PERE LÁSZLÓ

Bevezetés a LaTeX használatába

Verzió: 1.0

PÉCS, 2002

PERE LÁSZLÓ:
Bevezetés a LaTeX használatába

Lektorálta: KRUZSLICZ FERENC
Tipográfia: PERE LÁSZLÓ

Ennek a könyvnek a létrejöttét a PÉCSI TUDOMÁNYEGYETEM Informatikai Integrációs Bizottsága által meghírdetett „Oktatási modul elkészítése” című pályázata tette lehetővé.

Ennek a könyvnek a készítésekor kizárólag szabad szoftverek kerültek felhasználásra. A szövegszerkesztés vim szövegszerkesztő program segítségével történt, a nyomdai előkészítést a LaTeX tördelőprogram végezte. A teljes munkamenet szabad felhasználású GNU/Linux operációs rendszer alatt zajlott.

Tartalomjegyzék

1. fejezet

Bevezetés

1.1. A használt jelölések

A könyvben a bemutatott Unix parancsokat, a LaTeX parancsait és környezeteit, az állományok és könyvtárak neveit `írógépbetűkkel` szedtük. Ezt a betűváltozatot használtuk a példákban is, ahol a LaTeX állományok részleteit mutatjuk be.

A parancsok bemutatásánál néhány helyen használjuk a *dőlt sans seriff* betűváltozatot is. Ilyen formával jelöltük a behelyettesíthető (nemterminális) részeket. A `\textit{`*szöveg*`}` például azt jelzi, hogy a `\textit` egy parancs, amelyet változatlan formában kell begépelnünk, míg a *szöveg* helyére mást is írhatunk.

A könyvben található sorainak eleje logikai felépítésük szerint van elrendezve. A sorok beljebb írása olyan szerkezetet mutat, amely reményeink szerint segíti az olvasót, hogy minél könnyebben igazodjon el a zárójelek, parancsok rendszerében. Ez a módszer a programozásban igen hasznosnak bizonyult, a LaTeX-el való munkát is nagymértékben megkönnyíti.

A parancsok megtalálhatóak a könyv végén található tárgymutatóban. Ide nem minden oldal száma került be, ahol az adott parancs megtalálható, csak azok a helyek szerepelnek, ahol konkrétan az adott parancsról van szó. Ha fellapozzuk a tárgymutatóban szereplő oldalt, akkor a külső margón széljegyzetként könnyedén megtalálhatjuk a parancsot. A széljegyzetek az eligazodást hivatottak segíteni.

1.2. Köszönetnyilvánítás

A szerző szeretne köszönetet mondani Bugya Titusznak, Dr. Hegyi Sándornak, Koniorczyk Mátyásnak és Kruzslicz Ferencnek a könyv megírásához nyújtott technikai segítségért és bíztatásért.

Külön köszönet illeti meg a Pécsi Tudományegyetem Informatikai Integrációs Bizottságát a könyv megírásához és a megjelenéséhez nyújtott anyagi támogatásért.

2. fejezet

Állománykezelés

Ebben a fejezetben a legalapvetőbb ismereteket vesszük sorra, amelyek szükségesek, hogy dokumentumokat hozzunk létre és nyomtassunk számítógép felhasználásával.

Meglehetősen vázlatos áttekintést tudunk csak adni ehelyütt, mégis megkíséreljük az ismeretek egy olyan részterületét kiválasztani, amelyek valamilyen értelemben teljesek. Célunk az, hogy a könyv melyet az olvasó a kezében tart „megálljon a saját lábán”.

Mindazonáltal szeretnénk felhívni a figyelmet arra, hogy az itt tárgyalt téma messze túlmutat e könyv keretein, mélyebb megértése mindenképpen ajánlható mindenki számára.

2.1. Alapfogalmak

A számítógépprogramok világában való eligazodás érdekében szokás a programokat hierarhikus rendbe sorolni. A programok világában alul foglal helyet a gépet, a számítógép elemeit vezérlő *operációs rendszer*, amely a legalapvetőbb feladatokat vállalja magára, legfelül pedig azok az *alkalmazások* foglalnak helyet, amelyek speciális célfeladatot látnak el.

operációs rendszer

alkalmazások

Az alkalmazások igen változatos feladatokat látnak el, igen sokfélék és bonyolultak. Példaként megemlíthetjük a nyomdai anyagok előállítására használt *tördelőprogram*okat. A tördelőprogram egy speciális célra készült, csak egy feladatkörben használható. Az ilyen programok használata mé-

tördelőprogram

lyebb ismereteket feltételez a témában. Nem elegendő tehát a programot, annak kezelését megismerni, ismernünk kell a szakterületet is.

Egészen más a helyzet az operációs rendszerekkel. Az operációs rendszer, mint a rendszer alsó szintje, nem specializált, nem kötődik egyetlen szakmához sem oly szorosan mint az alkalmazások. Mindenkinek, aki számítógéppel akarja munkáját végezni ismernie kell legalább néhány operációs rendszert, ha nem is mélységében, felhasználói szinten mindenképpen.

Azért kell megismernünk az általunk használt operációs rendszert, mert az alapvető adatkezelési lépéseket az operációs rendszer segítségével fogjuk elvégezni. Bármelyik szakterület alkalmazásait használjuk is, az operációs rendszer az a rendszer, amelynek támogatásával adatainkat tároljuk, másoljuk, nyilvántartjuk esetleg töröljük. Az operációs rendszer az a programrendszer, amely felvállalja az általános adatkezelési feladatokat.

állomány
fájl

Az *állomány* (más néven *fájl*) az adatoknak az az egysége, amelynek kezelését az operációs rendszer végzi. Állomány lehet egy szöveges dokumentum, egy kép, egy hangfelvétel, egy program, bármi ami a számítógép nézőpontjából adatként jelenhet meg. A lényeg számunkra az, hogy az operációs rendszer mindenképpen képes elvégezni az állományok kezelését, akkor is ha számára az állomány tartalma nem értelmezhető.

könyvtár

Annak érdekében, hogy a felhasználó a rengeteg állomány közt ne tévedjen el az állományokat az operációs rendszerek legtöbbje *könyvtár*akba rendezve tárolja. A könyvtárak az állományok logikus elrendezésének érdekében születtek, létrehozásuk célja kizárólag az, hogy ne legyen minden állomány egy helyen.

A könyvtárak faszerkezetbe rendezve találhatóak meg adattárainkon, így könnyedén megtalálható bármelyikük egy pontból, a fa gyökeréből indulva.

A felhasználónak egy ilyen rendszerben néhány alapvetően fontos ismeretet kell megszereznie annak érdekében, hogy munkájának elvégzéséhez igénybe tudja venni az operációs rendszer szolgáltatásait. Elsősorban el kell igazodnia az könyvtárak rendszerében, meg kell találnia az általa keresett állományokat. Képesnek kell továbbá lennie a megtalált állományt törölni, másolni, ellenkező esetben nem tudja munkáját elvégezni. A következő oldalakon ezeknek az alapvető műveleteknek az elvégzéséhez próbálunk meg segítséget adni. Szeretnénk felhívni a figyelmet arra, hogy az itt leírt ismeretek vázlatosak, inkább csak azoknak nyújtanak segítséget, akik meglévő ismereteiket szeretnék felfrissíteni. Az olvasó az irodalomjegyzékben található egyéb szakkönyvekből tájékozódhat részletesebben a témában.

2.2. A legfontosabb UNIX parancsok

Operációs rendszerek közül ehelyütt a GNU/Linux rendszert tárgyaljuk, vagyis a Linux állománykezelő parancsainak egy részét mutatjuk be. Tudnunk kell, hogy a Linux teljesíti a vonatkozó szabványokat, ezért az itt bemutatott parancsok általában más Unix rendszeren is változatlanul használhatóak. Az egyszerűség érdekében a hajlékonylemezes állománykezelést a `mtools` programcsomag segítségével mutatjuk be, amely nem része a Unix operációs rendszernek, de a legtöbb Unix rendszerre telepíthető.

`mtools`

Szeretnénk felhívni a figyelmet, hogy a könyvben tárgyalt LaTeX tördelőrendszer más operációs rendszerek alatt is futtatható, így nincs akadálya annak, hogy MS-DOS-t vagy Windows-t használva készítsünk LaTeX dokumentumokat. Nem tárgyaljuk ezen operációs rendszerek állománykezelő parancsait, hiszen ezek az ismeretek igen sok forrásból elérhetőek.

2.2.1. Unix parancsok

Unix rendszereken minden állomány és könyvtár rendelkezik névvel. A név segítségével az adott állomány vagy könyvtár egyértelműen azonosítható. Egy könyvtárban ezért nem lehet két azonos bejegyzés (könyvtár vagy állomány).

A Unix alapú rendszerek különbséget tesznek nagy és kisbetűk között. A `lev` név tehát nem egyezik meg a `Lev` névvel. Erre különösen kell ügyelnünk a munkánk során.

A `pwd` parancs segítségével megállapíthatjuk, hogy mi a *munkakönyvtár*unk, melyik könyvtárban „tartózkodunk” éppen.

`pwd`
munkakönyvtár

A `cd` parancs segítségével megváltoztathatjuk a munkakönyvtárunkat, vagyis „más könyvtárba léphetünk”. A parancs után – szóközzel elválasztva – az könyvtár nevét kell megadnunk ahová „be akarunk lépni”.

`cd`

A könyvtárak azonosításakor használhatjuk az *abszolút könyvtárleíró*t vagy a *relatív könyvtárleíró*t. Az abszolút könyvtárleíró megadja milyen úton lehet eljutni az adott könyvtárba a gyökérkönyvtárból, a relatív könyvtárleíró pedig megadja, hogyan lehet eljutni a munkakönyvtárból a könyvtárba. Mivel Unix rendszereken a *gyökérkönyvtár* neve a `/` jel, az abszolút könyvtárleíró mindig ezzel a jellel kezdődik.

abszolút könyvtárleíró
relatív könyvtárleíró
gyökérkönyvtár
`/`

A relatív és abszolút könyvtárleíró a könyvtár felkeresésénél bejárt könyvtárakat írja le, a könyvtárneveket `/` jellel elválasztva. Ebben a könyvben a relatív és abszolút könyvtárleírókat mindig `/` jellel fejezzük be, bár ez nem volna kötelező.

A /home/joe/ könyvtárleíró például egy abszolút könyvtárleíró – hiszen a / jellel kezdődik. Ez a könyvtárleíró a joe alkönyvtárra vonatkozik, amely a gyökérkönyvtárból nyíló home nevű könyvtárban található.

A home/joe/ könyvtárleíró egy relatív könyvtárleíró – hiszen nem / jellel kezdődik – amely a munkakönyvtárban található home könyvtárban található joe könyvtárra vonatkozik. Ha például a /usr/ könyvtár a munkakönyvtárunk, akkor ez a relatív könyvtárleíró a /usr/home/joe/ abszolút könyvtárleíróval is jelölhető volna. Ha a munkakönyvtárunk a /, akkor a home/joe/ relatív könyvtárleíró a /home/joe/ abszolút könyvtárleíróval is leírható. Látható, hogy relatív könyvtárleírót használva a hivatkozott könyvtár mindig a munkakönyvtárhoz képest értendő, innen a „relatív” jelző.

Másképpen felfogva a relatív és abszolút könyvtárleírót azt is mondhatnánk, hogy:

- a munkakönyvtár egy tárolóhely, amelyben mindig található egy könyvtár abszolút könyvtárleírója
- ha az általunk használt könyvtárleíró első betűje nem a / jel, akkor a rendszer a munkakönyvtárként használt könyvtárleírót elé másolja, így kapja a pontos helyet leíró abszolút könyvtárleírót

A következő példa a munkakönyvtár lekérdezését és a munkakönyvtárban található latexfiles könyvtárba való belépést mutatja:

```
# pwd
/root
# cd latexfiles/
# pwd
/root/latexfiles
#
```

parancskérő jel

A példában szereplő # jel a *parancskérő jel*, amelynek a képernyőre írásával a rendszer azt jelzi, hogy kész a parancsok fogadására. A mai Unix rendszerekben ennél sokkal informatívabb parancskérő jeleket használnak, mi az egyszerűség kedvéért használjuk ezt a hagyományos jelet.

cd
saját könyvtár

A cd parancs önmagában kiadva a felhasználót a *saját könyvtár*ába viszi. A felhasználó saját könyvtára az a könyvtár, amely a felhasználó könyvtárait és állományait tartalmazza. A felhasználónak általában csak ebben a könyvtárban van joga könyvtárakat és állományokat létrehozni.

Az állományokat és könyvtárakat a képernyőre kiíratni az `ls` paranccsal lehet. Az `ls` parancs `-l` kapcsolója részletes listát eredményez, ahogyan az a következő példából is látszik: `ls`

```
# ls -l
Összesen: 12296
-rw-r--r--  1 root  root   49999 FEB  9 12:00 cupbook.sty
drwxr-xr-x  2 root  root    4096 JAN  3 15:50 english
-rw-r--r--  1 root  root     683 FEB  9 11:57 etikett.tex
#
```

A kapott lista első betűje jelzi, hogy könyvtárról vagy állományról van-e szó. Könyvtár esetén itt egy `d` betűt látunk, állomány esetén pedig egy `-` jelet. A példa egy könyvtárat és két állományt mutat.

A `tree` parancs segítségével a könyvtárszerkezetet kirajzolhatjuk a képernyőre. Ez igen hasznos parancs, mivel a kezdő felhasználó számára is átlátható formában mutatja be a könyvtárak rendszerét. Fel kell hívnunk azonban a figyelmet arra, hogy sok Unix rendszer nem tartalmazza. A `tree` egy olyan segédprogram, amelyet a legtöbb Linux rendszer biztosít, de nem „hivatalos” része a rendszernek. A következő példa az `english` könyvtár felépítését rajzolja a képernyőre: `tree`

```
# tree english/
english/
|-- english.tex
|-- hungarian.tex
|-- pipbook.sty
|-- pipbook.tex -> pipbook.sty
|-- szavak.sorted
‘-- tmp
    ‘-- pipas
#
```

A példában szereplő `english` könyvtár tartalmaz egy `tmp` nevű könyvtárat, amelyben egy `pipas` nevű állomány található. Láthatjuk, hogy a `tree` parancs által kirajzolt szerkezetben nem könnyű megkülönböztetni a könyvtárakat és az állományokat. Valójában a legtöbb Linux rendszeren mind az `ls` mind pedig a `tree` által kiírt lista színes, könnyen átlátható. E könyvben az nyomdai költségek csökkentésének érdekében a listák egy színnel jelennek meg.

mkdir

Az mkdir parancs segítségével hozhatunk létre könyvtárakat. A parancs után – szóközzel elválasztva – a létrehozni kívánt könyvtár relatív vagy abszolút könyvtárleíróját kell megadni. (A parancs használatakor ne feledkezzünk megg arról, hogy csak a saját könyvtárunkban hozhatunk létre könyvtárakat és állományokat.) A következő példa során a felhasználó először a saját könyvtárába lép, majd ott egy könyvtárat hoz létre latex néven, majd belép ebbe a könyvtárba:

```
# cd
# mkdir latex
# cd latex
#
```

touch

A touch parancs segítségével állományokat hozhatunk létre. A következő példa során a felhasználó az imént létrehozott könyvtárban egy állományt hoz létre majd kiíratja annak nevét és egyéb adatait:

```
# touch elso.tex
# ls -l elso.tex
-rw-r--r--  1 root     root      0 JÚN 10 12:03 elso.tex
#
```

rmdir

Az rmdir parancs segítségével törölhetünk üres könyvtárakat. Állomá-

rm

nyokat az rm parancs segítségével törölhetünk. Ügyelnünk kell, mert általában nincs mód arra, hogy a törölt állományokat helyreállítsuk.

cp

Állományokat másolni Unix alatt a cp parancs segítségével lehet. A parancsnak meg kell adnunk, hogy mit akarunk másolni és azt hogy hova. A következő példa bemutatja, hogy a munkakönyvtárban található latex könyvtárból hogyan másoljuk át az elso.tex állományt a munkakönyvtárban található regi könyvtárba.

```
# cp latex/elso.tex regi/
#
```

2.2.2. A hajlékonylemez kezelése

hajlékonylemez

A Unix rendszerek segítséget adnak a *hajlékonylemez* kezeléséhez. Fontos, hogy képesek legyünk a hajlékonylemez kezelésére, mert az adatainkat így képesek leszünk arhiválni, és más gépekre átmásolni akkor is, ha a számítógépek nincsenek számítógéphálózatba kötve.

Szeretnénk felhívni a figyelmünket, hogy az adatokat *mindig* több helyen is őriznünk kell. Szerencsés esetben a környezetünkben olyan tehnikai háttér áll a rendelkezésünkre, amely vállalja az adatok mentését, arhiválását. Ilyenkor a helyi számítógépes szakemberek rendszeresen készítenek biztonsági másolatot adatainkról és egy esetleges meghibásodás esetén az adatokat a arhívumból helyreállítják. Ahol ilyen szolgáltatásra nem számíthatunk – pl. az otthoni számítógépünkön – ott magunknak kell gondoskodnunk az arhiválásról. Ez kisebb adatmennyiség esetén megoldható hajlékonylemezre másolással. Ilyenkor az általunk létrehozott dokumentumokat rendszeresen hajlékonylemezre másoljuk és biztos helyen elzárjuk, hogy az esetleges meghibásodáskor ne vesszen el munkánk eredménye.

E könyvben nem a Unix rendszerek hajlékonylemez-kezelését mutatjuk be, mert az meglehetősen sok helyet venne igénybe, részletes magyarázatot igényelne. Bemutatjuk viszont az `mtools` programcsomag néhány parancsát. Ezen parancsok segítségével DOS formátumú hajlékonylemezeket használhatunk olyan parancsokkal, ahogyan azt a DOS alatt sokan már megszokhatták. `mtools`

A hajlékonylemez formázását az `mformat` paranccsal végezhetjük el. E formázás során a hajlékonylemezen található állományok és könyvtárak törlődnek, az adatok elvesznek. Ma már a kereskedelemben kapható hajlékonylemezek formázva vannak, néha azonban ennek ellenére szükségünk lehet a formázásra. A következő példa a hajlékonylemez formázását mutatja be: `mformat`

```
# mformat a:
#
```

Amint látjuk az `mformat` számára az `a:` jelenti a hajlékonylemezt. Tudnunk kell, hogy a Unix rendszerek nem használják az `a:` jelölést, de az `mtools` minden segédprogramja elfogadja, azért, hogy a DOS alatt megszokott formát használhassuk. `a:`

Az `mcopy` parancs segítségével másolhatunk állományokat a hajlékonylemezre vagy tölthetünk onnan vissza adatokat a saját könyvtárunkba. A következő példa egy állományt másol a hajlékonylemezre: `mcopy`

```
# mcopy elso.tex a:
#
```

Az `mdir` parancs segítségével írathatjuk ki a hajlékonylemezen található állományok és könyvtárak nevét a képernyőre. A következő parancs a hajlékonylemezen található `elso.tex` állomány nevét írja a képernyőre: `mdir`

```
# mdir a:
 Volume in drive A has no label
 Volume Serial Number is 7BFD-835F
Directory for A:/

elso     tex        25 06-10-2002  12:31  elso.tex
        1 file                   25 bytes
                          1 457 152 bytes free

#
```

2.2.3. Összefoglalás

Az állománykezelésről szóló fejezet végén bemutatunk néhány gyakorlati példát, összefoglaljuk a legfontosabb parancsokat.

Az elsői, gyakorlatban is használható példa során a következő lépéseket végezzük el:

1. A saját könyvtárunkba lépünk, mivel ez az a hely, ahol könyvtárakat és állományokat hozhatunk létre.
2. Létrehozunk egy könyvtárat, amely egy bizonyos munkát fog tartalmazni. Tesszük ezt azért, hogy az egyes munkáinkat alkotó állományok ne keveredjenek össze.
3. Belépünk ebbe a könyvtárba.
4. Ebben a könyvtárban létrehozunk egy állományt, amely a dokumentumot tartalmazza.
5. Elindítjuk a szövegszerkesztőt amellyel az állomány módosítjuk, tartalmát megírjuk. (Ennek a szövegszerkesztőnek a használatáról a 23. oldalon olvashatunk bővebben.)

```
# cd
# mkdir latex
# cd latex
# touch elso.tex
# vim elso.tex
```

A következő példa során bemutatjuk a hajlékonylemezre való mentést. Az előző példa során elkészített állományt mentjük a hajlékonylemezre. E során a munka során a következő lépéseket hajtjuk végre:

1. Formázzuk a hajlékonylemezt. Ezalatt törlődik minden adat, amely eredetileg a hajlékonylemezen volt.
2. Másoljuk az állományt a hajlékonylemezre.
3. A hajlékonylemezen található állományokról kérünk egylistát, hogy meggyőződjünk róla, az állomány a lemezre került.

```
# mformat a:
# mcopy elso.tex a:
# mdir a:
 Volume in drive A has no label
 Volume Serial Number is 4FD7-AD15
Directory for A:/

elso     tex        27 06-10-2002  12:47  elso.tex
        1 file                   27 bytes
                          1 457 152 bytes free

#
```

A következő példa során a hajlékonylemezről való visszatöltést vesszük sorra. A munka során a következő feladatokat végezzük el:

1. A saját könyvtárunkba lépünk, mert itt tudunk könyvtárakat és állományokat létrehozni.
2. Készítünk egy könyvtárat, amelybe vissza fogjuk tölteni az állományt.
3. Listázzuk a hajlékonylemezen található állományokat, hogy meggyőződjünk a keresett állomány a hajlékonylemezen van-e.
4. Az állományt visszatöltjük a hajlékonylemezről az imént létrehozott könyvtárba.
5. Kiíratjuk a képernyőre a visszatöltésre használt könyvtár tartalmát, hogy lássuk sikeres volt-e az állomány visszatöltése

Parancs	*Leírás*
`cd`	munkakönyvtár megváltoztatása
`cp`	állományok másolása
`exit`	kilépés
`halt`	számítógép leállítása
`ls`	állományok, könyvtárak neveinek kiíratása
`man`	a kézikönyv megjelenítése
`mcopy`	másolás hajlékonylemezre, hajlékonylemezről
`mdel`	állomány törlése hajlékonylemezről
`mdir`	hajlékonylemez tartalmának kiíratása
`mformat`	hajlékonylemez formázása
`mkdir`	könyvtár létrehozása
`mmd`	könyvtár létrehozása hajlékonylemezen
`passwd`	felhasználói jelszó megváltoztatása
`pwd`	munkakönyvtár kiíratása
`rm`	állomány törlése
`rmdir`	könyvtár törlése
`touch`	állomány létrehozása
`tree`	könyvtárszerkezet kirajzolása

2.1. táblázat. Néhány Linux parancs

```
# cd
# mkdir visszatoltott
# mdir a:
 Volume in drive A has no label
 Volume Serial Number is 4FD7-AD15
Directory for A:/

elso     tex        27 06-10-2002  12:47  elso.tex
        1 file                  27 bytes
                          1 457 152 bytes free
# mcopy a:/elso.tex visszatoltott/
# ls -l visszatoltott/
Összesen: 4
-rw-r--r--  1 root     root     27 JÚN 10 12:52 elso.tex
#
```

Az állománykezelésben a 2.1 táblázat által bemutatott parancsok hasznos segítséget nyújthatnak. Szeretnénk még egyszer felhívni a figyelmet, hogy az itt tárgyalt parancsok nem mindegyike része az összes Unix rendszernek, néhányat esetleg külön kell telepíteni.

2.3. Szöveges állományok

Az igény, hogy szöveges üzeneteket a betűk kódolásával reprezentáljunk, jóval régebbi a a számítástechnikánál. Kezdetben szöveges üzenetek átvitelére készültek ún. kódtáblázatok, amelyek alkalmasak voltak a betűk és írásjelek kódjaikkal való ábrázolására. Igen sokan ismerik pl. az ún. morze kódot, amely hosszú és rövid jeleket valamint hosszú és rövid szüneteket használ az átvitt szöveg kódolására.

A technika fejlődésével megszülettek azok az elektromehanikus szerkezetek, amelyek a kódolást és dekódolást, az átalakítást és a visszaalakítást automatizálták. Az ilyen eszközökkel már írógépszerű szerkezeteket lehetett kialakítani, amelyek használata egyszerűbb volt a morze jeleknél, nem kellett speciális tudás, évekig tartó gyakorlás. Az ilyen „távgépíró” eszközök elterjedésével megszületett az igény a szabványos kódrendszerek kialakítására, hogy a készülékek képesek legyenek egymással tartani a kapcsolatot.

ASCII

Az elektromehanikus szerkezetek elterjedésével született meg az *ASCII* szabvány (*American Standard Code for Information Interchange*), a hatvanas évek elején. Ezt a szabványt a számítástechnika is átvette, ma is elterjedten használatos.

bit
binary digit

Kezdetben az ASCII szabvány hét *bit*en (*binary digit*), bináris helyiértéken kódolt minden egyes betűt, írásjelet speciális kódot. A kettes számrendszer hét helyiértéken összesen 128 kombinációt képes kódolni, az ASCII szabvány tehát 128 jel megkülönböztetésére képes. Ebben a 128 jelet tartalmazó kódtáblázatban az angol nyelv kis és nagybetűi, a tízes számrendszer számjegyei, az írásjelek és néhány speciális jel kapott helyet az éveken át tartó fejlesztés során.

Ahogyan az ASCII kódtáblázatot egyre több helyen használni kezdték, igény jelentkezett arra, hogy az angol nyelvben nem szereplő, de egyes nemzetek nyelvében megtalálható speciális – pl. ékezettel ellátott – jeleket is kezelni lehessen. Szerencsére a számítástechnikában ekkorra elterjedt a nyolc bites kódolás, így a kódtáblázatot ki lehetett terjeszteni 256 jelre. Nyolc biten ugyanis 256 féle változat hozható létre.

A nyolc bitet használó ún. kiterjesztett ASCII már tartalmazza pl. a francia nyelv összes kis- és nagybetűjét. Sajnos a magyar nyelv speciális, két hosszú ékezetet tartalmazó jelei (őŐűŰ) nem kaptak helyet a kiterjesztett ASCII kódtáblázatban sem, mivel azokat csak a magyar nyelv használja.

Sajnos a kiterjesztett ASCII korántsem olyan szigorúan betartott szabvány mint a hét bites elődje. Az egyes gyártók sokszor más-más kódtáblát vezettek be, ezért az általuk kifejlesztett eszközök sokszor ma sem cseresz abatosak egymással.

Igen érdekesen alakult a sorok végének jelzésére használt kódok sorsa. A kezdeti elektromehanikus szerkezetek másodpercenként 30 jel átvitelére voltak alkalmasak, az írást végző mehanikus szerkezetnek – kocsinak – azonban $\frac{1}{15}$ másodpercre volt szüksége ahhoz, hogy a sor elejére fusson. Az ASCII kódtábla úgy oldotta fel ezt a problémát, hogy a sorok végének jelzésére két jelet használt, amelynek az átviteléhez $\frac{1}{30} + \frac{1}{30} = \frac{1}{15}$ másodpercre volt szükség. Később – amikor a jelek fogadását már nem mehanikusan végezték el – feleslegessé vált a sor végének ez a kettős kódolása. Érdekes módon megjelentek a piacon olyan számítógépes rendszerek, amelyek ma is két jelet használnak a kódolásra, olyanok, amelyek a két jelből az elsőt választották és olyanok is, amelyek a másodikat. Ma pl. a DOS-t futtató számítógépeken két jel szolgál a sor végének jelzésére, a UNIX lapú rendszerek a 10-es kódot, az Apple számítógépei pedig a 13-as kódot használják a sorok végének jelzésére.

Mivel a kiterjesztett ASCII sem adott teljes megoldást az írásjelek kódolására, születtek egyéb kódtáblázatok, szabványok, amelyek több kódlapot, esetleg a jelek kódolására több 8 bites egységet használva tettek kísérletet a teljes megoldás megalkotására. Bármelyik rendszert használják azonban a programjaink, tudnunk kell, hogy ezek legtöbbje az ASCII kódlapra épül, annak kiterjesztett, bővített változatai.

UNIX alapú rendszerek általában az ISO (*International Organization for Standardization*) által alkotott szabványokat használják a betűk és jelek kódolására. Az iso8859 szabvány írja le a betűk kódolását, ezen belül pedig az iso8859-2 szolgál a kelet európai nyelvek kódolására. Az iso8859-1 a nyugat európai nyelvek kezelését teszi lehetővé.

Fontos lehet tudnunk, hogy a UNIX alapú rendszerek a szöveges állományok, ASCII állományok kezelésére igen sok eszközt biztosítanak. A UNIX fejlesztésének egyik alapgondolata az volt, hogy minden információt lehet ASCII állományba kódolni és mivel ezek kezelése viszonylag egyszerű mindent érdemes is így tárolni.

A0	A1 ¡	A2 ¢	A3 £	A4 ¤	A5 ¥	A6 ¦	A7 §	A8 ¨	A9 ©	AA ª	AB «	AC ¬	AD -	AE ®	AF ¯
B0 °	B1 ±	B2 ²	B3 ³	B4 ´	B5 µ	B6 ¶	B7 ·	B8 ¸	B9 ¹	BA º	BB »	BC ¼	BD ½	BE ¾	BF ¿
C0 À	C1 Á	C2 Â	C3 Ã	C4 Ä	C5 Å	C6 Æ	C7 Ç	C8 È	C9 É	CA Ê	CB Ë	CC Ì	CD Í	CE Î	CF Ï
D0 Ð	D1 Ñ	D2 Ò	D3 Ó	D4 Ô	D5 Õ	D6 Ö	D7 ×	D8 Ø	D9 Ù	DA Ú	DB Û	DC Ü	DD Ý	DE Þ	DF ß
E0 à	E1 á	E2 â	E3 ã	E4 ä	E5 å	E6 æ	E7 ç	E8 è	E9 é	EA ê	EB ë	EC ì	ED í	EE î	EF ï
F0 ð	F1 ñ	F2 ò	F3 ó	F4 ô	F5 õ	F6 ö	F7 ÷	F8 ø	F9 ù	FA ú	FB û	FC ü	FD ý	FE þ	FF ÿ

2.2. táblázat. Az iso8859-1 kódlap

A0	A1 Ą	A2 ˘	A3 Ł	A4 ¤	A5 Ľ	A6 Ś	A7 §	A8 ¨	A9 Š	AA Ş	AB Ť	AC Ź	AD -	AE Ž	AF Ż
B0 °	B1 ą	B2 ˛	B3 ł	B4 ´	B5 ľ	B6 ś	B7 ˇ	B8 ¸	B9 š	BA ş	BB ť	BC ź	BD ˝	BE ž	BF ż
C0 Ŕ	C1 Á	C2 Â	C3 Ă	C4 Ä	C5 Ĺ	C6 Ć	C7 Ç	C8 Č	C9 É	CA Ę	CB Ë	CC Ě	CD Í	CE Î	CF Ď
D0 Đ	D1 Ń	D2 Ň	D3 Ó	D4 Ô	D5 Ő	D6 Ö	D7 ×	D8 Ř	D9 Ů	DA Ú	DB Ű	DC Ü	DD Ý	DE Ţ	DF ß
E0 ŕ	E1 á	E2 â	E3 ă	E4 ä	E5 ĺ	E6 ć	E7 ç	E8 č	E9 é	EA ę	EB ë	EC ě	ED í	EE î	EF ď
F0 đ	F1 ń	F2 ň	F3 ó	F4 ô	F5 ő	F6 ö	F7 ÷	F8 ř	F9 ů	FA ú	FB ű	FC ü	FD ý	FE ţ	FF ˙

2.3. táblázat. Az iso8859-2 kódlap

3. fejezet

Szövegszerkesztés

3.1. Alapfogalmak

A *szövegszerkesztés*en (*text editing*) ASCII állományok szerkesztését értjük, vagyis olyan folyamatot, amely során betűkből, számjegyekből, írásjelekből és speciális jelekből álló állományokat változtatjuk.

szövegszerkesztés
text editing

Fontos tudnunk, hogy a magyar szakirodalomban szövegszerkesztő programnak szokás nevezni azokat a programokat is, amelyekkel a szöveg megjelenési formája is módosítható. Ezeket a korlátozott tipográfiai eszközökkel felszerelt programokat az angol szakirodalom *word processor* kifejezéssel jelöli, magyarul pedig talán a irodai programcsomag (kiadványszerkesztő?) kifejezés a legszerencsésebb. Mi az ilyen programokkal ehelyütt nem foglalkozunk.

word processor

3.2. A vi használata

A `vi` szövegszerkesztő minden Unix rendszeren elérhető, de vannak más operációs rendszeren futtatható változatai is. Valójában elmondható, hogy szinte minden operációs rendszerre létezik `vi` szövegszerkesztő, ezért ha megtanuljuk használni bármilyen rendszeren el tudjuk végezni a feladatainkat. Unix rendszerek alatt a `vi` ismerete szinte kötelező.

`vi`

A legtöbb Linux rendszeren a `vi` továbbfejlesztett változata, a `vim` van telepítve. A `vim` indítható a `vi` paranccsal is, ilyenkor csak az „alap” szol-

`vim`

Parancs	*Jelentés*
`w`	az szerkesztett állomány mentése eredeti néven
`w!`	az állomány erőltetett mentése
`w nev`	az állomány mentése a `nev` néven
`n,m w nev`	sorok mentése `nev` néven
`r nev`	a `nev` betöltése kurzorhoz
`q`	kilépés
`q!`	kilépés mentés nélkül
`wq`	az állomány mentése és kilépés
`/minta`	minta keresése a szerkesztett szövegben

3.1. táblázat. A vi legfontosabb parancsai

gáltatásokat nyújtja, vagyis úgy működik, ahogyan minden `vi` működik bolygónkon.

A parancs kiadásakor megadhatjuk a szerkeszteni kívánt állomány nevét a parancs után:

```
# vim elso.tex
```

kilépés a vi-ból

A *kilépés a vi-ból* sokak számára problémát jelent, ezért a szövegszerkesztésről szóló fejezet elején ejtünk erről néhány szót!

Ha ki akarunk lépni a szövegszerkesztőből a módosítások mentésével, akkor járjunk el a következőképpen:

1. Nyomjuk le kétszer az Esc billentyűt!
2. Nyomjuk le a : billentyűt. Ekkor egy kettőspont jelenik meg a képernyő legalsó sorában!
3. Nyomjuk le a w q billentyűket (ekkor megjelennek a `wq` betűk a képernyő alsó sorában), majd nyomjuk le az Enter billentyűt!

Ha a változtatások mentése nélkül szeretnénk kilépni a szövegszerkesztőből, akkor a w q billentyűk lenyomása helyett használjuk a q ! billentyűket a kilépéshez, egyébként járjunk el a már bemutatott módon. A `q!` hatására a szövegszerkesztő mentés nélkül lép ki, a változtatások amelyeket a szöveges állományon végeztünk, elvesznek.

Billentyű	*Jelentés*
h és l	balra és jobbra 1 betűt
j és k	lefelé és felfelé 1 sort
0 és $	sor elejére és végére
g g és Shift+g	állomány elejére és végére
w és b	szavanként előre és vissza
{ és }	bekezdés elejére és végére
x	törlés betűnként előre
d h	törlés betűnként vissza
r	betű felülírása
d d	sor törlése
d w és d b	törlés szó végéig és elejéig
d g g és d Shift+g	törlés állomány elejéig és végéig
v	kijelölés üzemmód
V	kijelölés soronként
Ctrl+v	téglalap alakú kijelölés
x	kivágás
y	másolás
p	beillesztés

3.2. táblázat. A vi legfontosabb billentyűparancsai

3.2.1. A vi állapotai

A `vi` működésének megértéséhez elengedhetetlenül fontos, hogy megismerkedjünk a *vi állapotai*val. A `vi` egyes billentyűk lenyomására attól függő módon reagál, hogy éppen milyen állapotban van. Ez kissé meg szokta zavarni azokat a felhasználókat, akik még nem ismerik ezt a programot. *vi állapotai*

A következő felsorolás a `vi` állapotait mutatja be, valamint azt, hogyan tudunk az állapotok közt váltani:

normál mód : Normál módban a `vi` minden lenyomott billentyűnek különleges jelentőséget tulajdonít, minden lenyomott billentyű valamilyen komplex jelentéssel bír.

Normál módban a programmal nem lehet gépelni, a `vi` nem „írógépszerűen” viselkedik. Ha pl. lenyomjuk a k billentyűt, akkor nem je-

lenik meg a k betű a szövegben, hanem egy sorral feljebb lép a kurzor a képernyőn.

A vi indítás után normál módba van. Beszúró módból az Esc lenyomásával, parancs módból az Esc kétszeri lenyomásával léphetünk normál módba.

beszúró mód : Beszúró módban a vi szöveg írására használható. Ebben az üzemmódban a program úgy működik mint egy írógép, a lenyomott billentyűnek megfelelő betű bekerül a szövegbe a kurzor helyére.

Beszúró módba a normál módból az I betű lenyomásával juthatunk.

parancs mód : Parancs módban a programnak parancsokat adhatunk. A parancsok a képernyő alsó sorában jelennek meg, a vi az Enter lenyomása után hajtja végre őket.

Parancs módba normál módból a : billentyű lenyomásával juthatunk. Az Enter lenyomásakor – a parancs végrehajtása után – a program újra normál módba kerül. Ha az Esc billentyűt kétszer nyomjuk le parancs módban, a vi a parancs végrehajtása nélkül lép vissza normál módba.

A felhasználó számára igen fontosak a vi parancsai, hiszen ezek adnak lehetőséget az állomány mentésére és a kilépésre. A 3.1 táblázat foglalja össze a program legfontosabb parancsait. Szeretnénk felhívni a figyelmet arra, hogy itt csak a legfontosabb parancsokat soroltuk fel.

A vi normál módban használható billentyűkombinációit a 3.2 táblázatban foglaltuk össze. Szeretnénk felhívni a figyelmet arra, hogy ezek a táblázatok csak a legfontosabb, a használathoz elengedhetetlenül szükséges eszközöket mutatják be.

4. fejezet

A tipográfia alapjai

A tipográfia az írott (nyomtatott) szöveg előállításának művészete. A tipográfusok művészi szinten ismerik a szövegeket, ábrákat, képeket tartalmazó nyomdai termékek megjelenésével szemben támasztott szakmai, esztétikai és egyéb követelményeket.

Fontos tudnunk, hogy tipográfia már a számítógépek előtt is létezett, sőt azt kell mondanunk, hogy a tipográfia szabályrendszerét jóval a számítógépek megjelenése előtt kidolgozták és használták. Számítógépes nyomdai eszközöket használva a feladatunk csupán annyi, hogy a nyomdai termékeket olyan formában állítsuk elő, hogy azonkon ne lehessen észrevenni, hogy számítógéppel készült.

Nagyon nehéz összefoglalni, hogy mit is értünk nyomdai előkészítésen, mit csinál egy tipográfus. Ha nagyon általánosan kívánjuk megfogalmazni a feladatkörét, akkor azt mondhatjuk, hogy a szerző mondandója és stílusa, a téma és a terjedelem, az olvasóközönség és a használt nyomdai technológia alapján meghatározza, hogy hova kerüljenek a betűk, milyen legyen azok alakja és elrendezése.

Nyilvánvaló, hogy a tipográfusnak igen sokrétű ismeretekkel kell rendelkeznie, komoly stílusérzéket kell kifejlesztenie, valamint igen jól kell ismernie a nyomdatechnika korlátait és szabályait. Ma már bárki végezhet tipográfiai munkát akinek van egy számítógépe, de soha ne feledjük el, hogy a tipográfia művészet. Tartsuk szem előtt, hogy zenei programot is vásárolhat bárki, de attól még senki nem lesz zeneszerző, ha telepít a számítógépére egy programot.

Mértékegység	*SI*	*Eredet*
inch (Angol)	25,40 mm	
inch (Francia)	27,07 mm	
pont (Didot)	0.376 mm	1/72 Francia inch
pont (Angol)	0.3514598 mm	1/72,27 Angol inch
pont (TeX)	0.3514598035 mm	1/72.27 Angol inch
pont (Postscript)	0.3527777778 mm	1/72 Angol inch
pica (Angol)	4.2175176 mm	12 pont (Angol)
pica (TeX)	4.217517642 mm	12 pont (TeX)
pica (Postscript)	4.233333333 mm	12 pont (Postscript)
cicero	4.531 mm	12 pont (Didot)

4.1. ábra. Tipográfiai mértékegységek

4.1. Mértékegységrendszerek

A tipográfiában kétféle mértékegységrendszer terjedt el. Az Didot rendszert elterjedten használják Európában, míg Angliában és az Egyesült Államokban az angolszász mértékrendszert.

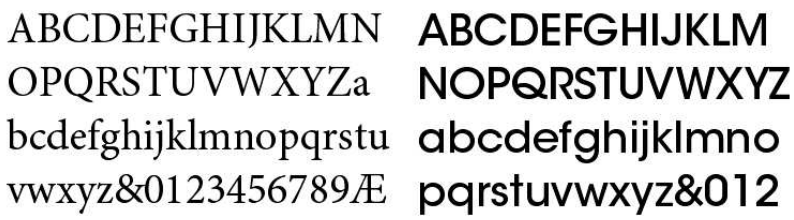
Mindkét mértékrendszer alapja a *pont*, de a két mértékegységrendszerben használt pont más-más távolságmértéket jelent. A Didot pont a Franciaországban használatos francia Inch 1/72-ed része, annak értéke 0,376 mm.

Valamivel bonyolultabb a helyzet az angolszász ponttal. (A francia és az angol mértékrendszer közti konfliktus legendás) Eredetileg az angol pont 1/72,27 Angol inch volt, de az egyszerűség kedvéért ma már 1/72 Angol inchet használunk, amely 0,353 mm. Természetesen nem mindenki ált át az egyszerűsítésre, így létezik régi angol pont is, amely 0.351 mm.

Mindkét rendszer értelmezi a pont 12-szeresét, amelynek neve az Európai rendszerben Cicero (4,51 mm), az Angolszász rendszerben pedig Pica (4.24 mm).

4.2. A betű

A nyomtatott szöveg legalapvetőbb építőköve a betű. Amikor betűről beszélünk, akkor általában ide soroljuk a kis és nagybetűket, a számokat, az írásjeleket és még jó néhány speciális jelet is.



4.2. ábra. Talpas és talpatlan fontok

4.2.1. A betűk alakja

A betűknek alakja, stílusa, hangulata van. Amikor a betűt készítő betűmetsző készít egy bizonyos stílusú betűt, akkor általában minden betűnek elkészíti az adott stílusú formaváltozatát, hogy a szöveget egységes stílusban lehessen kinyomtatni.

Az azonos stílusú, együvé tartozó betűk halmazát *font*nak nevezzük. A font tartalmazza pl. a magyar nyelven való íráshoz szükséges összes jelet, olyan formában, hogy „jól mutassanak” egymás mellett. *font*

A fontokat szokás *fontcsalád*okba, osztani a rájuk jellemző főbb stílusjegyek alapján. A tipográfia, a nyomdászat története során több felosztás is kialakult, elterjedt, de sajnos mindegyik túl bonyolult ahhoz, hogy ehelyütt használjuk. Az általunk használt felosztás igen primitív, arra viszont mindenképpen jó, hogy a legfontosabb betűstílusokat megkülönböztessük. Mi a következő felosztást használjuk: *fontcsalád*

talpas fontok: Ezek a betűk jellegzetes, talpakban végződő vonalakból állnak, ellentétben a talp nélküli betűkkel. Egyes szerzők szerint ezek a talpak nagymértékben megkönnyítik az olvasást, ezért ezeket a betűket hosszabb szövegek készítésére, szedésére ajánlják. (4.2 ábra)

Mivel a talpak használata a római időkben jött divatba, ezeket a fontokat szokás *roman* fontoknak is hívni. *roman*

talp nélküli fontok: Léteznek olyan fontok is, amelyeknél a vonalak végén nem található talp. Ezeket talp nélküli vagy francia eredetű kifejezéssel *sans seriff* fontoknak nevezzük. *sans seriff*

Egyesek szerint a talp hiánya megnehezíti az olvasást, ezért ezeket a fontokat rövidebb szövegek (pl. címek) szedésére használjuk.

4.3. ábra. Kis kapitális betűk, ugráló számok

egyméretű fontok: A betűk általában változó szélességűek, vagyis minden betű más és más szélességű helyet foglal el a papíron. Készülnek olyan fontok is, amelyekben minden betű azonos szélességű. Ezeket a betűket egyméretű vagy az angol kifejezéssel *monospace* fontoknak nevezzük.

monospace

A monospace fontok helypazarlóak és nehéz őket olvasni, tehát csak akkor használjuk őket, ha elengedhetetlenül fontos. A számítástechnikai szakirodalomban a számítógép által a képernyő írt szövegek, gépi nyelvek, programok szedésére terjedt el leginkább ez a fontcsalád, az angolszász tipográfiában pedig személytelen, hivatalos szövegek készítésére használják.

egyéb fontok: Egyszerűsített felosztásunkban minden fontot, amely jelentősen különbözik az eddig bemutatottaktól az egyéb kategóriába soroljuk. Ide tartozhatnak a plakátok, de igen feltűnő de rövid szövegek szedésére alkalmas *groteszk* fontok, a díszes, nehezen olvasható *kalligrafikus* fontok vagy a betűk helyett piktogramokat, szimbólumokat tartalmazó *szimbólumtáblák*.

A betűmetsző egy font elkészítésekor gondol arra is, hogy a szerző néha ki akar emelni egy szövegrészt, fel akarja hívni a figyelmet egy címre, ezért szeretne az adott fontban többféle *betűváltozat*ot használni. A font készítésekor ezért a font több változatban is elkészül. Általában a következő változatokat különböztetjük meg:

betűváltozat

kurzív: A kurzív betűváltozat (4.4 ábra) alakja teljesen különbözik a normális párjától, de ahhoz stílusában illeszkedik. A kurzív betűt a mű-

vész szövegben található kiemelések szedésére készítette, oly módon, hogy ízlésesen illeszkedjen a szöveghez.

Kurzív betűt használhatunk egy-egy szó, mondatrészek kiemelésére esetleg címek szedésére. Soha ne használjunk kurzív betűt hosszabb szöveg szedésére, mert ezek a fontok nehezebben olvashatóak, hosszú szöveg esetén fárasztják az olvasót.

A kurzív betűket szokás *italic* változatnak is nevezni (Olaszországban terjedtek el elsőként), de sokan hibásan döntöttnek is nevezik. *italic*

döntött: A döntött betűk (4.7 ábra) alakja többé kevésbé megegyezik normál párjukkal, de ahhoz képest kissé döntött formájúak. A betűt készítő művész azért készít döntött változatot, hogy hosszabb szöveget is ki tudjunk emelni anélkül, hogy az olvasót a nehezebben olvasható kurzív változattal terhelnénk.

A döntött betűk szinte kizárólag hosszabb – néhány mondatos – idézetek kiemelésére szolgálnak.

Tudnunk kell, hogy egyes programok a normál betű megdöntésével kísérelnek meg döntött és kurzív betűket létrehozni, ezek azonban soha nem lesznek olyan minőségűek, mint a betűmetsző által készített változatok.

kis kapitális: A kis kapitális (4.3 ábra) betűk alakja megközelítőleg azonos a nagybetűkkel, mérete pedig pontosan megegyezik a kisbetűkkel. A kis kapitális betűkkel olyan szöveg állítható elő, amely nagybetűs, az olvasó azonban képes különbséget tenni a nagy- és kisbetűk közt.

A kis kapitális betű szinte kizárólag nevek szedésére használatos. Soha ne szedjünk hosszabb szöveget ilyen fontokkal, mert az eredmény csúf és olvashatatlan.

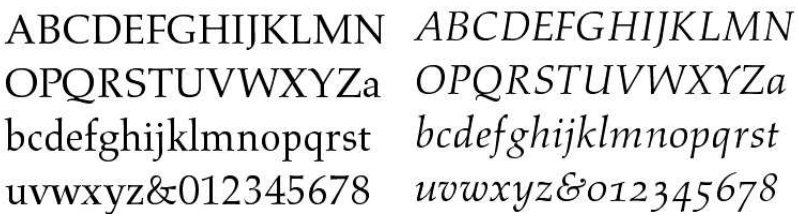
Egyes programok képesek a betűk átméretezésével ilyen jellegű szöveget előállítani, az ilyen betűk minősége azonban nem kielégítő.

Szokás a kis kapitális betűk mellé ugráló számokat helyezni a fontba.

A kis kapitális betűket angolul *small caps* változatnak nevezik. *small caps*

félkövér: A félkövér betű (4.5 ábra) vaskosabb mint az eredeti. Arra készült, hogy a szövegtörzsön kívül eső részeket kiemeljük.

Félkövér betűket használhatunk címekben, plakátokon, de soha ne használjuk a *kenyérszöveg*ben (a dokumentum fő szövegében, a szövegtörzsben) kiemelésre, mert az eredmény csúf és igénytelen lesz. *kenyérszöveg*



4.4. ábra. Normál font és kurzív párja

4.5. ábra. Normál, félkövér, kövér és kurzív kövér fontok

kövér: A kövér betű (4.5) a félkövér betűváltozatnál is vaskosabb képet ad. Használatakor fokozottan igazak a félkövér betűknél elmondottak.

félkövér- és kövér kurzív: Címek szedésére elterjedten használt a kurzív betűk félkövér ill. kövér változata (4.5 ábra). Erre a betűváltozatra igazak a kurzív és a félkövér betűknél elmondottak.

4.2.2. A betűk mérete

A betűk méretét általában pontban szokás megadni. Könyvek, dolgozatok szedésére kb. 12 pontos betűket, cikkek, újságok szedésére 10 pontos betűket szokás használni.

A betűk mérete a betűk magasságával van összefüggésben, de az elterjedt tévhittel ellentétben ez nem egzakt mérés eredménye. Amikor a betűmetsző elkészít egy 12 pontos betűt, akkor tulajdonképpen semmilyen szabály nem köti, nincs előírva pl. az, hogy az „x” betű magasságának éppen 12 pontnak kell lennie. Ez az oka annak, hogy az egyes betűk ugyanazon méretben nagyobbnak tűnnek.

A betűméretekkel kapcsolatban tanácsként csak azt tudjuk elmondani, hogy a helyes betűméret megválasztása igen fontos, érdemes próbanyomatokon megvizsgálni milyen képet ad az adott betűmérettel a font, mennyire olvasható, kellemes a szemnek.

A nyomtatott dokumentumban általában többféle betűméret is szerepel. A címek nagyobb, a szöveg egyéb részei kisebb betűvel készülnek. Igen fontos, hogy a betűméretek megfelelő arányban legyenek egymáshoz és ennek ellenőrzéséhez nem kevés stílusérzék szükséges. Szerencsére a későbbiekben tárgyalt LaTeX tördelőrendszer általában alapbeállítás szerint is olyan összképet produkál, amely megfelel az elvárásoknak.

4.2.3. Ligatúrák

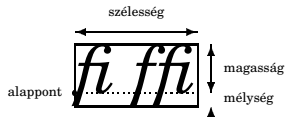
ligatúrák

A *ligatúrák* több betűből, írásjelből képzett jelek, amelyeket a szebb, olvashatóbb íráskép érdekében használunk.

A 4.6 ábra alsó részében láthatjuk milyen formában jelennek meg a betűk, amikor *fi* és *ffi* betűk kerülnek egymás mellé. Láthatjuk, hogy az *f* betű felső része túlságosan közel kerül az utána következő betűhöz. Ez kissé megnehezíti az olvasást – bár nem tragikus mértékben.

A betűket készítő művész általában elkészíti az *fi* és *ffi* betűcsoportokat egy jellé szerkesztve. Ezeket a ligatúrákat a szövegtördelő program helyettesíti a szövegbe. Ahol az *fi* betűk egymás mellett szerepelnek oda pl. az *fi* ligatúrát helyettesítve be, amelyet a 4.6 ábra felső részén láthatunk.

Ma már minden szövegtördelő programtól elvárható, hogy ha a ligatúra rendelkezésre áll az adott fontban, akkor azt használja, így a nyomtatott szöveget szebbé, olvashatóbbá tegye.




```
szélesség = 45.60464pt   magasság = 18.31163pt   mélység = 5.22458pt
```


```
szélesség = 48.34111pt   magasság = 18.31163pt   mélység = 5.22458pt
```

4.6. ábra. Ligatúrák

4.2.4. Ékezetek és mellékjelek

ékezetek

A különféle nyelvek által használt *ékezetek* és mellékjelek nagymértékben megnehezítik a nyomdai munkáklatokkal foglalkozó szakemberek, művészek dolgát. Még Európán belül is olyan változatos ékezeteket használnak az egyes nemzetek, hogy szinte már művészet azokat kezelni.

A professzionális programoknak minden ékezetet a megfelelőképpen kell kezelnie, hiszen bármelyik dokumentumban lehet egy név, egy idegen szó, amely szükségessé teszi az adott ékezet használatát. Nem szabad tehát olyan programot használnunk, amely csak bizonyos nemzet vagy nemzetek betűit képes kezelni, hiszen nagyon könnyen előfordulhat, hogy a a szövegben található egyetlen betű miatt lesz a végeredmény elfogadhatatlan.

Sok program és szabvány az ékezeteket és mellékjelek a betűktől elválasztva, külön kezeli. Ez azt jelenti a számunkra, hogy építőkockaként állíthatjuk elő a betűkből és mellékjelekből az írásképet. Tehetünk pl. egyszeres hosszú ékezetet az y betűre (ý) vagy hullámos ékezetet az n betű fölé (ñ).

A 53. oldalon található 5.4 táblázatban felsoroljuk a legfontosabb ékezeteket, amelyeket a LaTeX segítségével nyomtathatunk.

4.7. ábra. Döntött talpatlan font

4.2.5. Írásjelek kezelése

Az írásjelek igen sok kiadványban hibásan szerepelnek, ami rettenetesen zavaró és csúnya, ezért néhány szót mindenképpen ejtenünk kell a témáról. A nyomdai költségek csökkentése érdekében itt mutatjuk be a később tárgyalt LaTeX programcsomag írásjelek készítésére szolgáló eszköztárát is.

Magyar nyelvben a mondat végi írásjelek (.!? stb.) előtt soha nem hagyunk ki szóközt, míg utánuk mindig egy szóköz található. Ugyanez igaz a vesszőre, pontosvesszőre és általában a mondaton belüli tagmondatot elválasztó írásjelre is.

Az idézőjelek a magyar nyelvben a 9-es szám alakját követik, mindig alul kezdődnek és az idézett szöveg végén fent végződnek. A helyes alak tehát; „idézet”. Ez a példa LaTeX alatt `tehát; ,,idézet''. Ez` formában készült.

Idézeten belüli idézetet a » és « jelekkel jelzünk. A jeleknek mindig az „idézet felé kell »mutatniuk«, belül szóköz nélkül, kívül szóközzel”. Ezt LaTeX alatt `kell >>mutatniuk<<, belül` formában készíthetjük el.

A gondolatjel a magyar szabályok szerint „–” jelként jelenik meg (*félkvirtmínusz*). A gondolatjelek előtt és után – általában – szóközök találhatóak, de az írásjelek – vessző, pontosvessző stb. –, itt is szóköz nélkül kapcsolódnak. A LaTeX ezt a jelet a `--`. betűk hatására állítja elő. *félkvirtmínusz*

Az angol nyelv idézőjelei mindig befelé mutatnak és felül helyezkednek el, pl. “Yes” (`“Yes''`). A belső idézőjelek hasonlóak, de nem párosak: “I said ‘Yes’, nothing else.”.

Igen fontos az is, hogy az angol gondolatjelek a magyar nyelvben használt gondolatjeleknél hosszabbak (*kvirtmínusz*), azok előtt és után nem használunk szóközt: “Yes—he said–nothing else.”. Ezeket a jeleket a `---` három betűs kifejezéssel szedhetjük LaTeX-et használva. *kvirtmínusz*

Nagyon fontos az is, hogy a LaTeX csak a megfelelő nyelvet támogató üzemmódban szedi helyesen ezeket a jeleket, többnyelvű dokumentumnál el kell helyeznünk a `\selectlanguage` parancsot azokon a helyeken, ahol más nyelven kezdünk írni. Ennek a parancsnak a használatát mutatja be a következő példa:

\selectlanguage

```
\selectlanguage{english}
We can type english text here.
\selectlanguage{hungarian}
Magyarul pedig itt...
```

4.3. Sorok, sortörés

tördelőprogram

A *tördelőprogram* egyik legfontosabb feladata a szövegnek sorokra való tördelése. E művelet során a program kiszámítja, hogy a szöveg mekkora része fér el egy adott hosszúságú szakaszon és ahol szükséges új sort kezd.

Azt gondolhatnánk, hogy ez a művelet egyszerű, a valóságban azonban meglehetősen bonyolult. A számítógépek megjelenése előtt már kialakultak azok az összetett szabályok, amelyek alapján a sorokra tördelés elvégezhető és amelyeket a tördelőprogramoknak is figyelembe kell venniük. Az igényes programok ezen szabályok alapján egy optimalizációs problémára vezetik vissza a sorok végének elhelyezését és megkísérlik megkeresni a legjobb elrendezést.

Az első – és legfontosabb – kérdés az, hogy hol lehet új sort kezdeni. A szavak végénél például általában lehet új sort kezdeni, de nem minden esetben. Nem szokás megengedni a sortörést pl. a nevek elő- és utótagja után közvetlenül. A Dr. Ács J. név esetében pl. a Dr. után, a J. előtt nem szerencsés a sortörés, mert a szöveg olvashatatlanná válik. Ilyen esetben *nem törhető szóköz*ről beszélünk, olyan szóközökről, amelyek szóközként jelennek meg ugyan, de soha nem a sor végén.

nem törhető szóköz

Ha a sortöréseket csak a szóhatároknál engedélyeznénk, akkor sajnos a tördelést nem lehetne kielégítő minőségben elvégezni. Hosszú szavak esetében ugyanis a a sorok hossza olyan nagy mértékben különbözne, hogy a kialakuló íráskép csúf lenne.

Figyeljük meg a következő összefüggést; minél kevesebb betűt tartalmaznak a sorok, a hosszú szavak annál nagyobb részét foglalják el, tehát annál kevesebb az esély arra, hogy a szóhatároknál kielégítő módon sortörést

tudunk elhelyezni. Rövid sorhossz, nagy betűméret esetén tehát nehéz jó sortörést találni.

Azt gondolhatnánk ebből, hogy sok betűt tartalmazó sorokat kell készítenünk a megfelelő sortörések érdekében. Ez azonban nem járható út, mert minél több betűt tartalmaz egy sor, az olvasása során annál nagyobb az esély arra, hogy az olvasónak a sor elejének megtalálása problémát okoz. Sok betűt tartalmazó sorok olvasása tehát fárasztó. Úgy kell tehát megválasztanunk a sor hosszát az adott betűmérethez, hogy annak olvasása ne fárassza túlságosan az olvasót. Ez az oka például annak, hogy a helytakarékosság miatt kicsiny betűméretet használó újságok rövid hasábokba rendezik a szöveget.

Az olvashatóság miatt tehát korlátokat kell bevezetnünk a sorban szereplő betűk számára nézve, amely bizonyos mértékben megnehezíti a szöveg sorokra tördelését. A probléma megoldását az elválasztás adja. Az elválasztás során a szövegben található szavakat elvágjuk, a szó elejét a sor végére kiírjuk, a végét pedig a következő sor elejére tesszük. Az elválasztást egy - jellel jelezzük a sor végén, hogy az olvasó a szó elejének olvasásakor már felfigyeljen az elválasztásra.

Az elválasztás már a számítógépek megjelenése előtt kialakult szabályrendszerre alapul, ez pedig további problémákat vet fel. Ennek oka az, hogy az elválasztási szabályok, amelyek leírják, hogy egy szót hol lehet elválasztani, meglehetősen bonyolultak. A legtöbb elválasztást végző program nem minden lehetséges elválasztási helyet talál meg és – ami még rosszabb – néhány esetben olyan helyen is elválasztást engedélyez, ahol azt a szabályok nem engedik meg.

Tovább bonyolítja a helyzetet az, hogy minden beszélt nyelvnek saját elválasztási szabályai vannak. Ha pl. az angol nyelvben érvényes elválasztási szabályok alapján választunk el egy magyar nyelven írott szöveget, akkor nem csak igénytelen, de helyesírási szempontból is hibás írásképet kapunk.

További problémát okoznak a több nyelven írott dokumentumok. Ilyen dokumentumok esetében a programnak minden használt nyelv elválasztási szabályait ismernie kell, a szerzőnek pedig jeleznie kell a szövegben, hogy az adott ponttól más nyelvre tér át.

4.3.1. Vízszintes távolságok

A szöveg sorokra tördelése során vízszintes méreteket, vízszintes távolságokat kell vizsgálnia a tördelést végző programnak.

Minden betűnek van egy vízszintes mérete, amely megadja, hogy a betű vízszintes irányban mekkora kiterjedésű. Egy betűtípus esetén általában minden betű más-más méretű helyet foglal a sorban. Az m betű szedéséhez szélesebb, az i betű szedéséhez keskenyebb helyre van szükség. Ez természetesen azt eredményezi, hogy minden sorban más lesz a betűk száma.

Az egy szót alkotó betűk közt keskeny üres részek találhatóak. Ezek a keskeny üres részek választják el az egyes betűket egymástól, hogy azok ne follyanak össze.

Sajnos a betűket elválasztó üres részek sem lehetnek azonos szélességűek, azoknak akkoráknak kell lenniük, hogy az olvasó azonos méretűnek lássa őket. A 4.8 ábra alsó részén található szövegben pl. a távolság azonos minden betű közt, mégis úgy tűnik, minta az FA betűk közt nagyobb lenne a távolság. Ez nyilván valóan érzékcsalódás, de kompenzálunk kell, mert nehezíti az olvasást, sőt szélsőséges esetben értelemzavaró lehet (F A TIME?). Az ábra felső részében a kompenzált változat látható, amely látszólag azonos távolságra lévő betűket tartalmaz. Ha az ábra ezen részét figyelmesen megvizsgáljuk, akkor láthatjuk, hogy az F és A betűk közt tulajdonképpen nincs is vízszintes helykihagyás.

Természetesen nem csak a nagybetűk esetében van szükség a vízszintes térközök módosítására. A következő példaszöveg egyike állandó térközökkel készült, amely kissé zavaró:

Valami hiányzik ebből a szövegből. Csak nem a kerning?

Valami hiányzik ebből a szövegből. Csak nem a kerning?

Amint látjuk az állandó térközök használata nem túl feltűnő, a gyakorlott szem azonban azonnal észreveszi.

kerning

A szavakon belüli térközök beállítását a *kerning* módszerével végzik a programok. A fontok nem csak a betűk alakját, hanem az egyes betűpároknál használt távolságmódosítót is tartalmazzák. Ha pl. két M betű található egymás mellett, akkor a táblázatból az MM helyen kiolvasható, hogy mekkora mértékben kell növelni az alap betűtávolságot. A VA betűpár esetében pl. valószínűleg negatív értéket tartalmaz a táblázat, mivel ezeket a betűket formájuknál fogva kissé közelebb kell tennünk egymástól.

kerning táblázat

A fontot készítő művész tehát nem csak a betűk alakját készíti el, hanem a *kerning táblázat*ot is, amely megadja, hogy milyen mértékben kell eltérni az egyes betűpároknál az alap betűtávolságtól.

A betűk közti távolság után vizsgáljuk meg a szavak közti távolságot! A szavak közt található távolságok – a szóközök szélessége – szintén nem állandó, hiszen ezek méretének módosításával éri el a program, hogy a sorok egyforma hosszúak legyenek a bekezdésben. A szóközök szélessége tehát

szélesség = 101.037pt magasság = 18.18713pt mélység = 0.0pt

szélesség = 102.7287pt magasság = 18.18713pt mélység = 0.0pt

4.8. ábra. Kerning

bizonyos mértékben változik. (Szeretnénk felhívni a figyelmet arra, hogy bizonyos programok a szavakon belüli távolságok növelését is felhasználják arra, hogy a sorokat a megfelelő méretre nyújtsák. Ez nagyon igénytelen, csúnya írásképet, sokszor olvashatatlan formát eredményez.)

A szóközök mérete azonban nem növelhető tetszőleges mértékben. Ha túlságosan megnöveljük a szóközöket, akkor csúnya üres helyek jelennek meg a szövegben, amelyek már zavarják az olvasót. Különösen abban az esetben lesznek feltűnőek ezek az üres helyek, ha egymás fölé kerülve összeolvadnak. A megnyújtott vízszintes helyek összeolvadását *utcásodás*nak nevezzük. Az ilyen szöveg foltos, csúnya és olvashatatlan. Az utcásodás elkerülhető a sorhossz és a betűméret helyes megválasztásával, az elválasztás bekapcsolásával vagy a sorok balra zárásával.

utcásodás

A szavak közt található szóközök szélességén kívül szót kell ejtenünk a mondatok közti távolságról is. A magyar tipográfiai szabályok előírják, hogy a mondat végi írásjelek után egy szóköznyi helykihagyásnak kell következnie. Más nyelvek esetében más távolságértékek használata válhat szükségessé. Az angol nyelvben szokás pl. a mondatvégi írásjelek után a szóköznél szélesebb helyet kihagyni, ami a magyar olvasó számára zavaró. Különösen furcsa a magyar szövegben a mondat végi írásjelek előtt kihagyott vékony szóköz, amely szintén szokás néhány országban.

A vízszintes távolságok tárgyalása során feltételeztük, hogy a sorok szélessége ismert, állandó, ez a távolság azonban szintén nem változatlan érték. Ha minden sor pontosan egy vonalban végződne, akkor az érzékcsaló-

dás miatt úgy tűnne, hogy bizonyos sorok beljebb végződnek a többi sornál. A következő szöveg pl. úgy tűnik nem középen helyezkedik el:

. . . a folytonosan változó
távolságok. . .

kilógatás

Szokás az ilyen érzékcsalódás hatását *kilógatás*sal csökkenteni. Ilyenkor a kevésbé vaskos betűkre végződő sorokat a többinél kissé kijjebb befejezni, ezeket a sorokat „kilógatni”.

4.3.2. Függőleges térközök

A szöveg sorokra tördelése után a kapott sorokat függőlegesen egymás alá kell helyeznünk. Itt is elsőrendű feladat az, hogy a szem számára szabályosnak tűnő elrendezést alakítsunk ki, amelyet könnyű olvasni, amely nem fárasztja az olvasót.

alapvonal

A munka során a betűket az *alapvonal*ra illesztjük. A betűk úgy készültek, hogy az alapvonalra illesztve „egy vonalba legyenek”. A betűk mérete nem csak vízszintes, hanem függőleges irányban is különbözhet egymástól, vagyis a betűk különféle magasságúak lehetnek. Olyannyira így van ez, hogy néhány betű még az alapvonal alá is „lelóghat”. A 4.9 ábrán az alapvonal alá nyúló betűket is láthatunk.

A szövegtördelő programok az egymás után következő sorok távolságát megkísérlik úgy megállapítani, hogy a legolvashatóbb, legkevésbé zavaró, ugyanakkor esztétikus szövegképet kapják. Ez nem mindig sikerülhet maradéktalanul. Ha a sorban pl. igen magas vagy nagyon mély jelek találhatóak, akkor azok miatt a sorokat függőleges irányban el kell távolítani egymástól, ami meglehetősen csúnya képet ad.

4.4. Bekezdések

A bekezdés a szerző szándéka szerinti szövegtagolási egység, amely a téma tagolására szolgál. A bekezdés határainak kijelölése tehát a szerző feladata, a tipográfia csak a bekezdések jelzésében segít.

A bekezdéshatárok jelölésére kétféle módszer terjedt el. Az első módszer szerint a bekezdések közt kicsiny üres helyet hagyunk ki függőleges irány-

4.9. ábra. Alapvonal

ban. Ez a módszer magyarországon nem igazán terjedt el, de bizonyos esetekben használjuk.

A második módszer a bekezdések első sorainak beljebb kezdésével, *beütés*sel vagy *behúzás*sal jelzi a bekezdéshatárokat. Ilyenkor a bekezdések eleje a bal oldalra pillantva azonnal megtalálhatóak. *beütés* *behúzás*

Általában nem szokás a két módszert kombinálni – vagyis beütéssel és üres hely kihagyásával is jelölni a bekezdéshatárokat, de a beütéseket szokás elhagyni az első bekezdésben. Ilyen esetben a címek utáni első bekezdésnél nem használjuk a beütést, az első bekezdéseket *tompa szedés*sel szedjük. Ez a könyv is ilyen módszerrel jelzi a bekezdéseket. *tompa szedés*

A bekezdések többféle *rendezés*sel is készülhetnek. A rendezés határozza meg, hogy a bekezdés sorai vízszintes irányban hogyan legyenek egymás alá rendezve. A *balra rendezett* bekezdések sorai egy vonalban kezdődnek, végük nincs pontosan egy vonalban. Ezt a rendezést szokás *zászlós szedés*nek is nevezni. Angolszász területen szokás a dokumentumok fő szövegét – a kenyérszöveget – zászlós szedéssel szedni. *rendezés* *balra rendezett* *zászlós szedés*

A *jobbra rendezett* szedés esetén a sorok vége kerül egy vonalba, elejük kissé „ugrál”. Az ilyen szedés meglehetősen különleges, használhatjuk pl. mottók, széljegyzetek szedésére. *jobbra rendezett*

A *sorkizárt szedés*nél a sorok eleje és vége is egy vonalba kerül. Ezt a módszert szokás *blokkos szedés*nek is nevezni. Blokkos szedés esetén ügyelnünk kell, hogy ne alakuljon ki *utcásodás*, a túlságosan eltávolodott betűk miatt *sorkizárt szedés* *blokkos szedés* *utcásodás*

ne alakuljanak ki fehér foltok a szövegben. Blokkos szedést ezért soha ne alkalmazzunk elválasztás nélkül.

4.4.1. Távolságok

A tipográfia szempontjából igen fontos, hogy a nyomtatott dokumentumban hol milyen távolságok mérhetőek. Arányérzék, stílusérzék, a szabályok és a szakma pontos ismerete nélkül reménytelen a legegyszerűbb dokumentum megtervezése. Ehelyütt ennek ellenére nem fogjuk ezt a témát túl mélyen taglalni, aminek egyrészről a helyhiány az oka, másrészről pedig az, hogy az általunk tárgyalt szövegtördelő rendszer általában alapbeállításokkal is elfogadható eredményt produkál.

Az első és legfontosabb méret számunkra a papír mérete és magassága. A nyomdatechnikában szabványos papírméreteket használnak, amelyek egyrészről veszteségmentes hajtogatást biztosítanak, másfelől arányosak, ízlésesek. Az európai és az angolszász szabványok és szokások ebben a kérdésben is különböznek, bár a szabályok nem annyira szigorúak. Magyarországon szokás pl. az amerikai levélméretet is használni és az A5 méret sem kötelező. (Ez a könyv is B5 méretben készült, amely több helyet ad és talán kezelhetőbbé teszi az elkészült művet.)

A papírméret kapcsán meg kell jegyeznünk, hogy a könyveket és más fűzött, kötött dokumentumokat elkészültük után szokás körülvágni. A nyomdatechnika a körülvágás segítségével biztosítja azt, hogy az oldalak széle pontosan egy vonalba essen. Meg kell különböztetnünk tehát a *nyers méret*et a *körülvágott méret*től. Nyomdai tapasztalattal nem rendelkező számítógépfelhasználók sokszor a kiadvány készítésének utolsó szakaszában – a nyomdával való konzultáció során – ébrednek rá, hogy a kiadványt tévesen a nyers méretre tervezték meg, ezért az komoly átalakításra szorul.

nyers méret

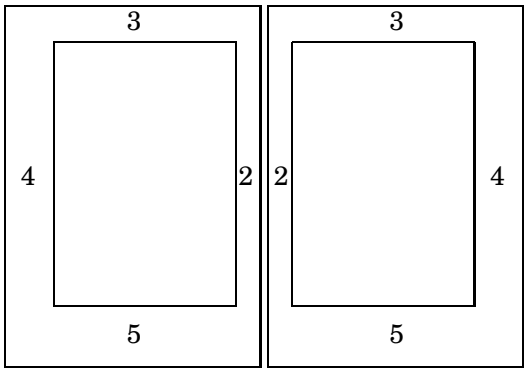
körülvágott méret

A lapméreten kívül igen fontosak a *margók*. A margók üres sávok, amelyek a szöveget minden irányból keretezik. A margók szerepe kettős; egyrészről lehetővé teszik az olvasó számára, hogy a kész művet a kezükbe tartsák anélkül, hogy a szöveget ujjukkal letakarnák, másrészről olvasás közben a zavaró hátteret kitakarják.

margók

A margók által határolt terület – a szöveg helye – a *szedéstükör*. Angolszász területen szokás a szedéstükrön kívülre, a margókba írni. Legfeltűnőbb eszköz ebből a szempontból a széljegyzet, amely ebben a könyvben is igen sok helyen megfigyelhető. Széljegyzet használatakor az angolszász rendszerben elterjedten használt széles külső margók használata javasolt. Ez a stílus hazánkban egyre inkább kezd elterjedni, ami elegáns külsőt,

szedéstükör



4.10. ábra. A margók arányai

kezelhetőbb formát kölcsönöz a könyveknek. (A könyv egyszerűen nem csukódik be, ha nyitva lefektetjük az asztalra, mert a lapok súlya ezt megakadályozza.) Érdemes még megemlíteni, hogy a széles külső margók sokszor „ingyen vannak”, hiszen a nyomdagépek sokszor amúgy is ilyen lapméretet használnak, csak a könyv körülvágásával lehet keskenyebbre állítani a lapméretet. A 4.10 ábra a helyesen megválasztott margók arányait mutatja.

A lapok felső és alsó részén szokás ismétlődő jeleket, szövegeket elhelyezni. Az ilyen elemek számára külön területet – a fent található *élőfej*et és a lent elhelyezkedő *élőláb*at – tartunk fent. Az élőfejben vagy az élőlábban helyezkedik el az oldalszámozás. Az oldalszámról feltétlenül tudnunk kell, hogy a jobb kéz felől elhelyezkedő oldalaknak mindig *páratlan oldalszám*ot kell viselniük, ellenkező esetben a nyomdatechnikában járatos kollégáink a legjobb esetben is kinevetnek minket, rosszabb esetben a szakma megvetése kíséri utunkat. Jellemző módon a könyv további részeiben tárgyalt LaTeX tördelőrendszer terminológiájában *páros és páratlan oldalak* szerelnek, a páratlan oldalak pedig természetesen jobb kéz felől esnek.

páratlan oldalszám

páros és páratlan oldalak

5. fejezet

A LaTeX használata

A TeX tördelőprogram DONALD KNUTH világhírű matematikus alkotása, aki a legenda szerint megelégelte könyveinek hosszú nyomdai átfutási idejét. A számítástechnikában járatos olvasó valószínűleg ismeri a magyar nyelven is megjelent; *A programozás művészete* c. többkötetes művét, amelyet fellapozva azonnal nyilvánvaló, hogy a szerzőnek matematikai szövegek szedésére is alkalmas számítógépes programra volt szüksége.

A TeX rendszerre épülve LESLIE LAMPORT készített egy makrócsomagot, amely könnyebb kezelhetőséget, egyszerűbb munkavégzést tesz lehetővé. Műve – a LaTeX – ma már annyira közkedvelt, hogy a felhasználók általában nem közvetlenül használják a TeX rendszert.

Fontos felhívnunk a figyelmet arra, hogy a LaTeX a TeX nyelvén megírt, a TeX fogalomrendszerét egy magasabb absztrakciós szintre emelő makrórendszer. Minden TeX parancs használható LaTeX alatt is, a LaTeX csak hozzátett a KNUTH által készített rendszerhez, el nem vett abból.

A TeX és a LaTeX nevét hagyomány szerint az itt látható formában használjuk, hogy ezzel is jelezzük a programok képességeit. Erre a `\TeX{}` és `\LaTeX{}` parancsok használhatóak. Ha az általunk használt adathordozó nem képes megjeleníteni ezeket a logókat, akkor a TeX és LaTeX szöveget használjuk. A TeX kiejtése magyarországon „*tehh*”, a LaTeX-é „*latehh*” formában terjedt el.

`\TeX{}`
`\LaTeX{}`

E könyvben a használt programrendszert LaTeX-ként emlegetjük, csak itt hívjuk fel a figyelmet arra, hogy sok kijelentésünk igaz a TeX-re is. Sok bemutatott eszköz az interneten megtalálható kiegészítők – LaTeX *csomagok* – használatával érhető el, amelyek szigorú értelemben nem a LaTeX részei.

csomagok

név	*Rövidítés*	*Átszámítás*
	em	M betű szélessége
	ex	x betű magassága
pont	pt	1 in = 72,27 pt
pica	pc	1 pc = 12 pt
inch	in	1 in = 25,4 mm
nagy pont	bp	1 in = 72 bp
centiméter	cm	1 cm = 10 mm
milliméter	mm	
Didot pont	dd	1157 dd = 1238 pt
cicero	cc	1 cc = 12 dd
kicsi pont	sp	65536 sp = 1 pt

5.1. táblázat. A LaTeX mértékegységei

em
ex

A LaTeX által használt mértékegységeket a 5.1 táblázat tartalmazza. A táblázatban megtalálhatóak a relatív mértékegységek (az `em` és az `ex`), amelyek konkrét mérete mindig a használt alap betűmérettől függ. Hasznos lehet ezekkel megadni a méreteket, hiszen így a lap arányos marad méretváltoztatás után is.

A mértékegységek használatánál szem előtt kell tartanunk, hogy a mérőszámot és a mértékegységet mindig közvetlenül egymás mellé kell írnunk, például `0.1mm`.

5.1. A LaTeX használata

A LaTeX használata roppant egyszerű. El kell készítenünk egy állományt, amelynek neve `.tex` betűkre végződik és a programmal le kell fordítanunk ezt nyomtatható formátumra. A művészet egyszerűen csak az, hogy kedvenc szövegszerkesztő programunkkal mit írunk ebbe az állományba.

A LaTeX állomány tartalmazza a szöveget, amelyet ki akarunk nyomtatni és a LaTeX parancsait, amelyek a nyomtatás formájára – a megjelenésre – vonatkozó utasításokat hordozzák. Ha meg akarjuk tekinteni, hogy az általunk elkészített állomány hogyan jelenne meg nyomtatásban, ha ki akarjuk nyomtatni az állományt, akkor meg kell kérnünk a LaTeX rendszert, hogy

tördelje a dokumentumot. Ekkor azt mondjuk, hogy a LaTeX-et futtatjuk az adott dokumentumon.

A LaTeX futtatása igen egyszerű feladat. Be kell írnunk a `latex` parancsot és közvetlenül utána – szóközzel elválasztva – az állomány nevét, amelyen a LaTeX-et futtatni akarjuk: `latex`

```
# latex elso.tex
...
```

Meglehetősen sok üzenet jelenik meg ennek hatására a képernyőn, a LaTeX ugyanis munkájának minden lépéséről értesít minket.

Hiba esetén a LaTeX hibaüzenetet ad és megkísérli folytatni a munkát. Ha a hiba nem kritikus, akkor a dokumentum elkészül, bár nyilvánvalóan lesznek benne olyan részek, amelyek nem felelnek meg elvárásainknak. Ha a hiba kritikus, akkor a LaTeX megszakítja a munkát. Ha a kritikus hibát pl. az 50. oldalon vétettük, akkor a kész anyag csak 49 oldalas lesz, attól függetlenül, hogy hány oldalnyi anyagot gépeltünk be a hiba után.

A LaTeX az elkészített anyagot *dvi* formátumban készíti el, amelyet megtekinthetünk az `xdvi` programmal vagy nyomtathatunk a `dvips` programmal. A következő lista a LaTeX-el végzett munka legfontosabb programjait mutatja be: *dvi* `xdvi` `dvips`

`latex`: A LaTeX állomány lefordítása, a szöveg tördelése. A program egyetlen paramétere a LaTeX állomány neve, amely általában `.tex` végződésű.

`latex2html`: A LaTeX állományból weblapok készítése. A program egyetlen paramétere a feldolgozandó állomány neve.

A weblapok a program által létrehozott könyvtárba kerülnek elhelyezésre. A könyvtár a munkakönyvtárból nyílik, neve megegyezik a feldolgozandó LaTeX állomány nevével, a `.tex` végződés elhagyásával.

`pdflatex`: A LaTeX állomány tördelése pdf formátumra. A program a `latex` programhoz hasonlóan használható.

`dvips`: A tördelt dvi dokumentum nyomtatása vagy Postscript formátumra alakítása (`-o` kapcsoló). A program paramétere a feldolgozni kívánt dvi állomány neve. Ha a `-o` kapcsolót is megadtuk, akkor azt a Postscript állomány nevének kell követnie, amelyben el kívánjuk helyezni a kész Postscript dokumentumot.

dvilj: A tördelt anyag átalakítása HP Laserjet formátumra.

dvi2fax: A tördelt anyag átalakítása fax formátumra.

dvipdf: Tördelt anyag átalakítása pdf formátumra.

ps2ascii: Postscript állomány alakítás ASCII formátumra.

ps2pdf: Postscript állomány alakítása pdf formátumra.

psnup: Több Postscript oldal egy oldalra másolása.

pstops: Postscript állomány átalakítása. Ez a parancs meglehetősen bonyolult módon kezelhető, de igen fejlett képességekkel rendelkezik. Gyakorlatilag minden munkát elvégezhetünk vele a Postscript állományon, amelyet egy nagyításra és kicsinyítésre képes fénymásolóval szokás csinálni.

xdvi: A tördelt dvi dokumentum megtekintése. A program grafikus felületen indítható, egyetlen paramétere a megtekintendő dvi állomány neve.

A programra kattintva az „észreveszi” ha az állomány megváltozott, ezért elegendő egyszer elindítani a munka során.

gv: Postscript állomány megtekintése, nyomtatás oldalanként. Ez a program fejlett eszközökkel rendelkezik segíségével nem csak megnézhetjük az állományt, azon különféle műveleteket is elvégezhetünk.

5.1.1. Parancsok, környezetek

Amint azt már láttuk, a LaTeX a megfelelő módon elkészített szöveges állományból készíti el a nyomdai minőségű, tördelt dokumentumot. A LaTeX állomány elkészítésekor néhány fontos dolgot figyelembe kell vennünk.

parancsok

A LaTeX a szövegben elhelyezett LaTeX *parancsok*at hajtja végre. Minden LaTeX parancs a \ (*backslash*) jellel kezdődik, amelyet a parancs neve követ. A parancs nevének végét az első olyan jel jelzi, amely nem része az angol nyelv betűinek.

\
backslash

{}

A {} zárójelek szintén fontosak a LaTeX számára, ezeket a a csoportosításra használhatjuk. Mind a \ mind pedig a {} jelek a LaTeX jelei, így azok nem kerülnek be a nyomtatott dokumentumba. Ha ezeket a jeleket ki

akarjuk nyomtatni, akkor a 5.4 táblázatban látható módon kell helyettesítenünk.

A `\text hello` begépelésekor például a parancs a `\text`, a `\text{}` esetén szintén, sőt a `\text12` használatakor is, hiszen az `1` nem része az angol nyelv betűinek.

A szövegben elhelyezett parancsok paramétereket kaphatnak, amelyeken a parancsok műveleteket végeznek. A *parancs paramétere* általában kapcsos zárójelek közé kerü. A `\textit{a)}` szöveg begépelésekor pl. a parancs a `\textit`, a paramétere pedig `a)`.

parancs paramétere

Bizonyos parancsok esetében a paraméterek némelyike elhagyható (*opcionális paraméterek*). Ezeket a paramétereket nem kötelező megadnunk, amit az is jelez, hogy ezek a paraméterek *[ ]* jelek közt találhatóak. A LaTeX alapértelmezés szerint a `[ ]` jeleket nyomtatja, csak ott tulajdonít nekik különleges jelentést, ahol elhagyható paramétert fogad el. A `\ak[ez]{itt}` parancs például két paramétert kap, amelyek közül az első elhagyható volna.

opcionális paraméterek

[ ]

Bizonyos parancsoknak nincs paramétere, korlátozni lehet azonban a hatókörüket a kapcsos zárójelekkel. Ezek a parancsok hatásukat az első `}` jelig vagy az első `\end{}` parancsig fejtik ki. A `{\it h}` például hatását csak a `h` betűre fejti ki.

Magunk is létrehozhatunk *új parancsok*at a `\newcommand` és a `\renewcommand` segítségével. A parancs létrehozásának formája a következő:

új parancsok

`\newcommand`

`\renewcommand`

```
\newcommand{\kover}[1]
 {\textbf{#1}}
```

Amint látjuk a `\newcommand` paramétere a létrehozni kívánt új parancs neve. A második – elhagyható – paraméter az új parancs paramétereinek száma. A harmadik paraméter az új parancs tartalma, azok a parancsok, amelyeket az új parancs végrehajt. Itt a `#1`,`#2`,... formában hivatkozhatunk a paraméterek aktuális értékére.

`#1`

`#2`

A dokumentum területén bárhol hozhatunk létre ilyen módon parancsokat és akár már a következő sorban használhatjuk őket. A példában szereplő parancsot `\kover{szöveg}` formában használhatjuk.

Ha az új parancs már létezik, vagyis nem létrehozni akarunk parancsot, hanem megváltoztatni, akkor a `\renewcommand` parancsot kell használnunk.

A LaTeX értelmezi az ún. *környezetek*et. A környezet hosszab szöveg stílusát, formáját, kezelését meghatározó stílus, amelyre nevével hivatkozhatunk.

környezetek

\begin
\end

A \begin és \end parancsok segítségével kezdhetünk és zárhatunk le környezeteket. E két parancs kötelezően megadandó paramétere a környezet neve. A következő példa a small környezet használatát mutatja be, amely a szöveget piciny betűkkel szedi:

```
\begin{small}
  Ide kerül a szöveg
\end{small}
```

A környezetek egymásba ágyazhatóak, de gondosan kell ügyelnünk arra, hogy mindig azt a környezetet fejezzük be előbb az \end paranccsal, amelyet később kezdtünk a \begin paranccsal:

```
\begin{center}
  \begin{small}
    A szöveg
  \end{small}
\end{center}
```

A példán láthatjuk, hogy az eligazodás érdekében a belső részeket kissé beljebb kezdtük. Ez a LaTeX működését nem befolyásolja, nagyban segíti azonban a szerzőt abban, hogy a LaTeX állományban munka közben eligazodjon, azt könnyebben átlássa.

új környezet
\newenvironment
\renewenvironment

Munka közben lehetőségünk van *új környezet*eket létrehozni. Erre a \newenvironment és a \renewenvironment parancsokat használhatjuk. Ha a környezet még nem létezik, akkor a \newenvironment parancsot, ha már létezik és csak módosítani akarjuk, akkor a \renewenvironment parancsot kell használnunk.

A parancsok a következő formában használhatóak:

```
\newenvironment{motto}[1]
               {\begin{flushright}#1 \it}
               {\end{flushright}\bigskip}
```

Amint látjuk a \newcommand parancsnál megismert formát kell követnünk, azzal a különbséggel, hogy itt található egy negyedik paraméter is. A \newenvironment és a \renewenvironment negyedik paramétere azokat a parancsokat tartalmazza, amelyeket az új környezet használatakor annak végén kell kiadni.

Osztály	*Leírás*
`article`	Cikkek írására szolgáló dokumentumosztály.
`report`	Hosszabb cikkek, beszámolók osztálya.
`letter`	Levelek írására szolgáló osztály.
`book`	Könyvek dokumentumosztálya.
`slides`	Írásvetítő fóliák készítésére szolgál.

5.2. táblázat. A legfontosabb dokumentumosztályok

5.1.2. A LaTeX állomány alapszerkezete

minimális LaTeX állomány

A következő példa a *minimális LaTeX állomány*t mutatja be:

```
\documentclass[10pt,a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage[latin2]{inputenc}
\usepackage[hungarian]{babel}
\begin{document}
  A szöveg...
\end{document}
```

`\documentclass`
dokumentumosztály

Az állománynak mindig az `\documentclass` paranccsal kell kezdődnie, amely meghatározza a *dokumentumosztály*t, vagyis azt, hogy milyen jellegű a készítendő dokumentum. A legfontosabb dokumentumosztályokat a 5.2 táblázat mutatja be. A parancs elhagyható paramétere egy vesszőkkel elválasztott lista, amely megadja, hogy az adott dokumentumosztály mely kapcsolóit, módosítóit kívánjuk használni. A legfontosabb kapcsolókat a 5.3 táblázat mutatja be.

`\begin{document}`
`\end{document}`
preambulum
`\usepackage`
csomagok betöltése

A dokumentum szövegét a `\begin{document}` és `\end{document}` közé kell írnunk. A `\documentclass` és a `\begin{document}` között helyezkedik el az ún. *preambulum*, ahova az egész dokumentum megjelenését befolyásoló parancsokat helyezhetjük el. Csak itt, a preambulumban elhelyezhető pl. a `\usepackage` parancs, amely a LaTeX képességeit kiegészítő csomagok betöltésére szolgál. A *csomagok betöltése* csak a `\usepackage` paranccsal lehetséges.

ékezetes karakterek

A fenti példa néhány csomag betöltését is tartalmazza. Ezek a csomagok felkészítik a LaTeX-et a magyar nyelvben használt *ékezetes karakterek* és a magyar nyelvben használt *elválasztási szabályok* kezelésére.

Kapcsoló	*Leírás*
`10pt`,`11pt`, `12pt`	A dokumentum alap betűmérete 10, 11, ill. 12 pontos.
`a4paper`	A lapméret 210mm×297mm.
`a5paper`	A lapméret 148mm×210mm.
`b5paper`	A lapméret 176mm×250mm.
`letterpaper`	A lapméret 8,5in×11in.
`legalpaper`	A lapméret 8,5in×14in.
`landscape`	A dokumentum laptájolása fekvő.
`draft`	A túlcsordult sorok jelzést kapnak.
`final`	A túlcsordulások nem kapnak jelzést.
`oneside`, `twoside`	Egyoldalas ill. kétoldalas szedés.
`openright`	A fejezetek páratlan oldalon kezdődnek.
`openany`	A fejezetek páratlan vagy páros oldalon is kezdődhetnek.
`onecolumn`, `twocolumn`	Egyhasábos ill. kéthasábos szedés.
`notitlepage`	A dokumentum címe nem kerül külön oldalra.
`titlepage`	A dokumentum címe külön oldalra kerül.
`leqno`	A képletek számozása bal oldalra kerül.
`fleqno`	A képletek balra rendezve kerülnek szedésre.

5.3. táblázat. Dokumentumosztályok kapcsolói

5.1.3. Szavak, bekezdések

Egyszerű szövegek gépelésekor is ismernünk kell hogyan kezeli a LaTeX a sorokat, szóközöket és bekezdéseket.

szóköz A *szóköz* a szavak elválasztására használt jel. Ha több szóközt használunk, az nem jelenti azt, hogy a LaTeX a szavakat távolabb helyezi el egymástól. Kettőnél több szóköz ugyanis pontosan olyan formában jelenik meg a nyomtatott anyagban mint egy.

újsor A szavakat nem csak szóközzel, hanem *újsor* karakterrel is elválaszthatjuk egymástól. Ha tehát a szövegszerkesztőben a sor végére érünk, akkor

\'o	ó	\‘o	ò	\^o	ô	\"o	ö
\=o	ō	\.o	ȯ	\H o	ő	\u o	ŏ
\v o	ǒ	\t oo	o͡o	\b o	o̲	\d o	ọ
\c o	ọ	\’ i{}	í	\’ I{}	Í	\oe	œ
\OE	Œ	\ae	æ	\AE	Æ		
	\o	ø	\O	Ø	\l	ł	
\L	Ł	\ss	ß	\dag	†	\ddag	‡
\S	§	\P	¶	\$	$	\&	&
\%	%	\#	#	\{	{	\}	}
_	_						

5.4. táblázat. Ékezetek, különleges jelek

két szó között lenyomhatjuk az Enter billentyűt, amely a LaTeX számára egy szóközzel egyenértékű. A LaTeX tehát nem ott kezd új sort, ahol a LaTeX állományban lenyomtuk az Enter billentyűt, hanem ott, ahol eléri a rendelkezésre álló terület végét.

Ha *új bekezdés*t kívánunk kezdeni kétszer kell lenyomnunk az Enter billentyűt. Ahol tehát kihagyunk egy üres sort az állományban, ott a LaTeX egy új bekezdést kezd. A LaTeX számára a kettőnél több Enter lenyomás pontosan annyit ér, mint a kettő, vagyis nem lehet vízszintes irányba helyet kihagyni az Enter többszöri lenyomásával.

új bekezdés

5.1.4. Betűk, jelek, ékezetek

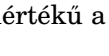
A LaTeX lehetőséget biztosít arra, hogy a különféle nyelvekben használt ékezeteket és különleges jeleket kinyomtassuk. Az 51. oldalon láttunk módszert arra, hogyan lehet felkészíteni a LaTeX-et a magyar ékezetek közvetlen használatára. Ha csak néhány betűt akarunk az adott nyelv ékezeteivel nyomtatni, akkor a 5.4 táblázatban látható rövid parancsok segítségével is előállíthatjuk ezeket.

További jelek nyomtatására használható a 71. oldalon található `\verb` parancs, amely a különleges jelek értelmezését tiltja le.

5.1.5. Dobozok és ragasztók

Ebben a fejezetben néhány olyan eszközt mutatunk be, amelyek a mindennapi munkához általában nem szükségesek, de nagymértékben megkönnyítik a LaTeX működésének megértését. Érdemes ezt a fejezetet átolvasni és az itt bemutatott eszközöket, parancsokat kipróbálni azért, hogy megértsük a program működését, „alapfilozófiáját”.

doboz
ragasztó

A LaTeX valójában minden kinyomtatandó dolgot *doboz*ként vagy *ragasztó*ként fog fel. Nem igazán foglalkozik azzal, hogy pl. az „a” betűnek milyen a pontos alakja, inkább csak az érdekli, hogy milyen magas – milyen magasra nyúlik az alapvonal fölé – milyen széles – vízszintes irányban milyen nagy helyet foglal el – és milyen mély – milyen mélyen nyúlik az alapvonal alá. A TeX és a LaTeX számára a „Szöveg.” például egynértékű a „  ” dobozsorozattal.

A rendszer a betűket tartalmazó dobozokat vízszintes ragasztókkal illeszti egymás mellé – így kapva sorokat, amelyek szintén dobozok. A „Szöveg.” például „  ” dobozként viselkedik a vízszintes összeállítás után. A soroknak megfelelő dobozokat függőleges ragasztókkal összeragasztva bekezdések, oldalak készülnek.

A ragasztók tulajdonképpen vízszintes vagy függőleges irányú üres helyek, amelyek mérete változtatható. Minden ragasztó rendelkezik összenyomhatósággal és nyújthatósággal, amely meghatározza, hogy az adott ragasztó a többi ragasztóhoz képest mennyire összenyomható ill. széthúzható. Ebben a bekezdésben például a szavak közt található szóközök olyan vízszintes ragasztók, amelyek bizonyos mértékben nyújthatóak, így lehetővé teszik, hogy a sorokat a rendszer egyforma hosszúra nyújtsa.

Amint látjuk a LaTeX automatikusan hozza létre a szöveg egyes elemeit tartalmazó dobozokat és ragasztókat, így a felhasználónak nem kell ezekkel foglalkoznia. Lehetőség van azonban arra is, hogy „kézzel” hozzon létre dobozokat és ragasztókat, így bármilyen íráskép megjeleníthető, ahogyan azt ebben a mondatban is látjuk. A következő oldalakon olyan parancsokról olvashatunk, amelyekkel dobozokat, ragasztókat hozhatunk létre.

`\dotfill`

A `\dotfill` parancs segítségével hozhatunk létre *végtelen könnyen nyújtható vízszintes ragasztó*t, pontokkal jelölve. Ez az eszköz pontsorozatként jelenik meg a nyomtatásban, szélességét pedig a LaTeX minden más ragasztónál könnyebben nyújtja. Ha a parancs környezetét vízszintes irányban nyújtani kell, akkor a `\dotfill` parancs fog nyúlni, ahogyan azt a következő példa is mutatja.

Ebben a bekezdésben elhelyeztünk egy sortörést a `\linebreak` parancsal, amely új sort kezd és ezzel arra kényszeríti a LaTeX-et, hogy a befejezetlen sort a többi sorral megegyező szélességűre nyújtsa. `\linebreak`

Ebben a bekezdésben szintén található egy `\linebreak` parancs, de itt található .....egy `\dotfill` parancs is. Látható, hogy a programnak nem kellett a szóközöket megnyújtania, mert a `dotfill` mindennél könnyebben nyúlik és a szükséges nyújtást ezen a legkönnyebb elvégezni. Felfoghatjuk a bekezdést úgy is, mint erős acélrugókkal összekapcsolt szavakat, a `\dotfill` parancsot pedig mint egy gyenge gumiszalagot.

A `\hfill` parancs segítségével végtelen könnyen nyújtható vízszintes ragasztót hozhatunnk létre, amely pontozás nélkül – „üresen” – jelenik meg. `\hfill`

A `\hrulefill` segítségével létrehozott végtelen könnyen nyújtható vízszintes ragasztót a LaTeX vonallal jelöli. `\hrulefill`

A `\hspace` meghatározott szélességű vízszintes ragasztó. A parancs formája: `\hspace{`*méret*`}`, ahol *méret* a helykihagyás mérete. A `\hspace` segítségével létrehozott helykihagyás nem nyújtható, de a nyújthatóság is megadható, ahogyan ezt a következő példa mutatja. `\hspace`

Itt kettő itt három a nyújthatóság.
Következő sor, közvetlenül a `\linebreak` után.

A `\stretch` parancs segítségével hoztuk létre a különféle mértékben nyújtható ragasztókat: `\stretch`

```
Itt kettő\hspace{\stretch{2}}itt
három\hspace{\stretch{3}}a
nyújthatóság.\linebreak
Következő...
```

Az `\addvspace` parancs segítségével meghatározott méretű függőleges ragasztót hozhatunk létre. A LaTeX mindig csak annyi üres helyet készít a parancs hatására, hogy a már meglévő üres helyet kiegészítse a kívánt méretre. A parancs formája: `\addvspace{`*méret*`}`, ahol *méret* a helykihagyás mérete. `\addvspace`

A `\bigskip`, `\medskip` és `\smallskip` parancsok segítségével kisebb és kisebb méretű függőleges ragasztót hozhatunk létre. A legnagyobb üres helyet a `\bigskip`, a legkisebbet a `\smallskip` hozza létre. Azok a távolságok amelyeket ezek a parancsok használnak relatívak, a dokumentumstílustól és módosítóiktól függenek. `\bigskip` `\medskip` `\smallskip`

\vfill A \vfill parancs segítségével *végtelen könnyen nyújtható függőleges ragasztó*t hozhatunk létre.

\vspace A \vspace parancs segítségével meghatározott méretű függőleges helykihagyást hozhatunk létre. A parancs formája: \vspace{*méret*}, ahol a *méret* a helykihagyás mérete.

\mbox Az \mbox segítségével olyan széles dobozt hoz létre, amely éppen alkalmas a szöveg tárolására. A doboz tartalmát a LaTeX nem tördeli sorokra!

\fbox Az \fbox parancs hatása ugyanaz mint az \mbox parancsé, de keretet is rajzol a doboz köré.

\makebox A \makebox paranccsal olyan széles dobozt hozhatunk létre, amelybe a megadott szöveg éppen elfér. Ennek a parancsnak a használatakor előírhatjuk a készítendő doboz méretét. A parancs formája

\makebox[*szélesség*][*elhelyezés*]{*szöveg*},

Az elhelyezés lehet c (középre), l (balra), r (jobbra) és s (kinyújtva). Az s használatához a doboznak nyújtható elemet is tartalmaznia kell. Ha nem adunk meg elhelyezést, akkor a LaTeX a doboz tartalmát megkísérli a doboz méretére nyújtani, ahogyan ez a következő példán is látszik:

Első 12

```
\makebox[5cm]{Első\hrulefill12}
```

\framebox A \framebox parancs hatása ugyanaz, mint a \makebox parancsé, de egy keretet is rajzol a doboz köré.

\rule A \rule parancs segítségével kitöltött téglalapokat hozhatunk létre. Vegyük észre, hogy ha a téglalap elég vékony, akkor akár vonalként is felfoghatjuk, ezért ezzel a paranccsal a szövegben függőleges és vízszintes vonalakat is készíthetünk. A parancs formája

\rule[*eltolás*]{*szélesség*}{*magasság*},

ahol a *szélesség* a téglalap szélessége, a *magasság* a téglalap magassága, az elhagyható *eltolás* pedig a doboz távolsága az alapvonaltól. Az alapvonaltól mért távolság lehet negatív is, ekkor a doboz az alapvonal alá nyúlik.

A \rule{2mm}{3mm} parancs például a ▮ téglalapot hozza létre, míg a \rule[0.5mm]{2mm}{3mm} parancs eltolva, ▮ formában.

\parbox A \parbox parancs segítségével egész bekezdéseket helyezhetünk el dobozban. A következő példa bemutatja a \parbox parancs használatát:

Ez egy keskeny bekezdés a \parbox parancs argumentumaként.
Ez egy újabb bekezdés, szintén az argumentumban.

A példában a \parbox parancsnak megadtuk, hogy milyen széles legyen az elkészített doboz:

```
\parbox{5cm}{Ez egy keskeny...

             Ez egy újabb...
            }
```

A \parbox parancsot nem tárgyaljuk részletesen, mivel az argumentumában meglehetősen kevés eszközt használhatunk. Szerencsésebb helyette a következő bekezdésekben tárgyalt minipage környezetet használni.

minipage

A minipage környezet segítségével szintén bekezdéseket helyezhetünk dobozokba, de itt a környezeten belül bármilyen formázóparancsot használhatunk. A következő példa két minipage környezetben szedett szövegrészletet mutat:

Tartalomjegyzék

1. A minipage környezet

Ez a szöveg 5cm széles minipage környezetben van. Ebben a környezetben szabadon használhatjuk a különféle szövegformázó eszközöket.

A \parbox parancs argumentumában meglehetősen sok eszközről le kell mondanunk, de ezek itt használhatóak.

1.1 A lábjegyzetek

Igen érdekes a lábjegyzetek kezelése[a], amelyet itt is láthatunk. Figyeljük meg, hogy a lábjegyzetek betűkkel vannak jelölve, hogy ne keveredjenek össze a minipage környezeten kívüli lábjegyzetekkel.

[a] A környezet által létrehozott dobozban kerülnek elhelyezésre.

A példa a különféle szövegdobozok és ragasztók felhasználását mutatja be, de tudnunk kell, hogy ilyen kicsiny oldalak szedésére más – könnyebben

használható és hatékonyabb eszközök is elérhetőek. A példa a következő parancsok segítségével készült:

```
\bigskip
\begin{minipage}{5cm}
\hspace{-1em}\textbf{Tartalomjegyzék}
\smallskip\footnotesize

\hbox to \textwidth{\textit{A minipage
      környezet}\enspace\hrulefill\enspace2}
\hbox to \textwidth{~A
      lábjegyzetek\enspace\hrulefill\enspace2}
\vspace{5.8cm}

\end{minipage}\hfill
\begin{minipage}{5cm}
 \hspace{-1em}\textbf{1. A minipage környezet}
 \smallskip\footnotesize

 Ez a szöveg 5cm széles \texttt{minipage}
 környezetben van. Ebben a...

 \hspace{1em}A \texttt{\bs parbox} parancs
 argumentumában meglehetősen...

 \medskip
 \hspace{-1em}{\small\bf 1.1 A lábjegyzetek}
 \smallskip

 Igen érdekes a lábjegyzetek kezelése\footnote{A
 környezet által létrehozott dobozban kerülnek
 elhelyezésre.}, amelyet itt is láthatunk. Figyeljük
 meg, hogy a lábjegyzetek...
\end{minipage}
\bigskip
```

Parancs	*Eredmény*
`\textit{Kurzív 123}`	*Kurzív 123*
`\textsl{Döntött 123}`	*Döntött 123*
`\textup{Nem döntött 123}`	Nem döntött 123
`\textmd{Nem kövér 123}`	Nem kövér 123
`\textbf{Kövér 123}`	**Kövér 123**
`\textsc{Kiskapitális 123}`	KISKAPITÁLIS 123
`\texttt{Írógép 123}`	`Írógép 123`
`\textrm{Talpas 123}`	Talpas 123
`\textsf{Sans seriff 123}`	Sans seriff 123
`\textnormal{Normál 123}`	Normál 123

5.5. táblázat. Betűváltozatok előállítása

5.2. Alapvető szerkezetek előállítása

5.2.1. Betűváltozatok és méretek

A leggyakrabban használt parancsok a betűformák beállítására vonatkoznak, ezeket ugyanis folyamatosan használjuk a szöveg egyes elemeinek kiemelésére.

A `\textit` parancs paraméterét *kurzívan szedi*. Ezt a betűváltozatot rövid szövegek kiemelésére használhatjuk. `\textit`

A `\textbf` parancs a szöveget **vastagítva** (kövér v. félkövér betűvel) szedi. Ezt a betűváltozatot általában címekben használjuk. `\textbf`

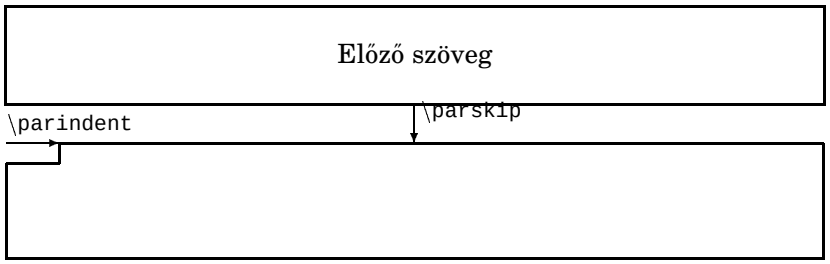
A `\textsl` parancs a paraméterét *döntött betűkkel* szedi, ezt a betűváltozatot általában hosszabb szövegek kiemelésére használjuk. Ilyen lehet pl. több bekezdésből álló idézet. `\textsl`

A `\textsf` a paraméterét talpnélküli betűkkel szedi. Ezt a betűalakot általában nem használjuk hosszabb szöveg szedésére, mert fárasztó az olvasása. `\textsf`

A `\texttt` a szöveget `írógép betűvel` szedi. Ezt a típust használhatjuk pl. a számítógép által a képernyőre írt szövegrészek szedésére. `\texttt`

A `\textsc` parancs segítségével KIS KAPITÁLIS betűket állíthatunk elő. Ezt az írásképet általában csak nevek szedésére használjuk. `\textsc`

A betűváltozatokat egymásba ágyazva is használhatjuk. Az egymásba ágyazott `\textit{\textbf{...}}` forma például ***kurzív félkövér írásképet ad***.



5.1. ábra. Bekezdések méretei

Ha különféle aláhúzásokat szeretnénk a szövegben használni, akkor hasznos az `ulem` csomag használata, `\usepackage{ulem}` paranccsal, mivel ez az egyszerű aláhúzáson kívül több változatot is tartalmaz, ahogyan a következő bekezdésben láthatjuk.

`\underline` Az `\underline` paranccsal elérhető egyszerű aláhúzáson kívül használhatunk dupla aláhúzást a `\uuline` paranccsal, hullámos aláhúzást az `\uwave` paranccsal, a szöveget ~~kihúzhatjuk~~ a `\sout` segítségével. A `\xout` parancs segítségével a szöveget ~~változással~~ is kihúzhatjuk.

`\uuline`
`\uwave`
`\sout`
`\xout`

5.2.2. A bekezdések formája

A LaTeX alapesetben a bekezdések első sorait kissé beljebb kezdi, kivéve a szakasz – fejezet, alfejezet, stb. – első bekezdését. Ennek a bekezdésnek az első sora nincs beljebb szedve, mivel ez az első bekezdés ebben a szakaszban, azokat pedig tompa szedéssel szedtük.

Ez a második bekezdés, ennek az első sora egy kissé beljebb van szedve. Az első bekezdéseken kívül az összes bekezdés alapértelmezésben beljebb van szedve néhány milliméterrel.

`\noindent` A `\noindent` parancs hatására az aktuális bekezdés első sora nem fog beljebb kezdődni. Ezt a parancsot a bekezdés elején kell kiadnunk.
Ebben a bekezdésben az első sor egy vonalba került a többivel, mivel a `\noindent` parancs megtalálható a bekezdés elején:

```
\noindent Ebben a bekezdésben...
```

Ha egy bekezdésnek az első sorát beljebb kívánjuk helyezni – holott az normális esetben nem kerülne beljebb, akkor az `indent` parancsot kell elhelyeznünk a bekezdés elején.

A LaTeX esetében a bekezdések alapesetben sorkizárt szedéssel készülnek, vagyis a bekezdés sorainak eleje és vége is egy vonalba kerül, ahogyan ebben a bekezdésben is megfigyelhető. A program ezt a nyomtatási képet úgy éri el, hogy a rövidebb sorokban található betűket egymástól távolabb helyezi el. Ez a ritkított szedés szerencsés esetben nem zavaró az olvasó számára.

Ez a bekezdés a `flushleft` környezet miatt balra igazítva jelenik meg. Az összes sor bal oldalon egy vonalban kezdődik, a sorok vége nem kerül egy vonalba. Ez a szedési mód egymással pontosan megegyező nagyságú szóközöket eredményez.

```
\begin{flushleft}
Ez a bekezdés a...
\end{flushleft}
```

5.2.3. A lap formája

A margók méreteit, a szedéstükör magasságát különféle LaTeX változók értékének beállításával módosíthatjuk. A 5.2 ábrán láthatjuk melyek a páros oldal jellemző méretei. A páratlan oldal is nagyon hasonlóan épül fel, az `even` (páros) kezdetű távolságneveket azonban ott cserélnünk kell `odd` (páratlan) kezdetre.

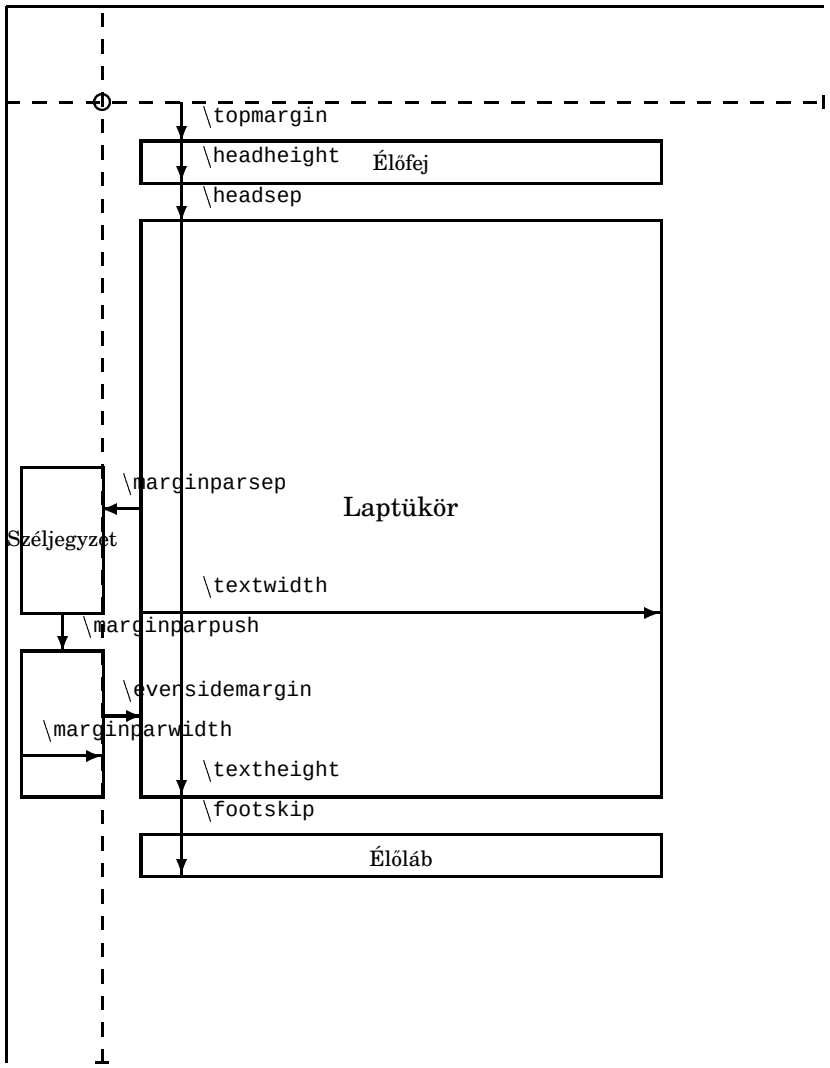
`anysize`
`\papersize`
`\marginsize`

Az `anysize` csomag betöltésével elérhetővé válik a `\papersize` parancs, amelynek a papír magasságát és szélességét adhatjuk meg. Szintén ez a csomag hozza létre a `\marginsize` parancsot, amelynek a bal, jobb, felső és alsó margók szélességét kell megadnunk. A következő példa a preambulumban beírt három sort mutatja, amelyek a csomag használatát mutatják be:

```
\usepackage{anysize}
 \marginsize{1mm}{3mm}{2mm}{4mm}
 \papersize{297mm}{210mm}
```

Amint látjuk a csomag segítségével igen egyszerűen és könnyen állíthatjuk be a oldal néhány jellemző méretét.

A szaggatott vonalak (\hoffset + 1 inch) és (\voffset + 1 inch) távolságra vannak a bal felső saroktól.



5.2. ábra. Páros oldal méretei

1) `\part{...}`	5) `\subsubsection{...}`
2) `\chapter{...}`	6) `\paragraph{...}`
3) `\section{...}`	7) `\subparagraph{...}`
4) `\subsection{...}`	

5.6. táblázat. A címeket létrehozó parancsok

5.2.4. Címek és tartalomjegyzék

A LaTeX hierarchikusan felépített címrendszerrel van felszerelve. A címeket létrehozó parancsokat (fontossági sorrendjük feltüntetésével) a 5.6 táblázatban soroltuk fel. A táblázatban található `...` helyére kerül a cím szövege.

A LaTeX a címeket hierarchikus rendszerben számozza és automatikusan elhelyezi a tartalomjegyzékben. Minden címet létrehozó parancsnak van egy csillaggal jelölt párja (pl. `\chapter*`), amely olyan címet hoz létre, amely nem kap számot és nem kerül a tartalomjegyzékbe.

Ha nem a teljes címet szeretnénk elhelyezni a tartalomjegyzékben, akkor a címet létrehozható parancsnak elhagyható paraméterként megadhatjuk, hogy mi kerüljön a tartalomjegyzékbe. Ezt a `\part[`*tart*`]{`*cím*`}` formában tehetjük meg, ahol a *tart* helyére a tartalomjegyzékben megjelenő szöveg kerül, a *cím* helyére pedig a szövegben címként megjelenő szöveg.

A címek számozásának mélysége – az, hogy a hierarchia mely tagjai kapjanak számot – a `secnumdepth` változó segítségével könnyedén beállítható. Ezt a `\setcounter{secnumdepth}{`*n*`}` formában tehetjük meg, ahol *n* a legmélyebb cím száma, amely még számozást kap. A tartalomjegyzékbe bekerülő címek hasonlóképpen határozhatóak meg a `tocdepth` beállításával.

`secnumdepth`

`tocdepth`

A `\tableofcontents` parancs helyére a LaTeX a tartalomjegyzéket illeszti be. Tudnunk kell, hogy a tartalomjegyzék frissítéséhez a `latex` parancsot kétszer kell lefuttatnunk.

`\tableofcontents`

A tartalomjegyzékhez hasonlóan a `\listoffigures` paranccsal az ábrák jegyzékét, a `\listoftables` paranccsal pedig a táblázatok jegyzékét szúrhatjuk a dokumentumba.

`\listoffigures`

`\listoftables`

5.2.5. Felsorolás és számozott felsorolás

Felsorolásokat az `itemize` környezetben készíthetünk. A környezetben a felsorolás pontjait az `\item` paranccsal kell jelölnünk. A következő néhány

`itemize`

`\item`

sor egy felsorolást tartalmaz.

- A felsorolás első bekezdése. Ez a bekezdés egy külön pontot alkot, ez a felsorolás első pontja.

  A felsorolás második bekezdése. Ez a bekezdés nem alkot külön pontot, az első pont második bekezdése.

- A felsorolás harmadik bekezdése, amely a második pont.

felsorolás

Amint láthatjuk a *felsorolás* egy pontjai elé címkék kerültek. A címkék minden pont előtt azonosak, ezért felsorolást akkor kell készítenünk, ha az egyes pontok sorrendjét nem akarjuk hangsúlyozni. A fenti példa a következő formában készült.

```
\begin{itemize}
 \item A felsorolás első bekezdése. Ez a...

       A felsorolás második bekezdése. Ez...
 \item A felsorolás harmadik...
\end{itemize}
```

Láthatjuk, hogy a felsorolás bizonyos sorait beljebb írtuk, hogy a szövegszerkesztőben könnyebben lehessen követni a szöveg szerkezetét. Szeretnénk hangsúlyozni, hogy ez az elrendezés nincs hatással a nyomatott formára, de mindenképpen fontos a szerző számára, hiszen így képes átlátni a szöveget.

Látható az is, hogy két bekezdésből áll az első pont. Sokszor van szükségünk arra, hogy a pontokba több bekezdést készítsünk, amikor a pontokon belül szeretnénk olvashatóan elrendezni a mondanivalónkat.

`enumerate`

A következő példa a `enumerate` környezetben készült, amely az előzőhöz hasonló formájú, de azzal ellentétben a pontok sorrendjét számozással hangsúlyozva *számozott felsorolás*t ábrázol.

számozott felsorolás

1. A számozott elsorolás első bekezdése. Ez a bekezdés egy külön pontot alkot, ez a felsorolás első pontja.

   A számozott felsorolás második bekezdése.

2. A számozott felsorolás harmadik bekezdése, amely a második pont.

A számozott felsorolás ugyanolyan módon készül, mint a felsorolás, csak az `itemize` környezetet kell lecserélnünk `enumerate` környezetre.

```
\begin{enumerate}
 \item A számozott felsorolás első bekezdése. Ez...

       A számozott felsorolás második bekezdése...
 \item A számozott felsorolás harmadik bekezdése...
\end{enumerate}
```

egymásba ágyazott felsorolások

Sokszor van szükségünk *egymásba ágyazott felsorolások*ra, amelyek nem csak pontokat, hanem alpontokat (esetleg al-alpontokat) tartalmaznak. A `enumerate` és `itemize` környezetek tetszőleges sorrendben egymásba ágyazhatóak, ahogyan a következő példa mutatja.

1. A számozott elsorolás első bekezdése. Ez a bekezdés egy külön pontot alkot, ez a felsorolás első pontja.

   A számozott felsorolás második bekezdése. Ez a bekezdés nem alkot külön pontot, az első pont második bekezdése.

   (a) Az első pont első alpontja. Amint látjuk az egymásba ágyazás tökéletesen működik.

   (b) A belső számozás második pontja.

2. A számozott felsorolás harmadik bekezdése, amely a második pont.

A példát két `enumerate` környezet egymásba ágyazásával készítettük. A parancsok szerkezete a következő volt:

```
\begin{enumerate}
  \item A számozott...

      A számozott felsorolás...
  \begin{enumerate}
    \item Az első pont...
    \item A belső számozás második pontja.
  \end{enumerate}

  \item A számozott felsorolás...
end{enumerate}
```

Figyeljük meg, hogy a sorok beljebb kezdése hogyan segíti a szöveg olvasását! Egymásba ágyazott felsorolások esetén igen fontos, hogy megfelelő

módon elrendezzük a szövegszerkesztőben a szöveget, ellenkező esetben nem látjuk át az állomány szerkezetét.

pifont A `pifont` csomag (117. oldal) újabb eszközöket biztosít felsorolások és számozott felsorolások készítésére. A következő szöveg is így készült.

- ☞ Töltsük be a `pifont` csomagot.
- ☞ Válasszunk ki egy jelet a ZapfDingbats fontkészletből.
- ☞ Használjuk a `dinglist` környezetet.

dinglist Az ehhez hasonló szerkezetek előállítására a `dinglist` környezet használható. A `dinglist` környezetnek egy paramétere van, amely megadja, hogy a ZapfDingbats szimbólumok közül melyiket kívánjuk címkeként használni. A 118. oldalon található 8.1 táblázat tartalmazza a szimbólumkészletet. A példa a 43-as szimbólumot használja, a következőképpen készült.

```
\begin{dinglist}{43}
 \item Töltsük be a \texttt{pifont} csomagot.
 \item Válasszunk ki egy jelet a ZapfDingbats
       fontkészletből.
 \item Használjuk a \texttt{dinglist} környezetet.
\end{dinglist}
```

dingautolist A `dingautolist` környezet – amelyet szintén a `pifont` csomag hoz létre – alkalmas ZapfDingbats szimbólumokkal létrehozott számozott felsorolás készítésére. A következő példa ilyen felsorolást ábrázol.

❶ Töltsük be a `pifont` csomagot a preambulumban.

❷ Válasszunk ki egy jelsorozatot a ZapfDingbats fontkészletből.

❸ Használjuk a `dingautolist` környezetet a megfelelő kezdőkóddal.

A `dingautolist` környezet ugyanolyan formában használható, mint a `dinglist`, de ez utóbbi minden ponthoz a sorban következő ZapfDingbats szimbólumot használja. A példa a következő formában készült:

```
\begin{dingautolist}{182}
 \item Töltsük be a...
 \item Válasszunk ki...
 \item Használjuk a...
\end{dingautolist}
```

Egyéni formájú felsorolásokat legegyszerűbben úgy készíthetünk, hogy az `\item` parancsnak `[]` jelek közt megadjuk milyen címkét szeretnénk a pont előtt látni. A következő felsorolás is így készült:

- első pont
- második pont

A felsorolás a következő parancsok felhasználásával készült:

```
\begin{enumerate}
 \item[--] első pont
 \item[--] második pont
\end{enumerate}
```

Ha a felsorolás egy pontjai elé hosszabb szöveget szeretnénk írni címkeként, akkor azt is az `\item` paramétereként kell megadnunk. A következő példa is így készült:

`enumerate` számozott felsorolás

`itemize` számozás nélküli felsorolás

Amint látjuk a címkék erőteljesen kilógnak a margóba, mivel túl hosszúak ahhoz, hogy elférjenek. A fenti példa hibás nyomtatási képet eredményez, nyomdai szempontból elfogadhatatlan. (Ezúton kérem a nyomdai ellenőrzést végző kollégát, hogy nézze el ezt a borzalmat most az egyszer.)

`description`

A `description` környezet éppen ilyen szerkezet készítésére szolgál. A következő példa ebben a környezetben készült:

`enumerate` Számozott felsorolás készítésére szolgáló környezet. A sorrend hangsúlyozásakor használjuk.

`itemize` Számozás nélküli felsorolás, amelyet akkor használunk, ha a sorrend nem fontos.

`description` Leíró jellegű felsorolások készítésére használt környezet. Képes kezelni a hosszú címkéket is.

Amint látjuk a környezet lehetővé teszi, hogy a címke „beljebb nyomja” a szöveget, ha szükséges. Így a pontok első sora nem egyvonalban kezdődik, de a címkék nem lógnak ki. Az is látható a a példában, hogy a `description` környezet a címkéket automatikusan félkövér betűkkel szedi. A példa a következőképpen készült:

```
\begin{description}
 \item[\texttt{enumerate}] Számozott felsorolás...
 \item[\texttt{itemize}]   Számozás nélküli felsorolás...
 \item[\texttt{description}] Leíró jellegű...
\end{description}
```

Megoldást jelenthet a kilógó címkékre a lista méretezésének megváltoztatása. Ezzel a módszerrel készült a következő példa, ahol a teljes felsorolást kissé beljebb szedtük.

`enumerate` Számozott felsorolás készítésére szolgáló környezet. A sorrend hangsúlyozásakor használjuk.

`itemize` Számozás nélküli felsorolás, amelyet akkor használunk, ha a sorrend nem fontos.

`list` A felsorolásokat testreszabni a `list` környezet felhasználásával lehet. A `list` környezet kimondottan arra készült, hogy a segítségével saját környezetet hozzunk létre.

`\newenvironment` Új környezetet a `\newenvironment` parancs segítségével hozhatunk létre. A parancs formája:

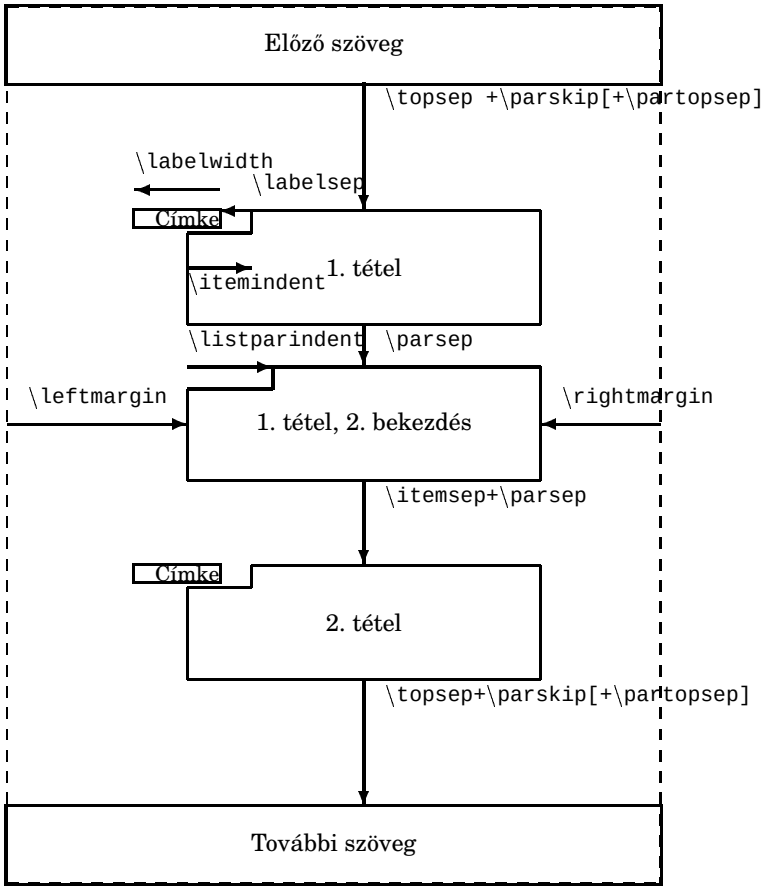
`\newenvironment{`*környezet*`}{`*elején*`}{`*végén*`}`

Itt a *környezet* a létrehozandó környezet neve, az *elején* a környezet használatakor a környezet elején végrehajtandó parancsokat tartalmazza, a *végén* pedig a környezet végén kiadandó parancsok listája.

A következő példa bemutatja hogyan kell a felsorolást testreszabni (új környezetet létrehozni) a `\newenvironment` paranccsal a `list` környezet alapján.

```
\newenvironment{mylist}{
                        \begin{list}
                              {$\diamondsuit$}
                              { \parsep=0pt
                                \itemsep=0pt
                                \leftmargin=2em
                                \rightmargin=2em
                                \topsep=1ex
                              }
                       }
                       {\end{list}}
```

5.3. ábra. A `list` környezet méretezése



Amint láthatjuk az új környezet neve `mylist`. A környezet használatakor a környezet elején a `list` környezetre térünk át, a környezet végén pedig a `list` környezetnek vége lesz.

A `list` környezet két paramétert fogad, melyek formája:

`\begin{list}{`*címke*`}{`*parancsok*`}`

Itt a *címke* az újonnan létrehozott felsorolásban alapértelmezés szerint használt címke. A *parancsok* helyén a létrehozott felsorolás formáját (méreteit) meghatározó parancsokat kell felsorolnunk. A használható parancsokat a 69. oldalon található 5.3 ábra mutatja be.

Ha a fenti módszerrel létrehozzuk az új környezetet, akkor azt az `enumerate` és az `itemize` környezeteknél látott módon tudjuk használni:

```
\begin{mylist}
 \item első
 \item második
\end{mylist}
```

Ha nem akarunk új környezetet létrehozni, akkor használhatjuk a következő – egyszerűsített – formát is:

```
\begin{list}
      {$\diamondsuit$}
      { \parsep=0pt
        \itemsep=0pt
        \leftmargin=2em
        \rightmargin=2em
        \topsep=1ex
      }
 \item A list...
 \item Az első paraméter...
 \item A második paramétert...
\end{list}
```

A fenti parancsok által létrehozott felsorolás alapértelmezés szerinti címkékkel a következő formában jelenik meg:

- ♢ A `list` környezet a kiindulópontja a saját felsorolásformátum készítésének.

◊ Az első paraméter az az alapértelmezés szerinti jel, amely a felsorolás egyes pontjai elé kerül, ha az `\item` parancsnál mást nem adunk meg.
◊ A második paramétert a méretezésre szolgáló parancsok adják.

5.2.6. Lábjegyzetek, végjegyzetek és széljegyzetek

A széljegyzet a margón található rövid szöveg, amely segíti az olvasót a szövegben való eligazodásban. A széljegyzetet a `\marginpar` parancs segítségével készíthetjük el. `\marginpar`

Ebben a bekezdésben egy rövid széljegyzet található, amely mindössze egy szóból áll. A széljegyzetet kurzív betűvel szedtük, hogy elüssön a szövegtől és az olvasót ne zavarja. A széljegyzet a következőképpen készült: *Széljegyzet*

```
Ebben a bekezdésben \marginpar{\textit{Széljegyzet}}...
```

Valójában a `\marginpar` parancs elfogad két paramétert is, teljes formája a következő:

`\marginpar[`*bal oldalra*`]{`*jobb oldalra*`}`

A parancs a *bal oldalra* szöveget írja a margóra, ha az adott oldalon a széljegyzetek a bal margóra kerülnek, különben pedig a *jobb oldalra* szöveget használja.

A széljegyzetek jobb oldalon jelennek meg alapesetben, kétoldalas nyomtatás esetén pedig mindig a külső oldalra. A `\reversemarginpar` parancs az elhelyezést megfordítja, a `\normalmarginpar` pedig az alapértelmezésre állítja vissza. `\reversemarginpar` `\normalmarginpar`

A `\marginparwidth` értéke adja a széljegyzetek maximális szélességét, a `\marginparsep` a vízszintes távolságot a kenyérszövegtől, a `\marginparpush` pedig a minimális függőleges távolságot két széljegyzet közt. `\marginparwidth` `\marginparsep` `\marginparpush`

5.2.7. Gépelt szöveg

A LaTeX a `\verb` parancs segítségével lehetőséget ad azoknak a jeleknek a kinyomtatására, amelyeket egyébként különlegesen kezel, speciális jelentéssel ruház fel. Ebben a bekezdésben pl. a következőképpen szedtük a parancsot: `\verb`

```
A \LaTeX{} a \verb.\verb. parancs...
```

A \verb parancs használata kissé eltér a szokványos parancsoknál megszokottaktól. Járjunk el a következőképpen:

1. Válasszunk egy írásjelet, amely nem szerepel a szövegben, amelyet változtatás nélkül szeretnénk kinyomtatni.
2. Helyezzük el ezt a jelet a \verb parancs után.
3. Gépeljük be a változtatás nélkül kinyomtatandó szöveget.
4. Zárjuk le a parancssor a \verb után írott írásjellel.

verbatim

A verbatim környezet hasonló célokra használható. Segítségével hosszabb szövegeket idézhetünk formázás nélkül a dokumentumban:

```
\begin{verbatim}
A szöveg...
\end{verbatim}
```

verbatim*
\verb*

A gépelt szöveg szedésére használhatjuk a verbatim* környezetet és a \verb* parancsot is. Ezek annyiban különböznek a bemutatott eszközöktől, hogy a szóközöket is jelölik. A következő szöveg a szóközök jelzését mutatja be:

```
int␣main(␣int␣argc,␣char␣*argv[]){

}
```

alltt

Az alltt csomag által létrehozott alltt környezet nagyon hasonló a verbatim környezethez. A különbség csak annyi, hogy a {} és \ jelek különleges jelentése megmarad, így a környezeten belül használhatunk LaTeX parancsokat is. A következő példa ilyen környezetben készült:

```
Az alltt olyan mint a verbatim, de használhatunk
parancsokat is a szövegben.
```

fancyvrb
Verbatim

A fancyvrb csomag betöltése után használhatjuk a Verbatim csomagot, amellyel fejletteb módon tudunk példaprogramokat szedni a verbatim környezethez hasonlóan. A következő példa is ezzel az eszközzel készült:

Példa

```
#! /bin/bash -login

VARIABLE=.bashrc
```

A keretet és a címet a következő utasítások hozták létre:

```
\begin{Verbatim}[frame=single,
                 label=Példa]
#! /bin/bash -login

VARIABLE=.bashrc
\end{Verbatim}
```

A következő felsorolás tartalmazza a `Verbatim` környezet opcionális paraméterében használható kulcsszavak jelentését:

`commentchar=` *betű* Azok a sorok, amelyek ezzel a betűvel kezdődnek nem jelennek meg a nyomtatásban.

`gobble=` *szám* Minden sor első *szám* számú karaktere nem jelenik meg a nyomtatásban.

`formatcom=` *parancs* LaTeX parancs vagy parancsok, amelyek a gépelt szöveg szedése előtt hajtódnak végre.

`fontfamily=` *fcs* A szedésre használt betűcsalád. Használhatjuk a `tt`, `courier` `helvetica` kulcsszavakat.

`fontsize=` *méret* A gépelt szöveg betűmérete. A *méret* lehet `small`, `tiny` stb. Az alapértelmezés szerinti érték az `auto`, ami az aktuális fontméretet jelenti.

`fontshape=` *vált* A betűváltozat, pl. `it` a kurzív betű jelölésére.

`fontseries=` *súly* A betű vastagsága, pl. `b` a kövér betű jelölésére.

`frame=` *keret* Azt adja meg, hogy a gépelt szöveg köré milyen szegélyt rajzoljon. Ez lehet `none` (nincs keret) `leftline` (vonal bal oldalon), `topline` (vonal fent), `bottomline` (vonal lent), `lines` (vonal fent és lent), `single` (keret körbe)

`framerule=` *méret* A szegély vastagsága. Ez alapértelmezésben `0.4pt`.

`framesep=` *méret* A szegély és a közrezárt szöveg távolsága.

`rulecorol=` *szín* A szegély színe, pl. `black`, `red`.

`fillcolor=` *szín* A szegély és a szöveg közti terület színe.

`label=` *szöveg* A cím szövege. Ha a címet fent és lent is meg akarjuk jeleníteni, akkor használhatjuk a `label={[`*fent*`]`*lent*`}` formát.

`labelposition=` *hely* A cím elhelyezkedése, lehetséges értékei; `none` (nincs címke), `topline` (fent), `bottomline` (lent), `all` (fent és lent).

`numbers=` *num* A sorok automatikus számozásának elhelyezkedése, amely lehet `none` (nincsenek számok), `left` (számok bal oldalon) `right` (számok jobb oldalon).

`numbersep=` *méret* A távolság a sorszámok és a szöveg között.

`firstnumber=` *szám* Azt határozza meg, hogy honnan induljon a számozás, mekkora legyen az első sor sorszáma. Ez lehet `auto` (első szám az 1), `last` (ez előző környezetből folytatólagosan), `13` (konkrét kezdőszámtól).

`stepnumber=` *szám* Egész szám, amely meghatározza, hogy mely soroknál jelenjen meg sorszám. Ha ez pl. 2, akkor minden második sor mellé kerül szám.

`numberblanklines=` *log* Logikai érték (`true`/`false`), amely megadja, az üres sorok mellé is kerüljenek-e számok.

`firstline=` *szám* Az első sor, amit már nyomtatni kell.

`lastline=` *szám* Az utolsó sor, amit még nyomtatni kell.

`showspaces=` *log* Logikai érték (`true`/`false`), amely meghatározza, hogy a szóközök megjelenjenek-e „␣” jelként szövegben.

`showtabs=` *log* Logikai érték (`true`/`false`), amely meghatározza, hogy a tabulátorjelek megjelenjenek-e a szövegben.

`obeytabs=` *log* Logikai érték (`true`/`false`), amely meghatározza, hogy a tabulátorkaraktereket a program figyelmen kívül hagyja-e.

`tabsize=` *szám* Szám, amely meghatározza, hogy a tabulátorkarakterek hány szóköz szélességű tabulátorpozíciókon jelenjenek meg.

`baselinestretch=` *parancs* A sorok közti függőleges távolságot meghatározó parancsok.

`commandchars=` *betűk* Azok a karakterek, amelyeket a környezeten belül LaTeX parancsokat jelölnek. Ha ennek értéke pl. `\\\{\}`, akkor a szokásos módon használhatunk LaTeX parancsokat a szövegben. A `commandchars` alapértelmezés szerint üres, LaTeX parancsokat tehát nem használhatunk a környezetben.

`xleftmargin=` *méret* A környezet és a bal margó közt található üres hely mérete.

`xrightmargin=` *méret* A környezet jobb széle és a jobb margó közt fennmaradó üres hely szélessége.

`resetmargins=` *log* Logikai érték (`true`/`false`), amely meghatározza, hogy hogy bal oldalon a behúzást visszaállítsuk-e a környezet kezdetekor. Ha igaz, akkor pl. a felsorolásban elhelyezett `Verbatim` környezet nem kerül beljebb.

`samepage=` *log* Logikai érték (`true`/`false`), amely meghatározza, hogy az egész környezetet egy oldalon szeretnénk-e látni. Ha értéke igaz, a LaTeX nem szakítja meg a környezetet laptöréssel.

Az automatikus számozást a `\theFancyVerbLine` parancs hozza létre a `FancyVerbLine` számláló megfelelő kiíratásával. Ezt a parancsot újraírva a számozást átalakíthatjuk. Ennek formája pl. a következő lehet:

`\theFancyVerbLine`
`FancyVerbLine`

```
\renewcommand{\theFancyVerbLine}{
  \arabic{FancyVerbLine}
}
```

A következő példa is ezt az egyszerű módszert használja, így az alapértelmezett kis méretű számok helyett normál méretű számok találhatóak a sorok előtt:

Makefile részlet

```
1
2   calculator: y.tab.o lex.yy.o $(OBJECTS)
3           $(CC) -o $(TARGET) y.tab.o \
4           lex.yy.o $(OBJECTS) $(LIBRARIES)
5
6   clean:
7           rm $(TARGET)
8
```

Makefile részlet

Figyeljük meg, hogy a tabulátorok és a szóközök ábrázolása olyan információkat is az olvasó elé tár, amelyek nyomtatásban általában nem jelennek meg, de az adott gépi nyelv megértésének szempontjából elengedhetetlenül fontos. A példát következőképpen készítettük:

```
\begin{Verbatim}[frame=lines,
                 label={[Makefile \textrm{részlet}]
                        Makefile \textrm{részlet}},
                 numbers=left,
                 showtabs=true,
                 showspaces=true,
                 xleftmargin=15mm]

calculator: y.tab.o lex.yy.o ...
                             ...
                             ...
\end{Verbatim}
```

listings
syntax highlight

A `listings` csomag segítségével a különféle gépi nyelveknek megfelelő kiemeléseket (*syntax highlight*) készíthetjük el automatikusan.

5.3. Táblázatok

tabular

A `tabular` környezet táblázatok szedésére használható. A következő néhány oldalon ennek a környezetnek és továbbfejlesztett változatainak használatáról esik szó.

5.3.1. Egyszerű táblázatok

A következő táblázat keretek nélkül jelenik meg, de ettől függetlenül táblázat. Figyeljük meg, hogy az első oszlop balra, a második középre, a harmadik pedig jobbra van rendezve!

balra	középre	jobbra
bal oldalra	középre	jobb oldalra

A táblázat a következő parancsok segítségével készült:

```
\begin{tabular}{lcr}
balra       & középre & jobbra       \\
bal oldalra & középre & jobb oldalra \\
\end{tabular}
```

Amint megfigyelhetjük a táblázat oszlopait a & jellel, sorait pedig a \\ jelekkel kell jelölnünk. Gondosan kell ügyelnünk ezekre a jelekre, mert a legkisebb hiba is teljesen rossz nyomtatási képet eredményez.

Az is megfigyelhető a példán, hogy a `tabular` környezet egy paramétert fogad. Ez a paraméter az oszlopok jellegének leírását adja, a példában `{lcr}`. Itt az `l` betű balra rendezést, a `c` betű középre rendezést, az `r` betű pedig jobbra rendezést eredményez az adott oszlopban.

Ha hosszú szöveget írunk a táblázatba, akkor az oszlop igen széles lesz. Arra gondolnánk, hogy amint eléri a táblázat szélessége a szedéstükör szélességét, a LaTeX az oszlopban található szöveget sorokra tördeli, hogy ne nyúljon a táblázat a margóba. Ez azonban nem így van. A táblázat – ha elegendő szöveget írunk a cellába – „lelóg a papírról”. Ez még akkor is így van, ha a cellába új bekezdést próbálunk készíteni. A LaTeX a táblázat cellájában figyelmen kívül hagyja a bekezdések végét, mindent egy sorban próbál elhelyezni.

Ha több bekezdést, sortördelést akarunk a táblázat valamely oszlopában használni, akkor `p{`*szélesség*`}` formában kell az oszlopot meghatároznunk. Ez a következő szedésképet adja:

Első oszlop	A második oszlop példánkban pontosan 4 cm széles, ha a szöveg nem fér el egy ilyen hosszú sorban, a LaTeX sorokra tördeli.

A táblázat a következőképpen készült:

```
\begin{tabular}{lp{4cm}}
  Első oszlop & A második...
\end{tabular}
```

Sokszor van szükségünk arra, hogy a táblázat valamely oszlopát egy adott betűhöz, jelhez igazítsuk. Általában akkor van ilyen eszközre szükségünk, amikor számokat írunk a táblázatba és szeretnénk a tizedesvessző szerint

igazítani, hogy helyiérték szerint egymás alá kerüljenek a számok. A következő táblázat ilyen szedést mutat be:

Batyu	4,5 kg
Táska	8,6 kg
Koffer	15,6 kg
Malaclopó	25,89 kg

A táblázat a következő parancsok hatására vette fel ezt a formát:

```
\begin{tabular}{lr@{,}l}
  Batyu &        4&5 kg\\
  Táska &        8&6 kg\\
  Koffer &       15&6 kg\\
  Malaclopó &    25&89 kg\\
\end{tabular}
```

Amint látjuk a `tabular` paramétereként a második oszlop rendezését a `r@{,}l` betűk határozták meg, a számokban a tizedesvesszőket pedig & jellel cseréltük le. Ez azt jelenti, hogy *a táblázat három oszlopból áll*, nem pedig kettőből, ahogyan azt gondolnánk.

A `@{,}` minden sorban elhelyezi a `,` jelet, melytől balra – a második oszlopban – az egészek jelennek meg (jobbra rendezve), balra pedig a tizedes jegyek – a harmadik oszlopban (balra rendezve). A bemutatott szerkezet használatát azért nehéz kissé megszokni, mert a `@{x}` nem rendezésre, hanem a szöveg soronként való beszúrására, oszlopelválasztó készítésére szolgál. Lássuk például az előző táblázatot továbbfejlesztve:

Batyu	4,5	kg
Malaclopó	25,89	kg

Itt megkíséreltük a mértékegységeket is elrendezni, de az eredmény sajnos nem kielégítő, mert túl nagy a távolság a mérőszám és a mértékegység közt. A táblázat az egyszerű `{lr@{,}ll}` formát használja az oszlopok meghatározására, de ez az utolsó két oszlop közt túl nagy vízszintes helykihagyást eredményez.

Batyu	4,5	kg
Malaclopó	25,89	kg

Ez a táblázat már megfelelő, nincsenek túl nagy távolságok a mértékegységek előtt. A táblázat a következőképpen készült:

```
\begin{tabular}{lr@{,}l@{ }l}
 Batyu &        4&5 & kg\\
 Malaclopó &    25&89 & kg\\
\end{tabular}
```

Amint látjuk az utolsó és utolsó előtti oszlop közé elhelyeztünk egy `@{ }` kifejezést, ami szóközt helyez el minden sorba. Azt gondolhatnánk, hogy ez növeli a két oszlop közti távolságot – és nekünk éppen a távolság csökkentése volt a célunk –, de ez nem így van. A `@{ }` kifejezés ugyanis megszünteti az oszlopok közt alapértelmezésben található oszlopelválasztó helykihagyást, amely nagyobb mint az elhelyezett szöveg.

5.3.2. Táblázatok keretezése

A táblázatokat általában szegéllyel látjuk el, ahogyan azt a következő táblázatnál láthatjuk:

balra	középre	jobbra
bal oldalra	középre	jobb oldalra

A táblázat sorai közé a `\hline` paranccsal tehetünk vízszintes vonalat, oszlopai közé pedig az oszlopmeghatározáskor használt `|` karakterrel: `\hline`

```
\begin{tabular}{||l|c|r||}              \hline
balra       & középre & jobbra       \\ \hline
bal oldalra & középre & jobb oldalra \\ \hline
\end{tabular}
```

Látható, hogy kettős vonalat a parancs és a `|` jel ismétlésével hozhatunk létre. Ezekkel az eszközökkel többszörös vonalat is létrehozhatunk.

A `\cline` paranccsal is húzhatunk vízszintes vonalat két sor közé, ennek a parancsnak azonban megadhatjuk paraméterként, hogy melyik oszlopokban kell a vonalat meghúzni. `\cline`

1.	Első név	Első cím
2.	Második név	Második cím
		Harmadik cím

```
\begin{tabular}{||r|l|l||}\hline
1. & Első név    & Első cím     \\ \cline{1-3}
```

```
2. & Második név & Második cím  \\ \cline{3-3}
   &             & Harmadik cím \\ \cline{1-3}
\end{tabular}
```

\vline
\hfill

Ha az oszlopon belül szeretnénk függőleges vonalat húzni, de nem az összes soron keresztül, akkor a `\vline` parancsot kell használnunk. A vonalat a cella széléig a `\hfill` végtelen könnyen nyújtható vízszintes ragasztóval mozgathatjuk. A következő táblázat ezzel a módszerrel készült:

Név	Indulási időpont	Beérkezési időpont

```
\begin{tabular}{|l|cc|} \hline
 Név & \hfill Indulási \hfill\vline& Beérkezési \\
     & \hfill időpont  \hfill\vline& időpont    \\ \hline
     &                             &            \\ \hline
\end{tabular}
```

Figyeljük meg, hogy az „Indulási időpont” csak úgy lehetett középre rendezett, ha az elejére elhelyeztünk egy `\hfill` parancsot. Ez azért szükséges, mert a jobb oldalán egy végtelen könnyen nyújtható `\hfill` található, amelyet ellensúlyozni kellett.

5.3.3. Oszlopok összevonása

cellák összevonása

A vázolt eszközök sokszor elégtelennek bizonyulnak, kénytelenek vagyunk egymás mellett álló cellákat összevonni. A *cellák összevonása* figyelhető meg a következő táblázatban is:

4 oszlop			
2 oszlop		2 oszlop	
1 oszlop	1 oszlop	1 oszlop	1 oszlop

```
\begin{tabular}{|l|l|l|l|}       \hline
 \multicolumn{4}{|c|}{\textit{4 oszlop}}    \\ \hline
 \multicolumn{2}{|c|}{2 oszlop} &
    \multicolumn{2}{c|}{2 oszlop}           \\ \hline
 1 oszlop & 1 oszlop & 1 oszlop & 1 oszlop  \\ \hline
\end{tabular}
```

Megnevezés	*Darabszám*
50 tillátor	50
5 ventillátor	5

5.7. táblázat. A táblázat címe

Az oszlopok összevonására használható `\multicolumn` parancs első paramétere az összevonandó cellák száma, második az elkészített cellában használt igazítás a már ismert formában, a harmadik pedig az elkészített cella tartalma. `\multicolumn`

5.4. Lebegő objektumok

A *lebegő objektumok* olyan elemek, amelyek nem pontosan a begépelés helyére kerülnek. A LaTeX a lebegő objektumokat a szövegben előre és hátra mozogva olyan helyre teszi, ahol talál elegendő szabad helyet. Lebegő objektumként általában képeket, táblázatokat, ábrákat helyezünk el, amelyekre a szövegben a számukkal hivatkozunk. *lebegő objektumok*

Lebegő objektumok elhelyezésére szolgál a `figure` (kép, ábra) és a `table` (táblázat) környezet. Ami ezeken a környezeteken belül található, az nem pontosan oda kerül ahová írtuk, de lehetőleg annak a lapnak a közelébe. `figure` `table`

A következő példa egy ilyen lebegő objektum létrehozását mutatja be:

```
\begin{center}
\begin{tabular}{l|r}
 \textit{Megnevezés} & \textit{Darabszám} \\ \hline
 50 tillátor         & 50                 \\
 5 ventillátor       &  5                 \\
\end{tabular}
\caption{A táblázat címe\label(pelda)}
\end{center}
```

A parancsok hatására létrejött táblázat a 5.7 számot kapta, megtalálható a 81. oldalon.

Amint láthatjuk a `\caption` parancs segítségével adhatunk címet a lebegő objektumnak. A `\caption` parancson belül hozhatunk létre címkét a `\label` paranccsal. A címke segítségével hovatkozhatunk a szövegben az `\caption`

adott objektum számára és oldalszámára. A `\label` parancsról és annak használatáról a 84. oldalon olvashatunk bővebben.

`floatflt`
`floatingtable`
`floatingfigure`

A `floatflt` csomag által létrehozott `floatingtable` és `floatingfigure` környezetek segítségével ábrákat, képeket és táblázatokat illeszthetünk a dokumentumba. Különlegessége ezeknek a környezeteknek, hogy lehetővé teszik a beillesztett kép vagy táblázat mellett a szöveg megjelenítését, a szöveggel való *körülfolyatás*t, ahogyan ezt a 5.8 táblázatnál látjuk.

körülfolyatás

Név	*Cím*

5.8. táblázat. Példa

Ha ezeket a környezeteket használjuk a dokumentum kisebb helyen is elférhet és sokak szerint esztétikusabb formában jelenik meg. Ügyelnünk kell azonban, mert a képek és táblázatok körülfolyatása bonyolult művelet, sokszor lehetetlen feladat elé állítja a LaTeX-et. Előfordulhat, hogy a beillesztendő kép vagy táblázat számára a program nem talál helyet és így az nem is jelenik meg a nyomtatásban. Általában akkor következik ez be, ha a kép, táblázat közvetlen környezetében olyan elemek találhatóak, amelyek szélessége nem korlátozható. Ilyenek pl. a címek, más képek és táblázatok.

Problémát okozhat az is, hogy a beillesztett elem mellett a bekezdések szélessége erősen lecsökken. Ilyenkor a sorok tördelésében a LaTeX erősen korlátozva van, ezért nem mindig sikerül a sorokra tördelés, túlcsordult vízszintes dobozok jelennek meg. Ilyen esetekben a bekezdéseket `sloppypar` környezetbe ágyazhatjuk, amely kissé csúnyább nyomtatási képet ad, de megszünteti a túlcsordulásokat. Az itt látható bekezdések elején `\sloppy` végén pedig `fussy` parancs található, így sikerült viszonylag korrekt szedésképet előállítani. A `\sloppy` és `\fussy` parancsokról a 85. oldalon olvashatunk.

A szöveggel körülfolyt ábrák, táblázatok létrehozásának alapformája a következő:

```
\begin{floatingtable}[elhelyezés]{méret}
  a táblázatot létrehozó parancsok
\end{floatingtable}
```

Fontos tudnunk, hogy ha ezt a formát használjuk, akkor a beillesztett elem szélességét ismernünk kell és a *méret* helyére be kell írnunk, különben az eredmény olvashatatlan lesz. Az *elhelyezés* helyén a következő betűk egyike állhat:

`r` A beillesztett elem jobb oldalon fog elhelyezkedni.

`l` A beillesztett elem bal oldalra kerül.

`p` A beillesztett elem mindig a külső oldalra kerül.

`v` A beillesztésnél a csomag betöltésekor megadott kapcsoló lesz az érvényes. A csomag betöltésekor a következő kapcsolókat használhatjuk:

- `rflt` Minden beillesztett elem jobb oldalra kerül.
- `lflt` A beillesztett elemek bal oldalra kerülnek.
- `vflt` Váltakozó oldalra kerülnek az elemek, mindig a külső oldalon lesznek.

A `floatingtable` és `floatingfigure` környezetek képesek az általuk elhelyezendő objektumok méretének meghatározására. Ha a táblázatot, képet a környezetnek adott paraméterként hozzuk létre, akkor a szélességet nem kell megadnunk. Ekkor a parancsokat a következőképpen kell kiadnunk.

```
\begin{floatingtable}[elhelyezés]{
  a táblázatot létrehozó parancsok
}
\end{floatingtable}
```

5.5. Keretek, szegélyek

A `fancybox` csomag betöltésével többféle keretet használhatunk a a dokumentumban. A `\shadowbox` parancs segítségével árnyékolt keretezés hozható létre. A `\doublebox` parancs dupla keretezést készíthetünk. Az `\ovalbox` parancs segítségével lekerekített sarkú keretezés hozható létre. Az `\Ovalbox` parancs használatával vastag kerekített keretezést készíthetünk.

`fancybox` `\shadowbox` `\doublebox` `\ovalbox` `\Ovalbox`

A `shadow` csomag betöltése után használhatjuk a `\shabox` parancsot. A következő bekezdés ennek a parancsnak a felhasználásával készült:

`shadow` `\shabox`

> Ez a bekezdés be van keretezve egy árnyékolt kerettel. Ezt az eszközt a `shadow` csomag betöltésével használhatjuk.

\parbox

Annak érdekében, hogy a kereten belül több sort is megjeleníthessünk a \parbox parancsot használtuk, amelyről bővebben a 56. oldalon olvashatunk:

```
\shabox{\parbox{8cm}{Ez a bekezdés be van keretezve egy
        árnyékolt kerettel. Ezt az eszközt a
        \texttt{shadow} csomag betöltésével
        használhatjuk.
 }
}
```

\sboxrule
\sboxsep
\sdim

A \shabox méretek beállítására használja az \sboxrule (keret vonalának vastagsága), \sboxsep (keret távolsága a szövegtől), és \sdim (árnyék vastagsága) parancsokat. A következő bekezdés ezeknek a távolságoknak az átállításával készült:

Ez a bekezdés be van keretezve egy árnyékolt kerettel. Ezt az eszközt a shadow csomag betöltésével használhatjuk.

```
\sdim=5pt \sboxrule=1pt \sboxsep=2pt
 \shabox{\parbox{8cm}{Ez a bekezdés... }
 }
```

5.6. Listák, hivatkozások

5.6.1. Kereszthivatkozás

\label
\ref
\pageref

A szövegben elhelyezett \label parancs segítségével címkéket hozhatunk létre, amelyekre a \ref és \pageref parancsokkal hivatkozhatunk. A szövegben a címkék nem jelennek meg, de a \pageref parancs helyére bekerül a címkét tartalmazó oldal száma, a \ref helyére pedig annak a fejezetnek, táblázatnak, képnek, képletnek a száma, amelyben az adott címke megtalálható.

Ebben a bekezdésben elhelyeztünk egy címkét, amelyre hivatkozunk is. A bekezdés a 84. oldalon található.

```
\label{pelda} Ebben a bekezdésben elhelyeztünk egy
címkét, amelyre hivatkozunk is. A bekezdés a
\pageref{pelda} oldalon található.
```

5.7. A tördelés befolyásolása

Ebben a szakaszban azokat az eszközöket, parancsokat ismertetjük, amelyeket felhasználva a LaTeX sor- és laptördelését befolyásolhatjuk, módosíthatjuk. Ezekkel a parancsokkal „finomhangolhatjuk”, hogy a program hol kezdjen új sort, új bekezdést.

A fejezet megértéséhez tudnunk kell, hogy a LaTeX a sorokat és az oszlopokat nyújtható dobozokként kezeli. A szöveg egy adott sora vízszintes irányba nyújtható dobozba helyezkedik el. Amikor a program megpróbálja eldönteni, hogy hol kell eltörni a sort (új sort kezdeni), akkor megkeresi, hogy melyik az a pont, ahol eltörve a sort a dobozt a legkisebb mértékben kell nyújtani. Probléma abból adódhat, hogy a sort nem lehet bárhol eltörni (hiszen a szöveget nem lehet bárhol megszakítani), de a sort tartalmazó doboz sem nyújtható bármilyen mértékben. Természetesen megtehetné a LaTeX, hogy a vízszintes dobozt extrém mértékben megnyújtja a szavak közt található szóközök megnövelésével, de ez csúnya nyomtatási képet eredményezne.

Ha a LaTeX *overfull hbox* (túlcsordult vízszintes doboz) hibaüzenettel jelzi, ha valamelyik vízszintes doboz „kilóg”, *underfull hbox* (alulcsordult vízszintes doboz) üzenettel pedig, ha „túl rövidre sikerül”. *overfull hbox* *underfull hbox*

A lapok tördelésénél hasonló problémák adódhatnak. Egy táblázat közepén például nem lehet új lapot kezdeni, ezért lehetséges, hogy a lapot tartalmazó függőleges doboz alulcsordul, ami azt fogja eredményezni, hogy túl kevés anyag kerül az oldalra.

A `\sloppy` parancs hatására a LaTeX szükség esetén kissé nagyobb szóközöket enged meg az alapértéknél. Ez több lehetőséget ad számára a sorok kiegyenlítésénél – így a túllógó soroktól megszabadulhatunk –, de az eredmény nem olyan szép. A `\fussy` parancs a szóközök méretének szigorúbb betartására utasítja a programot, amely szebb írásképet eredményez, de a túllógó sorokat sokszor kézzel kell tördelnünk. A két parancs egymás ellentéte, közülük a `\fussy` az alapbeállítás. `\sloppy` `\fussy`

A `sloppypar` környezet a `\sloppy` paranccsal megegyező hatású a közrezárt szövegre. Ha lehet inkább ezt a lehetőséget használjuk (nehogy „úgy felejtsük” a tördelési módot) és csak ott, ahol okvetlenül szükséges. `sloppypar`

A következő bekezdés parancsokat tartalmaz, amelyeket nem lehet elválasztani. Mivel a bekezdés szélessége erősen korlátozott, láthatjuk, hogy túlcsordulás történt:

A \begin{sloppypar} és a \end{sloppypar} parancsok közt ugyanolyan a tördelés, mint a \sloppy és a \fussy parancsok közt.

Ha a bekezdést sloppypar környezetben szedjük, a szavak közt megnövekednek a távolságok, de a túlcsordulás nem jelentkezik.

A \begin{sloppypar} és a \end{sloppypar} parancsok közt ugyanolyan a tördelés, mint a \sloppy és a \fussy parancsok közt.

\raggedbottom
\flushbottom

A lapok kialakítását a \raggedbottom és a \flushbottom parancsok nagymértékben befolyásolják. Ha a dokumentumra a \flushbottom parancs érvényes, akkor a szedéstükör mérete az összes lapon azonos lesz. Ez akkor különösen hasznos, ha a dokumentumot kétoldalas nyomtatással készítjük el, mivel így a jobb és bal oldalon az alsó sorok egyforma magasságban lesznek. A flushbottom hatására azonban – ha az oldal függőlegesen alulcsordul – a lapon túlságosan nagy függőleges utcák alakulnak ki. Ha a dokumentumra a \raggedbottom hatására a LaTeX a függőleges üres részeket nem nyújtja, így a laptükör alul feljebb végződik. A \flushbottom és a \raggedbottom parancsokat a preambulumban adhatjuk ki, és az egész dokumentumra érvényesek lesznek.

\clearpage

A \clearpage parancs segítségével laptörést hajthatunk végre a dokumentum bármely pontján. A \clearpage parancs hatására a LaTeX az adott ponton mindenképpen új oldalt kezd.

\cleardoublepage

A \cleardoublepage parancs hatására a program szintén új oldalt kezd, az új oldal azonban mindenképpen jobb oldalon elhelyezkedő, páratlan oldalszámmal jelzett oldal lesz. Ha szükséges a program egy üres oldalt szúr be annak érdekében, hogy az új oldal páratlan oldalszámú legyen.

\newpage

A \newpage hatására a LaTeX új oldalt kezd, előtte azonban az előzőt „nem fejezi be”. Ez azt eredményezi, hogy az előző oldal „szét lesz húzva” a szedéstükör tetejétől az aljáig és ez meglehetősen csúnya lehet. Ha a clearpage parancs hatásához hasonló formát akarunk elérni – vagyis az előző fejezetet a szedéstükör felső széléhez zárni –, akkor a következő formát használhatjuk:

```
\vfill
\newpage
```

A \pagebreak és \nopagebreak parancsokkal javaslatot tehetünk a LaTeX számára, hogy az adott ponton legyen vagy ne legyen laphatár. A \pagebreak parancs javaslatot tesz laptörés elhelyezésére, a \nopagebreak pedig megpróbálja „lebeszélni” a LaTeX-et arról, hogy az adott ponton laptörést helyezzen el. Mindkét parancs egy paramétert fogad, amely $1, \ldots, 4$ közt lévő szám. Minél kisebb a szám a javaslatunk annál kevésbé fogja befolyásolni a programot a munkájában, a paraméter értéke tehát azt jelzi, hogy mennyire ragaszkodunk a tördelési elképzelésünkhöz.

\pagebreak
\nopagebreak

Az \enlargethispage parancs hatására a program az adott lapot kissé megnöveli, hogy több szöveg férjen rá. A parancs egyetlen paramétere a szedéstükör megnövelésének mérete. A következő parancs például egy sornyi plussz szöveg elhelyezését teszi lehetővé az oldalon:

\enlargethispage

```
\enlargethispage{\baselineskip}
```

A példában szereplő \baselineskip parancs két egymás utáni sor alapvonalának távolságát határozza meg.

\baselineskip

6. fejezet

Matematikai formulák

6.1. A matematikai mód

A LaTeX a TeX-hez hasonlóan kétféle matematikai módot különböztet meg. A *sorközi matematikai mód*ban készített formulák a szövegben jelennek meg, ahogyan ezt a $\sum_{i=1}^{k} i$ példa mutatja. A *kiemelt matematikai mód*ban készített formulák általában középen, a sorok közt kihagyott üres helyen, önállóan jelennek meg, ahogyan ezt a

sorközi matematikai mód
kiemelt matematikai mód

$$\sum_{i=1}^{k} i$$

példa is mutatja. Érdemes felfigyelnünk arra, hogy a formulák nem teljesen azonos formában jelennek meg sorközi és kiemelt módban. Ennek a helytakarékosság az oka, amelyre azért van szükség, hogy a matematikai formulák miatt ne távolódjanak el a sorok egymástól túlságosan.

A TeX sorközi matematikai módba szedi a $ jelekkel kezdődő és $ jelekkel végződő szövegeket (pl. `$\sum_{i=1}^ki$`), kiemelt módban pedig a $$ és $$ jelekkel határolt részeket (pl. `$$\sum_{i=1}^ki$$`). Ezek az eszközök LaTeX-et használva is a rendelkezésünkre álnak.

`$`
`$$`

A LaTeX saját eszközei a sorközi matematikai mód elejének és végének jelzésére a `\( \)`, a kiemelt matematikai mód jelzésére pedig a `\[ \]` parancsok.

\qquad	a b
\quad	a b
\␣	a b
\;	a b
\>	a b
\,	a b
\!	a b

6.1. táblázat. Szóközök matematikai módban

6.1.1. Matematikai szimbólumok

A LaTeX képes szinte bármilyen matematikai formula szedésére. Ehelyütt csak a legelterjedtebbeb használt szimbólumokat mutatjuk be (elsősorban helyhiány miatt), de tudnunk kell, hogy a LaTeX gyakorlatilag *bármit*, bármilyen jelrendszert képes kezelni, hiszen számára a nyomtatott szöveg csak dobozok és ragasztók sokasága.

A matematikában nem járatos olvasó számára ezek az oldalak elhanyagolható jelentőségűek – bár kétségkívül esztétikai értékkel bírnak –, számukra csak annyi a mondanivalójuk, hogy a metamatikában minden apróságra ügyelni kell. Furcsa lehet, hogy életbevágóan fontos a $\geq$ és a $\succeq$ közti különbség, de azonnal erthetove valik, ha raebreduenk, hogy a reszletek masutt ugyancsak meglehetosen fontosak.

A következő oldalak a matematikai módban használható parancsok egy gyűjteményét mutatják be DAVID CARLISLE munkája nyomán.

(	(	)	)	↑	\uparrow
⇑	\Uparrow	[	[	]	]
↓	\downarrow	⇓	\Downarrow	{	\{
}	\}	↕	\updownarrow	⇕	\Updownarrow
⌊	\lfloor	⌋	\rfloor	⌈	\lceil
⌉	\rceil	⟨	\langle	⟩	\rangle
/	/	\	\backslash	|	|
‖	\\|				

6.2. táblázat. Határolójelek

$\pm$	\pm	$\cap$	\cap	$\diamond$	\diamond
$\oplus$	\oplus	$\mp$	\mp	$\cup$	\cup
$\bigtriangleup$	\bigtriangleup	$\ominus$	\ominus	$\times$	\times
$\uplus$	\uplus	$\bigtriangledown$	\bigtriangledown	$\otimes$	\otimes
$\div$	\div	$\triangleleft$	\triangleleft	$\sqcap$	\sqcap
$\oslash$	\oslash	$\ast$	\ast	$\sqcup$	\sqcup
$\triangleright$	\triangleright	$\odot$	\odot	$\star$	\star
$\vee$	\vee	$\lhd^a$	\lhd[a]	$\bigcirc$	\bigcirc
$\circ$	\circ	$\wedge$	\wedge	$\rhd^a$	\rhd[a]
$\dagger$	\dagger	$\bullet$	\bullet	$\setminus$	\setminus
$\unlhd^a$	\unlhd[a]	$\ddagger$	\ddagger	$\cdot$	\cdot
$\wr$	\wr	$\unrhd^a$	\unrhd[a]	$\amalg$	\amalg
$+$	+	$-$	-		

a) Nem tartalmazza a LaTeX, a `latexsym`, `amsfonts` vagy `amssymb` csomagot kell használnunk.

6.3. táblázat. Bináris operátorok

α	\alpha	θ	\theta	o	o
τ	\tau	β	\beta	ϑ	\vartheta
π	\pi	υ	\upsilon	γ	\gamma
ι	\iota	ϖ	\varpi	ϕ	\phi
δ	\delta	κ	\kappa	ρ	\rho
φ	\varphi	ϵ	\epsilon	λ	\lambda
ϱ	\varrho	χ	\chi	ε	\varepsilon
μ	\mu	σ	\sigma	ψ	\psi
ζ	\zeta	ν	\nu	ς	\varsigma
ω	\omega	η	\eta	ξ	\xi
Γ	\Gamma	Λ	\Lambda	Σ	\Sigma
Ψ	\Psi	Δ	\Delta	Ξ	\Xi
Υ	\Upsilon	Ω	\Omega	Θ	\Theta
Π	\Pi	Φ	\Phi		

6.4. táblázat. Görög betűk

,	,	;	;	:	\colon
.	\ldotp	·	\cdotp		

6.7. táblázat. Pontok, vesszők matematikai módban

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

a) Nem tartalmazza a LaTeX. A `latexsym`, `amsfonts` vagy az `amssymb` csomagot kell használnunk.

6.5. táblázat. Reláció jelek

$\widetilde{abc}$	\widetilde{abc}	$\widehat{abc}$	\widehat{abc}
$\overleftarrow{abc}$	\overleftarrow{abc}	$\overrightarrow{abc}$	\overrightarrow{abc}
$\overline{abc}$	\overline{abc}	$\underline{abc}$	\underline{abc}
$\overbrace{abc}$	\overbrace{abc}	$\underbrace{abc}$	\underbrace{abc}
$\sqrt{abc}$	\sqrt{abc}	$\sqrt[n]{abc}$	\sqrt[n]{abc}
f'	f'	$\frac{abc}{xyz}$	\frac{abc}{xyz}

6.6. táblázat. Csoportékezetek matematikai módba

$\leftarrow$	\leftarrow	$\longleftarrow$	\longleftarrow
$\uparrow$	\uparrow	$\Leftarrow$	\Leftarrow
$\Longleftarrow$	\Longleftarrow	$\Uparrow$	\Uparrow
$\rightarrow$	\rightarrow	$\longrightarrow$	\longrightarrow
$\downarrow$	\downarrow	$\Rightarrow$	\Rightarrow
$\Longrightarrow$	\Longrightarrow	$\Downarrow$	\Downarrow
$\leftrightarrow$	\leftrightarrow	$\longleftrightarrow$	\longleftrightarrow
$\updownarrow$	\updownarrow	$\Leftrightarrow$	\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	$\leadsto$[a]	\leadsto

a) A LaTeX nem tartalmazza, a `latexsym`, `amsfonts` vagy a `amssymb` csomagot kell használnunk.

6.8. táblázat. Nyilak

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

6.9. táblázat. Méretezhető operátorok

$\rmoustache$	\rmoustache	$\lmoustache$	\lmoustache	$\rgroup$	\rgroup
$\lgroup$	\lgroup	$\arrowvert$	\arrowvert	$\Arrowvert$	\Arrowvert
$\bracevert$	\bracevert				

6.10. táblázat. Nagyméretű határolójelek

…	\ldots	⋯	\cdots	⋮	\vdots
⋱	\ddots	ℵ	\aleph	′	\prime
∀	\forall	∞	\infty	ℏ	\hbar
∅	\emptyset	∃	\exists	□	\Box[a]
ı	\imath	∇	\nabla	¬	\neg
◇	\Diamond[a]	ȷ	\jmath	√	\surd
♭	\flat	△	\triangle	ℓ	\ell
⊤	\top	♮	\natural	♣	\clubsuit
℘	\wp	⊥	\bot	♯	\sharp
♢	\diamondsuit	ℜ	\Re	∥	\\|
\	\backslash	♡	\heartsuit	ℑ	\Im
∠	\angle	∂	\partial	♠	\spadesuit
℧	\mho[a]	.	.	|	|

a) A LaTeX nem tartalmazza. A latexsym, amsfonts vagy amssymb csomag használata szükséges.

6.11. táblázat. Matematikai szimbólumok

$\dot{a}$	\dot{a}	$\breve{a}$	\breve{a}	$\check{a}$	\check{a}
$\grave{a}$	\grave{a}	$\vec{a}$	\vec{a}	$\ddot{a}$	\ddot{a}
$\tilde{a}$	\tilde{a}				

6.12. táblázat. Matematikai ékezetek

⌜	\ulcorner	⌝	\urcorner	⌞	\llcorner	⌟	\lrcorner

6.13. táblázat. Az amsmath csomag határolójelei

F	\digamma	$\varkappa$	\varkappa

6.14. táblázat. Az amsmath csomag görög betűi

ℶ	\beth	ℸ	\daleth	ℷ	\gimel

6.15. táblázat. Az amsmath csomag héber betűi

$\dashrightarrow$	\dashrightarrow	$\dashleftarrow$	\dashleftarrow
$\leftleftarrows$	\leftleftarrows	$\leftrightarrows$	\leftrightarrows
$\Lleftarrow$	\Lleftarrow	$\twoheadleftarrow$	\twoheadleftarrow
$\leftarrowtail$	\leftarrowtail	$\looparrowleft$	\looparrowleft
$\leftrightharpoons$	\leftrightharpoons	$\curvearrowleft$	\curvearrowleft
$\circlearrowleft$	\circlearrowleft	$\Lsh$	\Lsh
$\upuparrows$	\upuparrows	$\upharpoonleft$	\upharpoonleft
$\downharpoonleft$	\downharpoonleft	$\multimap$	\multimap
$\leftrightsquigarrow$	\leftrightsquigarrow	$\rightrightarrows$	\rightrightarrows
$\rightleftarrows$	\rightleftarrows	$\rightrightarrows$	\rightrightarrows
$\rightleftarrows$	\rightleftarrows	$\twoheadrightarrow$	\twoheadrightarrow
$\rightarrowtail$	\rightarrowtail	$\looparrowright$	\looparrowright
$\rightleftharpoons$	\rightleftharpoons	$\curvearrowright$	\curvearrowright
$\circlearrowright$	\circlearrowright	$\Rsh$	\Rsh
$\downdownarrows$	\downdownarrows	$\upharpoonright$	\upharpoonright
$\downharpoonright$	\downharpoonright	$\rightsquigarrow$	\rightsquigarrow

6.16. táblázat. Az amsmath csomag nyilai

$\nleftarrow$	\nleftarrow	$\nrightarrow$	\nrightarrow
$\nLeftarrow$	\nLeftarrow	$\nRightarrow$	\nRightarrow
$\nleftrightarrow$	\nleftrightarrow	$\nLeftrightarrow$	\nLeftrightarrow

6.17. táblázat. Az amsmath csomag áthúzott nyilai

△	\vartriangle	▽	\triangledown
□	\square	◊	\lozenge
Ⓢ	\circledS	∠	\angle
∡	\measuredangle	∄	\nexists
℧	\mho	ⅎ	\Finv
⅁	\Game	𝕜	\Bbbk
‵	\backprime	∅	\varnothing
▲	\blacktriangle	▼	\blacktriangledown
■	\blacksquare	⧫	\blacklozenge
★	\bigstar	◁	\sphericalangle
∁	\complement	´	\eth
⁄	\diagup	╲	\diagdown

6.18. táblázat. Egyéb jelek az amsmath csomagban

∔	\dotplus	∖	\smallsetminus
⋒	\Cap	⋓	\Cup
⊼	\barwedge	⩒	\veebar
⩞	\doublebarwedge	⊟	\boxminus
⊠	\boxtimes	⊡	\boxdot
⊞	\boxplus	⋇	\divideontimes
⋉	\ltimes	⋊	\rtimes
⋋	\leftthreetimes	⋌	\rightthreetimes
⋏	\curlywedge	⋎	\curlyvee
⊝	\circleddash	⊛	\circledast
⊚	\circledcirc	⋅	\centerdot
⊺	\intercal		

6.19. táblázat. Az amsmath csomag bináris operátorai

$\leqq$	\leqq	$\leqslant$	\leqslant
$\eqslantless$	\eqslantless	$\lesssim$	\lesssim
$\lessapprox$	\lessapprox	$\approxeq$	\approxeq
$\lessdot$	\lessdot	$\lll$	\lll
$\lessgtr$	\lessgtr	$\lesseqgtr$	\lesseqgtr
$\lesseqqgtr$	\lesseqqgtr	$\doteqdot$	\doteqdot
$\risingdotseq$	\risingdotseq	$\fallingdotseq$	\fallingdotseq
$\backsim$	\backsim	$\backsimeq$	\backsimeq
$\subseteqq$	\subseteqq	$\Subset$	\Subset
$\sqsubset$	\sqsubset	$\preccurlyeq$	\preccurlyeq
$\curlyeqprec$	\curlyeqprec	$\precsim$	\precsim
$\precapprox$	\precapprox	$\vartriangleleft$	\vartriangleleft
$\trianglelefteq$	\trianglelefteq	$\vDash$	\vDash
$\Vvdash$	\Vvdash	$\smallsmile$	\smallsmile
$\smallfrown$	\smallfrown	$\bumpeq$	\bumpeq
$\Bumpeq$	\Bumpeq	$\geqq$	\geqq
$\geqslant$	\geqslant	$\eqslantgtr$	\eqslantgtr
$\gtrsim$	\gtrsim	$\gtrapprox$	\gtrapprox
$\gtrdot$	\gtrdot	$\ggg$	\ggg
$\gtrless$	\gtrless	$\gtreqless$	\gtreqless
$\gtreqqless$	\gtreqqless	$\eqcirc$	\eqcirc
$\circeq$	\circeq	$\triangleq$	\triangleq
$\thicksim$	\thicksim	$\thickapprox$	\thickapprox
$\supseteqq$	\supseteqq	$\Supset$	\Supset
$\sqsupset$	\sqsupset	$\succcurlyeq$	\succcurlyeq
$\curlyeqsucc$	\curlyeqsucc	$\succsim$	\succsim
$\succapprox$	\succapprox	$\vartriangleright$	\vartriangleright
$\trianglerighteq$	\trianglerighteq	$\Vdash$	\Vdash
$\shortmid$	\shortmid	$\shortparallel$	\shortparallel
$\between$	\between	$\pitchfork$	\pitchfork
$\varpropto$	\varpropto	$\blacktriangleleft$	\blacktriangleleft
$\therefore$	\therefore	$\backepsilon$	\backepsilon
$\blacktriangleright$	\blacktriangleright	$\because$	\because

6.20. táblázat. Az amsmath csomag relációjelei

$\nless$	\nless	$\nleq$	\nleq
$\nleqslant$	\nleqslant	$\nleqq$	\nleqq
$\lneq$	\lneq	$\lneqq$	\lneqq
$\lvertneqq$	\lvertneqq	$\lnsim$	\lnsim
$\lnapprox$	\lnapprox	$\nprec$	\nprec
$\npreceq$	\npreceq	$\precnsim$	\precnsim
$\precnapprox$	\precnapprox	$\nsim$	\nsim
$\nshortmid$	\nshortmid	$\nmid$	\nmid
$\nvdash$	\nvdash	$\nvDash$	\nvDash
$\ntriangleleft$	\ntriangleleft	$\ntrianglelefteq$	\ntrianglelefteq
$\nsubseteq$	\nsubseteq	$\subsetneq$	\subsetneq
$\varsubsetneq$	\varsubsetneq	$\subsetneqq$	\subsetneqq
$\varsubsetneqq$	\varsubsetneqq	$\ngtr$	\ngtr
$\ngeq$	\ngeq	$\ngeqslant$	\ngeqslant
$\ngeqq$	\ngeqq	$\gneq$	\gneq
$\gneqq$	\gneqq	$\gvertneqq$	\gvertneqq
$\gnsim$	\gnsim	$\gnapprox$	\gnapprox
$\nsucc$	\nsucc	$\nsucceq$	\nsucceq
$\nsucceq$	\nsucceq	$\succnsim$	\succnsim
$\succnapprox$	\succnapprox	$\ncong$	\ncong
$\nshortparallel$	\nshortparallel	$\nparallel$	\nparallel
$\nvDash$	\nvDash	$\nVDash$	\nVDash
$\ntriangleright$	\ntriangleright	$\ntrianglerighteq$	\ntrianglerighteq
$\nsupseteq$	\nsupseteq	$\nsupseteqq$	\nsupseteqq
$\supsetneq$	\supsetneq	$\varsupsetneq$	\varsupsetneq
$\supsetneqq$	\supsetneqq	$\varsupsetneqq$	\varsupsetneqq

6.21. táblázat. Az amsmath csomag tagadó relációjelei

$\Lbag$	\Lbag	$\Rbag$	\Rbag	$\lbag$	\lbag	$\rbag$	\rbag
$\llceil$	\llceil	$\rrceil$	\rrceil	$\llfloor$	\llfloor	$\rrfloor$	\rrfloor
$\llbracket$	\llbracket	$\rrbracket$	\rrbracket				

6.22. táblázat. Az stmaryrd csomag határolójelei

⟽	\Longmapsfrom	⟾	\Longmapsto
⇤	\Mapsfrom	⇥	\Mapsto
↗	\nnearrow	↖	\nnwarrow
↘	\ssearrow	↙	\sswarrow
↓	\shortdownarrow	↑	\shortuparrow
←	\shortleftarrow	→	\shortrightarrow
⟻	\longmapsfrom	↤	\mapsfrom
←	\leftarrowtriangle	→	\rightarrowtriangle
`	\lightning	)	\rrparenthesis
⇿	\leftrightarroweq	↭	\leftrightarrowtriangle

6.23. táblázat. Az stmaryrd csomag nyilai

/	\Arrownot	∣	\Mapsfromchar	∣	\Mapstochar
/	\arrownot	∣	\mapsfromchar		

6.24. táblázat. Az stmaryrd csomag egyéb jelei

⋵	\inplus	⋹	\niplus
⋐	\subsetplus	⋒	\subsetpluseq
⋑	\supsetplus	⋓	\supsetpluseq
⋪	\trianglelefteqslant	⋫	\trianglerighteqslant

6.25. táblázat. Az stmaryrd csomag relációjelei

⨅	\bigbox	⋎	\bigcurlyvee
⋏	\bigcurlywedge	⫼	\biginterleave
⨁	\bignplus	∥	\bigparallel
⨅	\bigsqcap	▽	\bigtriangledown
△	\bigtriangleup		

6.26. táblázat. Az stmaryrd csomag nagyméretű relációjelei

⋬	\ntrianglelefteqslant	⋭	\ntrianglerighteqslant

6.27. táblázat. Az stmaryrd csomag tagadó relációjelei

⋎	\Ydown	⋏	\Yleft
⋏	\Yright	⋎	\Yup
⧼	\baro	⧹	\bbslash
⅋	\binampersand	⧓	\bindnasrepma
⧆	\boxast	◫	\boxbar
⧈	\boxbox	⧅	\boxbslash
⧇	\boxcircle	⊡	\boxdot
□	\boxempty	⧄	\boxslash
⋎	\curlyveedownarrow	⋎	\curlyveeuparrow
⋏	\curlywedgedownarrow	⋏	\curlywedgeuparrow
⧹	\fatbslash	⨟	\fatsemi
⧸	\fatslash	⫴	\interleave
◁	\leftslice	⩛	\merge
⨵	\minuso	⨲	\moo
⨣	\nplus	⦶	\obar
□	\oblong	⦸	\obslash
⧁	\ogreaterthan	⧀	\olessthan
⦶	\ovee	⦵	\owedge
▷	\rightslice	⫽	\sslash
▯	\talloblong	◯	\varbigcirc
⋎	\varcurlyvee	⋏	\varcurlywedge
⊛	\varoast	⦶	\varobar
⦸	\varobslash	⊚	\varocircle
⊙	\varodot	⧁	\varogreaterthan
⧀	\varolessthan	⊖	\varominus
⊕	\varoplus	⊘	\varoslash
⊗	\varotimes	⦶	\varovee
⦵	\varowedge	⨯	\vartimes

6.28. táblázat. Az stmaryrd csomag operátorai

6.2. Alapvető matematikai szerkezetek

6.2.1. Alsó és felső index

felső index

Matematikai üzemmódban *felső index*et a ^ karakter segítségével hozhatunk létre. A a^2+b^2=c^2 szöveg matematikai üzemmódban a következőképpen jelenik meg:

$$a^2 + b^2 = c^2$$

Az *alsó index* létrehozására hasonlóképpen kell használnunk a _ karaktert. A `x_{1,2}=x_i+y_i` képlet matematikai módban a következő formát adja: *alsó index*

$$x_{1,2} = x_i + y_i$$

Az indexek írásánál figyelembe kell vennünk, hogy a ^ és _ jelek csak matematikai üzemmódban használhatóak, így mindent amit indexbe akarunk írni matematikai módban kell szednünk. Fontos tudnunk azt, hogy a ^ és _ karakterek egy jelre, parancsra vonatkoznak. Ha egynél több jelet akarunk indexbe írni, akkor a teljes indexet kapcsos zárójelek közt kell írnunk.

6.2.2. Osztás

A `\frac` parancs segítségével tudunk osztást készíteni matematikai módban. A parancs két paramétere a számláló és nevező, amelyeket akkor kell kapcsos zárójelek közt írnunk, ha több tagból állnak. `\frac`

A `y_1=\frac{x+1}{x-1}` kifejezés matematikai módban a következőképpen jelenik meg:

$$y_1 = \frac{x+1}{x-1}$$

A `a=\frac{1}{x+\frac{3}{y}}` formula egymásba ágyazott osztásokat ábrázol, amelyek a következőképp jelennek meg:

$$a = \frac{1}{x + \frac{3}{y}}$$

6.2.3. Gyökvonás

A négyzetgyök jelét a `\sqrt` parancs segítségével szedhetjük matematikai módban. A `\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}` képlet pl. kiemelt matematikai üzemmódban a következőképpen jelenik meg: `\sqrt`

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

A `\root` és `\of` parancsok segítségével készíthetünk tetszőleges kite- `\root` `\of`

vőjű gyököt. A `\root 3 \of {x^2 + y^3}` formula kiemelt matematikai módban a következő formát adja:

$$\sqrt[3]{x^2 + y^3}$$

6.2.4. Integrálás

`\int` `\iint` `\iiint`

Az `\int` parancs segítségével hozhatunk létre integráljelet. A parancsnak léteznek többszörös változatai (`\iint` , `\iiint`), amelyek többszörös integrálás szedésére alkalmasak. A `\iint_a^b \phi(u, v)~du~dv` formula szedésben:

$$\iint_a^b \phi(u, v)\ du\ dv$$

A `\iiiint \phi(x, y, u, v)` formula négyszeres integráljelet ad:

$$\iiiint \phi(x, y, u, v)$$

6.2.5. Zárójelek

Alapesetben a LaTeX a kerek és a szögletes zárójeleket értelmezi és a képletbe írja. Ezeket tehát egyszerűen szedhetjük. A `2(1+x)` formula pl. nyomtatásban $2(1 + x)$ alakban jelenik meg. A kapcsos zárójeleket `\{` és `\}` alakban tudjuk beírni. Az `A=\{1,2,\dots\}` szöveg pl. $A = \{1, 2, \dots\}$ alakú lesz matematikai módban.

Fontos lehet tudnunk, hogy ezeknek a zárójeleknek a párosítását a LaTeX nem ellenőrzi, a formulákat akkor is képes megjeleníteni, ha ezek a zárójelek nincsenek párban. Lehetséges, hogy az adott tudományterületen a `x=5[3c)` hatására megjelenő $x = 5[3c)$ képlet hibás, a LaTeX számára azonban csak egy megjeleníthető kifejezés.

`\left` `\right`

Ha nagyméretű zárójeleket szeretnénk létrehozni, akkor a `\left` és - `\right` szavakat kell a zárójelek elé írnunk, így azok méretét a LaTeX a szükséges mértékben megnöveli.

A `\left(\frac{1+x}{1-x}\right)+5` hatására a két zárójel olyan méretben jelenik meg, amelyet a közrezárt tört szükségessé tesz:

$$\left(\frac{1+x}{1-x}\right)+5$$

Fontos tudnunk, hogy a `\left` és `\right` parancsoknak párban kell állniuk, mert ezeket a LaTeX a feldolgozás során párba állítja. Ha csak egy nagyméretű zárójelet akarunk létrehozni, akkor a másik oldalon a `\right.` formában kell lezárnunk. Ez egy olyan nagyméretű zárójelet eredményez, amely nyomtatásban nem jelenik meg.

Ennek megfelelően a `x=\left\{\frac{1}{x}\right.` matematikai módban a következő szedésképet adja:

$$x = \left\{ \frac{1}{x} \right.$$

Kezdő felhasználók gyakorta követik el azt a hibát, hogy a megnövelt zárójeleket és a `{}` jeleket átlapolva írják. A `{ \left( } \right)` alakú formulák szedését a LaTeX megtagadja.

6.3. Különlegesen kezelt formulák

6.3.1. Tömbök, mátrixok

A szöveges üzemmódban táblázatok kezelésére használt `tabular` környezet matematikai módban nem használható. Helyette szinte változatlan formában az `array` környezet áll a rendelkezésünkre.

Az `array` környezetet sorokba és oszlopokba rendezett formulák szedésére használjuk, amilyenek pl. a mátrixok, tömbök. A következő példa egy mátrixot mutat be 2×2 méretben: `array`

$$X = \left[\begin{array}{rr} \pi^2 & -\pi \\ -\pi^2 & \pi \end{array} \right]$$

A mátrix matematikai módban, az `array` környezet felhasználásával készült:

```
X=\left[
     \begin{array}{rr}
       \pi^2  & -\pi \\
       -\pi^2 & \pi  \\
     \end{array}
  \right]
```

6.3.2. Formulák tördelése, igazítása

align

Egyenletrendszerek szedésére szolgál az `align` környezet. A környezeten belül a formulákat a `tabular` és `array` környezeteknél megszokott formában kell igazítanunk:

$$x = 5x + 12y \tag{6.1}$$

$$2y = 3x + 2y \tag{6.2}$$

Az egyenletrendszer a következőképp készült:

```
\begin{align}
 x  &= 5x + 12y \\
 2y &= 3x + 2y
\end{align}
```

6.3.3. Formulák számozása

equation
\label
\ref
\pageref

Az `equation` környezetbe írt formulákat a LaTeX matematikai módban szedi, azokat automatikusan számozással látja el. A `\label` paranccsal készíthetünk címkéket ezekhez, a `\ref` és `\pageref` paranccsal pedig hivatkozhatunk a formula számára és az oldalra, ahol elhelyezte a program.

A következő képlet az `equation` környezetben készült:

$$x^2 + 2x - 5y + 12 = 0 \tag{6.3}$$

A 6.3 képlet a következőképpen készült:

```
\begin{equation}
   \label{egyszeru}
   x^2+2x-5y+12=0
\end{equation}

A \ref{egyszeru} képlet a következőképpen készült:
```

equation*

Ha egy formulát nem akarunk számozni – mert pl. nem annyira fontos –, akkor az `equation*` környezetet kell használnunk. Ez mindenben megegyezik az `equation` környezettel, a különbség csak az, hogy tartalma mellett nem jelenik meg szám.

align

Ha az `align` környzetbe szedett többsoros egyenletrendszerek minden

sora önálló számot kap. A következő példa ezt mutatja be:

$$y^2 + 3y - 23 = 2 \tag{6.4}$$
$$x^2 + 2x - 5y + 12 = 0 \tag{6.5}$$

Ügyelnünk kell arra, hogy a környezeten belül a `\label` parancsokat a `\\` sorvégjelek elé írjuk. A 6.4-6.5 egyenletek a következőképpen készültek:

```
\begin{align}
   y^2+3y-23    &= 2  \label{elso}\\
   x^2+2x-5y+12 &= 0  \label{masodik}
\end{align}
```

Ha az egyenletrendszer valamelyik sora mellett nem akarunk számot megjeleníteni, akkor a `\nonumber` kulcsszót kell használnunk, hasonlóképpen az ismertetett `\label` használatához: `\nonumber`

$$y^2 + 3y - 23 = 2 \tag{6.6}$$
$$x^2 + 2x - 5y + 12 = 0$$

A példa a következőképpen készült:

```
\begin{align}
   y^2+3y-23    &= 2  \label{masik}\\
   x^2+2x-5y+12 &= 0  \nonumber
\end{align}
```

7. fejezet

Rajzok, ábrák

E fejezetben néhány egyszerű eszközt mutatunk be, amelyek rajzok készítését teszik lehetővé kizárólag LaTeX-et használva. Az itt leírtak a program grafikus képességeinek csak töredékét mutatják be.

7.1. Rajzok készítése

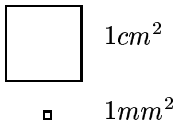
A `picture` környezetben LaTeX parancsok segítségével készíthetünk egyszerűbb ábrákat. A környezet létrehozásának formája a következő: `picture`

```
\begin{picture}(szélesség,magasság)(x kezdet,y kezdet)
  .
  .
  parancsok
  .
  .
\end{picture}
```

A környezet kezdetekor meg kell adnunk, hogy milyen széles és milyen magas képet akarunk készíteni, valamint azt, hogy hol kezdődjön a használni kívánt koordinátarendszer. Ugyanazok a parancsok eltérő képet adnak ha más méretet vagy kezdőpontot adunk meg a környezet elején. A kezdőpont megadása nem kötelező, ekkor értéke $0, 0$.

A megadott értékek alapértelmezés szerint pontban értendőek. A `\unitlength` segítségével meghatározhatjuk, hogy milyen léptéket kívánunk `\unitlength`

használni az elkészítendő képen, mi legyen a koordináták mértékegysége. A következő példa egy olyan kép készítését mutatja be, ahol milliméter beosztású koordinátarendszert használunk.

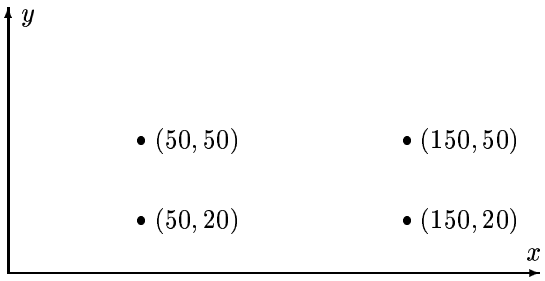



```
\setlength{\unitlength}{1mm}
\begin{center}
\begin{picture}(40,50)
 \put(5,5){\line(0,1){1}\line(1,0){1}}
 \put(6,6){\line(0,-1){1}\line(-1,0){1}}
 \put(13,5){$1mm^2$}
 \put(0,10){\line(0,1){10}\line(1,0){10}}
 \put(10,20){\line(0,-1){10}\line(-1,0){10}}
 \put(13,15){$1cm^2$}
\end{picture}
\end{center}
\setlength{\unitlength}{1pt}
```

Figyeljük meg, hogy a kép elkészülte után a koordinátarendszert visszaállítottuk az alapértelmezett 1 pontra. Ez mindenképpen hasznos, hacsak nem akarjuk a teljes dokumentumban az adott mértékrendszert használni.

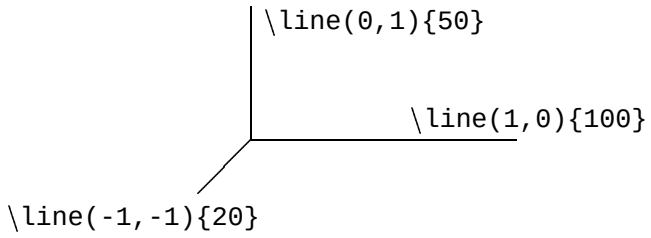
\put

A képre elemeket elhelyezni a `\put` parancs segítségével lehet. A parancs a `\put(`*x*`,`*y*`){`*elem*`}` formában használható, ahol x a vízszintes távolság a kép bal oldalától, y a függőleges távolság a kép aljától, az *elem* pedig a képre helyezendő elem:



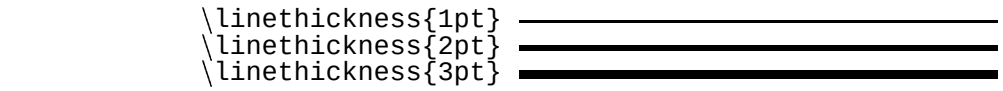
```
\begin{picture}(200,100)
  \put(0,0) {\vector(1,0){200}} \put(195, 5){$x$}
  \put(0,0) {\vector(0,1){100}} \put(5,95){$y$}
  \put(50,50){\circle*{3}}  \put(55,47){$(50,50)$}
  \put(150,50){\circle*{3}} \put(155,47){$(150,50)$}
  \put(50,20){\circle*{3}}  \put(55,17){$(50,20)$}
  \put(150,20){\circle*{3}} \put(155,17){$(150,20)$}
\end{picture}
```

A `\line` parancs segítségével egyenes vonalat rajzolhatunk. A parancs `\line(`*dx*`,`*dy*`){`*h*`}` formájú, ahol dx és dy az egyenes irányát határozzák meg, míg h a megrajzolt vonal hossza. `\line`



Az egyenes vonalakhoz hasonló formában nyilakat is rajzolhatunk a `\vector` paranccsal. A parancs formája megegyezik a `\line` parancs formájával. `\vector`

A képen található egyenes vonalak vastagságát a `\linethickness` paranccsal adhatjuk meg. A parancs egyetlen paramétere a vonalvastagság bármely ismert mértékegységben. A parancs kiadása után az összes egyenes szakasz az új vonalvastagsággal lesz rajzolva. `\linethickness`




```
\begin{picture}(300,50)
  \linethickness{1pt}
  \put(120,40) {\line(1,0){180}}
  \put(0,38){\texttt{\bs linethickness\{1pt\}}}
```

```
  \linethickness{2pt}
  \put(120,30) {\line(1,0){180}}
  \put(0,28){\texttt{\bs linethickness\{2pt\}}}

  \linethickness{3pt}
  \put(120,20) {\line(1,0){180}}
  \put(0,18){\texttt{\bs linethickness\{3pt\}}}
\end{picture}
```

\circle
\circle*

Kört rajzolni a `\circle` parancs segítségével lehet. A parancs egyetlen paramétere a kör átmérője. A `\circle*` parancs segítségével kitöltött kört rajzolhatunk.

●\circle*{5} ◦\circle{2}

```
\begin{center}
\begin{picture}(200,20)
  \put(0,10){\circle*{5}}
  \put(4,8) {\texttt{\bs circle*\{5\}}}
  \put(100,10){\circle{2}}
  \put(104,8) {\texttt{\bs circle\{2\}}}
\end{picture}
\end{center}
```

\dashbox

A `\dashbox` parancs segítségével szaggatott vonallal határolt téglalapot rajzolhatunk. A parancs formája `\dashbox{`*l*`}(`*dx*`,`*dy*`){...}`, ahol l a szaggatott vonalat alkotó szakaszok hossza, dx és dy a téglalap vízszintes és függőleges méretei, az utolsó paraméter pedig a rajzelem, amely a téglalapba kerül. A következő példa mutatja, hogy a szakaszok hosszát nem kötelező megadni:

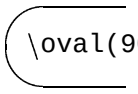
\dashbox{5}(180,20){...}

\dashbox(180,20){...}

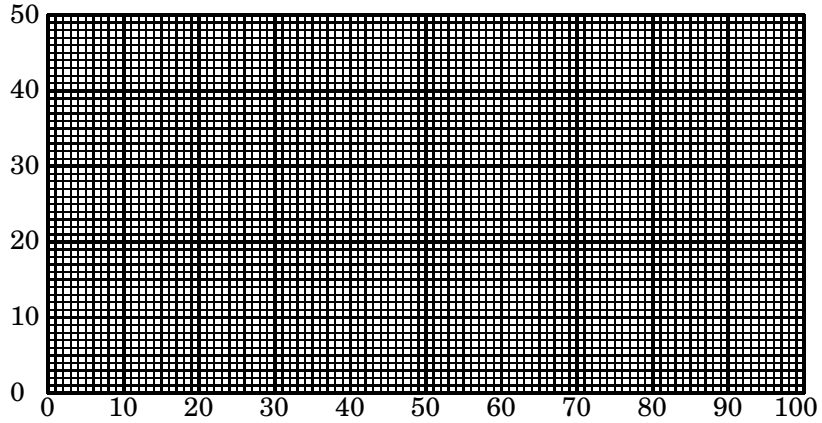
\oval

Az `\oval` parancs segítségével oválist rajzolhatunk. A parancs formá-

tuma \oval(*dx*,*dy*), ahol dx a szélesség, dy pedig a magasság. A parancsnak egy paramétert adva megadhatjuk, hogy a görbevonal melyik részét kívánjuk kirajzolni. A szakaszokat az `l` (bal), `r` (jobb), `t` (fent) és `b` (lent) betűkkel kérhetjük.



A \multiput parancs segítségével több rajzelemet helyezhetünk el egy lépésben. A parancs formája \multiput(*x,y*)(*dx,dy*){*n*}{elem}, ahol az x és y az első elem koordinátái, a dx és a dy az újabb elemek elhelyezésekor alkalmazott eltolás, míg az n az elhelyezendő elemek száma. Az *elem* helyére kerülnek az elhelyezendő elemek. A következő ábra egy milliméter beosztású négyzethálót ábrázol, amely ezzel a paranccsal készült. \multiput

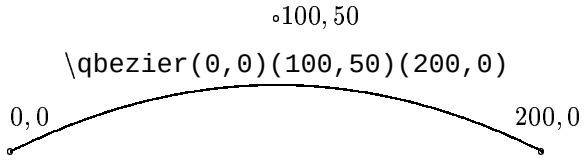



```
\setlength{\unitlength}{1mm}
\begin{picture}(100,50)
  \multiput(0,1)(0,1){50}{\line(1,0){100}}
  \multiput(1,0)(1,0){100}{\line(0,1){50}}
  \linethickness{1pt}
  \multiput(0,0)(0,10){6}{\line(1,0){100}}
  \multiput(0,0)(10,0){11}{\line(0,1){50}}
  \put(-1,-3){0}  \put(8, -3){10} \put(18,-3){20}
  \put(28,-3){30} \put(38,-3){40} \put(48,-3){50}
  \put(58,-3){60} \put(68,-3){70} \put(78,-3){80}
```

```
  \put(88,-3){90} \put(96,-3){100} \put(-5,-1){0}
  \put(-5,9){10}  \put(-5,19){20}  \put(-5,29){30}
  \put(-5,39){40} \put(-5,49){50}
\end{picture}
```

\qbezier

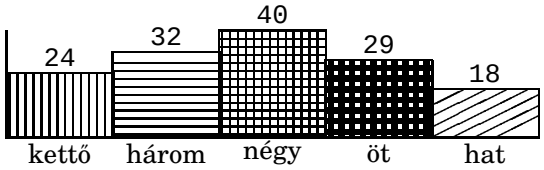
A \qbezier parancs segítségével görbét rajzolhatunk. A parancs paramétereként azt a három pontot kell megadnunk, amely meghatározza a görbevonal elhelyezkedését:



7.1.1. Oszlopdiagramok

oszlopdiagramok
bar
barenv

Az *oszlopdiagramok* készítésére a bar csomag betöltésével rendelkezésre álló barenv környezet is használható. A következő példa egy egyszerű oszlopdiagramot ábrázol.

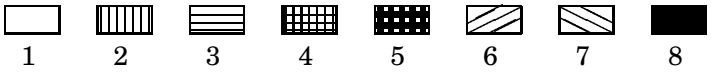



```
\begin{barenv}
  \bar{24}{2}[kettő]
  \bar{32}{3}[három]
  \bar{40}{4}[négy]
  \bar{29}{5}[öt]
  \bar{18}{6}[hat]
\end{barenv}
```

Amint láthatjuk, a diagram elkészítése igen egyszerű, csak az értékeket kell megadnunk, minden mást elvégez helyettünk a LaTeX.

\bar

A \bar parancs segítségével rajzolhatunk egy oszlopot a diagramra. A parancs formája \bar{*magasság*}{*minta*}[*szöveg*], ahol *magasság* az ábrázolt érték, a *minta* az oszlop kitöltési mintáját megadó szám (7.1 ábra), a *szöveg* pedig az oszlop által ábrázolt érték szöveges leírása, amely jelmagyarázatként az ábrára kerül.

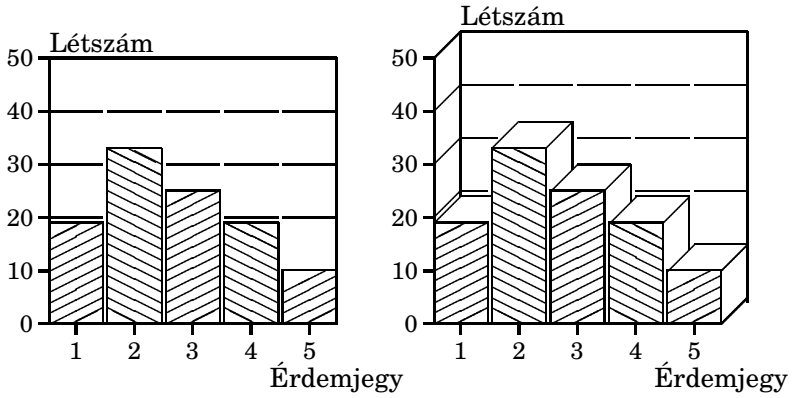


7.1. ábra. Oszlopdiagramok kitöltési minái

A `\hlineon` parancs segítségével bekapcsolhatjuk a diagramban az értékek leolvasását megkönnyítő vízszintes vonalak rajzolását. `\hlineon`

A `\legend` parancs segítségével a jelmagyarázatban felhasználható téglalapot hozhatunk létre. A parancs formája `\legend{`*minta*`}{`*szöveg*`}`, ahol a *minta* határozza meg, hogy a rajzolt téglalap milyen mintával legyen kitöltve (7.1 ábra), a *szöveg* pedig a jelmagyarázat szövege. `\legend`

A `\setdepth` parancs segítségével a három dimenziós oszlopdiagram mélységét adhatjuk meg. A parancs egyetlen paramétere a mélység, amely ≥ 10. A következő két példa közt csak az a különbség, hogy az egyik készítésekor a `\setdepth{10}` parancsot is használtuk. `\setdepth`




```
\begin{barenv}
  \setdepth{10}
  \setstretch{2}
  \setwidth{20}
  \sethspace{.1}
  \setstyle{\small\itshape}
  \setxname{Érdemjegy}
  \setyname{Létszám}
  \setstyle{\small}
```

```
   \setxaxis{1}{5}{1}
   \setyaxis{0}{50}{10}
   \setlinestyle{solid}
   \hlineon
   \setnumberpos{top}

   \bar{19}{6} \bar{33}{7} \bar{25}{6}
   \bar{19}{7} \bar{10}{6}
\end{barenv}
```

\sethspace A \sethspace parancs segítségével meghatározhatjuk, hogy mekkora üres hely maradjon az oszlopok közt. A parancs egyetlen paramétere az a távolságérték, amely meghatározza mekkora üres hely maradjon az oszlopok közt.

\setlinestyle A vízszintes vonalak típusát a \setlinestyle parancs segítéségével állíthatjuk be. A parancs paramétere lehet solid (folytonos vonal) vagy dotted (pontokból összeállított vonal).

solid

dotted

\setnumberpos A \setnumberpos parancs segítségével meghatározhatjuk, hogy az oszlopokhoz tartozó értékeket a LaTeX hogyan jelenítse meg. A parancs egyetlen paramétere a következő értékeket veheti fel:

empty	Az oszlopokon nem jelennek meg az értékek.
axis	Az érték a vízszintes tengely alatt jelenik meg.
down	Az érték az oszlopok alatt jelenik meg.
inside	Az érték az oszlopok belsejében jelenik meg.
outside	Az érték az oszlopon kívül jelenik meg.
up	Az érték az oszlopok felett jelenik meg.

\setprecision A \setprecision parancs segítségével beállíthatjuk, hogy a diagramban lévő értékek milyen pontossággal jelenjenek meg. A parancs egyetlen paramétere egy szám, amely megadja, hogy a tizedes jel után hány számjegy jelenjen meg.

\setstretch A \setstretch parancs segítségével a diagramot függőleges irányba nyújthatjuk ill. összenyomhatjuk. A parancs paramétere egy szám, amely megadja, hogy a diagramot hányszorosára nyújtjuk függőleges irányban.

A \setstyle parancs segítségével a diagramra kerülő feliratok stílusa adható meg. A parancs paramétere LaTeX parancsokat tartalmazhat, amelyek betűformázásra vonatkoznak. A munka során először beállítjuk a betűk formáját ezzel a paranccsal, majd megadjuk az egyes szövegelemeket más parancsok segítségével: \setstyle

\setstyle{*formázás*}
\setxname{*szöveg*}

A \setwidth parancs segítségével meghatározhatjuk a diagram szélességét. A parancs egyetlen paramétere egy szám, amely magadja, hogy a diagramban található oszlopok egyenként milyen szélesek. \setwidth

A \setxaxis parancs segítségével könnyedén helyezhetünk el számokat a diagram vízszintes tengelyén, \setxaxis{*kezdet*}{*vég*}{*lépés*} formában. Itt a *kezdet* határozza meg a vízszintes tengelyen elhelyezett első számot, a *vég* az utolsót, a *lépés* pedig azt, hogy mekkora lépésközzel növekedjen a számozás. \setxaxis

A \setxname parancs segítségével szöveget írhatunk az x tengelyre, amely megmutatja, hogy milyen adatokat tartalmaz a diagram. A parancs egyetlen paramétere a tengelyre írandó szöveg. \setxname

A \setyaxis parancs nagyon hasonló a \setxaxis parancshoz, különbség csak annyi, hogy egy elhagyható paraméterrel, az eltolással növelhetjük a kezdő és végértéket, anélkül, hogy a skála méretein változtatnánk. A parancs formája: \setyaxis[*eltolás*]{*kezdet*}{*vég*}{*lépés*}. \setyaxis

A \setylabel parancs segítségével a függőleges tengelyre írhatjuk, hogy mit ábrázol. A parancs egyetlen paramétere az y tengelyen megjelenő szöveg. \setylabel

A barenv környezetben használható parancsok leírásának végén el kell mondanunk, hogy a bar csomag ennek a környezetnek az előállítására a picture környezetet használja, ezért az ott ismertetett parancsokat is használhatjuk a környezeten belül.

8. fejezet

Szimbólumok

A pifont csomag betöltése után használhatjuk a 8.1 táblázatban látható ZapfDingBats szimbólumtáblát, amelynek lemei a \ding parancs segítségével érhetőek el. pifont \ding

A wasysym csomag által definiált szimbólumok és jelek a 8.2 és 8.3 táblázatokban láthatóak. wasysym

Az ifsym csomag betöltésével meglehetősen sok új szimbólum használatára kaphatunk lehetőséget. Ha a csomagot a weather kapcsolóval töltjük be, akkor használhatjuk a \Thermo{*n*} parancsot a  jelek előállítására, valamint a 8.4 táblázatban látható meterológiai szimbólumokat. Ha a csomagot a clock kapcsolóval töltjük be, akkor rendelkezésünkre áll a \showclock parancs, amellyel az adott időt mutató órát készíthetünk. A \showclock{5}{30} parancs például  eredményt ad, vagyis egy órát, amely 5 óra 30 percet mutat. A kapcsolóval használhatóvá válnak még a 8.5 táblázat jelei. ifsym weather clock \showclock

A misc kulcsszóval töltjük be a csomagot, akkor használhatjuk a \Cube parancsot \Cube{*n*} formában a  , , ... jelek előállítására. A kapcsoló hatására használhatjuk a 8.6 táblázat jeleit. misc \Cube

Ha a csomag betöltésekor használjuk a electronic kulcsszót, akkor digitális jelalakok rajzolására alkalmas eszközt kapunk. Ezzel pl. vagy formájú jelalakokat írhatunk le (8.7 táblázat). Ugyanakkor rendelkezésünkre áll a hétszegmenses kijelzőkből álló számkészlet, amellyel - 12.56 formában jeleníthetünk meg számokat. electronic

A marvosym csomag szimbólumait a 8.8 táblázatban láthatjuk. marvosym

33	✁	34	✂	35	✃	36	✄	37	✆	38	✇
39	✈	40	✉	41	✌	42	✍	43	✎	44	✏
45	✐	46	✑	47	✒	48	✓	49	✔	50	✕
51	✖	52	✗	53	✘	54	✙	55	✚	56	✛
57	✜	58	✝	59	✞	60	✟	61	✠	62	✡
63	✢	64	✣	65	✤	66	✥	67	✦	68	✧
69	✩	70	✪	71	✫	72	✬	73	✭	74	✮
75	✯	76	✰	77	✱	78	✲	79	✳	80	✴
81	✵	82	✶	83	✷	84	✸	85	✹	86	✺
87	✻	88	✼	89	✽	90	✾	91	✿	92	❀
93	❁	94	❂	95	❃	96	❄	97	❅	98	❆
99	❇	100	❈	101	❉	102	❊	103	❋	104	❌
105	❍	106	❎	107	❏	108	❐	109	❑	110	❒
111	❖	112	❘	113	❙	114	❚	115	❛	116	❜
117	❝	118	❞	119	❡	120	❢	121	❣	122	❤
123	❥	124	❦	125	❧	126	❨	161	❩	162	❪
163	❫	164	❬	165	❭	166	❮	167	❯	168	❰
169	❱	170	❲	171	❳	172	❴	173	❵	174	❶
175	❷	176	❸	177	❹	178	❺	179	❻	180	❼
181	❽	182	❾	183	❿	185	➀	185	➁	186	➂
187	➃	188	➄	189	➅	190	➆	191	➇	192	➈
193	➉	194	➊	195	➋	196	➌	197	➍	198	➎
199	➏	200	➐	201	➑	202	➒	203	➓	204	➔
205	→	206	↔	207	↕	208	➘	209	➙	210	➚
211	➛	212	➜	213	➝	214	➞	215	➟	216	➠
217	➡	218	➢	219	➣	220	➤	221	➥	222	➦
223	➧	224	➨	225	➩	226	➪	227	➫	228	➬
229	➭	230	➮	231	➯	232	➱	233	➲	234	➳
235	➴	236	➵	237	➶	238	➷	239	➸	241	➹
242	➺	243	➻	244	➼	245	➽	246	➾	247	➿
248	➸	249	➹	250	➺	251	➻	252	➼	253	➽
254	➾										

8.1. táblázat. A ZapfDingbats szimbólumkészlet

♈	\vernal	☊	\ascnode	☋	\descnode
○	\fullmoon	●	\newmoon	☾	\leftmoon
☽	\rightmoon	⊙	\astrosun	☿	\mercury
♀	\venus	♁	\earth	♂	\mars
♃	\jupiter	♄	\saturn	♅	\uranus
♆	\neptune	♇	\pluto	♈	\aries
♉	\taurus	♊	\gemini	♋	\cancer
♌	\leo	♍	\virgo	♎	\libra
♏	\scorpio	♐	\sagittarius	♑	\capricornus
≈≈	\aquarius	♓	\pisces		

8.2. táblázat. Asztronómiai jelek

♂	\male	♀	\female	œ	\currency
Σ	\recorder	ø	\clock	✠	\kreuz
☺	\smiley	☹	\frownie	☼	\sun
✓	\checked	🔔	\bell	¢	\cent
⌀	\diameter	♪	\eighthnote	𝅗𝅥	\halfnote

8.3. táblázat. Egyéb jelek

🌩	\Lightning	🌧	\Rain	🌤	\FilledSunCloud
🌫	\ThinFog	☁	\Cloud	🌧	\RainCloud
🌦	\WeakRain	❄	\Snow	🌦	\WeakRainCloud
☁	\FilledCloud	🌨	\SnowCloud	🌧	\FilledRainCloud
🌤	\HalfSun	☉	\Sun	🌨	\FilledSnowCloud
🌫	\Fog	🌥	\SunCloud		

8.4. táblázat. Meterológiai jelek

	\Taschenuhr		\VarTaschenuhr
	\StopWatchStart		\StopWatchEnd
	\Interval		\Wecker
	\VarClock		

8.5. táblázat. Órák

	\Irritant		\StrokeOne
	\Fire		\StrokeTwo
	\Letter		\StrokeThree
	\Radiation		\StrokeFour
	\Telephone		\StrokeFive

8.6. táblázat. Veszélyt jelző szimbólumok és strigulák

	\RaisingEdge		\FallingEdge
	\ShortPulseHigh		\ShortPulseLow
	\PulseHigh		\PulseLow
	\LongPulseHigh		\LongPulseLow
	\textifsym{lh}		\textifsym{LH}
	\textifsym{m<d}		\textifsym{d>m}
	\textifsym{M<<D}		\textifsym{D>>M}
	\textifsym{12.7}		\textifsym{-2b3}

8.7. táblázat. Elektronikus jelalakok

✂	\Cutright	---	\Cutline	✁	\Cutleft
€	\EUR	€	\EURtm	€	\EURhv
CE	\CEsign		\Mobilefone	☎	\Telefon
⚽	\Football		\Bicycle		\Bat
	\Womanside		\Womanfront		\Manside
	\Manfront		\Stopsign	♡	\Heart
▬	\Flatsteel	◘	\Squarepipe	▯	\Rectpipe
⌊	\Lsteel	⌶	\TTsteel	○	\Circpipe
⊤	\Tsteel	☉	\Sun	☽	\Moon
☿	\Mercury	♀	\Venus	♂	\Mars
♃	\Jupiter	♄	\Saturn	♅	\Uranus
♆	\Neptune	♇	\Pluto	♈	\Aries
♉	\Taurus	♊	\Gemini	♋	\Cancer
♌	\Leo	♍	\Virgo	♎	\Libra
♏	\Scorpio	♐	\Sagittarius	♑	\Capricorn
♒	\Aquarius	♓	\Pisces		

8.8. táblázat. A marvosym szimbólumkészlet

Ajánlott irodalom

1. VIRÁGVÖLGYI PÉTER: *A tipográfia mestersége számítógéppel*, Osiris Kiadó (2002), Budapest, ISBN: 9-633795-29-X

2. PERE LÁSZLÓ: *Linux felhasználói ismeretek I.*, Kiskapu (2002), Budapest, ISBN: 9-789639-301375

3. KONIORCZYK MÁTYÁS, ÁDÁM PÉTER: *Mérési adatok feldolgozása*: Pécsi Tudományegyetem, Pécs (2002)

4. MICHEL GOOSSENS – SEBASTIAN RAHTZ – FRANK MITTELBACH: *The LaTeX graphics companion*, Addison-Wesley (1997), US, ISBN: 0-201-85649-4

5. LESLIE LAMPORT: *LaTeX user's guide and reference manual*, Addison-Wesley (1994), US, ISBN: 0-201-52983-1

6. LINDA LAMB–ARNOLD ROBBINS: *Learning the vi Editor*, 6th Edition O'Reilly (1998), US, ISBN: 1-56592-426-6

7. ARNOLD ROBBINS: *vi Editor Pocket Reference*, O'Reilly (1999), US, ISBN: 1-56592-497-5

Tárgymutató