LaTeX dokumentumszerkesztés II.

Fejléc és lábléc, listák

Vadon Viktória

2023/24/I. félév

1. Fejléc és lábléc
   - Oldalstílus
   - A `fancyhdr` csomag
   - Hasznos parancsok
   - Elválasztó vonalak
   - Új és módosított oldalstílusok

Oldalstílus I

- futó fejléc és lábléc: tartalom, információ a felső és/vagy alsó margóban, mint oldalszám, aktuális fejezet/szakasz, intézmény logója, ...
- megközelítés: „sablon”, ún. *oldalstílus* (pagestyle) vezérli a futó fejlécet és láblécet
- `twoside` (ld. oldalbeállítás) dokumentum esetén páros-páratlan oldalon eltér(het) a sablon
- `oneside` dokumentum esetén minden oldalon azonos sablon

Beépített oldalstílusok

oldalstílus	fejléc bal/belső	fejléc jobb/külső	lábléc
empty	üres	üres	öres
plain	üres	üres	oldalszám
headings	aktuális szakasz*	oldalszám	üres
myheadings	*egyéni**	oldalszám	üres

Oldalstílus II

- oldalstílus alkalmazása
  - egész dokumentumra (adott ponttól) pl. `\pagestyle{headings}`
  - csak az aktuális oldalra (utána visszaáll az előző) pl. `\thispagestyle{empty}`
  - egyes beépített parancsok láthatatlanul meghívják a `\thispagestyle{}` parancsot
    - címoldalhoz (`titlepage` oldalbeállítással `\maketitle`) `empty` pagestyle
    - címsorok (`notitlepage` oldalbeállítással `\maketitle`) esetén első oldal `plain`

Oldalstílus III

- myheadings pagestyle
  - fejléc bal/belső mezője: egyéni tartalom
  - oneside dokumentum esetén `\markright{Saját fejléc}` paranccsal adható meg/írható felül
    - !! `right`, mert nyomtatásban jobb oldal, de a belső, avagy *bal* mező a miénk
    - amelyik oldalon kiadjuk, már érvénybe lép
  - `twoside` dokumentum esetén

    ```
    \markboth{Páros oldali fejléc}%
    {Páratlan oldali fejléc}
    ```

    - itt is él a `\markright{}` parancs, ami a páratlan fejlécet külön felül tudja írni
    - a betáplált páratlan ill. páros fejléc kiíratása: `\rightmark`, `\leftmark` parancsokkal
    - törlés: parancs meghívása üres argumentummal

Oldalstílus IV

- *haladó*: alapvetően a bal/belső mező a miénk, de trükközve középre is tudunk írni:
  - `\hfil` parancs: vízszintesen töltsd ki a rendelkezésre álló teret
  - oneside dokumentum

    ```
    \markright{fejléc bal\hfil fejléc közép}
    ```

  - azért működik, mert a háttérben ő is egyéni mező + `\hfil` + oldalszám módon tölti ki a fejlécet!
  - a két `\hfil` parancs között egyenlően osztja a térközt
  - twoside dokumentum

    ```
    \markboth{páros fejléc közép\hfil jobb}%
    {páratlan fejléc bal\hfil közép}
    ```

Oldalstílus V

- headings pagestyle
  - fejléc bal (`oneside` dokumentum) vagy belső (`twoside` dokumentum) mezője: aktuális szakasz*
    - oneside document esetén legmagasabb szintű címsor (article: section, book: chapter)
    - twoside document esetén páros (bal) oldalon legmagasabb szintű címsor, páratlan (jobb) oldalon második legmagasabb (article: section+subsection, book: chapter+section)
  - itt is élnek a `\markright{}` (és `\markboth{}{}`) ill. `\rightmark` (és `\leftmark`) parancsok
    - de a `\section` és hasonló parancsok láthatatlanul meghívják a `\markright{}` parancsot és ismét beírják magukat a futó fejlécbe
  - számozatlan egységek nem jelennek meg a fejlécben!
    - de! ilyenkor az előző címsor ott ragad!
    - `\markright{}` (vagy `\markboth{}{}`üres argumentumos hívással törölhető a korábbi

A fancyhdr csomag

- ha a fentiektől eltérőt, vagy jobban testre szakható sablont szeretnénk:
  - `fancyhdr` csomag – dokumentáció (link): `\usepackage{fancyhdr}`
  - !! a `geometry` csomag betöltése és beállítása *után* töltsük és állítsuk be a `fancyhdr`-t!
  - ezzel érkezik egy személyre szabható `fancy` pagestyle: `\pagestyle{fancy}`
- `fancy` oldalstílus alapértelmezett sablonja:

fejléc bal/belső	fejléc jobb/külső	lábléc
szakasz (vagy fejezet)	alszakasz (vagy szakasz)	oldalszám (középen)

- illetve, a fejléc alá rajzol egy elválasztó vonalat! (kivéve, ahol cím/fejezetcím van)

A fancy oldalstílus I

- a sablon „felépítése”:
  - oneside dokumentum: fejléc/lábléc × bal/közép/jobb = 6 mező
  - twoside dokumentum: fentiek × páros/páratlan oldal = 12 lehetséges mező
  - ezek akár egyesével is beállíthatók (azért sok üres szokott lenni)
- az egyes mezőket betűk kombinációi írják le:
  - H (head) = fejléc, F (foot) = lábléc
  - L (left) = bal, C (center) = közép, R (right) = jobb
  - O (odd) = páratlan, E (even) = páros
  - tetszőleges sorrendben és kombinációban használhatók – ha 1-2 betűs kombinációt használunk, akkor minden lehetséges mező értendő, pl. H a fejléc minden mezőjét jelenti

A fancy oldalstílus II

- beállítás: `\fancyhf[]{}` (fancy head-foot) parancs
  - kötelező argumentum: mező tartalma – üresen hagyva törli
  - opcionális argumentum: mezők betűkombinációi, vesszővel elválasztott listában
    - amelyik mező beleférr, *mindre* vonatkozik! pl. üresen minden mező
  - pl. `\fancyhf{}` minden mezőt töröl
  - pl. `\fancyhf[H,CF]{}` H = a fejléc minden mezőjét *és* CF = a lábléc középső mezőjét törli, stb.
  - pl. `fancyhf[RFO,LFE]{\thepage}` – lábléc külső mezője (páratlan jobb, páros bal) legyen az oldalszám
  - itt `\thepage`: oldalszám kiíratása
  - `\fancyhead[]{}`, `\fancyfoot[]{}` parancsok: a H, F betűk megspórolhatók
    - pl. `\fancyhead[L,R]{}` = `\fancyhf[HL,HR]{}`

Hasznos parancsok fejléc és lábléc konstrukciójához

Fejléc-lábléc sablon építéséhez:

- `\thepage`: oldalszám kiíratása
- itt is élnek a már látott `\rightmark` és `\leftmark` parancsok
  - oneside dokumentumban is létezik mindkettő!
  - `\leftmark` a legnagyobb, `\rightmark` a második legnagyobb címsor (a `\section` és hasonló parancsokkal beírják magukat)
  - `\leftmark` automatikusan nagybetűs (LaTeX a háttérben ráereszt egy `\MakeUppercase` parancsot)
    - ha ezt nem szeretnénk: `\nouppercase{\leftmark}` (csak a `\MakeUppercase` parancsot teszi hatástalanná, így eredeti formázás marad)

Számozatlan egységek a fejlécben

- számozatlan egységek *nem* írják felül a `\leftmark`-ot és `\rightmark`-ot!
- kézzel a `\markright{új rightmark}` vagy `\markboth{új leftmark}{}` paranccsal írhatjuk felül (vagy üresen hagyva törölhetjük)
- `\leftmark`-ot nem tudjuk külön – direkt! ha a nagyobb egység változik, a kisebbnek is muszáj!
  - ezért az üres kapcsos zárójel a `\markboth`-ban, töröljük a `\rightmark`-ot is
- ha minden jól megy, a következő számozott egység ismét beírja magát

Vonalzók

- figyeljük meg, alapértelmezésben a fejlécet egy vonal választja el a törzstől, a láblécet viszont nem
  - ki/bekapcsolás: diretkben nincs, a vonalvastagság módosításával!
  - módosítás: `\renewcommand{\headrulewidth}{0.4pt}` (ez az alapértelmezett vastagság)
  - lábléchez: `\footrulewidth`
  - törlés: állítsuk 0 vastagságúra

Módosított és új oldalstílusok I

- emlékeztető: `\maketitle` a háttérben használja a `\thispagestyle` parancsot, `plain` vagy `empty` stílussal
- ez nem feltétlen konzisztens a személyre szabott fancy stílusunkkal!
  - példánkban a `fancy` stílusban a külső oldalra tettük az oldalszámot, a `plain` stílusban viszont középen van
- megoldás: `fancydhr`-rel tudjuk módosítani a `plain` stílust
  - `\fancypagestyle{plain}{parancsok}`
  - itt parancsok = `\fancyhf` és/vagy `\renewcommand{\headrulewidth}{0pt}` típusú parancs(ok sorozata)
  - !! *ne* tegyünk közéjük vesszőt, és sortörést is csak % mögé rejtve – mert egy másik parancs hasában van, parsing error (hiányzó kapcsos zárójel) lesz!

Módosított és új oldalstílusok II

- például minden mező törlése és oldalszám a lábléc külső mezőjébe:

```
\fancypagestyle{plain}{%
\fancyhf{}%
\fancyfoot[LE,RO]{\thepage}%
}
```

- új stílust is tudunk definiálni vele, pl. `ujstilus` néven: `\fancypagestyle{ujstilus}{parancsok}`
  - a `parancsok` részben a fenti szintaxis érvényes!
  - alapul vehetünk egy létező stílust, pl. a plain-ből kiindulva: `\fancypagestyle{ujstilus}[plain]{parancsok}`
  - alapértelmezésben a már definiált `fancy`-re alapoz

2 Listák

- Alapok
- Többszintű listák
- Listák testreszabása – `enumitem` csomag
- Egysoros listák
- Listák formázása – alapok
- Egysoros listák formázása
- Számozás folytatása

Alapok I

- beépített funkcionalitás, hogy listákat tudunk készíteni
- logika HTML-hez hasonló

LaTeX felsorolás

```
\begin{itemize}
\item listaelem
\item másik listaelem
\end{itemize}
```

HTML felsorolás

```
<ul>
<li>listaelem</li>
<li>másik listaelem</li>
</ul>
```

- listák típusai:
  - fenti `itemize`: felsorolás (számozatlan lista)
  - `enumerate`: számozott lista (HTML: ol)
  - `description`: ún. leíró, címszavas lista (HTML: dl)

Alapok II

- az `\item` parancs csak a listakörnyezeten belül él!
- és a listának legalább egy `\item`-et tartalmaznia kell
- de az `\item` lehet üres
- és egy `\item`-en belül kezdhetünk új bekezdést
- az `\item` parancsnak van egy opcionális argumentuma
  1. pl. `\item[*]`, `\item[!!]`
     - ezzel *csak az adott listaelemre* felülírható a listaelem jele
  2. számozott listában az `\item[]` után úgy folytatódik a számozás, mintha az `\item[]` ott sem lenne!
- `description`, leíró listánál a szócímke is `\item[címke]` módon adható meg

  címke leíró lista eleme

Többszintű listák I

- többszintű listák is készíthetők, a listák egymásba ágyazásával:

```
\begin{enumerate}
  \item egy
  \begin{itemize}
    \item második szint!
  \end{itemize}
  \item kettő
\end{enumerate}
```

- mint a példán is látszik, különböző listatípusok is egymásba ágyazhatók
- a beágyazás mélysége korlátozott – „too deeply nested” error

Többszintű listák II

- azonos típus esetén is különbözőképpen jelöli a szinteket
  - `itemize` listánál különböző jelek[1]
  - `enumerate` listánál arab számok, római számok, betűzés, stb.
- legalább egy `\item` kell a következő szint megkezdése előtt
  - logikusan is, miért hagynánk ki egy szintet
  - ha nem, „missing `\item`” error
- ha egy másik listapont alatt is szeretnénk alpontokat, ott újra kell a `\begin{itemize}` (azaz, egy külön listát ágyazunk be megint)

[1] diasorban nem, mert a beamer különleges :)

Listák testreszabása – enumitem

- „gyalogosan” is lehet, de enumitem csomaggal egyszerűbb
- \usepackage{enumitem} – dokumentáció
- célok
  - formázás testreszabása: számozás formátuma, hivatkozási formátum, térköz és elrendezés
  - korábbi listák folytatása
  - inline, „egysoros” listák
  - beágyazási mélység módosítása(!)
  - stb.

Egysoros listák

- mit jelent? egy sorban, némi térköz kihagyásával, utána a felsorolás/számozás jelével folytatja a következő listaelemmel
- például[2]
  • egy • kettő • három
- mikor használjuk? ha a felsorolás elemei rövidek (és nincs belőlük túl sok) – különben átláthatatlan lesz!
- `inline`/„egysoros” listák engedélyezése: `inline` csomag opció (`\usepackage[inline]{enumitem}`)
- *-ozott verzióként érhetők el, pl. `\begin{itemize*}`

[2] közelítőleges demonstráció; beamer-rel nem kompatibilis, ezért a tényleges hatást nem tudom bemutatni

Lista formátuma I

- felsorolás szimbóluma `label=szimbólum`, pl. `\begin{itemize}[label=\textbullet]`
- számozás formátuma

  `\begin{enumerate}[label=formátum]`

  ahol a `formátum` részei:
  - tartalmaznia kell egy számformátumot
    - `\arabic*` – arab számok
    - `\alph*`, `\Alph*` – kis- vagy nagybetűk
    - `\roman*`, `\Roman*` – kis-vagy nagybetűs római számok
  - tartalmazhat még egyéb formázási karaktereket, pl. pont, zárójel
  - pl. `\begin{enumerate}[label=(\arabic*)]` hatása (1), (2), stb.

Lista formátuma II

- alternatív (a kettő együtt nem működik!): `label*` kulccsal *hozzáfűzés* az előző szint számához, pl.

```
\begin{enumerate}[label=\arabic*.]
\item ...
  \begin{enumerate}[label*=\alph*.]
  ...
```

- hatás: első szinten (külső enumerate) `1.`, `2.`, stb.
  - pl. `1.` elem alpontjai a második szinten (belső enumerate) `1.a.`, `1.b.`, stb.
- címke betűstílusa – leíró listánál fontos, de számozásnál is alkalmazható
  - `font={formátum}`
  - `formátum`-ban az *argumentum nélküli* betűstílus parancsok használhatók!
  - pl. `\begin{description}[font={\scshape}]`: kiskapitális betűk

Lista formátuma III

- sorszám/jel/szócímke igazítása
  - a címke egy neki szánt láthatatlan *dobozba* kerül, a margótól kb. a listaelem szövegéig tart (kis térköz kerül közéjük)
  - ezen a dobozon belül befolyásolhatjuk az igazítást
  - alapértelmezés: `align=right` – a címkék jobb széle van egy vonalban, akkor is, ha különböző hosszúak – számozásnál ajánlott
  - vagy `align=left` – balra
  - `align=parleft` – balra + tördelés bekapcsolása*, pl. ha leíró listában hosszúak a szócímkék
- függőlegesen sűrített listák:
  - `noitemsep` kapcsoló: csak az elemek közti térközt törli
  - `nosep` kapcsoló: törli a térközt az elemek között, *továbbá* a lista előtt és után

Egysoros listák formázása

formázás: `\begin{itemize*}[opciók]`, opcionális argumentum

- `afterlabel`: címke és felsoroláselem között, alapért. `afterlabel=~`, avagy törhetetlen szóköz
- `itemjoin`: két elem között, alapért. `itemjoin=␣`, avagy szóköz
- utolsó elem előtt `itemjoin*`, pl. `itemjoin*={␣és␣}`
- számozás, címke stílusa: ld. 5 szakasz
- például

```
\begin{itemize*}[afterlabel=~,%
itemjoin=\hspace{1em},%
itemjoin*={\hspace{1em}és }]
\item ...
\end{itemize*}
```

Számozás folytatása

- mindentől függetlenül, egy lista tetszőleges számról is indítható, pl. 3-ról: `start=3`
- `\begin{enumerate}[resume]`: onnan folytatja a számozást, ahol az *előző* számozás abbahagyta
  - ha annak a listának a formátumát is szeretnénk áthozni, `resume*`
- ha egy *korábbi* (nem legutolsó) számozott listát szeretnénk folytatni: szériát hozhatunk létre
  - a korábbi, folytatni kívánt listához `series=név`, ahol `név` tetszőleges string
  - széria folytatása: `resume=név`, vagy `resume*=név`-vel – a különbség itt is a formátum másolása a *-ozott verzióban
  - akár több részletben is – széria nem feltétlen csak két felsorolás, mindig a legutolsótól folytatjuk
- egyéb (kézi) manipuláció a számozással (pl. egy-egy sorszám átugrása) számlálók segítségével, 1 szakasz

3 Haladó listakezelés

- Számlálók
- Leíró listák formázása
- Listák elrendezése, méretezése, térközök
- Egy leíró lista példa formázással
- Listák alapbeállítása, saját listák

Számlálók

számláló vagy *counter* : LaTeX változótípus

- értékei egész számok (tipikusan nemnegatívakkal dolgozik, de lehet negatív is)
- támogatott műveletek:
  - létrehozás és inicializálás 0-ra `\newcounter{mycounter}`
  - kiíratás `\themycounter` (alapért. arab számokkal)
    - kiíratás egyéb számformátummal:
    - kis-és nagybetűk: `\alph{mycounter}`, `\Alph{mycounter}` – 1-től 26-ig definiáltak
    - kis-és nagybetűs római számok: `\roman{mycounter}`, `\Roman{mycounter}` – 1-től definiáltak
    - szimbólumok `\fnsymbol{mycounter}` – 1-től 9-ig definiáltak
  - növelés 1-gyel `\stepcounter{mycounter}`
  - egész szám hozzáadása `\addtocounter{mycounter}{3}` – negatív hozzáadásával csökkenthető is
  - adott értékre állítás `\setcounter{mycounter}{5}`
  - érték lekérdezése belső változóként való használatra `\value{mycounter}`

Listák számlálói

- számozott lista minden egyes szintjéhez tartozik egy counter:
- `enumi`, `enumii`, `enumiii`, `enumiv` – `enum` + lista szintje kisbetűs római számmal
- az `\item` parancs tulajdonképpen `\stepcounter{enumi}` + `\theenumi`
  - kivéve: `\item[]` nem léptet!
  - formátum a kiíratás parancsába égetve, `\theenumi` = `\aracic{enumi}.`, `\theenumii` = `(\roman{enumii})`, stb.
  - formátum változtatása „gyalogosan” és csak aktuális listára: `\renewcommand{\theenumii}{\roman{enumii}.}`
- manipuláció: számláló műveleteivel
  - léptetés 1-gyel: `\stepcounter{enumi}`
  - léptetés többel, pl. 2-vel: `\addtocounter{enumi}{2}`
  - tetszőleges új érték megadása: `\setcounter{enumi}{3}`, új értéke 3 – itt vegyük figyelembe, hogy a következő `\item` először léptetni fog!

Leíró listák formázása I

- leíró lista beépített „stílusai” – méretezési (ld. 3 szakasz) és igazítási opciók kombinációja (szükség szerint méretezési opciókkal együtt használandók)
  - `style=standard`: az alap documentclass-ok leíró lista stílusa
    - a szócímke dobozba van foglalva – ezen belül nincs automatikus tördelés, ha nem fér ki egy sorba!
    - szöveg a szócímkével egy sorban folytatódik
    - üres szócímke esetén a szöveg balra csúszik a címke helyére – mintha függő behúzás lenne
  - `standard` az alapértelmezés, a többinél az ehhez képesti különbséget emeljük ki
  - `style=unboxed`
    - a szócímke nincs dobozba zárva – automatikus tördelés garantált
  - `style=sameline`
    - szócímke nincs dobozban
    - üres címke esetén a szöveg nem csúszik ki balra

Leíró listák formázása II

- `style=nextline`
  - ha a címke nem fér a margóra, a szöveg a következő sorban folytatódik
  - üres címke esetén a szöveg nem csúszik ki balra
- `style=multiline`
  - a címke egy margó szélességű dobozba kerül, de akár több sorba
  - kifejezetten ajánlott az átméretezés (ld. 3 és 4 szakaszok) – elválasztással együtt is nagyon kicsi a hely a címkének!

Listák elrendezése, méretezése, térközök I

- ábra a dokumentáció 3. oldalán

Lista függőleges térközei

- `topsep`: térköz a lista előtt és után
- `partopsep`: ha a listát üres sor előzi meg (azaz új bekezdésbe kerül), ennyi *extra* térköz kerül elé és utána is
- `parsep`: egy listaelemen belüli bekezdések közti térköz
- `itemsep`: listaelemek közé a `parsep`-en felül ennyi extra térköz kerül

Listák vízszintes elrendezése

- `rightmargin`: (teljes lista) jobb margója
  - *extra* margó a *környező szöveghez* képest
  - azaz nem kell beleszámolni az oldal margóját
  - többszintű listánál halmozódik

Listák elrendezése, méretezése, térközök II

- `leftmargin`: listaelem szövegtörzsének bal margója, *a 2. sortól* – szintén a *környező szöveghez* képest
- `itemindent`: a listaelem első sora ennyivel beljebb kerül a `leftmargin`-hoz (azaz a későbbi sorokhoz) képest(!)
  - `listparindent`: az `itemindent` megfelelője a listaelem későbbi bekezdéseiben
- `labelindent`: a címke (sorszám/jel/szócímke) behúzása, a *környező szöveghez* képest
- `labelwidth`: címke (vagy dobozának) szélessége
- `labelsep`: címke távolsága a listaelem szövegétől (1. sortól)

Listák elrendezése, méretezése, térközök III

!! nehézség: túlhatározott rendszer!
listaelem első sorának behúzása
= `leftmargin + itemindent`
= `labelindent + labelwidth + labelsep`

- alapértelmezésben a `labelindent`-et számolja a többiből
  - ez az új mennyiség az `enumitem` csomagban, a többi hossz létezik alap LaTeX-ben is
- számolandó mennyiség kijelölése: `!` vagy `*` értékként, pl. `leftmargin=*`
  - `!` értékkel másik 4-et meg kell adni (vagy marad az alapértelmezett)
  - `*` esetén 3-at kell megadni (vagy marad az alapértelmezett), `labelwidth`-et (címkeszélességet) (is) automatikusan határozza meg a *címkék tartalmából*

Listák elrendezése, méretezése, térközök IV

- `labelwidth` számolása a címkéből:
  - „legszélesebb címke” megadása `widest=string`, `string` szélességét veszi címkeszélességnek
  - `enumerate`-re címke stílusa elég: ismert a legszélesebb betű/római szám
  - ha ettől keskenyebb is elég, pl. nem megyünk el `viii`-ig, lehet pl. `widest=iv`
  - ! nem feltétlen a ténylegesen legszélesebb címkét érdemes megadni – pl. leíró listánál választhatjuk a leghosszabb előforduló szót, de hagyjuk a többszavas címkéket tördelni

Példa leíró listára többsoros címkével

```
\begin{description}[align=parleft,%
leftmargin=*,widest={hosszabb}]
\item[címke] \hulipsum[1]
\item \hulipsum[2]
\item[ez egy hosszabb címke!] \hulipsum[3]
\end{description}
```

- `align=parleft`: többsoros címke
- `leftmargin=*`: `leftmargin` azaz bal margó számítása a többi méretből *és* `labelwidth` azaz címke(doboz) szélessége *automatikus*
  - `widest={hosszabb}`: a `hosszabb` szó szélességéhez igazít

Listák alapbeállítása, saját listák I

- listatípushoz *alapbeállítások* megadása – preambulumban:

  `\setlist[típusok,szintek]{beállítások}`

  - típusok,szintek
    - pl. `\setlist[itemize,enumerate,1,2]{...}`
    - minden lehetséges típus × szint kombinációra érvényes! példánkban `enumerate` 1-2 szintjeire és `itemize` 1-2 szintjeire vonatkozik
    - ha kihagyjuk a típust és/vagy szintet: mindre vonatkozik
  - beállítások: már látott, listáknál helyileg is használható `enumitem` parancsok
    - pl. `\setlist[description]{nosep,style=nextline}`
  - ha több `\setlist` is hatna egy listára: hatásuk kumulatív; ellentmondás esetén a specifikusabb nyer
  - az alapbeállítások helyileg továbbra is felülírhatók, megszokott `\begin{enumerate}[opciók]` szintaxis

Listák alapbeállítása, saját listák II

- saját listatípusok definiálása – létező listatípus klónozásával
- például, egy `enumerate` klón:

  `\newlist{enuma}{enumerate}{5}`

  - `enuma`: új listatípus neve, tetszőleges string (csak ne legyen létező LaTeX parancs/kulcsszó!)
  - `enumerate`: melyik listát klónozzuk – `inline`, avagy egysoros verziók is használhatók!
  - 5: max beágyazási mélység
    - ne állítsuk fölöslegesen nagyra, mert létrehozza minden szinthez a számlálókat, példáunkban `enumai-enumav` számlálókat, ami foglalja a memóriát

    !! 5-nél nagyobbhoz először általánosan meg kell növelni a határt: pl. `\setlistdepth{7}`

Listák alapbeállítása, saját listák III

!! létrehozáskor az alapbeállítások üresek, a címkék nem definiáltak, *valamit* meg kell adnunk, hogy használni tudjuk! pl. legyen `enuma` (minden szinten) kisbetűs számozott lista:

```
\setlist[enuma]{label=\alph*)}
```

- használat szokásos módon:

```
\begin{enuma}
\item ...
\end{enuma}
```

- itt is megadhatunk opcionális argumentumban helyi formázást
- létező listák *újradefiniálása*: `\renewlist`, `\newlist`-tel azonos szintaxissal
  - miért kell, ha `\setlist` létezik? hogy a beágyazás mélységét is meg tudjuk változtatni!
  - !! itt is törli a címkék alapbeállítását!

Listák alapbeállítása, saját listák IV

- egy összetettebb példa, nem minden szinten azonos formázással:
- jogszabályokhoz lista, \S = § jellel, és a mélyebb szinteken ponttal egymás után fűzött arab számokkal:

```
\newlist{jogi}[enumerate]{3}
\setlist[jogi]{font=\ttfamily,label*=.\arabic*}
\setlist[jogi,1]{label=\S\arabic*}
```

jogi lista *első* szintjére mi lesz érvényes?

- ha nincs ellentmondás, mindkét `\setlist` parancsai vonatkoznak rá
- ellentmondásos kulcsok esetén a specifikusabb felülírja
- példánkban `font=\ttfamily` és `label=\S\arabic*` lesz