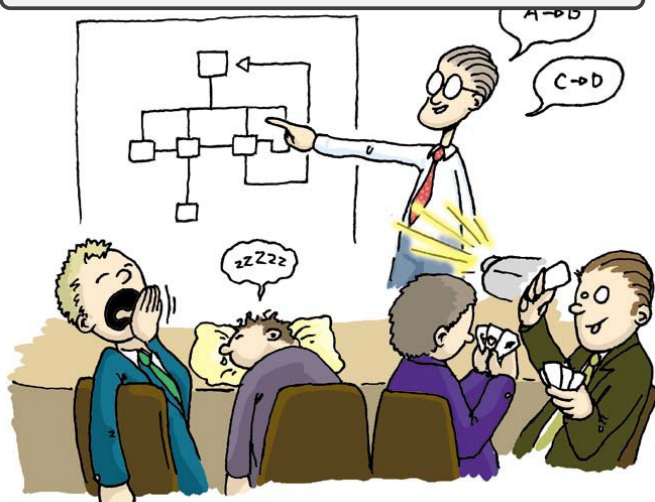


Végh János

Hogyan használjuk  
a MultEdu csomagot  
(Hogyan készítsünk érdekes  
és vonzó tananyagot)





Ez a Mű a

"<http://creativecommons.org/licenses/by/4.0/>

Creative Commons Nevezd meg! 4.0 Nemzetközi Licenc feltételeinek megfelelően felhasználható.

©Szerzői jogok 2011-2017 Végh János

(Janos.Vegh@unideb.hu)

Minden jog fenntartva

Kizárólag újra hasznosított elektronokkal nyomtatva

## Kivonat

Kurzusaim oktatásához saját makrókészletet fejlesztettem, mivel az oktatandó anyag megjelenítéséhez különböző körülmények között különböző formákra van szükség. Az elméleti anyagot az előadásokon diasorozat alapján mutatom be, és a diákhoz fűzött magyarázatokat (természetesen tömörítve) jegyzet-szerű formában a hallgatóság számára is rendelkezésre bocsátom. A hallgatóság ezt az anyagot részben kinyomtatva, részben képernyőn olvasva (akár mobil eszközökön is) tanulmányozza. A terület folyamatos fejlődése miatt a tananyag is állandó fejlesztésre szorul, ezért feltétlenül szükséges, hogy az említett megjelenési formákat egy-

mással szinkronban lehessen fejleszteni. Ennek legegyszerűbb megvalósítási formája, hogy egyazon forrásból, megfelelő formattálási utasításokkal készítem a tananyagokat. Számítógéppel alaposan megtámogatott, nagy felbontáshoz és vonzó grafikához szokott hallgatóság számára a fenti feltételeknek megfelelő tananyagot készíteni komoly kihívás.

Közös alapként a LaTeX nyelvet használtam, amely nyelven készült forrásból az elterjedten használt Beamer prezentáció készítő makró csomaggal állítom elő az előadáson bemutató diákat, és a memoir könyv készítő makró csomaggal a hallgatóság számára rendelkezésre bocsátandó "tananyagot". Ez utóbbi akár az "on demand printing" minőséget is elérheti. Vonzó grafikus megjelenéssel, a szokásoshoz képest sokkal több ábrával igyekszik felkelteni az anyag a hallgatóság figyelmét (de lehet belőle kevésbé "fancy", inkább "plain" stílusú, de még mindig könyv minőségű változatot is készíteni). A jegyzet-szerű változat az előadáson bemutatott ábrákat és szöveget teljes egészében tartalmazza, az előadás szövegének egy tömörített változatával kiegészítve. Ugyanez a könyv-szerű anyag jelenik meg a képernyős olvasásra szánt WEB-es formátumban, illetve az eBook kompatibilis (natív PDF) változatban. Ebben a változatban az anyag egy-képernyőnyi darabokra van tördelve, és (főként kisebb képernyőjű mobil eszközökre gondolva) nagyobb betűkkel, egy ábra/képernyő módon jelenik meg.

A többféle, egymásnak ellentmondó megjelenítési igény

természetesen csak kompromisszumokkal oldható meg, így a tananyag megírása során a szöveg megjelenés formázására több gondot és időt kell fordítani. A makrócsomaggal akár egyidejűleg idegen nyelvre is lehet ugyanazon tananyagot fejleszteni. A LaTeX lehetőségeivel akár animáció, mozgófilm, WEB-lap, hang, stb. is beépíthető, természetesen gondolni kell a nyomtatott anyag ekvivalens megjelenítésére.

Első kiadás: 2016 Szeptember

V0.5.5 @2016.08.19

(<https://github.com/jvegh/MultEdu> V0.5.4)

10 09 08 07 06 05 04 03 02 01      19 18 17 16 15 14 13



---

# Tartalomjegyzék

<b>Tartalomjegyzék</b>	<b>i</b>
<b>1 Általános</b>	<b>1</b>
1.1. Bevezetés . . . . .	2
1.2. Beüzemelés . . . . .	5
1.3. Szerkezet . . . . .	6
1.3.1. common . . . . .	7
1.3.2. Workstuff . . . . .	8
1.3.3. Generált fájlok . . . . .	10
1.4. A csomag . . . . .	12
<b>2 Tagolás</b>	<b>14</b>

2.1. Egységek . . . . .	15
2.1.1. Dia keretek . . . . .	15
2.1.2. Fejezet . . . . .	15
2.1.3. Szakasz és az alatt . . . . .	16
2.2. Kétnyelvű forráskódok . . . . .	17
2.2.1. Átváltás a nyelvek között . . . . .	17
2.2.2. Dia keretek . . . . .	18
2.2.3. Fejezet . . . . .	19
2.3. Fejezet illusztráció . . . . .	21
2.4. Nyomtatott és vetített szöveg . . . . .	22
2.5. Lebegő objektumok . . . . .	23
<b>3 Program listák</b>	<b>24</b>
3.1. Megjelenítés . . . . .	25
3.2. Töredék kód . . . . .	26
3.3. Teljes program . . . . .	27
3.4. Díszítések . . . . .	30
3.4.1. Kijelölés . . . . .	30
3.4.2. Megjegyzések . . . . .	31
3.4.3. Megjegyzés . . . . .	32
3.4.4. Golyók . . . . .	33
3.4.5. Ábrák . . . . .	36
3.5. Egyéb . . . . .	38

3.5.1.	Forrás fájlok összehasonlítása .	38
3.5.2.	Eredményel . . . . .	38
3.6.	Program nyelvek . . . . .	42
<b>4</b>	<b>Ábrák</b>	<b>43</b>
4.1.	Hagyományos . . . . .	44
<b>5</b>	<b>Finomhangolás</b>	<b>45</b>
5.1.	Alap beállítások . . . . .	46
5.2.	Beállítások . . . . .	47
5.2.1.	Beamer . . . . .	47
5.3.	Fájlok . . . . .	48
5.3.1.	Alapértelmezett . . . . .	48
<b>6</b>	<b>Fordítás</b>	<b>51</b>
6.1.	Kézi . . . . .	52
6.2.	Kötegetl . . . . .	54
6.3.	Beállítások . . . . .	56
6.3.1.	Verziók . . . . .	56
6.3.2.	Nyelvek . . . . .	57
<b>7</b>	<b>Kiegészítések</b>	<b>59</b>
7.1.	Rövidítések . . . . .	60
7.1.1.	Használatuk . . . . .	61

<i>TARTALOMJEGYZÉK</i>	<i>iv</i>
7.1.2. Meghatározásuk . . . . .	62
7.1.3. Használatuk . . . . .	63
7.2. Indexek . . . . .	64
7.3. Irodalom jegyzék . . . . .	65
<b>Tárgymutató</b>	<b>66</b>
<b>Acronyms</b>	<b>68</b>
<b>Glossary</b>	<b>69</b>
<b>Ábrák jegyzéke</b>	<b>71</b>
<b>Listings</b>	<b>72</b>



---

# Általános információ

## 1.1. Bevezetés

Kurzusaim tartásához saját tananyagot fejlesztettem, különböző megjelenési formákban; a jelen csomag ennek mellékterméke. A jó kurzusok tananyaga gyorsan fejlődik, különösen akkor, ha maga a tudományág is naponta megújul. Az informatikában évről évre változik a technológia, a statisztikák, a termékek, a segédeszközök, stb.; és már csak emiatt is minden tanévre frissíteni kell a tananyagot.

Manapság a tananyagot a hallgatóság változatos formákban igényli: előadáson nagy méretű, jól áttekinthető, kivetíthető anyagot kell használni, amely képekkel gazdagon illusztrált és az előadó számára is jó sorvezetőként szolgál. A vizsgára készüléshez pedig arra a magyarázatra is szükség van, amit az előadó a kivetített anyaghoz élő szóban hozzáfűz. Azaz, olyan magyarázó szöveggel ellátott anyagra is szükség van, amelyet kinyomtatva, asztali gép vagy mobil eszköz képernyőjén lehet elolvasni. Esetleg ugyanazt a változatot idegen nyelven is közzétenni, külföldi hallgatók számára. Bár sokszor lehet elérhető könyvekre és megvásárolható jegye-

tekre hagyatkozni, a kicsit is speciálisabb anyagok esetén ez a segédlet lesz a felkészüléshez szükséges tananyag.

A jelen makró csomag olyan, amelyet saját kurzusaim készítéséhez fejlesztettem, és igyekeztem olyanná tenni, hogy tananyag fejlesztés közben már ne kelljen a megjelenítés technikájával foglalkozni, és ilyen módon mások is tudják hasonló célra felhasználni, ha követik a fejlesztés logikáját. A csomag bizonyos vonatkozásokban egészen jó, néhol tudatosan kompromisszumot kellett kötni a sokféle igény között, néhol még nem tökéletes, és sok vonatkozásban még nem jutott időm további tulajdonságok fejlesztésére.

A jelen dokumentum egyúttal bemutató és tulajdonság tesztelő is. A dokumentum megkísérli bemutatni, mit hogyan kell és lehet használni, egyúttal azt is megvizsgálva, hogy tényleg működik-e az elvárt módon. A sokféle tulajdonság és különösen azok kölcsönhatása miatt sok munkát és időt igényel a fejlesztés, ezért a tényleges tulajdonságok nem mindig egyeznek meg a dokumentációval, különösen

a kezdeti fázisban.

A makró csomag (legalább) három különböző felhasználói szinten alkalmazható. Már a legalacsonyabb szinten is szükségesek a L<sup>A</sup>T<sub>E</sub>X-re vonatkozó elemi ismeretek. Az alap szinten a felhasználó egyszerűen csak helyettesíti és módosítja a rendszert bemutató dokumentumokat. Haladó szinten (ehhez már el kell olvasni a felhasználói leírást is ☺) megtanulja a csomagban található makrók által biztosított lehetőségeket, és azokat aktívan használva fejleszti dokumentumait. Tapasztalt felhasználóként saját makrókat is készíthet (jó, ha azokat a letölthető anyaghoz hozzáadja), azaz aktívan részt vesz a fejlesztésben.

## 1.2. A MultEdu beüzemelése és használata

A MultEdu (mint minden L<sup>A</sup>T<sub>E</sub>X alapú rendszer) feltételezi, hogy a felhasználó már rendelkezik tapasztalatokkal a L<sup>A</sup>T<sub>E</sub>X használatában. Azaz, a felhasználó rendszerén már működnie kell valamilyen L<sup>A</sup>T<sub>E</sub>X rendszernek.

Az egyszerű használat és a gyors elindulás érdekében célszerű a lentebb megadott módon saját projekt csoportjainak egy főkönyvtárat és azon belül az egyes projekteknek alkönyvtárakat létrehozni. A leggyorsabb magát a `./Workstuff` könyvtárat (a megfelelő átnevezésekkel és törlésekkel) lemásolni, és csekély módosításokkal elkészíteni saját 'Helló Világ' programját. Ezután érdemes legalább átlapozni a felhasználói kézikönyvet, ami után már elkezdheti saját fejlesztését. Eleinte csak szöveget, aztán sorjában megtanulni a használni kívánt tulajdonságok programozását. Ne feledje: a L<sup>A</sup>T<sub>E</sub>X nehéz nyelv, pontos kódolást igényel, és ezért ilyen a MultEdu is. A gyakori mentések és a verziókövető rendszerek használata nagy segítséget jelentenek.

### 1.3. A MultEdu könyvtár szerkezete

A MultEdu rendszert az alábbi könyvtár szerkezetben célszerű használni. Két fő könyvtára: a `./common`, amely tartalmazza a MultEdu összes fájlját, és a `./Workstuff`, amely a felhasználói könyvtár szerkezetet modellezi.

```
.  
|-- common  
|-- WorkStuff
```

A felhasználói projekt csoportokat ilyen szerkezetben érdemes hozzáadni:

```
.  
|-- Exams  
|-- Labs  
|-- Lectures  
|-- Papers
```

amely könyvtáraknak a `-- WorkStuff` könyvtárhoz hasonló belső alkönyvtárai vannak

### 1.3.1. A common alkönyvtár

A `./common` különleges célú al-alkönyvtárakat, valamint általános célú makró fájlokat tartalmaz. A **MultEdu** megpróbál a lehető legbarátságosabb lenni: alapértelmezett beállításokat, fájlokat, képeket, stb használ, hogy gyorsan el lehessen kezdeni egy új fejlesztést.

```
.  
|-- common  
| |-- defaults  
| |-- formats  
| |-- images
```

A `./defaults` al-alkönyvtár olyan alapértelmezett szöveget tárol, mint a szerzői jogok. Alapértelmezetten, ha a felhasználó nem adja meg saját dokumentum elemeit, a **MultEdu** automatikusan az alapértelmezetteket használja helyettük (feltéve, hogy azok használata nincs megtiltva, lásd később).

A `./formats` al-alkönyvtár tartalmazza a formátumokat meghatározó makrókat; itt adhatja hozzá a felhasználó esetleges saját formátum leíró makróit.

Az `./images` al-alkönyvtár képeket tartalmaz, ame-

lyek egy része alapértelmezett képként használható.

### 1.3.2. A Workstuff *alkönyvtár*

A `./Workstuff` al-alkönyvtár tartalmazza (a példa programként is szolgáló) felhasználói leírás fájljait. Egy olyan `./Workstuff/Demo` projektet tartalmaz, amelyik (a saját főkönyvtárában) három fájlból áll.

```
-- WorkStuff
| |-- Demo
| . |-- CMakeLists.txt
| . |-- Demo.tex
| . |-- Main.tex
```

A valódi főprogram `Main.tex`, és ehhez készült egy `Demo.tex` nagyon egyszerű boríték. Ha használja a UseLATEX csomagot, a `Main.tex` file használata (ezzel a névvel) kötelező, a boríték fájl nevét pedig a `CMakeLists.txt` fájllal egyeztetni kell.

A `./Workstuff` al-alkönyvtárai különböző célokat szolgálnak. Célszerű a felhasználói projekt könyvtárakat is hasonlóan berendezni.

```
-- WorkStuff
```



```
| |-- Demo
| . |-- build
| . . . |-- build
| . |-- dat
| . |-- fig
| . |-- lst
| . |-- src
```

A fő `Main.tex` menet közben magába olvassa az alkönyvtárakban levő egyéb fájlokat.

| . |-- `src` tartalmazza a felhasználó forráskód fájljait,

| . |-- `fig` a képeit,

| . |-- `lst` a programlisták forrás kódját,

| . |-- `dat` a többi adatot (például táblázatok, adatok a pgfplot vagy kód a TikZ ábrák számára).

További alkönyvtárak is készíthetők, de azokat a felhasználónak kell kezelni, és módosítania kell a `CMakeLists.txt` fájlt is.

A CMake rendszeren keresztül a UseLATEX csomag is használható arra, hogy egy szerkesztés után, a köteget feldolgozási módot használva, egyetlen lépésben elő lehessen állítani a forrásnyelvi fájlból a

különböző nyelvű és formátumú dokumentumokat; erre való a `CMakeLists.txt` fájl.

A

```
|-- build és  
| . . |-- build
```

alkönyvtárak csak akkor kellenek ha a CMake rendszert használjuk; ezek a feldolgozás során szükséges átmeneti fájlokat tartalmazzák. A rendszer készít a projekt könyvtárába (ami a `Demo` alkönyvtárnak felel meg) egy saját másolatot a `common` alkönyvtárról. Ezek a fájlok bármikor törölhetők: amikor fordít, a CMake újra generálja azokat.

### 1.3.3. Generált fájlok

A fordítás során a  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  számos munka fájlt állít elő. Ezek sajnos a projekt gyöker könyvtárába kerülnek. Amint az 1.3.2 szakaszban látható, a működéshez csak 3 fájl szükséges, a többi bármikor törölhető.

A köteget feldolgozás is készít a projekt gyöker könyvtárába `.tex` forrás fájlokat. Ezek is bármikor törölhetők, de akár 'kézi' fordítással kimenő fájlt

is készíthetünk belőlük. Ez utóbbi esetben érdemes előtte az `src/Defines.tex` fájlt átszerkeszteni.

## 1.4. A MultEdu csomagról

A MultEdu csomag teljes forráskódot tartalmaz (szépítgetés nélkül). A szerző nem L<sup>A</sup>T<sub>E</sub>X szakértő, csak régi felhasználó. A makrók nagy része adaptált az Interneten megtalálható eredeti forrásokból. A forráskód tartalmazza a hivatkozást az eredeti kódra, a felhasználói kézikönyv nem veszteget helyet köszönetnyilvánításra. A szerző azonban köszönetét fejezi ki az eredeti szerzőknek, mint az eredeti kódért, mind a különböző felhasználói közösségekben nyújtott támogatásért.

A csomag tartalmaz pár .pdf fájlt, különböző formátumban és nyelven. A fájl nevében nem szerepel a verzió szám (a címlapon igen). Eme fájlok célja (amellett, hogy felhasználói kézikönyvként is szolgálnak), hogy a leendő felhasználók gyorsan fel tudják mérni, ilyen tulajdonságokkal rendelkező dokumentáló rendszert akarnak-e.

A MultEdu makró csomagot úgy tettem közzé, ahogy van ('as is'). Folyamatosan és egyenetlenül fejlesztem, én magam már jól tudok vele tananyagot fejleszteni. A makrókat és a dokumentációt

is fejleszttem, de az (sok) időt igényel. Működési és dokumentációs hibák leírását, még esetleges tulajdonságok fejlesztésének kérését is örömmel fogadom.

---

# A dokumentum tagolása

## 2.1. Dokumentum egységek

A dokumentumot a 'beamer' csomag követelményeinek megfelelően kell szervezni. A nyomtatható formában való megjelenítéshez a **MultEdu** a 'beamerarticle' csomagot használja, és a tagolást is megfelelően változtatni kell. Ennek érdekében a **MultEdu** saját tagolási egységeket vezet be, amelyek valójában a 'book' formátumnak felelnek meg és amelyeket dia készítéshez megfelelően átalakít. A szöveg viszont 'diakeret' egységekből áll össze.

### 2.1.1. Dia keretek

Használata:

```
\MEframe[keys]{subtitle}{content}
```

Értelmezett kulcsok

**shrink=true|false** and **plain=true|false**

Alapértelmezetten mindkettő false.

### 2.1.2. Fejezet

A dokumentum legnagyobb egysége a fejezet.

Használata:

```
\MEchapter[short title]{long title}
```

Ha diákat készítünk, `\section` lesz belőle.

### 2.1.3. Szakasz és az alatt

A következő, kisebb egység a szakasz. Használata:

```
\MSection[r~Ovid cím]{hosszú cím}
```

Hasonló módon létezik

```
\Msubsection [r~Ovid cím] {hosszú cím}
```

és

```
\Msubsubsection [r~Ovid cím] {hosszú cím};
```

ez utóbbi dia készítés esetén `\paragraph` alakot ölt.



## 2.2. Kétnyelvű forráskódok

Előfordul, hogy ugyanazt az anyagot saját nyelvenem oktatom hazai hallgatóknak, és angolul, külföldi hallgatóknak. A tananyag megegyezik, és együtt kell fejleszteni. Nyilván előnyös, ha a két anyag ugyanabban a forrásnyelvi fájlban, egymás mellett fejleszthető.

Erre szolgál a `\UseSecondLanguage`. A fent bevezetett makróknak van egy 'D' (Dual) taggal kibővített változata, amelyikben mind az elsődleges, mind a másodlagos nyelven megadjuk a szükséges tartalmakat.

### 2.2.1. Átváltás a nyelvek között

Használata:

```
\UseSecondLanguage{YES}
```

ahol az {}-ben megjelenő argumentum nem számít, csak az, hogy definiálva van-e ez a makro.

A kétféle makrókészlet keverhető, de csak a 'D' makrók reagálnak a nyelv változtatásra.

### 2.2.2. Dia keretek

Kétnyelvű dokumentumokban általában a

```
\MEDframe[keys]{subtitle, first language}
{content, first language } {subtitle,
second language} {content, second
language}
```

keretet használjuk. Azaz a felhasználó megadja mindkét nyelven a címet és a tartalmat, majd fordítás előtt `\UseSecondLanguage` használatával kiválasztja az egyik nyelvet.

Bár tananyag készítésekor kevésbé fontos szempont, egy konferencia előadás bemutatásakor nagyon fontos az előadásra szánt idő megfelelő felhasználása. A **MultEdu** a kivetített diákon a felhasznált idő kivetítésével tudja ezt támogatni. Ez a lehetőség alapállapotban tiltott, külön engedélyezni kell `\def\EnableTimer{YES}` utasítással, célszerűen a `src/Defines.tex` fájlban. Ezt az utasítást célszerű az első "valódi" keret címében elhelyezni, különben üres keretet eredményezhet. Példa:

```
\MEframe{Keret cim
```

```
\ifx\EnableTimer\undefined \else
\initclock\fi}
```

A **MultEdu** a kijelzett idő színének megváltoztatásával figyelmezteti az előadót, ha az előadás végéhez közel kerül. A maximális értéket

```
\def\LectureTime{perc}
```

utasítással lehet beállítani, az alapértelmezett érték 15. Az időmérés a második dia megjelenytésekor indul, az idő újra indul, ha oda visszamegyünk.

### 2.2.3. Fejezet

Hasonlóképpen, a kétnyelvű dokumentum legnagyobb egysége a 'Dchapter'. (Amint említettük, dia készítéskor ez átalakul 'Dsection' egységgé.)

Használata:

```
\MEDchapter[r~0vid cím1]{hosszú
cím1}{r~0vid cím2}{hosszú cím2}
```

ami aztán átalakul

```
\MEchapter[r~0vid cím1]{hosszú cím1} vagy
\MEchapter[r~0vid cím2]{hosszú cím2}
```

attól függően, hogy `\UseSecondLanguage` definiált vagy sem.

Teljesen hasonló a kisebb formázási egységek használata is.

## 2.3. Fejezet illusztráció

Néhány könyv stílus lehetővé teszi, hogy a fejezetek elején egy illusztrációt helyezzünk el.

Használata:

`\MEchapterillustration{file}`

Dia készítéskor, a kép egy 'plain' dián jelenik meg. Nyomtatható változatban a fejezet elején jelenik meg a kép.

Ha a fájl név üres, a csomag a `fig/DefaultIllustration.png` képet keresi. Ha a fájl nem található, nem készül illusztráció.

Ha definiáljuk a `\DisableChapterIllustration` makrót, a csomag nem generál képet.

## 2.4. Nyomtatott és vetített szöveg összehangolása

A nyomtatott anyag jelentősen több szöveget szokott tartalmazni, mint a diák. Ezt az extra szöveget úgy lehet a forrás fájlban elhelyezni, hogy az `\ao{text}` (article only) makró belsejében adjuk meg az extra szöveget. Az így megadott szöveg csak a nyomtatott változatban látható, a diákon nem jelenik meg. Vigyázzunk rá, hogy a szöveg mindkét változatban értelmes legyen, különösen, ha mondat belsejében használjuk.



---

## Program listák készítése

Programozás tanításakor alapvető követelmény programlisták megjelenítése. A 'listings' csomag felhasználásával a **MultEdu** ezt jó minőségben tudja biztosítani. Az itt nem ismertetett részletekért lásd a 'listings' csomag leírását.

Ebben a szakaszban szokatlanul sok elhelyezendő programlista van, ami nagyon megnehezíti a fordító-program dolgát. Valódi szövegek esetén az készített oldal sokkal esztétikusabb.



### 3.1. A megjelenítés beállítása

A 'listings' csomag számos lehetőséget biztosít arra, hogy a programlista megjelenítés stílusát ízlésünknek (és a követelményeknek) megfelelően állítsuk be. A **MultEdu** beállít valamilyen alapstílust, amit tetszés szerinti alkalommal és módon módosíthatunk.

`\MSESetStandardListingFormat`

beállít egy alap-megjelenítést, de nem állít be programnyelvet.

`\MSESetListingFormat[options]{language}`

beállítja a nyelvet, és

`\MSESetStandardListingFormat`

szerinti alap-megjelenítést és 'options' használatával lehetővé teszi a 'listings' alapértelmezett argumentumainak felülírását.

## 3.2. Sorközi töredék megjelenítése

Gyakori feladat egy rövidebb töredék, mint egyetlen sor vagy akár kulcsszó/változó megjelenítése. Ezt a `\lstinline|code|` módon tehetjük meg.

Az ebben a leírásban is kiterjedten használt LaTeX parancsok megjelenítéséhez a fejezet elején használnunk egy

```
\MSESetListingFormat{TeX}
```

```
\lstset{basicstyle=  
\ttfamily\color{black}\normalsize}
```

vagy

```
\MSESetListingFormat[basicstyle=  
\ttfamily\color{black}\normalsize]{TeX}
```

parancsot. (különben túl kicsi lesz a megjelenített program kód karaktereinek mérete)

### 3.3. Teljes programlista megjelenítése

A

```
\MESourceFile[keys] {filename} {caption}  
{label}{scale}
```

makróval jeleníthetők meg programlisták. Lehetőséges kulcs: `wide[=false]`, `decorations[={}]`. A 3.1 programlista megjelenítéshez használt programsor:

```
\MESourceFile[language={ [ISO]C++}]  
{lst/HelloWorld.cpp} {A "Hello World"-  
C++ program} {lst:hello.cpp}{}
```

Sokszor van szükség szélesebb programlista megjelenítésére. Ennek hatására a két oszlopos nyomtatás teljes szélességében jelenik meg a lista. Egyoszlopos megjelenítés esetén a keskeny lista az oldalszélesség 70%-ára terjed ki, a széles pedig a teljes oldal szélességet igénybe veszi. A széles programlistákat még nehezebb elhelyezni az oldalon (a megjelenítő utasítás helye utáni oldal tetejére kerülhet legelőször), ráadásul nem is szabad felcserélni a normál és széles programlisták sorrendjét. Emiatt

Listing 3.1. A "Hello World"- C++ program

```
#include <iostream>
using namespace std;
int
main ( int argc, char ** argv )
{
    // print welcome message
    cout << "Hello World" << endl;
    return 0;
}
```

a megjelenési hely eléggé messze is kerülhet a hivatkozás helyétől.

A 3.2 programlista megjelenítéséhez használt programsor:

```
\MESourceFile[language={ [ISO]C++},wide]
{1st/HelloWorld.cpp} {A "Hello World"-
C++ program, wide} {1st:Whello.cpp}{} }
```

Listing 3.2. A "Hello World"- C++ program, wide

---

```
#include <iostream>
using namespace std;
int
main ( int argc, char ** argv )
{
    // print welcome message
    cout << "Hello World" << endl;
    return 0;
}
```

---

### 3.4. Programlista díszítései

A programlistán különféle díszítményeket helyezhetünk el. Ehhez a programlista készítésekor használnunk kell a **decorations** kulcsszót és annak argumentumaként az e szakaszban bemutatott makrókat kell megadni.

Az általános forma:

```
\MESourceFile[options, decorations={ list  
of decorations } ] {source file}  
{caption} {label}{}{}
```

ahol a díszítések listája a szakaszban felsorolt bármelyik fajta díszítést tartalmazhatja.<sup>1</sup> Az **options** argumentumaként a 'listings' csomagban használt bármely opció használható.

#### 3.4.1. Programsorok kijelölése

A 3.3 programlistán a programtörzs utasításainak kijelöléséhez a

---

<sup>1</sup> A program az első menetben elhelyezi a programlistát és egy újabb fordítás során tudja a díszítéseket is felrakni.

```
\MESourceFile[language={ [ISO]C++},  
decorations={ \MESourcelinesHighligh  
{HelloBalloon} {lst:HLhello.cpp} {6}{8} }  
] {lst/HelloWorld.cpp} {"Hello World" --  
a C++ way, kijel~0lt} {lst:HLhello.cpp}{}
```

parancsot kell kiadni.

Listing 3.3. "Hello World" – a C++ way, kijelölt

```
#include <iostream>  
using namespace std;  
int  
main ( int argc, char ** argv )  
{  
    // print welcome message  
    cout << "Hello World" << endl;  
    return 0;  
}
```

### 3.4.2. Megjegyzés kijelölt programsorokhoz

Az előbbi programlistán a kijelöléshez megjegyzést is fűzhetünk. Ennek formája

```
\MEBalloonComment[keys]{BallonName}  
{ShiftPosition} {Comment} {CommentShape}
```

amivel az előzőleg felrajzolt ballonhoz fűzhetünk megjegyzést. Itt

`BallonName` az `\MEHighlightLines` első argumentuma, `ShiftPosition` a megjegyzésdoboz eltolása, `Comment` pedig maga a megjegyzés. A lehetséges opciók: `width[=3cm]` és `color[=deepeach]`.

A 3.4 programlista készítéséhez a

```
\MESourceFile[language={ [ISO]C++},wide,
decorations={ \MESourcelinesHighlight
{HelloBalloon} {1st:HLHello.cpp} {6}{8}
\MESourceBalloonComment {HelloCBalloon}
{0cm,0cm} {This is the body}
{CommentShape} } ] {1st/HelloWorld.cpp}
{"Hello World" -- egy C++ program }
{1st:HLhello.cpp}{}
```

parancsot kell kiadni.

### 3.4.3. Megjegyzés forráskód programsorhoz

Az egyes forráskód sorokhoz is fűzhetünk megjegyzéseket, lásd 3.5 programlista. Ehhez a

```
\MESourceFile[language={ [ISO]C++},
decorations={
```



## Listing 3.4. "Hello World" – egy C++ program

```
#include <iostream>
using namespace std;
int
main ( int argc, char ** argv )
{
    // print welcome message
    cout << "Hello World" << endl;
    return 0;
}
```

Ez a csoport

\MESourcelineComment{lst:Chello.cpp} {6}  
{-1cm,0cm} {This is a comment}  
{CommentShape} } ]{lst/HelloWorld.cpp}  
{"Hello World" -- a C++ way, commenting  
source lines} {lst:Chello.cpp}{} utasítást  
kellett kiadni.

### 3.4.4. Hivatkozási pontok elhelyezése a programlistán

Az előbbi programlistán különböző programsorokat is megjelölhetünk. Ennek formája

Listing 3.5. "Hello World" – a C++ way, megjegyzés a forráskód sorához

```
#include <iostream>
using namespace std;
int
main ( int argc, char ** argv )
{
    // print welcome message ← Ez megjegyzés
    cout << "Hello World" << endl;
    return 0;
}
```

\MESourceListBalls[keys]{ListingLabel}{List of lines}

amivel a megjelölt programsorok végére kerül egy-egy számozott golyó. Itt `ListingLabel` a programlista címkéje, `List of lines` pedig azon sorszámok listája, ahová golyót szeretnénk elhelyezni. Lehetséges kulcsok, az alapértelmezett értékkel:


`color[=orange]` and `number[=1]`.

Megjegyzések:

- Dia készítéskor az egyes golyók a egy dia

sorozatra kerülnek

- A golyók elhelyezése csak geometria pozíció alapján történik, nem veszi figyelembe a 'firstline' paramétert.
- a golyók számozása a `number`[=1] értéktől indul.

Az így megjelölt sorokra később így hivatkozhatunk: "( Listing 3.6 

`\MEBall{lst:LBhello.cpp}{2}` makrót kell használnunk.

A 3.6 lista készítéséhez a

```
\MESourceFile[language={ [ISO] C++},  
decorations={  
\MESourcelineListBalls{lst:LBhello.cpp}{3,8,  
} ] {lst/HelloWorld.cpp} {"Hello World"  
-- a C++ way, golyokkal}  
{lst:LBhello.cpp}{}
```

parancsot kell kiadni.

Listing 3.6. "Hello World" – a C++ way, golyókkal

```

#include <iostream>
using namespace std;
int ①
main ( int argc, char ** argv )
{ ③
    // print welcome message
    cout << "Hello World" << endl;
    return 0; ②
}

```

### 3.4.5. Ábra elhelyezése a programlistán

Néha ábrát is akarhatunk elhelyezni a programlistán. Az ezt a célt szolgáló makró

```

\MESourcelineFigure[keys] {SourceLabel}
{LineNo} {ShiftPosition} {GraphicsFile}.

```

Lehetséges kulcs: `width [=3cm]`

A 3.7 programlista előállításához használt makró:

```

\MESourceFile[language={Verilog},wide,
decorations={
\MESourcelineFigure[width=5.2cm]
{lst:forloops.v}{8} {3.0,-.3}
{fig/forloops} } ] {lst/forloops.v}

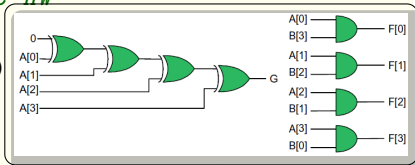
```

```
{Implementing \lstinline|for| loop with  
repeating HW} {\lst:forloops.v}{}
```

Listing 3.7. 'for' ciklus megvalósítása HW ismétléssel

```
// for == repeat HW
```

```
always @(A or B)  
begin  
    G = 0;  
    for (I = 0; I < 4; I = I + 1)  
    begin  
        F[I] = A[I] & B[3-I];  
        G = G ^ A[I];  
    end  
end
```



## 3.5. Kapcsolódó egyéb makrók

### 3.5.1. Forrás fájlok összehasonlítása

Néha érdemes forrás kód fájlokat egymás mellé helyezve összehasonlítani. Az erre szolgáló makró

```
\MESourceFileCompare[keys]{source file1}  
{source file2} {caption} {label}
```

A 3.9 programlista előállításához használt utasítás

```
\MESourceFileCompare[language={ [ANSI]C}]  
{lst/lower1.c} {lst/lower2.c} {Comparing  
two routines for converting string to  
lower case} {lst:lower12.c}
```

A makró a forrásfájlt nem kezeli; az ábrán a jobb összehasonlítás kedvéért beiktatott üres sorokat kézzel kellett beírni.

### 3.5.2. Forrás eredménnyel

A 3.10 program lista a

Listing 3.8. A két kisbetűssé alakító rutin összehasonlítása

```
/* Convert string to lowercase:
   slow */
void lower1(char *s)
{
    int i;

    for (i = 0; i < strlen(s); i++)
        if (s[i] >= 'A' && s[i] <= 'Z')
            s[i] -= ('A' - 'a');
}
```

Listing 3.9. A két kisbetűssé alakító rutin összehasonlítása

```
/* Convert string to lowercase:
   faster */
void lower2(char *s)
{
    int i;
    int len = strlen(s);
    for (i = 0; i < len; i++)

        if (s[i] >= 'A' && s[i] <= 'Z')
            s[i] -= ('A' - 'a');
}
```

```
\MESourceFileWithResult[language=C++,wide,
decorations={ \MESourcelineListBalls
{lst:calculatorwithresult}
{13,14,16,18,19} }]
{lst/expense_calculator.cpp}
{lst/calculatorresult.txt} {The
calculator program with its result}
{lst:calculatorwithresult}
```

utasítás eredménye.

Néha hasznos egy forrásfájlt a futtatás eredményével együtt megmutatni. A

```
\MESourceFileWithResult[keys]{source  
file} {result file} {caption} {label}
```

makró ezt teszi lehetővé. A forráskódban it is megjelölhetünk 'nevezetes pontokat', az eredményfájlban ez nem lehetséges.



### Listing 3.10. A kalkulator program és eredménye

```
// Expensive Calculator
// Demonstrates built-in
// arithmetic operators

#include <iostream>
using namespace std;

int main()
{
    cout << "7 + 3 = " << 7 + 3 <<
        endl;
    cout << "7 - 3 = " << 7 - 3 <<
        endl;
    cout << "7 * 3 = " << 7 * 3 <<
        endl;

    cout << "7 / 3 = " << 7 / 3 <<
        endl;
    cout << "7.0 / 3.0 = " << 7.0 /
        3.0 << endl;

    cout << "7 % 3 = " << 7 % 3 <<
        endl;

    cout << "7 + 3 * 5 = " << 7 + 3
        * 5 << endl;
    cout << "(7 + 3) * 5 = " << (7 +
        3) * 5 << endl;

    return 0;
}
```

### Listing 3.11. A kalkulator program és eredménye

```
7 + 3 = 10
7 - 3 = 4
7 * 3 = 21
7 / 3 = 2
7.0 / 3.0 = 2.33333
7 % 3 = 1
7 + 3 * 5 = 22
(7 + 3) * 5 = 50
```

## 3.6. További program nyelvek

Saját céljaimra a 'listings' csomagban definiáltakon felül, további program nyelveket definiáltam:

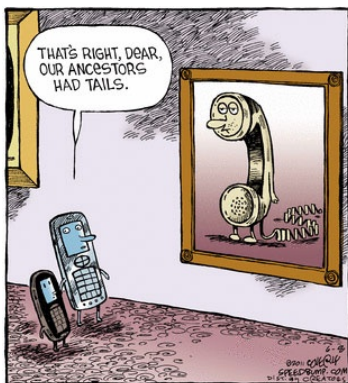
- diff
- [DIY]Assembler
- [ARM]Assembler
- [x64]Assembler
- [y86]Assembler

---

# Ábrák beszúrása

## 4.1. Hagyományos ábrák

©2011 <http://pinterest.com>



4.1. ábra. Amikor régi és új telefonok találkoznak

Az 4.1 ábra előállításához a

```
\MEfigure{fig/phone_ancestors} {{Regi es  
uj telefonok ha talalkoznak}}  
{fig:phonenachestors} {2011  
http://pinterest.com}{.8}
```

---

# Finomhangolás

A **MultEdu** rendszer tökéletesen működik alapértelmezett beállításokkal is, de nem gondolatolvasó. A beállításokat `\def{\xxx}` formájú definíciókkal lehet megváltoztatni. A beállítások helye üzemmódtól függ, a részleteket lásd a **6** szakaszban. Az alapértelmezett beállítások az egyes beállítások hatásának részletes leírásánál találhatók. A fejezet következő szakaszai az üzemmódok használatát mutatja be.

## 5.1. Alap beállítások

A **MultEdu** beállítási lehetőségként vagy fájlok megadott helyen és néven való előfordulását, vagy pedig `\def{Option}{Value}` formájú definíciók előfordulását tudja értelmezni. Ezek hiánya esetén az alapértelmezett viselkedés lét életbe az eredmény fájl előállítása során.

A beállítási lehetőségek lehetnek kötelezően használandók; az eredmény fájl nagy mértékben befolyásolók, vagy csak kisebb finomítást jelentők; csak bizonyos típusú eredmény fájl készítésekor hatásosak.

## 5.2. A MultEdu csomag beállítási lehetőségei

### 5.2.1. Beamer alapú formátum beállítások

A MultEdu lehetővé teszi kétféle elterjedt formátum használatát. Egyre gyakoribb 16:9 arányú képformátum így az az alap beállítás. A 4:3 arányú képformátumot a

```
{\def\DisableWideScreen{YES}}
```

definiálásával lehet beállítani.

Néha (főként rövid bemutatók esetén) egyáltalán nincs szükség tartalomjegyzékre. Ezt a

```
{\def\DisableTOC{YES}}
```

definiálásával lehet elérni. Az is előfordul, hogy a fejezet-szintű tartalomjegyzék még szükséges, de a szakasz szintű már nem. Ezt a

```
{\def\DisableSectionTOC{YES}}
```

definiálásával lehet elérni.

## 5.3. A MultEdu csomag fájljai

A használt fájloknak illeszkedni kell a fájlok általános rendszerébe, lásd 1.3.2 szakasz. Tanácsos csak a projekt könyvtárba tartozó fájlokat változtatni, mivel a csomag közösen használt fájljai a köteget feldolgozás során felülíródnak.

### 5.3.1. Alapértelmezett

A dokumentumokhoz tartozik néhány fejzet leíró definíció. Mintaként a felhasználói leírás `src/Heading.tex` fájlja szolgál.

A fejzetet olyan fázisban olvassa a program, amikor még nem használhatók a magyar ékezetes betűk, ezért azokat a szokásos L<sup>A</sup>T<sub>E</sub>X kódolással kell írni. A fejzet tartalma:

```
A \def\LectureAuthor{V\’egh J\’anos} sor ad-
ja meg a szerzőt, a \def\LectureTitle{Hogyan
haszn\’aljuk\\ a MultEdu csomagot} a címét, a
\def\LectureSubtitle{(Hogyan
k\’esz\’\i{}ts\’unk
\’erdekes\\ \’es vonz\’o tananyagot)} pedig
```



a dokumentum címét és alcímét. Megadhatunk egy `\def\LecturePublisher{Egyetem neve vagy konferencia neve}` meghatározást is. Javasolt egy `\def\LectureRevision{V\Version\ \at year.mm.dd}` formájú sor használata is

```
\ifthenelse{\equal{\LectureLanguage}{magyar}}
{% in Hungarian
}% true
{% NOT magyar
}
```

blokkban kell elhelyezni.

Megadhatunk számítógépes címet is

```
\def\LectureEmail{Janos.Vegh\at unideb.hu}
```

Ugyancsak itt célszerű megadni a dokumentumban használt BibTeX fájlokat, akár a nyelv, vagy a fájl tényleges fellelhetősége alapján:

```
\IfFileExists{src/Bibliographyhu}
```

```
{\def\LectureBibliography{src/Bibliography,  
src/Bibliographyhu}}
```

```
{\def\LectureBibliography{src/Bibliography}}
```

---

# A dokumentum fordítása

## 6.1. Kézi fordítás

A `Main.tex` fájl a közös és a két fordítási módban egyformán használt rész: ez tartalmazza a tényleges forráskódot. Az ebben a fájlban (továbbá az ide beolvasott fájlokban) szereplő bármely beállítás változtatás megváltoztatja a rendszer beállításait, azaz itt nem tanácsos bármiféle beállítást használni. Érdeemes az összes beállítást egyetlen fájlba gyűjteni, amit aztán a fő fájl magába olvas.

A tananyag fejlesztést általában valamilyen szerkesztőbe integrált fejlesztő rendszerrel érdemes végezni. A szerkesztőbe be kell olvasni a boríték fájlt (a `Demo.tex` megfelelőjét) és azt gyökér dokumentumként megjelölni. A `Main.tex` fájlban érdemes hozzáadni a hivatkozásokat a tananyag fejezeteire, ami anyagokat természetesen a `src` alkönyvtárban célszerű elhelyezni, követve a demonstrációs anyag elrendezését.

A beállítások tárolására szolgáló fájlt is a `src` alkönyvtárban érdemes elhelyezni, célszerűen `Defines.tex` néven. A burkolóként szolgáló `Demo.tex` feladata, hogy ezt és a fő fájlt beolvassa.

A kötegelt mód a konfigurálás során készít egy `Defines.tex` fájlt, de az a `build/build/src` alkönyvtárba kerül. (Onnét lehet puskázni, hogy mit és hogyan érdemes beállítani; miután egyszer már futott a kötegelt fordítás.) A kötegelt fordítás egy "minta" fájlt is készít `Defines.tex.in` néven a `src` alkönyvtárba. Ennek a két fájlnek a tartalma a kötegelt fordítás utolsó menetének felel meg.

## 6.2. Kötegelt fordítás

A kötegelt fordítás (főként) arra szolgál, hogy a közös forráskódból kényelmesen tudjuk előállítani a különféle formátumokban és nyelveken anyagainkat.

Technikai okokból a tényleges fordítás előtt a rendszer saját másolatot készít a **MultEdu** szükséges fájljairól a projekt **common** alkönyvtárába. Ezzel a saját kópiával lehet kísérletezni, vagy akár törölni; a következő kötegelt fordítás majd helyreállítja. (azaz a következő fordítás előtt az értékes fejlesztést el kell menteni, akár a `../..common` alkönyvtárba, ha azt másutt is használni akarjuk.)

A fordítás három lépésből áll.

- a projekt könyvtárban a **CMakeLists.txt** fájlban be kell állítani az adott fordításban használni kívánt beállításokat
- a projekt **build/build** alkönyvtárára váltani, majd kiadni a **cmake ../..** parancsot. Ennek hatására a **MultEdu** elkészíti a konfigurációs és forrás fájlokat (ki is írja, hogy milyen fájlokat fog elkészíteni)

- ugyanitt adjuk ki a **make** parancsot, aminek hatására a tényleges fordítás elindul.

A fordítás alapértelmezetten is hosszabb, mint ami az IDE használata esetén megszokott. A kötegelt feldolgozás a legrosszabb esetre számít: mindent újrafordít, akkor is, ha tulajdonképpen nincs szükség rá. Ezt az időt szorozni kell a nyelvek és a formátumok számával, azaz hosszabb dokumentumok esetén akár több perces fordításra számítsunk.

Viszont, a fordítás függetlenül folyik, és a **MultEdu** saját másolatot készít saját fordításához, azaz a kötegelt fordítás alatt további szerkesztéseket végezhetünk. Hasonlóan független a forrás és eredményfájlok kezelése is. A **MultEdu** elkészít magának egy saját forráskódot, a megfelelő beállításokkal, és azt fordítja le egy saját eredményfájllá. A saját másolatok nevei tartalmazzák a nyelv, a formátum és a verzió kódját is.

## 6.3. Az alapbeállítások megváltoztatása

A **MultEdu** alap-beállításait `\def{OptionName}` utasításokkal lehet meghatározni. Amennyiben a fordítás előtt a fordítóprogram nem talál ilyen meghatározást, az alapbeállítást használja. A kézi és a köteget fordítás beállításai különböznek. A köteget feldolgozás esetén a fordítóprogram a `CMakeFiles.txt` fájlban megadott beállításokkal újonnan létrehozott `build/build/src/Defines.tex` meghatározásokat használja, a kézi fordítás pedig a `src/Defines.tex` meghatározásokat. Ezek célszerűen megegyeznek, de az utóbbi beállításokat a felhasználónak kell megadni.

### 6.3.1. A verziók kezelése

A **MultEdu** a standard háromszintű verzió számozást használja (fő és alszám, valamint folt). A **MultEdu**val készült anyagoknak kétféle verziója van: a saját tananyagának verzióját a felhasználó tartja karban, a **MultEdu** változatát pedig a fejlesztő.



A **MultEdu** verziószáma a `../../common/MEMacros.tex` fájlban található; célszerű változtatlanul hagyni. A saját kurzus anyag verzióját a `CMakeFiles.txt` file tartalmazza, az minden köteget fordítás alkalmával frissül a `Defines.tex` fájlban. A kézi fordításnak saját beállításai vannak, de célszerű azt átvenni a generált fájlból.

A saját verzió száma a generált kimeneti fájl nevében is szerepel, tehát érdemes következetesen használni azt. Használata:

```
\def\Version{nagy.kis.folt}
```

### 6.3.2. Nyelvek kezelése

A **MultEdu** egy- és két-nyelvű dokumentumokat tud kezelni. A különböző nyelvekhez különböző helyesírás, fejezetcímek, feliratok tartoznak. A beállításoknál kell megadni a nyelvet: ezt pl. a `\LectureLanguage{magyar}` beállítással lehet megtenni (enélkül az alapbeállítás `\LectureLanguage{english}`).

A kiválasztott nyelv neve az eredmény file nevében is megjelenik.

A két-nyelvű dokumentumokban van egy első és egy második nyelv, amelyen sorrendben szerepelnek a nyelvi szövegek a dokumentumban. Ez lehetővé teszi, hogy az egymás alatt levő kétféle nyelvű kurzus anyagot összhangban tudjuk fejleszteni. A nyelv kiválasztásával a két anyag bármelyikéből tudunk eredmény fájlt generálni. Ha a `\UseSecondLanguage{}` definiálva van, a sorrendben második nyelvet fogja a csomag feldolgozni, és arra a `\LectureLanguage{}` által megadott szabályokat használja.

Kötegelt fordítás esetén meg kell adnunk a `FirstLanguage` és `SecondLanguage` értékét (azaz, hogy az elsőként és másodikként megtalált szöveg milyen nyelvű). Ha bekapcsoljuk a `NEED_BOTH_LANGUAGES` kapcsolót, a kötegelt feldolgozás során mindkét nyelvű kimenő fájlt előállítja a rendszer. Ha ez ki van kapcsolva, akkor a `USE_SECOND_LANGUAGE` kapcsoló dönti el, melyik nyelvet fogja a rendszer használni.

---

# Kiegészítések használata

## 7.1. Rövidítések és szómagyarázat használata

Különösen technikai jellegű tárgyak esetén, gyakran szerepelnek rövidítések, betűszavak, illetve bizonyos fogalmak egyértelmű meghatározása. A **MultEdu** a **glossaries** csomagot használva teszi lehetővé, hogy a dokumentumokban ilyeneket használjon, ráadásul hiper-hivatkozásként.

Az ilyen elemeket a szövegben a `\gls{ref}` módon kell elhelyezni. A szövegben ennek hatására megjelenik az elem rövid neve, és annak első előfordulásakor annak rövid leírása is. Bővebben lásd a **glossaries** csomag leírását.

Különösen technikai jellegű tárgyak esetén, gyakran szerepelnek rövidítések, betűszavak, illetve bizonyos fogalmak egyértelmű meghatározása. A **MultEdu** a **glossaries** csomagot használva teszi lehetővé, hogy a dokumentumokban ilyeneket használjon, ráadásul hiper-hivatkozásként.

Az ilyen elemeket a szövegben a `\gls{ref}` hivatkozásként kell elhelyezni, és a nyomtatott szövegben

ennek hatására azon a helyen az elem rövid neve jelenik meg, és a rövidítések feloldására, a hivatkozás első előfordulásakor annak rövid leírása is. Bővebben lásd a **glossaries** csomag leírását.

### 7.1.1. Rövidítések és szómagyarázat használata

Ha mintaként használja a számítógép fogalmát, ahol Central Processing Unit, központi egység (CPU) valamint Direct Memory Access, közvetlen memória elérés (DMA) is előfordul akkor a szövegben a `\gls{minta}k\’ent haszn\’alja a \gls{szamitogep} fogalm\’at, ahol \gls{CPU} valamint \gls{DMA} is el\H{o}fordul` módon kell azt használni. Ilyenkor a **MultEdu** hozzáfűzi a dokumentumhoz a **Acronyms** and **Glossary** fejezeteket, ahol a megjelölt hivatkozások kifejtése található. A dokumentum olvasásakor a hivatkozásra kattintva, az olvasó program a kifejtésre ugrik, ahonnet az oldalszámra kattintva, folytathatja az olvasást.

A **MultEdu** azt várja, hogy (ha használni akar ilyen lehetőséget) a projekt tartalmaz egy

`src/Glossary.tex` fájlt, ahol a hivatkozások részletes kifejtése megtalálható. A bemutatott mintában a bejegyzések kódolása:

### 7.1.2. Rövidítések és szómagyarázat meghatározása

```
\ifthenelse{\equal{\LectureLanguage}{english}
{
\newglossaryentry{computer}
{
name={computer},
description={is a programmable machine
that receives input,
stores and manipulates data, and provides
output in a useful format}
}
\newglossaryentry{sampleone}{name={sample},description={
little example}}
\newacronym{CPU}{CPU}{Central Processing
Unit}
\newacronym{DMA}{DMA}{Direct Memory
Access}
}
```

{}

### 7.1.3. Rövidítések és szómagyarázat használata

Ezeknek a lehetőségeknek csak a nyomtatható változatok esetén van szerepe. A `beamer` alapú formátumok nem generálnak ilyen jegyzékeket, de a `\gls{ref}` természetesen ott is használható.

Nagyon jó lehetőség arra, hogy a rövidítés kifejtés, fogalom magyarázat, stb. ne törje meg a szöveget, de azért mindig kéznél legyen.

## **7.2. Indexek használata**



## **7.3. Irodalom jegyzék**



---

# Tárgymutató

<code>\MESourceFileCompare,</code>	<code>\MEchapterillustration</code>
38	21
<code>\MESourceFileWithResult,</code>	<code>\MEDchapter,</code>
40	20
	<code>\MEDframe,</code>
<code>\EnableTimer,</code>	20
19	<code>\MEframe,</code>
	15
<code>Heading.texx,</code>	<code>\MEsection,</code>
48	16
<code>idő kijelzése dián,</code>	<code>\MESetListingFormat,</code>
19	25
	<code>\MESetStandardListin,</code>
<code>\LectureTime,</code>	25
19	<code>MESourceFile</code>
<code>\MEchapter,</code>	<code>\MESourceFile</code>
16	

- megjegyzés,  
33
- \MESourceFile, 27
- \MESourcelineComment,  
33
- \MESourcelineFigure,  
37
- \MESourcelinesHighlight,  
31, 32
- \MESourceListBalls,  
35
- package
  - beamer, 15
  - beamerarticle, 15
  - listings, 24
- src/Defines.tex, 19
- src/Heading.tex, 48
- \UseSecondLanguage,  
17



---

# Acronyms

C | D

C

CPU

Central Processing Unit, központi egység. 61

D

DMA

Direct Memory Access, közvetlen memória  
elérés. 61



---

# Szójegyzék

**M** | **S**

**M**

**minta**

egy minta. [61](#)

**MultEdu**

A MultEdu L<sup>A</sup>T<sub>E</sub>X-alapú makró csomag, ami különböző formátumú és nyelvű .pdf eredményfájlokat készít, ugyanabból a forrás fájlból. Elsősorban tananyag készítés céljára.. [5–7](#), [12](#), [15](#), [18](#), [19](#), [24](#), [25](#), [45–47](#), [54–57](#), [60](#), [61](#)

## S

**számítógép**

olyan programozható gép, amelyik adatokat fogad, tárol és feldolgoz, valamint értelmes formátumú eredményt szolgáltat. 61



---

# Ábrák jegyzéke

4.1. Amikor régi és új telefonok találkoznak .	44
--	----



---

# Listings

3.1.	A "Hello World"- C++ program . .	28
3.2.	A "Hello World"- C++ program, wide	29
3.3.	"Hello World" – a C++ way, kijelölt	31
3.4.	"Hello World" – egy C++ program .	33
3.5.	"Hello World" – a C++ way, megjegyzés a forráskód sorához . . . . .	34
3.6.	"Hello World" – a C++ way, golyókkal	36
3.7.	'for' ciklus megvalósítása HW ismétléssel . . . . .	37
3.8.	A két kisbetűssé alakító rutin összehasonlítása . . . . .	39
3.9.	A két kisbetűssé alakító rutin összehasonlítása . . . . .	39



3.10. A kalkulator program és eredménye .	41
3.11. A kalkulator program és eredménye .	41

