LaTeX float-ok

Úszó objektumok: ábrák, táblázatok, programkódok

Vadon Viktória

2023/24/I. félév

Mi az az úsztatás?

- mi az, hogy úsztatás?
- probléma: pl. egy képet, vagy nagy táblázatot (tördelhetetlen objektumot) szeretnénk elhelyezni, hová tegyük?
- lehetséges megoldások:
  - alapértelmezés: az objektum a szöveggel egy sorba kerül
    - pro: előfordulási helyén marad
    - kontra: de ha nem fér ki a lap aljára, akkor a következő oldalra csúszik, lap alja üres – csúnya!
  - áthelyezés kézzel – többször is szükséges lehet, ahogy fejlődik a dokumentum – nehézkes
  - LaTeX-re bízzuk, hogy dinamikusan helyezze át – ezt hívjuk úsztatásnak (mert az objektumot kiemeljük a szöveg *folyamából*)
    - ilyenkor a float-ot először memóriába olvassa, és adandó alkalommal írja ki

Mit tudnak még a float-ok?

- az úsztatás opcionális
  - alternatíva: szöveggel egy sorba helyezett kép/táblázat/stb.
  - ha úsztatni akarunk, a létező objektumot (kép, táblázat, stb.) egy *extra* úszó környezetbe tesszük
- miért javasolt az úszó környezet?
  - nyilván a dinamikus áthelyezés
  - lehetőséget ad az objektum feliratozására, számozására
  - ha van sorszám, akkor lehet `\label`-t elhelyezni és hivatkozni is rá!
  - LaTeX nyilvántartja őket, automatikusan listázhatók az úszó objektumok, tartalomjegyzékhez hasonlóan
  - hogy mindezt hogyan, azt később megnézzük (objektum típusától függően beépített vagy nem, változhat a szintaxis, stb.)

2 Képek

- Képek beszúrása – úsztatás nélkül
- Képek úsztatása
- Float elhelyezése
- Képfelirat
- Elrendezés, helykitöltés
- Részábrák

Képek beszúrása I

- `\usepackage{graphicx}` kell hozzá!
- ismert formátumok: .jpg, .png, .pdf
- `\includegraphics[opciók,méretezés]%`
  `{relatív/elérési/út/képnév}`
- *nem* úsztat, „szöveggel egy sorba” szúr be – akár önmagában is használható, ha úgy látjuk jónak
- `relatív/elérési/út/fájlnév`
  - ha `TEX/3ora/3ora.tex` fájlban dolgozunk
  - és `TEX/forras/kep.png`-t szeretnénk beilleszteni
  - relatív elérés: 1 mappát föl (`..`), be a `forras` mappába: `\includegraphics{../forras/kep}`
  - fájl kiterjesztés (sokszor) elhagyható

Képek beszúrása II

- `méretezés` – opcionális
  - `scale=0.5`: kép eredeti méretének 0.5-szöröse
    - lehet nagyítani, ill. negatív scale-lel 180°-is forgatás
    - nehézkes lehet használni, ha az eredeti méret nem ismert
  - `width=5cm` vagy `width=0.8\linewidth`: szélesség beállítása abszolút hosszra (cm, mm, pt), vagy sorszélesség (tört)részére
    - hivatkozáshoz hasznos lehet még `\columnwidth`: oszlop szélessége
  - hasonlóképp `height=...`
    - ha *forgatjuk*, használjuk helyette a `totalheight` kulcsot
  - ha adunk meg `width`-et *és* `height`-et is, nyújtja a képet

Képek beszúrása III

- `opciók` – opcionális
  - `keepaspectratio`: méretarány megtartása – nem nyújt, csak nagyít/kicsinyít
    - ha `width`-et és `height`-et is megadunk, *maximumnak* veszi őket; az egyiket veszi csak fel pontosan, hogy ne kelljen nyújtani a képet
  - forgatás szöge `angle`=90, +90° (óramutatóval ellentétesen)
  - forgatás középpontja – mint első óráról `\rotatebox`-nál: `origin=..`, 1-2 betűs kombináció:
    - `l`, `r`: bal vagy jobb él
    - `t`, `b`: felső vagy alsó él, `B` alapvonal
    - `c`: közép – önmagában használva a kép közepe függőlegesen *és* vízszintesen is; valamivel kombinációban az adott él közepe
    - alapértelmezés `origin=lB`
- keretezés, egyéb transzformációk (pl. tükrözés, nyújtás) első óráról: `\framebox{\includegraphics{kep}}`, stb.

Képek úsztatása I

```
\begin{figure}[bt]
\centering
\caption{Felirat/képaláírás}
\label{fig:kepek}
\includegraphics[width=.4\linewidth]{kep1}
\hspace{1em} % vízszintes helykihagyás
\includegraphics[width=.4\linewidth]{kep2}
\end{figure}
```

- a `figure` környezet az úsztatásról gondoskodik
  - figure-ön belül él a `\caption` parancs: felirat megadása, ő generálja a sorszámot is – bővebben ld. 4 szakasz
  - opcionális argumentum, itt `[bt]`: javaslat, hogy hová helyezze át a LaTeX – bővebben ld. 3 szakasz

Képek úsztatása II

- `figure` tulajdonságai
  - a teljes oldalszélességet kitölti, képmérettől függetlenül – csomagokkal érhető el, hogy szöveg kerülhessen mellé
  - csak egy csomagoló, az úsztatásról gondoskodik, de a tartalmával nem foglalkozik – kerülhet bele szöveg, vagy több `\includegraphics` parancs, stb.
  - a tartalom elrendezéséhez használhatók helykitöltési parancsok is, ld. lentebb az 5 szakaszban
- ábrák listázása:
  - csak a `\caption`-nel ellátott képeket tartja számon
  - lista kiíratása: `\listoffigures`, tartalomjegyzékhez hasonlóan működik
  - babel-lel magyar címet ad neki
  - vagy kézzel átnevezhető: `\renewcommand{\listfigurename}{Ábralista új neve}`

Float elhelyezése I

- `figure` opcionális argumentuma: javasolhatjuk, hogy hová helyezze át a float-ot, pl.

  `\begin{figure}[bt]`

  - akkor teljesíti, ha „elég jól néz ki” – a LaTeX tipográfiai elvárásai szerint
  - az alábbi betűk tetszőleges kombinációja használható (sorrend *nem* számít, az összeset kipróbálja az algoritmus):
  - `h`: „itt” – minél közelebb ahhoz, ahol a szövegben előfordult, akár az oldal közepén is
  - `t`: következő oldal tetején
  - `b`: oldal alján
  - `p`: külön float oldalon (amikor elég float összegyűlt)
  - `!`: „erősebben kérjük”
  - alapértelmezés: ha kihagyjuk az opcionális argumentumot, `tbp`

Float elhelyezése II

- kéthasábos dokumentum esetén `figure*` változat – lap tetejére kerül és *mindkét* hasábot elfoglalja
- `\clearpage` parancs nem csak oldaltörést tesz, kényszeríti, hogy a még memóriában lévő float-okat kiírja
- csomagokkal még több kontroll:
- `float` csomag
  - `H` betű: *mindenképpen* oda kerül, ahol a kódban van
  - ezzel nem úszik, visszatérünk a szöveggel egy sorba megoldáshoz, de elérhető `\caption`, `\label`, listázás
- `placeins` csomag
  - `\FloatBarrier` parancs[1]: float „ütköző”, ezen nem tud „átúszni” a float – azaz amit a kódban eddig beolvasott, ez előtt ki is kell írnia
  - `\usepackage[section]{placeins}` csomag opció: minden `\section` címsorba beépítve `\FloatBarrier`

[1] figyeljünk a nagybetűkre!

Képfelirat

- `\caption{Képfelirat}`: megformázza a képfeliratot + sorszámoz
  - kereszthivatkozásnak a `\caption` a horgonya, `\caption` *után* (vagy bele) kerüljön a `\label`!
- ábralistába is akkor kerül, ha van `\caption`-je
  - opcionális argumentum `\caption[rövid]{Képfelirat}`, az ábralistába a rövid cím kerül, ha van
- `\caption` és `\includegraphics` sorrendjétől függően a kép alá vagy fölé kerül a felirat
  - ízlés kérdése, de egy dokumentumon belül legyen konzisztens
- csak egyetlen `\caption` legyen egy `figure` környezeten belül!
  - gyakorlatilag lehet több is, de akkor több sorszámot kap
  - ha több kép kerül bele és külön szeretnénk őket feliratozni (és/vagy hivatkozni), használjunk *részábrákat*, ld. 6 szakasz

Elrendezés, helykitöltés I

- emlékeztető: `figure` teljes oldal- vagy hasábszélességet elfoglalja
- figure-ön belül lehet több kép, extra szöveg, stb.
- alapvetően minden egy sorba, balról jobbra; ha kifogy a helyből, akkor sortörés
- *kézi elrendezésükhöz*:
  - igazítás parancsai, pl. `\centering`, `\begin{center}`, `\end{center}` működnek
  - sortörés `\\`
  - új bekezdés üres sorral vagy `\par` paranccsal
  - függőleges helykihagyás a sortöréssel: `\\[12pt]` opcionális argumentum
  - függőleges helykihagyás: `\vspace{12pt}` – alternatív: 0.5cm, 1em, stb.

Elrendezés, helykitöltés II

- vízszintes helykihagyás: `\hspace{12pt}`
- rendelkezésre álló tér kitöltése vízszintesen: `\hfill`
  - több `\hfill` parancs esetén egyenlően osztja köztük a rendelkezésre álló teret
  - például teszteljük:

```
\begin{figure}[hbt]
\caption{Három kép}
\label{f:haromkep}
\includegraphics[width=0.3\linewidth]{kep1}
\hfill
\includegraphics[width=0.3\linewidth]{kep2}
\hfill
\includegraphics[width=0.3\linewidth]{kep3}
\end{figure}
```

  - függőleges verzió: `\vfill` (itt teljes oldalmagasságot képes kitölteni, ritkábban használt)

Részábrák I

- cél: egy `figure`-ön belül több képnek külön felirat
- `subcaption` csomag, `subfigure` = *részábrák* használatával
  - létezik `subtable` = résztáblázat is, de kevésbé releváns
- `figure`-ön *belül* használható a `subfigure` környezet:
  - `\begin{subfigure}{5cm}` – kötelező argumentum a szélessége!, `\end{subfigure}`
  - nyilván több is lehet belőle a `figure`-ben
  - opcionális argumentum a függőleges igazításhoz:
  - pl. `\begin{subfigure}[c]{5cm}`
  - a kép melyik részét illesztjük az alapvonalra/környező szöveghez:
  - `c`,`t`,`b`: a `subfigure` középső, legfelső vagy legalsó *alapvonalát*
  - `T`,`B`: `subfigure` tetejét vagy alját

Részábrák II

- `subfigure`-ön belül újra használható a `\caption`
  - sorszám helyett betűzi a részábrákat
    - ha a figure sorszáma 2, akkor a részábrák 2a, 2b, stb.
  - másképp formázza a feliratot
  - részábra, ha kapott `\caption`-t és vele sorszámot, kaphat külön `\label`-t!
    - hivatkozás rá `\ref` paranccsal: teljes szám+betű, pl. 2a
    - új: `\subref` hivatkozási parancs, csak a részábra betűjét adja vissza
- subfigure-ök elrendezése:
  - alapértelmezésben soronként tölt fel
  - itt is használhatók igazítási, helykitöltési parancsok, ld. 5 szakasz
  - subfigure-ök egymásba ágyazásával akár oszloponként is feltölthetjük!
  - és ezzel a részábra sorszáma nem kap több betűt

Részábrák: egy példa

```
\begin{figure}[hb]
\centering
\caption{ábra felirata}
  \begin{subfigure}{5cm}
  \centering
  \caption{első részábra}
  \includegraphics{kep1}
  \end{subfigure}
\\
  \begin{subfigure}{5cm}
  \centering
  \caption{második részábra}
  \includegraphics{kep2}
  \end{subfigure}
\end{figure}
```

3 Táblázatok

- Táblázatok – úsztatás nélkül
  - Példa, alapvető szintaxis
  - Oszlopok
  - Rácsvonalak, térköz
- Táblázatok úsztatása
- Táblázatok testreszabása – `array` csomag
- Cellák egyesítése

Táblázatok – példa, alapvető szintaxis I

- táblázatok készítése (úsztatás nélkül) `tabular` környezettel
- például:

```
\begin{tabular}{r|ll}
objektum & beillesztés & úsztatás \\ \hline
kép & \verb|\includegraphics| & figure \\
táblázat & tabular & table \\
\end{tabular}
```

objektum	beillesztés	úsztatás
kép	\includegraphics	figure
táblázat	tabular	table

- &: soron belül cellák elválasztása
- \\: sor lezárása
  - itt is él az opcionális argumentum, pl. `\\[.5ex]`, ezzel tehetünk extra térközt a sor után

Táblázatok – példa, alapvető szintaxis II

- ha nem úsztatjuk, releváns lehet a függőleges igazítás
  - opcionális argumentum, `\begin{tanular}[c]{r|ll}`
  - `c`, alapértelmezés – táblázat közepe igazodik a környező szöveg alapvonalához
  - alternatív: `t`, vagy `b`: táblázat legfelső, ill. legalsó sora, szintén a környező szöveg alapvonalához
- létezik `tabular*`, kötött szélességű táblázat
  - szélessége +1 kötelező argumentum (sorrendre figyeljünk!): `\begin{tabular}{12cm}{r|ll}`
  - függőleges igazítás itt is opcionális argumentum: `\begin{tabular}{12cm}[c]{r|ll}`

Oszlopok I

`tabular` kötelező argumentuma, példánkban `\begin{tabular}{r|ll}`: oszlopok deklarációja

- !! fel kell sorolni mindet*, az igazítás betűjelével
- közéjük kerülnek a függőleges rácsvonalak is
- példánkban: `r` = jobbra zárt oszlop, `|` = függőleges rácsvonal, majd `ll` = két balra zárt oszlop
- *automatikus szélességű oszlopok*
  - `l`: balra zárt oszlop
  - `r`: jobbra zárt oszlop
  - `c`: középre zárt oszlop
  - az oszlopok szélessége a legszélesebb cellatartalomhoz igazodik
  - !! nekünk kell gondoskodni róla, hogy a táblázat kiférjen az oldalra (kivéve `tabular*`)
  - !! ilyen típusú oszlopok celláiban nincs sortörés! (sem automatikus tördelés, sem kézi)

Oszlopok II

- kézi szélességű oszlop
  - `p{3cm}`: szélessége kötelező argumentum!
  - cellán belül automatikusan tördeli a sorokat
  - cellán belül kézi sortörés `\newline` paranccsal
- több azonos oszlop(csoport) deklarálása
  - `*{darabszám}{ismétlődő minta}` (ha egyetlen karakter az argumentum, a `{}` elhagyható)
  - például `*3l` = `lll`
  - vagy `*2{c|}` = `c|c|`
- oszlopelválasztó &-k
  - ha nincs köztük semmi: üres cella
  - ideális esetben oszlopok-1 darab (pl. 3 oszlop közt 2 elválasztó)
  - ha túl kevés, nincs error, üres cellák a sor végén
  - ha túl sok, error

Rácsvonalak, térköz I

- függőleges rácsvonal
  - oszlopdeklarációban | az oszlopok betűi közé
  - elhagyható, vagy duplázható is
- oszlopok közti térköz
  - alapértelmezés: 6pt
  - átállítás `\setlength{\tablcolsep}{6pt}`
  - !! még `\begin{tabular}` előtt, hogy érvénybe lépjen!
- oszlopok betűi közé kerülhet még: `@{kód}`
  - az oszlopok elválasztását *teljesen* újra lehet definiálni, rácsvonalat *és* térközt
  - például `@{\hspace{12pt}\vline\hspace{12pt}}` – függőleges rácsvonal, mindkét oldalán 12 pont térközzel
  - !! a függőleges rácsvonalat itt `\vline` paranccsal jelezzük (próbáljuk ki a hatást | jellel!)
  - vagy például `@{.}` – 0 térköz, az elválasztás egy pont (tizedestörtek írásához, `r@{.}l` oszlopokkal)

Rácsvonalak, térköz II

- teljes hosszúságú vízszintes rácsvonal: `\hline`
  - elhelyezés: két sor *közé*
  - javasolt: sortörés után, `a & b \\ \hline`, aktuális sor *alá* kerül
  - alternatív: sor tartalma elé `\hline a & b \\` – az aktuális sor *fölé* kerül
  - első sor fölé: `\hline` táblázat elejére
  - elhagyható, vagy duplázható is
- részleges vízszintes rácsvonal: `\cline{1-2}`, 1-2 oszlopokban
  - mindenképp tartomány kell! csak az 1. oszlopba: `\cline{1-1}`
  - nem duplázható – nincs error, de nem rajzolja duplán
  - több is használható, ha nem összefüggő vonal kell, pl. `\cline{1-1} \cline{3-3}`
- részleges függőleges rácsvonal: nincs, cellák egyesítésénél látunk majd kerülő megoldást

Táblázatok úsztatása

- `table` környezet – `figure`-höz hasonlóképp, `table`-be ágyazzuk be a `tabular`-t
- `table`-ön belül él a `\caption` parancs
  - szokásos módon felirat + számozás
  - kerülhet a táblázat fölé vagy alá (ha korábban ill. ha később van a kódban mint a `tabular`)
  - szükséges a `\label` elhelyezéséhez és listázáshoz
  - opcionális argumentumban megadható rövidített verzió, az kerül a listába
- listázás `\listoftables` paranccsal
  - `babel`-lel lefordítja a lista nevét, vagy kézzel átnevezhető:

```
\renewcommand{\listtablename}%
{Táblázatlista új neve}
```

Táblázatok testreszabása I

`array` csomaggal extra funkciók:

- sormagasság növelése *általánosan* pl. 2pt-tal: `\setlength{\extrarowheight}{2pt}`
- `array` által definiált oszloptípusok
  - `p{szélesség}` oszloptípushoz hasonlóan
  - adott szélességű, tördelt cellák, függőleges igazítás változik
  - `m{szélesség}` függőlegesen középre igazított bekezdés
  - `b{szélesség}` függőlegesen alulra igazított bekezdés
- kód beszúrása
  - mintha az oszlop minden cellájába a tartalom elé, ill. mögé szúrnánk az adott tartalmat és/vagy parancsokat
  - oszlop elé: `>{kód}c`
  - pl. vastagbetűs `>{\bfseries}c` – formázásból csak az argumentum nélküli parancsok működnek!
  - hamarosan: oszlop színezése is ilyen szintaxissal

Táblázatok testreszabása II

- mögé: `c<{kód}`
- pl. % automatikus beillesztése: `r<{\%}`
- egyazon oszlop elé és mögé is szúrható kód

- saját oszloptípus is definiálható
  - `\newcolumntype{jel}{oszlopspecifikáció}`
  - pl. vastagbetűs oszlop elnevezése másképp: `\newcolumntype{h}{>{\bfseries}c}`
  - cél: átláthatóbb kód
  - a `jel` egy betű legyen, amilyen oszlop még nincs!
  - tartalmazhat elé- és/vagy mögé szúrt kódot, több oszlopos csoportot, rácsvonalat, stb. is!
- oszlopelválasztó módosítása, de a térköz marad: `!{kód}`
  - szintaxis: mint `@{}`
  - cél: pl. függőleges rácsvonalak színezése, bármi más módosítása nélkül

Cellák egyesítése I

- vízszintes egyesítés: `\multicolumn` parancs (alap LaTeX) = többoszlopos cella, például:

```
\begin{tabular}{lll}
\multicolumn{2}{|c|}{1-2} & 3 \\
1 & 2 & 3 \\
\end{tabular}
```

1-2	3
1	2	3

- 2: hány oszlopot foglal el – kevesebb oszlopelválasztó & jelet is kell beírni az adott sorban!
- |c|: az új, összevont cella igazítása, ill. rácsvonalak mellette – felülírja az alapértelmezett függőleges rácsozást
- 1-2: az összevont cella tartalma

Cellák egyesítése II

- függőleges egyesítés: `multirow` csomag `\multirow` parancsa = többsoros cella

!! más a szintaxis!

```
\begin{tabular}{lll}
1 & \multirow{2}{2em}{1-2} & 1 \\
2 &  & 2 \\
\end{tabular}
```

hatás:

1	1-2	1
2		2

- 2: magassága hányszorosa az alap sormagasságnak – lehet törtszám is (!)

!! minden további sorban hagyjuk ki neki az üres cellát, és legyen meg az összes oszlopelválasztó!

- 2em: a cella szélessége
  - * használható automatikus méretezéshez
- 1-2: az összevont cella tartalma

Cellák egyesítése III

- egyesítés függőleges *és* vízszintesen is
  - lehetséges, `\multicolumn` és `\multirow` egymásba ágyazásával
  - !! de csak ha a `\multicolumn` van *kívül*
  - a további sor(ok)ban hagyjuk ki az (összes) üres cellát, mintha `\multirow`-t használnánk!
  - rácsvonalak miatt szükség lehet a kihagyott sor(ok)ba üres `\multicolumn`-(öke)t tenni – akkor ott is kevesebb `&` jel

4 Kiterjesztések

- Szöveg körbefuttatása
- Táblázatok színezése
  - Oszlopok
  - Sorok
  - Cellák és rácsvonalak
- Törhető táblázat – `longtable`
- Float-ok testreszabása és saját float-ok – `float` csomag

Szöveg körbefuttatása – wrapfig I

- szöveg körbefuttatása: keskeny ábra elhelyezése a lap szélére, mellette rövidebb szövegsorok
- `wrapfig` csomag `wrapfigure` és `wraptable` környezetei
- `\begin{wrapfigure}[12]{r}[\marginparwidth]{5cm}`
  `\caption{felirat}`
  `\includegraphics[width=4.5cm]{kep}`
  `\end{wrapfigure}`
- `{r}`: kötelező, elhelyezés
  - `l`,`L` bal vagy `r`,`R` jobb (egyoldalas dokumentum)
  - `i`,`I` belső vagy `o`,`O` külső oldal (kétoldalas dokumentum)
  - kisbetű: „ide”, nagybetű: úszhat (azaz automatikusan áthelyezhető, ha szükséges)
  - sokszor érdemes kisbetűsen hagyni és kézzel áthelyezni a kódban

Szöveg körbefuttatása – wrapfig II

- {5cm}: kötelező, ábra szélessége
  - ha 0pt-t adunk meg, automatikusan méretez
  - de ha van \caption, érdemes kézzel méretezni
- [12]: opcionális, hány sor magas lesz a kép
  - akkor érdemes próbálkozni vele, ha túl sok whitespace-t hagy magától
  - vagy negatív \vspace-eket rakni a kép alá és/vagy fölé
- [\marginparwidth]: opcionális, túllógás a margóra (alapértelmezés 0pt)
- alternatív szintaxis:

```
\begin{wrapfloat}{figure}%
[12]{r}[\marginparwidth]{5cm}
```

Táblázatok színezése

- `\usepackage{colortbl}`, vagy `\usepackage[table]{xcolor}`
- szükséges hozzá `xcolor` és `array` csomag is
- használhatók az `xcolor` színei
  - legegyszerűbb név szerint
  - gyors színkeverés: `\textcolor{red!60!black}{60\% piros, 40\% fekete}` 60% piros, 40% fekete
  - ha a második szín hiányzik, fehéret kever bele: `\colorbox{red!20}{20\%-os piros}` 20%-os piros
  - szín definiálása RGB modellben: pl. `\definecolor{red}{RGB}{255,0,0}`, piros, zöld, kék ebben a sorrendben, 0-255 közti értékek
  - szín definiálása HTML modellben (ugyanez hexadecimálisan), pl. `\definecolor{red}{HTML}{FF0000}`

Oszlopok

- oszlop háttérszíne: `\columncolor{}` – az oszlopspecifikációval kell megadni, oszlop elé illesztendő kódként (ld. array csomag):

```
\begin{tabular}{>{\columncolor{cyan}}l%
>{\columncolor{red}\color{white}}l}
világoskék & piros \\
a &  \\
b \\
\end{tabular}
```

világoskék	piros
a	
b	

  - átláthatóság kedvéért érdemes új oszloptípust definiálni rá!
  - figyeljük meg, mi történik az üresen hagyott és a kihagyott cellával!

Sorok I

- sor háttérszíne: `\rowcolor{}` – sor elejére:

```
\begin{tabular}{ll}
\rowcolor{cyan}
világoskék & a \\
\rowcolor{red}\color{white}
piros & b \\
\end{tabular}
```

világoskék	a
piros	b

  - vegyük észre: szöveg színe sajnos nem alkalmazható egész sorra
    (majd programozós megoldást nézünk rá)
- sor színe felülírja az oszlop színét(!)

Sorok II

- sorok váltakozó színezése: `\rowcolors` parancs – a táblázat *előtt* használandó!
- példa:

```
\rowcolors[\hline]{2}{red!20}{blue!20}
\begin{tabular}{ll}
szám & betű \\
a & 1 \\
b & 2 \\
c & 3 \\
\end{tabular}
```

szám	betű
a	1
b	2
c	3

Sorok III

- szintaxis:
- `[\hline]`: opcionális argumentumba tetszőleges, minden sorban végrehajtandó `parancsok` kerülnek
- `{2}`: hanyadik sortól érvényes a váltakozó színezés
- `\rowcolors*` verzióval a `parancsok` is csak a fenti sortól érvényesek
- `{red!20}`: páratlan sorok háttérszíne
- `{blue!20}`: páros sorok háttérszíne
- üresen is hagyhatók, akkor fehér háttér

!! rendszertől, compiler-től függően vagy működik, vagy nem

Cellák és rácsvonalak I

- egyedi cella háttérszíne: `\cellcolor{}` – bárhová a cellába elhelyezve működik, felülírja az oszlop- és sorspecifikációt
- függőleges rácsvonal helyi színezése pl. `r!{\color{red}\vline}l`
- rácsvonalak „alapértelmezett” színe: `\arrayrulecolor{}`
  - `\arrayrulecolor` használható táblázat előtt vagy közben
  - a *parancs után definiált* rácsvonalakra érvényes – vízszintes rácsvonalakra jól működik
  - a függőleges rácsvonalakat a táblázat elején definiáljuk, ezért a táblázat/vonal közepén nem változtatható meg a színük! (kivéve `\multicolumn`-os trükközést)

Törhető táblázat – longtable I

- ha oldalak közt megtörhető táblázatra van szükségünk, mert olyan hosszú: `longtable` csomag `longtable` környezete
- egyszerre helyettesíti a `tabular`-t és `table`-t!
- önmagában foglalja a feliratozást és számozást, `\caption` használható, *de* sortörés kell utána(!)
- `\kill` parancs: adott sort nem jeleníti meg, de az elrendezésbe, sorszélességekbe beleszámolja
- elhelyezhető benne `\footnote`!
- készíthető vele táblázatfejléc és -lábléc, ami az összes oldalon a táblázat tetejére ill. aljára kerül – avagy, címsorok automatikus ismétlése, akár `\caption`-nel együtt!

Törhető táblázat – longtable II

```
\begin{longtable}{ll}
% első oldal fejléce
\caption{longtable}\\
oszlop1 & oszlop2 \endfirsthead
% többi oldal fejléce
oszlop1 & oszlop2 \endhead
% utolsó sortörés helyett \endhead(!)
% táblázat lábléce - opcionális
\hline \endfoot
%% hasonlóképp utolsó oldal lábléce \endlastfoot-ig
% rendes sorok...
\end{longtable}
```

A float csomag I

- float csomag: létező float-ok testreszabása és új float típusok definiálása
  - miért kellhet(nek) új típus(ok)? mert a float-ok számozása, listázása típusonként történik
- új float definiálása

  \newfloat{program}{hbt}{lop}[section]

  - {program}: float típus neve, tetszőleges
  - {hbt}: alapértelmezett elhelyezés (ld. 3. szakasz)
  - {lop}: segédfájl kiterjesztése, itt .lop = list of programs
    - float listázáshoz *típusonként külön* segédfájl, pl. már létező .lof = list of figures (ábrák), .lot = list of tables (táblázatok)
    - tetszőleges, még nem használt kiterjesztés használható
  - [section]: opcionális, ha megadjuk, section-ön belüli számozást kapcsolja be (lehetne chapter stb.)

A float csomag II

- float megjelenítendő neve (a \caption-ben generált név):

  \floatname{program}{Programkód}

  {program}: float típus, {Programkód}: megjelenítendő név
- listázás:

  \listof{program}{Programkódok listája}

  {program}: listázandó típus, {Programkódok listája}: címsor hozzá
- adott float típus alapértelmezett elhelyezése (beépített típusokra is):

  \floatplacement{figure}{hbt}

  {figure}: típus, {hbt}: új alapértelmezés

A float csomag III

- float stílusok: \floatstyle
  - pl. \floatstyle{plain}: eredeti stílus, de \caption mindig alul
  - plaintop: hasonló, de \caption mindig felül
  - boxed: keretezett float, \caption alatta
  - ruled: \caption felül, vízszintes vonalak körülötte és a float alatt
  - az *utána definiált* float-okra érvényes(!)
  - alkalmazás beépített típusokra: pl. \restylefloat{figure}
- szöveg körbefuttatása
  - együtt tud működni a wrapfig csomaggal (ld. 1 szakasz)
  - a float csomagot töltsük be előbb, és a wrapfig csomagot utána
  - az új float definíciók *mindkét* csomag betöltése után jöjjenek!
  - akkor a program típus mellett automatikusan definiálja a wrapprogram típust is

5 Forráskód, programkód

- Verbatim
- Programkód: `listings` csomag
  - Kód beillesztése
  - Kód formázása
  - Környezet formázása
  - Úsztatás

Verbatim I

- `verbatim`: nyers, feldolgozatlan szöveg, szinte karakterről karakterre* megjelenít, akár LaTeX kódot is(!)
  - * kivétel pl. tabulátor
  - cél: kód beillesztése, vagy whitespace megtartása
  - `typewriter` betűkészlettel formázza
- `\verb||` parancs: inline, vagy „egysoros” verbatim
  - szöveggel egy sorba kerül; tartalmát nem tördeli a LaTeX, és nem lehet benne sortörés!
  - argumentumát *nem* kapcsos zárójelek határolják! a `\verb` parancsot követő *első karakter*, ami tetszőleges, de a következő ugyanolyan karakterig olvas. általában: |
  - pl. kód: `\verb|#&$ \emph{hangsúlyos...}|`
    hatás: `#&$ \emph{hangsúlyos...}`

Verbatim II

- `verbatim` *környezet*: többsoros verbatim
  - `\begin{verbatim}` nyitja, *új sorba tett* `\end{verbatim}` zárja
  - a sortöréseket megtartja!
  - előtte, utána új bekezdés
  - például kód:

```
\begin{verbatim}
\begin{figure}
\includegraphics{kep}
\end{figure}
\end{verbatim}
```

    hatás:

```
\begin{figure}
\includegraphics{kep}
\end{figure}
```

Programkód beillesztése – listings I

- javasolt: `listings` csomag – dokumentáció
  - sokféle programnyelvet ismer és annak megfelelően formáz
  - itt csak ízelítő a formázásból, ld. a dokumentáció 4. szakaszában a reference guide-ot minden lehetséges beállításért
- kód beszúrása:
  - inline, egysoros
    - mint a `\verb||`
    - `\lstinline|kód|`, vagy `\lstinline[opciók]|kód|`
    - tetszőleges* határoló karakterrel, kivéve `[` (nyitó szögletes zárójel)
  - többsoros, `lstlisting` környezet
    - mint a `verbatim` környezet
    - `\begin{lstlisting}`, vagy `\begin[opciók]{lstlisting}`
  - külső fájlból
    - `\lstinputlisting{relatív/elérés/fájl.kit}`, vagy `\lstinpulisting[opciók]{fájl.kit}`

Programkód beillesztése – listings II

- beállítása:
  - általánosan, minden listing-re vonatkozik: csomag betöltéskor `\usepackage[opciók]{listings}`
  - szintén általánosan: `\lstset{opciók}`
    - preambulumban vagy dokumentumtörzsben is!
    - minden egyes opció külön-külön felülírásig érvényes, minden ezt követő listing-re
  - helyileg, a parancs vagy környezet opcionális argumentumában – csak az adott listing-re érvényes változtatások
- opciók alkalmazása adott listing-ekre: stílusokkal
  - stílus definiálása: `\lstdefinestyle{név}{opciók}`
  - használata: `style=név`, akár `\lstset`-tel, vagy listing opcionális argumentumában
  - stílusban bármelyik lentebb látott parancs használható

Programkód formázása I

- programnyelv megadása
  - e szerint ismeri fel és emeli ki a kulcsszavakat/parancsokat
  - `language=...`, pl. `python`, `c`, stb.
  - `\lstinline[language=python]|kód|`
  - `\begin{lstlisting}[language=python]`
  - `\lstinputlisting[language=python]{fájlnév.kit}`
- megjelenített sorok (pl. ha egy hosszabb fájlból csak egy részletet szeretnénk beilleszteni)
  - tartomány(ok): `linerange={5-20,56-82}`
  - csak első és/vagy utolsó beemelt sor megadása (ha csak némi whitespace-t kel lecsípni, vagy egyetlen tartomány kell): `firstline=`, `lastline=`

  !! ezek lokális opciók, stílusban hatástalanok
- szóközök és tabulátorok
  - `tabsize=4`: tabulátor hány szóköz
  - `showspaces`, vagy `showspaces=true`: szóközök mutatása

Programkód formázása II

- kikapcsolás: `showspaces=false`
- hasonlóképp tabulátorra: `showtabs`
- megjelenített tabulátor karakter cseréje: `tab=karakter`, pl. `tab=$\longrightarrow$`: $\longrightarrow$
- sor eleji whitespace elnyelése
  - pl. `gobble=8`: minden kódsor elején első 8 karakter* elnyelése (tabulátorokat `tabsize` szerint szóközökké alakítva!)

- kód elemeinek formázása/betűstílusa
  - alapértelmezés: kulcsszavak/parancsok vastaggal, komment dőlt betűvel
- átállítás lehetséges elemenként:
  - általános érvényű formázás: `basicstyle=stílus`, pl. `basicstyle=\small\ttfamily`, minden kisebb betűkkel és `typewriter` family-vel

Programkód formázása III

- kulcsszavak/parancsok `keywordstyle=\color{blue!50!black}\underbar`, pl. sötétkékkel, aláhúzva
- függvénynevek, stb. `identifierstyle=\color{green!30!black}`, pl. sötétzölddel
- kommentek `commentstyle=\color{gray}`, pl. szürkével
- sztringek `stringstyle=\itshape`, pl. *italic*-kal
- látható szóközök a sztringekben `showstringspaces=true`, kikapcsolás `=false`

- általánosan:
  - mint láttuk, több parancs is használható
  - !! az argumentum nélküli betűstílus és szín parancsokat használjuk!
    - basicstyle kivételével a többiben az *utolsó* parancsnak lehet *egyetlen* argumentuma, pl. a keywordstyle-ban `\underbar{parancs}` <u>parancs</u>

Környezet formázása I

- sorok számozása
  - ki-bekapcsolás és elhelyezés: `numbers=left` balra, vagy `=right` jobbra, `=none` kikapcsolás
  - hány soronként: pl. minden 3. sor számozása `stepnumber=3`
  - milyen számmal kezdi az első sor számozását: pl. `firstnumber=2`
- sorszámozás folytatása
  - folytatás az előző `listing`-ből `firstnumber=last`
  - szériák létrehozása: `name=szérianév` minden összetartozó `listing`-be
  - `firstnumer=auto` opcióval folytonosan számozza a szériát
- háttérszín: pl. halványszürke `backgroundcolor=\color{gray!20}`
  - ha a számozás kilóg a keretezett vagy színezett háttérből, növeljük a margót: pl. `framexleftmargin=20pt`: (hasonlóképp right, top, bottom)

Környezet formázása II

- keretezés `frame=`
  - beépített stílusok:
    - `frame=none` nincs keret (alapértelmezett)
    - `=leftline`, `=topline`, `=bottomline`: egyetlen vonal bal oldalt, felül, ill. alul; `=lines`: felül és alul vonal
    - `=single`: egyszeres keret körbe, `=shadowbox`: „árnyékolt” doboz
  - kézzel: `lrtbLRTB` betűk tetszőleges kombinációja
    - l – left, bal, r – right, jobb, t – top, felül, b – bottom, alul
    - kisbetű: egyszeres, nagybetű: dupla vonal; hiányzó: nincs vonal
  - keret lekerekítése: pl. `frameround=tfft`, óramutató szerint a 4 sarok, t=true = lekerekítve, f=false = szögletes
  - `framerule=.5pt`: keret vastagsága
  - `framesep=5pt`: keret távolsága a tartalomtól

listing úsztatása I

- több módszer is lehetséges az úsztatásra
- float csomaggal új float típus(ok), és abba csomagoljuk
  - például ha több programnyelven íródnak a listing-ek és külön szeretnénk számozni őket
- vagy: a listing csomagoló nélkül is képes magát úsztatni!
  - `\begin{lstlisting}[float,opciók]`
  - float elhelyezése: `float=hbt!` formában; szokásos betűk használhatók
- szintén opcionális argumentumba(!)
  `caption={[rövid]Kód neve}`, `label=` – úsztatás *nélkül* is működnek(!)

listing úsztatása II

- például

```
\begin{lstlisting}[float=hb!,%
caption={[Buborék]A buborék rendezési algoritmus},%
label=lst:buborek]
```

- *számozás nélküli* felirat: `title=` (a `caption=` helyett)
- listázás: `\lstlistoflistings` (betűzésre figyeljünk!)
  - egy adott kód elrejtése a listából: `nolol` az opcionális argumentumba
- babel nem fordítja le a lista címsorát és a `listing` típusát a `caption`ben – kézzel újradefiniálhatók:
  - típus: `\renewcommand{\lstlistingname}{Programkód}`
  - lista címe:

    ```
    \renewcommand{\lstlistlistingname}%
    {Programkódok listája}
    ```

    (betűzésre figyeljünk!)