LaTeX nem túl röviden

Csárdi Gábor

1998. október

Tartalom

Előszó

A LaTeX egy szövegformázó rendszer. A legkülönfélébb dokumentumok előállítására képes nyomdai minőségben, de igazán a tudományos, matematikai írásoknál csillogtatja képességeit. Segítségével az olvasó otthoni számítógépén állíthat elő olyan minőségű dokumentumokat, amelyek minden kiadónak becsületére válnának.

A LaTeX a TeX programot használja a szövegek megformázására. Tulajdonképpen a LaTeX nem más, mint egy makrócsomag, TeX parancsok gyűjteménye. A TeX egy szövegszedő és szövegformázó program, amit Donald E. Knuth amerikai matematikus készített. A program szinte minden számítógéptípusra és operációs rendszerre létezik, az IBM PC gépektől kezdve egészen a nagyszámítópekig, a DOS-tól a VMS-en át a Unix-ig.

Mind a TeX, mind a LaTeX teljesen ingyenes, a programért semmit nem kell fizetnünk.

A LaTeX rendszert eredetileg Leslie Lamport készítette. A rendszert A LaTeX3 csapat Frank Mittelbach vezetésével továbbfejlesztette, kiegészítéseket írt hozzá, és a LaTeX 2.09 után megjelent sokféle változatot egységesítette. Az új változat neve LaTeX 2_ε lett. Könyvemben ezt a változatot mutatom be.

A LaTeX rendszer a legtöbb egyetemi számítógépes rendszeren megtalálható, telepített, használható állapotban. Mivel a LaTeX dokumentumok fordítása és nyomtatása rendszerenként eltérő lehet, erre a könyvben nem térek ki, fordulj bizalommal a helyi LaTeX szakemberhez, vagy ahhoz az emberhez, akitől ezt a könyvet kaptad.

Jelen könyv alapja a *The not so short introduction to LaTeX 2_ε*, amit Tobias Oetiker, Hubert Partl, Irene Hyna és Elisabeth Schlegl írt. Az eredeti könyv megtalálható a `CTAN:/tex-archive/info/lshort` címen.

1. fejezet

Miért LaTeX?

1.1. Miért mi?

Ha az olvasó még mindig nem tudná miről is van szó, mert mondjuk nem érti a „szövegszedő”, „szövegformázó” kifejezéseket, ne csüggedjen, rögtön kiderül mivel is áll szemben.

Ha egy szerző – azaz a Kedves Olvasó – publikálni szeretné egy művét, akkor elviszi annak kéziratát egy kiadóhoz. Tegyük fel, hogy a kiadó kiadásra érdemesnek találja a művet. A kiadó műszaki szerkesztője meghatározza azt, hogyan fog majd kinézni a könyv. Meghatározza az oszlopok szélességét, a betűtípust, hogy mennyi üres hely legyen a címsorok felett és alatt, és még ezernyi más dolgot.

A szerkesztő megpróbálja kitalálni mi járhatott a szerző fejében, amikor a kéziratot írta. Minden tudását beveti, hogy a lehető legjobban visszaadja a szerző gondolatait. A kész tervet hozzákapcsolja a kézirathoz, aztán az egészet odaadja a szedőnek, aki elkészíti a könyvet végső formájában.

Az általam bemutatott rendszerben a tervező szerepét a LaTeX, a szedő szerepét a TeX veszi át. Persze, mivel a LaTeX csak egy számítógépprogram, nem gondolkodó ember, több utasításra van szüksége, mint egy könyvtervezőnek. Ezeket az utasításokat a szerző beleírja a kéziratba. Ha a kézirattal elkészült, „odaadja” azt a LaTeX-nek, az értelmezi az utasításokat, és megmondja a TeX-nek hogyan nézzen ki pontosan a dokumentum. A folyamat menetéből következően a szerző a kézirat gépelése közben nem látja milyen lesz a végső kinyomtatott forma. Lehetősége van azonban rá, hogy nyomtatás előtt a képernyőn megtekintse azt.

1.2. Pro és kontra

Az írónak ezzel a fejezettel az a célja, hogy rábeszélje a LaTeX használatára azokat, akik eddig idegenkedtek tőle. Lássunk hát tíz érvet a LaTeX mellett.

Gyönyörűszép dokumentumok. Ez nem túlzás! A mai WYSIWYG (What You See Is What You Get – Amit Látsz Azt Kapod, a kifejezés azt jelenti, hogy a kinyomtatott dokumentum pontosan olyan lesz, amilyennek a képernyőn látjuk) szövegszerkesztőkhöz képest ez feltétlenül így van.

Olvasható dokumentumok. Ha dokumentumunkat WYSIWYG rendszerrel készítjük, akkor valószínűleg izlésesen elrendezett, szemet gyönyörködtető oldalakat kapunk, hiszen erre figyelünk oda leginkább, hosszú percekig, órákig alakítgatjuk a mű formáját, amíg tökéletes nem lesz. A könyvek oldalai azonban nem abból a célból készülnek, hogy a galéria falán gyönyörködjünk bennük, hanem sokkal inkább azért, hogy a Tisztelt Olvasó elolvassa őket. Ízléses, szemet gyönyörködtető dokumentumunk vajon olvasható-e is lesz-e? Nem feltétlenül.

A LaTeX sok fontos apróságra figyel az olvashatóság érdekében: Vigyáz, hogy ne legyenek túl hosszúak a sorok, ne fárasszák a szemet, kicsit több helyet hagy ki két mondat között, mint a szavak között, sőt a betűk távolságát is változtatja azok alakjától függően.

Könnyű kezelhetőség. Értem ezalatt, hogy a LaTeX kézirat megírása nagyon egyszerű. Ezzel a WYSIWYG felhasználók többsége nem ért egyet, pedig tényleg így van. Egyrészt csak néhány egyszerű parancsot kell megjegyeznünk. Másrészt pedig a kézirat írásakor nem kell azzal foglalkoznunk, hogy a dokumentum adott része hogyan fog kinézni, csak magára a tartalomra koncentrálhatunk.

Rengeteg kiegészítés. Szinte nincs is olyan feladat, amit a LaTeX, és esetleg valami kiegészítés, segítségével ne oldhatnánk meg. Példaként említhetjük kották, keresztrejtvények szedését, kémiai képletek írását, vagy sakkjátszmák nyomtatását. Ha valamilyen speciális dokumentumot szeretnénk előállítani LaTeX-hel, nézzünk közül, mert a kellő kiegészítést lehet, hogy már régen megírta valaki. A szerző például a könyvben szereplő kétoszlopos példák nyomtatásához írt egy saját kiegészítést, amikor megtalálta az akkor már két éve létező, ugyanezt a feladatot ellátó csomagot.

Ingyenes. Mint már említettem, a TeX, a LaTeX, és természetesen a hozzájuk írt kiegészítések is mind ingyenesek, sőt forráskódjuk is hozzáférhető.

Eszközfüggetlen. A TeX és a LaTeX szinte minden számítógépen és operációs rendszeren használható. A különböző gépeken futó TeX változatok ugyanabból a kéziratból ugyanazt a dokumentumot állítják elő. Az előállított dokumentum bármilyen számítógépen, bármilyen operációs rendszeren, bármilyen nyomtatón kinyomtatható.

Tudományos munkák egyszerűen. A LaTeX segítségével nagyon egyszerű tudományos, főleg matematikai témájú könyvek készítése. Ez ugyan első próbálkozáskor még a szerzőnek sem tűnt így, mégis igaz. Főleg akkor fogja ezt az olvasó tapasztalni, ha nem TeX alapú rendszeren próbál matematika tárgyú könyveket írni.

Többnyelvű. Szinte minden nyelvre létezik LaTeX változat, vagy LaTeX kiegészítés. Példaként elég csak a cirill betűs, vagy az arab változatot említeni. Ez a könyv egy teljesen magyar LaTeX-hel készült, ékezetekkel, magyar elválasztási szabályokkal. Ez nem azt jelenti, hogy a kézirat írásakor beírt parancsok magyar szavak angol szavak helyett, de igény esetén ez is megoldható. Egy dokumentumon belül több nyelvet is használhatunk.

A TEX Users Group. Létezik egy nemzetközi szervezet, a TUG (TEX Users Group), ami folyamatosan beszámol a TEX-hel kapcsolatos eredményekről saját folyóiratában, a TUGBOAT-ban.

Ráadás. A fenti kilenc érv ellenére a TEX-et jóval kevesebben használják, mint a WYSIWYG szövegszerkesztőket. A felhasználók többségének a TEX és a LATEX valami misztikus dolog, amit csak profik képesek használni. Így ha TEX-et használsz, környezeted jó eséllyel számítástechnikai zseninek fog tartani.

Ezek után az olvasó is láthatja, hogy a LATEX rendszert érdemes kipróbálni. Biztosíthatom, nem fogja megbánni.

Még akkor sem, ha a LATEX-nek vannak hátrányai is.

- WYSIWYG rendszerről LATEX-re áttérve, nehéz megszokni azt, hogy a dokumentum formáját nem látjuk a képernyőn, nem tudjuk, hogyan fog kinézni. A kész dokumentum viszont nyomtatás előtt megtekinthető a képernyőn, ekkor elvégezhetjük az esetleg szükséges módosításokat.

- A teljes LATEX rendszer nagyon sok merevlemezterületet foglal. Ez azonban még mindig kevesebb, mint amennyit például a *Microsoft Word for Windows 6.0*. A processzoridő tekintetében pedig egyetlen WYSIWYG rendszer sem veheti fel a versenyt a LATEX-hel, mivel ezek a kézirat írásakor is rengeteg processzoridőt követelnek, míg a LATEX csak a dokumentum formázásakor terheli le a processzort.

- Egy teljesen egyedi stílus létrehozása nagyon nehéz, kezdők számára lehetetlen feladat. Azt pletykálják, hogy ez a most készülö LATEX3 egyik fő fejlesztési területe.

- A LATEX-nek van – pontosabban volt – még egy nagy hátránya. Ez előtt a könyv előtt egyáltalán nem volt hozzá magyar nyelvű dokumentáció. Persze a könyv nem tér ki a LATEX összes parancsára, meg sem próbálja bemutatni határtalan lehetőségeinek teljes skáláját – a szerző úgy érzi erre nem is vállalkozhat – csak annyira merül bele a LATEX mélységeibe, amennyire egy felhasználónak erre szüksége lehet nem különösebben speciális dokumentumok formázásához.

2. fejezet

Amit feltétlenül tudnod kell

2.1. Fájlok

A LaTeX a kéziratból legalább három fájlt állít elő, de neked ezek közül csak eggyel, rosszabb esetben kettővel kell foglalkoznod. A legfontosabb fájl természetesen a megformázott dokumentum, ennek kiterjesztése `.dvi` lesz. A kiterjesztés arra utal, hogy a fájl eszközfüggetlen (DeVice Independent). A másik fájl, amire esetleg még szükséged lehet, a naplófájl, ennek kiterjesztése `.log`. Ebbe a fájlba ír a LaTeX minden fontos információt a dokumentum formázása közben. Ezeket általában a képernyőre is kiírja, ha azt nincs időd elolvasni (nem azért mert nem érsz rá, hanem mert gyorsan eltűnik a kérpernyőről), vagy később van szükséged valamilyen információra, nézd meg a naplófájlt!

2.2. A LaTeX kézirat

A LaTeX kézirat egy egyszerű `ASCII` fájl, bármilyen szövegszerkesztővel létrehozhatjuk. Álljunk csak meg egy szóra! Ezek szerint a LaTeX használatához egy másik szövegszerkesztő is szükséges? Igen, és nem. Az egész zűrzavar a szövegszerkesztő szóból adódik. A magyar „szövegszerkesztő" szónak az angol nyelvben két kifejezés felel meg. Az egyik a „text editor", a másik a „word processor". Az első egy olyan program, aminek a segítsével `ASCII` szövegeket, számítógépes szövegfájlokat állíthatunk elő, egy számítógépprogram forráskódját, vagy épp egy LaTeX kéziratot. Ilyen program például az `emacs`, a `vi`, vagy a DOS `edit` programja. A második kifejezés pedig olyan programot jelent, ami kifejezetten nyomtatott dokumentumok előállítására szolgál, általában WYSIWYG rendszerben. Itt a szerző jobbnak tartja a „szövegformázó" kifejezést. Ilyen programok például a *Microsoft Word*, vagy a *Wordperfect*. Nos a LaTeX-hez csak egy „text editor" programra lesz szükségünk, ami `ASCII` szöveget állít elő. Mivel ilyen program minden rendszeren megtalálható, ez nem jelent akadályt.

Tehát a LaTeX kézirat egy `ASCII` fájl. Ez a fájl tartalmazza a dokumentum szövegét is, és a LaTeX-nek szóló utasításokat is.

2.2.1. Üres helyek

A LaTeX kéziratban a szavakat természetesen *szóköz* karakterek választják el egymástól. A LaTeX azonban nem veszi figyelembe, hogy hányszor ütjük le a szóköz billentyűt két szó között, több szóközt is egy szóköznek tekint. Sőt, az újsor karaktereket is szóközöknek tekinti, a kézirat sortörései semmilyen hatással nincsenek a kész dokumentum sortöréseire. Ha viszont két újsor karakter áll egymás után (ez egy üres sort jelent a kéziratban), azzal jelezhetjük a LaTeX-nek, hogy új bekezdést akarunk kezdeni. Ha több üres sor van a kéziratban, azt a LaTeX egy üres sornak tekinti. Egyszerű, igaz? Figyeljük meg a következő példát! A jobb oldalon a kézirat egy részlete látható, a bal oldalon pedig ugyanez a részlet a kész dokumentumból.

Teljesen mindegy, hogy egy vagy több szóközt hagyunk ki a szavak után.
Egy üres sorral kezdhetünk új bekezdést.

```
Teljesen mindegy, hogy egy vagy
több     szóközt
hagyunk              ki a szavak
után.

Egy üres sorral kezdhetünk új
bekezdést.
```

2.2.2. Különleges karakterek

Van néhány karakter, ami a LaTeX számára különleges jelentéssel bír. Ha ezeket a karaktereket beírjuk a kéziratba, azzal valószínűleg nem a kívánt hatást érjük el. Ezek a karakterek:

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

Ha esetleg a dokumentum szövegében ezen jelek valamelyikére van szükségünk, akkor a jel elé írj egy *balraper* ('\') karaktert az adott jel elé, illetve a balraper karakter helyett a `$\backslash$` szót gépeld be, pontosan úgy, ahogy itt áll, '$' jelek között:

Az emberek 25%-a havi 15$-t fizet, hogy nézhesse a Tom & Jerry Show-t.
`~/public_html#ide`
A megoldás a {Páros számok} halmaza.
A '\' karakterrel jelezd, ha parancsot írsz.

```
Az emberek 25\%-a havi 15\$-t
fizet, hogy nézhesse a
Tom \& Jerry Show-t.

\texttt{\~{}/public\_html\#ide}

A megoldás a \{Páros számok\}
halmaza.

A '$\backslash$' karakterrel
jelezd, ha parancsot írsz.
```

2.2.3. LaTeX parancsok

Tudjuk már, hogy a kézirat tartalmazza mind a dokumentum szövegét, mind a LaTeX-nek szóló utasításokat. Ezeket valahogy el kell különítenünk egymástól. Ezt úgy érjük el, hogy kiválasztunk egy speciális karaktert, ami szövegben ritkán

fordul elő, ezzel a karakterrel kezdődnek majd az utasítások. Ez a karakter a \ karakter, minden LaTeX parancs ezzel a karakterrel kezdődik. Természetesen ez azt jelenti, hogy a \ karaktert nem használhatjuk szövegeinkben, hiszen a LaTeX azt egy parancs kezdetének veszi. Azt, hogy a parancs után hol kezdődik ismét a dokumentum szövege, a LaTeX automatikusan állapítja meg.

Ebből a szempontból kétféle LaTeX parancs létezik:

- Ha a \ karaktert egy betű követi, akkor a parancs végét az első nem betű karakter jelenti, ez már nem tartozik a parancshoz. Ha az első nem betű karakter szóköz, azt figyelmen kívül hagyjuk. Ezeket a parancsokat *parancsszavaknak* nevezzük.

- Ha a \ karaktert nem betű karakter követi, akkor a parancs csak egy karakter hosszú lesz, az ilyen parancsot nevezzük *parancskarakternek*. Ekkor a \ jel és a nem betű karakter után álló következő karakter már a szöveg részének számít (hacsak nem ismét egy különleges jel, pl. \).

Figyelem! A LaTeX mindig megkülönbözteti a kis- és nagybetűket, ez a parancsszavak esetében is így van! Így tehát a `\latex`, a `\LaTeX`, és a `\LATEX` három különböző parancsot jelent.

Lássunk egy példát! A `\LaTeX` parancs beszúrja a szövegbe a LaTeX logot, a \\ parancskarakter pedig egy sortörést eredményez.

A LaTeX-et igazán könnyen használhatod, csak néhány rövid
parancsot
kell megjegyezned.

```
A \LaTeX-et igazán könnyen
használhatod, csak néhány
rövid\\
parancsot\\
kell megjegyezned.
```

A példában három parancs található. A `\LaTeX` parancsszó, és a \\ parancskarakter kétszer. Az első parancs egészen a - karakterig tart, mivel ez az első nem betű karakter. Figyeljük meg mi történik, ha egy parancszó után szóköz karakter áll!

Az igazán TeXnikás LaTeX felhasználó tudja hogyan hagyjon ki helyet a parancsszavak után.

```
Az igazán \TeX nikás
\LaTeX\ felhasználó tudja hogyan
hagyjon ki helyet a parancsszavak
után.
```

Ebben a példában is három LaTeX parancs szerepel, ezek a `\TeX` (megszerkeszti a TeX logot), a `\LaTeX`, és a \␣ (balraperjel és egy szóköz). Az utolsó nem más, mint egy parancskarakter, hiszen \ jel után áll, és nem betű. A \ parancs a LaTeX számára egy figyelmen kívül nem hagyható szóközt jelent. Ezt jegyezd meg, mert még szükséged lehet rá. Például ha három szóköz karaktert szeretnél írni egymásután, a \␣\␣\␣ parancsokat használd! Ha csak egyszerű szóközöket írnál, akkor a LaTeX ezeket csak egy szóköznek tekintené.

Némely parancsnak extra paraméterekre is szüksége van, ilyenkor ezeket kapcsos zárójelek között kell megadni. Opcionális paraméterek is előfordulhatnak,

ez azt jelenti, hogy a paramétert nem kötelező, de meg lehet adni. Az opcionális paramétereket szögletes zárójelek közé kell írni. Lássunk néhány példát!

A speciálisan iskolai pszichés problémák megoldásával az *iskolapszichológia* foglalkozik.

```
A speciálisan iskolai pszichés
problémák megoldásával
az \emph{iskolapszichológia}
foglalkozik.
```

Szeretnék itt új sort kezdeni. Köszönöm.

```
Szeretnék itt új sort\linebreak[3]
kezdeni. Köszönöm.
```

A `\emph` parancs a paraméterét kiemelten szedve – általában kurzíválással – nyomtatja ki, a `\linebreak` parancs pedig sortörést kér. Utóbbinak meg lehet adni – nem kötelező – a sortörés fontosságát egy számmal, ezt láthatod a példában.

2.2.4. Csoportosítás

Akik járatosak valamely programozási nyelvben, azok tudják, hogy esetenként szükség van a program egy részének egy egységként kezelésére. Ez a dokumentumok esetében sincs másképp. Nem kell megijedned, nagyon egyszerű dologról van szó. Sőt, a fogalom nem is új, az előző szakaszban már találkoztál is vele, csak nem vetted észre. Amikor ugyanis egy parancsnak megadod a paraméterét, akkor a szöveg egy részét egységként kezeled, ezt az egységet adod át a parancsnak. A szöveg egy részének ilyen elkülönítését csoportosításnak nevezik.

Csoportosításra a LaTeX-ben két okból van szükség. Az egyiket már láttad, parancsok paraméterének megadásáról van szó. A másik ok, hogy egyes parancsok egy csoporton belül használva csak a csoporton belüli szövegre vonatkoznak. Ilyenek például a betűméretet megváltoztató parancsok. A következő példában a `\large` parancs nagyobb betűtípust állít be.

Magas költségek • Sajnos az elmúlt évben

```
{\large Magas
költségek}
\quad$\bullet$\quad
Sajnos az elmúlt évben
```

Egy csoport kezdetét a `{` karakter, végét a `}` karakter jelenti, mint az az előbbi példában is látható. Lehetőség van csoportok egymásba ágyazására is, azaz egy csoporton belül újabb csoportot is megadhatsz. A csoportok tartalmazhatják egymást, de nem lehetséges átfedés közöttük. A `}` jel mindig a legutolsó `{` jelet zárja le.

2.2.5. Megjegyzések

A kéziratfájlban saját magunknak vagy másoknak szóló megjegyzéseket is elhelyezhetünk, amelyeket a LaTeX figyelmen kívül hagy, sem parancsként, sem szövegként nem értelmez. Megjegyzések elhelyezése a `%` jellel lehetséges, ez után

a jel után a LaTeX mindent figyelmen kívül hagy egészen az első újsor karakterig, azaz a sor végéig.

Ez egy példa

```
Ez egy % elég hülye
% inkább elég jó
példa
```

2.2.6. A kéziratfájl szerkezete

A LaTeX kéziratfájl írásakor be kell tartanod néhány szabályt. A dokumentumnak a

`\documentclass{...}`

paranccsal kell kezdődnie, ez a parancs határozza meg milyen típusú (osztályú) dokumentumot szeretnél írni. Ezt követheti a

`\usepackage{...}`

parancs. Már említettem, hogy a LaTeX-hez nagyon sok kiegészítés készült. Ezeket a kiegészítéseket LaTeX csomagoknak hívják. A fenti paranccsal megadhatod a LaTeX-nek, hogy milyen kiegészítéseket, csomagokat szeretnél használni. A következő parancs a dokumentum kezdetét jelenti:

`\begin{document}`

Elkezdheted begépelni a dokumentum szövegét a LaTeX-nek szóló utasításokkal tarkítva. Végül az utolsó parancs az

`\end{document}`

Ez a dokumentum szövegének a végét jelöli. Ami ez után a parancs után áll, azt a LaTeX figyelmen kívül hagyja.

2.3. A dokumentum típusa

2.3.1. Dokumentumosztályok

Amikor a LaTeX megtervezi dokumentumodat, szüksége van egy fontos információra még a tervezés előtt, ez pedig a dokumentum típusa. Világos, hogy egy szemináriumi dolgozatnak nem úgy kell kinéznie, mint egy könyvnek, meg kell adnod hát, hogy milyen típusú dokumentumot szeretnél. Ezt a

`\documentclass[`*kapcsolók*`]{`*osztály*`}`

paranccsal teheted meg, ez a parancs meg kell hogy előzzön minden más parancsot és szöveget a kéziratban. Az *osztály* paraméter határozza meg, milyen típusúra formázza a LaTeX a dokumentumot. A LaTeX 2_ε által definiált osztályokat a 2.1. táblázatban láthatod.

A felsoroltakon kívül még több dokumentumosztály létezik, sőt bárki készíthet új dokumentumosztályt, akár egy teljesen új forma létrehozásával, akár egy már létező megváltoztatásával.

`article`	Rövid tudományos cikkeknek, programok dokumentációjának, rövid dolgozatoknak vagy meghívóknak a formázására használhatod.
`report`	Rövid könyvek, több fejezetből álló dolgozatok, PHD dolgozatok, diplomamunkák írására készült.
`book`	Valódi könyveket hozhatsz létre a segítségével.
`letter`	Üzleti és magánlevelek formázása és nyomtatása a feladata.
`slide`	Fóliákat nyomtathatunk vele, mert nagy és jól látható betűket használ.

2.1. táblázat. LaTeX 2_ε dokumentumosztályok

Különböző *kapcsolók* megadásával egy dokumentumosztály formáját kisebb-nagyobb mértékben módosíthatod. A standard LaTeX 2_ε dokumentumosztályok a 2.2. táblázatban felsorolt kapcsolókat értik meg.

2.3.2. Csomagok

Előfordulhat, hogy dokumentumod írásakor akadályba ütközöl, valami olyan speciális dolgot szeretnél – EPS grafika beillesztése, szines szöveg nyomtatása, stb. – amit az alap LaTeX nem támogat. Ekkor tehetnek jó szogálatot a már sokszor emlegetett kiegészítő csomagok. Ha egy dokumentumhoz csomagokat szeretnél használni, akkor ezt a kézirat elején, a dokumentum osztályának megadása után jelezned kell a

`\usepackage[`*kapcsolók*`]{`*csomag*`}`

paranccsal. Láthatod, hogy az egyes csomagoknak is adhatsz meg kapcsolókat, különböző csomagok más és más kapcsolókkal rendelkezhetnek. A LaTeX a `\documentclass` parancsnál megadott kapcsolókat automatikusan továbbítja a `\usepackage` paranccsal betöltött csomagoknak. Ez akkor hasznos, ha bizonyos kapcsolókat több csomagnak is át akarsz adni. Ilyenkor nem kell a kapcsolót minden csomagnak átadni, elég a `\documentclass` parancsnak, a többiről ő gondoskodik.

A 2.3. táblázatban láthatsz néhány hasznos LaTeX csomagot, és a későbbiekben is megemlítünk még néhányat. Természetesen a felhasználó is hozhat létre csomagokat, ennek fortélyaira könyvemben annak bevezető jellege miatt nem térek ki.

2.3.3. Oldalstílusok

Oldalstíluson a LaTeX azt érti, hogy mi kerül a fejlécbe és a láblécbe. A LaTeX háromféle oldalstílust definiál előre. Ezeket a 2.4. táblázatban olvashatod. Egy adott stílus beállítását a

`\pagestyle{`*stílus*`}`

`10pt, 11pt, 12pt` A dokumentum alapértelmezett betűméretét változtathatod meg. Az alapértelmezett méret 10pt.

`a4paper, letterpaper, ...` A papírméretet állíthatod be vele. Lehetséges méretek a fenti kettőn kívül: `a5paper`, `b5paper`, `executivepaper`, `legalpaper`. Magyarországon általában az `a4paper` használatos, az alapértelmezés az amerikai `letterpaper` lapméret.

`fleqn` A kiemelt egyenleteket nem középre helyezve, hanem balra igazítva jeleníti meg.

`leqno` Az egyenletszámozás az egyenletek bal oldalára kerül a jobb oldal helyett.

`titlepage, notitlepage` Előbbi megadásakor a LaTeX 2ε készít címoldalt, utóbbi megadásakor nem. A címoldal az `article` osztály esetében nem jelent külön oldalt, csak a cím és a szerző felírását az első oldalra, a `report` és `book` osztályoknál a címoldal külön lap.

`twocolumn` A teljes dokumentum kéthasábos lesz.

`twoside, oneside` Az első akkor kell, ha kétoldalas dokumentumot nyomtatunk, a második, ha egyoldalast. Kétoldalas a dokumentum, ha a papír mindkét oldalán van szöveg, egyoldalas, ha csak az egyiken. Alapból csak a `book` osztály kétoldalas, a többi egyoldalas.

`openright, openany` Az első megadásakor új fejezet csak jobb oldali lapon kezdődhet, a második esetén bal- és jobboldalon egyaránt. Ez az `article` osztály esetén nem működik, ebben ugyanis nincsenek fejezetek. A `report` osztályú dokumentumoknál az `openany`, a `book` osztályúaknál az `openright` érvényes.

2.2. táblázat. A LaTeX 2ε által elfogadott kapcsolók

paranccsal érhetjük el. Ez a beállítás az aktuális oldaltól (azt is beleértve) lesz érvényben. A LaTeX alapértelmezésben a dokumentumosztálynak megfelelő stílust választ. Csak az aktuális oldal stílusának átállítása a

`\thispagestyle{`*stílus*`}`

parancs segítségével történhet. Természetesen lehetséges saját stílus használata is, erre most nem térnék ki részleteiben, de ajánlom figyelmedbe a `fancyhdr` csomagot. Ennek segítségével könnyedén helyezhetsz el akár díszes grafikákat is a fejlécben illetve a láblécben.

2.3.4. Hosszú dokumentumok

Ha egy nagyon hosszú munkán, mondjuk egy könyvön dolgozol, akkor jól jöhet, hogy a LaTeX kéziratot több fájlba is széttördelheted. Három ezzel kapcsolatos parancs létezik. Segítségükkel a dokumentumnak mindig csak azt a részét kell megformáznod, amin épp dolgozol. Az első az

`array`	Speciális táblázatok készítésekor veheted hasznát.
`doc`	A csomag a LaTeX, és a LaTeX csomagok dokumentációjának elkészítésében segít.
`index`	Segítségével tárgy-, név-, és egyéb mutatókat készíthetünk.
`multicol`	Többoszlopos dokumentumok készítéséhez nagyon hasznos segítség.
`verbatim`	Programok dokumentációjának írásához, forráskódot is írhatsz vele.
`color`	Színes dokumentumokat nyomtathatsz vele.
`fancyhdr`	Speciális fejlécek, és láblécek létrehozására szolgál.
`graphics`	Grafikák, képek beillesztése, nagyítása, transzformálása.
`babel`	Nem angol nyelvű írásokhoz nélkülözhetetlen segédeszköz. Bővebben lásd a 6. fejezetben.

2.3. táblázat. Néhány LaTeX csomag

`empty`	A legegyszerűbb stílus, mind a fejléc, mind a lábléc teljesen üres lesz.
`plain`	Egyszerű stílus. A fejléc üres, a láblécben pedig csak az oldalszám van, középen.
`headings`	A fejlécben található az oldalszám a lap szélén és a fejezet címe, a lábléc üres.

2.4. táblázat. Az előre definiált LaTeX oldalstílusok

`\include{`*fájlnév*`}`

parancs, ezt a dokumentum törzsében használhatod. A megadott fájl tartalmát beszúrja az utasítás helyére, de előtte egy új oldalt kezd. Az

`\includeonly{`*fájlnév, fájlnév, ...*`}`

parancsot a bevezető részben használhatod, ennek hatására csak az itt megadott fájlok lesznek majd a dokumentum törzsében `\include` paranccsal beolvashatók.

Néha hasznos az, hogy a beolvasott fájl szerkesztése új oldalon kezdődik, – pl. mert így nem változnak meg az oldaltörések – néha pedig nem. Ha szeretnénk kikerülni ezt a viselkedést, akkor használjuk az

`\input{`*fájlnév*`}`

parancsot. Ez egyszerűen fogja és beolvassa a fájlt mindenféle ellenőrzés és oldaltörés nélkül, így az `\includeonly` parancs sem vonatkozik rá.

3. fejezet

Szövegformázás

3.1. Sor- és oldaltörések

3.1.1. Sorkiegyenlítés

Láttuk már, hogy a LaTeX a sortörések helyét automatikusan állapítja meg. A szavak közötti helykihagyásokat úgy variálja, hogy a sorok pontosan egyenlő hosszúak legyenek. Ha ez másként nem lehetséges, akkor elválasztja a szavakat. A bekezdések között általában helyet hagy ki, és a bekezdés első sorát beljebb írja, de mindezek a dokumentumosztálytól függően nem biztos, hogy igazak. Esetenként szükség lehet sortörésre egy adott helyen, új bekezdés kezdése nélkül. Ekkor használhatjuk a

`\\`

parancsot. Ez a parancs sokszor tesz majd jó szolgálatot a későbbiekben is, mindig a sor végének a jelzésére szolgál majd, főleg táblázatok szedésekor használatos gyakran.

A parancs csillagos változata, az adott helyen sortörést generál, de megakadályozza az oldaltörést:

`\\*`

Új oldalt kezdhetünk a

`\newpage`

parancs beírásával.

`\linebreak[`n`]`, `\nolinebreak[`n`]`, `\pagebreak[`n`]`, `\nopagebreak[`n`]`

Ezek a parancsok rendre sortörés kérésére és tiltására, illetve oldaltörés kérésére és tiltására szolgálnak. Minden parancsnak megadhatunk egy opcionális n paramétert, ennek értéke 0 és 4 közé eshet, és a kérés fontosságát jelzi. Minél

magasabb a szám, a kérés annál fontosabb. A 4 alatti számokat a LaTeX esetenként figyelmen kívül hagyhatja, ha az eredmény nagyon rosszul nézne ki. Ha nem adunk meg paramétert, azt a LaTeX n=4-nek tekinti.

3.1.2. Overfull és underfull dobozok

Most a kedves olvasó bizonyára megijed, és kezdi sejteni, hogy mégiscsak neki volt igaza, és a LaTeX nemcsak misztikus dolognak látszik, hanem az is. Persze ez egyáltalán nincs így, a címben szereplő két kifejezést csak azért nem fordítottam magyarra, mert az olvasó is ebben a formában fogja látni a képernyőjén.

A LaTeX tényleg mindent megtesz azért, hogy egy bekezdést a lehető legjobban formázzon meg. Ez az esetek nagy többségében sikerül is, néha azonban nem tud úgy sort törni, hogy ez lehetséges legyen. A LaTeX a szavak közti helykihagyásokat úgy változtatja, hogy a sorok ugyanolyan hosszúak legyenek, de ezt csak egy bizonyos határon belül teszi, a betűk közötti helykihagyásokhoz pedig egyáltalán nem nyúl, ez az olvashatóság rovására menne. Ha a sorkiegyenlítéskor két szó között túl kicsi vagy túl nagy hely keletkezne, akkor hibajelzést ad. A hiba a képernyőre is kiíródik, és a naplófájlba is bekerül. Kétféle ilyen jellegű hibát különböztetünk meg, az első amikor a szavak közötti helyet túl kicsire kellene összenyomni, így a sor vége jobb oldalt „kilóg". Ezt nevezzük *Overfull hbox*-nak. A második esetben a szóközöket túl nagyra kellene nyújtani, annyira, hogy az már nem nézne ki jól. Ekkor a LaTeX elvégzi a nyújtást, de hibajelzést ad, ez az *Underfull hbox*. Az ilyen hibák tipikusak akkor, ha túl keskeny az a sáv, ahova a szöveget be kellene tördelni és/vagy egy hosszú szót a LaTeX nem tud elválasztani. A hibát úgy oldhatod meg, hogy megadod a LaTeX-nek, hogy hol lehet a hosszú szót elválasztani (ezzel kapcsolatban lásd a következő szakaszt), vagy azt mondod neki, hogy ne törödjön azzal, hogy a szöveg rosszul fog kinézni. Utóbbit a

```
\sloppy
```

paranccsal teheted meg. Ha ezt a parancsot kiadtad, akkor a LaTeX a szavak közötti helyeket akármekkorára megnyújtja. Ha már nem kívánatos ez működés, akkor a

```
\fussy
```

paranccsal kapcsolhatunk vissza az eredeti, igényes formázásra.

3.1.3. Elválasztás

Ha a LaTeX egy bekezdést másképpen nem tud megformázni, akkor kénytelen elválasztani a szavakat. A program alaphelyzetben a szavakat az angol helyesírás szabályai szerint választja el, mivel angol nyelvterületen írták. Persze, mint korábban már említettük, szinte minden európai nyelv elválasztási szabályait képes használni, a megfelelő kiegészítésekkel. Ezzel kapcsolatban lásd a LaTeX magyarosításáról szóló részt a 6. fejezetben!

Előfordul, hogy a beépített szóelválasztó program egyes szavakat rosszul választ el, az algoritmus nem ismerheti az összes kivételt. Erre a problémára ad

megoldást a \hyphenation parancsszó és a \- parancskarakter.

\hyphenation{*szólista*}

A parancsot a dokumentum bevezető részében (a \begin{document} parancs előtt) használhatod. Akkor hasznos, ha a LaTeX egy szót következetesen rosszul választ el, vagy egy szót mindig meg szeretnénk óvni az elválasztástól. A parancs argumentuma a szavak listája, azokat a helyeket, ahol a szavakat el szabad választani kötőjellel (- karakterrel) jelölheted ki. A LaTeX a szavakat a dokumentumban csak a megjelölt helyeken fogja elválasztani, máshol soha. A kis- és nagybetűk egyenértékűek. Lássunk egy példát! A

```
\hyphenation{rend-őr FORTRAN}
```

parancs hatására a LaTeX a rendőr szót csak a megadott helyen választja el, a FORTRAN, fortran, Fortran szavakat pedig sehol.

\-

A parancs segítségével egyszeri elválasztási helyet adhatsz meg. Az adott szót a LaTeX csak a \- paranccsal megjelölt helyeken fogja elválasztani. A parancs különösen hasznos speciális karaktereket tartalmazó szavak esetében, ezeket a LaTeX automatikusan nem választja el.

Itt van például a megszentségtelenítheteteltenségetekért szó.

```
Itt van például a
meg\-szent\-ség\-te\-le\-%
n\'\i t\-he\-tet\-len\-sé\-%
ge\-te\-kért szó.
```

Figyeld meg a % karaktereket a sorok végén. Ennek segítségével érheted el, hogy az új sort a LaTeX ne tekintse új szónak.

Ha több szót szeretnél együtt, egy sorban tartani, megóvni az elválasztástól, akkor használd az

\mbox{*szöveg*}

parancsot. A paraméterként megadott *szöveg*-et a LaTeX minden körülmények között együtt tartja.

A telefonszámom megváltozott. Az új szám: 06 93 555-555.

```
A telefonszámom megváltozott.
Az új szám:
\mbox{06 93 555-555}.
```

Sortörés elkerülésére szolgál a '~' különleges karakter is. Ez egy eltörhetetlen szóközt generál, nagyon gyakran használhatjuk.

Mikor uralkodott nagy királyunk, IV. Béla?
Mikor uralkodott nagy királyunk, IV. Béla?

```
\raggedright
Mikor uralkodott nagy királyunk,
IV. Béla?

Mikor uralkodott nagy királyunk,
IV.~Béla?
```

3.2. Jelek és különleges karakterek

3.2.1. Idézőjelek

Ha idézőjeleket szeretnél írni, semmiképp ne használd az írógépi idézőjelnek megfelelő " jelet, mert nagyon rossz írásképet eredményez. Amit kapsz valójában nem is idézőjel lesz, hanem az inch angolszász mértékegység jele.

Az idézőjelek formája az adott nyelvtől függően más és más lehet. Angol nyelvű szövegekben a `` és a '' jeleket használd nyitó, illetve kezdő idézőjelként. Ha egyszeres idézőjelekre van szükséged, értelemszerűen a ` és a ' jeleket használhatod. Magyarországon a nyitó idézőjelek nem fent, hanem lent helyezkednek el, az alsó idézőjeleket a ,, (egymás után két vessző) karakterekkel írhatod be. A magyar alsó idézőjelekhez szükséges a `t1enc` csomag betöltése, írj egy `\usepackage{t1enc}` parancsot a dokumentum bevezető részébe. Ha magyar nyelvű szövegeket írsz – s ez elég valószínű – olvasd el a 6. fejezetet!

„Legyen szives lenyomni az 'X' gombot!”

```
,,Legyen szives lenyomni az 'X'
gombot!''
```

3.2.2. Kötőjelek, gondolatjel

Ha írógépen gépelsz, ugyanazt a jelet használod kötőjel, mínusz jel, gondolatjel gyanánt. Egy igényesen megszerkesztett dokumentumban ezek a jelek mind különbözőek. Kötőjeles szavak írásához használd a '-' karaktert, a rövid kötőjelet! Ha két kötőjelet írsz egymás mellé a kéziratba, akkor hosszabb kötőjelet kapsz eredményül, ezt használhatod évszámoknál.

A gondolatjelet illetően ismét eltérés van az angolszász és a magyar forma között. Az eltérés abból adódik, hogy angol szövegben a gondolatjelet nem veszik körül szóközök, a magyarban ennek ellenkezője igaz. Az angolok a '---' jelet (a kéziratban három kötőjel egymás után) használják gondolatjel írására. Magyar szövegekben ez a jel – a kvirtmínusz – nem használható. Ha gondolatjelként használnánk ugyanis, akkor a hosszú vonal és az azt körülvevő szóközök túl nagy lyukat eredményeznének a szövegben. Magyar szövegekben a '--' nagykötőjelet használd gondolatjel írására, természetesen szóközökkel előtte és utána. Ha mínusz jelre van szükséged akkor át kell kapcsolnod matematikai módba a $ jellel (A matematikai mód tárgyalására később kerül sor). Írd ezt: '$-$'!

A nagykötőjel használatának speciális esete, amikor szerzőpárt jelölsz. Ezzel is csak akkor van probléma, ha a keresztneveket is kiírod. Ilyenkor ugyanis, ha nem teszel szóközt a nagykötőjel két oldalára (keresztnevek nélkül így helyes),

akkor a nagykötőjel az egyik személy keresztnevét a másik vezetéknevéhez kapcsolja, az összetartozó nevek pedig látszólag szét lesznek választva, ha pedig teszel szóközt a nagykötőjel két oldalára, akkor abból gondolatjel lesz. A megoldás, hogy a közöket a nagykötőjel két oldalán csak egy kicsit növeled. Ezt megteheted a `\,` paranccsal, ami nagyon kis hely kihagyására szolgál.

oda-vissza, X-akták
Az 1848–1849-es forradalom.
Lenni – vagy nem lenni.
A megoldás tehát 3 vagy −5.
Brian W. Kernighan – Rob Pike

```
oda-vissza, X-akták\\
Az 1848--1849-es forradalom.\\
Lenni -- vagy nem lenni.\\
A megoldás tehát 3 vagy $-$5.\\
Brian W.~Kernighan\,--\,Rob Pike
```

3.2.3. . . . stb.

Az írógépen a pont vagy a vessző ugyanolyan széles, mint minden karakter. Ez a LaTeX dokumentumok esetében természeten nem így van, minden karakternek saját szélessége van. Ez gondot okoz akkor, amikor három pontot szeretnél írni egy felsorolás végére, olyat amilyen a szakasz címében van. Hiszen ha a kéziratban egyszerűen három pontot írsz egymás mellé, akkor azok nagyon közel kerülnek egymáshoz. A megoldás az

`\ldots`

parancs, ami a három pontot mindig az adott betűméretnek és betűtípusnak megfelelően szúrja be a szövegbe.

New York, Tokio, Budapest ...
New York, Tokio, Budapest . . .

```
New York, Tokio, Budapest ...\\
New York, Tokio, Budapest \ldots
```

3.2.4. Ligatúrák

Ha közelebbről megnézel egy igazi könyvet (például ezt), és megfigyeled az 'ff', 'fi', 'fl' és 'ffi' betűsorozatokat, láthatod, hogy ezek nem csak egyszerűen a két – illetve három – betű egymás mellé helyezéséből állnak, máshogy néznek ki. Ez azért van, mert ezek a betűk rosszul néznének ki egyszerűen egymás mellé helyezve, ezért a LaTeX egy új formájú betűvel helyettesíti őket. Ezt a program teljesen automatikusan teszi, nem kell odafigyelned rá. Előfordulhat azonban, hogy az eredeti betűket szeretnéd, helyettesítés nélkül. Ilyen eset például ha összetett szóról van szó, és az egyik betű még az előtaghoz, a másik már az utótaghoz tartozik. Ekkor az `\mbox` paranccsal lebeszélheted a LaTeX-et a helyettesítésről.

Kifinomult, Szaffi, de nem széffúró, hanem széffúró.

```
Kifinomult, Szaffi, de nem
széffúró, hanem
széf\mbox{}fúró.
```

ò	\‘o	ó	\’o	ô	\^o	õ	\~o
ō	\=o	ȯ	\.o	ö	\"o		
ǒ	\u o	ǒ	\v o	ő	\H o	o̧	\c o
ọ	\d o	o̲	\b o	o͡o	\t oo		
œ	\oe	Œ	\OE	æ	\ae	Æ	\AE
å	\aa	Å	\AA	ß	\ss		
ø	\o	Ø	\O	ł	\l	Ł	\L
ı	\i	ȷ	\j	!‘	!‘	?‘	?‘

3.1. táblázat. Ékezetes betűk és jelek

3.2.5. Ékezetek

Már volt szó róla, hogy a LaTeX sok nyelvet támogat. Ennek megfelelően nagyon sokféle ékezetet használhatsz LaTeX dokumentumaidban. Ezeket többnyire parancskarakterek segítségével érheted el. A `\’` parancskarakter például azt jelenti, hogy „tegyél egy vesszőt a következő karakterre!”, a `\"` pedig, hogy „tegyél két pontot a következő karakterre!”.

Egy kis probléma van néha az ’i’ és ’j’ betűkre helyezendő ékezetekkel. Ekkor ugyanis az ’i’ és ’j’ pontjára nincs szükség. Ezért a betűkészletek tartalmaznak egy speciális ’i’ és egy speciális ’j’ karaktert, pontok nélkül. Ezeket a `\i`, (’ı’) illetve a `\j` (’ȷ’) parancsokkal érheted el. Ekkor azonban vigyáznod kell, mert ezek nem parancskarakterek, hanem parancsszavak, így egy szóközt kell hagynod utánuk, ha a szó belsejében fordulnak elő és egy „figyelmen kívül nem hagyható szóközt”, ha a szó végén! Az összes ékezetet és különleges jelet generáló parancsot az 3.1. táblázatban láthatod. Néhány ezek közül megintcsak parancsszó, itt is ügyelj az előbb elmondottakra!

Persze ha olyan szöveget gépelsz, amiben nagyon sok az ékezet, és billentyűzeted ismeri az ékezetes betűket, akkor megoldható, hogy a kéziratba közvetlenül az ékezetes betűket írd a fenti parancsok helyett. Erről bővebben a LaTeX magyarosításáról szóló 6. fejezetben olvashatsz.

3.3. Üres hely a szavak között

Annak érdekében, hogy a sorok ugyanolyan hosszúak legyenek, a LaTeX változtatja, megnyújtja és összenyomja a szavak közötti helykihagyásokat. Arra is figyel, hogy a mondatok között egy kicsit több helyet hagyjon ki. A mondat végét pont, felkiáltójel, vagy kérdőjel jelzi. Ha a pont nagybetű után áll, azt nem tekinti mondat végének, ekkor ugyanis nagyon gyakran rövidítésről van szó.

Természetesen van lehetőséged ezen szabályok megváltoztatására. A már tárgyalt figyelmen kívül nem hagyható szóközt (’\␣’) nem lehet megnyújtani (és persze összenyomni). A ’~’ karakter olyan szóközt generál, amit nem lehet megnyújtani, és nem lehet „eltörni” (megakadályozza a sortörést). Ha a

`\@`

parancs egy pont után áll, az azt jelenti, hogy a mondatnak itt akkor is vége van, ha a pont előtt nagybetű állt.

Mr. Smith nagyon örült, hogy látta.
Az én kedvencem a BASIC. És a tied?

```
Mr.~Smith nagyon örült, hogy
látta.

Az én kedvencem a BASIC.\@ És a
tied?
```

A mondatok utáni plusz helykihagyás teljesen ki is kapcsolható a

`\franchspacing`

paranccsal. Ekkor a `\@` parancsra nincs is szükség. A visszakapcsolást a

`\nofrenchspacing`

paranccsal végezhetjük el.

A kedves olvasó ne ijedjen meg ettől a sok érthetetlen nevű parancstól, egyáltalán nem olyan nehéz őket megjegyezni, sőt rövid idő után egészen természetes lesz használatuk. Főleg igaz ez a `~` karakter használatára, ami talán a legfontosabb.

3.4. Címsorok, fejezetek és szakaszok

A könyveket általában részekre, fejezetekre és szakaszokra osztják, hogy az olvasó könnyebben eligazodjon egy terjedelmes mű olvasásakor. Természetesen ezt a LaTeX is támogatja a következő parancsokkal:

`\part`	– rész	`\chapter`	– fejezet
`\section`	– szakasz	`\paragraph`	– bekezdés
`\subsection`	– alszakasz	`\subparagraph`	– albekezdés
`\subsubsection`	– alalszakasz	`\appendix`	– függelék

Ezek közül a `\chapter` az `article` dokumentumosztály esetében nem használható, csak `report` vagy `book` dokumentumosztály mellett. Így nagyon könnyen illeszthetsz `article` dokumentumokat `book` dokumentumokba, minden egyes `article` dokumentum egy fejezet.

Mindegyik parancsnak – kivéve az `\appendix`-et – pontosan egy kötelező paramétere van, ez adott fejezet, szakasz, ... címe. Minden egyébről – a számozásról, kiemelésről, helykihagyásról – a LaTeX gondoskodik. Az így megadott címek kerülnek majd a tartalomjegyzékbe, feltéve ha szeretnél tartalomjegyzéket.

Ha igen akkor a

`\tableofcontents`

paranccsal készíthetsz, a program oda fogja beszúrni a tartalomjegyzéket, ahova ezt a parancsot írod. Nézzük meg a tartalomjegyzék készítésének műveletét egy kicsit részletesebben! Természetesen a formázás során a LaTeX nem olvassa be a teljes kéziratot, sőt a már megformázott oldalakat elfelejti. Ezért a tartalomjegyzék (a tágymutatóra és a kereszthivatkozásokra ugyanez igaz) készítését úgy oldották meg, hogy amikor a LaTeX a kéziratfájlban egy címsort talál, aminek szerepelnie kell a tartalomjegyzékben, azt formázatlanul kiírja egy adott fájlba. (Természetesen minden címsort ugyanabba a fájlba ír ki, ennek kiterjesztése `.toc`.) Ezzel látszatra nem sokra mész, hiszen a tartalomjegyzék még mindig nincs a dokumentumban. De ha a LaTeX-et újból végigfuttatod a kéziraton, akkor az észreveszi az előbb létrehozott fájlt, és ha `\tableofcontents` parancsot talál, akkor automatikusan beolvassa és megformázza a tartalmát. Ennek a következménye az, hogy ha dokumentumod tartalomjegyzéket tartalmaz, akkor a dokumentum formázását kétszer kell elvégezni, kétszer kell lefuttatni a LaTeX-et. Az első futás létrehozza a tartalomjegyzékfájlt, a második pedig megformázza azt a kívánt helyen.

Az előbb felsorolt fejezetkezdő parancsoknak van ún. „csillagos” változata is, a parancs neve után (még a paraméter elé) írj egy csillagot! Ekkor a címsor nem kerül be a tartalomjegyzékbe, nem lesz számozott sem, a LaTeX csak a megfelelő formázásokat végzi el.

Az eseményalgebra

```
\section*{Az esem\'enyalgebra}
Meggondolásaink alapján
teljesen világos, hogy \ldots
```

Meggondolásaink alapján teljesen világos, hogy ...

A fejezetek, szakaszok címei ugyanúgy jelennek meg a tartalomjegyzékben, mint a szövegben. Ez nem előnyös, akkor ha egy fejezetnek nagyon hosszú címe van, és ez rosszul mutat a tartalomjegyzékben. Természetesen erre is van megoldás. Ha `\chapter`, `\section`, ... parancs után közvetlenül megadsz egy opcionális paramétert szögletes zárójelekben, akkor ez a paraméter kerül a tartalomjegyzékbe. Például így:

```
\section[Rövid cím]{Ennek a fejezetnek nagyon hosszú címe van}
```

Jobb azonban, ha tartalomjegyzékbe és a szövegbe ugyanazok a fejezetcímek kerülnek, az opcionális paraméter használata nem javasolt.

Dokumentumodnak egyszerűen készíthetsz címoldalt, a

`\maketitle`

parancs végzi el ezt a feladatot. Még a parancs kiadása előtt meg kell adnod a dokumentum címét, a szerző nevét, és ha akarod, akkor a dátumot rendre az

`\title{`*cím*`}`, `\author{`*szerző*`}` és `\date{`*dátum*`}`

parancsokkal. Ha a dátumot nem adod meg, akkor a formázás dátumát szúrja be a LaTeX. Több szerzőt is megadhatsz, a neveket az

`\and`

paranccsal válaszd el! Ha a dokumentum címében egy adott helyen szeretnél sortörést, akkor oda írj egy \\ parancsot!

Az `article` dokumentumosztály esetében a „címoldal” nem jelent külön oldalt, hanem csak a cím és a szerző nevének felírását az első oldalra.

A fentebb felsoroltakon kívül van még három parancs, ami a dokumentum részekre osztását segíti, és csak a `book` dokumentumosztállyal használható. Ezek:

`\frontmatter`, `\mainmatter` és `\backmatter`

Ezek a parancsok a fejezetek számozását, a fejléceket és az oldalszámozást változtatják meg, aszerint, hogy a könyv melyik részét írjuk éppen. Közülük az első, a `\frontmatter` parancs a könyv elejét jelzi, az előszót, tartalomjegyzéket, a `\mainmatter` a könyv törzsének kezdetét jelzi, a `\backmatter` parancs pedig a lezáró részek, a tárgymutatónak és az irodalomjegyzéknek a kezdetét. Érdemes használni ezeket a parancsokat, sok apró formázási finomságot végeznek el. A `\frontmatter` utáni részben például a program nem számozza meg a címsorokat, ez általában így szokás, az előszó nem számozott.

3.5. Hivatkozások

Egy tudományos könyv írásakor gyakran előfordul, hogy egy oldalra, fejezetre, képletre vagy táblázatra kívánsz utalni, hivatkozni. A LaTeX-ben különösen nagy szerepe van a hivatkozásoknak, mivel a kézirat írásakor nem tudod megmondani, hogy az úsztatott objektumok hova kerülnek majd. Erről bővebben a 3.9. szakaszban olvashatsz. Egyszerűen készíthetsz hivatkozásokat, nem kell oldalszámokat, táblázatszámokat megjegyezned. Ha egy helyre hivatkozni kívánsz, akkor azt a helyet meg kell jelölnöd, oda egy címkét kell írnod. Az adott hely lehet egy képlet, egy ábra, egy táblázat, egy számozott felsorozás egy pontja, és természetesen egy fejezet, egy szakasz. A címkét a

`\label{`*címke neve*`}`

paranccsal helyezheted el. A címke helyére – azaz az ábra, táblázat, képlet számára – illetve a címke oldalszámára a

`\ref{`*címke neve*`}`, `\pageref{`*címke neve*`}`

parancsokkal hivatkozhatsz. Természetesen egy címkére akárhány hivatkozás történhet. A tartalomjegyzék tárgyalásánál már említett okok miatt, ha hivatkozásokat használsz, akkor a dokumetumot kétszer kell megformáznod. (Ez persze nem azt jelenti, hogy ha dokumentumodban tartalomjegyzéket is, és hivatkozásokat is szeretnél, akkor a dokumentumot már összesen négyszer kell megformáznod `:)` .) A LaTeX figyelmeztet, ha olyan címkére hivatkozol, ami nem létezik, megadja a hivatkozás sorának számát is a kéziratban, sőt a formázás végén még egy figyelmeztetést ad:

```
LaTeX Warning: There were undefined references.
```

Vagyis a kéziratban történtek hivatkozások definiálatlan címkékre. A definiálatlan hivatkozások helyére a dokumentumba a két kérdőjel (??) kerül.

Most épp a 30. oldalon tartasz az olvasásban és a 3.5. szakaszt olvasod.

```
Most \label{itt} épp a
\pageref{itt}.~oldalon tartasz az
olvasásban és a \ref{itt}.~szakaszt
olvasod.
```

Ha a `\label` parancsot egy táblázaton, képleten, ábrán belül adod meg, akkor a `\ref` parancs az adott táblázat, képlet, ábra számát szúrja be a szövegbe.

Legyen $a = (a_n)$ egy valós vagy komplex számsorozat. Az

$$a_1 + a_2 + a_3 + \ldots = \sum_{n=1}^{\infty} a_n \qquad (3.1)$$

szimbólumot végtelen számsornak nevezzük.
Azt mondjuk, hogy (3.1) konvergens, ha részletösszegeinek sorozata konvergens.

```
Legyen $a=(a_n)$ egy val\'os vagy
komplex sz\'amsorozat. Az
\begin{equation}\label{eq:sor}
a_1+a_2+a_3+\ldots=
  \sum_{n=1}^\infty a_n
\end{equation}
szimb\'olumot v\'egtelen
sz\'amsornak nevezz\"uk.

Azt mondjuk, hogy
(\ref{eq:sor}) konvergens,
ha r\'eszlet\"osszegeinek
sorozata konvergens.
```

3.6. Lábjegyzetek

Lábjegyzeteket a következő paranccsal hozhatsz létre:

`\footnote{`*lábjegyzet szövege*`}`

Természetesen a lábjegyzetek számozásáról és elhelyezéséről a LaTeX gondoskodik.

Knuth szerint a lábjegyzetek[a] áttekinthetlenné teszik a könyvet.

[a] Az olyanok, mint ez.

```
Knuth szerint a
lábjegyzetek\footnote{Az olyanok,
mint ez.} áttekinthetlenné teszik a
könyvet.
```

3.7. Kiemelt szavak

Az írógépen készült kéziratban a kiemelt szavakat általában aláhúzzák. Nyomtatott könyvekben a kiemelt szavakat dőlt vagy vastag betűvel szedik. Én a magam részéről a dőlt betűs kiemeléseket kedvelem, mert nem fárasztják úgy a szemet, mint a vastag betűs kiemelések, nem keverednek össze a címsorokkal, és elég szembetűnőek. Kiemelt szavak írására az

`\emph{`*szöveg*`}`

parancs szolgál. Ennek hatására a LaTeX a *szöveg*-et dőlt betűvel szedi. A kiemeléseken belüli kimemelések is működnek.

Ha kiemelésen belül újabbb kiemelést használsz, a duplán kiemelt szöveget a LaTeX álló betűkkel szedi.

```
\emph{Ha kiemelésen belül
újabbb kiemelést használsz,
a duplán kiemelt szöveget a
\LaTeX{} \emph{álló}
bet\H{u}kkel szedi.}
```

3.8. Környezetek

Esetenként szükség lehet arra, hogy a szövegnek egy részét a többi szövegtől elkülönítve, speciálisan kezeljük. Ilyen eset például, ha egy vers néhány versszakát akarod idézni, vagy egy program forráskódját szeretnéd leírni. Az ilyen elkülönített szövegrészeket a LaTeX terminológia környezeteknek nevezi. Sokféle típusú környezet létezik, aszerint, hogy az elkülönített szöveget hogyan kell megformázni. Az egyes típusú környezetekre nevükkel hivatkozhatsz. Környezetet a

`\begin{`*környezet neve*`}` szöveg `\end{`*környezet neve*`}`

parancsokkal hozhatsz létre. Egy környezeten belül egy másik környezet használható, a környezetek egymásba ágyazhatók. De mindig figyelj arra, hogy a `\begin` paranccsal megnyitott környezetet zárd is le, mert a LaTeX nagyon összekeveredhet. Egymásba ágyazott környezeteknél vigyázz, hogy először mindig a legutoljára megnyitott környezetet zárd le!

A következő oldalakon sorra tárgyaljuk a LaTeX környezeteit.

3.8.1. Listák, felsorolások: az `itemize`, `enumerate` és `description` környezetek

A három említett környezetet együtt tárgyaljuk, mert mindhárom különféle listák, felsorolások formázását segíti.

Az `itemize` környezet ún. pontozott listát, az `enumerate` pedig számozott listát hoz létre. Mindkét környezet esetében a lista egyes elemeinek a kezdetét az

`\item`

parancs jelzi. Mindkét környezetben megadhatsz egy opcionális paramétert szögletes zárójelben az `\item` parancs után. Ha ezt teszed, akkor az elem előtti sorszám, pont, kötőjel helyére a paraméterként megadott jel kerül. Az így meg-

adott elem a számozott listában nem számít bele a számozásba, amit a LaTeX természetesen automatikusan végez.

1. Közönséges elem.
   – Különleges elem.
2. Na és ez hányadik lesz?

```
\begin{enumerate}
\item Közönséges elem.
\item[--] Különleges elem.
\item Na és ez hányadik lesz?
\end{enumerate}
```

A `description` környezet is listák készítésére szolgál. A lista elemei előtt azonban nem lesz jel vagy szám, hanem az első (néhány) szó kiemelten lesz szedve. A kiemelt szavakat szögletes zárójelek között kell megadnod az `\item` parancs után. Ennek a környezetnek a segítségével készült ezen könyv több táblázata is.

1. A listákat
   - Tetszés szerint ágyazhatod egymásba.
   – Típusaikat keverheted.
2. De attól, hogy valami

   butaságot ilyen csodás listába szedsz, még nem lesz

   okosság belőle.

   Ezt jól jegyezd meg!

```
\begin{enumerate}
\item A listákat
\begin{itemize}
\item Tetszés szerint
ágyazhatod egymásba.
\item[--] Típusaikat keverheted.
\end{itemize}
\item De attól, hogy valami
\begin{description}
\item[butaságot] ilyen csodás
listába szedsz, még nem lesz
\item[okosság] bel\H{o}le.
\end{description}
Ezt jól jegyezd meg!
\end{enumerate}
```

3.8.2. Bekezdések igazítása: a `flushleft`, `flushright` és `center` környezetek

Néha szükség van rá, hogy egy bekezdés ne sorkiegyenlítve legyen szedve, hanem balra, vagy jobbra legyen igazítva, vagy sorai középre legyenek helyezve. Ezeket a feladatokat oldják meg a `flushleft`, `flushright` és `center` környezetek. A sorok végét természetesen Te is meghatározhatod a `\\` paranccsal, ha ezt nem teszed meg, akkor a LaTeX automatikusan állapítja meg a sortöréseket.

Vasárnap Ausztráliába
érkezett Tom Telez, a kilencszeres
olimpiai bajnok Carl Lewis edzője.

```
\begin{flushleft}
Vasárnap Ausztráliába\\ érkezett
Tom Telez, a kilencszeres olimpiai
bajnok Carl Lewis edz\H{o}je.
\end{flushleft}
```

Mint ismeretes, az amerikai trénert az ausztrál szövetség szerette volna megnyerni a szövetségi kapitányi posztra.

```
\begin{flushright}
Mint ismeretes, az amerikai\\
trénert az ausztrál szövetség
szerette volna megnyerni a
szövetségi kapitányi posztra.
\end{flushright}
```

Mont Blanc-kupa, Megève
Franciaország

```
\begin{center}
Mont Blanc-kupa, Meg\‘eve\\
Franciaország
\end{center}
```

Előfordulhat, hogy a sorkiegyenlített szedés nem ad jó eredményt, ennek oka leggyakrabban az, hogy a hasábok túl keskenyek, így nehéz megtalálni a sortörések helyét, túl sok üres helyet kell rakni a szavak közé. Ekkor legjobb balra zárt szöveget írni, hiszen elsősorban a szöveg olvashatósága számít, és a balra zárt szövet ilyenkor jobb képet is ad. Néhány magyar folyóiratban is találkozhatunk ezzel a módszerrel. Kéthasábos szöveget írhatsz a `twocolumn` kapcsoló megadásával a dokumentum elején, a `\documentclass` parancs után. Így az egész szöveg kéthasábos lesz, gyakran erre is van szükség. Ha csak a dokumentum egy részét szeretnéd többhasábosan írni, akkor a `multicol` csomagot kell használnod.

3.8.3. Idézetek és versek: a `quote`, `quotation` és `verse` környezetek

A `quote` környezet jól használható idézetek, mondások, példák írásához.

A következő tipográfiai szabály a sorok hosszára vonatkozik.

> Egy sor sem tartalmazhat 66 karakternél többet.
>
> Ezért olyan szélesek alapban a LaTeX margói.

Ezért nyomtatják az újságokat többhasábosan.

```
A következ\H{o} tipográfiai szabály
a sorok hosszára vonatkozik.
\begin{quote}
Egy sor sem tartalmazhat
66~karakternél többet.

Ezért olyan szélesek alapban
a \LaTeX{} margói.
\end{quote}
Ezért nyomtatják az
újságokat többhasábosan.
```

Két ehhez hasonló környezet is létezik. Hosszabb idézetekhez használható jól a `quotation`, mert ez az új bekezdéseket beljebb kezdi. A `verse` környezet

versek idézéséhez kitűnő, ügyel a sortörésekre, és az igazításokra. Az egyes verssorokat a \\ paranccsal zárd le, a verszakok között pedig hagyj ki egy sort!

Pipacsot éget a kövér
határra
A lángoló magyar nyár
tűzvarázsa.
A Tisza szinte forr, mint
néma katlan,
Mit izzó part ölelget
lankadatlan.

```
\begin{flushleft}
\begin{verse}
Pipacsot éget a kövér határra\\
A lángoló magyar nyár
t\H{u}zvarázsa.\\

A Tisza szinte forr,
mint néma katlan,\\
Mit izzó part ölelget
lankadatlan.\\
\end{verse}
\end{flushleft}
```

3.8.4. Nyomtatás szó szerint: a `verbatim` környezet

A `\begin{verbatim}` és `\end{verbatim}` parancsok közé zárt szöveget a LaTeX „szó szerint” formázza meg, úgy mintha azt egy írógépen írtad volna, ugyanazokkal a sortörésekkel, és szóközökkel. Sőt, a szövegben szereplő parancsokat sem hajtja végre, azok is a dokumentum részei lesznek. Ez főleg programok forráskódjának írásakor hasznos. A szerző is ezt a környezetet használta a példák kéziratbeli szövegének (a jobb oldalnak) a formázásához.

Ha csak néhány szót szeretnénk „szó szerint” formázni, akkor használhatjuk a

`\verb+`*szöveg*`+`

parancsot. A LaTeX a *szöveg*-et szó szerint fogja megformázni. Az itt szereplő + karakter csak egy példa, ehelyett bármilyen jelet használhatunk határolójelként, kivéve a betűket, a szóköz karaktert és * karaktert. Lássunk néhány példát.

Az `\ldots` parancs ...

```
Az \verb!\ldots! parancs \ldots
```

```
10 PRINT "Hello mindenkinek!"
20 GOTO 10
```

```
\begin{verbatim}
10 PRINT "Hello mindenkinek!"
20 GOTO 10
\end{verbatim}
```

A `verbatim` környezetnek is, és a `\verb` parancsnak is van csillagos változatata (a parancs, illetve a környezet neve után egy csillagot írunk). Ezek abban különböznek az eredetitől, hogy a szó szerint formázott szövegben szereplő szóközöket a LaTeX kiemeli, jól láthatóvá teszi.

A \␣ parancs ...

```
A \verb*+\ + parancs \ldots
```

```
while␣(also␣<=␣felso)␣{
␣␣␣␣kozep␣=␣(also␣+␣felso)␣/␣2;
␣␣␣␣if␣(x␣<␣v[kozep])
```

```
\begin{verbatim*}
while (also <= felso) {
   kozep = (also + felso) / 2;
   if (x < v[kozep])
\end{verbatim*}
```

3.8.5. Táblázatok: A `tabular` környezet

A `tabular` környezettel egyszerűen és gyorsan hozhatsz létre csodálatos táblázatokat. A táblázat megformázásához szükséges adatokat a környezet kezdetén kell megadnod.

`\begin{tabular}[`*függőleges pozíció*`]{`*oszlopok formája*`}`

A *függőleges pozíció* azt adja meg, hogy egy soron belül a szöveg hogyan legyen igazítva. Az alapértelmezés a középre igazítás, ezt bírálhatjuk felül. A '`t`' betű felülre, a '`b`' pedig alulra igazítást jelent. Az *oszlopok formája* paraméternek több szerepe is van, a legfontosabb, hogy itt adhatod meg az oszlopok számát. Ez a paraméter valójában több elem felsorolása egymás után, ezek az elemek a következők lehetnek:

- `l` - Egy balra igazított oszlopot jelöl. Az oszlop minden egyes eleme az oszlopon belül balra lesz igazítva.
- `r` - Egy jobbra igazított oszlopot jelöl.
- `c` - Egy középre igazított oszlopot jelöl.
- `|` - Függőleges vonal. A példákat megfigyelve azonnal megérted, hogyan is működik.
- `@{`*szöveg*`}` - A táblázat minden sorában az adott két oszlop közé beszúrja a *szöveg*-et. A `tabular` oszlopainak szélességét a LaTeX automatikusan állapítja meg. Az egyes oszlopok között persze helyet is hagy ki. A `@{...}` parancs ezt a helyet írja felül, az alapértelmezés szerinti helykihagyás helyett *szöveg*-et írja ide.
- `p{`*szélesség*`}` - Ezt az elemet akkor használhatod, ha a táblázat egy oszlopába hosszabb szöveget akarsz írni, és szeretnéd, ha a LaTeX állapítaná meg a sortöréseket. A paraméter az oszlop szélességét adja meg. Figyeld meg a példákat!
- `*{`*szám*`}{`*oszlopok*`}` - Ha táblázatod több azonos típusú oszlopból áll, akkor az *oszlopok formája* paraméter megadásakor ugyanazt a szöveget kellene sokszor leírnod, és ez esetenként nagyon hosszú is lehet. Ezen segít ez a parancs. Az *oszlopok* oszlopmegadást (ez természetesen az itt felsorolt elemeket tartalmazhatja, akár a `*` elemet is) *szám*-szor használja fel a LaTeX, azaz úgy értelmezi, mintha *oszlopok*-at *szám*-szor írtad volna le egymás után.

A táblázat egyes sorait a \\ paranccsal, a sorok egyes elemeit a '&' jellel válaszd el egymástól!

Pajzshasználat	Af
Lovaglás	Af
Kocsihajtás	Af

```
\begin{center}
\begin{tabular}{ll}
Pajzshaszn\'alat & Af\\
Lovagl\'as & Af\\
Kocsihajt\'as & Af\\
\end{tabular}
\end{center}
```

Vízszintes vonalat a

`\hline`

paranccsal húzhatsz, a vonal hossza egyenlő a táblázat teljes szélességével. A parancsot értelemszerűen két sor közé kell írnod. Ha nem a táblázat teljes szélességében akarsz vonalat húzni, csak bizonyos oszlopok között, akkor használd a

`\cline{`*i*-*j*`}`

parancsot! Ez csak az i. oszloptól a j. oszlopig húz vonalat.

7C0	hexadecimális
3700	oktális
11111000000	bináris
1984	decimális

```
\begin{tabular}{|r|l|}
\hline
7C0 & hexadecimális \\
3700 & oktális \\
11111000000 & bináris \\
\hline
\hline
1984 & decimális \\
\hline
\end{tabular}
```

Ha függőleges vonalat szeretnél, de nem a táblázat teljes magasságában, csak néhány sorban, akkor a kérdéses sorban, a két oszlop között használd a

`\vline`

parancsot!

Ha több oszlopot össze akarsz vonni egy oszlopba, akkor a

`\multicolumn{`*oszlopok*`}{`*pozíció*`}{`*szöveg*`}`

parancs a megoldás. A következő *oszlopok* számú oszlopot vonja össze egy oszlopba. Ez az új oszlop vízszintesen *pozíció* szerint lesz igazítva. Az 'l' balra, az

'r' jobbra, a 'c' pedig középre igazítást jelöl. A *szöveg* pedig természetesen az összevont oszlopokba írandó szöveget jelenti.

Képzettségek	
Futás	Af
Titkosajtókeresés	18%
Csapdafelfedezés	13%

```
\begin{center}
\begin{tabular}{ll}
\multicolumn{2}{c}{Képzettségek}\\
\hline
\\
Futás & Af \\
Titkosajtókeresés & 18\% \\
Csapdafelfedezés & 13\% \\
\hline
\end{tabular}
\end{center}
```

A LaTeX nem tudja a számokat helyiérték szerint igazítani alapból, de a `@{...}` parancs megoldja ezt a feladatot. Hozz létre egy jobbra igazított oszlopot az egészrésznek, egy balra igazított oszlopot a törtrésznek, a közöttük lévő helyet pedig cseréld ki a tizedesvesszőre! Természetesen ekkor tizedesvessző helyett `&` karaktert kell írnod.

Kifejezés	Értéke
π	3,1416
π^π	36,46
$(\pi^\pi)^\pi$	80662,7,

```
\begin{tabular}{@{}c|r@{,}l@{}}
\hline\vline
\ Kifejez^^e9s &
\multicolumn{2}{c}{\'Ert\'eke}
\vline \\
\hline
$\pi$ & 3&1416 \\
$\pi^\pi$ & 36&46 \\
$(\pi^\pi)^\pi$ & 80662,7 \\
\hline
\end{tabular}
```

A következő táblázat arra példa, hogyan írhatunk táblázatokba hosszabb szöveget, automatikus sortörésekkel.

Erő: 17	Az erő képesség azt határozza meg, hogy a karakter fizikálisan mennyire erős.
Á.k.: 12	A magasabb állóképességű karakter több sérülést bír ki, és ellenállóbb a fájdalommal szemben
Gy.: 13	Meghatározza, hogy a karakter milyen gyorsan reagál az őt ért hatásokra

Íme végül az előbbi táblázatot előállító forrásszöveg.

```
\begin{tabular}{|lr|p{3.5cm}|}
\hline
Erő: & 17 &
Az erő képesség azt határozza meg, hogy a karakter fizikálisan
```

```
mennyire erős.\\\hline
Á.k.: & 12 & A magasabb állóképességű karakter több sérülést
bír ki, és ellenállóbb a fájdalommal szemben\\\hline
Gy.: & 13 & Meghatározza, hogy a karakter milyen gyorsan
reagál az őt ért hatásokra\\
\hline
\end{tabular}
```

3.9. Úsztatott objektumok, ábrák, táblázatok

A manapság megjelenő könyvek – főleg a tudományosak – tele vannak ábrákkal és táblázatokkal. Ezekkel az a probléma, hogy egészben kell őket megformázni, nem lehet őket kettétörni két oldalra. De mi van akkor, ha a táblázat vagy ábra már nem fér ki az oldalra? Egyszerű. Hát új oldalt kell kezdeni. Ez viszont nagy táblázatok esetén azt jelentheti, hogy az oldal többi része üres marad, ezt pedig valószínűleg a szerző nem szeretné. A megoldás az, hogy az ábráknak és táblázatoknak nincs kötött helyük a dokumentumban, a LaTeX máshova helyezheti őket a legjobb külalak érdekében. Ez a módszer azt eredményezi, hogy ha sok ábrát és táblázatot használunk, akkor azok össze-vissza fognak úszkálni a dokumentumban. Alapban kétféle úsztatott objektum létezik, az egyik a táblázat, a másik az ábra.

Olvasd el figyelmesen a következőket, különben a LaTeX soha nem oda fogja tenni táblázatodat ahova szeretnéd.

A LaTeX terminológiában a táblázat és ábra mást jelent, mint általában a WYSIWYG szövegszerkesztőknél. Sőt, az előző fejezetben tárgyalt táblázatok sem azok a táblázatok, amelyekről most szó lesz. A LaTeX terminológiában a táblázat (az angol LaTeX terminológia csak ezt a táblázatot nevezi táblázatnak) csak egy logikai egységet jelent, amelyet a dokumentum többi részétől külön kell kezelni. Az, hogy ez a logikai egység valójában micsoda fizikailag az a LaTeX számára tökéletesen mindegy. Lehet táblázat egy szövegrész, egy `description` környezet (erre ebben a könyvben is sok példa van), vagy – ez a leggyakoribb eset – egy `tabular` környezet. Ugyanezek vonatkoznak természetesen az ábrákra is, ez sem szükségszerűen ábrák.

A logikai táblázat és ábra fogalmára csak az elhelyezés miatt van szükség, a fentebb már kifejtett okok miatt. Az úsztatott objektumok két csoportra osztásának már nincs nagy jelentősége, mint látni fogjuk, bár az praktikus, hogy az ábrák és táblázatok számozását a LaTeX egymástól függetlenül, automatikusan végzi. A következőkben csak a táblázatokról fogok írni, de az alábbiak ugyanúgy érvényesek az ábrákra is. Ha ez nem így lenne, akkor arra külön kitérek.

Táblázatot a `table` környezettel adhatsz meg. A `table` környezetbe zárt összes szöveg (vagy más egyéb) egy úszó objektumba kerül. Ábrát a `figure` környezettel adhatsz meg. A

`\caption{`*táblázat címe*`}`

paranccsal a táblázat címét adhatod meg. A „táblázat” szót (eredetileg persze angolul), és a táblázat sorszámát a LaTeX automatikusan beszúrja a megadott cím elé. A parancsnak megadhatsz egy opcionális paramétert is, ez akkor hasz-

`h`	Jelentése „here” azaz „itt”. A program oda próbálja tenni a táblázatot, ahol az a kéziratban van. Főleg kicsi táblázatoknál használható jól.
`t`	Az oldal tetején.
`b`	Az oldal alján.
`p`	A „tutajok oldalán”. Ez egy olyan oldal, ahol csak táblázatok és ábrák vannak, összegyűjtve.
`!`	„Akkor is, ha nem néz ki jól.” Ezt a karaktert a többivel együtt használhatod.

3.2. táblázat. A tutajok elhelyezését szabályozó karakterek

nos, ha hosszú táblázatneveket használsz. Ekkor az opcionális paraméter kerül a táblázatok jegyzékébe:

```
\caption[Rövid cím]{Nagyon-nagyon hosszú cím}
```

A táblázatok és az ábrák jegyzékét rendre a

`\listoftables` és `\listoffigures`

parancsokkal szúrhatod be a dokumentumba.

Ha magyar táblázatneveket és sorszámozást szeretnél, akkor olvasd el a 6. fejezetet.

A `table` környezeten belül elhelyezett `\label` paranccsal hivatkozhatsz a táblázatra. Vigyázz arra, hogy a `\label` parancsot mindig a `\caption` parancs után írd, különben a címkére hivatkozó `\ref` parancsok nem a helyes számot adják.

Nem esett még szó az *elhelyezés* opcionális paraméterről. Ez, mint a neve is mutatja, a táblázat elhelyezését adja meg, és a 3.2. táblázatban megadott karakterekből állhat. A LaTeX a megadott karaktereknek megfelelő helyekre próbálja elhelyezni a táblázatot. Az alapértelmezés a `[tbp]`. Az egyébként teljesen mindegy, hogy milyen sorrendben adod meg a betűket, a LaTeX a 3.2. táblázatban megadott sorrend szerint próbálja ki őket.

Ha sok tutajt használsz a dokumentumban, akkor előfordulhat, hogy a LaTeX nem oda teszi ezeket, ahova szeretnéd, vagy hogy a sok tutaj összetorlódik. Ebben az esetben megteheted, hogy a táblázatokat és ábrákat összegyűjtve nyomtatod ki a 'p' elhelyezést szabályozó karakterrel. Jól jöhet még a

`\clearpage`

parancs, ennek hatására a LaTeX az összes eddig ki nem nyomtatott tutajt kinyomtatja, majd egy új oldalt kezd. A

`\cleardoublepage`

parancs ettől annyiban különbözik, hogy új dupla-oldalt kezd, azaz az utána következő szöveget baloldalra teszi kétoldalas dokumentumok esetén.

3.10. Saját parancsok és környezetek

Azt már a könyv eleje óta tudod, hogy a LaTeX számára a dokumentum megformázához szükséges információkat LaTeX parancsokkal adhatod meg. Ez akkor válik csak kényelmetlenné, ha sokszor kell ugyanazokat a parancsokat beírnod egy speciális rész formázásához. Természetesen a LaTeX ezt a problémát is megoldja. A sokszor használt parancssorozatok végrehajtására egy új parancsot definiálhatsz, és a hosszú gépelés helyett elég az új parancsot beírnod. Sőt, saját környezetek definiálására is lehetőséged van, mint azt látni fogod.

Tulajdonképpen a LaTeX-hez készített bővítések is új parancsok és környezetek gyűjteményei. Így akár az olvasó is készíthet LaTeX kiegészítéseket, bár természetesen ez közel sem olyan egyszerű. Ahhoz, hogy saját makrócsomagokat írj, nagyon sok parancsot kell ismerned, sőt esetenként még a TeX belső működését is meg kell értened. De ha nem vállalkozol ilyen nagy munkára, akkor is hasznosnak fogod találni a következőket.

3.10.1. Parancsok

Új parancsot a

`\newcommand{`*parancs neve*`}[`*paraméterek száma*`]{`*definíció*`}`

paranccsal hozhatsz létre. Két kötelező paramétert kell megadnod, a parancs nevét és definícióját. A parancs nevének '\' karakterrel kell kezdődnie.

A LaTeX dokumentum ...

```
\newcommand{\ldok}
   {\LaTeX\ dokumentum}
A \ldok\ \ldots
```

Természetesen olyan parancsokat is megadhatsz, amelyek paraméterek feldolgozására is képesek. A paraméterek számát egy opcionális paraméterben adhatod meg. Ha ez a paraméter elmarad, az azt jelenti, hogy a definiálandó parancsnak nincsen paramétere. Az egyes paraméterekre a parancs definíciójában a `#`*szám* formában hivatkozhatsz, ahol *szám* a paraméter sorszáma. Íme egy példa:

- A *csodálatos* LaTeX
- A *fantasztikus* LaTeX

```
\newcommand{\LT}[1]
  {A \emph{#1} \LaTeX\ }
\begin{itemize}
\item \LT{csodálatos}
\item \LT{fantasztikus}
\end{itemize}
```

Ha a `\newcommand` parancsban megadott parancsnév már létezik, akkor a LaTeX hibajelzést ad, nem engedi felülírni azt. Ha egy már létező parancsnevet

szeretnél felülírni, akkor használd a `\renewcommand` parancsot, ennek szintaktikája a `\newcommand` paranccsal megegyezik, de megengedi a már létező parancsok felülírását. Esetleg szükséged lehet a `\providecommand` parancsra is, ez is ugyanúgy működik, mint a `\newcommand`, de ha a definiálni kívánt parancs már létezik, akkor a LaTeX az új denifíciót figyelmen kívül hagyja.

3.10.2. Környezetek

Természetesen új környezetek definiálása is lehetőséged van. Ez a parancsok definiálásához hasonló módon történik. Egy új környezet definiálásakor meg kell adnod a környezet nevét, és a környezet kezdetekor és végekor végrehajtandó parancsokat. A környezet kezdetekor végrehajtásra kerülő parancsoknak adhatunk át paramétereket is, a megszokott módon.

`\newenvironment{`*környezet*`}[`*paraméterek száma*`]{`*kezdetkor*`}{`*végén*`}`

A *kezdetkor* parancsokat a `\begin{`*környezet*`}` parancskor, a *végén* parancsokat pedig az `\end{`*környezet*`}` sornál hajtja végre a LaTeX.

Mit tehetünk veled, te kis tigrismacska?
Ha nem eszöl, sose leszöl nagyobbacska.

```
\newenvironment{vers}
 {\begin{flushleft}\begin{verse}}
 {\end{verse}\end{flushleft}}
\begin{vers}
Mit tehetünk veled, te kis
tigrismacska?\\
Ha nem eszöl, sose leszöl
nagyobbacska.\\
\end{vers}
```

A parancsokhoz hasonlóan, a környezetek újradefiniálására létezik egy parancs, a `\renewenvironment`, szintaktikája megegyezik a `\newenviromnent` parancséval. Ennek segítségével megváltoztathatjuk a LaTeX beépített környezeteit.

4. fejezet

Matematikai képletek írása

4.1. Matematikai mód

Végre elérkeztél hát a LaTeX fő erősségének bemutatásához. Hatalmas képleteket írhatsz majd gyönyörűen megszerkesztve egyszerűen, ha elolvasod ezt a fejezetet. Az itt leírtak a legtöbb képlet írásának módját tartalmazzák. Ám akkor se keseredj el, ha nem találod meg, amit szeretnél, problémádra ugyanis nagy valószínűséggel megoldást kínál az $\mathcal{AMS}$-LaTeX csomag és annak magyar nyelvű leírása, amit szintén jelen sorok írója követett el. A csomagot az Amerikai Matematikai Társaság adta ki, ennek megfelelően magas szinvonalú, ha matematikai témájú könyvet, cikket szeretnél írni, tapasztalni fogod előnyeit. Az $\mathcal{AMS}$-LaTeX a legtöbb működő LaTeX rendszerben megtalálható, kérdezd meg a helyi TeX szakembert a használatával kapcsolatban, és olvasd el az előbb említett leírást.

A LaTeX képletek írásakor speciális üzemmódban dolgozik. Ez kétféle lehet, a képlet típusától függően. Ha a képlet a szöveg közben bukkan fel, akkor *sorközi matematikai mód*ról beszélünk. Nagy képletek esetén ez az elhelyezés nem szerencsés, ezeket jobb „kivetítve", esetleg megszámozva, a szövegtől külön helyezni. Ilyen képletet *kiemelt matematikai mód*ban nyomtathatsz. A két üzemmód között csak annyi a különbség, hogy a LaTeX máshogy formázza meg a képletet. Az itt leírtak, ha erre külön nem utalunk, mindkét módra vonatkoznak.

Sorközi matematikai módba a

`\(` és `\)`, `\begin{math}` és `\end{math}` vagy a `$` és `$`

parancsokkal kapcsolhatsz át. Pontosabban az első parancs bekapcsolja a matematikai módot, ezt írd a képlet elé, a második pedig kikapcsolja azt, ezt persze a képlet után kell írni. A `$` jeleket csak nagyon kicsi, két-három betűs képleteknél célszerű használni. Ennek az az oka, hogy a `$` jel nem jelzi egyértelműen, hogy a képlet kezdetéről vagy végéről van szó, ha a LaTeXszövegmódban van, akkor a képlet elejét, ha matematikai módban van, akkor a képlet végét jelenti. Ha azonban lefelejtesz egy `$` jelet a képlet elejéről, vagy végéről, akkor a LaTeX pont azt írja matematikai módban, amit szövegmódban kellene, és ez az esetek többségében csak később, sok szöveg megformázása után derül ki. Nagyon nehéz ilyenkor megtalálni, hogy honnan maradhatott le a `$` jel. A `\(` és `\)` parancsok

egyértelműen jelzik a képlet elejét és végét, így könnyebb megtalálni, hogy hol volt a hiba.

Kiemelt matematikai módba a

`\[` és `\]`, vagy a `\begin{displaymath}` és `\end{displaymath}`

parancsokkal térhetsz át. Ha eddig csak TEX-et használtál LATEX nélkül, akkor a kiemelt képleteket `$$` közé kellett zárnod. Ezt a LATEX-ben ne használd, mert nem mindig ad jó eredményt!

Ha meg szeretnéd számozni a kiemelt egyenleteket, például, hogy hivatkozhass rájuk, akkor az `equation` környezetet kell használnod, ez automatikusan matematikai módba kapcsol át, és a számozást is elvégzi.

Az ABC hegyesszögű háromszögünk C-ből induló magasságának talppontja T.

```
Az \(ABC\) hegyesszög\H{u}
háromszögünk $C$-b\H{o}l induló
magasságának talppontja $T$.
```

Mely $P(x, y)$ pontokra teljesül, hogy

$$\frac{2x - y - 1}{x - 2y + 1} \leq 0.$$

Egy négyzet alapú ...

```
Mely $P(x,y)$ pontokra teljesül,
hogy
\[\frac{2x-y-1}{x-2y+1}\leq 0.\]
Egy négyzet alapú \ldots
```

Rövid algebrai rendezés után:

$$s_a^2 = \frac{2b^2 + 2c^2 - a^2}{4} \tag{4.1}$$

Ehhez az eredményhez más ...

```
R\"ovid algebrai rendez\'es ut\'an:
\begin{equation}
s_a^2=\frac{2b^2+2c^2-a^2}{4}
\end{equation}
Ehhez az eredm\'enyhez
m\'as \ldots
```

Felhívnám a figyelmedet az első példára. Ebben a példában ugyanis egyetlen igazi képlet sincs, mégis matematikai módot használtam. Matematikai módban ugyanis a LATEX másfajta betűket használ, ezért ha olyan jeleket, betűket írsz, amelyek matematikai mennyiségeket, fogalmakat jelölnek, mindig használj matematikai módot!

A matematikai módokra azért van szükség, mert képletek formázásához teljesen más módszereket kell használni, mint szövegekéhez. Lássuk miben tér el a matematikai mód a szövegmódtól.

- A szóköz karaktereket a LATEX nem veszi figyelembe, ezek csak arra valók, hogy a képlet egyes elemeit elválasszák egymástól. A helykihagyásokat a LATEX a képlet alapján állapítja meg. Természetesen lehetőség van a helykihagyások módosítására, speciális parancsokkal.
- Minden képlet pontosan egyetlen bekezdés, egy képlet sem tartalmazhat üres sort.
- A képletben szereplő összes betűt matematikai változóként értelmezi a program. Külön parancsokkal lehetőség van szövegek írására képleteken belül

4.2. A képletek elemei

4.2.1. Görög betűk

A képletekben nagyon gyakran előforduló görög betűk mindegyikére egy külön LaTeX parancs létezik, ami nem más, mint a betű neve. Görög nagybetűket hozol létre, ha a betű nevének első betűjét nagy betűvel írod. Az egyes betűkhöz tartozó parancsokat a 4.3. és a 4.4. táblázatokban találod. Néhány betűnek többféle változata van, a második változat neve `var` előtaggal kezdődik, nézd meg az említett táblázatokat!

A görög ábécé: $\alpha, \beta, \gamma, \ldots\ \omega$

$$F(x) = \Phi\left(\frac{x-m}{\sigma}\right)$$

```
A görög ábécé:
$\alpha,\beta,
  \gamma,\ldots\ \omega$
\[F(x)=\Phi\left(
  \frac{x-m}{\sigma}\right)\]
```

4.2.2. Felső és alsó indexek

Felső indexek írására a '^', alsó indexek írására pedig a '_' parancskarakter szolgál. Ezek az utánuk álló karaktert helyezik az indexbe, ha egynél több karakter áll az indexben, akkor csoportok használata a megoldás.

$$(x_0+y_0)^{n+1} = (x_0+y_0)^n(x_0+y_0)$$

```
\begin{displaymath}
(x_0+y_0)^{n+1}=
  (x_0+y_0)^n(x_0+y_0)
\end{displaymath}
```

4.2.3. Gyökjelek

A gyökjelek meglehetősen gyakran fordulnak elő képletekben. Gyökjeleket az

`\sqrt`

paranccsal hozhatsz létre. Ha nem négyzetgyökről, van szó, hanem n. gyökről, akkor n-t add meg opcionális paraméterként! A gyökjel hosszát a LaTeX automatikusan állapítja meg.

$$\sqrt[3]{a^3b^3} = ab \qquad (4.2)$$

```
\begin{equation}
\sqrt[3]{a^3b^3}=ab
\end{equation}
```

Ha csak egyszerűen egy gyökjelre ($\surd$) van szükséged, akkor a

`\surd`

parancs a megoldás.

4.2.4. Aláhúzás, föléhúzás, kapcsos zárójelek

A következő két parancs vízszintes vonalak létrehozására szolgál.

`\underline`, `\overline`

Az első parancs aláhúzva, a második pedig „föléhúzva” jeleníti meg argumentumát.

$\underline{a+b}$ $\overline{m+n}$

```
$\underline{a+b}$\qquad
$\overline{m+n}$
```

Az alábbi parancsok pedig hosszú, vízszintes kapcsos zárójeleket adnak.

`\overbrace` és `\underbrace`

Felül illetve alul kapcsolják összes paraméterként átadott képletet. Természetesen lehetőség van a „kapocs” fölé, illetve alá szöveget írni, a ’^’, illetve a ’_’ parancsokkal. Figyeld meg a példát!

$$\underbrace{a_1+a_2+\ldots+a_n}_{n}$$

```
\[\underbrace{a_1+a_2+
  \ldots+a_n}_n\]
```

$$\overbrace{x_1^2\,x_2^2\ldots x_n^2}^{(x_1\ldots x_n)^2}$$

```
\[\overbrace{x_1^2 x_2^2\ldots
  x_n^2}^{(x_1\ldots x_n)^2}\]
```

4.2.5. Vektorok

A vektorokat gyakran ábrázolják a nevük fölé húzott nyíllal, erre szolgál LaTeX-ben a

`\vec`

parancs. Ha a vektor neve nem csak egy karakterből áll, akkor az

`\overrightarrow` és `\overleftarrow`

parancsokat ajánlom.

$\vec{a}+\vec{a}=\vec{b}$ vagy $\overrightarrow{AB}$, illetve $\overleftarrow{BA}$

```
$\vec{a}+\vec{a}=\vec{b}$ vagy
$\overrightarrow{AB}$, illetve
$\overleftarrow{BA}$
```

\arccos	\cos	\csc	\exp	\ker	\limsup	\min	\sinh
\arcsin	\cosh	\deg	\gcd	\lg	\ln	\Pr	\sup
\arctan	\cot	\det	\hom	\lim	\log	\sec	\tan
\arg	\coth	\dim	\inf	\liminf	\max	\sin	\tanh

4.1. táblázat. A LaTeX beépített függvénynevei

4.2.6. Ékezetek

Matematikai módban nem használhatod az ékezetes betűket előállító parancsokat, a helyettük használhatókat a matematikai módban használható speciális ékezetekkel együtt a 4.2. táblázatban találod meg a fejezet végén, a 56. oldalon. Az eredeti ékezetek tiltásának az az oka, hogy képletekben az ékezeteknek speciális szerepük van. Valójában nem nagy korlátozásról van szó, hiszen matematikai mennyiségeket amúgy sem jelölnénk 'ő' vagy 'ű' betűkkel. Annál gyakoribb azonban például a '$\dot{o}$' vagy '$\dot{u}$' jelölés.

Ha a deriváltak jelzésére apró vesszőket szeretnél használni, ezt nagyon egyszerűen megteheted:

$y = x^2 \quad y' = 2x \quad y'' = 2$

```
$y=x^2$\quad $y'=2x$\quad $y''=2$
```

Az ékezetet előállító parancsok közül kicsit különleges még a

`\widehat` és a `\widetilde`

parancs, mert ezek nem csak egy, hanem több betű fölé helyezik ékezeteiket.

$\widehat{ABC} \quad \widetilde{ABC}$

```
$\widehat{ABC}$\quad
$\widetilde{ABC}$
```

4.2.7. Függvénynevek

Említettem már, hogy a LaTeX a matematikai módban írt betűket rendre változóneveknek tekinti. Ez nem túl hasznos akkor, ha függvényneveket írsz. (Itt most nem csak a matematikai értelemben vett függvénynevekre gondolok, mint azt majd meglátod.) Ezért a leggyakrabban használt függvények nevének leírására külön parancsok vannak, ezek használata nagyon fontos, a LaTeX a függvény típusának megfelelően fogja megformázni a függvénynevet. A parancsokat a 4.1. táblázatban találod.

```
\[\lim_{x\to0}\frac{\sin x}{x}=1\]
```

$$\lim_{x\to 0} \frac{\sin x}{x} = 1$$

Az egyes függvényneveknek lehetnek olyan paraméterei, amit a nevek alá kell, írni, ilyen például a példában a `\lim` parancs. Ezeket a '_' karakterrel adhatod meg.

A számelméletben gyakran használt `\mod` hiányzik a táblázatból, mert ennek kétféle változata is létezik, a

`\pmod` és a `\bmod`.

Használatukkal kapcsolatban lásd a következő példát!

$a \bmod b$
$a \equiv b \pmod m$

```
$a \bmod b$\\
$a \equiv b \pmod m$
```

4.2.8. Törtek és binomiális együtthatók

Törtek írására szolgál a

`\frac{...}{...}`

parancs, amelynek két paramétere a tört számlálója és nevezője. Ha a törtek indexben szerepelnek, akkor néha jobban néz ki a '/' jellel előállított forma.

$$\frac{x}{x+1} \quad x^{\frac{x}{x+1}} \quad x^{1/2}$$

```
\[\frac{x}{x+1}\quad
x^\frac{x}{x+1}\quad
x^{1/2}\]
```

A binomiális együtthatók formázása csak annyiban különbözik a törtekétől, hogy nincs szükség törtvonalra, ezért is tárgyalom őket együtt. Két parancs tartozik ehhez a témához, a

`{... \choose ...}` és az `{... \atop ...}`,

az első a zárójeleket is kirakja, míg a második nem.

$$\binom{r}{m}\binom{m}{k} = \binom{r}{k}\binom{r-k}{m-k}.$$

$$\left\{ {s-t(n-k) \atop k} \right\}$$

```
\[{r \choose m}{m \choose k}=
  {r \choose k}{r-k \choose m-k}.\]
\[{s-t(n-k) \atop k}\]
```

4.2.9. Összegek és integrálok

A két jel formázása magyon hasonló, és mindkettő alá és fölé szöveg kerülhet. A használandó parancsok a

`\sum` és az `\int`.

Az operátorok alá és fölé szöveget természetesen a '_' és a '^' parancsokkal írhatsz.

$$\exp(x)=\sum_{n=0}^{\infty} \frac{x^n}{n!}$$

```
\[\exp(x)=\sum_{n=0}^{\infty}
  \frac{x^n}{n!}\]
```

$$m\int_a^b \rho \leq \int_a^b f\rho \leq M\int_a^b \rho.$$

```
\[m\int_a^b\rho\leq
  \int_a^b f\rho\leq M
  \int_a^b\rho.\]
```

4.2.10. Zárójelek, határolójelek

A határolójelek széles skáláját használhatod. A kerek és szögletes zárójelek, a billentyűzeten megtalálhatók, nem is okoznak gondot. Bár a kapcsos zárójelek is rajta vannak a legtöbb billentyűzeten, ezeket nem használhatod „simán", hiszen a LaTeX csoportosításra használja őket, a

`\{` és `\}`

forma a megoldás. A többi speciális határolójelet a 4.9. táblázatban láthatod, a 58. oldalon.

$$(a,b,c) \neq \{a,b,c\}$$

```
\begin{displaymath}
(a, b, c)\neq \{a, b, c\}
\end{displaymath}
```

Gyakran szükség lehet az alapértelmezettnél nagyobb zárójelekre, határolójelekre. Ha egy határolójel elé a

`\left` illetve a `\right`

parancsot írod, akkor a LaTeX automatikusan a megfelelő nagyságú zárójelet fogja használni. A `\left` parancs *nyitó*, a `\right` pedig *záró* határolójelek esetén használatos. Fontos, hogy minden `\left` parancsot egy `\right` parancs zárjon le! Ha esetleg nincs szükséged a lezáró határolójelre, azt nem akarod kiírni, akkor a

`\right.`

parancsot használd, ez egy üres záró határolójelet eredményez, logikailag lezárja a nyitó határolójelet, a papírra azonban nem kerül záró határolójel. Használatára később, az esetszétválasztásnál láthatsz egy jó példát.

$$\left(1+\frac{1}{1+x^2}\right)^3$$

```
\[\left(1+\frac1{1+x^2}\right)^3\]
```

Esetenként szükség van a határolójel méretének kézzel történő beállítására, mert a LaTeX rosszul állapította a meg a használandó méretet. El kell néznünk neki ezeket az apró hibákat, hiszen csak egy számítógépprogram. Mint mindig, most is megvan azonban a lehetőség a beavatkozásra. A

`\big`, `\Big`, `\bigg`, `\Bigg`

parancsok egyre nagyobb határolójeleket adnak meg. A határolójel típusát a parancs után kell írni, mint a `\left` és a `\right` esetében.

$$((a+b)(c+d))(e+f)$$

$$\big((a+b)(c+d)\big)(e+f)$$

$$\big(\Big(\bigg(\Bigg(\quad \big\}\Big\}\bigg\}\Bigg\} \quad \big\|\Big\|\bigg\|\Bigg\|$$

```
\begin{displaymath}
((a+b)(c+d))(e+f)
\end{displaymath}
\begin{displaymath}
\big((a+b)(c+d)\big)(e+f)
\end{displaymath}
$\big(\Big(\bigg(\Bigg($\quad
$\big\}\Big\}\bigg\}\Bigg\}$\quad
$\big\|\Big\|\bigg\|\Bigg\|$
```

4.2.11. Pont, pont, pont...

Korábban már említettem, hogy három pontot az `\ldots` paranccsal állíthatsz elő. Matematikai módban további hasonlók állnak rendelkezésedre:

`\cdots`, `\vdots`, `\ddots`

A `\cdots` parancs is három pontot állít elő vízszintesen, de azt függőlegesen a sor közepére igazítja. Hasznos például, ha a pontok műveleti jelek között állnak. A `\vdots` parancs függőleges három pontot, a `\ddots` pedig átlós három pontot ír. Ezek a parancsok – mint azt később látni fogod – általános mátrixok írásakor hasznosak.

$x_1,\ldots,x_n \qquad x_1+\cdots+x_n$

```
$x_1,\ldots,x_n$\qquad
$x_1+\cdots+x_n$
```

4.3. Helykihagyások

Amint azt már tudod, a LaTeX a helykihagyásokat a képletekben automatikusan állapítja meg. Előfordulhat persze, hogy a LaTeX által adott íráskép nem a legjobb, elvégre a LaTeX csak egy program. Természetesen van lehetőséged a változtatásra, a helykihagyások szabályozására. A

`\,`, `\␣`, `\quad`, `\qquad`

parancsok rendre kicsi, közepes, nagy, illetve még nagyobb helykihagyást hoznak létre.

Az is megeshet, hogy a LaTeX által megadott helykihagyás túl nagy, kisebbre van szükség. Ekkor használhatod a

`\!`

parancsot, ami valójában egy `\,` szélességű negatív szóköz. Ez azt jelenti, hogy a parancs a kurzort balra mozgatja. A következő példából rögtön világos lesz, miről is van szó.

$$\int\int_D g(x,y)\mathrm{d}x\mathrm{d}y$$

helyett

$$\iint_D g(x,y)\,\mathrm{d}x\,\mathrm{d}y$$

```
\newcommand{\ud}{\mathrm{d}}
\begin{displaymath}
\int\int_D g(x,y)
  \ud x\ud y
\end{displaymath}
helyett
\begin{displaymath}
\int\!\!\!\int_D g(x,y)
  \, \ud x\,\ud y
\end{displaymath}
```

4.4. Többsoros mindenféle

4.4.1. Mátrixok

Mátrixok írására a `tabular` környezethez nagyon hasonló `array` környezet szolgál. Az elemeket a '`&`', a sorokat pedig '`\\`' jelek választják el egymástól.

$$\mathbf{A} = \left(\begin{array}{ccc} x_{11} & x_{12} & \ldots \\ x_{21} & x_{22} & \ldots \\ \vdots & \vdots & \ddots \end{array} \right)$$

```
\[ \mathbf{A}=\left(
\begin{array}{ccc}
x_{11} & x_{12} & \ldots \\
x_{21} & x_{22} & \ldots \\
\vdots & \vdots & \ddots
\end{array} \right) \]
```

4.4.2. Esetszétválasztás

Főleg függvények megadásánál használatos a következő formula. Megintcsak az `array` környezetet használd, de a `\right.` üres határolójellel!

$$F(x) = \left\{ \begin{array}{ll} 0 & \text{ha } x \leq 0 \\ \cos x & \text{ha } 0 < x < \pi/2 \\ 1 & \text{ha } \pi/2 \leq x \end{array} \right.$$

```
\[ F(x)=\left\{ \begin{array}{ll}
0 & \textrm{ha $x\leq 0$} \\
\cos x &
\textrm{ha $0<x<\pi/2$} \\
1 & \textrm{ha $\pi/2\leq x$}
\end{array} \right. \]
```

4.4.3. Képletek

Gyakran egy képlet nem fér ki egy sorba, sőt az is lehet, hogy direkt több sorba szeretnéd írni. Erre szolgálnak az `eqnarray` és `eqnarray*` környezetek, ezeket

használd az `equation` helyett! A kettő között a különbség csak annyi, hogy az `eqnarray` minden sort külön képletként számoz, az `eqnarray*` pedig nem számozza meg a formulákat.

A két környezet valójában egy '`{rcl}`' formátumú táblázat szerint működik, a középső oszlopba kell írni a relációs jelet.

$$\begin{aligned} f(x) &= \cos x && (4.3)\\ f'(x) &= -\sin x && (4.4)\\ \int_0^x f(y)\,\mathrm{d}y &= \sin x && (4.5) \end{aligned}$$

```
\begin{eqnarray}
f(x) & = & \cos x \\
f'(x) & = & -\sin x \\
\int_0^x f(y)\,\mathrm{d}y
  & = & \sin x
\end{eqnarray}
```

Az előbbi példában a LaTeX túl sok helyett hagyott ki az egyenlőségjelek két oldalán. Ez orvosolható a `\setlength\arraycolsep{`*hely*`}` paranccsal, figyeld a következő példákat!

Mint már említettem, a hosszú képleteket több sorba kell törni. Két módszert mutatok ennek a feladatnak a megoldására, ezek lényegében azonosak. Bonyolultabb, nagyobb képleteknél megint csak célszerűbb az $\mathcal{AMS}$-LaTeX használata. Ez a csomag több új környezetet definiál erre a célra teljes megoldást adva a problémára.

$$\begin{aligned} \sin x = x &- \frac{x^3}{3!} + \frac{x^5}{5!} - \\ &- \frac{x^7}{7!} + \cdots \end{aligned} \qquad (4.6)$$

```
{\setlength\arraycolsep{2pt}
\begin{eqnarray}
\sin x & = & x-\frac{x^3}{3!}
  +\frac{x^5}{5!}-  \nonumber\\
& & -\frac{x^7}{7!}+\cdots
\end{eqnarray}}
```

$$\begin{aligned} \cos x = 1 &- \frac{x^2}{2!} + \\ &+ \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots \end{aligned} \qquad (4.7)$$

```
\begin{eqnarray}
\lefteqn{\cos x = 1
   -\frac{x^2}{2!}+}\nonumber \\
& & +\frac{x^4}{4!}-
   \frac{x^6}{6!}+\cdots
\end{eqnarray}
```

Figyeld meg az első példában használd csoportosítást! Így a `\setlength` parancs csak a csoporton belül érvényes, a csoporton kívülre nem fejti ki hatását.

Ha a

`\nonumber`

parancsot a sort lezáró '`\\`' jel előtt használhatod, hatására a LaTeX nem számozza meg a sort.

Esetenként a hosszú egyenletek kezelése nagyon bonyolult lehet, könnyebbséget az $\mathcal{AMS}$-LaTeX használata jelent.

4.5. Betűméret matematikai módban

Matematikai módban a LaTeX automatikusan, a tartalomtól függően állapítja meg az egyes részek nagyságát. Erre persze szükség is van, hiszen például az alsó és felső indexeket kisebb betűvel kell szedni, ezek indexeit pedig még kisebbel. Ha egyenletbe szöveget akarsz írni, és a `\textrm` parancsot használod, akkor vigyázz, mert az így írt szöveg mérete nem fog automatikusan változni, hiszen a `\textrm` parancs szövegmódba kapcsol. Létezik egy

`\mathrm`

parancs is, ennél a méretnagyság beállítása már működik, de mivel matematikai módban vagyunk, a szóközöket a LaTeX figyelmen kívül hagyja, és a parancs csak rövid szövegek esetén működik jól. Mint már eddig annyiszor, most is az $\mathcal{AMS}$-LaTeX csomag nyújt tökéletes megoldást, ez módosítja a `\textrm` parancsot.

Néha a LaTeX-nek meg kell adnod a betűméret nagyságát, mert előfordulhat, hogy a program rosszul választja meg azt. (Ez ugyan ritka, de megeshet, hiszen a LaTeX csak egy számítógépprogram.) Erre szolgálnak – nagyság szerint csökkenő sorrendben – a

`\displaystyle`, `\textstyle`, `\scriptstyle`, `\scriptscriptstyle`

parancsok.

$$\frac{\displaystyle\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\displaystyle\Big[\sum_{i=1}^{n}(x_i-\overline{x})^2\sum_{i=1}^{n}(y_i-\overline{y})^2\Big]^{1/2}}$$

```
\begin{displaymath}
\frac
 {\displaystyle
 \sum_{i=1}^n(x_i-\overline x)
  (y_i-\overline y)}
 {\displaystyle
 \bigg[\sum_{i=1}^n
  (x_i-\overline x)^2
  \sum_{i=1}^n(y_i-\overline y)^2
 \bigg]^{1/2}}
\end{displaymath}
```

4.6. Definíciók, tételek

Matematikai témájú könyvekben nagyon gyakran fordulnak elő különböző definíciók és tételek, segédtételek. Ezeket általában meg is számozzák, hogy a rájuk történő hivatkozásokat megkönnyítsék. Természetesen a számozást, és a tételek elkülönítését a szöveg többi részétől a LaTeX automatikusan végzi.

Még a tételek előtt defiálnod kell, hogy hányféle különböző típusú tételt, definíciót, segédtételt, következményt akarsz használni. Ezt még a dokumentum bevezető részében kell megtenned, a `\begin{document}` parancs előtt, így a biztosan megelőzöl vele minden tételt. A definíció a

`\newtheorem{`*név*`}[`*számláló*`]{`*szöveg*`}[`*fejezet*`}`

paranccsal történik. *név* az a név, amit az épp definiált típusú tétel beírásakor használsz, *szöveg* pedig az a szöveg amit a kész dokumentumba szeretnél. Még mielőtt a másik két argumentumot megnéznéd, vess egy pillantást a példára, hogy minden világos legyen.

A következő példához szükséges az, hogy az alábbi két sort a dokumentum elejére, a bevezető részbe írjuk, a `\documentclass{...}` parancs után, de még a `\begin{document}` parancs elé.

```
\newtheorem{defin}{Definíció}
\newtheorem{tetel}{Tétel}
```

Ezek után már működik a következő:

1. Definíció. *Az olyan (a_n) számsorozatot, amelyre $\lim a_n = 0$, nullsorozatnak nevezzük.*

1. Tétel (Bolzano). *Bármely korlátos (valós vagy komplex) számsorozatnak van konvergens részsorozata.*

```
\begin{defin}
Az olyan $(a_n)$
számsorozatot, amelyre
$\lim a_n=0$, nullsorozatnak
nevezzük.
\end{defin}
\begin{tetel}[Bolzano]
Bármely korlátos (valós vagy
komplex) számsorozatnak van
konvergens részsorozata.
\end{tetel}
```

A példában két típust definiáltunk, a `defin` és a `tetel` nevűt, mert a könyvben csak ez a két típus szerepel. A tétel egy opcionális argumentumot is kapott, a LaTeX ezt hozzáírta a tétel címéhez, sőt automatikusan zárójelbe is tette.

Térjünk vissza a `\newtheorem` parancs opcionális argumentumaira. Mint láttad a LaTeX megszámozza a különböző típusú tételeket, és minden típust a többitől függetlenül számoz. Ha azt szeretnéd, hogy két típus számozása ne legyen független egymástól, akkor az utóbb definiált típusnál add meg a *név* után opcionális paraméterként a már definiált típus nevét.

A bevezető részbe, a `\begin{document}` parancs elé kerül:

```
\newtheorem{axi}{Axióma}
\newtheorem{alap}[axi]{Alapigazság}
```

A példa pedig:

1. Axióma. *A valós számok halmaza a rajta értelmezett összeadás (jele: $+$), szorzás (jele $\cdot$) műveletekkel testet alkot.*

2. Alapigazság. *Az $\mathcal{R}$-en értelmezve van egy ($a \leq$ szimbólummal jelölt) rendezési reláció, és $\mathcal{R}$ $\leq$-re nézve rendezett test.*

```
\begin{axi}
A valós számok halmaza a rajta
értelmezett összeadás (jele: $+$),
szorzás (jele $\cdot$)
m\H{u}veletekkel testet alkot.
\end{axi}
\begin{alap}
Az $\mathcal{R}$-en értelmezve van
egy (a $\leq$ szimbólummal jelölt)
rendezési reláció, és $\mathcal{R}$
$\leq$-re nézve rendezett test.
\end{alap}
```

A másik opcionális argumentumot a *szöveg* után lehet megadni, ez határozza meg, hogy a LaTeX mikor nulláza le tételtípus számlálóját. Ha ez például `[section]`, akkor minden fejezetben elölről kezdődik a számozás. A lehetséges értékek: `part`, `chapter`, `section`... A tételtípus számozása is megváltozik, a megadott értéknek megfelelően.

A `\begin{document}` elé, a dokumentum elejére:

```
\newtheorem{murphy}{Murphy}[section]
```

A dokumentumba, a megfelelő helyre pedig:

4.6.1. Murphy. *Ami elromolhat az el is romlik.*

```
\begin{murphy}
Ami elromolhat az el is romlik.
\end{murphy}
```

Ha a fenti példákat kipróbálod, azt fogod tapasztalni, hogy a forrásszöveg nem pontosan az itt mutatott képet eredményezi, a tétel száma és neve fel van cserélve, az angol forma szerint áll. Ezt megváltoztathatod magyar formájúra, a 6. fejezetben leírtaknak megfelelően.

4.7. Kövér betűk, szimbólumok

Csak a LaTeX-et használva meglehetősen nehéz kövér matematikai jelek előállítása, de persze nem lehetetlen. A

`\mathbf`

parancs az argumentumában kapott szöveget kövéren szedi, de álló betűkkel, a matematikai módban használt betűk pedig dőltek. A

`\boldmath`

parancs kövér matematikai jelek írására vált át, de ez csak matematikai módon kívül működik, így ideiglenesen szövegmódba kell kapcsolni használatakor.

$\mu, M \qquad \mu, \mathbf{M} \qquad \boldsymbol{\mu}, \boldsymbol{M}$

```
\begin{displaymath}
\mu,M \qquad \mathbf{\mu, M} \qquad
\textrm{\boldmath $\mu$, $M$}
\end{displaymath}
```

Az $\mathcal{AMS}$-LaTeX csomag új parancsokat definiál kövér betűk és jelek írására, ezzel a példában használt bonyolult szerkezet feleslegessé válik, lásd az $\mathcal{AMS}$-LaTeX leírását!

4.8. Matematikai jelek

Az alábbi táblázatok a matematikai módban használható, különleges jeleket vagy betűket tartalmazó parancsokat tartalmazzák.

A relációs jelek (4.5. táblázat) áthúzott változatát kapod, ha a jel elé a

`\not`

parancsot írod, pl. $\not=$`\not\succ`.

Bár az itt szereplő jelek látszólag minden igényt kielégítenek, az $\mathcal{AMS}$-LaTeX továbbiakat definiál, ezeket megtalálod az $\mathcal{AMS}$-LaTeX leírásában.

$\hat{a}$	`\hat{a}`	$\check{a}$	`\check{a}`	$\tilde{a}$	`\tilde{a}`
$\grave{a}$	`\grave{a}`	$\dot{a}$	`\dot{a}`	$\ddot{a}$	`\ddot{a}`
$\bar{a}$	`\bar{a}`	$\vec{a}$	`\vec{a}`	$\widehat{A}$	`\widehat{A}`
$\acute{a}$	`\acute{a}`	$\breve{a}$	`\breve{a}`	$\widetilde{A}$	`\widetilde{A}`

4.2. táblázat. Ékezetek matematikai módban

α	`\alpha`	θ	`\theta`	o	`o`	υ	`\upsilon`
β	`\beta`	ϑ	`\vartheta`	π	`\pi`	ϕ	`\phi`
γ	`\gamma`	ι	`\iota`	ϖ	`\varphi`	φ	`\varphi`
δ	`\delta`	κ	`\kappa`	ρ	`\rho`	χ	`\chi`
ϵ	`\epsilon`	λ	`\lambda`	ϱ	`\varrho`	ψ	`\psi`
ε	`\varepsilon`	μ	`\mu`	σ	`\sigma`	ω	`\omega`
ζ	`\zeta`	ν	`\nu`	ς	`\varsigma`		
η	`\eta`	ξ	`\xi`	τ	`\tau`		

4.3. táblázat. Görög kisbetűk

Γ	`\Gamma`	Λ	`\Lambda`	Σ	`\Sigma`	Ψ	`\Psi`
Δ	`\Delta`	Ξ	`\Xi`	Υ	`\Upsilon`	Ω	`\Omega`
Θ	`\Theta`	Π	`\Pi`	Φ	`\Phi`		

4.4. táblázat. Görög nagybetűk

$<$	<	$>$	>	$=$	=
$\leq$	\leq vagy \le	$\geq$	\geq vagy \ge	$\equiv$	\equiv
$\ll$	\ll	$\gg$	\gg	$\doteq$	\doteq
$\prec$	\prec	$\succ$	\succ	$\sim$	\sim
$\preceq$	\preceq	$\succeq$	\succeq	$\simeq$	\simeq
$\subset$	\subset	$\supset$	\supset	$\approx$	\approx
$\subseteq$	\subseteq	$\supseteq$	\supseteq	$\cong$	\cong
$\sqsubset$	\sqsubset	$\sqsupset$	\sqsupset	$\bowtie$	\Join
$\sqsubseteq$	\sqsubseteq	$\sqsupseteq$	\sqsupseteq	$\bowtie$	\bowtie
$\in$	\in	$\ni$	\ni, \owns	$\propto$	\propto
$\vdash$	\vdash	$\dashv$	\dashv	$\models$	\models
$\mid$	\mid	$\parallel$	\parallel	$\perp$	\perp
$\smile$	\smile	$\frown$	\frown	$\asymp$	\asymp
$:$	:	$\notin$	\notin	$\neq$	\neq, \ne

4.5. táblázat. Relációs jelek

$+$	+	$-$	-		
$\pm$	\pm	$\mp$	\mp	$\triangleleft$	\triangleleft
$\cdot$	\cdot	$\div$	\div	$\triangleright$	\triangleright
$\times$	\times	$\setminus$	\setminus	$\star$	\star
$\cup$	\cup	$\cap$	\cap	$*$	\ast
$\sqcup$	\sqcup	$\sqcap$	\sqcap	$\circ$	\circ
$\vee$	\vee, \lor	$\wedge$	\wedge, \land	$\bullet$	\bullet
$\oplus$	\oplus	$\ominus$	\ominus	$\diamond$	\diamond
$\odot$	\odot	$\oslash$	\oslash	$\uplus$	\uplus
$\otimes$	\otimes	$\bigcirc$	\bigcirc	$\amalg$	\amalg
$\triangle$	\bigtriangleup	$\bigtriangledown$	\bigtriangledown	$\dagger$	\dagger
$\triangleleft$	\lhd	$\triangleright$	\rhd	$\ddagger$	\ddagger
$\trianglelefteq$	\unlhd	$\trianglerighteq$	\unrhd	$\wr$	\wr

4.6. táblázat. Operátorok

$\sum$	\sum	$\bigcup$	\bigcup	$\bigvee$	\bigvee	$\bigoplus$	\bigoplus
$\prod$	\prod	$\bigcap$	\bigcap	$\bigwedge$	\bigwedge	$\bigotimes$	\bigotimes
$\coprod$	\coprod	$\bigsqcup$	\bigsqcup			$\bigodot$	\bigodot
$\int$	\int	$\oint$	\oint			$\biguplus$	\biguplus

4.7. táblázat. Nagy operátorok

$\leftarrow$	\leftarrow, \gets	$\longleftarrow$	\longleftarrow	$\uparrow$	\uparrow
$\rightarrow$	\rightarrow, \to	$\longrightarrow$	\longrightarrow	$\downarrow$	\downarrow
$\leftrightarrow$	\leftrightarrow	$\longleftrightarrow$	\longleftrightarrow	$\updownarrow$	\updownarrow
$\Leftarrow$	\Leftarrow	$\Longleftarrow$	\Longleftarrow	$\Uparrow$	\Uparrow
$\Rightarrow$	\Rightarrow	$\Longrightarrow$	\Longrightarrow	$\Downarrow$	\Downarrow
$\Leftrightarrow$	\Leftrightarrow	$\Longleftrightarrow$	\Longleftrightarrow	$\Updownarrow$	\Updownarrow
$\mapsto$	\mapsto	$\longmapsto$	\longmapsto	$\nearrow$	\nearrow
$\hookleftarrow$	\hookleftarrow	$\hookrightarrow$	\hookrightarrow	$\searrow$	\searrow
$\leftharpoonup$	\leftharpoonup	$\rightharpoonup$	\rightharpoonup	$\swarrow$	\swarrow
$\leftharpoondown$	\leftharpoondown	$\rightharpoondown$	\rightharpoondown	$\nwarrow$	\nwarrow
$\rightleftharpoons$	\rightleftharpoons	$\Longleftrightarrow$	\iff	$\leadsto$	\leadsto

4.8. táblázat. Nyilak

$($	(	$)$	)	$\uparrow$	\uparrow	$\Uparrow$	\Uparrow
$[$	[, \lbrack	$]$	] \rbrack	$\downarrow$	\downarrow	$\Downarrow$	\Downarrow
$\{$	\{, \lbrace	$\}$	\}, \rbrace	$\updownarrow$	\updownarrow	$\Updownarrow$	\Updownarrow
$\langle$	\langle	$\rangle$	\rangle	$\vert$	|, \vert	$\Vert$	\\|, \Vert
$\lfloor$	\lfloor	$\rfloor$	\rfloor	$\lceil$	\lceil	$\rceil$	\rceil

4.9. táblázat. Határolójelek

$\left(\right.$	\lgroup	$\left.\right)$	\rgroup	$\left\lmoustache\right.$	\lmoustache	$\left.\rmoustache\right\rangle$	\rmoustache
$\left\vert\right.$	\arrowvert	$\left.\Vert\right\Vert$	\Arrowvert	$\left\vert\right.$	\bracevert		

4.10. táblázat. Nagy határolójelek

$\ldots$	\dots	$\cdots$	\cdots	$\vdots$	\vdots	$\ddots$	\ddots
$\hbar$	\hbar	$\imath$	\imath	$\jmath$	\jmath	ℓ	\ell
$\Re$	\Re	$\Im$	\Im	$\aleph$	\aleph	$\wp$	\wp
$\forall$	\forall	$\exists$	\exists	$\mho$	\mho	∂	\partial
$'$	'	$\prime$	\prime	$\emptyset$	\emptyset	∞	\infty
∇	\nabla	$\triangle$	\triangle	$\Box$	\Box	$\Diamond$	\Diamond
$\bot$	\bot	$\top$	\top	$\angle$	\angle	$\surd$	\surd
$\diamondsuit$	\diamondsuit	$\heartsuit$	\heartsuit	$\clubsuit$	\clubsuit	$\spadesuit$	\spadesuit
$\neg$	\neg, \lnot	$\flat$	\flat	$\natural$	\natural	$\sharp$	\sharp

4.11. táblázat. Egyéb matematikai jelek

†	\dag	§	\S	©	\copyright
‡	\ddag	¶	\P	£	\pounds

4.12. táblázat. Szöveg módban is használható jelek

5. fejezet

További lehetőségek

Ha idáig eljutottál az olvasásban, akkor eleget tudsz a LaTeX-ről ahhoz hogy nyugodtan elkezdhesd dolgozataid vagy akár diplomamunkád írását. Az ezután következő részek csupán egy kis ízelítőül szolgálnak a LaTeX további lehetőségeiről. Azt javaslom, gyorsan lapozd át a könyv hátralévő részét, s csak azokat az oldalakat olvasd el, amelyekre szükséged van. Az itt bemutatottak messze nem merítik a LaTeX lehetőségeit, csak apró morzsák.

5.1. Betűtípusok, betűméretek

Ha tanulmányoztál már szövegszerkesztő (azaz szövegformázó) programokat bemutató könyveket, akkor jelen könyv olvasásakor bizonyára feltűnt, hogy eddig sehol sem szerepelt ez a téma. Természetesen ez nem azért van, mert a LaTeX-nek nem képes különböző méretű és típusú karakterek megjelenítésére. Sokkal inkább azért, mert LaTeX-et használva szinte soha nincs szükség arra, hogy a szerző a betűk méretét megváltoztassa. (Általában a betűtípus megváltoztatására sincs szükség, szavak kiemelésére jó az `\emph` parancs.) A LaTeX ugyanis automatikusan állapítja meg a betűk méretét, és igencsak jó munkát végez. (A szerzőnek jelen könyv írásakor egyszer sem volt szüksége a betűméret beállítására.) Ha esetleg mégis meg akárnád változtatni valamiért a betűk méretét, az 5.1. táblázat parancsait használd!

Valójában ezek a parancsok sem a fizikai betűméretet adják meg, a fizikai

`\tiny`	apró betűk	`\Large`	nagyobb betűk
`\scriptsize`	nagyon kicsi betűk	`\LARGE`	nagyon nagy
`\footnotesize`	elég kicsi betűk	`\huge`	hatalmas
`\small`	kicsi betűk	`\Huge`	legnagyobb
`\normalsize`	alap betűméret		
`\large`	nagy betűk		

5.1. táblázat. Betűméretetek

`\textrm{...}`	antikva	`\textsf{...}`	groteszk
`\texttt{...}`	írógép		
`\textmd{...}`	normál	`\textbf{...}`	**kövér**
`\textup{...}`	álló	`\textit{...}`	*kurzív*
`\textsl{...}`	*dőlt*	`\textsc{...}`	KISKAPITÁLIS
`\emph{...}`	*kiemelt*	`\textnormal{...}`	alap betűtípus

5.2. táblázat. Betűtípusok

Parancs	*Példa*	*Kimenet*
`\mathcal{...}`	`$\mathcal{B}=c$`	$\mathcal{B} = c$
`\mathrm{...}`	`$\mathrm{K}_2$`	K_2
`\mathbf{...}`	`$\sum x=\mathbf{v}$`	$\sum x = \mathbf{v}$
`\mathtt{...}`	`$\mathtt{L}(b,c)$`	$\mathtt{L}(b, c)$
`\mathnormal{...}`	`$\mathnormal{R_{19}}\neq R_{19}$`	$R_{19} \neq R_{19}$
`\mathit{...}`	`$\mathit{ffi}\neq ffi$`	$\mathit{ffi} \neq ffi$

5.3. táblázat. Matematikai betűtípusok

betűméretet a LaTeX állapítja meg figyelembe véve a dokumentum osztályát is. A `\large` parancs például az átlagosnál nagyobb betűméretre kapcsol, ez a méret természetesen más, amikor könyvet nyomtatunk, és más amikor fóliákat, ahol az alap betűméret is eleve nagyobb. A betűk típusát megváltoztató parancsokat az 5.2. táblázatban találod.

A kicsi és **kövér** rómaiak uralták egész *Itáliát.*

```
{\small A kicsi és
\textbf{kövér} rómaiak uralták}
{\Large egész
\textit{Itáliát}.}
```

Fontos, hogy ha megváltoztatjuk a betűk méretét, akkor a betűk típusa változatlan marad és fordítva. Azaz, ha dőlt betűvel írunk és nagyobb betűtípusra kapcsolunk, akkor nagyobb, de még mindig dőlt betűket kapunk. Azt hiszem ez azok számára, akik most ismerkednek a LaTeX-hel evidens, de nem az azok számára, akik használták már használták a LaTeX 2.09 változatot.

Matematikai módban a betűtípust megváltoztató parancsokat arra használhatod, hogy pár szó erejéig visszatérj szövegmódba. Ha matematikai módban akarsz más betűtípust használni, az 5.3. táblázat parancsai állnak rendelkezésedre.

Fel kell hogy hívjam a figyelmedet egy már említett, LaTeX kézirat írásakor

gyakran használt technikára, a csoportosításra. Ha egy csoporton belül megváltoztatod a betűk méretét vagy típusát, akkor a csoport lezárásával az eredeti (a csoport megkezdésekor használatos) betűméret és betűtípus áll vissza.

Szeretem a nagy és kicsi betűket együtt.

```
Szeretem a {\LARGE nagy és
{\small kicsi} bet\H{u}ket}
együtt.
```

Természetesen a betűk méretét megváltoztató parancsok megváltoztatják a sorok távolságát is, de csak akkor, ha a parancs még érvényben van a bekezdés lezárásakor. Figyeld meg a `\par` parancs pozícióját a következő példákban!

Ne olvassd el! Nem igaz. Higgy nekem!

```
{\Large Ne olvassd el! Nem
igaz. Higgy nekem!\par}
```

Ne olvassd el! Nem igaz. Higgy nekem!

```
{\Large Ne olvassd el! Nem
igaz. Higgy nekem!}\par
```

5.2. Helykihagyások

5.2.1. Sorköz

Ha meg szeretnéd növelni a sorközt dokumentumodban, akkor a

`\linespread{`*faktor*`}`

parancsot írd a dokumentum bevezető részébe. Másfélszeres sorközhöz használd a `\linespread{1.3}`, dupla sorközhöz a `\linespread{1.6}` parancsot! Alapban a sortávolság nincs megnövelve, megnyújtva, az alapértelmezett faktor 1. Általában nem néz ki jól, ha túl nagy sorközt állítasz be, és árt a dokumentum olvashatóságának, a gondolatok folyamatosságának is.

5.2.2. Bekezdések formázása

A bekezdések jelölésére, egymástól való elválasztására kétféle módszer létezik. Az első, hogy az egyes bekezdések között egy kis helyet hagyunk ki, a második pedig, hogy a bekezdések első sorát egy kicsivel beljebb kezdjük, innen a bekezdés szó. A LaTeX alapban a második módszert használja. Ha a következő parancsokat elhelyezed a dokumentum bevezető részébe, az első módszerre térhetsz át.

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1ex plus 0.5ex minus 0.2ex}
```

Előfordulhat, hogy a LaTeX valamilyen alapértelmezett „hosszúságát” (a margók mérete, a bekezdések közötti üres hely mérete,... stb.) szeretnénk megváltoztatni, az előbb is erről volt szó. Erre szolgál a

`mm`	milliméter 1/25 inch
`cm`	centiméter = 10 mm
`in`	inch 25.4 mm
`pt`	pont 1/72 inch $\frac{1}{3}$ mm
`em`	az aktuális betűkészlet `m` betűjének szélessége
`ex`	az aktuális betűkészlet `x` betűjének magassága

5.4. táblázat. A LaTeX fontosabb mértékegységei és nagyságuk

`\setlength{`*minek a hossza*`}{`*új hossz*`}`

parancs. Az új hosszot többféle mértékegységben adhatjuk meg, lásd az 5.4. táblázatot! Az előbbi példában szereplő `\parindent` azt adja meg, hogy mennyivel kezdődjön beljebb a bekezdés első sora, a `\parskip` pedig a bekezdések közötti távolság mértéke. Vigyázz, ha megváltoztatod a `\parindent` mértékét, akkor a tartalomjegyzék nem lesz olyan szép, mint előtte, persze a tartalomjegyék előtt visszaírhatod az eredetit.

Ha egy olyan bekezdést szeretnél beljebb kezdeni, ami alapból nem kezdődik beljebb, a bekezdést kezd az

`\indent`

paranccsal! Ez persze csak akkor működik, a `\parindent` értékét nem állítottad nullára. Ha az összes címsor utáni bekezdést beljebb szeretnéd kezdeni (ezeket a LaTeX alapban nem kezdi beljebb), akkor használd az `indentfirst` csomagot!

Ha egy bekezdést, amit a LaTeX beljebb kezdene nem szeretnél beljebb kezdeni, a bekezdés első parancsa a

`\noindent`

parancs legyen. Ez jól jöhet akkor, ha a dokumentumod nem címsorral kezdődik, és így az első bekezdést nem akarod beljebb kezdeni.

Általában elmondható, hogy nem érdemes a `\parindent` és `\parskip` nagyságával manipulálni, a LaTeX jól állapítja meg ezeket. Csak akkor szabad őket bolygatni, ha valamilyen különleges hatást szeretnél elérni.

5.2.3. Vízszintes helykihagyás

A LaTeX automatikusan állapítja meg a szavak és mondatok közötti helykihagyásokat. Plusz helykihagyást eredményez a

`\hspace{`*hossz*`}`

parancs. Ha a helykihagyás a sor végére vagy elejére kerülne, akkor elveszik. Ha nem szeretnéd elveszteni a sor eleji vagy sor végi üres helyet sem, a parancsnak

van egy csillagos változata (`\hspace*`) az ilyen esetekre. A *hossz* paraméter egyszerűbb esetekben egy értékből és egy mértékegységből áll. A legfontosabb mértékegységek az 5.4. táblázatban vannak felsorolva.

Ez a hely pontosan 1,5 cm.

```
Ez\hspace{1.5cm}a hely
pontosan 1,5 cm.
```

Speciális, „megnyújtható” helykihagyást eredményez a

`\stretch{`*n*`}`

parancs. Ez a hely addig nyúlik, amíg a sorban az összes fennmaradó helyet kitölti. Ha két `\hspace{\stretch{`*n*`}}` parancs van egy soron belül, akkor azok az *n* paraméternek megfelelően nyúlnak meg. Nagyobb szám nagyobb nyúlást eredményez.

x x x

```
x\hspace{\stretch{1}}
x\hspace{\stretch{3}}x
```

5.2.4. Függőleges helykihagyás

Mint tudjuk a bekezdések, címsorok közötti helykihagyásokat a LaTeX állapítja meg. Ha szükséges plusz helykihagyást hozhatunk létre *két bekezdés között* a

`\vspace{`*hossz*`}`

paranccsal. Ezt a parancsot legcélszerűbb a kéziratban két üres sor közé írni, így biztos két bekezdés közé kerül. Ha a helykihagyást lap alján vagy lap tetején sem szeretnéd elveszíteni, használd a csillagos (`\vspace*`) változatot! A `\stretch` parancs a `\pagebreak` paranccsal együtt lehetőséget ad, hogy egy oldal utolsó sorába írjunk, vagy valamilyen szöveget függőlegesen középre helyezzünk egy oldalon.

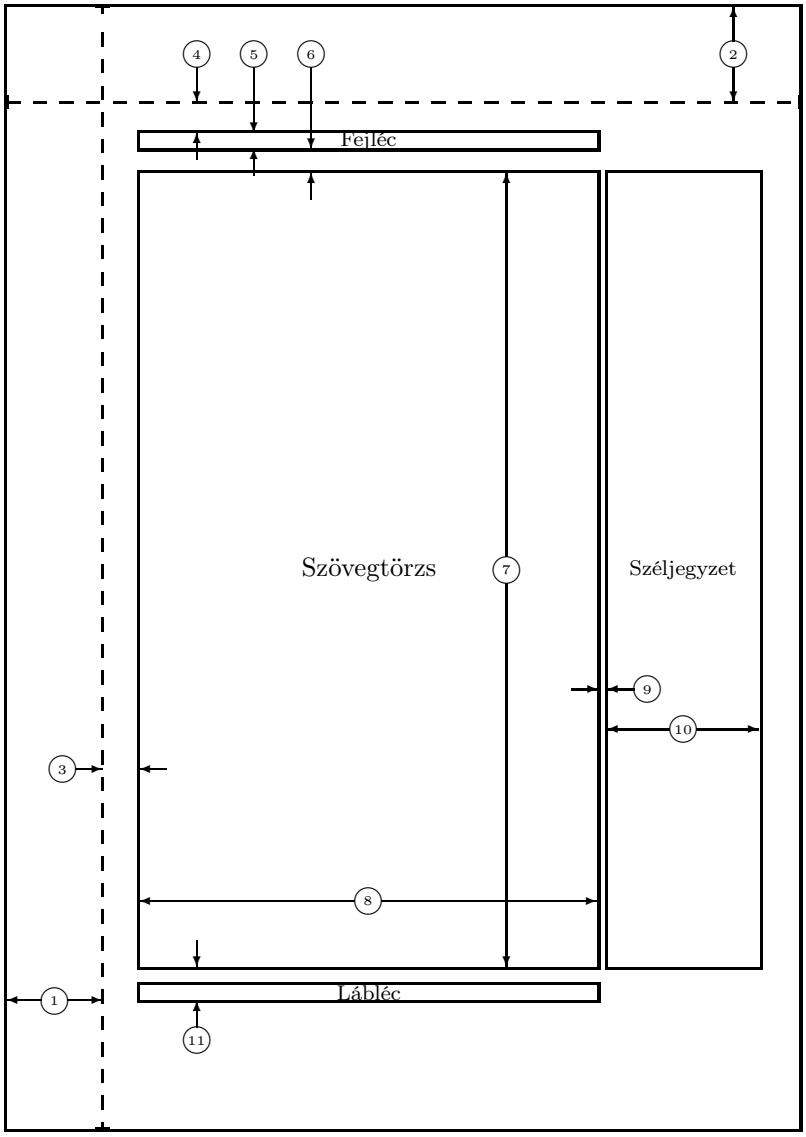
```
Mindenféle szöveg...

\vspace{\stretch{1}}
Ez a sor az oldal utolsó sora.\pagebreak
```

Ha egy bekezdésen belül szeretnél két sor között helyet kihagyni, akkor a hely kívánt hosszát írd a \\ parancs után!

`\\[`*hossz*`]`

Vigyázz, mielőtt a különféle helykihagyásokkal manipulálnál, nézz körül, érdeklődj LaTeX szakemberektől a feladattal kapcsolatban, mert általában nem a fenti parancsok jelentik a legjobb megoldást.



1	egy inch + \hoffset	2	egy inch + \voffset
3	\oddsidemargin = 28pt	4	\topmargin = 23pt
5	\headheight = 12pt	6	\headsep = 18pt
7	\textheight = 598pt	8	\textwidth = 345pt
9	\marginparsep = 7pt	10	\marginparwidth = 115pt
11	\footskip = 25pt		\marginparpush = 5pt (nincs az ábrán)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 597pt		\paperheight = 845pt

5.1. ábra. Egy oldal felépítése

5.3. Az oldal szerkezete

A `\documentclass` parancs opcionális paramétereként megadhatjuk a LaTeX-nek a lapméretet különféle kapcsolókkal. Ezután a LaTeX automatikusan állapítja meg a margókat, a lábjegyzetek helyét és távolságát a szövegtől, az összes paramétert. Előfordulhat, hogy nem felelnek meg az alapértékek, meg kell őket változtatnod. Az 5.1. ábra mutatja az összes paramétert, amit megváltoztathatsz. Az ábrát a `layout` csomag segítségével hoztam létre.

A paraméterek megváltoztatására két parancs áll rendelkezésedre. Az elsőt, a `\setlength` parancsot már ismered. A második, az

`\addtolength{`*paraméter*`}{`*hossz*`}`

parancs a megadott hosszúsággal megnöveli a *paraméter* értékét, ez a parancs néha még hasznosabb is lehet, mint a `\setlength`. Például, ha egy centiméterrel meg szeretnéd növelni a szövet szélességét az oldalon, az alábbi parancsokat írd a dokumentum bevezető részébe:

```
\addtolength{\hoffset}{-0.5cm}
\addtolength{\textwidth}{1cm}
```

5.4. Irodalomjegyzék készítése

Irodalomjegyzék készítésére szolgál a `thebibliography` környezet szolgál. Minden bejegyzés egy

`\bibitem{`*jelző*`}`

paranccsal kezdődik. A *jelző* paraméter arra szolgál, hogy a dokumentumon belül hivatkozz a műre a

`\cite{`*jelző*`}`

paranccsal. A bejegyzések számozása automatikusan történik. A környezet kezdetét jelző `\begin{thebibliography}` parancs utáni paraméter ezeknek a számoknak a maximális szélességére vonatkozik. Íme egy példa:

```
Knuth~\cite{tb} szerint a \TeX\ldots

\begin{thebibliography}{99}
\bibitem{tb} D.~E.~Knuth: \emph{The \TeX{}book},
Addison-Wesley
\end{thebibliography}
```

A `thebibliography` környezetet csak néhány bejegyzésből álló irodalomjegyzék készítéséhez ajánlom. Könyvek sok bejegyzést tartalmazó irodalomjegyzékének elkészítése bonyolult feladat is lehet. Hatékony segítség lehet ekkor a LaTeX egy kiegészítése, a BibTeX, egy speciális TeX változat, amit speciálisan irodalomjegyzékek készítésére fejlesztettek ki. A BibTeX része minden

Példa	*Bejegyzés*	*Megjegyzés*
`\index{hello}`	hello, 1	Egyszerű bejegyzés
`\index{hello!mindenki}`	mindenki, 3	Albejegyzés
`\index{Sam@\textsl{Sam}}`	*Sam*, 2	Formázott bejegyzés
`\index{Lin@\textbf{Lin}}`	**Lin**, 7	Ez is az
`\index{Jenny|textbf}`	Jenny, **3**	Formázott oldalszám
`\index{Joe|textit}`	Joe, *5*	Ez is az

5.5. táblázat. Különféle bejegyzések a tárgymutatóban

LaTeX változatnak. Használatával kapcsolatban lásd a programhoz mellékelt dokumentációt.

5.5. Tárgymutató

A könyvek nagyon jól használható része a tárgymutató. A LaTeX és a `makeindex` program segítségével tárgymutatók készítése nagyon egyszerű. A `makeindex` program a legtöbb rendszeren rendelkezésedre áll, része a TeX programcsomagoknak (idegen szóval diszribúcióknak).

A tárgymutató készítése a következőképpen történik. Először is be kell töltened a `makeidx` csomagot a `\usepackage{makeidx}` paranccsal, és ki kell adnod egy

`\makeindex`

parancsot is, ez kapcsolja be a tárgymutatót készítő parancsokat. Ezt még a dokumentum bevezető részében kell megtenned. A tárgymutatóban szereplő bejegyzéseket a dokumentumban elhelyezett

`\index{`*bejegyzés*`}`

parancsokkal adahatod meg. Amelyik oldalon az `\index` parancsot elhelyezed, annak az oldalnak a száma áll majd a bejegyzés után, célszerű közvetlenül (szóköz nélkül) az adott szó után írni, így semmiképpen sem kerülhet másik oldalra. A bejegyzések típusait az 5.5. táblázat tartalmazza.

Amikor a LaTeX egy `\index` parancsot talál, kiírja azt egy `.idx` kiterjesztésű fájlba. Amikor a kéziratot először lefordítod, a tárgymutató összes bejegyzése ebbe a fájlba kerül. Ezután kell lefuttatni ezen a fájlon a `makeindex` programot:

`makeindex` *fájlnév*

Ez sorba rendezi a bejegyzéseket, elkészíti a tárgymutatót LaTeX formátumban és kiírja azt egy `.ind` kiterjesztésű fájlba. Ha most a kéziratot újra lefordítjuk a LaTeX-hel, a program beszúrja az elkészített tárgymutatót arra a helyre, ahol

`\printindex`

parancsot talál.

Hasznos lehet még tárgymutató készítésekor a `showindex` csomag. Ez az adott oldalon szereplő összes bejegyzést kiírja az oldal margójára, jól megmutatva mi kerül a tárgymutatóba.

5.6. EPS grafikák beillesztése

Korábban már megismerkedtél vele hogyan lehet logikailag ábrákat úszó objektumként definiálni a `figure` környezet segítségével.

Természetesen arra is lehetőséget a LaTeX, hogy ezeket az ábrákat fizikailag is elkészítsd. Létezik egy `picture` környezet, amely ábrák készítésére szolgál, és számos csomag van, amely új parancsokat definiál ennek a feladatnak az elvégzésére. Ezeket azonban a legtöbb felhasználó bonyolultnak találja, így itt nem is foglalkozom vele. Megvizsgálom inkább hogyan lehetséges egy Encapsulated PostScript formátumú grafika vagy kép beillesztése a dokumentumba. A PostScript formátum a nyomdai előkészítésben elterjedt, számos szoftver (XFig, CorelDraw!, Gnuplot...) ismeri, ezért a LaTeX-et használók körében is a leggyakrabban használt.

D. P. Carlisle csomagja a `graphics` és ennek egy változata a `graphicx`, ezek nagyon egyszerűen használhatók és minden LaTeX rendszerben megtalálhatók, így ezeket mutatom be, pontosabban csak a `graphicx` nevűt. A `graphics` ezzel teljesen egyenértékű, csak annyiban különbözik társától, hogy másféle formában várják a paramétereket parancsai.

Amikor egy EPS grafikát a dokumentumodba illesztesz, valójában a grafika nem kerül bele a `.dvi` fájlba. Általánosan használt módszer, hogy a `.dvi` fájlt is PostScript fájllá alakítják, mivel mint említettem ez széles körben elterjedt, sok nyomtató ismeri. Az átalakításhoz leggyakrabban használt program a `dvips`, ez szintén része a legtöbb rendszernek. A PostScript formátumúvá alakítás során természetesen a beillesztett képek is bekerülnek a végső PostScript fájlba.

Legelső teendőnk a `graphicx` csomag betöltése a dokumentum bevezető részében megadott `\usepackage[`*meghajtó*`]{graphicx}` paranccsal, ahol a *meghajtó* a `.dvi`-ből PostScriptba korvertáló program neve. A lehetséges értékek: `dvips`, `xdvi`, `dvipdf`, `dvipsone`, `dviwindo`, `emtex`, `dviwin`, `pctexps`, `pctexwin`, `pctexhp`, `pctex32`, `truetex`, `tcidvi`, `oztex`, `textures`. Ha nem tudod melyiket kellene használnod, Unix rendszer esetén írj `dvips`-t! Lehetőség van új meghajtók deiniálására is, lásd a `graphics` csomag dokumentációját.

Ezek után már működik az

`\includegraphics[`*opció=érték, ...*`]{`*fájl*`}`

parancs, ami a képet beilleszti a dokumentumba. A parancsot természetesen egy figure környezeten belül érdemes használni. Az opcionális paraméterben vesszővel elválasztva több kapcsolót adhatsz meg. Ezekkel a kapcsolókkal módosíthatod a kép méreteit, elforgathatod a képet. Az 5.6. táblázat felsorolja a legfontosabb kapcsolókat. Fontos, hogy ezek a kapcsolók a megadott sorrendben hajtódnak végre. Az alábbi példában a `teszt.eps` fájlban lévő grafikát

`width`	A kép szélessége, nagyítás vagy kicsinyítés
`height`	A kép magassága, nagyítás vagy kicsinyítés
`angle`	A képet elfogatja az óramutató járásával megegyező irányba.

5.6. táblázat. A `graphicx` csomag kapcsolói

előbb elforgatjuk 90 fokkal, aztán nagyítjuk (vagy kicsinyítjük) akkorára, hogy szélessége 10 cm legyen.

```
\begin{figure}
\begin{center}
\includegraphics[angle=90, width=10cm]{teszt.eps}
\end{center}
\end{figure}
```

További példák találhatók a `graphicx` csomag dokumentációjában.

6. fejezet

Magyarítás

6.1. Ékezetes karakterek, írásjelek

6.1.1. Az `inputenc` csomag

A TEX-ben eredetileg külön parancsok szolgáltak az ékezetes betűk írására, mint például a `\'` vagy a `\"`.

Kényelmetlen lenne, ha egy hosszabb magyar szöveget úgy kellene begépelni, hogy minden egyes ékezetes betű helyére egy parancsot írunk, még ha ezek a parancsok rövidek is.

A LATEX `inputenc` csomagja lehetővé teszi, hogy az ékezetes karaktereket egyszerűen beírjuk a kéziratba. Természetesen ehhez szükséges, hogy az operációs rendszer és/vagy a használt szövegszerkesztő lehetővé tegye ékezetes karakterek bevitelét. Szerencsére ez szinte minden operációs rendszerben lehetséges.

A gond csak az, hogy a különböző típusú számítógépeken a különböző operációs rendszerek az ékezetes betűket nem azonos módon ábrázolják. Természetesen történtek kísérletek a kiosztás szabványosítására, ezek eredményként ma már rengeteg szabvány közül választhatunk. Szerencsére az `inputenc` csomag a legtöbb kódolást ismeri. A használni kívánt kódolás nevét paraméterként kell átadnod a csomagnak. A következő kódolásokat használhatod: `ascii`, `latin1`, `latin2`, `cp850`, `cp852`, `cp437`, `applemac`, `ansinew`, és még néhány kevésbé elterjedt kódolás.

Ha fogalmad sincs róla, hogy milyen kódolást használ az operációs rendszered, próbáld ki az imént felsoroltakat egy pár soros dokumentumon, amiben az összes ékezetes karakter szerepel. Amelyiknél helyes eredményt kapsz, azt használd!

Az `inputenc` csomagnak köszönhetően egy másik számítógépről kapott, más kódolású kéziratot nem szükséges a saját kódolásodra alakítanod, csak egyetlen szót, az `inputenc` csomagnak paraméterként átadott kódolásnevet kell átírnod. Esetenként az adott kézirat egy szerkesztőbe betöltve elég furcsán nézhet ki, ettől függetlenül – ha a kódolás helyesen van beállítva – a LATEX tudja kezelni.

6.1.2. Idézőjelek

Azt már tudod, hogy záró idézőjelet a `''` jelekkel írhatsz. Azt is tudod, hogy az angol formájú kezdő idézőjeleket a ``` jelek generálják. A magyar nyelvben azon-

ban a kezdő idézőjel nem fent, hanem lent van, formailag pedig az angol/magyar záró idézőjellel azonos. Ahhoz, hogy ilyen idézőjelet írhass a dokumentumodba, be kell töltened a `t1enc` csomagot. A csomag működésére nem térek ki, mert az mélyebb ismereteket igényelne, elég ha annyit tudsz, hogy ha magyar szöveget gépelsz, akkor feltétlenül töltsd be a `t1enc` csomagot. A csomagnak egyébként az alsó idézőjelek írásán kívül még egy magyarok számára fontos hatása van, erről bővebben a 6.3. szakaszban olvashatsz.

Ha a `t1enc` csomagot betöltötted, akkor idézőjelet két egymás után írt vesszővel generálhatsz.

6.2. A `babel` csomag

A LaTeX standard dokumentumosztályait az amerikai tipográfiai hagyományok szerint, angol nyelvű szövegek írásához tervezték.

Magában a TeX-ben megvan azonban a lehetőség több nyelv kezeléséhez, egymástól független elválasztási szabálygyűjteményeket képes kezelni.

A `babel` csomag megpróbálja kihasználni a TeX lehetőségeit és lehetővé tenni többnyelvű illetve nem angol nyelvű dokumentumok készítését. A `babel` nagyon sok nyelvet támogat, egyeseket hatékonyabban, másokat kevésbé. A magyar nyelv támogatása például a standard disztribúcióban levő `babel` csomagban csak annyit jelent, hogy a Tartalom, Tárgymutató, táblázat, ábra... szavak magyarul jelennek meg, valamint ha van a rendszerben használható magyar elválasztási szabálygyűjtemény, akkor betölti azt. Nagyon sok problémát azonban nem old meg a csomag. Például a tételek címénél a tétel szám hátul áll, hasonlóan igaz ez az ábrák, táblázatok feliratairа.

Az interneten elérhető egy `magyar.ldf` nevű fájl, amit eredeti disztribúció azonos nevű fájljának helyére másolva ezen hiányosságok nagy részét pótolhatjuk.

A csomag betöltésekor opcionális argumentumként kell megadni a dokumentumban használni kívánt nyelveket vesszővel elválasztva, utolsónak az alapértelmezett nyelvet. Ha például írásunk magyar nyelvű, de helyenként angol részeket is tartalmaz, ezt írhatjuk a `\begin{document}` parancs elé:

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

A `babel` csomag több lehetőséget is biztosít a nyelv váltására. Természetesen ha csak magyarul írz, akkor nincs szükséged ezekre a parancsokra.

A `\selectlanguage` parancs az argumentumaként megadott nyelvre kapcsol át. Akkor használandó, ha hosszabb szöveget kívánunk az adott nyelven írni, a dokumentum összes jellemzőjét – a táblázat-, ábrafeliratokat, a fejléceket, ... – az adott nyelvhez igazít. Más kérdés, hogy helyes-e az, hogy egy dokumentumon belül kétféle típusú fejlécet használsz.

Az `otherlanguage` környezet alapjaiban ugyanazt tudja, mint az előbb bemutatott `\selectlanguage`, természetesen csak a környezeten belül. Akkor használd, ha olyan nyelven írsz, ami jobbról balra ír. A kívánt nyelvet a `\begin{otherlanguage}` parancs paramétereként kell megadnod. A környezet csillagos változata csak a tipográfiai szabályokat változtatja meg, a 'táblázat', 'fejezet', ... szavakat nem.

A `\foreignlanguage` parancsnak két argumentuma van. A második argumentumként megadott szöveget az első argumentumként megadott nyelven

szerkeszti meg. Csak az ún. extra definíciókat és az elválasztási szabályokat változtatja meg, a 'táblázat', 'ábra', stb. szavakat és a dátum formáját nem. Hasznos, ha csak néhány szót írsz az adott nyelven.

A `\languagename` parancs mindig az aktuális nyelv nevét tartalmazza.

Olyan parancsokat is írhatsz, amelyek különbözően működnek az egyes nyelveken. Az `\iflanguage` parancsnak három argumentuma van. A második argumentumot hajtja végre a parancs, ha éppen az első argumentumként megadott nyelv az aktív, különben pedig a harmadikat.

Végül felsoroljuk a támogatott nyelveket, minden nyelvnél megadva a csomagnak szóló paramétert is: afrikai (`afrikaans`), bahasa (`bahasa`), breton (`breton`), katalán (`catalan`), horvát (`croatian`), cseh (`czech`), dán (`danish`), holland (dutch), angol (`english`, `USenglish`, `american`, `UKenglish`, `british`), eszperantó (`esperanto`), észt (`estonian`), finn (`finnish`), francia (`french`), galíciai (`galician`), német (`austrian`, `german`, `germanb`), görög (`greek`), magyar (`magyar`, `hungarian`), ír gaelic (`irish`), olasz (`italian`), alsószorbiai (`lowersorbian`), norvég (`norsk`, `nyorsk`), lengyel (`polish`), portugál (`portuges`, `portuguese`, `brazilian`, `brazil`), román (`romanian`), orosz (`russian`), skót gaelic (`scottish`), spanyol (`spanish`), szlovák (`slovak`), szlovén (`slovene`), svéd (`swedish`), török (`turkish`), felsőszorbiai (`uppersorbian`), walesi (`welsh`).

6.3. Elválasztás

A TEX több nyelv elválasztási szabályait képes betölteni, és köztük váltani. LATEX-ben ez a `babel` csomag nyelvváltó parancsai végzik.

A standard LATEX diszribúció sok nyelvhez tartalmaz elválasztási szabálygyűjteményt, de a magyar nyelvhez sajnos nem. A magyar elválasztási szabálygyűjteményt kézzel kell a telepíteni a LATEX rendszerbe.

A szabálygyűjtemény letölthető az internet számítógépes hálózatról, a fájl neve `huhyph.tex`, erre keressünk rá. A telepítés menete az adott diszribúciótól függ, ezért csak tippeket adunk. Keressük meg a többi nyelvhez tartozó szabálygyűjteményeket, ezek többsége '`.tex`' végű fájl, amelynek nevében szerepel a '`hyph`' szócska, ami az elválasztásra utal (hyphenation). Keressünk, egy `language.dat` nevű fájlt, ebben van megadva, hogy egy adott nyelvhez milyen nevű szabálygyűjtemény-fájl tartozik. Ha találunk ilyen fájlt, akkor írjuk bele – a többi nyelv mintájára – a magyar nyelvet is. Ezután már csak újra kell generálnunk a LATEX formátumfájlt, ennek menete erősen az adott disztribúciótól függ. Egyes disztribúciókban nincs `language.dat` fájl, a formátumfájl generálásakor kell megadni a használni kívánt nyelveket.

Ha magyar elválasztást használsz, akkor mindenképpen töltsd be a `t1enc` csomagot is. Ha ezt nem teszed meg a LATEX képtelen lesz elválasztani az ékezetes betűket, így pedig a magyar szabálygyűjtemény nem sokat ér.

6.4. Tárgymutató magyarul az Xindy programmal

Sajnos a `makeindex` program nem támogatja nem angol vagy német nyelvű tárgymutatók készítését. Az ékezetes karaktereket különleges jelekként kezeli, így ezek rendezésekor túl sok jóra nem számíthatunk. Esetleg használható

az `\index{almoskonyv@álmoskönyv}` alak, így rendezéskor az `almoskonyv`, a tárgymutatóban az `álmoskönyv` alakot kapjuk. A magyar nyelvű tárgymutatókban azonban rendezéskor az 'ö' betű nem egyezik az 'o' betűvel, és persze a kettősbetűs szavak rendezése is speciális.

A problémán az Xindy segít. Az Xindy a `makeindex`-hez hasonló tárgymutató-készítő program. Használata ingyenes, az internet hálózatról beszerezhető. Nagyon jól konfigurálható, gyakorlatilag az összes különleges tárgymutató elkészíthető a segítségével, tetszőleges nyelven. Magyar nyelvű tárgymutató készítéséhez egy saját Xindy stílusfájlt kell készítened, ez pedig nem is annyira triviális feladat. Ezért a A. függelékben megadom egy magyar tárgymutatókat készítő stílusfájl listáját. Ha ezt begépeled egy mondjuk `magyar.xdy` nevű fájlba, akkor az `\index` parancsba tetszőleges ékezetes betűket is írhatsz: `\index{álmoskönyv}`. A LaTeX által az első fordítás során kiírt `.idx` fájlon először a `tex2xindy`, majd az `xindy` programot kell futtatni, utóbbinak megadva a használni kívánt stílusfájlm, esetünkben a `magyar.xdy` nevét.

A pontos paraméterezést lásd az Xindy dokumentációjában.

A. Függelék

Magyar Xindy stílusfájl

```
(define-location-class "page-numbers" ("arabic-numbers"))
(define-attributes (("default")("usage")))

(markup-locref :open "\textbf{" :close "}" :attr "usage")

(define-crossref-class "see")
(markup-crossref-list :open "\see{" :close "}{}" :class "see")

(define-location-class-order("page-numbers" "see"))

(markup-index :open  "~n\begin{theindex}~n\raggedright~n"
              :close "~n\end{theindex}~n"
              :tree)

(markup-letter-group-list :sep "~n~n\indexspace~n")

(markup-locclass-list :open "\quad{}")
(markup-range :sep "\nolinebreak--\nolinebreak")
(markup-locref-list :sep ", ")

(markup-indexentry :open "~n  \item "             :depth 0)
(markup-indexentry :open "~n     \subitem "       :depth 1)
(markup-indexentry :open "~n        \subsubitem " :depth 2)

(markup-letter-group :open-head "~n\pagebreak[1]\textbf{"
       :close-head "}\nopagebreak" :capitalize)
(markup-letter-group :open-head "~n%" :close-head "" :group "default")

(require "tex/isolatin2m-tex.xdy")
(require "tex/isolatinm.xdy")
(require "rules/latin-tolower.xdy")

(use-rule-set :run 0
              :rule-set ("latin-tolower"))

(sort-rule "\" "")

(merge-rule "\\texttt *{(.*)}" "\1" :eregexp :again)
(merge-rule "\\textsc *{(.*)}" "\1" :eregexp :again)
(merge-rule "\\verb+[(^+)]*+" "\1" :eregexp :again)

(merge-rule "Á" "á")
(merge-rule "É" "é")
```

```
(merge-rule "Í" "í")
(merge-rule "Ő" "ó")
(merge-rule "Ö" "ö")
(merge-rule "Ő" "ő")
(merge-rule "Ű" "ú")
(merge-rule "Ü" "ü")
(merge-rule "Ű" "ű")

(define-letter-groups
  ("b" "c" "cs" "d" "dz" "dzs" "f" "g" "gy" "h" "j" "k"
   "ly" "m" "n" "ny" "p" "q" "r" "s" "sz" "t" "ty" "v" "w"
   "x" "y" "z" "zs"))

(define-letter-group "a, á" :prefixes ("a" "á") :before "b")
(define-letter-group "e, é" :prefixes ("e" "é") :before "f" :after "d")
(define-letter-group "i, í" :prefixes ("i" "í") :before "j" :after "h")
(define-letter-group "l"    :prefixes ("l")     :before "ly" :after "k")
(define-letter-group "o, ó" :prefixes ("o" "ó") :before "p" :after "ny")
(define-letter-group "ö, ő" :prefixes ("o~e")   :before "p" :after "o, ó")
(define-letter-group "u, ú" :prefixes ("u" "ú") :before "v" :after "ty")
(define-letter-group "ü, ű" :prefixes ("u~e")   :before "v" :after "u, ú")

(sort-rule "\" "")
(sort-rule "-" "")
(sort-rule " " "")

(sort-rule "gypsy" "gypsy")

(sort-rule "á" "a")
(sort-rule "é" "e")
(sort-rule "í" "i")
(sort-rule "ó" "o")
(sort-rule "ö" "o~e")
(sort-rule "ő" "o~e")
(sort-rule "ú" "u")
(sort-rule "ü" "u~e")
(sort-rule "ű" "u~e")

(sort-rule "ccs" "cscs")
(sort-rule "ggy" "gygy")
(sort-rule "lly" "lyly")
(sort-rule "nny" "nyny")
(sort-rule "ssz" "szsz")
(sort-rule "tty" "tyty")
(sort-rule "zzs" "zszs")
```

Név- és Tárgymutató

Irodalomjegyzék

[1] Donald E. Knuth. *The TEXbook*. Addison-Wesley, 1984.

[2] Leslie Lamport. *LATEX 2ε, The macro package for TEX*, 1994.

[3] Tobias Oetiker. *The Not So Short Introduction to LATEX 2ε*, 1996.