól abban különbözik, hogy több mindent, így a szerzők és a folyóiratok nevét is rövidíti.

`alpha` Ez számok helyett egyéb címkéket használ a hivatkozáshoz.

`unsrt` A rendezés itt az előfordulás sorrendjében történik, és nem a szerző neve szerint.

A TEX-disztribúciókban és az interneten több száz egyéb stílusállomány található.

Mielőtt lefordítanánk a fenti LATEX-állományt, gondoskodjunk arról, hogy az előzőleg megírt `b_konyv.bib` nevű állomány is legyen ugyanebben a könyvtárban. Az első fordítás, azaz egy

```
latex b_latex
```

parancs kiadása után a LaTeX hiányol egy `b_latex.bbl` nevű fájlt. Ezt a

```
bibtex b_latex
```

parancs kiadása után kapjuk meg, a `bibtex` nevű program hozza létre. (Aki grafikus felhasználói felületű programot használ, lehetséges, hogy a `bibtex` futtatását egy megfelelő gombra való kattintással tudja elindítani.) Ezek után a LaTeX-et tipikus esetben még legalább kétszer újra kell futtatni: először a bibliográfiai adatok kerülnek az outputban a helyükre, majd az ezekre való hivatkozások sorszámai is bekerülnek a szövegbe.

**Nevek és címek a BibTeX-ben** ♦ A BibTeX úgy tekint a névre, hogy az négy részből áll: vezetéknév, keresztnév, von-név, Jr-név. A vezetéknév és a keresztnév nagy kezdőbetűvel írandó, a von-nevek kicsivel. A Jr-név rangot, családi állapotot vagy családi viszonyt kifejező szó vagy rövidítés (pl. Özv., Ifj., Jr.), melyet a BibTeX a vezetéknév elé, vagy vesszővel elválasztva a keresztnév után ír. Név BibTeX-ben az alábbi módokon adható meg:

| | |
|---|---|
| "*Keresztnév von-név Vezetéknév*" | pl. `"Ludwig van Beethoven"` |
| "*von-név Vezetéknév, Keresztnév*" | pl. `"van der Waerden, Bartel Leendert"` |
| "*von-név Vezetéknév, Jr, Keresztnév*" | pl. `"King, Jr, Martin Luther"` |

Mind a négy névrész több szóból is állhat, vagy üres is lehet. Ha az első alakban csak vezeték- és keresztnév szerepel, akkor az utolsót tekinti vezetéknévnek, a többit keresztnévnek (pl. `"Anna Zsuzsa Pataky"`). Kapcsos zárójel használandó, ha a vezetéknév áll több szóból, pl. `"Anna {Pataky Kis}"`. Ha több szerzőről van szó, az `and` szót kell a nevek közé tenni.

Ha repülő ékezetes betűket használunk, vagy különleges betűket paranccsal megadva, akkor azokat kapcsos zárójelbe kell zárni (l. Szőnyi Tamás nevét a következő példában)! Ha pl. latin2-es betűket akarunk használni a BibTeX adatbázisban, akkor a LaTeX-állományban gondoskodni kell a megfelelő csomagok betöltéséről. Kapcsos zárójelbe kell tenni a címekben azokat a nagybetűket, amelyeknek mindenképp nagybetűsnek kell maradniuk (ilyenek pl. a címben szereplő nevek első betűi, mert a stílustól függően a BibTeX a nagybetűket kicsire változtathatja). A következő példában a címbeli Galois szónak mindenképp nagy betűvel kell kezdődnie:

```
@preamble{"\newcommand{\Zs}{Zs}"}
@article{BSzW,
  author  = "Aart Blokhuis and Sz{\H{o}}nyi, Tam{\'a}s and Weiner, {\Zs}uzsa",
  title   = "On Sets without Tangents in {G}alois Planes of Even Order",
  journal = "Designs, Codes and Cryptography",
  year    = "2003",
  volume  = "29",
  number  = "1/3",
  pages   = "91--98"}
```

A magyar keresztnevekkel rövidítéskor baj lehet, mivel a BibTeX csak az első betűt használja rövidítésre. Ennek megoldására ad egy ötletet a fenti példa. A `@preamble` paranccsal TeX-utasítások írhatók az adatbázis állományba. Az első sorban defini-

álunk egy \Zs nevű utasítást, mely kiír egy „Zs'' betűt. Ezután a Zsuzsa név első eleme nem a „Z'' betű, hanem a \Zs parancs lesz, így pl. az abbrev stílusban a harmadik szerző neve Zs. Weiner lesz és nem Z. Weiner.

Ha egy interneten elérhető mű adatait akarjuk a BibTeX adatbázisba vinni, akkor a publisher és az address mezőkkel ezt megtehetjük, pl.:

```
publisher  = "World Wide Web",
address    = "http://mek.oszk.hu/00700/00707/",
```

Az address mezőben az \url parancs "\url{http://mek.oszk.hu/00700/00707/}" alakban használható, de akkor a LaTeX állományban gondoskodni kell az url csomag betöltéséről is. A másik lehetőség, hogy olyan BibTeX-stílust használunk, mely ismeri az url-mezőt (pl. a huplain):

```
url        = "http://mek.oszk.hu/00700/00707/",
```

› **Gyakorlat:** *BibTeX*
Készítsünk egy téma irodalmi adataiból BibTeX adatbázist, és egy rövid írást, melyben ezekre hivatkozunk! Az írásban legyenek utalások, az adatbázisban legyenek internetcímek. Használjunk interneten elérhető BibTeX-forrásokat is! ‹

## 2.5.8. Jegyzékek

A dokumentum járulékos részeinek némelyikét a LaTeX képes a főszövegből kiolvasva összeállítani. Ilyen pl. a tartalomjegyzék, az ábrák vagy táblázatok jegyzéke, és ilyen a következő alszakasz témája, a tárgymutató is. Az ebben a szakaszban tárgyalt mindegyik jegyzék készítésekor tartsuk szem előtt, hogy az aktuális állapotot csak két egymás utáni fordítás után tükrözik. (A magyarázatot lásd az egyes jegyzékek ismertetésénél.)

**Tartalomjegyzék készítése** ♦ Tartalomjegyzék készítése igen egyszerű, a dokumentum megfelelő helyére – általában az elejére vagy a végére – a \tableofcontents parancsot kell írni (*table of contents* jelentése tartalomjegyzék). E parancs hatására az első fordítás során a fejezetcímek egy toc kiterjesztésű fájlba íródnak (a fájl neve azonos a lefordított .tex fájl nevével), ahonnan a második fordításkor tartalomjegyzékként beszerkesztődnek az outputba (pl. a .dvi fájlba).

**A tartalomjegyzék mélysége** ♦ A tocdepth számláló beállításával lehet befolyásolni azt, hogy a tartalomjegyzékbe milyen szintű fejezetcímek íródjanak be. A szintszámok ugyanazok, mint amelyeket a fejezetekről írva megadtunk (l. 30. oldal, l. még E.18. táblázat). Például, ha azt akarjuk, hogy egy cikk paragrafusainak címe is bekerüljön a tartalomjegyzékbe (a részek, a szakaszok, az alszakaszok és az al-alszakaszok címe mellett), de az alparagrafusok címe ne, akkor adjuk ki a

```
\setcounter{tocdepth}{4}
```

parancsot a preambulumban (a paragrafus szintszáma 4). Csak a rész- és a szakaszcímek jelennek meg, ha a \setcounter{tocdepth}{1} parancsot használjuk.

**Ábrák és táblázatok jegyzéke** ♦ A tartalomjegyzéket készítő `\tableofcontents` parancshoz hasonlóan az ábrákról és a táblázatokról is készíthető jegyzék. Az ábrák jegyzéke a `\listoffigures` parancs hatására készül el, és oda kerül a szövegben, ahol e parancsot kiadjuk. Hatására egy azonos nevű, de `lof` kiterjesztésű fájl keletkezik. A táblázatok jegyzékét a `\listoftables` paranccsal kaphatjuk meg, mely egy `lot` kiterjesztésű fájlt hoz létre. E két parancs hatására a `figure` és `table` környezetek `\caption` parancsába írt ábrafelirat, ill. táblázatcím, valamint a sorszámuk és oldalszámuk kerül a jegyzékbe. A `figure` és `table` környezetekről részletesen írunk a 6.3. szakaszban.

**A jegyzékek szerkesztése** ♦ Ha a tartalomjegyzékbe mást akarunk írni, mint ami a valódi fejezetcím, pl. a cím egy rövidített változatát, akkor a fejezetcímet adó parancs opcionális paraméterét használjuk. Ha egy szakasz címe az, hogy „A matematikaoktatás fejlődése egy feladat változásai tükrében'', a rövid cím pedig az, hogy „A matematikaoktatás fejlődése'', akkor a

```
\section[A matematikaoktatás fejlődése]
  {A matematikaoktatás fejlődése egy feladat változásai tükrében}
```

parancsot alkalmazzuk. Ekkor ez a rövid cím fog bekerülni az élőfejbe is (az élőfejről a 198. oldalon lesz szó).

Ha egy fejezetkezdő parancs valamelyikének *-os változatát használjuk (l. 30. oldal), a cím nem kap sorszámot, de nem kerül a tartalomjegyzékbe sem. Ilyenkor egy külön paranccsal tehető oda be. Egy sort a tartalomjegyzékhez adni (angolul: *add (to) contents (a) line*) az `\addcontentsline` paranccsal lehet:

```
\section*{Előszó}
\addcontentsline{toc}{section}{Előszó}
```

Az `\addcontentsline` parancs első paramétere arra utal, hogy a tartalomjegyzék (angolul ***t**able **o**f **c**ontents*) egy `.toc` kiterjesztésű fájlba kerül. A `section` azt jelenti, hogy a szöveg úgy kerüljön ebbe a fájlba, mint egy szakasz címe, azaz ugyanolyan méretű, típusú stb. betűkkel. A parancs harmadik argumentumába kerül az a szöveg, amit a tartalomjegyzékbe szánunk, példánkban ez az „Előszó''. Ugyanígy vihetünk be egy sort az ábrák vagy a táblázatok jegyzékébe is, de ott az első két argumentum ábra esetén kötelezően `{lof}{figure}`, táblázatnál `{lot}{table}`.

Az `\addtocontents` paranccsal tetszőleges szöveg, illetve LaTeX-parancs vihető a jegyzékbe. Az `\addtocontents` első argumentumába az állomány típusa (`toc`, `lof`, `lot`), második argumentumába szöveg, vagy akár végrehajtható parancs is írható, mely a `.toc`-, `.lof`- vagy a `.lot`-állományba kerül. Például új oldalt kezdhetünk a tartalomjegyzékben az

```
\addtocontents{toc}{\newpage}
```

paranccsal. Hasonlóképp egyéb parancsok is kiadhatók, grafikát illeszthetünk a jegyzék sorai közé stb.

› **Gyakorlat:** *Tartalomjegyzék*
Tegyünk a 2.5.1. alszakaszban készített mű elejére tartalomjegyzéket, ebben legyen

benne a Bevezető is! Készítsünk két változatot, az elsőbe kerüljön bele az alszakaszok címe, a második változatból maradjon ki! ‹

## 2.5.9. Mutatók

Tárgymutató, névmutató, szógyűjtemény vagy index alatt a dokumentumból kigyűjtött és ábécé sorrendbe szedett szavak gyűjteményét értjük. Glosszáriumban általában a kigyűjtött szavakhoz még rövid magyarázat is társul. LaTeX-ben nagyon könnyű az ilyen szógyűjtemények készítése. Az első lépésben ki kell jelölni azokat a szavakat, amelyeket a gyűjteménybe akarunk írni. Erre való az `\index` parancs. Az `\index` argumentumába nemcsak kifejezések, hanem parancsok is kerülhetnek (még a `\verb` parancs is, amit egyébként nem lehet más parancs argumentumába tenni). Tehát az `\index{latex}` vagy az `\index{\LaTeX}` parancs egyaránt érvényes, az első esetben a „latex'', a másodikban a „LaTeX'' szó kerül az indexbe. Még a %-ot is beírhatjuk az indexbe, például az `\index{semmi%valami}` parancs hatására a „semmi'' bekerül az indexbe, de a „valami'' nem, az már megjegyzésnek számít. Minthogy az `\index` parancs ezt csak úgy tudja megvalósítani, hogy a parancs neve utáni { zárójel párját keresi, és addig bármilyen karakterláncot elfogad, ezért az `\index` argumentumában csak párosával szerepelhetnek a kapcsos zárójelek. Tehát az `\index{\{}` vagy a `\index{\}}` parancs helytelen, de az `\index{\{\}}` helyes.

Ahhoz, hogy a LaTeX foglalkozzon az indexekkel, a preambulumban ki kell adni a

```
\makeindex
```

parancsot. Ez az `\index` parancsok argumentumát kiírja egy `idx` kiterjesztésű fájlba: ha a `file.tex` forrásállományban a LaTeX talál egy `\index{`*szöveg*`}` parancsot, akkor a `file.idx` fájlba az `\indexentry{`*szöveg*`}{`*n*`}` parancs kerül, ahol *n* az `\index` parancs kiadásának oldalszáma. Ennek a `file.idx` fájlnak a rendezését, átalakítását a `makeindex` programmal lehet elvégezni. Ez létrehoz egy `ind` kiterjesztésű fájlt, példánkban a `file.ind` fájlt. E program futtatásához csak annyit kell az operációs rendszerünk parancssorába írni, hogy

```
makeindex file.idx
```

Végül gondoskodni kell arról is, hogy az elkészült index fájl tartalma bekerüljön a dokumentumba. Ehhez arra a helyre, ahol a tárgymutatót szeretnénk megjelentetni (általában valahol a mű végén), be kell tölteni az `ind` kiterjesztésű fájlt egy `\input{file.ind}` paranccsal. E parancs helyettesíthető a `\printindex` paranccsal abban az esetben, ha betöltjük a `makeidx` csomagot. Ez a csomag csak annyit tesz, hogy „kitalálja'' az `ind`-fájl nevét.

Az alábbi mintafájl egymás mellé helyezett két példányában a tárgymutató elkészítéséhez szükséges parancsok elhelyezését mutatjuk meg a bal oldalon a `makeidx` csomag használata nélkül, a jobb oldalon annak használatával.

```
01 \documentclass{article}
02 ...
03 \makeindex
04 ...
```

```
\documentclass{article}
\usepackage{makeidx}
\makeindex
...
```

```
\begin{document}
...
This \index{word} is...
...
\input{file.ind}
\end{document}
```

```
\begin{document}
...
This \index{word} is...
...
\printindex
\end{document}
```

A fenti példából annyi látszik, hogy a „word'' szó benne lesz a tárgymutatóban. Magyar nyelvű szöveg indexének kezeléséről a 8.4.3. és a 8.4.5. fejezetben írunk.

Az indexben a kapcsos zárójelek között bármilyen karakter szerepelhet, négy karakternek azonban speciális jelentése van. Ez a négy karakter a `@`, `!`, `|` és a " jel. Ha ezeket akarjuk az indexbe írni, akkor helyettük a `"@`, `"!`, `"|`, `""` karakterláncokat kell írni, azaz eléjük kell írni egy dupla idézőjelet. A `@`, `!`, `|` karakterek speciális jelentéséről és az indexekre vonatkozó további ismeretekről szól a 8.4. fejezet.

# 2.6. Illusztrációk

Az illusztráció fogalmát tág értelemben fogjuk használni, ide értjük a dokumentum – gyakran nem szöveges – szemléltető részeit: rajz, fénykép, térkép, grafikon, táblázat, programkód stb. Először a tabulált vagy táblázatba szedett információk megjelenítésével foglalkozunk, majd a LaTeX grafikai képességeit tekintjük át, végül megvizsgáljuk az illusztrációk elhelyezésének, az ún. úsztatásnak a módját.

## 2.6.1. Tabulált szöveg

**A `tabbing` környezet** ♦ Ha csak tabulálni akarunk egy szöveget, és különösen, ha a tabulátorhelyeket szöveg közben akarjuk kijelölni, a `tabbing` környezetet használjuk. E környezetben a \= paranccsal jelölhetjük ki a tabuláció helyeit, majd ezekre a \> paranccsal ugorhatunk. A Tab billentyű használatával ellentétben itt mindig a logikailag következő tabulátorhelyre ugrunk, akkor is, ha a szöveg ezen már túlszaladt. A sorokat itt is a \\ paranccsal zárjuk (kivéve az utolsót). Lássuk Majakovszkij „Beszélgetés az adófelügyelővel a költészetről'' című versének utolsó sorait:

```
\begin{tabbing}
akkor,\= \\
     \>elvtársak, itt a tollam,\\
írjanak\= \\
      \>maguk\= \\
      \>     \>helyettem!
\end{tabbing}
```

akkor,
elvtársak, itt a tollam,
írjanak
maguk
helyettem!

Mint látjuk, a negyedik programsorban kiadott \= parancs felülírja a másodikban kiadott parancs hatását, és átteszi az első tabulátorhelyet máshová.

Egy mintasorban előre kijelölhetők a tabulátorhelyek úgy is, hogy annak szövege ne jelenjen meg. Ekkor a sort a `\kill` paranccsal kell lezárnunk. E parancs használatával megjegyzéssorokat is tehetünk a tabulált szövegbe.

```
\begin{tabbing}
IF \=THEN \= \kill
IF \>korán kelsz\\
   \>THEN \>aranyat lelsz\\
   \>ELSE \>nagyot alszol
\end{tabbing}
```

IF korán kelsz
THEN aranyat lelsz
ELSE nagyot alszol

További parancsok is kiadhatók a `tabbing` környezetben. Ezeket a 6.1.1. alszakaszban részletezzük. A `tabbing` környezet a tabulátorhelyek közötti minél egyszerűbb lavírozás érdekében a `\=` mellett a `\'` és a `\‘` parancsokat használja illesztésre, ezért ékezetesítésre ezek nem használhatók: e környezetben a `\a=`, `\a'`, illetve a `\a‘` parancsokat használjuk helyettük. Nem kell törődnünk e megszorításokkal, ha ékezetes betűket használunk, és nem a TEX repülő ékezeteit.

A `tabbing` környezet jól használható számítógépes programok szedésére, bár erre kiváló csomagok is rendelkezésre állnak (ezekről lásd a 137. oldalt).

› **Gyakorlat:** *Tabulált szöveg egyenletes tabulátortávolsággal*
Gépeljük be az alábbi programkódot rögzített szélességű – mondjuk három „x'' betű szélességnyi – tabulátorokat használva!

```
loop while (DistanceToWoman > 0) {
    if  ((DistanceToWoman mod 2) = 0) {
        MoveLeftFoot(forward);
    } else {
        MoveRightFoot(forward);
    }
}
```

(A kód a `http://www.comedycode.com/showcode.cfm/code/16.html` címen található „How to meet the woman of your dreams'' című program néhány sora.) ‹

## 2.6.2. Táblázatok szerkesztése

Amikor olyan információkat szeretnénk megjeleníteni, melyek áttekinthető módon egy kétdimenziós téglalaprács csúcsain helyezhetők el, táblázatról beszélünk. A táblázat első sorában, illetve első oszlopában gyakran az oszlopokra, illetve sorokra vonatkozó valamilyen tömör meghatározás – az ún. *fejléc* – szerepel.

**A `tabular` környezet** ♦ Táblázat készítése előtt elég csak annyit tudnunk, hogy hány oszlopa lesz, és elég annyit eldöntenünk, hogy melyik oszlopot merre szeretnénk igazítani. A háromféle igazításhoz (balra, középre, jobbra) egy-egy betű tartozik: „l'' (bal, ***l**eft*), „c'' (közép, ***c**enter*) és „r'' (jobb, ***r**ight*). Amikor megszerkesztünk egy táblázatot, minden oszlophoz meghatározzuk az igazításának megfelelő betűt, és ezekből alkotunk egy karaktersorozatot (pl. e sorozat `llrc`, ha az első két oszlop tartalmát balra, a harmadikét jobbra, a negyedikét pedig középre szeretnénk igazítani). Ezután ezt a karaktersorozatot beírjuk a táblázatok szerkesztésére használható `tabular` környezet egyetlen argumentumába. Végül soronként felsoroljuk a táblázat

összes elemét balról jobbra haladva. Minden elemet az & karakterrel zárunk, kivéve a sorban utolsóként szereplőt. A sorokat a \\ paranccsal választjuk el egymástól. Ha a táblázatban egy mező üres, akkor a forrásállományban két & szerepel egymás után – köztük tetszőleges számú szóköz lehet. Példaként készítsünk egy kis menetrendet három buszjárat adataival. Minthogy a szöveget általában balra szoktuk igazítani, az egész számot jobbra, ezért a sorozat legyen `lrrr`.

```
\begin{tabular}{lrrr}
Budapest & 7:00 &  9:30 & 13:15 \\
Dömsöd   & 7:58 & 10:40 & 14:38
\end{tabular}
```

| | | | |
|---|---|---|---|
| Budapest | 7:00 | 9:30 | 13:15 |
| Dömsöd | 7:58 | 10:40 | 14:38 |

**Vízszintes és függőleges vonalak** ♦ A táblázatokat vízszintes és függőleges vonalakkal lehet áttekinthetőbbé tenni. A `\hline` parancs egy teljes táblázat szélességű vízszintes vonalat tesz oda, ahol kiadjuk, míg a `\hline\hline` egy dupla vízszintes vonalat. A `\hline` paranccsal megrajzolt vízszintes vonal nem számít külön sornak, azt mindig a sorzáró \\ és a következő sor első eleme közé kell írni. Ha az utolsó sor után is akarunk vonalat húzni, az utolsó sort is zárjuk le egy \\ paranccsal. A táblázat argumentumába írt karakterlánc elemei közé írt | karakter azt jelzi, hogy mely oszlopok mellé kerüljön függőleges vonal. A || karakterlánc beszúrása dupla függőleges vonalat eredményez.

```
\begin{tabular}{||l|rrr||}
\hline\hline
Budapest & 7:00 &  9:30 & 13:15 \\\hline
Dömsöd   & 7:58 & 10:40 & 14:38 \\
\hline\hline
\end{tabular}
```

| | | | |
|---|---|---|---|
| Budapest | 7:00 | 9:30 | 13:15 |
| Dömsöd | 7:58 | 10:40 | 14:38 |

**Ami az oszlopok között van** ♦ A `tabular` környezet argumentumában az „l'', „r'', „c'' betűk mellett további karakterek is használhatók formátumvezérlésre. Ezek egyike a `@` jel, mellyel a táblázat oszlopai közé tetszőleges elválasztó jelsorozat rakható. A `@` formátumvezérlő egyetlen argumentuma ezt az elválasztó jelsorozatot tartalmazza.

Lássunk néhány példát! A `@{,}` parancs a két oszlop közé minden sorban egy ',' karaktert tesz. Ezt felhasználhatjuk tizedes törtek tizedesvesszőre vagy tizedespontra való igazításához abban az esetben, ha a tizedesjegyek száma nem minden törtben ugyanannyi (erre a feladatra a 262. oldalon egy fejlettebb módszert is megismerünk majd). Az alábbi táblázat néhány nyomdai mértékegységet fejez ki mm-ben. Az első oszlop balra, az arányszámok egész része jobbra, majd a tört része balra van igazítva. Az első oszlop elé egy 1-es, az első és második oszlop közé egy szóközökkel körülvett egyenlőségjel, az arányszámok egészrésze és törtrésze közé egy tizedesvessző, végül az utolsó oszlop után egy `@{\,mm}` utasítással egy kis térköz és a mértékegység (mm) van beírva:

```
\begin{tabular}
    {@{1 }l@{ = }r@{,}l@{\,mm}}
  pont &  0 & 35 \\
  pica &  4 & 22 \\
  inch & 25 & 4
\end{tabular}
```

1 pont = 0,35 mm
1 pica = 4,22 mm
1 inch = 25,4 mm

A `@{}` utasítással 0 mm szélességűre lehet csökkenteni az oszlopok közötti távolságot. Figyeljük meg az alábbi kis táblázatok sorainak elejét és végét is:

```
\begin{tabular}{|cc|}
  \hline X & X \\\hline
\end{tabular}
\begin{tabular}{|@{}c@{}c@{}|}
  \hline X & X \\\hline
\end{tabular}
```

| X X | XX |
|---|---|

**Összevont oszlopok** ♦ A `\multicolumn` paranccsal egy sorban több oszlop összefogható egyetlen oszloppá. Ezt például akkor használjuk, ha a táblázat fejlécében több oszlophoz közös felirat tartozik. A parancs alakja: `\multicolumn{`*n*`}{`*igazítás*`}{`*szöveg*`}`, ahol *n* az összevonandó oszlopok száma, *igazítás* az `l`, `r` vagy `c` betűk valamelyike, ezen kívül szerepelhet még a `|` karakter. Azt, hogy hány oszlop van a táblázatban, az mutatja, hogy hány `l`, `r`, `c` vagy `p` formátumvezérlő parancs szerepel a `tabular` környezet argumentumában (a `p` parancsról az 56. oldalon írunk). Ahhoz, hogy a `\multicolumn` parancsot használhassuk, abban is meg kell egyezni, hogy az igazító parancsok közti `|`, `@` elválasztások hova számítsanak. A szabály az, hogy az első oszlop kivételével minden oszlop az igazító paranccsal, tehát az `l`, `r`, `c` vagy `p` betűkkel kezdődik. Például az `|l|r|rr|` argumentum a következő módon oszlik négy oszlopba:

`|l|` `r|` `r` `r|`
1. 2. 3. 4.

Ha tehát az 1. és a 2. oszlopot kapcsoljuk össze, akkor az `|l|r|` karakterlánc helyettesítéséről, ha a 2. és 3. oszlopot kapcsoljuk össze, akkor a `r|r` helyettesítéséről, ha a 3. és 4. oszlopot kapcsoljuk össze, akkor az `rr|` karakterlánc helyettesítéséről kell gondoskodni a `\multicolumn` parancs második argumentumában.

A sorokat elválasztó vízszintes vonal hossza is szabályozható. Erre a `\cline` parancs szolgál, melynek argumentumába két oszlop sorszámát írjuk, kötőjellel elválasztva. A két szám azt mondja meg, hogy hányadik oszloptól hányadikig tart a vonal. A `\cline{3-4}` parancs csak a 3. és 4. oszlopban húz egy vízszintes vonalat, a `\cline{1-1}` parancs csak az 1. oszlopban. A `\cline` parancs is a fent megfogalmazott szabály szerint határozza meg az oszlopok határait. Lássunk egy példát, melyben a jövedelemadót adjuk meg a jövedelem ismeretében:

```
\begin{tabular}{|l| c c|}
\cline{2-3}
\multicolumn{1}{c|}{}
&\multicolumn{2}{c|}{Év}\\\cline{2-3}
\multicolumn{1}{c|}{}&2002&2003\\\hline
Jövedelem & 775000 & 866500\\
Adó       & 165000 & 194950\\\hline
\end{tabular}
```

| | Év | |
|---|---|---|
| | 2002 | 2003 |
| Jövedelem | 775000 | 866500 |
| Adó | 165000 | 194950 |

Az első `\multicolumn` parancsban az eredeti formátumvezérlő `|l|` részét helyettesítjük egy `c|` vezérlővel, míg a másodikban a `cc|` részét.

› **Tipp:** *Néhány hibalehetőség a* `tabular` *környezetben*
Ne felejtsük el a `\multicolumn` parancs előtt, illetve után az `&` jeleket megfelelően kitenni. Például a fenti példa első sorában a két `\multicolumn` parancs közé kell egyet írni. Hibaüzenetet kapunk, ha a `\cline` parancsban hiányzik a „-" jel: például a `\cline{1}` parancs hibás, csak a `\cline{1-1}` alak a helyes. ‹

› **Gyakorlat:** *Oszlopösszevonás – a formátum megváltoztatása*
Készítsük el az alábbi ún. Young tablót (Young tableaux):

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 5 | 6 | 13 | | |
| 7 | 15 | | | |

‹

**Kiadói minőségű táblázatok** ♦ Teljesen igaza van az olvasónak, ha az eddig tanultak alapján néhány táblázatot elkészítve így vélekedik:

- ezek a táblázatok többnyire nem szépek,
- a dupla vonalak különösen nem,
- a vízszintes vonalak ráülnek a szövegre,
- ráadásul arról volt szó, hogy vizuális szerkesztéssel a szerzőnek nem kell bajlódnia, itt pedig ezt kell tennie (merre legyen igazítva az oszlop, hová húzzunk vonalat és az szimpla legyen vagy dupla…).

Valóban, táblázat szerkesztésekor néha elkerülhetetlen, hogy a táblázat megjelenésével kapcsolatos döntést kelljen hoznunk. Másrészt az is igaz, hogy a LaTeX szerzője nem dolgozott ki tipográfiai automatizmusokat a táblázatok megjelenítésére. Ilyesmire eddig csak Simon Fear vállalkozott – használjuk az általa készített `booktabs` csomagot. Az alábbi szabályok egy része az ő ajánlásai alapján készült (`CTAN/macros/latex/contrib/booktabs/`):

1. A táblázat tartalmáról:

   *a)* Igyekezzünk, hogy a táblázat minél egyszerűbb és áttekinthetőbb legyen!
   *b)* Ha egy oszlopban egy elem megegyezik a fölötte lévővel, akkor egyszerűen írjuk le újra (ne használjunk semmilyen egyszerűsítő módszert: macskaköröm…)!
   *c)* Ha egy oszlopban (vagy sorban) minden elemnek ugyanaz a mértékegysége, akkor azt a fejlécbe írjuk, ne az elemek után!

2. Az oszlopok igazításáról:

   *a)* Szöveget balra, egész számot jobbra, formulát középre igazítsunk!
   *b)* Több oszlop közös fejléce középre kerüljön!

3. A vonalakról:

   *a)* Soha ne használjunk függőleges vonalat!
   *b)* Soha ne használjunk dupla vonalat!
   *c)* A táblázat tetejére a `booktabs` csomag `\toprule`, aljára a `\bottomrule` parancsával húzzunk vízszintes vonalat!
   *d)* A táblázat fejléce alá, illetve ahol a táblázat tartalma szerint kettéoszlik, a `\midrule` paranccsal húzzunk vízszintes vonalat!
   *e)* Rövidebb vízszintes vonalat a `\cmidrule` paranccsal húzhatunk. Ennek argumentuma azonos a `\cline` parancséval, de azon kívül megadható előtte egy kerek zárójelekbe zárt opcionális argumentummal az is, hogy mely végét kell e vonalnak levágni (`(r)`, ha a jobb, `(l)`, ha a bal, és `(lr)`, ha mindkét végét). További részletek e parancsokról a könyv 6.1.2. fejezetében.
   *f)* A formátumvezérlő két szélére tegyünk egy-egy `@{}` parancsot!

A 3. *a)* szabály talán meglepő, de fogadjuk el. Nem ritka az a kiadói gyakorlat, mely követi e szabályt! Lássuk példaként legutóbbi táblázatunkat!

```
%^\usepackage{booktabs}
\begin{tabular}{@{}lcc@{}} \toprule
  &\multicolumn{2}{c}{Év}\\\cmidrule{2-3}
  & 2002 & 2003\\\midrule
Jövedelem (Ft)& 775000 & 866500\\
Adó (Ft)      & 165000 & 194950\\
\bottomrule
\end{tabular}
```

| | Év | |
|---|---|---|
| | 2002 | 2003 |
| Jövedelem (Ft) | 775000 | 866500 |
| Adó (Ft) | 165000 | 194950 |

› **Gyakorlat:** *Statisztikai táblázat készítése*

| | 2001 | | 2002 | |
|---|---|---|---|---|
| Megnevezés | jan. 1. | dec. 31. | jan. 1. | dec. 31. |
| Jogi személyiségű társas vállalkozás | 145 868 | 163 824 | 151 152 | 171 584 |
| Jogi személyiség nélküli társas vállalkozás | 185 735 | 207 954 | 193 748 | 213 760 |
| Egyéni vállalkozás | 430 031 | 468 797 | 442 900 | 474 678 |
| Vállalkozás összesen | 761 634 | 840 575 | 787 800 | 860 022 |
| Költségvetési és társadalombiztosítási szervezet | 15 436 | 15 615 | 15 621 | 15 401 |
| Nonprofit szervezet | 65 335 | 67 153 | 67 147 | 69 074 |
| MRP-szervezet | 263 | 228 | 228 | 194 |
| Összesen | 842 668 | 923 571 | 870 796 | 944 691 |

Készítsünk táblázatot a KSH-nak a Magyarországon működő szervezetek számáról szóló adatai (`http://www.ksh.hu/`) alapján, a fenti minta szerint! ‹

**Adott szélességű oszlopok** ♦ Rögzíthetjük is egy oszlop szélességét. Ilyenkor a szöveg automatikusan sorokra törik, ha nem fér el egy sorban, és tömbösítve jelenik meg. Ezt a `p` formátumvezérlő utasítással érhetjük el, melynek argumentuma az oszlop szélessége. A `p` az angol *paragraph* (bekezdés) szóra utal, és azt jelenti, hogy a mező tartalma nem feltétlenül egyetlen sorba fog kerülni, hanem szükség esetén bekezdésszerűen több sorba lesz törve.

```
\begin{tabular}{|p{4cm}|} \hline
Ez a szöveg bekeretezve jelenik meg.
A~keret szélessége 4\,cm, a magassága
a szöveg hosszától függ. \\\hline
\end{tabular}
```

| Ez a szöveg bekeretezve jelenik meg. A keret szélessége 4 cm, a magassága a szöveg hosszától függ. |
|---|

A `p` formátumvezérlővel könnyen készíthetünk olyan táblázatot, melyben minden oszlop, s így maga a táblázat is, adott szélességű. Például a `{|p{2cm}|p{2cm}|}` argumentum hatására egy kétoszlopos, 4 cm széles táblázatot kapunk. A fenti példa mutatja, hogy a több sorra tört táblaelem első sora igazodik a többi oszlopelemhez. Ha azt szeretnénk, hogy a többsoros elem közepe, illetve utolsó sora igazodjon a vele egy sorban lévő elemekhez, akkor használjuk az `array` csomag `m` (közepe), illetve `b` (utolsó sora) vezérlőjét a `p` helyett.

**Adott szélességű táblázat** ♦ A `tabular*` környezettel adott szélességű táblázatot hozhatunk létre. E környezetnek nem egy, hanem két argumentuma van. Az első argumentum a táblázat szélességét adja meg, a második a `tabular` környezetnél már ismert oszlopleíró argumentum. A következő példában egy 40 mm széles táblázatot mutatunk. Ahhoz, hogy oszlopai rugalmasan széthúzódhassanak, az oszlopok közti távolságot egy `\fill` paranccsal rugalmassá kell tenni (erről később lesz szó), ehhez az `\extracolsep{\fill}` parancsot kell használni, amelynek hatása nemcsak a kiadás helyére, hanem az összes további oszlopra is vonatkozik. Hatását megszüntethetjük egy későbbi oszlop után az `\extracolsep{2\tabcolsep}` paranccsal történhet, ami visszaállítja az eredeti oszloptávolságot.

```
\begin{tabular*}{40mm}
{@{}l@{ -- }l@{\extracolsep{\fill}}r@{}}
\toprule FTC  & MTK & 1:1 \\
         Vasas & ETO & 0:0 \\ \bottomrule
\end{tabular*}
```

| FTC – MTK | 1:1 |
|---|---|
| Vasas – ETO | 0:0 |

› **Gyakorlat:** *Adott szélességű táblázat szerkesztése*
Készítsünk egy táblázatot a totóeredményekről! Az első oszlopban a két csapat neve, a másodikban az eredmények, a harmadikban az 1, 2, x jelek valamelyike legyen! A táblázat szélessége legyen a sorhossz 0,8-szerese: azaz a `tabular*` első argumentumába írjuk be, hogy `0.8\textwidth`! Az `\extracolsep` paranccsal érjük el, hogy csak az első és második oszlop közötti távolság nyúljon meg! ‹

### 2.6.3. Rajzok

Egy LaTeX dokumentumba kétféleképpen kerülhet ábra. Az egyik lehetőség, hogy a LaTeX saját grafikai környezetének, a `picture` környezetnek a parancsait használjuk. E környezet a számítástechnika (h)őskorára emlékeztető megoldáson alapul: a rajzot különböző meredekségű egyenesdarabokat, illetve különböző görbületű ívdarabokat tartalmazó karakterekből állítja össze. Egy véges (márpedig nagyon véges) karakterkészletből rajzoknak csak igen korlátozott típusai rakhatók össze. Egyszerűbb ábrák megrajzolására azonban ez is elég.

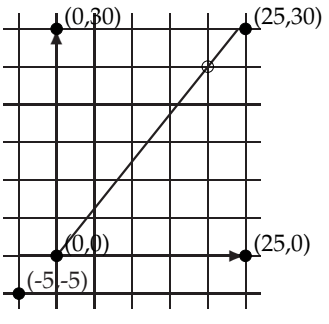
Más programokkal készített rajzokat, képeket is beilleszthetünk a dokumentumba. Ilyenkor a LaTeX parancsaival egy olyan programot, ún. eszközmeghajtót (angolul *device driver*) vezérlünk, mely képes valamelyik eszközön (pl. a képernyőn, valamilyen adott típusú nyomtatón stb.) a grafikát megjeleníteni. Így ez a megoldás rendszer- és eszközfüggő. A leggyakoribb az EPS (*encapsulated PostScript*) és a PDF formátumban mentett ábrák használata.

A `picture` környezet alkalmazásának van néhány olyan előnye, ami miatt érdemes egy kis időt szánni rá: az így készült rajzok például teljesen hordozhatók, minden platformon megjeleníthetők, nincs gond a különféle grafikai fájlformátumok kezelésével, utólag is könnyen változtathatunk az ábrán, az ábrák szövegei automatikusan azzal a betűvel készülnek, amelyikkel a szöveg, az így készült ábra lekicsinyítése, felnagyítása közben a vonalak vastagsága nem változik. Ráadásul vannak olyan egyszerűbb, ingyenes rajzoló programok, melyek a LaTeX `picture` formátumába konvertálják a velük készült rajzot. Ilyen Windows alatt a TeXCAD, Linux alatt az xfig program. Egyszerűbb ábrát érdemes a `picture` parancsaival készíteni!

Léteznek öszvér megoldások, melyek a LaTeX picture környezete által nyújtott szolgáltatásait kiterjesztik más grafikai rendszerek (pl. PostScript) lehetőségeivel (l. epic, eepic csomag). Az xfig program támogatja ezt a lehetőséget is (mi nem szeretjük).

**A LaTeX grafikai módja** ♦ A LaTeX saját rajzainak megszerkesztésére a `picture` környezet szolgál. Itt a legtöbb LaTeX-parancs nem használható, van viszont olyan, ami csak itt adható ki (e környezetben a LaTeX grafikai módban van).

A LaTeX egy `picture` környezettel előállított rajzot egyetlen egységként kezel, mint egyetlen betűt. Egy ilyen rajz betehető a szövegbe, vagy beépíthető egy másik rajzba is.




A `picture` környezet parancsai egy képzeletbeli koordináta-rendszer koordinátáit használják, ezért a rajz megtervezéséhez tegyünk magunk elé egy kockás papírt. A LaTeX grafikai parancsaiban $(a, b)$ alakban adjuk meg a pontok koordinátáit, ahol $a$ és $b$ valós számok. A picture környezeten belül a tizedes törtekben csak tizedespont használható, tizedesvessző nem. A mellékelt rajz egy kockás papír egy darabját mutatja, amire elkészítettük a vázlatot. A rajzunkat $6 \times 7$-es méretűre terveztük. Egy szokásos kockás papíron egy négyzet oldalhossza jó közelítéssel 5 mm, így rajzunk 30 mm $\times$ 35 mm méretű. Segít-

ségül berajzoltunk néhány pontot, és mellé írtuk a pont koordinátáit milliméterben számolva. Ezeket a koordinátákat fogjuk használni a rajzoláshoz. Egy üres körrel jelöltük a függvénykapcsolatot ábrázoló egyenesnek azt a pontját, melyből leolvasható az irányvektora: a megrajzolandó koordináta-rendszer origójától 4 egységnyi jobbra, 5 egységnyi föl, tehát az irányvektor $(4,5)$.

**Az egység megválasztása** ♦ Először megadjuk, hogy a rajzoláshoz használt koordináta-rendszeren egy egység milyen hosszú legyen. Erre való a `\unitlength` parancs. Első rajzunkhoz az 1 mm-t választottuk (az csak „véletlen'', hogy amit rajzolunk, *abban* is van egy koordináta-rendszer, ahol más az egység). Az értékadásnak két módja van, írhatjuk, hogy `\unitlength 1mm` vagy hogy `\setlength{\unitlength}{1mm}`.

A továbbiakban a `picture` környezet minden pontjának koordinátáját és minden hosszúságadatát ebben az egységben fejezzük ki. A `picture` környezeten belül nem tanácsos változtatgatni az egységet.

**A rajz mérete és az origó** ♦ Miután megválasztottuk az egységet, kijelöljük a `picture` környezetet, azaz választunk egy akkora téglalapot, amelyre ráfér a rajzunk. 1 mm-es egység esetén a példának szánt rajzunk $30\times35$-ös, így a környezet nyitó parancsa `\begin{picture}(30,35)`. Ezután opcionálisan megadható a kép referenciapontjaként használt bal alsó sarkának koordinátája. Ha nem adunk meg semmit, akkor alapértelmezésként e pont koordinátája $(0,0)$ lesz. Példánkban ekkor a jobb felső sarok koordinátája $(30,35)$ lenne, míg a megrajzolandó koordináta-rendszerünk origója a picture környezet $(5,5)$ koordinátájú pontjába kerülne. Így nehezebb lenne számolnunk. Azt szeretnénk, ha a bal alsó sarok koordinátája $(-5,-5)$ lenne; ezt a `\begin{picture}(30,35)(-5,-5)` paranccsal érhetjük el. (Figyeljük meg a picture környezet egy sajátosságát: paramétere és opcionális paramétere is kerek zárójelet használ!)

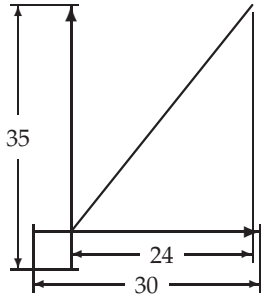
**Vonalvastagság** ♦ Miután megvan a papír, amire rajzolunk, tollat választunk. Kétféle toll van. Az egyikkel vastag (—), a másikkal vékony (—) vonal rajzolható. Mindkét toll választásának megfelel egy-egy parancs. A `\thicklines` parancs kiadása után minden vonal vastag lesz, a `\thinlines` kiadása után vékony (ez utóbbi az alapértelmezés). Ha egy ábrát lekicsinyítünk vagy felnagyítunk, e két vonalvastagság nem fog arányosan megváltozni, így elkerülhető az a gyakori és kellemetlen hiba, hogy a megváltoztatott méretű ábrán nem megfelelő a vonalvastagság.

A vonalvastagság méretének tetszőleges változtatását a 6.2. fejezetben tárgyaljuk (l. 277. oldal).

**Képelemek elhelyezése** ♦ A picture környezetbe ún. képelemeket helyezhetünk el, úgymint egyenest, vektort, kört, téglalapot, oválist, stb. Minden képelemnek van egy referenciapontja, például a köré a középpontja, a vektoré a kezdőpontja. A képelemeket a `\put` paranccsal helyezzük a képbe, mégpedig úgy, hogy először megadjuk, hogy a képelem referenciapontja hová kerüljön, majd kapcsos zárójelek között megadjuk a képelemet. Például a `\put(4,3){\circle{2}}` parancs egy 2-átmérőjű (nem 2-sugarú!) kört helyez el a képben úgy, hogy annak középpontja a $(4,3)$ koordinátájú pont legyen.

**Szakasz, vektor** ♦ Vektor kirajzolásához három adatra van szükségünk: a *kezdőpont*ra, ahová majd a `\put` parancs mutat, a vektor irányát megadó *irányvektor*ra és a *vetület hosszá*ra, függőleges vektor esetén a függőleges vetület hosszára, minden egyéb irányú vektor esetén a vízszintes vetület hosszára. Egyenes szakasz esetén ugyanezekre az adatokra van szükség, csak bármelyik végpontot kijelölhetjük kezdőpontnak.

A mellékelt ábra azt a három hosszméretet mutatja, mely a két vektor (mint koordináta-tengely) és az egyenes (mint függvénygrafikon) megrajzolásához kell. A vízszintes tengely mérete 30 mm, a függőlegesé 35 mm. Az egyenes szakasz legyen akkora, hogy vízszintes vetülete 24 mm legyen.




Egyenest a `\line`, vektort a `\vector` paranccsal rajzolhatunk.

Annak a $(0,0)$ pontból induló $(4,5)$ irányvektorú szakasznak, melynek *vízszintes vetülete* 24 egység, a függőleges vetülete 30 egység, hisz $24 \cdot \frac{5}{4} = 30$. Így másik végpontjának koordinátája a kockás papíron $(24,30)$. Ha ebből a pontból indítjuk a szakaszt, akkor viszont az irányvektora $(-4,-5)$ lesz, hiszen ha a szakasz másik végét választjuk referenciapontnak, akkor az irányvektor mindkét koordinátája előjelet vált.

Mindezek alapján e szakaszt vagy a

```
\put(0,0){\line(4,5){24}}
```

vagy pedig a

```
\put(24,30){\line(-4,-5){24}}
```

paranccsal jeleníthetjük meg. Hasonlóan adhatók meg a vektorok is:

```
\put(-5,0){\vector(1,0){30}}
```

a vízszintes, míg

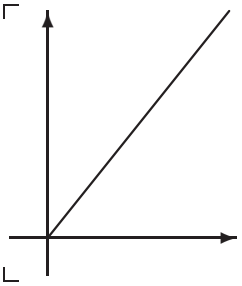
```
\put(0,-5){\vector(0,1){35}}
```

a függőleges tengelyt rajzolja meg.

A rajzoláshoz használt karakterkészlet korlátossága miatt alaphelyzetben az egyenes irányvektorának koordinátái $-6$ és 6 közé kell hogy essenek, és egész számoknak kell lenniük, míg a vektor irányvektorának koordinátáira $-4$ és 4 a határ. A két koordinátának nem lehet $\pm 1$-től különböző közös osztója, pl. nem lehet megadni a $(4,2)$ vektort! A vízszintestől és függőlegestől különböző egyenesek vízszintes vetülete nem lehet 10 pt-nál (kb. 3,5 mm) kisebb, tehát nem lehet akármilyen rövid egyenest rajzolni. E korlátok eltűnnek az `eepic` csomag használatával, vagyis a `\usepackege{eepic}` parancs kiadása mellett tetszőleges irányvektorú, tetszőleges méretű egyeneseket rajzolhatunk, de a vektorok irányvektorára vonatkozó korlát megmarad. Az `eepic` csomag más elven működik, megfelelő külső eszközmeghajtó programokat igényel, ezért nem minden megjelenítő programmal használható.

Ezek után lássuk első ábránkat! Használjunk vastag vonalat:

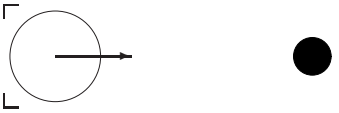
```
\unitlength 1mm
\begin{picture}(30,35)(-5,-5)
  \thicklines
  \put(-5,0){\vector(1,0){30}}
  \put(0,-5){\vector(0,1){35}}
  \put(0,0){\line(4,5){24}}
\end{picture}
```



**Kör ♦** A `\circle*{`*átm*`}` parancs kirajzol egy *átm* átmérőjű kört. A `*` opcionális, jelenlétében tömör, nélküle üres kört kapunk.

Alaphelyzetben kört is csak egy véges halmazból tudunk választani. Maga a LaTeX választja ki azt a kört, melynek átmérője a megadott *átm* értékéhez a legközelebb áll. Az `eepic` csomag itt is segíthet, használatával megszűnnek az átmérőre vonatkozó korlátozások.

```
\put(6,6){\vector(1,0){10}}
\put(6,6){\circle{12}}
\put(40,6){\circle*{5}}
```



**Dobozok – szöveg elhelyezése a rajzban ♦** A `\makebox(`*x,y*`)[`*poz*`]{`*szöveg*`}` parancs létrehoz egy *(x,y)* méretű dobozt, melybe beírja a *szöveg*et. E szöveg a doboz tetejére kerül, ha a *poz* helyén a `t` betű áll (*top*), az aljára, ha `b` (*bottom*), balra, ha `l` (*left*), jobbra, ha `r` (*right*) áll, és a dobozban a *szöveg*et a szélekig széthúzza, ha `s` (*stretch*) áll. Lehet egyszerre egy vízszintes és egy függőleges igazító parancsot is alkalmazni: `lt` (balra fönt), `lb` (balra lent), `rt` (jobbra fönt), `rb` (jobbra lent). A doboz referenciapontja a bal alsó sarka (ezt teszi a `\put` által megjelölt helyre). A `\framebox` parancs ugyanígy működik, de az a doboz keretét is megrajzolja. Ezt a parancsot használjuk akkor is, ha csak egy téglalapot akarunk kirajzolni, ekkor a parancs alakja `\framebox(`*x,y*`)[`*poz*`]{}`.

Lássunk néhány példát! A dobozok bal alsó sarkának helyét a `\put` paranccsal, a szöveg helyét a dobozban a `\framebox` opcionális paraméterével szabályozzuk.

```
\put(0,30){\framebox(19,9)[lt]{Bal felső}}
\put(20,30){\framebox(19,9)[t]{Közép felső}}
\put(40,30){\framebox(19,9)[rt]{Jobb felső}}
\put(0,15){\framebox(19,9)[l]{Bal közép}}
\put(40,15){\framebox(19,9)[r]{Jobb közép}}
\put(20,20){\framebox(19,9){Közép}}
\put(20,10){\framebox(19,9)[s]{Szét közép}}
\put(20,0){\framebox(19,9)[sb]{Szét alsó}}
\put(0,0){\framebox(19,9)[lb]{Bal alsó}}
\put(40,0){\framebox(19,9)[rb]{Jobb alsó}}
```

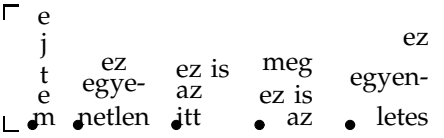

Ha egysoros szöveget akarunk írni a rajzba, akkor a `\makebox` parancsot használjuk. Leggyakrabban egy 0 méretű dobozt jelölünk ki, és „abba'' írjuk a szöveget, ami

persze kilóg belőle, de nem baj, úgysem látszik. Az alábbi ábrán tájékoztatásként egy kis pontot teszünk arra a helyre, ahová a doboz kerül. A bal oldali forrásszövegből természetesen hiányoznak a pontokat kirakó parancsok, sőt, a szürkén szedett szövegeké is!

```
\put(30,0){\makebox(0,0)[b]{{Alsó}}}
\put(0,10){\makebox(0,0)[l]{Bal közép}}
\put(30,10){\makebox(0,0){Közép}}
\put(60,20){\makebox(0,0)[rt]{Jobb felső}}
```

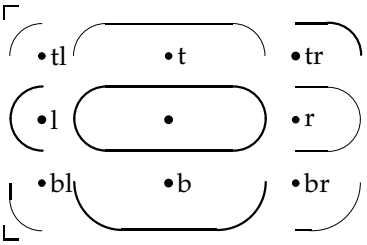

Ha többsoros szöveget akarunk a rajzba tenni, használjuk a `\shortstack` parancsot. A `\shortstack` működik a picture környezeten kívül is. Alakja `\shortstack[`*poz*`]{`*szöveg*`}`, ahol a *poz* helyén `l` (bal) vagy `r` (jobb) állhat, a szöveg viszont tartalmazhat \\ parancsokat is. A `\shortstack` a \\ parancsokkal elválasztott sorokat külön dobozokba teszi, és ezeket rakja egymás alá, a sorok alapvonalát nem figyeli, a sorok között nem tart egyenletes sorközt. Ennek az az oka, hogy a LaTeX e dobozokat is grafikai elemként kezeli. Egyenletes folthatású, elfogadható képet kapunk, ha minden sorban csak egy betű van (lásd az alábbi első példát). Ha azonban egy-egy sorba több betű is kerül, tegyünk minden sorba gyámfát, azaz írjunk egy `\strut` parancsot (lásd 149. oldal)!

```
\put(3,0){\shortstack{e\\j\\t\\e\\m}}
\put(9,0){\shortstack{ez\\egye-\\netlen}}
\put(22,0){\shortstack[l]{ez is\\az\\itt}}
\put(33,0){\shortstack[r]{meg\\ez is\\az}}
\put(45,0){\shortstack[r]{
  \strut ez\\\strut egyen-\\\strut letes}}
```




**Ovális** ♦ Az ovális a picture környezetben egyszerűen csak lekerekített sarkú téglalapot jelent. Az ovális referenciapontja a középpontja. Megadása az `\oval` paranccsal történik, melynek alakja `\oval(`*x*, *y*`)[`*részlet*`]`, ahol *x* a téglalap vízszintes, *y* a függőleges mérete, *részlet* pedig a már ismert betűopciókat használva megadja, hogy az ovális melyik részletét rajzoljuk ki (`t` felső, `b` alsó, `l` bal, `r` jobb, `tl` bal felső stb.). Az alábbi ábra az összes lehetőséget megmutatja, de a forráskód csak négyet. A jobb áttekinthetőség kedvéért az oválisok középpontjába (ahová a `\put` parancsok mutatnak) kis pontot tettünk, és mellé írtuk a megfelelő opcionális paramétert. Innen azonosítható, hogy melyik oválishoz melyik parancs tartozik.

```
\thicklines
  \put(45,25){\oval(20,10)[tr]}
  \put(5,15){\oval(10,10)[l]}
  \put(25,15){\oval(30,10)}
  \put(25,5){\oval(30,15)[b]}
```



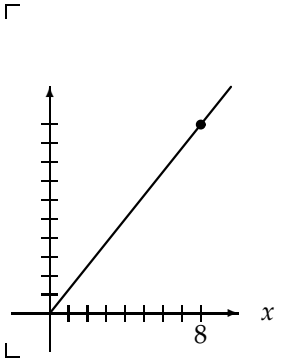
› **Gyakorlat:** *Képelemek megjelenítése*
Rajzoljuk meg az alábbi ábrát, melyen egy 40 mm hosszú vektor, egy 10 mm hosszú egyenes, egy 1 mm átmérőjű körlap, egy 10 mm átmérőjű ovális fele és egy felirat látható!


‹

**Ismétlődő képelemek** ♦ Hasznos parancs a `\multiput`, mellyel egy képelemnek nemcsak egy, hanem több példánya is elhelyezhető. Alakja `\multiput`$(x,y)(\Delta x,\Delta y)\{db\}$ `{`*képelem*`}`, ahol az $(x,y)$ koordinátájú helyre tesszük a *képelem* első példányát, majd az összesen *db* példányt úgy helyezzük a képbe, hogy mindegyik az előzőből a $(\Delta x,\Delta y)$ vektorral való eltolás útján kapható meg.

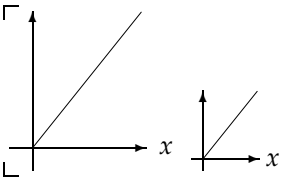
E parancs segítségével beosztásokat rajzolunk a koordináta-tengelyekre, majd az imént tanult módon feliratokat is elhelyezünk melléjük:

```
\unitlength 1mm
\begin{picture}(45,45)(-5,-5)
\thicklines
  \put(0,0){\line(4,5){24}}
  \put(20,25){\circle*{1}}
\thinlines
  \put(-5,0){\vector(1,0){30}}
  \put(0,-5){\vector(0,1){35}}
  \multiput(2.5,-1)(2.5,0){8}{\line(0,1){2}}
  \multiput(-1,2.5)(0,2.5){10}{\line(1,0){2}}
  \put(28,0){\makebox(0,0)[l]{$x$}}
  \put(20,-4){\makebox(0,0)[b]{8}}
\end{picture}
```




**Részábrák** ♦ Bonyolult ábrák részábrákra bonthatók, majd azok egy egyszerű `\put` paranccsal berakhatóak egy közös ábrába. Annak szemléltetésére, hogy hogyan (nem) változnak a feliratok és a vonalvastagság egy ábra kicsinyítése során, egy grafikont teszünk egymás mellé két példányban, de a második egysége fele a másikénak.

```
\unitlength .6mm
\begin{picture}(60,35)
  \put(0,0){%
    \begin{picture}(30,35)(-5,-5)
      \put(-5,0){\vector(1,0){30}}
      \put(0,-5){\vector(0,1){35}}
      \put(0,0){\line(4,5){24}}
      \put(28,0){\makebox(0,0)[l]{$x$}}
```



```
    \end{picture}}
  \put(40,0){\unitlength.3mm
    \begin{picture}(30,35)(-5,-5)
      \put(-5,0){\vector(1,0){30}}
      \put(0,-5){\vector(0,1){35}}
      \put(0,0){\line(4,5){24}}
      \put(28,0){\makebox(0,0)[l]{$x$}}
    \end{picture}}
\end{picture}
```

Ábrarészlet készítésére és többszöri felhasználására használhatjuk a `\savebox` parancsot, melyet később ismertetünk (l. 277. oldal).

## 2.6.4. Képek beillesztése

Gyakran előfordul, hogy valamilyen képet vagy grafikai programmal készült ábrát szeretnénk a dokumentumunkba beépíteni. Olyan grafikai formátumú fájlokat lehet használni, amelyekhez az adott LaTeX-implementációban van megfelelő eszközmeghajtó.

Az alábbi példában betöltünk egy `mikulas.eps` néven elmentett „encapsulated PostScript'', azaz `eps` formátumú grafikát (l. 623. és fej:szotar. o.). A betöltéshez a `graphics` vagy a `graphicx` csomag `\includegraphics` parancsát `\includegraphics{mikulas}` formában kell használni. A grafikáról további információt találunk a 6. fejezetben.

```
%^\usepackage{graphics}

\includegraphics{mikulas}
```

## 2.6.5. Illusztrációk úsztatása

Egy dokumentumnak lehet olyan része, objektuma, melyet nem lehet eltörni, vagy nem szeretnénk, hogy a TeX eltörje, amikor új oldalt kezd, ugyanakkor elég nagy ahhoz, hogy az esedékes töréspont jó eséllyel essen a területére. Ilyen pl. egy ábra, rajz vagy táblázat.

Ezek az objektumok legegyszerűbben úgy jeleníthetők meg üres fél oldalak kihagyása nélkül, ha megengedjük, hogy a (La)TeX kicsit arrébb tegye, a szöveg „folyamán'' belül arrébb „úsztassa'' az eltörhetetlen objektumot. Elterjedt szokás például egy táblázatot annak a lapnak a tetejére tenni, amelyen először kerül szóba, míg egy egész oldalas ábra leggyakrabban a szóba kerülése melletti oldalra kerül. Az ilyen módon kezelt objektumokat *úsztatott* vagy *úszó objektumoknak* nevezzük. Angolban a *float* (magyarul uszály) vagy a *floating body* (úszó test) kifejezéseket használják.

Eddig azt tanulmányoztuk, ami az „uszályra'' kerül: tabulált szöveg, táblázat, rajz, kép. Most magáról az uszályról lesz szó, melyet ábrának *(figure)* nevezünk, ha rajzot *(draw)* vagy képet *(picture)* úsztat, és táblázatnak *(table)*, ha táblázatot *(tabular matter)*. Mint látjuk, az angol különbséget tesz táblázat, mint uszály, és táblázat, mint úszó objektum között, mint ahogy a magyar szóhasználat is különbséget tesz ábra, mint uszály, és kép, mint úszó objektum között. Ezt az árnyalatnyinak tűnő különbséget megerősíti az, hogy az „uszályon'' az „úszó objektum'' mellett egy felirat is szerepel, egy táblázatcím, vagy egy ábraaláírás.

Illusztrációk *úsztatására* külön parancsok szolgálnak: táblázatokhoz a `table` és a `table*`, ábrákhoz a `figure` és a `figure*`. A csillagos alakokat a kéthasábos szedésnél használjuk, amikor az objektumnak nem az egyik hasábon belül, hanem mindkét hasáb szélességét felhasználva kell megjelennie. E környezeteknek egyetlen opcionális argumentumuk van, ahol az objektum elhelyezésére vonatkozó instrukciók adhatók meg. Ezek az egybetűs rövidítéssel megadható instrukciók a következők:

`h` *(here)* – tedd ide,
`t` *(top)* – tedd a lap tetejére,
`b` *(bottom)* – tedd a lap aljára,
`p` *(page)* – tedd egy külön oldalra, ahol csak úsztatott objektumok vannak.

Az előző betűk kombinálhatók is, például a `[tb]` argumentum megadása után a LaTeX először megpróbálja a lap tetejére tenni az úszó objektumot, ha oda valamilyen ok miatt nem sikerül, a lap aljával próbálkozik. A „próbálkozik'' szó azt takarja, hogy az úsztatott objektumok elhelyezését bonyolult tipográfiai szabályok korlátozzák, és megeshet, hogy e szabályok nem engedik meg az objektum elhelyezését a kívánt módon. Ilyenkor a LaTeX későbbre, gyakran a fejezet végére hagyja az ábra kinyomtatását. E szabályok megfelelő paraméterek átállításával megváltoztathatók. Az is lehetséges, hogy a LaTeX-et felmentsük a szabályok mindegyikének szigorú betartása alól, ehhez egy `!`-jelet kell tenni az imént felsorolt egybetűs parancsok elé. Például az alábbi kódban szereplő `[!t]` argumentum hatására a LaTeX a beágyazott `tabular` környezetben lévő táblázatot bizonyos szabályok felrúgása árán is a lap tetejére próbálja tenni.

```
\begin{table}[!t]
\begin{tabular}
....
\end{tabular}
\end{table}
```

Ha nem adunk meg opcionális argumentumot, akkor a `[tbp]` opció lép életbe, ami az alapértelmezés.

A táblázatokhoz, ábrákhoz gyakran társul sorszámozott cím, rövid magyarázat. Erre a `\caption` parancs[8] használható. A `\caption` az argumentumában megadott szöveg elé automatikusan generál sorszámot, és attól függően, hogy a `table` vagy a `figure` környezetben adtuk-e ki, a „Table'' vagy „Figure'' szavakat írja ki címül. Ha a `babel` csomagot betöltjük a

[8] *caption* – képszöveg, (film)felirat.

```
\usepackage[magyar]{babel}
```

paranccsal, akkor a „táblázat'' vagy az „ábra'' szavak jelennek meg a címben. Az alábbiakban egy ilyen példát mutatunk:

```
01 \begin{table}[h]
02 \caption{Gazdasági számítások}
03 \begin{tabular}{lr}
04 Bevétel & 10\$ \\
05 Kiadás  & 8\$  \\
06 Haszon  & 2\$
07 \end{tabular}
08 \end{table}
```

6.8. táblázat. Gazdasági számítások

| | |
|---|---|
| Bevétel | 10$ |
| Kiadás | 8$ |
| Haszon | 2$ |

A képeknél a szöveget általában a kép alá teszik, táblázatoknál inkább a táblázat fölé, mint mi is tettük a fenti példában. Azt, hogy a táblázat- vagy képszöveg hová kerüljön, egyszerűen a `\caption` parancs megfelelő helyre írásával szabályozhatjuk.

## 2.7. A dokumentum megírása

### 2.7.1. Téma, megfogalmazás, begépelés

Az eddiekben megismertük a főszöveg legfontosabb elemeit, és néhányat a főszöveg járulékos részei közül. Most tekintsünk művünk egészére.

A LaTeX azt az elvet tartja szem előtt, hogy a szerzőnek saját műve megírásával kell foglalkoznia, művének belső struktúráját kell felépítenie, annak eldöntése pedig, hogy mindez milyen formában jelenjen meg, nem a szerző dolga, hanem a tipográfusé és a szedőé. A tipográfusi munkát beépített tudásával a LaTeX-nek kell elvégeznie, a szedői munkát a LaTeX a TeX-re bízza. Persze a LaTeX nem tudja a választ minden tipográfiai kérdésre. Időnként szükség van a beavatkozásunkra, néha tipográfiai kérdésekben is döntenünk kell. Ezt egy egyszerű példán szemléltetjük. A szerző tudja, hogy mi művének a címe, s csak az a dolga, hogy ezt pontosan megmondja a LaTeX-nek. Arról viszont, hogy a címnek mekkora, milyen típusú betűkből kell állnia, hogy mennyi legyen előtte és utána a térköz, már a LaTeX dönt mindazoknak az információknak az alapján, amelyeket mi a művünkről közöltünk vele. Ha a cím hosszú, és nem fér el egy sorban, döntsön ő arról, hogy hol törje el. Lehet azonban, hogy olyan helyen töri el, ami a cím tartalmával nincs összhangban. Ekkor viszont nekünk kell döntenünk, és felülbírálva a LaTeX-et, nekünk magunknak kell megmutatnunk, hogy hol legyen a töréspont.

A 2.1. ábrán a dokumentumszerkesztés három különböző módját szemléltetjük nagyon leegyszerűsített módon. A bal hasábban a kész dokumentum és írógéppel való begépelésének főbb lépései láthatók. A jobb hasábban ugyanennek – a webes megjelenítéshez ma használt – HTML-kódú változata, míg középen a LaTeX-kód. Érdemes kicsit összevetni e három megoldást. Mindhárom megoldás a dokumentum egészére vonatkozó bizonyos döntések meghozásával kezdődik: nyelv, betűkészlet, betűméret (ha nincs megadva, akkor az alapértelmezés). Innen azonban egyre több a

különbség. A bal oldali megoldásban a szöveg írásával párhuzamosan folyamatosan kell gondoskodni annak vizuális megjelenítéséről is! Ez a jobb oldali változatban, ha sokkal kisebb mértékben is, de még így van. A LaTeX-változatban viszont semmi vizuális szerkesztés nincs, ugyanakkor olyan automatizmusok vannak benne (az a/az névelők közül a megfelelő kiválasztása, a hivatkozás számának meghatározása, a hivatkozás utáni -ra/re ragok közül a megfelelő kiválasztása, amihez hasonló képességeket egyetlen más dokumentumkészítő rendszer sem tud!

## 2.7.2. Fűzzük be a papírt!

Ha írógéppel írunk, kezdéskor – miután eldöntöttük mit írunk, cikket, levelet, könyvet – befűzzük a papírt. Ennek a fázisnak a LaTeX-ben való írásnál a `\documentclass` és néhány további parancs kiadása felel meg. Itt döntünk, és tudatjuk a LaTeX-hel, hogy mekkora papírra írjunk, A4-esre, B5-ösre, hogy hány hasábos legyen a szedés, hogy mekkora legyen a betűk mérete stb.

A következőkben röviden áttekintjük a leggyakrabban használt paramétereket, ezek teljes listáját az 5. fejezetben részletezzük.

**Papírméret** ♦ A `\documentclass` parancs neve és argumentuma között szögletes zárójelekben – tehát opcionális paraméterként – megadható a papír mérete, amelyre művünket legvégül nyomtatjuk. A leggyakrabban használt papírméret az A4-es, amihez a

```
\documentclass[a4paper]{article}
```

parancs kerül a forrásfájl első sorába. A papírméretet megadó további paraméterek: `a5paper` (A5), `b5paper` (B5), `letterpaper` (amerikai levélpapír, a papírméretek közül ez az alapértelmezés), `legalpaper`, `executivepaper`. A méretek megtalálhatók a táblázatok között (697. oldal, E.23. táblázat). E paramétereken kívül a szögletes zárójelpár között, vesszővel elválasztva továbbiak is megadhatók, melyeket a következőkben részletezünk.

**Betűméret** ♦ A betűmagasság mérésére a számítógépes tipográfiában ma legelterjedtebb mértékegységet, az ún. pontot használjuk. A standard dokumentumosztályok a 10, 11 és a 12 pontos betűket használják. Alapértelmezésben 10 pontosak a betűk. Ha ezen változtatni akarunk, a `11pt`, illetve a `12pt` paramétereket használjuk. Ha például egy cikket 12 pontos betűkkel, A4-es papírra szeretnénk írni, akkor a

```
\documentclass[12pt,a4paper]{article}
```

parancsot kell a forrásfájl első sorába írni. Vannak osztályok, melyek más méreteket is ismernek, például a James Kilfiger és Wolfgang May által írt `extsizes` csomaggal 8, 9, 10, 11, 12, 14, 17, 20 pontos betűmérettel is tudunk dolgozni. Az általuk használt osztályok neve a standardokéból az `ext` eléírásával keletkezik: `extarticle`, `extbook`, `extreport`, `extletter` és `extproc`. Például egy 20 pontos cikkhez így kezdjünk:

```
\documentclass[20pt,a4paper]{extarticle}
```

Befűzzük a papírt. 12 pontos írógép előtt ülünk.
Magyar betűket használunk. Követjük a magyar nyelvtani szabályokat.

Gépelni kezdünk.

**CÍM**
Szerző
2003-12-06

**Tartalomjegyzék**

1. Első fejezet ..... 1
   1.1. Első alfejezet ..... 1

**Kivonat.** Ez a kivonat.

**1. Első fejezet**

Szöveg *kiemelve*.

**1.1. Első alfejezet**

Hivatkozás az [1]-re.

**Irodalomjegyzék**

[1] ...

Kész a mű! Kifűzzük a papírt!

```
\documentclass[12pt]{article}
\usepackage{t1enc}
\usepackage[latin2]{inputenc}
\usepackage[magyar]{babel}

\begin{document}

\title{CÍM}
\author{Szerző}
\date{2003-12-06}
\maketitle

\tableofcontents

\begin{abstract}
   Ez a kivonat.
\end{abstract}

\section{Első fejezet}

Szöveg \emph{kiemelve}.

\subsection{Első alfejezet}

Hivatkozás \atold\cite{C}+ra.

\begin{thebibliography}{9}
\bibitem{C} ...
\end{thebibliography}



\end{document}
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<HTML lang="hu">
  <HEAD>
    <TITLE>CÍM</TITLE>
    <META http-equiv="Content-Type"
      content="text/html;
charset=ISO-8859-2">
    <META name="author" content="Szerző">
    <META name=date content="2003-12-06">
  </HEAD>
  <BODY>
    <H1>CÍM</H1>
      <P>Szerző</P>
      <P>2003-12-06</P>
    <H2>Tartalomjegyzék</H2>
      <P><A href="#1">1. Első fejezet</A>
      <BR><A href="#1.1">1.1.
Első alfejezet</A></P>
    <H4>Kivonat</H4>
      <P>Ez a kivonat.</P>
    <A name="1"></A>
    <H2>Első fejezet</H2>
      <P>Szöveg <EM>kiemelve</EM></P>
    <A name="1.1"></A>
    <H3>Első alfejezet</H3> <P>
      Hivatkozás az <A href=
"#C">[1]</A>-re.
    <H2>Irodalomjegyzék</H2>
      <OL><LI><A name=C></A>...</LI></OL>
  </BODY>
</HTML>
```

2.1. ábra. Különböző szerkesztési technikák összevetése egy nagyon egyszerű dokumentumon: bal oldalon a kész mű, mellette LaTeX-, jobb oldalon HTML-kód.

## 2.7.3. Oldalformátum

**Oldal és hasáb** ♦ Ha két hasábba szedjük cikkünket, a `\documentclass` parancsnak az opcionális `twocolumn` paramétert kell megadnunk, ha kétoldalasra szedjük, a `twoside` paramétert. A *kétoldalas szedés* itt azt jelenti, hogy a kinyomtatott szöveg megjelenése függ attól, hogy páros vagy páratlan oldalra kerül. Például ebben a könyvben a lapok fejléce és a kiszedett szövegnek az oldalon való elhelyezkedése más aszerint, hogy az oldal páros vagy páratlan sorszámú (részletesen lásd az 5. fejezetben).

Kétoldalas nyomtatásnál mindig a páratlan oldal a jobb oldali és a páros a bal. Az *egyoldalas szedés* esetén minden oldalt jobb oldalinak tekintünk.

› **Tipp:** *Kétoldalas nyomtatás, kétoldalas nyomtató*
Az, hogy a nyomtatónk meg tudja-e fordítani a papírt, és tud-e egy papírnak mindkét oldalára nyomtatni, az előbb tárgyalttól teljesen független kérdés. Ha kétoldalas nyomtatásra alkalmas nyomtatóval olyan szöveget nyomtatunk ki, melyet a LaTeX-hel kétoldalasra szedtünk, akkor ügyeljünk arra, hogy mindig páratlan sorszámú oldaltól kezdjük a nyomtatást (például az elsőtől). ‹

Előfordulhat, hogy nem az egész dokumentumot akarjuk kéthasábosra szedni, hanem csak egy részét, például a tárgymutatót. Ezt a `\twocolumn` paranccsal érhetjük el. Hasonlóan egy kéthasábos szedés közben egyhasábosra válthatunk a `\onecolumn` paranccsal. E parancsok kiadási helyén a LaTeX új oldalt kezd. Azt, hogy hogyan lehet egy oldal közepén egyhasábosról kéthasábosra váltani, vagy fordítva, a 4.6. szakaszban tárgyaljuk.

**Az oldal stílusa** ♦ Egy tipikus oldal három részből áll: fejléc, lábléc és köztük maga a szöveg. Általában a fejlécbe vagy a láblécbe kerül az oldalszám és a fejlécbe az úgynevezett *élőfej*. A könyvek, folyóiratok lapjainak tetején megjelenő tájékoztató címet, feliratot nevezzük élőfejnek. Az is gyakori, hogy a fejléc vagy a lábléc üres. Például ha egy rövid, egyetlen oldalas szöveget írunk, mindkettőt üresen hagyjuk.

LaTeX-ben az *oldal stílusán* azt értjük, hogy mit tartalmaz a fejléc és a lábléc (például a fejléc az élőfejet, a lábléc az oldalszámot). Az egész dokumentumra vonatkozóan a `\pagestyle`, csak egyetlen oldalra vonatkozóan a `\thispagestyle` paranccsal szabályozható. A `\pagestyle{empty}` parancs hatására minden oldalon üres marad a fej- és a lábléc is, a `\thispagestyle` parancs hatásásra csak egyetlen oldalon. A `\pagestyle{plain}` hatására üres marad a fejléc, a lábléc közepén pedig megjelenik az oldalszám. A `\pagestyle{headings}` parancs esetén attól függően, hogy milyen dokumentumosztálybeli műről van szó, az élőfejben megjelenik a fejezet és az alfejezet címe és az oldalszám, a lábléc üres marad. Kétoldalas szedés esetén az oldalszám a páratlan oldalak jobb felső sarkába, illetve a páros oldalak bal felső sarkába kerül. Egy kinyitott könyvben mindig a jobb oldal páratlan, a bal páros oldalszámú. A következő táblázat azt mutatja, hogy melyik fejezetcímadó parancs argumentuma kerül az élőfejbe a dokumentumosztálytól és az opcionális paraméterek értékétől függően. A zárójelbe tett paraméter az alapértelmezés.

| dokumentumosztály | article | | book, report | |
|---|---|---|---|---|
| opcionális paraméter | oneside | (twoside) | (oneside) | twoside |
| bal oldali élőfej tartalma | | \section | | \chapter |
| jobb oldali élőfej tartalma | \section | \subsection | \chapter | \section |

Végül a \pagestyle{myheadings} parancs segítségével az élőfejbe kerülő információt mi határozhatjuk meg. További részleteket erről a könyv 198. oldalán találunk.

Az oldal stílusát egyetlen oldal esetén is megváltoztathatjuk. Erre szolgál a \thispagestyle parancs. Hatása arra az oldalra vonatkozik, amelynek feldolgozása közben a LaTeX e parancsba botlik. Argumentumai megegyeznek a \pagestyle parancséival.

Nagy segítséget nyújt a fejléc és a lábléc testreszabásában a fancyhdr csomag, melyet a 200. oldalon tárgyalunk.

**Oldalszámozás** ♦ Az oldalakat általában arab számokkal számozzuk, de van példa a római számokkal, vagy éppen betűkkel való számozásra is.

LaTeX-ben a \pagenumbering paranccsal válogathatunk az oldalszámozás stílusai közül. Ennek alakja \pagenumbering{*oldalszám stílusa*}, ahol az *oldalszám stílusa* lehet arabic, roman, Roman, alpha, Alpha. Könnyű megjegyezni az oldalszámozás stílusát e paraméterekből. A következő táblázatban megadjuk az első négy oldalszámot mind az öt stílusban:

| *Parancs* | *Oldalszámok* |
|---|---|
| \pagenumbering{arabic} | 1, 2, 3, 4... |
| \pagenumbering{roman} | i, ii, iii, iv... |
| \pagenumbering{Roman} | I, II, III, IV... |
| \pagenumbering{alpha} | a, b, c, d... |
| \pagenumbering{Alpha} | A, B, C, D... |

Előfordul, hogy egy könyv vagy nagyobb cikk bevezető oldalai római számozásúak, a további rész arab. Ilyen oldalszámozás eléréséhez bárhol a könyv szövege előtt ki kell adnunk a \pagenumbering{Roman} (angol könyvben a \pagenumbering{roman}) parancsot, majd a könyv első fejezetét megnyitó \chapter (cikk esetén az első \section) parancs után a \pagenumbering{arabic} parancsot. A book dokumentumosztály \frontmatter és \mainmatter parancsai automatikusan ilyen oldalszámozást adnak.

A \pagenumbering parancs hatására az oldalszámozás mindig elölről kezdődik.

Az oldalszámok értékének átállításáról, általában a számlálókról a 3.2. szakaszban írunk.

## 2.7.4. Betűk

Befűztük a papírt, és döntöttünk a mű kinézetével kapcsolatos néhány kérdésben. A további esetleges döntésekhez többet kell tudnunk a betűkről.

A betűk legfontosabb jellemzőit a LaTeX négy szempont szerint csoportosítja: ezek a betű alakja, testessége, a betű családja és mérete.

**Betűváltozatok ♦** LaTeX-ben a betűváltozatokat a fenti első három ismertetőjegyük alapján különböztetjük meg, ezek szerint soroljuk osztályokba. Így tehát alakja szerint a betű lehet *álló*, *döntött* (más néven hamis kurzív), *kurzív* (más néven *dőlt*) vagy *kiskapitális* (más néven *kapitälchen*), testessége szerint lehet normál vagy félkövér, végül családja szerint lehet antikva, groteszk vagy írógép típusú.

E fogalmakat összefoglaljuk egy táblázatban magyar és angol nevükkel, kétbetűs TeX-rövidítésükkel és két LaTeX-paranccsal, majd mindent részletesen elmagyarázunk. (A betűváltozatok nevét azzal a betűvel szedtük, amit a neve kifejez.)

2.3. táblázat. Betűváltozatok

| Magyar | | Angol | | Rövid | Parancs | Deklaráció |
|---|---|---|---|---|---|---|
| alak | álló | shape | upright | `up` | `\textup` | `\upshape` |
| | *döntött* | | *slanted* | `sl` | `\textsl` | `\slshape` |
| | *kurzív, dőlt* | | *italics* | `it` | `\textit` | `\itshape` |
| | | | | `sc` | `\textsc` | `\scshape` |
| testesség | normál | series | medium | `md` | `\textmd` | `\mdseries` |
| | **félkövér** | | **boldface** | `bf` | `\textbf` | `\bfseries` |
| család | antikva | family | roman | `rm` | `\textrm` | `\rmfamily` |
| | groteszk | | sans serif | `sf` | `\textsf` | `\sffamily` |
| | írógép | | typewriter | `tt` | `\texttt` | `\ttfamily` |

Vizsgáljuk meg a különböző betűváltozatokat!

**Alak:** A döntött betű úgy néz ki, mintha az állót megdöntenénk. Figyeljük meg, hogy a kurzív, vagyis dőlt betűk teljesen különböznek az álló és a döntött betűktől. Összehasonlításul leírjuk az „igaz'' szót álló, döntött és kurzív betűkkel is: igaz – *igaz* – *igaz*, majd még egyszer, de most a TeX alapértelmezett Computer Modern fontjaival: igaz – *igaz* – *igaz*. A kurzív (dőlt) és a döntött betűket leggyakrabban szavak, mondatok hangsúlyozására, kiemelésére szokták használni. (A magyar tipográfiai hagyomány nem támogatja az antikva betűcsalád döntött betűinek használatát!)

A hétköznapi nyelven nagybetűknek nevezett betűket a tipográfiában *kapitális* vagy *verzál* betűknek nevezik. Innen a kiskapitális elnevezés arra a betűtípusra, melyben a kisbetűk helyett kisebb méretű nagybetűk állnak. Például Arany János neve álló és kiskapitális betűkkel leírva így néz ki: Arany János, A J´ .

**Testesség:** Testessége szerint a betű lehet *normál* vagy *félkövér*. A tipográfia ennél jóval több változatot ismer. Azokat a TeX is tudja, de alaphelyzetben csak a két említett lehetőséget kínálja fel. A testességbe a betű két jellemzője van beleolvasztva, a betűvonal *vastagsága* (*weight*) és a betű *szélessége* (*width*).

**Betűcsalád:** A LaTeX betűcsalád fogalmába a betűk két jellemzője lett beolvasztva. Az egyik aszerint különbözteti meg a betűket, hogy azok vonalvastagsága állandó, vagy változó. Megfigyelhetjük, hogy az éppen most használt betűk vonalai kis talpakban végződnek, a vonalak pedig nem egyenletesen vastagok. E betűk ne-

ve *antikva*. A talp nélküli, egyenletesen vastag vonalakkal megrajzolt betűk neve *groteszk*, ilyen betűk például ezek itt.

A betűk egy másik fontos jellemzője a *proporcionalitás*, vagyis az, hogy a betűk egyformán szélesek-e vagy sem. Technikai okokból a régi írógépek minden betűje ugyanolyan széles helyet foglalt el. Ugyanez igaz a DOS, illetve általában az ASCII terminálablakok karaktereire még ma is. Ugyanakkor a könyvekben leggyakrabban használt betűk régóta *proporcionálisak*, azaz változó szélességűek. Ilyenek az antikva betűk is. Az TEX harmadik betűcsaládját az *írógép típusú* betűk alkotják. Ez olyan betűkészlet, melynek minden egyes karaktere – egy adott betűméreten belül – azonos szélességű helyet foglal el (*monospaced*).

Következő példánk tíz 'm' és alatta tíz 'l' betűt mutat egyenlő szélességű, azaz írógép típusú betűvel és mellette proporcionális antikva betűvel leírva.

```
mmmmmmmmmm      mmmmmmmmmm
llllllllll      llllllllll
```

**Betűparancsok** ♦ A fenti betűváltozatokat előállító parancsok nevének `\text` az eleje, ami után a fent megadott kétbetűs rövidítés következik. Például döntött betűt a `\textsl` paranccsal lehet kiírni, tehát a `\textsl{döntető}` eredménye *döntető*. E parancsok egymásba ágyazhatók. Lássunk néhány példát:

```
01 abcdefghijklmnopqrstuvwxyz                  abcdefghijklmnopqrstuvwxyz
02 \textit{abcdefghijklmnopqrstuvwxyz}         abcdefghijklmnopqrstuvwxyz
03 \textsl{abcdefghijklmnopqrstuvwxyz}         abcdefghijklmnopqrstuvwxyz
04 \textsc{abcdefghijklmnopqrstuvwxyz}
05 \textbf{abcdefghijklmnopqrstuvwxyz}         abcdefghijklmnopqrstuvwxyz
06 \textsf{abcdefghijklmnopqrstuvwxyz}         abcdefghijklmnopqrstuvwxyz
07 \texttt{abcdefghijklmnopqrstuvwxyz}         abcdefghijklmnopqrstuvwxyz
08 \textsl{\texttt{döntött írógép}}            kurzív írógép
09 \textbf{\textit{félkövér kurzív}}           félkövér kurzív
```

E parancsoknak csak egy argumentumuk van, ha tehát nem használunk a parancs neve után kapcsos zárójeleket, akkor a parancs hatása csak a parancs utáni első betűre fog vonatkozni.

```
01 Csak \textbf ő sovány!                      Csak ő sovány!
```

**Deklarációk** ♦ Van a LATEX-parancsoknak egy osztálya, melyeknek nincs argumentumuk, és amelyek bizonyos paraméterek értékét átállítják, bizonyos parancsok hatását megváltoztatják anélkül, hogy közben bármit kinyomtatnának. Ezeket *deklarációknak* nevezzük. Egy deklaráció hatása (általában csak) arra a környezetre vagy blokkra terjed ki, amelyben szerepel. Ez azt jelenti, hogy ha egy `\begin{`*név*`}`, `\end{`*név*`}` parancspár között, vagy egy egybetartozó `{...}` zárójelpár között szerepel, akkor ezeken kívül már nem lesz érvényben. A hatása mindig ott kezdődik, ahol a deklarációs parancsot kiadtuk, és addig tart, míg a blokkon vagy környezeten belül egy azonos típusú másik deklaráció azt meg nem szünteti, vagy amíg egy `}` le nem zárja a blokkot, illetve amíg egy `\end` parancs le nem zárja a környe-

zetet. A betűváltozatokat felsoroló táblázatunk utolsó oszlopában a \text kezdetű parancsok deklarációs típusú változatai szerepelnek. Ezek mind a kétbetűs rövidítéssel kezdődnek, ami után a shape, series vagy family szavak közül a megfelelő következik.

A betűváltozatokat állító parancsok közül rövidebb szövegek esetén a \text- kezdetű parancsok javasolhatók, a deklarációkat inkább csak LaTeX makrók írásához használjuk. Így az alábbi példák is inkább csak a deklarációk működésének szemléltetésére szolgálnak.

```
{\ttfamily Dalszövegírógép}-programja két
slágert írt, \itshape ezt \upshape állítja.
{Nem tudom igaz-e, \slshape döntse el} más.
```

Dalszövegírógép-programja két slágert írt, *ezt* állítja. Nem tudom igaz-e, *döntse el* más.

**Kétbetűs deklarációk** ♦ Ha az Olvasó ismeri a TeX-et, vagy használta a LaTeX régi, 2.09-es változatát, akkor rövidebb, kétbetűs deklarációkra emlékszik, olyanokra, mint \it, \sl, \sc, \bf, \rm, \sf, \tt. Ezek használhatók szöveg- és matematikai módban is, de nem keverhetők, egymás hatását megszüntetik, azaz a később tárgyalandó ortogonalitási tulajdonsággal nem rendelkeznek:

```
{\it dőlt {\bf csak félkövér}}\\
{\itshape dőlt {\bfseries + félkövér}}
```

*dőlt* **csak félkövér**
*dőlt* ***+ félkövér***

**Kiemelés** ♦ A LaTeX, ezt később látni fogjuk, maga választja meg a fejezetcímek betűinek fajtáját, méretét. A címeken kívül a leggyakoribb ok, amiért a betű jellemzőin változtatni akarunk az, amikor egy szót vagy gondolatot ki akarunk emelni, ki akarunk hangsúlyozni. Az, hogy mit kell kihangsúlyozni, a szerző dolga, azt hogy hogyan, a tipográfusé. A gondolatok ritkán születnek tipográfiai megoldással együtt, nemigen szól senki így: „Támadt egy remek gondolatom, két kurzív és egy félkövér szó van benne!'' Ne mi döntsük el tehát, hogy hogyan emeljünk ki egy szót a szövegből.

Kiemelni, hangsúlyozni (angolul *emphasize*) a LaTeX \emph parancsával és \em deklarációjával lehet. Mindketten figyelik környezetük betűváltozatát, és eszerint választják ki a kiemelés megfelelő módját. A gyakran használt dokumentumosztályokban, pl. az article, report vagy book osztályokban antikvával szedett szövegben kiemelésre e parancsok a kurzív betűket használják (l. az alábbi példa első sorát), kurzív vagy döntött betűkkel írt, kiemelt szövegben pedig az antikva betűket (l. az alábbi példa második és harmadik sorát).

```
Kiemelés \emph{normál} szövegben.
\textit{Kiemelés \emph{kurzív} szövegben.}
\emph{Kiemelés \emph{kiemelt} szövegben.}
```

Kiemelés *normál* szövegben.
*Kiemelés* kurzív *szövegben.*
*Kiemelés* kiemelt *szövegben.*

Szavak kiemelésének egyéb módjai is vannak (l. a 2.5.3. és a 4.1.1. szakaszt).

**Normál stílus** ♦ Előfordul, hogy azt szeretnénk, bármilyen betűt is használ épp a LaTeX, a következőkben minden álljon vissza a kiindulási állapotba, vagyis a doku-

mentumunkat jellemző betűstílusba. Ezt teszi a `\textnormal` parancs és a `\normalfont` deklaráció.

**Kurzív kiegyenlítés** ♦ Egy kurzív vagy döntött betűs szöveg és egy utána következő álló betűs szöveg között kicsit több szóközre van szükség az egyenletes folthatás érdekében, a ferdén álló betű ugyanis „rádől'' az állóra, illetve az azt követő szóközre. E plusz köz kitételét nevezik *kurzív kiegyenlítés*nek. Ezt a LaTeX automatikusan elvégzi, ha a `\textit`, a `\textsl` vagy az `\emph` parancsot használjuk. Egy deklaráció csak paramétereket állít, nem ír ki semmit, még kis szóközöket sem, így az `\itshape`, a `\slshape` és az `\em` parancsok után magunknak kell gondoskodni az extra méretű közről. Ennek a szóköznek a beírására a `\/` parancs szolgál. A kurzív vagy döntött betűs rész utolsó parancsa legyen a `\/`. A kurzív kiegyenlítés nagysága láthatóvá válik, ha egy döntött és egy álló betűt egymás után írunk kiegyenlítéssel (*I*I) és anélkül (*II*).

Az alábbi példán három megoldást mutatunk egy szó kiemelésére, majd öt megoldást kurzívval való szedésére.

Ha csak az a célunk, hogy egy szót kiemeljünk, akkor az alábbi megoldások közül mindig csak az első megoldást használjuk, az összes többiben vizuális szerkesztőparancsokat kell kiadni! Három megoldás ráadásul tipográfiailag is rossz a kurzív kiegyenlítés hiánya miatt!

```
A \emph{zsiráf} bedőlt.
A {\em zsiráf\/} bedőlt.
A {\em zsiráf} bedőlt.
A \textit{zsiráf} bedőlt.
A {\itshape zsiráf\/} bedőlt.
A {\itshape zsiráf} bedőlt.
A {\it zsiráf\/} bedőlt.
A {\it zsiráf} bedőlt.
```

| | |
|---|---|
| A *zsiráf* bedőlt. | legjobb |
| A *zsiráf* bedőlt. | jó |
| A *zsiráf* bedőlt. | rossz |
| A *zsiráf* bedőlt. | jó |
| A *zsiráf* bedőlt. | jó |
| A *zsiráf* bedőlt. | rossz |
| A *zsiráf* bedőlt. | jó |
| A *zsiráf* bedőlt. | rossz |

**A betűk mérete** ♦ A *betűk méretén* vagy *fokozatán* ólombetűs szedés esetén annak az ólomlapkának a magasságát értik, amin a betű szerepel. Egy közös ábécébe tartozó betűk mindegyike ugyanakkora ólomlapkára kerül, így valójában nem a betű, hanem az ábécé méretéről kell beszélnünk. Eszerint ez az „o'' és ez a „k'' betű azonos méretű (10 pontos).

A számítógépes betűkre az előbb leírt gyakorlat lett átültetve: a betűket tervezője két olyan képzeletbeli vonal közé rajzolja, amelyek közé mind elférnek, és e két vonal távolsága lesz a betűk mérete (fokozata). E két vonal általában nem tapad mindkét oldalon az ábécé betűihez. Ez azért van, hogy a betűk ne ragadjanak össze még akkor sem, ha helykihagyás nélkül írunk egymás alá két sort. Ez a kis helykihagyás egyébként az ólomlapkákon is megvan. (E magyarázatból kis iróniával az olvasható ki, hogy egy betű mérete akkora, amekkorának azt tervezője és forgalmazója nevezi.)

Betűméret mérésére a legelterjedtebb mértékegység a *pont* (rövidítve `pt`) és a *didot pont* (ejtsd didó, rövidítve `dd`, eredeti helyesírással írva didôt). 1 pont kb. 0,3515 mm, míg 1 didot kb. 0,3761 mm, ami közel 7%-kal nagyobb. 1 mm kb. 2,845 pont és 2,659

2.4. táblázat. Betűméretek

| A LaTeX standard betűméret-deklarációi | | Az $\mathcal{AMS}$-LaTeX betűméret-deklarációi | |
|---|---|---|---|
| `\tiny` | ekkora | ekkora | `\Tiny` |
| | | ekkora | `\tiny` |
| `\scriptsize` | ekkora | ekkora | `\SMALL` |
| `\footnotesize` | ekkora | ekkora | `\Small` |
| `\small` | ekkora | ekkora | `\small` |
| `\normalsize` | ekkora | ekkora | `\normalsize` |
| `\large` | ekkora | ekkora | `\large` |
| `\Large` | ekkora | ekkora | `\Large` |
| `\LARGE` | ekkora | ekkora | `\LARGE` |
| `\huge` | ekkora | ekkora | `\huge` |
| `\Huge` | ekkora | ekkora | `\Huge` |

didot pont. Az amerikai nyomdászatban és a számítástechnikában főként a pontot használják. A didot az európai nyomdászat hagyományos mértékegysége. E könyv 10 pontos betűkkel készült, a címek betűnagysága 10-től egészen 25 pontig terjed, az apró betűs részek némelyike[9] 8 pontos. Legkisebb a matematikai képletekben az index indexe, mely 5 pontos, mint például a 4-es a $b^{a_4}$ képletben.

Érdemes tudni, hogy egy egész betűkészlet elkészítésekor azt minden fontosabb méretben és változatban külön megtervezik. Nem igaz, hogy a nagyobb méretű betűk egyszerűen csak a kisebbek felnagyításából kaphatók meg, mint ahogy az sem, hogy a kiskapitális betűk a nagybetűk lekicsinyítésével, vagy kisebb méretű nagybetűk alkalmazásával adódnak – még ha gyenge programok használják is e primitív technikát.

Láttuk, hogy a betűméretek megadhatók hosszmértékegységek segítségével. A dokumentum szerzőjének szerencsére nem kell azzal foglalkoznia, hogy mi hány pontos legyen, mert a LaTeX minden helyzetben dönt a betűk méretéről, tudja, mekkora legyen a cím és mekkora a lábjegyzet betűmérete. Arra az esetre, amikor nem felel meg a LaTeX által felkínált betűméret, egyszerű deklarációs parancsokat kínál a betűméret megváltoztatására. Alaphelyzetben 10 betűméret közül választhatunk, ha az $\mathcal{AMS}$ valamelyik dokumentumosztályát használjuk (`amsart`, `amsbook`,...), akkor 11 betűméret közül. Ezeket mutatja a 2.4. táblázat.

**Ortogonalitás** ♦ E matematikából kölcsönvett szó itt azt jelenti, hogy a betűk négy jellemzője, az alak, a testesség, a család és a méret egymástól függetlenül állítható. Például ha kiadjuk az `\sffamily` deklarációt vagy a `\textsf` parancsot, akkor a további szöveg családja groteszkre változik, de alakja, testessége és mérete nem változik. Ez a tulajdonsága nincs meg a betűváltozatokat állító kétbetűs deklarációknak (`\it`, `\bf` stb.), azok egymás hatását kioltják, csak a betűméreten nem változtatnak (csak a betűméretre ortogonálisak, egymásra nem). Egy álló, félkövér, groteszk, lábjegyzet méretű **szöveg** megkapható az alábbi két parancssorozat bármelyikével:

[9] Például ez a lábjegyzet ilyen.

```
\textup{\textbf{\textsf{\footnotesize szöveg}}}
{\upshape \bfseries \sffamily \footnotesize szöveg}
```

A különböző jellemzőkhöz tartozó parancsok sorrendje tetszőleges, így az előző két parancssorozattal azonos hatású az alábbi kettő is:

```
\textsf{\textup{\footnotesize \bfseries{szöveg}}}
{\sffamily \upshape \footnotesize \bfseries szöveg}
```

Előfordulhat, hogy valamely négy jellemzővel megadott betűkészlet számítógépünkön nem létezik. Ekkor a LaTeX egy automatikus fonthelyettesítő algoritmussal kiválaszt egy hasonló betűkészletet, és azt használja, a helyettesítésről pedig figyelmeztető hibaüzenetet ad.

## 2.7.5. Testreszabás

**Parancsok definiálása** ♦ Talán meglepi az Olvasót, de e részfejezetet azzal a tanáccsal kezdjük, hogy se az előzőekben tanult betűváltozatot állító parancsokat (l. 70. oldal, l. még 687. oldal), se a betűméretet állító parancsokat (l. 74. oldal) szöveg közben ne használjuk!

Egy példán mutatjuk meg, mire gondolunk. Képzeljük el, hogy egy olyan dolgozatot írunk, melybe programlistát helyezünk. A program parancsszavait tipográfiailag is meg kell különböztetni az egyéb szavaktól. Hogyan szednénk ki az alábbi programrészletet?

**if** korán kelsz
**then** aranyat lelsz
**else** nagyot alszol

Ha a dokumentumunkban csak ez az egyetlen programrészlet, akkor a megoldáshoz egyszerűen használható a `\textbf` parancs. Ha azonban több programrészletet is mellékelünk, mást kell tenni.

Egy olyan új parancsot kell konstruálni, mely az argumentumát programnyelvi utasításként írja ki. Új parancs definiálására használható a `\newcommand` parancs. Ha azt akarjuk, hogy pl. egy `\ut` nevű paranccsal jelöljük ki az utasítás kulcsszavát, és ez azt tegye, amit a `\textbf`, akkor csak annyit kell írni, hogy `\newcommand{\ut}{\textbf}`. Lássuk az előbbi példát:

```
\newcommand{\ut}{\textbf}
\ut{if} korán kelsz\\
\ut{then} aranyat lelsz\\
\ut{else} nagyot alszol
```

**if** korán kelsz
**then** aranyat lelsz
**else** nagyot alszol

A `\newcommand` első argumentuma az új parancs neve, míg a második argumentumban kell megadni azt, hogy mit csináljon. Ha ezután kiderül, hogy jobb lenne az utasítás kulcsszavait félkövér betű helyett írógép típusú nagybetűkkel írni, csak egyetlen sort, az `\ut` definícióját kell megváltoztatni:

```
\newcommand{\ut}[1]
    {\texttt{\MakeUppercase{#1}}}
\ut{if} korán kelsz\\
\ut{then} aranyat lelsz\\
\ut{else} nagyot alszol\\
```

IF korán kelsz
THEN aranyat lelsz
ELSE nagyot alszol

A \newcommand parancsot itt más alakban írtuk fel, aminek az az oka, hogy az \ut parancs definíciója összetettebb, ezért meg kell adnunk azt is, hogy az \ut parancs argumentuma hová kerüljön. A #1 jelzi az \ut egyetlen argumentumának helyét. Az \ut fenti definíciója egy ún. *makrót* definiál, ami azt jelenti, hogy az \ut{*szöveg*} minden meghívásakor a helyébe helyettesítődik a \texttt{\MakeUppercase{*szöveg*}} kód, és ez hajtódik végre.

A szögletes zárójelbe tett 1-es azt jelenti, hogy az \ut parancsnak csak 1 argumentuma lesz. A következő példában egy kétargumentumos parancs definíciójára adunk példát. Az \sztr (szótár) parancs az első argumentumában lévő szöveget normál betűkkel, a másodikban lévőt dőlt betűkkel írja ki, közéjük gondolatjelet tesz:

```
\newcommand{\sztr}[2]
  {\textnormal{#1} -- \textit{#2}}
\sztr{bold}{félkövér}\\
\sztr{extra bold}{kövér}
```

bold – *félkövér*
extra bold – *kövér*

› **Gyakorlat:** *Parancs definiálása*
Írjuk át az \sztr definícióját az előző programban úgy, hogy az eredetileg angol–magyar szótár magyar–angol szótárrá váljon! ‹

Ha egyetlen dokumentumon belül egy parancs definícióját meg akarjuk változtatni, akkor a \renewcommand parancsot kell használni a \newcommand-éval azonos szintaktika mellett. Például az előző szótári szöveg második része folytatódhat így:

```
\renewcommand{\sztr}[2]
  {\textit{#1} -- \textnormal{#2}}
\sztr{groteszk}{sans serif}\\
\sztr{kurzív}{italics}
```

*groteszk* – sans serif
*kurzív* – italics

› **Tipp:** *Rövidítések definiálása*
Hasznos a gyakran előforduló hosszú kifejezések helyébe egy parancsot definiálni, például a

```
\newcommand{\BME}{Budapesti Műszaki és Gazdaságtudományi Egyetem}
```

definíció után „a Budapesti Műszaki és Gazdaságtudományi Egyetemről'' írva csak annyit kell begépelni, hogy a \BME ről. Ugyanakkor viszont (LA)TEX parancsszavakat sose rövidítsünk így, mert az olvashatatlanná teszi a forráskódot! Ha nem akarunk annyit gépelni, használjunk olyan szövegszerkesztőt, melyben egyszerű billentyűkombinációval helyettesíthetőek a parancsok, vagy amelyekben menüből kiválaszthatjuk a parancsok, környezetek nevét. ‹

Parancsok és környezetek definiálásáról részletesen írunk a 3.3. és a 3.4. szakaszokban.

**Méretek változtatása ♦** A TEX-ben használható mértékegységek fel vannak sorolva a 696. oldalon és az E.21. táblázatban. Itt most csak négyet emelünk ki közülük, ezek a mm (miliméter), pt (pont), az em és az ex. Az em és az ex – a másik kettővel ellentétben – relatív mértékegységek, azaz méretük a szövegben épp használt betűtől függ. 1 em a betűfokozat nagyságával egyezik meg, vagy ahhoz közeli értékű; 1 ex a betűszem magasságával, pontosabban az 'x' betű magasságával egyezik meg.

A hosszúságadatokat a LATEX hosszúságparancsok formájában tárolja. Például azt, hogy a bekezdések első sorát mennyivel kell behúzni, a `\parindent` parancs tárolja. Ezt a méretet például a `\setlength` paranccsal állíthatjuk át:

```
\setlength{\parindent}{2em}
```

A TEX-ben a mértékegységek lehetnek rugalmasak, vagyis olyanok, amelyek értékét a TEX bizonyos határok között megváltoztathatja. Például szokás a bekezdések közti – a `\parskip` hosszúságparancsban tárolt – távolságot rugalmasra állítani, hogy a TEX elkerülhesse azt, hogy a bekezdés egyetlen sora kerüljön át a másik oldalra, vagy hogy csak az első sora maradjon az adott oldalon. Ilyen rugalmas mérték definíciója az alábbi:

```
\setlength{\parskip}{12pt plus 6pt minus 3pt}
```

Az `\addtolength` paranccsal növelni vagy csökkenteni tudjuk a hosszúságparancs méretét. Például növeljük meg az oldal tükörméretét vízszintesen 14 mm-rel, függőlegesen 2 cm-rel.

```
\addtolength{\textwidth}{14mm}  %% sorszélesség  = sorszélesség + 14mm
\addtolength{\textheight}{2cm}  %% tükörmagasság = tükörmagasság + 2cm
```

Ha e két sort valóban beírjuk a preambulumba, a tükör bal felső sarka helyben marad, és a tükör jobbra 14 mm-rel, lefelé 2 cm-rel megnő. Ezért az egész tükröt el kéne tolni balra 7 mm-rel és fölfelé 1 cm-rel, amit megtehetünk a következő módon:

```
\addtolength{\hoffset}{-7mm}  %% tükör eltolása vízszintesen balra
\addtolength{\voffset}{-1cm}  %% tükör eltolása függőlegesen fölfelé
```

## 2.7.6. Nagyobb feladatok szervezése

**Hosszú művek felosztása ♦** Ha nagyobb művet – könyvet, disszertációt – írunk, érdemes a szöveget kisebb, könnyen kezelhető részekre, külön állományokba osztani. Tegyük fel, hogy művünket három részre osztottuk: `bevezetes.tex`, `targyalas.tex`, `befejezes.tex`, a témában megszületett egyéb véleményeket egy `velemeny.tex` nevű fájlba gyűjtjük, melyet a `befejezes.tex` egy adott pontján akarunk betölteni, a `targyalas.tex` betölt egy ábrát, és végül egy táblázat készítését is tervezzük, amit majd egy `tablazatok.tex` nevű fájlba írunk, de az még nincs kész.

A munka összefogására készítünk egy `master.tex` nevű fájlt, ez lesz a lefordítandó állomány. Ebbe azonban nem sok szöveg kerül, minden a fent felsorolt fájlokban

van. Azokat a fájlokat, melyek szövegét új oldalon akarjuk kezdeni, az `\include` paranccsal töltjük be, a többit az `\input` paranccsal (`master.ind`, `velemenyek.tex`). A mutatókról már írtunk a 2.5.9. szakaszban, így a `master.ind` állomány elkészítésének módját ismerjük.

Amikor írunk egy részt, nem akarjuk folyvást az egész művet lefordítani, csak azt az egy részt, de a fejezetszámozást meg akarjuk tartani. Erre való az `\includeonly` parancs, mely lehetővé teszi, hogy csak az argumentumában – köztes szóközök nélkül, vesszővel elválasztva – felsorolt fájlok legyenek lefordítva. Fontos, hogy a `.tex` kiterjesztést nem szabad kitenni sem itt, sem az `\include` parancsban.

A 2.2. ábra ezt a „projektet'' szemlélteti. Látható rajta minden – a szerző által megírt – fájl, és látszanak az állományok betöltésének pontjai is. Az `\includeonly` parancs argumentumának itt mutatott felírása lehetővé teszi, hogy ha egy állományt nem akarunk lefordítani, elég legyen csak a neve elé tenni egy % jelet. Nem ugyanezt a hatást érjük el, ha csak a megfelelő `\include` parancsok elé teszünk %-ot, hisz akkor ezeket semmilyen módon sem veszi figyelembe a LaTeX, tehát pl. megváltoznak a fejezetsorszámok.

› **Gyakorlat:** *Hosszú művek felosztása*
Egy több fejezetből álló (pl. az internetről letöltött) művet osszunk fejezetenként külön állományokba, az `\includeonly` paramétereit pedig a terminálról olvassuk be a `\typein` paranccsal! ‹

**Több fájl egybefűzése** ♦ Az előbb leírt eljárás fordítottjára is szükségünk lehet, ha például el akarunk küldeni egy dokumentumot az Interneten. Egy dokumentum LaTeX forrásállománya is több darabból állhat, és külön fájlokban lehetnek a beillesztendő rajzok és a felhasznált stílusfájlok is. Lehetőség van arra, hogy ezeket egyetlen állományba fűzzük. Ehhez a `filecontents` környezet használható. Abba a fájlba, amelyik a `\documentclass` parancsot tartalmazza, a `\documentclass`-szal kezdődő sor elé beilleszthető az összes többi fájl egy-egy `filecontents` környezetbe:

```
\begin{filecontents}{file.tex}
  a beillesztett állomány tartalma
\end{filecontents}
\documentclass{article}
  az eredeti állomány tartalma
```

A `filecontents` környezet egyetlen argumentuma[10] a fájl neve. Egymás után több fájl is beilleszthető a `\documentclass` parancs elé. Amikor egy ilyen összefűzött fájlt lefordítunk, akkor a LaTeX automatikusan kiírja a `filecontents` környezetben lévő szöveget a környezet argumentumában megadott nevű fájlba, feltéve, hogy ilyen nevű fájl még nincs. Ha ilyen nevű állomány már létezik, nem írja fölül! A LaTeX egy rövid megjegyzést is ír a fájl elejére. Például tekintsük az alábbi egyszerű kódot, melyet egy `master.tex` fájlba írtunk:

```
01 \begin{filecontents}{sajat.sty}
```

[10] A `\begin` parancs argumentuma a `filecontents` környezetnév, a `filecontents` környezet argumentuma ezután következik.

master.tex

```
\documentclass[11pt]{book}

\usepackage[T1]{fontenc}
\usepackage[latin2]{inputenc}
\usepackage[magyar]{babel}
\usepackage{graphics,amsmath}

\title{A mű címe}
\author{Szerző Neve}
\date{}

\includeonly{
    bevezetes,%
    targyalas,%
    befejezes,%
%    tablazatok,%
}
```

```
\begin{document}

\frontmatter
\maketitle
\tableofcontents
\include{bevezetes}

\mainmatter
\include{targyalas}
\include{befejezes}

\appendix
\include{tablazatok}

\backmatter
\addcontentsline{toc}
     {chapter}{Tárgymutató}
\input{master.ind}

\end{document}
```

bevezetes.tex

```
\chapter{Bevezetés}

Blabla, blab labla
Blabla, blab labla
...
Blabla, blab labla
```

targyalas.tex

```
\chapter{Tárgyalás}

Glugy glubb gugylu
...
\includegraphics{abra}
...
```

befejezes.tex

```
\chapter{Befejezés}

Klitty, klotty kli
...
\input{velemenyek}
...
```

velemenyek.tex

```
\section{Egyéb vélemények}

Blablablablabla
Blablablablabla
```

abra.eps

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: abra.eps
...
%%BoundingBox: 0 0 120 110
```

2.2. ábra. Az ábra az \include, az \includeonly, az \includegraphics és az \input parancs használatát és egy nagy dokumentum összeállításának módját szemlélteti

```
\newcommand*{\EU}{Európai Unió}
\end{filecontents}

\documentclass{article}
\usepackage[latin2]{inputenc}
\usepackage{t1enc,sajat}
\begin{document}
Beléptünk az \EU ba.
\end{document}
```

Ha ezt lefordítjuk, a LaTeX egy `sajat.sty` fájlt hoz létre a következő tartalommal:

```
%% LaTeX2e file `sajat.sty'
%% generated by the `filecontents' environment
%% from source `master' on 2004/02/21.
%%
\newcommand*{\EU}{Európai Unió}
```

Abban az esetben, ha egy olyan fájlt illesztünk LaTeX-programunk elejére – például egy grafikai állományt –, amelyiket a fenti négy sor hozzáadása tönkre tenne, akkor a `filecontents` helyett a `filecontents*` környezetet kell alkalmazni.

## 2.7.7. Vizuális szerkesztés

**Térközök** ♦ A megfelelő térközök megválasztása a különböző tipográfiai elemek közt fontos lépése a vizuális szerkesztésnek. Vízszintes térköz létrehozására a `\hspace`, míg függőlegesére a `\vspace` parancs használható. Mindkét parancs egyetlen argumentuma a térköz nagyságát adó méret, vagy hosszúságparancs. A TeX a sor elején a vízszintes, a lap tetején a függőleges térközt automatikusan lenyeli, ezért mindkét parancs *-os változata arra szolgál, hogy e helyekre is lehessen térközt rakni. A következő bekezdésben példát találunk alkalmazásukra, részletesen a 3.1.3. és a 3.1.4. szakaszban foglalkozunk velük.

› **Tipp:** *A vizuális parancsok használata*
Egy nagyon elterjedt, kényelmesnek tűnő, de a dokumentum szerkezetét csak összekuszáló gyakorlat az, amely a szöveg elemeinek (pl. fejezetcím, tétel) a LaTeX vizuális jelölő parancsaival (pl. `\textbf`, `\textit`, `\small` stb.) ad formát, majd az így kapott elemek közé a `\smallskip`, `\medskip` és `\bigskip` parancsokkal kisebb-nagyobb térközöket rak. Ne kövessük e gyakorlatot! ‹

**Bekezdésdobozok** ♦ Ha olyan dokumentumot készítünk, mely nem sorolható a standard dokumentumosztályok közé, ugyanakkor sajátos tipográfiai elemeket használ (pl. szórólap, hirdetmény), akkor sokat segíthet a `\parbox` parancs és a `minipage` környezet, melyek adott szélességű, bekezdésszerűen szedett dobozt hoznak létre. Ezeknek az úgynevezett *bekezdésdobozok*nak opcionálisan megadható a magasságuk is. Az alábbi példában először egy 50 mm széles, majd alatta két 23 mm széles, egymástól 4 mm-re lévő dobozt adunk meg, melyek közül az elsőnek a közepe, a

másodiknak a felső sora van igazítva az alapvonalra. Az alapvonalat a sor végére tett XX jelzi.

```
\small
\parbox{50mm}{Ez egy 50\,mm széles
bekezdésdoboz, melybe most valamit írok.}
\vspace{5mm}

\noindent\parbox[2cm]{23mm}{Ez egy 23\,mm
széles, 2\,cm magas doboz.}%
\hspace{4mm}%
\parbox[t]{23mm}{Ez egy 23\,mm széles,
felső sorával igazított bekezdésdoboz.} XX
```

Ez egy 50 mm széles bekezdésdoboz, melybe most valamit írok.

Ez egy 23 mm széles, 2 cm magas doboz.

Ez egy 23 mm széles, felső sorával igazított bekezdésdoboz. XX

**Bekezdések elválasztása ♦** A bekezdéseknek a szövegben egymástól való elválasztására két megoldás terjedt el széles körben. Az egyik az ún. *behúzás* alkalmazása minden bekezdés elején. A behúzás azt jelenti, hogy a bekezdés első sora „beljebb van húzva'', mint a többi sor, azaz beljebb kezdődik. A LaTeX-ben a behúzás méretét a `\parindent`[11] parancs tárolja. Az ilyen hosszúságparancsok használatát részletesen a 3.1. fejezetben írjuk le.

Ritkán, de előfordulhat, hogy egy új bekezdésként, behúzással kezdődő szöveget behúzás nélkül szeretnénk kezdeni. Ekkor a `\noindent` parancsot kell használni.

A másik gyakori megoldás bekezdések elválasztására az, amikor a bekezdés első sora és az előző bekezdés utolsó sora között nagyobb a sorköz a normál sorköznél. A `\parskip` nevű parancs tartalmazza azt a távolságot, amennyivel a bekezdések közti távolság nagyobb a normál sorköznél.

**Sorzárás ♦** A kiemelés egyik módja lehet a szövegrész sorainak balra, jobbra vagy középre zárása, illetve a sorzárás megváltoztatása. Balra zárásra a `flushleft`, jobbra zárásra a `flushright`, középre zárásra a `center` környezet használható.

```
\begin{flushleft}
Ezt a szöveget balra zárjuk. Minden
sor a bal margónál kezdődik. A~sortörő
,,rep-rep'' parancs itt is működik,\\
mint látjuk.
\end{flushleft}
```

Ezt a szöveget balra zárjuk. Minden sor a bal margónál kezdődik.
A sortörő „rep-rep'' parancs itt is működik,
mint látjuk.

A jobbrazárás néhánysoros szövegek kiemelésére használatos (pl. fejezetkezdő mottóknál).

[11] Behúzás – angolul *indent*.

```
\begin{flushright}
Ezt a szöveget jobbra zárjuk. Minden
sor a jobb margónál végződik. A~sortörő
,,rep-rep'' parancs itt is működik,\\
mint látjuk.
\end{flushright}
```

Ezt a szöveget jobbra zárjuk. Minden sor a jobb margónál végződik. A sortörő „rep-rep” parancs itt is működik, mint látjuk.

A középre zárást is rövid szövegek kiemelésére, például címek esetén használjuk, hosszabb szöveg így szedve nehezen olvasható.

```
\begin{center}
Ezt a szöveget középre zárjuk. Minden
sor középre kerül. A~sortörő
,,rep-rep'' parancs itt is működik,\\
mint látjuk.
\end{center}
```

Ezt a szöveget középre zárjuk. Minden sor középre kerül. A sortörő „rep-rep” parancs itt is működik, mint látjuk.

Egy bekezdésre való érvénnyel a `flushleft`, `flushright`, `center` környezetek helyett a `\raggedright`, `\raggedleft`, ill. `\centering` parancsok is használhatók.

**Sortörés, oldaltörés** ♦ A \\ parancs új sort, míg a `\newpage` parancs új oldalt kezd kiadása helyén. A `\nolinebreak[0]`, `\nopagebreak[0]` parancsok a sor-, illetve oldaltörés elkerülését kérik, míg a `\nolinebreak[4]`, `\nopagebreak[4]` parancsok megtiltják azt. Bővebben lásd a könyv 140. oldalán.