Dr. Hartung Ferenc, Pozsgai Tamás, Virágh János:

Előadásvázlat matematikai programcsomagok kurzushoz

A felsőfokú informatikai oktatás minőségének fejlesztése, modernizációja

TÁMOP-4.1.2.A/1-11/1-2011-0104

Főkedvezményezett:
Pannon Egyetem
8200 Veszprém
Egyetem u. 10.

Kedvezményezett:
Szegedi Tudományegyetem
6720 Szeged
Dugonics tér 13.

2014

A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Matematikai programcsomagok Szövegkészítés TEX/LATEX-ben

Dr. Hartung Ferenc - Dr. Virágh János - Pozsgai Tamás

2014. május 2.

Félév felépítése

- Matematikai dokumentumkészítés LaTeX nyelven.
- Matematikai dokumentumok webes megjelenítése.
- Matematikai feladatmegoldás Maple segítségével.

Szövegszerkesztés TEX nyelven

- A TEX a nyomdászat "assemblere".
- Nagyon jól makrózható.
- Előre elkészített makrócsomagok.
  - plain TEX
  - $\mathcal{AMS}$-TEX
  - LATEX

Általánosan elfogadott a LATEX lett. Jelenleg az 1994-ben megjelent LATEX2_ϵ használjuk. Azóta több csomaggal bővült.

Előnyök

- Professzionális tipográfia.
- Egyszerű, könnyen megjegyezhető nyelv.
- Platformfüggetlen.
- Kis méretű forráskód.
- Ingyenes.

Hátrányok

- Hibák javítása nehéz, sok "álhiba".
- Az első néhány dokumentum elkészítése hosszú időbe telhet.
- Készítés közben nem látható, mit is csinálunk.
- Új stílus kialakítása bonyolult.

Telepítés

- Windowsos környezetben a legáltalánosabban elterjedt TEX programcsomag a MikTeX. Letölthető: www.miktex.org
- Telepítése kétféleképpen lehetséges, Basic, illetve Complete módon. Javasolt a Complete (Net installer). Utóbbihoz élő hálózati kapcsolat szükséges a telepítés során.
- TEX/LATEX file-ok már fordíthatók ekkor.
- A TEX/LATEX file-ok szerkesztésére több eszköz is rendelkezésre áll.

Telepítés Linuxos környezetben

- Bármelyik disztibúcióban telepíthető.
- Telepítő cd-n, dvd-n rajta vannak a telepítő csomagok.
- A TEX/LATEX file-ok szerkesztésére itt is bármelyik szövegszerkesztő használható.
- A népszerű emacs program TEX-es verziója.
  http://www.xemacs.org/

TEX szerkesztők

Önmagukban nem fordítanak, csak szerkeszteni lehet a fájlokat.

- Szöveges fájlok, szerkeszthetők akár Jegyzettömbben is.
- A MikTeX saját szerkesztőke a TeXwords.
- Könnyen, jól használható program a TexnicCenter. Letölthető: http://www.texniccenter.org/
- Telepítését a MikTex telepítése után javasolt megtenni.

A LyX szerkesztő

- Másik lehetőség TEX fájlok szerkeztésére a Lyx. Letölthető: http://www.lyx.org/
- Ez a szerkesztő WYSIWYM tipusú, de nem tex file-okat hoz létre, hanem lyx projekteket.
- Képes tex fájlokat is megjeleníteni.

Ajánlott oldalak

- TEX linkgyűjtemény
  http://latex.lap.hu
- Mini tanfolyam
  http://www.komal.hu/forum/forum.cgi?a=tk
- BME math LATEX
  http://www.math.bme.hu/latex/

Alap LaTeX dokumentum

```
\documentclass[opció]{dokumentumosztály neve}}
deklarációs rész
  \begin{document}}
   dokumentum törzse
 \end{document}}
```

Első LaTeX dokumentum

A Helló világ! minden formázás nélkül:

```
\documentclass{article}
   \begin{document}
     Helló világ!
   \end{document}
```

Csomagok

Amennyiben az alaputasításokon kívül szeretnénk használni újabb utasításokat, akkor sokféle csomagot vehetünk igénybe.

- Deklarációs részben kell megadni a használni kívánt csomagokat a következő utasítással:
  `\usepackage[opciók]{csomag neve}`
- Opcióból lehet több is, ekkor vesszővel elválasztva soroljuk fel őket.
- A { és a } közötti részt hívjuk a parancs argumentumának.

Blokk, környezet

Ha egy utasítást érvényességét egy adott szövegrészre szeretnénk alkalmazni, akkor a következőt tehetjük:

```
\textsl{Ez dölt szöveg lesz.}
```

A kapcsoszárójelpár közötti részt nevezzük blokknak.
Hasonló eredményt érünk el a következő utasítással:

```
\begin{textsl} Ez dölt szöveg lesz. \end{textsl}
```

Ezt egy környezetnek nevezzük. Sok formai elemet tudunk megvalósítani környezetek segítségével (felsorolások, számozások, keretek).

Kommentek

- Egysoros komment elhelyezése: a % jel mögötti szöveget nem veszi figyelembe a fordító.
- Többsoros komment megvalósítása: a `comment` csomag `comment` környezetével történik.

Dokumentumosztályok

- article - cikk
- report - jelentés
- book - könyv
- letter - levél
- slides - prezentáció

Foglalt karakterek

Foglalt karaketerek:

```
\ - parancsok kezdete
% - kommentek
{} - blokkok, parancsok
$  -matematikai mód
&  -táblázatok
# - parancsok definiálása
_ - alsó index
^ - felso index
```

Megjelenítő parancsok

```
\textbackslash
\%
\{, \}
\$
\&
\#
\_
\textasciicircum
```

Magyar LaTeX

Ha magyar szöveget szeretnénk némi tipográfiával, akkor a következő beállításokra lesz szükségünk:

```
\documentclass[a4paper, 12pt]{article}
\PassOptionsToPackage{defaults=hu-min}{magyar.ldf}
\usepackage[magyar]{babel}
\usepackage{t1enc}
\$usepackage[latin2]{inputenc}
```

A babel csomag magyar opciója a magyar.ldf fájlt használja. A MikTeX nem mindig a legújabb változatát használja. A legfrissebb változat letölthető:

`http://www.math.bme.hu/latex/`

Bekezdések

- Új bekezdés készítése a `\par` utasítással.
- Behúzások szabályozása a `\indent` illetve `\noindent` utasításokkal.
- Bekezdések igazítása alapértelmezetten sorkizárt, ha ettől el szeretnénk térni, akkor a következő környezetekkel tudjuk megtenni:
  - balra: flushleft környezet
  - jobbra: flushright környezet
  - középre: center környezet

Tabulátorok

- Tabbing környezetben valósítható meg.
- \= - új tabulátor elhelyezése a sorban.
- \> - ugrás a következő tabulátorpozícióba.
- \\ - új sor kezdése.

Felsorolások

- Itemize környezetben.
- `\item` után következik egy-egy felsoroláselem.
- Négy szintig ágyazható egymásba.

Számozott lista

- Enumerate környezetben.
- \item után következik egy-egy felsoroláselem.
- Opcióként megváltoztatható az alapértelmezett számozás.

Széljegyzet, lábjegyzet

- Széljegyzet készítése: `\marginpar{széljegyzet szövege}`.
- Alapértelmezésben a bal margóra kerül.
- Lábjegyzet készítése: `\footnote{lábjegyzet szövege}`.
- A lábjegyzet számozása fejezetenként folyamatos.

Idézetek

- Egysoros idézet: `\verb+ idézet szövege +`.
- A verb utáni + jel helyett bármilyen határolójelet használhatunk (kivéve a foglalt karakterek).
- Többsoros idézet: verbatim környezetben.

Programkód

- Programkódok megjelenítésére a listings csomagot használhatjuk.
- A kódot magát az lstlisting környezetben adhatjuk meg.
- Lehetőségünk van különböző programozási nyelvek szintaxisának megfelelő kiemelésére.

Táblázatok

Táblázatok

- Táblázatok készítése az egyik legbonyolultabb és legnehezebb feladat.
- Alapesetben a tabular környezetben valósítható meg.
- Általános szintaxisa:

```
\begin{tabular}{beállítások}
a & b & c\\
d & e & f
\end{tabular}
```

A beállítások lehetnek igazítások, szélesség, de itt lehet megadni egyéni táblázatformátumot is.

Táblázatok

- Oszlopok elválasztójele a &.
- Új sor nyitása: \\.
- Igazítások lehetnek l (balra), r (jobbra) , c (középre)
- Szegélyek:
  - Vízszintes: `\hline` utasítással
  - Függőleges: Tabulátor argumentumában | jellel.

Táblázatok példa

```
\begin{tabular}{|l|rrr|}
\hline
Gipsz Jakab & 40 & 38 & 10\\
\hline
Kerek Elek & 23 & 24 & 10\\
\hline
\end{tabular}
```

A bal oldalon látható kód az alábbi táblázatot hozza létre:

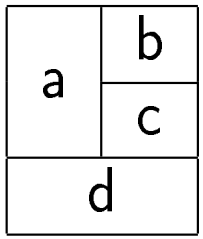
Gipsz Jakab	40	38	10
Kerek Elek	23	24	10

Táblázatok

Egy kicsivel bonyolultabb táblázat:

```
\begin{tabular}{|c|c|}
 \hline
 \multirow{2}*{a}& b\\
 \cline{2-2}
                 & c\\
 \hline
 \multicolumn{2}{|c|}{d}\\
 \hline
\end{tabular}
```

Ennek a táblázatnak az elkészítéséhez már be kell töltenünk a multirow csomagot.



Táblázatok

Bonyolultabb báblázatok elkészítéséhez további csomagok állnak a rendelkezésünkre.

- multirow
- booktabs
- supertabular
- longtable
- colortbl

Dobozok

A dobozok olyan részei a dokumentumnak, melyet egy egységként kezel a TEX, tartalma nem törhető meg.

- LR dobozok (egysoros doboz)
- Bekezdésdoboz
- Vonaldoboz

LR dobozok

- `\makebox[doboz szélessége][szöveg pozíciója]{szöveg}`
- `\framebox[doboz szélessége][szöveg pozíciója]{szöveg}`
- `\mbox{szöveg}` = `\makebox{szöveg}`
- `\fbox{szöveg}` = `\framebox{szöveg}`
- `\raisebox{emelés}{szöveg}`

Bekezdésdoboz

- `\parbox[dpoz][magasság][szpoz]{szélesség}{szöveg}`
- Minipage környezet segítségével:

```
\begin{minipage}[dpoz][magasság][szpoz]{szélesség}
szöveg
\end{minipage}
```

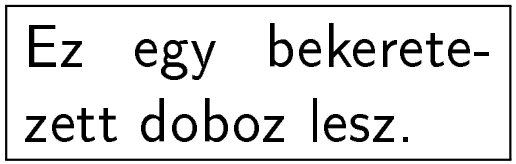
- Minipage esetén például lábjegyzet is használható.

Dobozok egymásba ágyazása

A dobozokat természetesen egymásba is ágyazhatjuk. Ez még olyan egyszerű esetben is jól jöhet, mint a dobozok szegélyezése. Tekintsük a következő példát:

```
\fbox{\parbox{3cm}
{Ez egy bekeretezett
   doboz lesz.}}
```

Ennek a táblázatnak az elkészítéséhez már be kell töltenünk a multirow csomagot.



Vonaldoboz

Tömör téglalap rajzolására használható a vonaldoboz is.
Szintaxisa: `\rule[emelés]{szélesség}{magasság}`
Ahol

- emelés: alapvonaltól ennyivel lesz feljebb
- szélesség: vonaldoboz hossza
- magasság: vonaldoboz magassága.

További dobozkészítési lehetőségek

További dobozkészítő utasítások találhatók különböző csomagokban.

- Szines dobozok - xcolor csomag
  `\colorbox{keretszin}{háttérszin}{szöveg}`
- Elforgatott dobozok - graphix csomag
  `\rotatebox[origin=forgatáscentrum]{szög}{szöveg}`
- Elnyújtott dobozok - graphix csomag
  `\scalebox{x}[y]{szöveg}`

Hasábokra tagolás

Kéthasábos szedést a multicol csomag segítségével tudunk létrehozni.
Szintaxisa:

```
\begin{multicols}{n}
szöveg
\end{multicols}
```

ahol n a hasábok száma.
A hasábok közötti távolság szabályozására használható a
`\setlength{\columnsep}{táv}` utasítás.

Dokumentum tagolása

- cím
- kivonat (kivéve book osztály)
- tartalomjegyzék
- úszó objektumok jegyzéke
- főszöveg szintjei
  - részek
  - fejezetek (kivéve article osztály)
  - szakaszok
  - alszakaszok
  - paragrafusok
  - alparagrafosok
  - bármelyiken belül tételszerű bekezdések
- függelék
- bibliográfia
- tárgymutató

Címlap

A dokumentum törzsében a következők adhatók meg:

```
\title{cím}
\author{szerzo}
\date{dátum}
\maketitle
```

Az aktuális dátum automatikus beírására használhatjuk a `\date{\today}` parancsot.
Ezen adatok megadása után ha a `\documentclass titlepage` opciót bekapcsoljuk, a `\maketitle` utasítás elkészíti a címoldalt.

Élőfej és élőláb

- Book osztálynál automatikusan megjelenik
- Article, report osztálynaál a következőképen adható meg:
  `\pagestyle{headings}`
- Adott oldalon a stílus megváltoztatása:
  `\thispagestyle{stílus}`
  ahol a stilus lehet `headings`, `myheadings`, `empty`, `plain`

Tartalomjegyzék

Tartalomjegyzék készítsét automatikusan a `\tableofcontents` utasítással tudjuk elvégezni. A dokumentumban megadott szintek és címeik kerülnek be.
A fordítást többször is el kell végezni.
Első fordításkor a szintcímek és a hozzájuk tartozó oldalszámok egy `.toc` kierjesztésű fájlba írodnak ki.
Következő fordításkor ebből a fájlból olvassa ki a fordító a szükséges adatokat és helyezi el ténylegesen a dokumentum adott pontján a tartalomjegyzéket.

Táblázatok, ábrák jegyzéke

Táblázatok címeiből is készíthető jegyzék. Ahhoz, hogy a táblázat bekerüljön a jegyzékbe, a `table` környezet `\caption{név}` mezőnek kitöltöttnek kell lennie. A jegyzék a `\listoftables` utasítás hatására készül el.
Hasonlóan a tartalomjegyzékhez itt is legalább két fordításra lesz szükségünk. Az első elkészít egy `.lot` kierjesztésű fájlt, a második helyezi el ténylegesen a dokumentumban.
Ábrajegyzék készítése a `\listoffigures` utasítással lehetséges. Az eddigiekhez hasonloan a `figure` környezet `\caption{név}` mezőnek kitöltöttnek kell lennie. Itt az első fordításnál egy `.lof` kiterjesztésű fájl készül, amiből a tényleges jegyzéket készíti a dokumentumba a következő fordításkor a fordítóprogram.

Bibliográfia

A bibliográfia neve a `report` és a `book` oszályban "Irodalomjegyzék", a `article` oszályban "Hivatkozások". Létrehozása:

```
\begin{thebibliography}{példacímke}
\bibitem[címke]{kulcs} elemleírás
...
\end{thebibliography}
```

Elektronikus publikáció

A deklarációs részben az adjuk ki az alábbi utasítást:
`\usepackage[unicode]{hyperref}`
Ennek a hatására a dokumentumunkban a hivatkozások és a könyvjelzők átalakulnak linkekké.
Ha használjuk a `setspace` csomagot is, akkor annak meg kell előzni a `hyperref` csomagot a deklarációs részben. Ellenkező esetben a lábjegyzetek linkjei hibásan működhetnek.
A csomag néhány opciója:

- colorlinks= true, false
- pdfpagemode=FullScreen
- pdfstartview=FitH
- pdfstartview=FitV

Matematikai programcsomagok
Matematikai formulák készítése

Dr. Hartung Ferenc - Dr. Virágh János - Pozsgai Tamás

2014. május 2.

Matematikai mód

Matematikai képletek szerkesztéséhez szükségünk lehet speciálisan erre a célra kialakított csomagok betöltésére. A két leggyakrabban használ csomag a `amsmath` és az `amssymb`. Matematikai szövegek készítésekor különleges betüket is használhatunk, hiszen másképp jelenik meg egy "a" betű, ha névelő és másképpen, ha egy változó neve.
Alapvetően két különböző képletszerkesztési módod különböztetünk meg:

- Soron belüli matematikai mód
- Kiemelt matematikai mód

Matematikai mód

Szövegközi matematikai mód megadása:

- `$ formula $`
- math környezetben

Kiemelt matematika mód megadása:

- `\[ formula \]`
- displaymath környezetben
- equation környezetben

Említést érdemel még a `\ensuremath{formula}` utasítás is, aminek az argumentuma mindenképpen matematikai módban lesz, függetlenül attól, hogy az utasítást normál szövegként, vagy matematikai módban használjuk.

Szöveg matematikai módban

Kétféle lehetőségünk van rá, hogy szöveget helyezzünk el a formuláinkban:

- `\text{szöveg}`
- `\mbox{szöveg}`

Tekintsük a következő példát:

```
\[
\text{Legyen } f(x)=\mathrm{e}^x \quad
\text{minden}\quad x\in\mathbb{R}\quad \text{esetén.}
\]
```

$$\text{Legyen } f(x)=\mathrm{e}^x \quad \text{minden}\quad x\in\mathbb{R}\quad \text{esetén.}$$

Betűstílusok matematikai módban

A normál szövegben megszokott karakterformázó környezetek helyett a következőket használhatjuk matematikai módban:

- `\mathbf{szöveg}`
- `\mathit{szöveg}`
- `\mathrm{szöveg}`
- `\mathtt{szöveg}`
- `\mathsf{szöveg}`
- `\mathnormal{szöveg}`

Matematikai jelek vastagon szedése

Az előző dián bemutatott `\mathbf{szöveg}` utasítás csak a szöveget szedi vastagon, ha formula is szerepel benne, akkor azt nem. A formulák vastagon szedésére a következő lehetőségeink vannak:

- `\boldmath` és `\unboldmath` között
- `\boldsymbol{karakter}`
- `\pmb{karakter}`

Görög betűk

Görög betűket gyakran használunk matematikai formulákban. Többnyire a nevükkel hivatkozunk rájuk, néhány esetben kétféleképpen is:

Görög kisbetűk

- α - \alpha
- ϵ - \epsilon
- π - \pi
- ϕ - \phi
- ε - \varepsilon
- φ - \varphi

Görög nagybetűk

- Γ - \Gamma
- Σ - \Sigma
- Ω - \Omega
- $\varGamma$ - \varGamma
- $\varPi$ - \varPi
- $\varOmega$ - \varOmega

Matematikai ékezetek

Léteznek speciális matematikai ékezetek, amiket például bizonyításoknál használhatunk:

- $\hat{c}$ - `\hat{c}`
- $\tilde{c}$ - `\tilde{c}`
- $\bar{c}$ - `\bar{c}`
- $\vec{c}$ - `\vec{c`
- $\dot{c}$ - `\dot{c}`
- $\mathring{c}$ - `\mathring{c}`

Műveleti jelek

- $\pm$ - \pm
- $\mp$ - \mp
- $\cdot$ - \cdot
- $\times$ - \times
- $\div$ - \div
- $\setminus$ - \setminus
- $\cap$ - \cap
- $\cup$ - \cup
- $\wedge$ - \wedge
- $\vee$ - \vee
- $\star$ - \star
- $\circ$ - \circ
- $\bullet$ - \bullet
- $\oplus$ - \oplus
- $\odot$ - \odot
- $\otimes$ - \otimes

Relációjelek

- $\equiv$ - \equiv
- $\sim$ - \sim
- $\simeq$ - \simeq
- $\approx$ - \approx
- $\cong$ - \cong
- $\leq$ - \le
- $\geq$ - \ge
- $\leqslant$ - \leqslant
- $\geqslant$ - \geqslant
- $\in$ - \in
- $\ni$ - \ni
- $\subset$ - \subset
- $\subseteq$ - \subseteq
- $\parallel$ - \parallel

Relációjelek negálása

Általában a relációs jeleket a `\not` utasítással lehet. Néhány jelnél azonban nemvárt eredményt kapunk. Példaként tekintsük a következőket:

`A \not\subset B`	$A \not\subset B$
`3 \not\mid n`	$3 \not\mid n$
`3 \nmid n`	$3 \nmid n$
`e \not\parallel f`	$e \not\parallel f$
`e \nparallel f`	$e \nparallel f$
`a \not\leqslant b`	$a \not\leqslant b$
`a \lneqq b`	$a \lneqq b$
`A \not\subseteqq B`	$A \not\subseteqq B$
`A \subsetneqq B`	$A \subsetneqq B$

Nyilak

- $\rightarrow$ \rightarrow
- $\longrightarrow$ \longrightarrow
- $\leftrightarrow$ \leftrightarrow
- $\longleftrightarrow$ \longleftrightarrow
- $\Rightarrow$ \Rightarrow
- $\Longrightarrow$ \Longrightarrow
- $\Leftrightarrow$ \Leftrightarrow
- $\Longleftrightarrow$ \Longleftrightarrow
- $\uparrow$ \uparrow
- $\downarrow$ \downarrow
- $\mapsto$ \mapsto
- $\longmapsto$ \longmapsto
- $\nearrow$ \nearrow
- $\searrow$ \subseteq
- $\swarrow$ \swarrow
- $\nwarrow$ \nwarrow

Feliratos nyilak

- $\overset{abc}{\rightarrow}$ `\stackrel{abc}{\rightarrow}`
- $\overset{abc}{\leftarrow}$ `\stackrel{abc}{\leftarrow}`
- $\overset{abc}{\leftrightarrow}$ `\stackrel{abc}{\leftrightarrow}`
- $\overset{\rightarrow}{abc}$ `\stackrel{\rightarrow}{abc}`
- $\overset{\leftarrow}{abc}$ `\stackrel{\leftarrow}{abc}`
- $\overset{\leftrightarrow}{abc}$ `\stackrel{\leftrightarrow}{abc}`

Szebb megoldás változó hosszú nyilakkal:

- $\xleftarrow{abc}$ `\xleftarrow{abc}`
- $\xrightarrow{abc}$ `\xrightarrow{abc}`
- $\xleftarrow[abc]{}$ `\xleftarrow[abc]{}`
- $\xrightarrow[abc]{}$ `\xrightarrow[abc]{}`

Egyéb matematikai jelek

- $\forall$ - \forall
- $\exists$ - \exists
- $\nexists$ - \nexists
- $\Re$ - \Re
- $\Im$ - \Im
- ∇ - \nabla
- $\emptyset$ - \emptyset
- ∞ - \infty
- $\triangle$ - \triangle
- $\square$ - \square
- $\blacksquare$ - \blacksquare
- $\sharp$ - \sharp

Pontozások

Matematikai szövegekben gyakran használunk három pontot. Ennek a használatára több lehetőségünk is van.

- `\dots`
- `\ldots`
- `\cdots`
- `\vdots`
- `\ddots`
- `\dotsc`
- `\dotsb`
- `\dotsm`
- `\dotsi`

- $\dots$
- $\ldots$, alapvonalra igazítva
- $\cdots$, középre igazítva
- $\vdots$, függőlegesen
- $\ddots$, átlósan
- $\dotsc$, vesszők közé
- $\dotsb$, bináris műveletek között
- $\dotsm$, szorzásjelként
- $\dotsi$, integráljelek között

Pontozások

Teljes sort pontokkal kitölteni a következő módokon tudunk.

`\dotfill szöveg`	.....szöveg
`szöveg \dotfill`	szöveg .....
`szöveg \dotfill szöveg`	szöveg .....szöveg

Mátrixokban további lehetőségként megadott számú oszlopot tudunk kipontozni a következő utasítással.

`\hdotsfor[sűrűség]{oszlopszám}`

Matematikai zárójelek

A zárójeleket matematikai módban némileg másképpen használjuk, mint normál szövegben. Emellett szükségünk van olyan jelek megjelenítésére, melyek a TEX-ben más szerepet játszanak (például a {,} zárójelpár).

- ((
- [\lbrack
- { \lbrace
- ⌈ \lceil
- ⌊ \lfloor
- | \vert

-))
-] \rbrack
- } \rbrace
- ⌉ \rceil
- ⌋ \rfloor
- ‖ \Vert

Matematikai zárójelek

A metematikai zárójelektöbbféle szempontból is különlegesen viselkednek. Fontos a megfelelő méretezés, az őket megelőző és követő térközök nagysága. A nyitó és a záró zárójeleknél ez eltérő. Nyitó zárójelszerepet a `left`, záró szerepet pedig a `right` utasítással adunk meg.

```
$\left[0,1\right]$ zárt interv.
$\left|-x\right|=x$
$\left\{\,2n\mid n\
\text{egész}\,\right\}$
```

$\left[0,1\right]$ zárt interv.

$\left|-x\right|=x$

$\left\{\,2n\mid n\ \text{egész}\,\right\}$

Adott méretű matematikai zárójelek

Abban az esetben, amikor az autómatikus méretű zárójel nem megfelelő, lehetőségünk van megadott méretű zárójeleket használni.

A `left(` helyett

- $($ `\bigl(`
- $\Big($ `\Bigl(`
- $\biggl($ `\biggl(`
- $\Biggl($ `\Biggl(`

A `right)` helyett

- $)$ `\bigr)`
- $\Big)$ `\Bigr)`
- $\biggr)$ `\biggr)`
- $\Biggr)$ `\Biggr)`

Adott méretű matematikai zárójelek

Abban az esetben, amikor az autómatikus méretű zárójel nem megfelelő, lehetőségünk van megadott méretű zárójeleket használni.

A `left(` helyett

- $($ `\bigl(`
- $\Big($ `\Bigl(`
- $\biggl($ `\biggl(`
- $\Biggl($ `\Biggl(`

A `right)` helyett

- $)$ `\bigr)`
- $\Big)$ `\Bigr)`
- $\biggr)$ `\biggr)`
- $\Biggr)$ `\Biggr)`

Láthatatlan zárójelek

Abban az esetben, amikor az egyik zárójelre nincs szükségünk, láthatatlan zárójelet használnunk.A méretek igazítása miatt minden esetben szükséges a zárójelpár mindkét tagját megadnunk. Láthatalan nyitó zárójelet a `\left.`, záró zárójelet a `\right.` utasítással tudunk megadni.Tekintsük a következő példát:

$$(1+x)^3\Big|_{x=0}=1$$

Forráskódja:

```
\[
\left.\left(1+x\right)^3\right|_{x=0}=1
\]
```

Esetszétválasztásos függvény

Az esetszétválasztásos függvény szép példáját adja a láthatatlan zárójeleknek. Tömbök segítségével megoldható a függvény definiálása, azonban lehetőségünk van szebb megoldást is elkészíteni a cases környezetben.

$$f(x) := \begin{cases} -1 & \text{ha } x < 0, \\ 0 & \text{ha } x = 0, \\ 1 & \text{ha } x > 0. \end{cases}$$

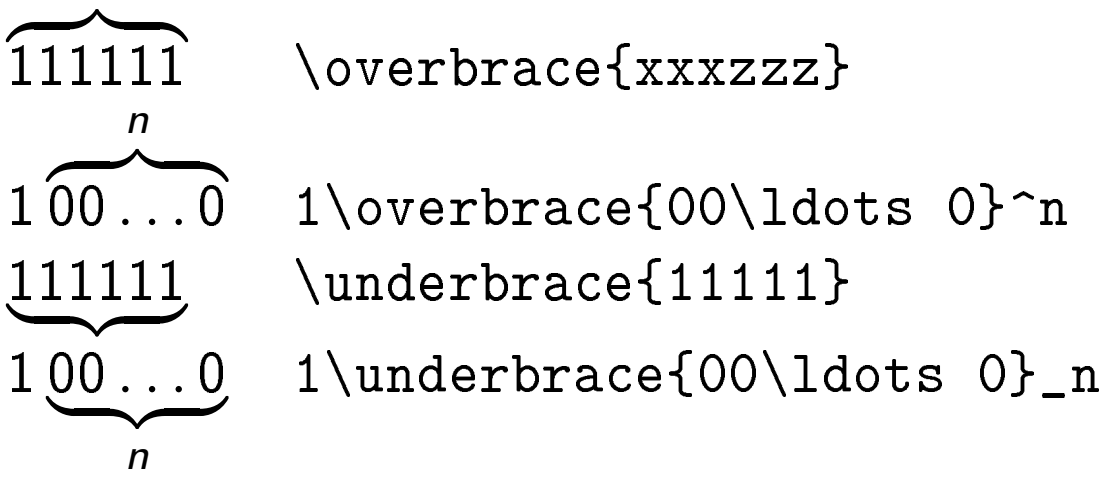
Forráskódja:

```
\[ f(x):=
\begin{cases}
-1 & \text{ha $x<0$,}\\
0 & \text{ha $x=0$,}\\
1 & \text{ha $x>0$.}
\end{cases} \]
```

Vízszintes jelek

- $\widehat{abc}$ - \widehat{xabc}
- $\widetilde{abc}$ - \widetilde{abc}
- $\overline{abc}$ - \overline{abc}
- $\underline{abc}$ - \underline{abc}
- $\overleftarrow{abc}$ - \overleftarrow{abc}
- $\underleftarrow{abc}$ - \underleftarrow{abc}
- $\overrightarrow{abc}$ - \overrightarrow{abc}
- $\underrightarrow{abc}$ - \underrightarrow{abc}
- $\overleftrightarrow{abc}$ - \overleftrightarrow{abc}
- $\underleftrightarrow{abc}$ - \underleftrightarrow{abc}

Vízszintes jelek




```
\overbrace{xxxzzz}
1\overbrace{00\ldots 0}^n
\underbrace{11111}
1\underbrace{00\ldots 0}_n
```

Vízszintes matematikai térközök

Normál szövegben vízszintes térköz a \hspace{hossz} utasítással készíthető. Matematikai módban lehetőségünk van előre definiált térközöket használni.

\ ill. \space	normál szóköz (\hspace{0.33333em})
\enskip	\hspace{0.5em}
\quad	\hspace{1em}
\qquad	\hspace{2em}
\hfill	\hspace{\fill}
\thinspace	\hspace{3/18em}
\medspace	\hspace{4/18em}
\thickspace	\hspace{5/18em}
\negthinspace	\hspace{-3/18em}
\negmedspace	\hspace{-4/18em}
\negthickspace	\hspace{-5/18em}

Egymásra helyezett matematikai jelek

Az eddig ismertetett lehetőségek (nyilaknál illetve az `\overbrace` stb. utasítások) végeredménye egy reláció volt. Ha ettől el szeretnénk térni, akkor használhatjuk a következő utasításokat:

- `\overset{mit}{mire}`
- `\underset{mit}{mi alá}`

Ezeknek az utasításoknak a típusa megegyezik a második argumentumba írt jel típusával. Tekintsük az alábbi példát:

```
a\overset{*}{/}b
```
$a \overset{*}{/} b$

```
a\underset{/}{*}b
```
$a \underset{/}{*} b$

Egymásra helyezett matematikai jelek

Abban az esetben, amikor matematikai jeleket több sorban, a következő két lehetőségünk adódik:

- `\substack{sor1\\ sor2\\ ...}`
- `subarray` környezet

Tekintsük az alábbi két példát:

```
\[\lim_{\substack{x\to x_0\\y\to y_0}}\]
```

$$\lim_{\substack{x\to x_0\\y\to y_0}}$$

```
\[\lim_{\begin{subarray}{l}
x\to x_0\\ y\to y_0
\end{subarray}}\]
```

$$\lim_{\begin{subarray}{l}x\to x_0\\ y\to y_0\end{subarray}}$$

Indexelés

Alsó illetve felső indexek készítésekor figyelnünk kell arra, hogy az indexelő utasítás csak a mögötte álló karakterre vagy utasításra érvényes. Ha több jelet szeretnénk az indexbe elhelyezni, akkor használjuk az indexelést környezetszerűen.

- x_n
- x_{n+1}
- x_{n+1}
- x^n
- x^{n+1}
- x^{n+1}

- `x_n`
- `x_{n+1}`
- `x\sb{n+1}`
- `x^n`
- `x^{n+1}`
- `x\sp{n+1}`

Nagy operátorok

Ezeknek az operátorjeleknek általános jellemzője, hogy szövegközi matematikai módban más méretben jelennek meg, mint kiemelt módban.

- $\sum$ - `\sum`
- $\prod$ - `\prod`
- $\int$ - `\int`
- $\iint$ - `\iint`
- $\oint$ - `\oint`
- $\bigcap$ - `\bigcap`
- $\bigcup$ - `\bigcup`
- $\bigotimes$ - `\bigotimes`
- $\bigoplus$ - `\bigoplus`
- $\biguplus$ - `\biguplus`

Nagy operátorok indexelése

Szövegközi módban ezeknek az operátoroknak az indexei a jel mellett jelennek meg, kiemelt módban pedig felette illetve alatta. A következő példában látható a különbség.
A `$\sum_{n=1}^\infty a_n$` szövegközi módban megadott formula a következőképpen kezeli az indexeket:

$$\sum_{n=1}^\infty a_n$$

Ugyanez a formula kiemelt módban:

```
\[\sum_{n=1}^\infty a_n\]
```

$$\sum_{n=1}^\infty a_n$$

Nagy operátorok indexelése

Amennyiben szerenénk megváltoztatni az indexelés formáját, akkor a `\limits` és a `\nolimits` parancsokkal tehetjük ezt meg. Azonban szövegközi módban nem biztos, hogy jobban mutat a $\sum\limits_{n=1}^{\infty} a_n$ formula, mint az eredeti $\sum_{n=1}^{\infty} a_n$.
Az előző mondatban a megváltoztatott formula forráskódja:
`$\sum\limits_{n=1}^\infty a_n$`
Kiemelt módban az indexek a szöveg mögé kerülnek a `\nolimits` hatására:
`\[\sum\nolimits_{n=1}^\infty a_n\]`

$$\sum\nolimits_{n=1}^{\infty} a_n$$

Tételszerű bekezdések

A tételek, definíciók, bizonyítások speciális formátumot igényelnek. Adott esetben ezeket a bekezdéseket számozni is szeretnénk. Ilyenkor használhatjuk a `\newtheorem` paranccsal létrehozott környezeteket. Ezeket a definíciókat a definíciós részben kell megtennünk. Háromféleképpen hozhatunk létre ilyen környezetet:

- `\newtheorem{tételnév}{tételcím}`
- `\newtheorem{tételnév}{tételcím}[számláló]`
- `\newtheorem{tételnév}[együttnév]{tételcím}`

A newtheorem paraméterei

- `tételnév`: a létrehozott környezet neve.
- `tételcím`: a létrehozott környeet típuscíme (pl. tétel, definíció, példa)
- `számláló`: egy, már korábban definiált számláló. Akkor használatos általában, ha többszintű számozást szeretnénk elérni (2.1. tétel).
- `együttnév`: másik tételszerű környezet neve. Ekkor a tételnév és az együttnév szmámlálói együtt növekednek

Tételszerű bekezdések példa

```
\begin{tetel}
Legyen E az emberek halmaza.
\end{tetel}
\begin{tetel}[Euklidesz\label{eukl}
Minden test ...
\end{tetel}
\Aref{eukl}.~tételből következően\dots
```

1. tétel. *Legyen E az emberek halmaza.*

2. tétel (Euklidesz). *Minden test ...*

A 2. tételből következően...

A proof környezet

A matematikai bizonyítások létrehozására egy előre definiált környezet, a `\proof` ad lehetőséget. Ez a környezet az `amsthm` csomagban található.

```
\begin{proof}
A bizonyítás szövege.
\end{proof}
```

Bizonyítás. A bizonyítás szövege. □

A □ az ún. Q.E.D. jel. Ezt átdefiniálhatjuk pl. △ jelre az alábbi módon:

```
\renewcommand{\qedsymbol}{\triangle}
```

Ha nem akarunk Q.E.D. jelet, akkor a `\quedsymbol` utasításnak üres paramétert adjunk:

```
\renewcommand{\qedsymbol}{}
```

A proof környezet

A bizonyítás vége jel (Q.E.D.) egy sorral lejjebb kerül, mint a bizonyítás utolsó sora, ha ez a sor kiemelt matematikai módban van. Ennek elkerülésére a következőt tehetjük:

```
\begin{proof}
...
\[0=0. \qedhere\]
\end{proof}
```

Bizonyítás. ...

$$0=0. \qquad \square$$

A proof környezet

Jobban áttekinthetővé tesz ia dokumentumunkat,h a bizonyítás kezdetekor hivatkozunk a bizonyítandó tétel sorszámára.

```
\begin{tetel}\label{bf}
A tétel szövege.
\end{tetel}
\begin{proof}[\Aref{bf}.~tétel bizonyítása]
A bizonyítás szövege.
\end{proof}
```

1.1. tétel. A tétel szövege.

Az 1.1. tétel bizonyítása. A bizonyítás szövege. □

Beépített függvények

A beépített függvényeket két részre oszhatjuk. Többségében a függvényeknél az indexek kiemelt matematikai módban is a függvény mögé kerülnek (`\nolimits` módon lettek definiálva).

- $\sin(x)$ `\sin(x)`
- $\sinh(x)$ `\sinh(x)`
- $\cos(x)$ `\cos(x)`
- $\cosh(x)$ `\cosh(x)`
- $\arccos(x)$ `\arccos(x)`
- $\arcsin(x)$ `\arcsin(x)`
- $\arctan(x)$ `\arctan(x)`
- $\tan(x)$ `\tan(x)`
- $\tanh(x)$ `\tanh(x)`
- $\cot(x)$ `\cot(x)`
- $\coth(x)$ `\coth(x)`
- $\log(x)$ `\log(x)`
- $\lg(x)$ `\lg(x)`
- $\ln(x)$ `\ln(x)`

Beépített függvények

Tekintsünk néhány olyan függvényt, amelyek `\limits` módon lettek definiálva.

- $\det(A)$ `\det(A)`
- $\min(H)$ `\min(H)`
- $\max(H)$ `\max(H)`
- $\lim(f(x))$ `\lim(f(x))`
- $\limsup(f(x))$ `\limsup(f(x)`
- $\liminf(f(x))$ `\liminf(f(x))`

Függvény definiálása

Több olyan függvény is található, melyeket másképpen nevez a magyar szokás (tangens, cotangens, hiperbolikus függvények). Lehetőségünk van definiálni függvényt. Példaként lássuk a tangens függvényt:

```
\newcommand{\tg}{\mathop{\mathrm{tg{}}}\nolimits}
```

Ezen utasítás hatására a `$\tg^2(x)$` hatása a következő lesz: $\tg^2(x)$.

Másik lehetőség a deklarációs részben definiálni a függvényt:

```
\DeclareMathOperator{\tg}{tg}
```

Harmadik lehetőség, amikor a `\tan` utasítást definiáljuk újra:

```
\renewcommand{\tan}{\mathop{\mathrm{tg{}}}\nolimits}
```

Gyökfüggvény

Gyök, illetve négyzetgyök füffvényat a `\sqrt[n]{x}` utasítással lehet. Tekintsük a következő példákat:

- $\sqrt{x}$ `\sqrt{x}`
- $\sqrt{x+\sqrt{x}}$ `\sqrt{x+\sqrt{x}}`
- $\sqrt[n]{x}$ `\sqrt[n]{x}`
- $\sqrt{x}+\sqrt{y}$ `\sqrt{x}+\sqrt{y}`
- $\sqrt{x}+\sqrt{y}$ `\sqrt{x}+\sqrt{\smash[b]y}`

Az utolsó két példa közötti különbség a $\sqrt{y}$ megjelenítésében van. Az első példában látható, hogy a két gyökjel nincs egy magasságban, nem helyes az igazítás. Ezt korrigálja a `\smash[b]` utasítás, ami az y mélységét 0-nak tekinti.

Törtek, binomiális együttható

Gyök, illetve négyzetgyök füffvényat a `\sqrt[n]{x}` utasítással lehet. Tekintsük a következő példákat:

- `\frac{számláló}{nevező}`
- `\dfrac{számláló}{nevező} = \displaystyle\frac{számláló}{nevező}`
- `\tfrac{számláló}{nevező} = \textstyle\frac{számláló}{nevező}`

A különbség a függvények között a méretezésben van. Szövegközi módban a törtek megjelenítése kisebb.
Hasonlóképpen három lehetőségünk van a binomiális együtthatók megjelenítésére is:

- `\binom{n}{k}`
- `\dbinom{n}{k} = \displaystyle\binom{n}{k}`
- `\tbinom{n}{k} = \textstyle\binom{n}{k}`

Saját törtfüggvény

Lehetőségünk van a \genfrac utasítás segítségével saját formátumú tört függvény létrehozására. Általános szintaxisa:
\genfrac{balz}{jobbz}{vastagság}{stílus}{felül}{alul}.
A paraméterek jelentése:

- balz: a tört nyitó zárójele (ha üresen marad akor nincs).
- jobbz: a tört záró zárójele (ha üresen marad akor nincs).
- vastagság: a törtvonal vastagsága, alapértelmezett a $0,4pt$.
- stílus:
  - üres: alakalmazkodik a környezethez
  - 0: \displaystyle
  - 1: \textstyle
  - 2: \scriptstyle
  - 3: \dscriptscriptstyle

Például egy zárójelezett tört:
\newcommand{\tort}[2]{\genfrac(){}{1}{#1}{#2}}
A \[\tort{x+1}{x-1}\] utasítás hatása:

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

Lánctört

Lánctörtet létrehozhatunk a törteket készítő utasítással is, azonben ekkor a törtek egyre kisebbek lesznek. Helyes megoldást a `\cfrac` utasítás adja. Általános szintaxisa:
`\cfrac[igazítás]{számláló}{nevező}`,
ahol az igazítás a számláló igazítását jelenti, alapértelmezésben középre igazít, de lehet jobbra (r) illetve balra is (l).
Tekintsük az alábbi példát kétféleképpen:

```
\frac{1}{1+\frac{1}{1+
\frac{1}{1+\dots}}}
```

$$\frac{1}{1+\frac{1}{1+\frac{1}{1+\dots}}}$$

```
\cfrac{1}{1+\cfrac{1}{1+
\cfrac{1}{1+\cdots}}}
```

$$\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cdots}}}$$

Matematikai programcsomagok
Összetett formulák készítése

Dr. Hartung Ferenc - Dr. Virágh János - Pozsgai Tamás

2014. május 2.

Mátrixok

Mátrixokat alapesetben tömbök segítségével tudunk létrehozni.

```
\[ A=
\left(
 \begin{array}{cccc}
 1 & 2 & 3 & 4 \\
 1 & 4 & 9 & 16 \\
 1 & 8 & 27 & 64 \\
 1 & 16 & 81 & 256
 \end{array}
 \right)
\]
```

$$A = \left(\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \\ 1 & 16 & 81 & 256 \end{array} \right)$$

Mátrixok

Az amsmath csomag egyszerűsíti a mátrixok létrehozását a következő utasításokkal:

$$\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}$$ `\begin{matrix} 1&2\\ 3&4 \end{matrix}`

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$ `\begin{pmatrix} 1&2\\ 3&4 \end{pmatrix}`

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$ `\begin{bmatrix} 1&2\\ 3&4 \end{bmatrix}`

$$\begin{Bmatrix} 1 & 2 \\ 3 & 4 \end{Bmatrix}$$ `\begin{Bmatrix} 1&2\\ 3&4 \end{Bmatrix}`

$$\begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix}$$ `\begin{vmatrix} 1&2\\ 3&4 \end{vmatrix}`

$$\begin{Vmatrix} 1 & 2 \\ 3 & 4 \end{Vmatrix}$$ `\begin{Vmatrix} 1&2\\ 3&4 \end{Vmatrix}`

Mátrixok

Normál szövegben a $\left(\begin{smallmatrix}1 & 2\\ 3 & 4\end{smallmatrix}\right)$ szebb, mint a $\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$. Ez az úgynevezett \smallmatrix környezet.

```
$\left(
\begin{smallmatrix}
1 & 2\\
3 & 4
\end{smallmatrix}
\rigth)$
```

Egységmátrix

Általános $n \times n$-es mátrix definiálása:

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}$$

```
\begin{pmatrix}
1 & 0 & \ldots & 0 \\
0 & 1 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 1
\end{pmatrix}
```

Mátrixok

Abban az esetben, mikor egynél több oszlopban szeretnénk pontsorozattal kitölteni, a három pont egymás utáni elhelyezése nem ad szép képet. Helyette használható a `\hdotsfor[d]{oszlopszám}`, ahol d a pontok sűrűsége, oszlopszám pedig a pontozott oszlopok száma.

$$\begin{pmatrix} 1 & 2 & 3 & \dots\dots\dots\dots & n \\ 2 & 3 & \dots\dots\dots & n & n+1 \\ \vdots & \vdots & & \vdots & \vdots \\ n & \dots. & 2n-2 & 2n-1 & 2n \end{pmatrix}$$

```
\begin{pmatrix}
1 & 2 & 3 & \hdotsfor{2} & n\\
2 & 3 & \hdotsfor{2} & n & n+1\\
\vdots & \vdots & & & \vdots\\
n& \hdotsfor{2}  & 2n-2 & 2n-1 & 2n
\end{pmatrix}
```

Vektorok

A vektorok, mint speciális mátrixok hasonló módon jeleníthetők meg. Természetesen egy oszlopuk, illetve egy soruk van.

Sorvektor $\begin{pmatrix} 1 & 2 & 3 & \ldots & n \end{pmatrix}$
Forráskódja:

```
$\begin{pmatrix}
1 & 2 & 3 & \ldots & n
\end{pmatrix}$\\
```

Oszlopvektor $\begin{pmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ n \end{pmatrix}$

Forráskódja:

```
$\begin{pmatrix}
1 \\ 2 \\ 3 \\ \vdots \\ n
\end{pmatrix}$\\
```

Képletek törése

Hosszú képleteket ha több sorba szeretnénk szedni, akkor a `multiline` környezetet kell használnunk. Amennyiben a képletet nem akarjuk számozni, akkor a `multiline*` környezetet.

```
\begin{multline*}\label{címke}
1+8+27=\\
=1+3+5+7+{}\\
+9+11
\end{multline*}
```

$$1+8+27= \\ =1+3+5+7+ \\ +9+11$$

Képletek törése

Másik lehetőségünk a `split` környezet.

```
\begin{equation*}
\begin{split}
1+8+27=\\
=1+3+5+7+{}\\
+9+11\end{split}
\end{equation*}
```

$$\begin{aligned}1+8+27&=\\ =1+3+5+7&+\\ +9&+11\end{aligned}$$

Képletek egymás alatt

Egymás alá írt képletek írására a következő lehetőségeink vannak:

- `gather` illetve `gather*`
- `gathered[poz]`
- `align` illetve `align*`
- `aligned[poz]`
- `flalign` illetve `flalign*`
- `alignat{n}` illetve `alignat*{n}`
- `alignedat[poz]{n}` illetve `alignedat[poz]{n}*`

Ahol a `[poz]` lehet `t` vagy `b` (top vagy bottom), `n` pedig az oszloppárok száma (egymás mellé illesztett képletek leírására használjuk).

Gather* környezet

Ebben a környezetben igazítás nélkül adhatunk meg több képletet. A képleteket középre igazítja.

```
\begin{gather*}
x+y=6\\
x^2+2xy+y=16
\end{gather*}
```

$$x + y = 6$$
$$x^2 + 2xy + y = 8$$

Gathered környezet

Több sorból álló részformulák egymás mellé igazítását tudjuk elvégezni vele.

```
\[
\begin{gathered}
x+y=6\\
x^2+2xy+y=28
\Rightarrow
\begin{gathered}
x=2\\
y=4
\end{gathered}
\]
```

$$\begin{gathered} x+y=6 \\ x^2+2xy+y=28 \end{gathered} \Rightarrow \begin{gathered} x=2 \\ y=4 \end{gathered}$$

align* környezet

Egymás alá igazítottan elhelyezni képleteket az `align` illetve az `align*` környezetekben tudjuk megtenni.

```
\begin{align*}
1+8+27&=\\
&=1+3+5+7+9+11\\
&=36
\end{align*}
```

$$\begin{aligned}
1+8+27 &= \\
&= 1+3+5+7+9+11 \\
&= 36
\end{aligned}$$

align* környezet

Átalakítás sorozat mellé megjegyzések írására is használható ez a környezet.

```
\begin{align*}
x_1^2-1&=x_2^2-1 && \text{/ +1}\\
x_1^2&=x_2^2 && \text{/ gyököt vonunk}\\
|x_1|&=|x_2| && \text{/ mivel }x_1;x_2>0\\
x_1&=x_2
\end{align*}
```

$$
\begin{aligned}
x_1^2-1&=x_2^2-1 && / +1\\
x_1^2&=x_2^2 && / \text{ gyököt vonunk}\\
|x_1|&=|x_2| && / \text{ mivel } x_1;x_2>0\\
x_1&=x_2
\end{aligned}
$$

aligned környezet

Hasonlóan működik, mint az `align*` környezet,azonban szövegközi módban is használható. A `poz` paraméter a szöveghez való illesztést adja (alapértelmezetten középre igazít).

```
\[
\left.\begin{aligned}
x-y&=y+3\\
x+y&=6
\end{aligned}\right\}
\text{ ekkor }
\left.\begin{aligned}
x&=5\\
y&=1
\end{aligned}\right\}
\]
```

$$\left.\begin{aligned} x-y&=y+3\\ x+y&=6 \end{aligned}\right\} \text{ ekkor } \left.\begin{aligned} x&=5\\ y&=1 \end{aligned}\right\}$$

flalign* környezet

Hasonlóan működik az `align` illetve `align*` környezetekhez, a különbség az, hogy az első oszlopot balra, az utolsót jobbra igazítja.

```
\begin{flalign*}
x_1^2-1&=x_2^2-1 && \text{/ +1}\\
x_1^2&=x_2^2 && \text{/ gyököt vonunk}\\
|x_1|&=|x_2| && \text{/ mivel }x_1;x_2>0\\
x_1&=x_2
\end{flalign*}
```

$$x_1^2 - 1 = x_2^2 - 1 \qquad / +1$$
$$x_1^2 = x_2^2 \qquad / \text{ gyököt vonunk}$$
$$|x_1| = |x_2| \qquad / \text{ mivel } x_1; x_2 > 0$$
$$x_1 = x_2$$

alignat* környezet

Lineáris egyenletrendszerket tudunk ezekkel a környezetekkel szépen megjeleníteni. Paraméterként az oszlopok számát kell megadni.

```
\begin{alignat*}{4}
3x_1+{} & & 12x_3 &={}&9\\
x_1-{}  & 12x_2 & &={}&14
\end{alignat*}
```

$$\begin{alignat*}{4}
3x_1+{} & & 12x_3 &={}&9\\
x_1-{} & 12x_2 & &={}&14
\end{alignat*}$$

alignedat környezet

Az `alignat` illetve `alignat*` környezetek részformulaképző környezete, ami szövegközi módban is használható.

```
Oldja meg a $\begin{alignedat}{4}
3x_1+{} & & 12x_3 &={}&9\\
x_1-{}  & 12x_2 & &={}&14
\end{alignedat}$ egyenletrendszert!
```

Oldja meg a $\begin{alignedat}{4} 3x_1+{} & & 12x_3 &={}&9\\ x_1-{} & 12x_2 & &={}&14 \end{alignedat}$ egyenletrendszert!

Szöveg többsoros képletek között

Az egymás alá elhelyezett képletek közé magyarázó szöveget helyezhetünk el az `\intertext` környezetben. A következő környezetekben használható:

- `align`
- `align*`
- `flalign`
- `flalign*`
- `alignat`
- `alignat*`

Szöveg többsoros képletek között

Az `alignat` illetve `alignat*` környezetek részformulaképző környezete, ami szövegközi módban is használható.

```
\begin{align*}
\dfrac{x}{x-2} &=\dfrac{x-2+2}{x-2}= \\
\intertext{Ekkor már látható, milyen
          függvénytranszformációkat kell végrehajtani}
&=1+\dfrac{2}{x-2}
\end{align*}
```

$$\frac{x}{x-2} = \frac{x-2+2}{x-2} =$$

Ekkor már látható, milyen függvénytranszformációkat kell végrehajtani

$$= 1 + \frac{2}{x-2}$$

Táblázat

Matematikai módban a `tabular` környezet helyett használjuk az `array` környezetet.

```
\[\begin{array}{|c|c|}
\hline
a<x<b & b\ge x\\
\hline
\nearrow & \searrow\\
\hline
\end{array}
\]
```

$a < x < b$	$b \geq x$
$\nearrow$	$\searrow$

Mintafeladatok

Legyen $f,g \in \mathbb{R} \to \mathbb{R}$, $f(x) = \sqrt{2x - x^2}$ és $g(x) = \frac{1}{x-3}$. Adja meg az $f \circ g$ függvényt, ha létezik.

Megoldás

Az $f \circ g$ létezik, ha $dom(f \circ g) \neq \emptyset$

$$x \in dom(g) = \mathrm{R}\backslash\{-3\}$$
$$dom(f) = [0;2]$$
$$g \in dom(f)$$
$$0 \leq \frac{1}{x-3} \leq 2$$
$$2,5 \leq x$$
$$dom(f \circ g) = [2,5;\infty[$$

A hozzárendelési utasítás:

$$f \circ g = \sqrt{\frac{2}{x-3} - \frac{1}{(x-3)^2}}$$

Mintafeladatok

Megoldás forráskódja

```
Az $f\circ g$ létezik, ha $dom(f\circ g)\neq \emptyset$
\begin{align*}
x \in dom(g)=\mathrm{R}\backslash \{-3\}\\
dom(f)=[0;2]\\
g \in dom(f)\\
0\leq \frac{1}{x-3} \leq 2\\
2,5\leq x\\
dom(f\circ g)=[2,5;\infty[
\intertext{A hozzárendelési utasítás:}
f\circ g=\sqrt{\dfrac{2}{x-3}-\dfrac{1}{(x-3)^2}}
\end{align*}
```

Mintafeladatok

Határozza meg az $a_n = \left(\frac{n+3}{n-4}\right)^n$ sorozat határértékét:

Megoldás

$$
\begin{aligned}
a_n = \left(\frac{n+3}{n-4}\right)^n &= \\
&= \left(\frac{n\left(1+\frac{3}{n}\right)}{n\left(1-\frac{4}{n}\right)}\right)^n = \\
&= \left(\frac{\left(1+\frac{3}{n}\right)}{\left(1-\frac{4}{n}\right)}\right)^n = \\
&= \frac{\left(1+\frac{3}{n}\right)^n}{\left(1-\frac{4}{n}\right)^n} \to \frac{\mathrm{e}^3}{\mathrm{e}^{-4}} = \mathrm{e}^7
\end{aligned}
$$

Mintafeladatok

Megoldás forráskódja

```
\begin{align*}
a_n&=\left(\dfrac{n+3}{n-4}\right)^n=\\
&=\left(\dfrac{n\left(1+\frac{3}{n}\right)}
      {n\left(1-\frac{4}{n}\right)}\right)^n=\\
&=\left(\dfrac{\left(1+\frac{3}{n}\right)}
      {\left(1-\frac{4}{n}\right)}\right)^n=\\
&=\dfrac{\left(1+\frac{3}{n}\right)^n}
      {\left(1-\frac{4}{n}\right)^n}\rightarrow
      \dfrac{\mathrm{e}^3}{\mathrm{e}^{-4}}=\mathrm{e}^7
\end{align*}
```

Mintafeladatok

Határozza meg az $\lim\limits_{x\to 3} \frac{x-3}{x-\sqrt{x+6}}$ határértéket:

Megoldás

$$A = \lim_{x\to 3} \frac{x-3}{x-\sqrt{x+6}} = \qquad \text{bővítsük a törtet}$$

$$= \lim_{x\to 3} \frac{(x-3)(x+\sqrt{x+6})}{(x+\sqrt{x+6})(x-\sqrt{x+6})} = \qquad \text{felbontva a zárójelet}$$

$$= \lim_{x\to 3} \frac{x^2-3x+x\sqrt{x+6}-3\sqrt{x+6}}{x^2-x-6} = \qquad \text{rendezve a számlálót}$$

$$= \lim_{x\to 3} \frac{x^2+x(\sqrt{x+6}-3)-3\sqrt{x+6}}{x^2-x-6} = \qquad \text{kiemelve } x^2\text{-et}$$

$$= \lim_{x\to 3} \frac{x^2\left(1+\frac{\sqrt{x+6}-3}{x}-\frac{3\sqrt{x+6}}{x^2}\right)}{x^2\left(1-\frac{1}{x}-\frac{6}{x^2}\right)} = \qquad \text{egyszerűsítve}$$

$$= \lim_{x\to 3} \frac{1+\frac{\sqrt{x+6}-3}{x}-\frac{3\sqrt{x+6}}{x^2}}{1-\frac{1}{x}-\frac{6}{x^2}} = 1$$

Mintafeladatok

Megoldás forráskódja

```
\begin{align*}
A&=\lim_{x\to 3}\frac{x-3}{x-\sqrt{x+6}}= &
   \text{bővítsük a törtet}\\
&=\lim_{x\to 3}\frac{(x-3)(x+\sqrt{x+6})}
  {(x+\sqrt{x+6})(x-\sqrt{x+6})}= &
   \text{febontva a zárójelet}\\
&=\lim_{x\to 3}\frac{x^2-3x+x\sqrt{x+6}-3
   \sqrt{x+6}}{x^2-x-6}= &
    \text{rendezve a számlálót}\\
&=\lim_{x\to 3}\frac{x^2+x(\sqrt{x+6}-3)-
   3\sqrt{x+6}}{x^2-x-6}= &
   \text{kiemelve }x^2\text{-et}\\
&=\lim_{x\to3}\frac{x^2\left(1+\frac{\sqrt{x+6}-3}{x}-
  \frac{3\sqrt{x+6}}{x^2}\right)}{x^2\left(1-
  \frac1x-\frac{6}{x^2}\right)}= & \text{egyszerűsítve}\\
&=\lim_{x\to 3}\frac{1+\frac{\sqrt{x+6}-3}{x}-
  \frac{3\sqrt{x+6}}{x^2}}{1-\frac1x-\frac{6}{x^2}}= 1
\end{align*}
```

Mintafeladatok

Adja meg az $f(x) = \frac{\ln(x^2)}{e^{-\sin(2x)}}$ függvény deriváltját!

Megoldás

$$\frac{df(x)}{dx} = \frac{\left(\frac{2x}{x^2}\right) \cdot e^{-\sin(2x)} - \ln\left(x^2\right) e^{-\sin(2x)} \left(-\cos(2x) \cdot 2\right)}{\left(e^{-\sin(2x)}\right)^2} =$$

$$= \frac{e^{-\sin(2x)} \left(\left(\frac{2}{x}\right) - \ln\left(x^2\right) \left(-2\cos(2x)\right)\right)}{e^{-2\sin(2x)}} =$$

$$= \frac{\left(\frac{2}{x}\right) + 2\ln\left(x^2\right)\cos(2x)}{e^{-\sin(2x)}}$$

Mintafeladatok

Megoldás forráskódja

```
\begin{align*}
\frac{\mathrm{d}f(x)}{\mathrm{d}x} &=
\dfrac{\left(\frac{2x}{x^2}\right)\cdot
  \mathrm{e}^{-\sin(2x)}-\ln\left(x^2\right)
  \mathrm{e}^{-\sin(2x)}\left(-\cos(2x)\cdot
 2\right)}{\left(\mathrm{e}^{-\sin(2x)}\right)^2}=\\
&=\dfrac{\mathrm{e}^{-\sin(2x)}\left(\left(\frac{2}{x}
  \right)-\ln\left(x^2\right)\left(-2\cos(2x)
  \right)\right)}{\mathrm{e}^{-2\sin(2x)}}=\\
&=\dfrac{\left(\frac{2}{x}\right) +2\ln\left(x^2\right)
\cos(2x)}{\mathrm{e}^{-\sin(2x)}}
\end{align*}
```

Mintafeladatok

Végezze el az $f(x) = x^3 + 2x^2 - x - 2$ függvény teljes vizsgálatát!

Megoldás

- Értelmezési tartomány: $dom(f) = \mathbb{R}$
- Zérushelyek:

$$f(x) = x^3 + 2x^2 - x - 2 = (x-1)(x+1)(x+2)$$

$$\Downarrow$$

$$x_0 = 1,\ x_1 = -1,\ x_2 = -2$$

Mintafeladatok

Megoldás folytatása

- Határértékszámítás $-\infty$-ben és ∞-ben:

$$\lim_{x \to -\infty} x^3 + 2x^2 - x - 2 = -\infty$$

$$\lim_{x \to \infty} x^3 + 2x^2 - x - 2 = \infty$$

Mintafeladatok

Megoldás folytatása

- Menettulajdonságok az első deriált segítségével:

$$f'(x) = 3x^2 + 4x - 1 = 0$$

$$\Downarrow$$

$$x_1 = -\frac{2}{3} + \frac{\sqrt{7}}{3} \approx -1,549, \quad x_2 = -\frac{2}{3} - \frac{\sqrt{7}}{3} \approx 0,215$$

	$x < x_1$	$x = x_1$	$x_1 < x < x_2$	$x = x_2$	$x < x_2$
$f'(x)$	$+$	0	$-$	0	$+$
$f(x)$	$\nearrow$	max	$\searrow$	min	$\nearrow$

Mintafeladatok

Megoldás folytatása

- Görbületi viszonyok a második deriált segítségével:

$$f''(x) = 6x + 4 = 0$$

$$\Downarrow$$

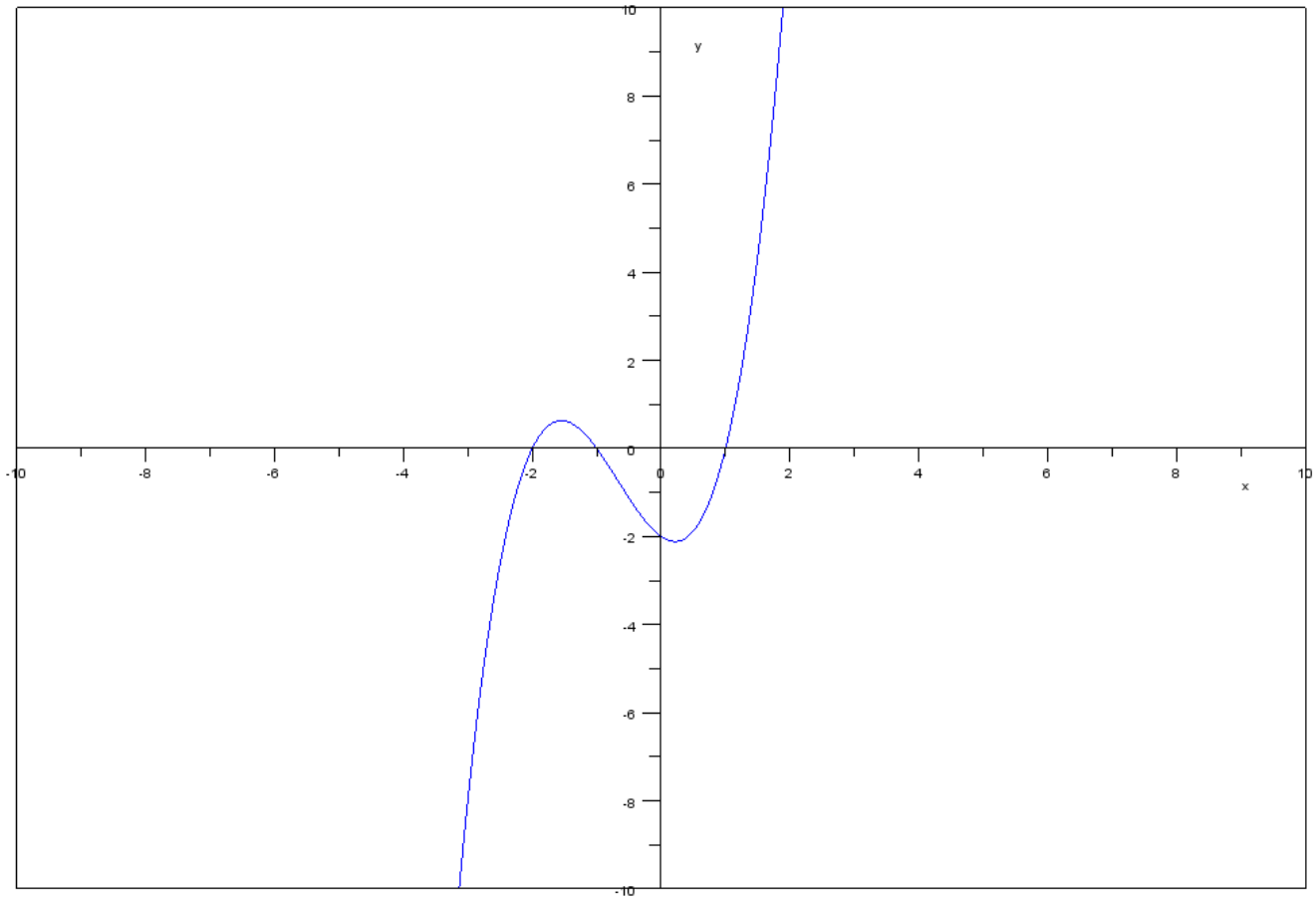
$$x_3 = -\frac{2}{3}$$

	$x < x_3$	$x = x_3$	$x_3 < x$
$f''(x)$	$+$	0	$-$
$f(x)$	$\cup$	inf. p.	$\cap$

Mintafeladatok

Megoldás folytatása

- A függvény grafikonja:



Mintafeladatok

Megoldás forrásrészlete (menettulajdonságok)

```
\begin{gather*}
  f'(x)=3x^2+4x-1=0\\
\Downarrow\\
x_1=-\dfrac{2}{3}+\dfrac{\sqrt{7}}{3}\approx -1,549,\
x_2=-\dfrac{2}{3}-\dfrac{\sqrt{7}}{3}\approx 0,215
\end{gather*}
\[ \begin{array}{|c|c|c|c|c|c|}
\hline
 & x<x_1 & x=x_1 & x_1<x<x_2 & x=x_2 & x<x_2\\
\hline f'(x) & + & 0 & - & 0 & +\\
\hline
f(x) & \nearrow & \text{max} & \searrow
& \text{min} & \nearrow \\
\hline
\end{array}
\]
```

Mintafeladatok

Adja meg a $f(x) = \dfrac{3}{x \ln^2(x)}$ függvény összes primitív függvényét!

Megoldás

$$\int \frac{3}{x \ln^2(x)} \, \mathrm{d}x = \int \frac{3}{x} \ln^{-2}(x) \, \mathrm{d}x =$$

$$= -3 \ln^{-1}(x) = \frac{-3}{\ln(x)} + C$$

Mintafeladatok

Megoldás forrása

```
\begin{align*}
\int \dfrac{3}{x\ln^2(x)}\ \mathrm{d}x &=
     \int \dfrac{3}{x}\ln^{-2}(x)\ \mathrm{d}x=\\
&=-3\ln^{-1}(x) = \dfrac{-3}{\ln(x)} + C
\end{align*}
```

Mintafeladatok

Számolja ki az $y = x^2 - 2x + 2$ és az $y = 14 - x^2$ egyenletű görbék által közrezárt korlátos síkidom területét!

Megoldás

- Metszéspontok megkeresése:

$$x^2 - 2x + 2 = 14 - x^2$$
$$2(x^2 - x - 6) = 0$$
$$\Downarrow$$
$$x_1 = -3 \; x_2 = 2$$

- A terület kiszámítása:

$$\int_{-3}^{2} -2(x^2 - x - 6)\,\mathrm{d}x = \left[-\frac{2}{3}x^3 + x^2 + 12x\right]_{-3}^{2} = \frac{95}{3}$$

Mintafeladatok

Megoldás forrása

```
Metszéspontok megkeresése:\\
\begin{gather*}
x^2-2x+2=14-x^2\\
2(x^2-x-6)=0\\
\Downarrow\\
x_1=-3 \ x_2=2
\end{gather*}
A terület kiszámítása:\\
\[
\int_{-3}^2 -2(x^2-x-6)\ \mathrm{d}x
= \left[-\dfrac{2}{3}x^3+x^2+12x\right]_{-3}^2=
\dfrac{95}{3}
\]
```

Mintafeladatok

Számolja ki az $\int_1^3 \frac{1}{\sqrt[3]{x-1}}\,\mathrm{d}x$ értéket!

Megoldás

$$\int_1^3 \frac{1}{\sqrt[3]{x-1}}\,\mathrm{d}x = \lim_{t\to 1}\int_t^3 (x-1)^{-\frac{1}{3}}\,\mathrm{d}x =$$

$$= \lim_{t\to 1}\left[\frac{(x-1)^{\frac{2}{3}}}{\frac{2}{3}}\right]_t^3 =$$

$$= \lim_{t\to 1}\left(\frac{3}{2}\left((3-1)^{\frac{2}{3}} - (t-1)^{\frac{2}{3}}\right)\right) = \left(\frac{3}{2}\right)\sqrt[3]{4}$$

Mintafeladatok

Megoldás forrása:

```
Számolja ki az $\int\limits_1^3
    \dfrac{1}{\sqrt[3]{x-1}}\ \mathrm{d}x$ értéket!\\
\textbf{Megoldás}
\begin{align*}
\int_1^3 \dfrac{1}{\sqrt[3]{x-1}}\ \mathrm{d}x &=
  \lim_{t \to 1}\int_t^3 (x-1)^{-\frac{1}{3}}\ \mathrm{d}x=\\
&=\lim_{t \to 1}\left[\dfrac{(x-1)^{\frac{2}{3}}}{\frac{2}{3}}\right]_t^3 =\\
&=\lim_{t \to 1} \left( \dfrac{3}{2}\left( (3-1)^{\frac{2}{3}}-
                        (t-1)^{\frac{2}{3}}\right)\right)
 =\left(\dfrac{3}{2}\right) \sqrt[3]{4}
\end{align*}
```

Matematikai programcsomagok
Ábrák, grafikák

Dr. Hartung Ferenc - Dr. Virágh János - Pozsgai Tamás

2014. május 2.

Képek beszúrása

Képek beszúrásához töltsük be a `graphicx` csomagot. A képek helyét megadhatjuk relatívan (ekkor a gyökérkönyvtár a fő dokumentum könyvtára), de megadhatjuk abszolút módon is.

- `\includegraphics{abra.jpg}`
- `\includegraphics{D:/kepek/abra.jpg}`

Abban az esetben, amikor minden képet egy könyvtárban helyezünk el, akkor ezt a deklarációs részben megadhatjuk (ekkor a relatív hivatkozás ebből a könyvtárból indul).
`\graphicspath{{./kepek/}}`

includegraphics opciói

- trim - a képet nem az ereteti mérethez, hanem a megadott kerethez igazítja.
- clip - trimmel együtt használva a megadott keret része jelenik meg a képnek.
- scale - Nagyítás/kicsinyítés mértéke.
- angle - Forgatás szöge fokban megadva.
- origin - A forgatás középpontja.
- width - A kép szélessége.
- height - A kép magassága.
- page - Az oldal kiválasztása többboldalas pdf fájl esetén.

Képek úsztatása

Képek elhelyezése a dokumentumban sok esetben nem úgy történik, ahogyan szeretnénk. Ezen segít a `figure` környezet.

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

Alapértelmezésben a képet megpróbálja a fordító beilleszteni az oldal tetejére, ha nem sikerül, akkor az oldal aljára, ha oda sem lehet, akkor külön oldalra (`[tbp]` paraméterezés).

Képek úsztatása

Amennyiben a képet ott szeretnénk megjeleníteni, ahová a forrásfájlban elhelyeztük, akkor a `[h]` paramétert használjuk. Néhány alapelv az úsztatásról:

- Mindig a legelső helyre kerül, ahová a paraméterezés megengedi.
- Nem kerülhet a definiálását megelőző oldara.
- Nem okozhat oldaltúlcsordulást.
- Paraméterként használt felkiáltójel használatakor bizonyos korlátozások felfüggesztődnek.
- `\suppressfloats[tb]` utasítás hatására a megadott helyen nem jelenhet meg a kép. Csak oldal teteje vagy alja lehet a paramétere `[tb]`.

Képaláírás

Képekre hivatkozhatunk, jegyzéket készíthetünk belőlük, ehhez azonban szükségünk van arra, hogy megcimkézzük őket.

```
\begin{figure}
   \centering
   \includegraphics{kép}
   \caption{ábra neve.}\label{rnev}
   \label{fig:minta}
\end{figure}
```

Ekkor a kép kap egy autómatikus sorszámot, melyre hivatkozhatunk a későbbiekben a \ref és a \pageref utasításokkal.

Grafika készítés

Ábrák, grafikák készítésére több csomag is elérhetp a LaTeX-ben.

- `picture`
- `pict2e`
- `pstricks`
- `emp`
- `musictex`

Ezek közül a `picture`, a `pict2e` és a `pstricks` csomagokat mutatjuk be.

Ábrákról általában

- Ne használjunk túl sok különböző vastagságú vonalat.
- Ügyeljünk arra, hogy a betűtipus ne üssön el a szöveg tipusától.
- Az ábra betűmérete általában kisebb, mint a szövegé.
- Használjuk a `\footnotesize` vagy a `\small` környezetet.
- Ne keltsen az ábra túlzsúfolt benyomást.

Picture környezet

Használatához a deklarációs részben be kell töltenünk a picture csomagot.
A grafikai elemek leírásához a parancsok minden eseben egy rögzített koordináta rendszert használnak.
A grafikai környezet alapegységét a `\unitlength` utasítással tudjuk beállítani. Alapértéke 1pt.
A picture környezet a következő paraméterekkel használható:

```
\begin{picture}(x,y)(a,b)
```

ahol x és y az ábra mérete (egy $x \times y$ méretű doboz). Az (a, b) paraméter a doboz bal alsó sarkának koordináta értékei. Alapértéke a $(0, 0)$.

Put, multiput utasítások

Szintaxisa:
`\put(x,y){k}`
A parancs hatására a "k" elem referenciapontja a doboz (x, y) pontjára kerül. Az elem lehet szöveg is, illetve valamilyen képelem is.
`\multiput(x,y)(a,b){n}{k}`
Az előzőhöz hasonló módon elhelyezi a "k" elemet a dobozban, de most összesen n-szer, az egymást követő elemeket pedig (a, b) vektorral tolja el az előzőhöz képest.

Példa

Szintaxisa:

`\put(x,y){k}`

A parancs hatására a "k" elem referenciapontja a doboz (x, y) pontjára kerül. Az elem lehet szöveg is, illetve valamilyen képelem is.

`\multiput(x,y)(a,b){n}{k}`

Az előzőhöz hasonló módon elhelyezi a "k" elemet a dobozban, de most összesen n-szer, az egymást követő elemeket pedig (a, b) vektorral tolja el az előzőhöz képest.

Qbezier utasítás

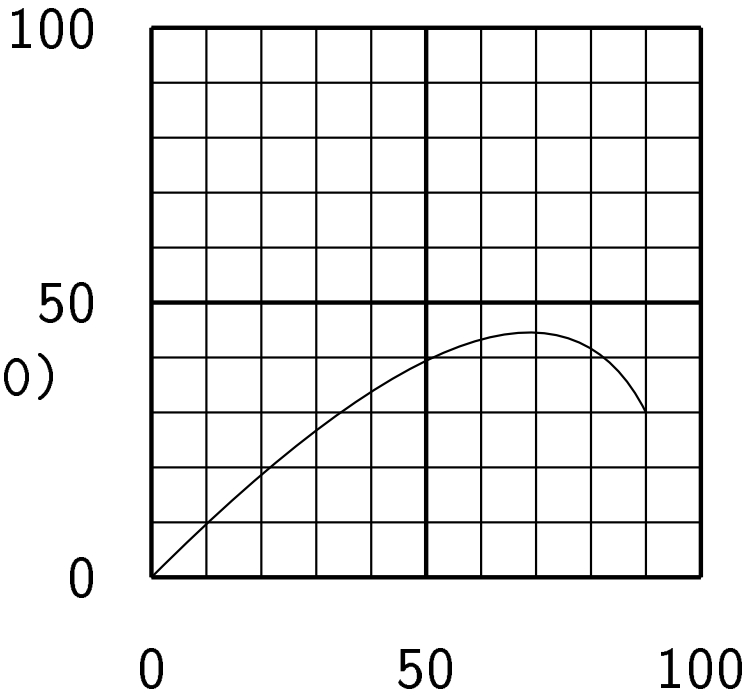
Három megadott pont által meghatározott kvadratikus Bezier görbét rajzol.
Szintaxisa:
`\qbezier[p]`$(x_1, y_1)(x_2, y_2)(x_3, y_3)$
Ha a "p" paraméter meg van adva, akkor a görbének csak ennyi pontját rajzolja ki. Ha nincs megadva, akkor folytonos vonal lesz a görbe.

Graphpaper utasítás

Adott méretű berácsozott koordináta-rendszert rajzol. A graphpap csomag része.
Szintaxisa:
`\graphpaper[l](a,b)(x,y)`
Az "l" paraméter adja meg a rácsvonalak közötti lépésközt, ha nincs megadva, akkor 10 az alapértelmezett. Az (a, b) paraméter adja meg a koordináta-rendszer bal alsó sarkának a koordinátáját, az (x, y) pedig a koorodináta-rendszer méretét.

Példa

```
\begin{picture}(100,100)
\graphpaper(0,0)(100,100)
\qbezier(0,0)(70,70)(90,30)
\end{picture}
```



Line és vector utasítások

A \line utasítás egy szakaszt, a \vector utasítás egy vektort rajzol. Szintaxisuk:

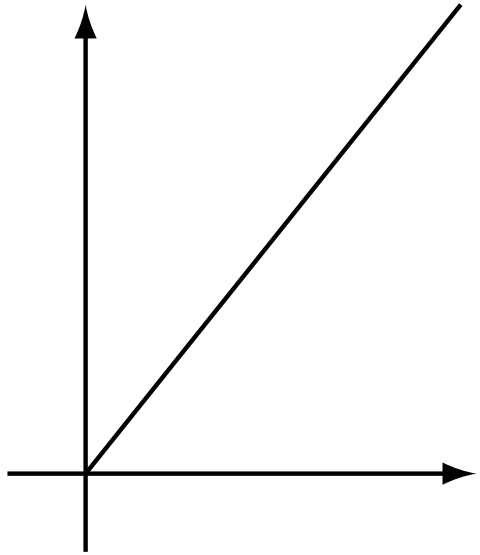
```
\line(x,y),{v}
\vector(x,y),{v}
```

Az (x, y) paraméter a szakasz illetve vektor irányvektora, a "v" paraméter pedig a vizszintes vetület hosszát adja meg. Ezalól a $x = 0$ (függőleges vonal) érték a kivétel, ekkor 'v' a szakasz illetve vektor hosszát adja meg. Az irányvektor értékei csak egészek lehetnek.

A szakasz (vektor) kezdőpontja a \put utasításban megadott érték lesz.

Példa

```
\unitlength 1mm
\begin{picture}(30,35)(-5,-5)
 \thicklines
 \put(-5,0){\vector(1,0){30}}
 \put(0,-5){\vector(0,1){35}}
 \put(0,0){\line(4,5){24}}
\end{picture}
```



Circle és circle* utasítások

A `\circle` utasítás egy üres kört, a `\circle*` utasítás egy tele kört rajzol.
Szintaxisuk:
`\circle{d}`
`\circle*{d}`
A "d" paraméter a kör átmérőjét adja meg. Referenciapontja a kör középpontja.

Oval utasítás

Az `\oval` utasítás egy lekerekített sarkú téglalpot rajzol.
Szintaxisa:
`\oval(x,y)[p]`
Az (x, y) paraméter a téglalap méretét adja meg. A "p" paraméterrel megadható, hogy a téglalap melyik részei (fele, negyede) legyen kirajzolva. Értkei a következők lehetnek:

- t - felső fele
- b - alsó fele
- l - bal fele
- r - jobb fele

Ezek kombinációja adja a negyed téglalap rajzolását (tl a bal felső például).

Szövegdoboz elhelyezése az ábrán

Szövegdoboz elhelyezésére három lehetőséget ad a picture környezet:

- `\makebox{x,y},[p]{szoveg}` - nem keretezi be
- `\framebox{x,y},[p]{szoveg}` - bekeretezi
- `\dashbox{x,y},[p]{szoveg}` - szaggatott vonallal keretezi be

Mindhárom utasítás egy (x, y) méretű dobozt helyez el. A "p" paraméter a szöveg igazítását vezérli (értékei lehetnek t,b,l,r és ezek kombinációi).

Vonalvastagság

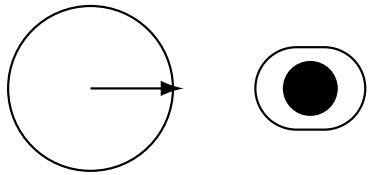
Az alábbi, vonalvastagságot beállító utasítások általános utasítások, de használhatók a picture környezeten belül is.

- `\thinlines` - vékony vonal
- `\thicklines` - vastag vonal
- `\linethickness{d}` - adott vastagságú vonal vonalat rajzol

A harmadik esetben a vonal vastagsága csak vízszintes illetve függőleges vonalak esetén változik.

Példa

```
\begin{picture}(100,100)
  \put(10,30){\vector(1,0){17}}
  \put(10,30){\circle{30}}
  \put(50,30){\circle*{10}}
  \put(50,30){\oval(20,15)}
\end{picture}
```



Pict2e környezet

A picture környezet továbbfejlesztése. A már meglévő elemeket kibővítették, emellett újabb utasítások is belekerültek.
Használatához a deklarációs részben be kell töltenünk a pict2e csomagot. A környezet betölthető a

```
\begin{pict2e}(x,y)(a,b)
```

utasítással, ahol x és y az ábra mérete (egy $x \times y$ méretű doboz). Az (a, b) paraméter a doboz bal alsó sarkának koordináta értékei. Alapértéke a $(0, 0)$.

Pict2e kibővítések

A picture környezetben meglévő utasítások mindegyike bővült, elsősorban a megadható paraméterek nagyságrendje lett nagyobb.

- `\qbezier` - a rajzolandó pontok számára nincs felső határ.
- `\line` és `\vector` - a vonalak vastagságának, hosszának beállítási lehetőségei bővültek.
- `\circle` és `\circle*` - nagyobb körök illetve lemezek rajzolhatók.
- `\oval[rad]` - a rad paraméterrrel a lekerekítés mértéke szabályozhatóvá válik.

Arc és arc* utasítások

Hasonlóan a `\circle` és a `\circle*` utasításokhoz, a `\arc` egy körívet rajzol.
Szintaxisuk:
`\arc[a,b]{d}`
`\arc*[a,b]{d}`
A "d" paraméter a körív átmérőjét adja meg. Referenciapontja az ív középpontja. Az (a, b) paraméter "a" értéke a körív kezdőpontja, "b" pedig a végpontja.

Vonalak, sokszögek

Szakaszok és sokszögek ajzolására új utasítások is bekerültek a csomagba.

- `\Line(x1,y1)(x2,y2)` - a két pont közötti szakaszt rajzolja meg.
- `\polyline(x1,y1)(x2,y2)...(xn,yn)` - a megadott pontokat köti össze.
- `\polygon(x1,y1)(x2,y2)...(xn,yn)` - sokszöget rajzol a megadott pontok alapján.
- `\polygon*(x1,y1)(x2,y2)...(xn,yn)` - kitöltött sokszöget rajzol.

Alakzatok

Az alábbi utasítások segítségével tudunk alakzatokat rajzolni:

- `\moveto(x,y)` - a görbe kezdőpontját határozza meg.
- `\lineto(x,y)` - kezdőpontból a megadott pontig egyenest rajzol.
- `\curveto(x2,y2)(x3,y3)(x4,y4)` - négyzetes Bezier görbét rajzol (két kontrolpont és a végpont a három paraméter).
- `\circlearc[n]{x}{y}{d}{r1}{r2}` - körívet rajzol az alábbi paraméterekkel:
  - n - pozícionálást segítő elhagyható paraméter (alapértéke 0).
  - x,y - a középpont koordinátái.
  - d - körív sugara.
  - r1,r2 - a körív kezdő szöge, illetve végszöge.

Alakzatok

Az alakzat lezárását a következő utasításokkal tudjuk megtenni:

- `\closepath` - a görbe kezdőpontjába.
- `\strokepath` - megrajzolja a definiált görbét.
- `\fillpath` - megrajzolja a definiált zárt görbét.

Alakzatok

Az alakzat lezárásának módját tudjuk vezérelni a következő utasításokkal:

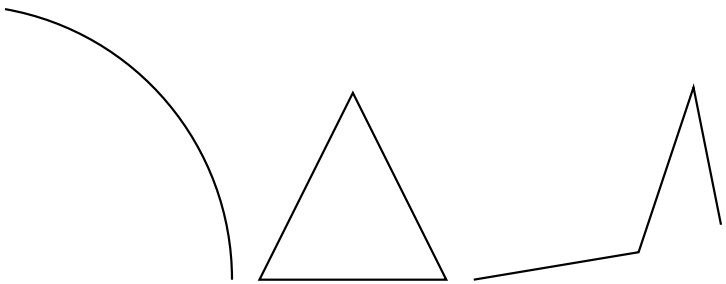
- `\buttcap` - egyenes lezárás.
- `\roundcap` - félkör lezárás.
- `\squarecap` - fél négyzet lezárás.

Az alakzat összefűzésének módját tudjuk vezérelni a következő utasításokkal:

- `\mitterjoin` - egyenes lezárás.
- `\roundjoin` - félkör lezárás.
- `\beveljoin` - fél négyzet lezárás.

Példa

```
\begin{picture}(100,100)
  \put(6,6){\arc[80,0]{50}}
  \put(55,0){\polygon(6,6)(40,6)(23,40)}
  \put(100,6){\polyline(0,0)(30,5)(40,35)(45,10)}
\end{picture}
```



Pstricks csomag

A pstricks egy makrócsomag, ami a legtöbb TEX csomaggal kompatibilis. Segitségével könnyebben tudunk grafikai elemeket elhelyezni a dokumentumunkban. Emellett egyszerűen lehet szineket használni, elforgatni elemeket. Használata:

```
{\red Ez a szöveg piros lesz.}
```

A csomag alapértelmezett mértékegysége 1cm, ezt megváltoztatni a a `\psset` utasítással lehet.

Alapbeállítások

- `linewidth=v` - vonalvastagság.
- `linecolor=szin` - vonalszín.
- `fillsyle=stilus` - kitöltés stílusa (lehet none, solid stb.).
- `fillcolor=szin` - kitöltés szine.
- `arrows=tipus` - nyil tipusa (lehet <->, <-, ->, -).

Nyilakat a következőképpen is definiálhatunk (paraméterként megadva a tipusát):

```
\psline[linewidth=2pt]{<-}(2,1)
```

Alapbeállítások

Az alap grafikai elemek a következő paraméterekkel is rendelkezhetnek:

- `linearc=v` - egyenes illeve sokszőög rajzolásánaál a törés ívét szabályozza ("v" pozitív).
- `framearc=sz` - a keretek sarkainak lekerekítését szabályozza ("sz" radiánban megadott paraméter).
- `cornersize=relative/absolute` - a lekerekítés mértékét szabályozza.

Egyenesek, sokszögek

- `\psline[par]{nyil}(x0,y0)(x1,y1)... (xn,yn)` - szakasz(oka)t rajzol.
- `\psline*[par]{nyil}(x0,y0)(x1,y1)... (xn,yn)` - ha a szakaszok zárt alakzatot alkotnak, akkor kitölti.
- `\qline(coor0)(coor1)` - megadott koordináták közé szakaszt rajzol.
- `\pspolygon*[par](x0,y0)(x1,y1)(x2,y2)...(xn,yn)` - megadott paraméterekkel sokszöget rajzol.
- `\psframe*[par](x0,y0)(x1,y1)` - keretet rajzol (a két paraméter a bal alsó és a jobb felső sarok koordinátái).
- `\psdiamond*[par](x0,y0)(x1,y1)` - rombuszt rajzol, (x0,y0) a középpontja, (x1,y1) a magasság, illetve a szélesség paramétere.
- `\pstriangle*[par](x0,y0)(x1,y1)` - egyenlő szárú háromszöget rajzol, (x0,y0) az alap középpontja, (x1,y1) a magasság, illetve a szélesség paramétere.

Az utolsó két utasítás rendelkezhet egy gangle paraméterrel, ami az alakzat elforgatását szabályozza.

Példák

Tekintsük az alábbi példákat:

```
\psset{unit=3cm}
\begin{pspicture}(0, 0)(2, 2)
\psset{linewidth=1pt}
\psline[linewidth=2pt,linearc=.25]{->}(4,2)(0,1)(2,0)
\qline(0,0)(2,1)
\pspolygon[linewidth=1.5pt](0,2)(1,2)
\pspolygon*[linearc=.2,linecolor=darkgray](1,0)(1,2)(4,0)(4,2)
\psframe[linewidth=2pt,framearc=.3,fillstyle=solid,fillcolor=lightgray]
\psframe*[linecolor=white](1,.5)(2,1.5)
\psdiamond[framearc=.3,fillstyle=solid,fillcolor=lightgray](2,1)(1.5,1)
\pstriangle*[gangle=10](2,.5)(4,1)
\end{pspicture}
```

Kör, körív, ellipszis

- `\pscircle[par](x0,y0){sugár}` - adott középpontú, sugarú kört rajzol.
- `\pscircle*[par](x0,y0){sugár}` - adott középpontú, sugarú körlapot rajzol.
- `\qdisk(coor){sugár}` - adott középpontú, sugarú körlapot rajzol.
- `\pswedge*[par](x0,y0){sugár}{szog1}{szog2}` - körívet rajzol.
- `\psellipse*[par](x0,y0)(x1,y1)` - ellipszist rajzol, (x0,y0) a középpont koordinátái, (x1,y1) a szélesség illetve magasság fele.
- `\psarc*[par]{arrows}(x,y){sugár}{szog1}{szog2}` - ívet rajzol szog1-től szog2-ig, (x,y) a középpontja, óramutató járásával megegyező irányban.
- `\psarcn*[par]{arrows}(x,y){sugár}{szog1}{szog2}` - ívet rajzol, de óramutató járásával ellenkező irányba.
- `\psellipticarc*[par]{arrows}(x0,y0)(x1,y1){szog1}{szog2}` - ellipszisívet rajzol.
- `\psellipticarcn*[par]{arrows}(x0,y0)(x1,y1){szog1}{szog2}` - ellipszisívet rajzol, de óramutató járásával ellenkező irányban.

Példák

Tekintsük az alábbi példákat:

```
\psset{unit=3cm}
\begin{pspicture}(0, 0)(2, 2)
\psset{linewidth=1pt}
\pscircle[linewidth=2pt](.5,.5){1.5}
\qdisk(2,3){4pt}
\pswedge[linecolor=gray,linewidth=2pt,fillstyle=solid]{2}{0}{70}
\psellipse[fillcolor=lightgray](.5,0)(1.5,1)
\psarc*[showpoints=true](1.5,1.5){1.5}{215}{0}
\psline[linewidth=2pt](4;50)(0,0)(4;10)
\psarc[arcsepB=2pt]{->}{3}{10}{50}
\psellipticarc[showpoints=true,arrowscale=2]{->}(.5,0)(1.5,1){215}{0}
\end{pspicture}
```

Görbék

- `\psbezier[par]{arrows}(x0,y0)(x1,y1)(x2,y2)(x3,y3)` - Bezier görbét rajzol adott alappontokkal.
- `\psbezier*[par]{arrows}(x0,y0)(x1,y1)(x2,y2)(x3,y3)` - kitöltött Bezier görbét rajzol adott alappontokkal.
- `\parabola*[par]{arrows}(x0,y0)(x1,y1)` - parabolát rajzol, ami átmegy (x0,y0) ponton és (x1,y1) a maximuma vagy minimuma.
- `\pscurve*[par]{arrows}(x1,y1)...(xn,yn)` - nyitott görbét rajzol.
- `\psecurve*[par]{arrows}(x1,y1)...(xn,yn)]` - nyitott görbét tajzol, ami az első ponttól az utolsóig tart.
- `\psccurve*[par]{arrows}(x1,y1)...(xn,yn)` - zárt görbét rajzol.

Példák

Tekintsük az alábbi példákat:

```
\psset{unit=3cm}
\begin{pspicture}(0, 0)(2, 2)
\psset{linewidth=1pt}
\psbezier[linewidth=2pt,showpoints=true]{->}(0,0)(1,4)(2,1)(4,3.5)
\parabola*(1,1)(2,3)
\psset{xunit=.01}
\parabola{<->}(400,3)(200,0)
\pscurve[showpoints=true]{<->}(0,1.3)(0.7,1.8)(3.3,0.5)(4,1.6)(0.4,0.4)
\psecurve[showpoints=true](.125,8)(.25,4)(.5,2)(1,1)(2,.5)(4,.25)(8,.125)
\psccurve[showpoints=true](.5,0)(3.5,1)(3.5,0)(.5,1)
\end{pspicture}
```

Pontok, rácsok

Pontot, pontokat az alábbi módon rajzohatunk:

- `\psdot*[par](x1,y1)` - adott koordinátára pontot rajzol.
- `\psdots*[par](x1,y1)(x2,y2)...(xn,yn)` - koordinátákra pontokat rajzol.

A `par` értéke a következő lehet:

- `dotstyle=style` - a pont stílusa (alapértelmezett a *).
- `dotsize=dim 'num'` - a pont mérete (alapértelmezett: 2pt 2).
- `dotscale=num1 'num2'` - a pont arányát adja (num1 a vizszintes, az opcionális num2 a függőleges, alapértelmezett: 1).
- `dotangle=angle` - pont elforgatását szabályozza megadott szögben (× stílusú pontoknál például).

Koordináta rendszert az alábbi utasítással rajzolhatunk:
`\psgrid(x0,y0)(x1,y1)(x2,y2)`
Ahol:

- `(x0,y0)` - a koordináta rendszer origója.
- `(x1,y1)` - a koordináta rendszer bal alsó sarka.
- `(x2,y2)` - a koordináta rendszer jobb felső sarka.

Pontok, rácsok példa

```
\psset{unit=3cm}
\begin{pspicture}(0, 0)(2, 2)
  \psgrid(0,0)(-1,-1)(4,3)
  \psdot*[dotstyle=pentagon](3,3)
  \psdots*[dotstyle=oplus](3,2)(2,3)
\end{pspicture}
```

Függvények ábrázolása

- `\\fileplot*[par]{fájl}` - {fájl}-ban megadott koordinátákat rajzolja.
- `\dataplot*[par]{utasítások}` - {utasítások}-nak megfelelően rajzol.
- `\psplot*[par]{xmin}{xmax}{fgv}` - {fgv} függvényt rajzolja ki.
- `\parametricplot*[par]{tmin}{tmax}{fgv}` - paraméteres alakban megadott függvényt rajzol.

Matematikai programcsomagok Prezentációkészítés TEX/LATEX-ben

Dr. Hartung Ferenc - Dr. Virágh János - Pozsgai Tamás

2014. május 2.

Bevezetés

- Tudományos előadást támogató prezentációt elkészíteni viszonylag egyszerűen lehet. A LaTeX-en belül a beamer nevű csomagot fogjuk használni.
- Dokumentumosztályként a `\documentclass{beamer}` parancscsal adjuk meg.
- Nagy előnye, hogy minden platformon ugyanazt a végeredményt látjuk.
- Az `Beamer` első változata 2003-ban jelent meg, készítője Till Tantau.

A prezentáció alkotóelemei

- A prezentáció építőelemei a keretek (frame).
- A kereteknek címet és alcímet tudunk adni.
- Egy keret állhat diák sorozatából, de akár egyetlen diából is.
- Ha a keret több diából áll, akkor a diasorozat a keretben egymás után jelenik meg.

Keretek létrehozása

Egy keret felépítése kétféle módon adható meg:

```
\begin{frame}
\frametitle{A keret címe}
\framesubtitle{A keret alcíme}
   A keret tartalma
\end{frame}
```

vagy

```
\frame{
\frametitle{A keret címe}
\framesubtitle{A keret alcíme}
      A keret tartalma
}
```

Keretek alkotóelemei

- Fej- illetve lábléc
- Bal illetve jobboldali sáv
- Keret címe, alcíme
- Navigációs gombok
- Háttér
- Keret szövege

A keretek fontosabb opciói

- Szöveg igazítása: `t`, `b`, `c` (top, button, center). Alapértelmezett a középre igazítás.
- plain: A fejléc, lábléc, oldalsávok nem jelennek meg.
- `shrink=érték`: A szöveget felülre igazítja és érték %-ban kicsinyíti.
- fragile: Alapértelmezésben idézet (verbatim környezet) nem írható a keretbe. Ezen opció segítségével már igen.

Címoldal készítése

Címoldalnak nevezzük a prezentáció első oldalát. Az ehhez szükséges adatokat a következő utasításokkal adhatjuk meg:

- `\title{Prezentáció címe}`
- `\subtitle{Prezentáció alcíme}`
- `\author{Szerző}`
- `\institute{Intézet}`
- `\date{Dátum}`
- `\subject{A prezentáció tárgya}`
- `\keywords{Kulcsszavak}`

A szöveg tagolása

A prezentáció a következő részekre tagolható:

- part: rész
- section: szakasz
- subsection: alszakasz

Részek

Részekre bontásra hosszabb prezentációk esetében lehet szükség. Új részt a következőképpen tudunk definiálni:

`\part[rövid név]\{A rész címe}`

Ha új oldalt szeretnénk egy új rész kezdetekor definiálni, akkor a deklarációs részben adjuk meg a következőket:

```
\AtBeginPart{
 \begin{frame}[fragile][plain]
 \begin{center}
 {\Large\insertromanpartnumber. rész\\[10mm]}
 {\large\insertpart\\}
 \end{center}
 \end{frame}
```

Ezek után a \part\{A rész címe} utasítást kiadva kapunk egy

Szakaszok

A prezentáció rövidebb, tartalmilag összefügő szakaszokra bontását a következőképpen tehetjük meg:
`\section{A fejezet címe}`
Minden fejezetet egy `\end{section}` utasítással kell lezárni.
A fejezeteinket alfejezetekre bonthatjuk a `\subsection` utasítással.
A fejezetek, alfejezetek rendszere a tartalomjegyzék készítését segíti elő, illetve megjelenthetnek a navigációs sávban, vagy a pdf meg/-je/-lenítő program könyvjelzői között.

Tartalomjegyzék készítése

A részek, szakaszok, alszakaszok rendszeréből külön keretben tartalomjegyzék készíthető. A tartalomjegyzék részekre vonatkozik. Ha nincs részekre osztva a dokumentum, akkor az egész prezentációra érvényes.
A következő kóddal tudunk tartalomjegyzéket külön keretbe készíteni:

```
\frame{\tableofcontents}
```

Tartalomjegyzék készítése

Ha a tartalomjegyzék elemeit egyesével szeretnénk megjeleníteni, akkor két lehetőségünk van:

- Mindegyik elem elé egy `\pause` parancsot írunk.
- `\tableofcontents` parancs `pausesections` opcióját használjuk.

További opciói a `\tableofcontents` parancsnak:

- sectionstyle=stílus
- subsectionstyle=stílus
- subsubsectionstyle=stílus

Ahol a stílus lehet show, hide illetve shaded.

Irodalomjegyzék

Adott kereten belül készíthetünk irodalomjegyzéket is.

```
\begin{frame}[fragile][plain]{Irodalomjegyzék}
\begin{thebibliography}{12}
\bibitem{Micimacko} A. A. Milne, ...
\bibitem{Kisherceg} A. de Saint-Exupéry, ...
.
.
.
\end{thebibliography}
\end{frame}
```

Lista készítése

A listák készítésére szolgáló csomag (enumerate) automatikusan betöltődik. Az alap listatípusok (itemize, enumerate, description) használhatók. Általános definiciója:

```
\begin{frame}[fragile]
 \begin{itemize}
  \item 1. elem
  \item 2. elem
  \item 3. elem
 \end{itemize}
\end{frame}
```

Lista készítése

A \begin{enumerate}[stilus] utasítással a lista számozási jeleit tudjuk megváltoztatni, ahol a stílus a következő lehet:

- 1 - arab szám
- a - latin abc kisbetű
- A - latin abc nagybetű
- i - kicsi római szám
- I - nagy római szám

Ha összetett jelet szeretnénk, akkor a a következő lehetőségünk lehet:
\begin{enumerate}[{I}.a]
Ez az utasítás a következő számozást hozza létre: I.a, I.b, I.c, stb.

Dobozok készítése

Hasonlóan használhatók, mint bármilyen más dokumentum készítésekor. Két bekezdésdoboz közül választhatunk.

- `beamercolorbox` - cím nélküli, több paraméterezési lehetőség
- `beamercolorboxesrounded` - cím adható

Beamercolorbox

Alapszintaxis:

```
\begin{beamercolorbox}[opció]{szín}
Doboz tartalma
\end{beamercolorbox}
```

Az opciók:

- wd=szélesség
- dp=mélység
- ht=magasság
- left, right, center
- sep=távolság
- shadow=true, false
- rounded=true, false

Beamerboxesrounded

Alapszintaxis:

```
\begin{beamerboxesrounded}[opció]{Cím a fejrészben}
    Doboz tartalma
\end{beamerboxesrounded}
```

Az opciók:

- width=szélesség
- shadow=true, false
- lower=szin1
- upper=szin2

A lower és az upper opciók a doboz tartalmának, illetve fejrészének színösszeállítására szolgálnak.

Dobozok színösszeállítása

A dobozok színeit a \setbeamercolor utasítással tudjuk megadni.

```
\setbeamercolor{sajat szin}{fg=white,bg=blue}\\
```

Ahol

- fg - a doboz tartalmának a színe
- bg - a háttérszín

Doboz definiálása

A két definicó egy-egy példán bemutatva:

```
\setbeamercolor{szin}{fg=blue,bg=yellow}
\begin{beamercolorbox}[wd=7cm,shadow=true,
rounded=true,left]{szin}
Ez egy 7cm széles, árnyékolt, lekerekített
          sarkú balra igazított doboz lesz.
\end{beamercolorbox}
```

Illetve:

```
\setbeamercolor{szin1}{fg=white,bg=blue}
\setbeamercolor{szin2}{fg=black,bg=white}
\begin{beamerboxesrounded}[upper=szin1,
    lower=szin2,shadow=true]{Doboz címe}
A cím kék alapon fehér, a szöveg fehér
    alapon fekete színű. Ez a doboz is árnyékolt.
```

Hasábokra osztás

```
\begin{columns}[opció]
\begin{column}{1. oszlop szélessége}
1. oszlop tartalma
\end{column}
\begin{column}{2. oszlop szélessége}
2. oszlop tartalma
\end{column}
...
\end{columns}
```

Hasábok opciói

- `totalwidth=szélesség` - a többhasábos terület teljes szélessége
- `b` - az oszlopok alsó sorainak alapvonalát igazítja
- `c` - az oszlopok vertikális közepét igazítja
- `t` - az oszlopok felső sorainak alapvonalát igazítja
- `T` - az oszlopok felső sorainak tetejét igazítja

Képek

- Minden keretre alkalmazott háttérkép beillesztése a deklarációs részben a következőképpen lehetséges:

```
\setbeamertemplate{background canvas}
 {\includegraphics[width=\paperwidth]{kep}}
```

- Kép beillesztése a diára:

```
\includegraphics[width=szélesség]{kep}
```

Animáció

- Deklarációs résznél töltsük be az `xmpmulti` csomagot. Az animációt diasorozatként játssza le. Az animációt az alábbi utasítással tudjuk elkészíteni:

```
\multiinclude[<+>][format=jpg,graphics={width=5cm}]{kep}
```

- Deklarációs résznél töltsük be az `animate` csomagot. Az animációt videóként játsza le. Az animációt az alábbi utasítással tudjuk elkészíteni:

```
\animategraphics[opciók]{sebesség}{kep}{elso}{utolsó}
```

Animategraphics opciói

- sebesség - Pozitív egész, ennyi kép/másodperc sebességgel játsza le.
- kep - az animációt alkotó alapképek közös neve (példánkban kep volt) .
- elso - A képsorozat első elemének sorszáma.
- utolsó - A képsorozat utolsó elemének sorszáma.
- autoplay - Az oldal megnyitásakor automatikusan induljon.
- loop - A lejátszás végén automatikusan újraindul.
- width=szélesség - A képek szélessége.
- height=magasság - A képek magassága.
- controls - Lejátszó gombok megjelenítése.
- buttonsize=gombméret - Lejátszó gombok mérete.
- buttonbg=szín - Lejátszó gombok hátterének a színe.

Videó

- Külön alkalmazással lejátszatni a videót a következőképpen tudjuk:

  `\href{run:video.avi}{szöveg}`

- Ha a prezentációban szeretnénk lejátszani a videót, akkor a deklarációs részbe be kell töltenünk a `multimedia` csomagot.
- A video fájlt külön kell bemásolni a pdf mellé.
- A következő utasítással tudjuk elhelyezni a videót:
  `\movie[opciók]{szöveg}{videófájl neve}`

Movie opciói

- width=szélesség - A videó szélessége.
- height=magasság - A videó magassága.
- poster - A videó elindulásáig nem a "poszter szöveg" látható, hanem a videó első képkockája. Erre kattintva indul a lejátszás.
- showcontrols - Megmutatja a videó alatt a navigációs sávot.
- start=ido - A videó lejátszási kezdopontjának megadása.
- duration=ido - A videóból milyen hosszú részt játszon le.

Nagyítás

Dia egy adott területét kinagyíthatjuk az alábbi utasítással:

```
\framezoom<1><2>[border=keret](bal távolság,
          fent távolság)(szélesség,magasság)
```

Ekkor a megadott képen megjelenik egy keret vastagságú téglalap, amit nagyítani lehet. A kijelölt terület a szélesség $\times$ magasság méretű, pozíciója pedig a bal felső saroktól számított érték.

- Csak teljes képernyős üzemmódban működik helyesen.
- A <1><2> jelentése: a teljes kép a keret első diáján jelenik meg, míg a kinagyított terület a 2. dián.

Témák

Előre definiált stílusokat, úgynevezett témákat tudunk betölteni.
Típusai a következők lehetnek:

- Teljes
- Belső elemek
- Külső elemek
- Szinek
- Betütipus

Teljes témák

Deklarációs részben kell megadni a következőképpen:

```
\usetheme[opciók]{név}
```

Az opciók témánként változnak, legtöbb esetben nincs lehetőségünk a használatukra. Látványképeket találhatunk a témákról az alábbi címen:

http://deic.uab.es/~iblanes/beamer_gallery/

Teljes témák

Előre elkészített témák teljesség igénye nélkül:

- AnnArbor
- Berkeley
- Berlin
- CambridgeUS
- Dresden
- Goettingen
- Madrid
- Malmoe
- PaloAlto
- Singapore
- Szeged
- Warsaw

Belső témák

Deklarációs részben kell megadni a következőképpen:

`\useinnertheme[opciók]{név}`

Az opciók a következők lehetnek:

- circles
- rectangles
- rounded - shadow opció
- inmargin

Külső témák

Deklarációs részben kell megadni a következőképpen:

`\useoutertheme[opciók]{név}`

Az opciók a következők lehetnek:

- infolines
- miniframes
- smoothbars
- sidebar
- split
- shadow
- tree
- smoothtree

Színtémák

Deklarációs részben kell megadni a következőképpen:

```
\usecolortheme[opciók]{név}
```

Teljes színtémák

- albatross
- beetle
- crane
- dove
- fly
- seagull

Belső színtémák

- lily
- orchid
- rose

Külső színtémák

- dolphin
- seahorse
- whale

Betűtipus témák

Deklarációs részben kell megadni a következőképpen:

```
\usefonttheme[opciók]{név}
```

Ahol az opciók a következők lehetnek:

- serif
- structurebold
- structureitalicserif
- structuresmallcapsserif

Több dia egy keretben

A keret elemeit egymás után több dián legegyszerűbben a pause utasítással tudjuk.
Overlay megjelenések:

- `\uncover<spec>{szöveg}`
- `\visible<spec>{szöveg}`
- `\only<spec>{szöveg}`
- `\alt<spec>{szöveg1}{szöveg2}`
- `\temporal<spec>{elotte}{szöveg}{utána}`

Overlay megjelenések

Lehetőséget biztosít a beamer a kereten belüli diák vezérlésére. Különböző prezentáció elemekhez rendelhetjük hozzá, hogy a keret melyik diáin jelenjenek meg. Mindíg a megadott elem után kell megadni. Tekintsük a következő példát:

```
\begin{frame}[fragile]
\begin{itemize}
  \item<1-2> 1. elem
  \item<2> 2. elem
  \item<3> 3. elem
\end{itemize}
\end{frame}
```

Overlay lehetőségek

- <0> - egy dián sem jelenik meg.
- <1> - az első dián jelenik meg.
- <1-2> - az első és a második dián jelenik meg.
- <1,3> - az első és a harmadik dián jelenik meg.
- <2-> - a második diától az utolsóig jelenik meg.
- <-3> - az első három dián jelenik meg.
- <-2,4-5,9-> - az első két dián, a negyedik és ötödik dián illetve a nyolcadik diától az utolsóig lesz látható.

Léptető overlay

A számok helyére írható léptető overlay utasítás is. Ehhez a program a `beamerpauses` számlálót használja. Kezdeti értéke 1.

- `<+>` - a beamerpauses értékénél eggyel nagyobb számú dián jelenik meg.
- `<+(2)>` - a bemaerpausue értékénél kettővel nagyobb számú dián jelenik meg.
- `<.>` - a beamerpauses értékénél eggyel kisebb értékű dián jelenik meg.
- `<.->` - a beamerpauses értékénél eggyel kisebb értékű diától az utolsó diáig jelenik meg.

Overlay alapértékek

Azon utasítások, amellyeknél lehetőségünk van az overlay használatára, különböző alapértelmezett megjelenési értékkel rendelkeznek. Ez az érték az általunk bemutatott elemeknél mindenhol <1->.
Kivetel ezalól a `\temporal` utasítás, amelynek nincs alapértelmezett értéke, itt kötelező megadni overlay értéket.
Overlay használható keretek címére és alcímére is. Ekkor a keret megadott diái fognak csak megjelenni.

Tartalom cseréje

Lehetőségünk van arra is, hogy az egymást követő diákon a tartalom teljesen különböző legyen.

- `\only` - függőlegesen középre igazított.
- `overprint` környezetben `\onside` felsorolás.

Utóbbi függőlegesen egy sorba igazítja a dia tartalmát.

Automatikus diaváltás

A prezentációt, vagy egyes részeit automatikusan is lejátszhatjuk. Ez csak teljes képernyős módban lehetséges.

`\transduration<spec>{ido}`

Az overlay alapértelmezése <1->. Az idő helyére azt az időt kell írni másodpercben, amennyi ideig látni szeretnénk a diát.

Kereszthivatkozás

Keret adott diájára ugrani a keret label opciójával lehetséges.

```
\begin{frame}[fragile][label=cimke]
\begin{itemize}
\item<+-> 1. elem
\item<+-> 2. elem
\end{itemize}
\end{frame}
```

Ekkor másik diában a következőképpen hivatkozhatunk rá:

```
\begin{frame}[fragile]
\ref{cimke<2>}
\end{frame}
\end{verbatim}
```

Ekkor a második keretben levő `\ref{cimke<2>}` utasítás a második diára létrehoz egy linket, amire kattintva átugrik a cimkével megjelölt keret második diájára.

Ha a `\ref{cimke<2>}` helyett `\hyperlink{cimke<2>}{Link szövege}`

Nyomógombok

A \hyperlink{cimke<2>}{Link szövege} utasítás második paraméterének nem csak szöveget, hanem nyomógombot is megadhatunk.

- \beamerbutton{gomb szövege}
- \beamergotobutton{gomb szövege}
- \beamerskipbutton{gomb szövege}
- \beamerreturnbutton{gomb szövege}

Keret ismétlése

Amennyiben a keretünket megjelöltük cimkével, akkor lehetőségünk van a prezentáció másik részében is megjeleníteni. A második, illetve további megjelenéseknek természetesen adhatunk más overlay megjelenési utasításokat is. Általános szintaxis:

```
\againframe<overlay>[opciók]{keret cimke}
```

Matematikai programcsomagok
Matematika és az internet

Dr. Hartung Ferenc - Dr. Virágh János - Pozsgai Tamás

2014. május 5.

Tartalomjegyzék

Matematikai leíró nyelvek céljai, előnyei

- Matematikai anyagokat kódolhatunk tudományos kommunikációhoz, és oktatási célra.
- Együttesen jelölhetjük meg a matematikai jelölést és jelentést.
- Többféle konvertálási lehetőség matematikai formákról, vagy formákra (pl.: Grafikus megjelenítés, Braille írás, beszédszintetizáció, egyszerű szöveges megjelenítés).
- Emberi olvasásra alkalmas dokumentumokat készíthetünk.
- Matematikai funckiók könnyebb megjelenítése akár webes felületeken is.

Matematikai leíró nyelvek céljai, előnyei

- Adatokat adhatunk át más alkalmazásoknak (pl.: TEX).
- Számítógépek által könnyen legenerálható és végrehajtható kódokat készíthetünk.
- Nyomtatótól függő legmagasabb szintű nyomtatott anyagokat tesznek lehetővé.
- A Web-oldalakon található egyenletek egyszerűen átadhatók más alkalmazásoknak az egér és a böngésző segítségével.
- Az egyenletszerkesztők, és -konvertálók könnyen továbbfejleszthetők.

A HTML korlátai

Megjelenítési akadályok:
A legtöbb szövegszerkesztő, nem szövegként jeleníti meg a matematikai egyenleteket, pl.:

$$2^{2^x} = 10$$

Ez nem szabványos, így megjelenítő programonként változhat akár a betűméret, vagy típus. Ha például Wordben mentjük HTML formátumban, abban az esetben képként illeszti a dokumentumba.

A HTML korlátai

Másik probléma, hogy egy egyenlet képe mindig fehér háttérrel jelenik meg. Ha egy másik rendszerben másik háttér van beállítva, akkor a kép határáig fehér marad a háttér. Példa:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

HTML-be illesztés után fellépő gondok:

- Szövegsor aljához van igazítva a kép.
- Kép mérete.
- Környezet betűmérete eltérő.

Nyomtatási gondok:

- Eltérő felbontás (pixeles, széteső kép).
- Minden a HTML-nél felsorolt probléma.

A HTML korlátai

Kódolási problémák.

- HTML-nél képalapú egyenletben nem tudunk szövegre keresni.
- Nem tudunk vágólapra illeszteni egyenletet, vagy egyenlet részletet.
- Az <IMG> elem ALT="..." paramétere valamely szinten segítséget nyújt.
- Kép megjelenítéséhez nagyobb sávszélességre van szükség, míg egy formula lekódolása végezhető kliens oldalon.

MathML és OpenMath

Matematikai formulák internetes megjelenítésére két lehetőségünk van.

- MathML szabvány
- OpenMath szabvány

MathML történet

- w3.org/Math/
- 1995, első tervezet
- 1997, munkacsoport megalakul a w3c-n belül
- 1999, 1.01 verzió specifikációja
- 2001, 2.0, majd 2003-ban a 2. verzió
- 2010, 3.0
- 2014 MathMl3 2nd Edition

MathML

Két részből áll a szabvány, két megközelítési mód

- Megjelenítést leíró rész (Presentation MathML).
  Pl. <mrow>, <msup>
- Tartalmat leíró rész (Content MathML).
  Pl. <plus/>, <vector>

Példa megjelenítő mód

$$(a+b)^2$$

```
<msup>
  <mfenced>
      <mrow>
          <mi>a</mi>
          <mo>+</mo>
          <mi>b</mi>
      </mrow>
  </mfenced>
  <mn>2</mn>
</msup>
```

Példa tartalmi mód

$$(a+b)^2$$

```
<apply>
  <power/>
  <apply>
     <plus/>
     <ci>a</ci>
     <ci>b</ci>
  </apply>
  <cn>2</cn>
</apply>
```

OpenMath

Az **OpenMath** egy leíró nyelv, amellyel matematikai formulákat jeleníthetünk meg webes felületen.

- www.openmath.org
- Kifejlesztése 1993-ban indult.
- Európai fejlesztésű nyelv.
- Létrejött az *OpenMath Society* ami az első specifikációt adta ki.
- Első kiadás: 1996.
- 1997-ben vált az EU egyik szabványává (EU Negyedik Keretprogramjának hatására).
- 2002 1.1 verzió
- 2004 2.0 verzió, jelenleg is érvényes

OpenMath

Az *OpenMath* nyelvnek kétféle kódolási formája van:

- XML formátumú kódolás
- ún. bájtfolyam típusú

Az XML formátumú kódolás:

- Emberi megértésre alkalmas.
- Hátránya: elég terjedelmes.

Példa XML kódra

$$(a + b)$$

```
<OMOBJ>
  <OMA>
      <OMS cd="arith" name="plus"/>
      <OMV name="a"/>
      <OMV name="b"/>
  </OMA>
</OMOBJ>
```

Bájtfolyam típusú kódolás

- Minden XML-kódolású *OpenMath* szimbólumnak van bájtokban leírt megfelelője.
- rövidebb mint az XML kód.
- Hátránya: az ember számára kevésbé könnyen érhető formátum.

Példa bájtfolyam típussal:

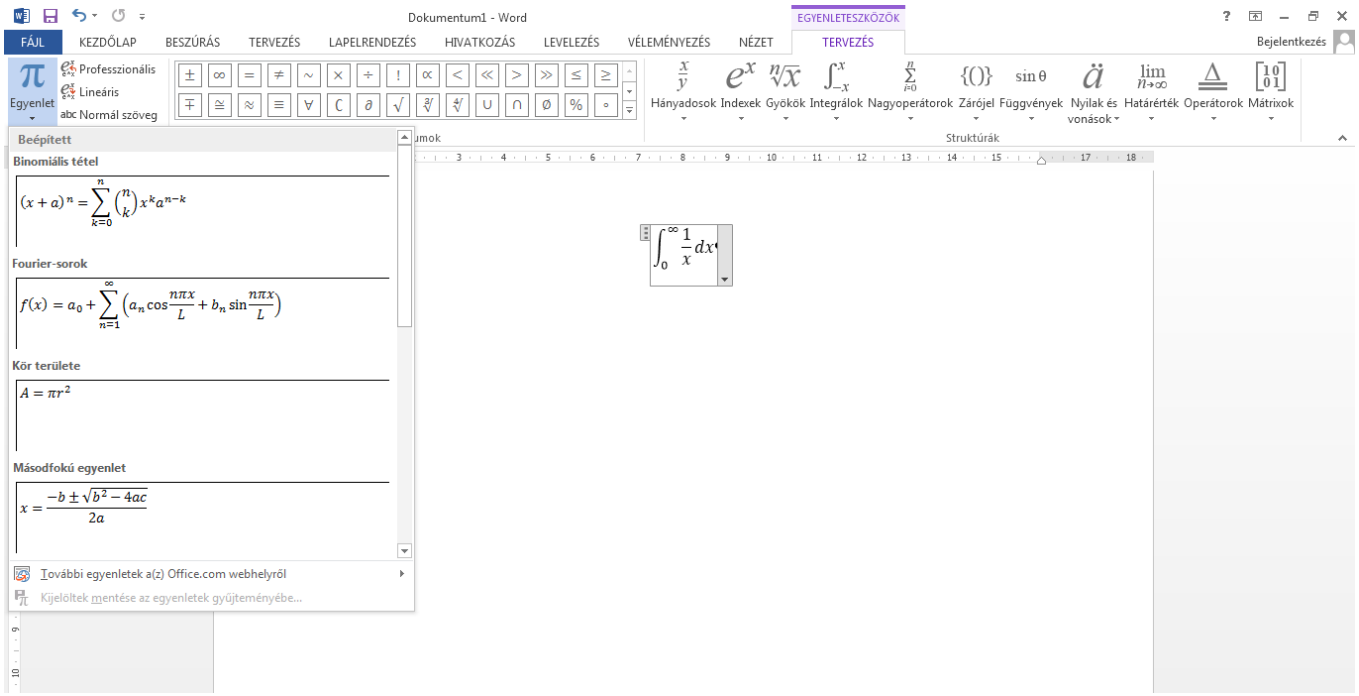
```
18 10 08 05 05 61 72 69 74 68 74 69 6d 65 73 10 08 05 04 61
72 69 74 68 70 6c 75 73 05 01 78 05 01 79 11 10 48 01 45 00
05 01 7a 11 11 19
```

Microsoft Word

- Microsoft Word 2003 verzióig - Equation 3.0
- Microsoft Word 2007 verziótól - Egyenletszerkesztő

Egyenletszerkesztő MS Word 2013

Egy integrál létrehozása.



Egyenletszerkesztés MS Word 2013

Előnyei:

- Megszokott környezet, széles beállítási választék
- Néhány kattintás segítségével létrehozható a képlet.
- Bármelyik Office programmal használható.

Hátrányai:

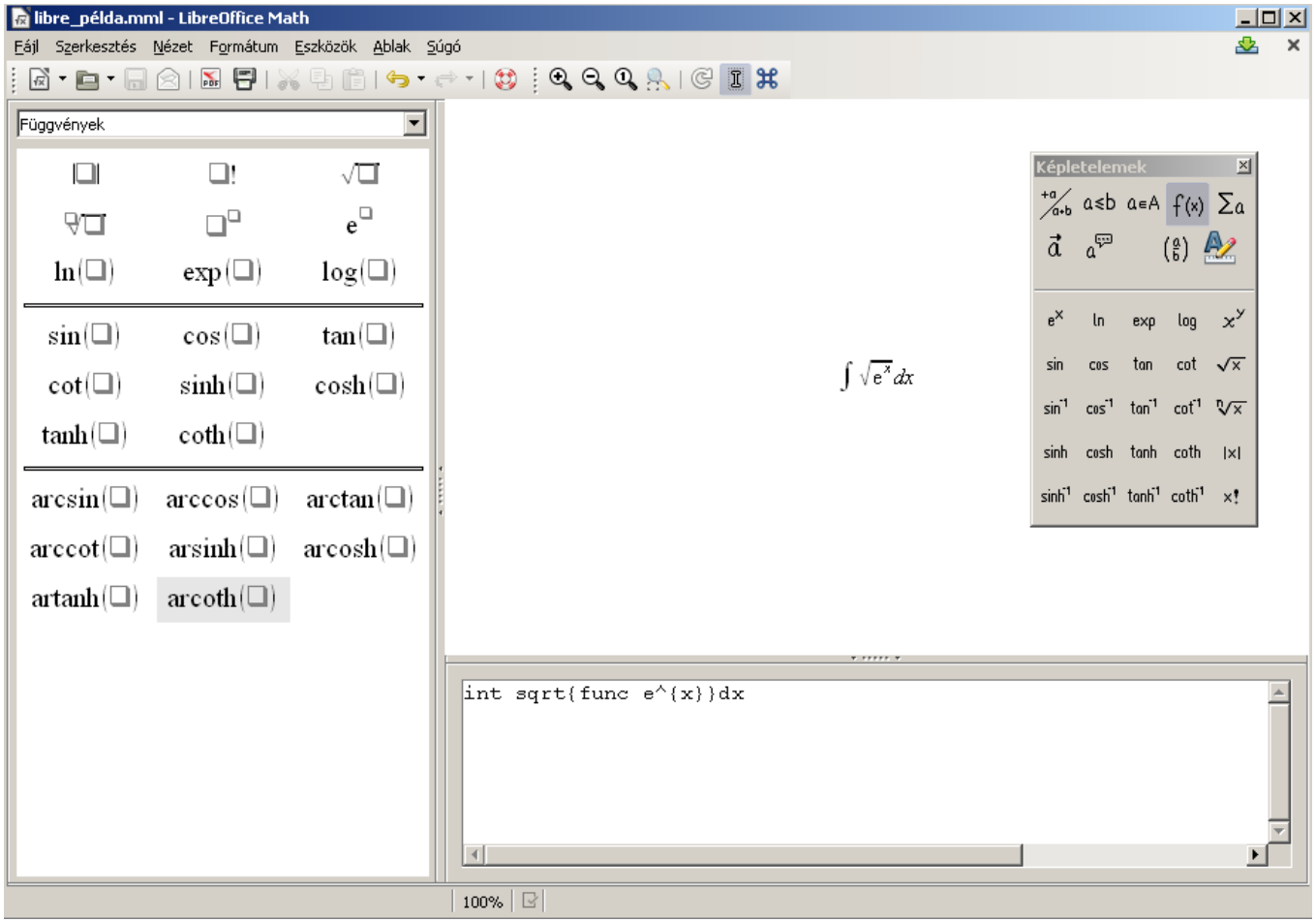
- Nagyon egérfüggő.
- .html kiterjesztés esetén képként menti a képleteket.

Libreoffice Writer

- Math képletszerkesztő, önállóan is használható.
- Az Office csomag bármelyik részével együttműködik.

Libreoffice Writer

Egy integrál létrehozása Libreoffice Math használatával.



Libreoffice Writer

Előnyei:

- Az utasítások könnyen megjegyezhetők.
- Kevesebb kattintással lehet képletet létrehozni.
- A Libreoffice bármelyik programjával együtt tud működni.

Hátrányai:

- Kevésbé felhasználóbarát.
- .html kiterjesztés esetén képként menti a képleteket.

Lyx

- www.lyx.org
- TEX alapú szerkesztő.
- Saját kiterjesztést használ, de képes .tex file-okat is megjeleníteni.
- Sokféle kiterjesztésre konvertálható (pdf, ps, dvi)

eLyxer

- A Lyx programhoz készített konvertálóprogram
- Letölthető a http://elyxer.nongnu.org/ címről.
- MathMl szabvány szerinti .html fájlt hoz létre.

Konvertálóprogramok

Amikor *TeX* fájlokat weboldalakra szeretnénk konvertálni, akkor több lehetőségünk is van.

- Latex2html
- TtM (TeX to MathML)
- MathToWeb

Latex2html

A legrégebbi konvertáló program.
Letölthető a www.latex2html.org oldalról Előnyei:

- Könnyen használható.

Hátrányai:

- Nem fejlesztik 2008 az utolsó változat), újabb utasításokat nem képes fordítani.
- A képleteket képként menti el.

TtM

A TtM (Tex to MathMl) program, mint a neve is mutatja MathMl nyelvre fordítja le a *TeX* forrásunkat.
Letölthető a hutchinson.belmont.ma.us/tth/mml oldalról.
Előnyei:

- Képleteket nem képként fordítja.
- Egyszerű használni.

Hátránya:

- Bonyolulabb fájlok fordítására nem alkalmas.

MathToWeb

Az egyik legjobban használható konvertáló program.
Letölthető a www.mathtoweb.com oldalról.
Előnyei:

- Képleteket nem képként fordítja.
- Létezik online változata is.
- Bonyolult fájlok konvertálására is használható.

Hátránya:

- Nehéz helyesen beállítani.

Online szerkesztők

- Word online
- Googledocs
- LaTeXLab

Word online

- Microsoft Word online változata
- Az Office 2013 verzió alapján készült
- Jelenleg még nem tud képleteket szerkeszteni.

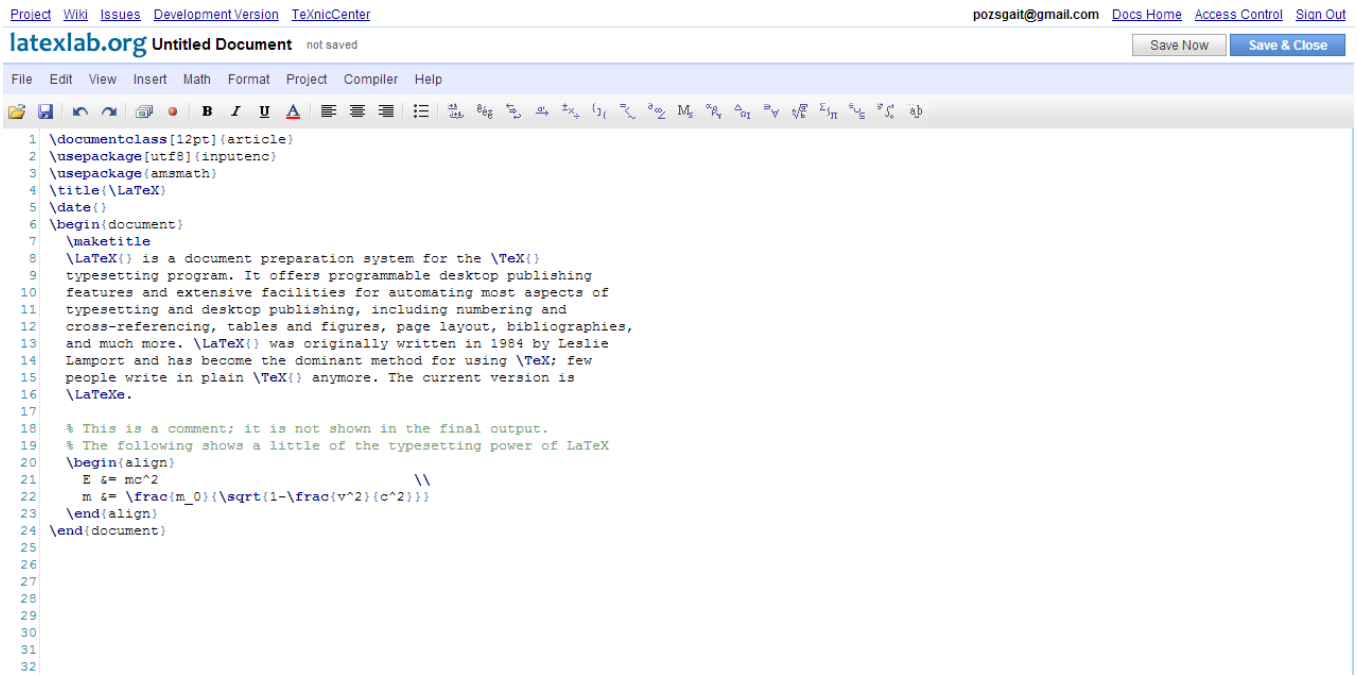
Googledocs

- A Google online office csomagja.
- Jelenleg nem tud képleteket szerkeszteni.
- A Google *Tex* szövegek készítésére alkalmas programja a LaTeXLab.

LatexLab

- docs.latexlab.org
- Google dokumentumokkal szorosan együttműködik.
- Off-line fordítót is képes használni.
- Saját fordító használatára is lehetséges.
- Egyik változata sem támogatja .html fájlokat.

LatexLab


Project Wiki Issues Development Version TeXnicCenter
pozsgait@gmail.com Docs Home Access Control Sign Out
latexlab.org Untitled Document not saved
Save Now
Save & Close
File Edit View Insert Math Format Project Compiler Help
\documentclass[12pt]{article}
\usepackage[utf8]{inputenc}
\usepackage{amsmath}
\title{\LaTeX}
\date{}
\begin{document}
\maketitle
\LaTeX{} is a document preparation system for the \TeX{}
typesetting program. It offers programmable desktop publishing
features and extensive facilities for automating most aspects of
typesetting and desktop publishing, including numbering and
cross-referencing, tables and figures, page layout, bibliographies,
and much more. \LaTeX{} was originally written in 1984 by Leslie
Lamport and has become the dominant method for using \TeX; few
people write in plain \TeX{} anymore. The current version is
\LaTeXe.
% This is a comment; it is not shown in the final output.
% The following shows a little of the typesetting power of LaTeX
\begin{align}
E &= mc^2 \\
m &= \frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}}
\end{align}
\end{document}

Webmathematics Interactive matematikai protál

- Szegedi Egyetemen fejlesztik
- Egyetemi matematikai témakörök
- Folyamatos fejlesztés, több változat egyszerre elérhető

Webmathematics Interactive matematikai protál

- A megoldandó feladatok szintaxisát nekünk kell kitalálni
- Függvények használatánál okoz nehézséget
- A megoldások képleteinek megjelenítését LaTeX2html végzi
- Képként jeleníti meg a képleteket

Sage

- A nyílt forráskódú matematikai szoftvercsomag
- Teljeskörű matematikai programcsomag
- Jól dokumentált

Sage használata

- Regisztálni a Sage központi szerverére
- Saját Sage szerver
- Localhostként futtatva

Wolframalpha portál

- Nem csak matematikai portál
- Alap matematikai feladatok elvégzésére alkalmas
- Widgetek használata

Wolframalpha widgetek

- Körülbelül 200 matematikai widget
- Egyszerűen készíthető saját widget
- Könnyen beilleszthető bármilyen weboldalba

Matematikai programcsomagok MathML

Dr. Hartung Ferenc - Dr. Virágh János - Pozsgai Tamás

2014. május 2.

Tartalomjegyzék

Bevezetés
Definíció

MathML definíciója

Mathematical Markup Language azaz Matematikai Leíró Nyelv, egy XML szabványt használó nyelv, mely segít a matematikai egyenletek formájának és tartalmának megörzésében, valamint ábrázolásában. Célja, a matematikai formulák integrálása internetes oldalakra, és egyéb dokumentumokba. A W3C ajánlása.

Bevezetés
MathML története

- 1998: Megjelenik a MathML 1.0 a World Wide Web Consortium (W3C) első XML ajánlásaként
- 1999: MathML 1.01
- 2001: MathML 2.0 feltünése
- 2003: MathML 2.0 kiadása
- 2006: W3C ajánlására a MathML Working Group nekilát a MathML 3 megtervezésének
- 2010: MathML 3.0 hivatalos, mai napig használt, kompatibilis a korábbi 2.0-ás verzióval

Bevezetés
MathML céljai

- matematikai anyagot kódolhatunk tudományos kommunikációhoz, és oktatáshoz minden szinten
- a kódolás során, együttesen határozhatjuk meg a matematikai jelölést és a jelentést
- konvertálható más nyelvekre, formákra, megjelenítésekre
  - grafikus megjelenítés
  - beszédszintetizáció
  - számítógépes algebrai rendszerek bemenete
  - egyéb matematikai leíró nyelvek (pl.: TEX)
  - egyszerű szöveges megjelenítés
  - nyomtatott forma, akár Braille is
- könnyű kereshetőség
- gondoskodhatunk a kiterjeszthetőségről
- sablonok létrehozása
- gépi programokkal könnyen generálható, olvasható

Elemek rendszerezése

A MathML elemei három csoportba sorolhatók:

- megjelenítő elemek
- tartalmi elemek
- interfész elemek

Megjelenítő elemek

A megjelenítő elemek leírják a matematikai jelrendszer struktúráját.
Példák:

<mrow>

A vízszintesen egymás után írt karaktereket jelzi.

<msup>

A felső index jelölésére szolgál.

Tartalmi elemek

A tartalmi elemek a matematikai objektumokat írják le közvetlenül, szemben a jelöléssel, amivel azokat csak megjelenítjük.
Példák:

<plus/>

Számok, elemek összeadását írja le.

<vector>

Vektorok megjelenítésére használható.

Interfész elemek

A <math> elemet soroljuk ide.

<math>

- A legfelső szinten helyezkedik el,és interfész elemnek nevezzük. Egyik funkciója, hogy továbbadja a paramétereket egy MathML processzornak, és így befolyással van a teljes kifejezésre.
- A másik feladata kommunikálni a böngészővel, amely megjeleníti a MathML kifejezéseket, így integrálja azokat a HTML környezetbe.

Kifejezés-fák és token-elemek

Mind a megjelenítési, mind a tartalmi kifejezések számos formális tulajdonságra bonthatták. Mindekét esetben a kifejezés részkifejezésekre, darabokra bontható. Például a $(a+b)^2$ felbontható az $(a+b)$ alapkifejezésre, és egy leíró kifejezésre, ami most: 2. Ez is tovább bontható, és így tovább.

A MathML kifejezések ezekből a felbontásokból képzett fák.

- **szülőcsoportok**: az eredeti kifejezés
- **utódok**: a fa levelei

Kifejezés-fák és token-elemek

Levelek csoportjai:

üres elem

Közvetlenül a MathML-ben található szimbólumot jeleníti meg. Például: <**plus**/> tartalmi elem.

token-elem (jelöléselem)

Az egyetlen olyan MathML-es elem, mely megenged közvetlen karakter kódolást.

magyarázó elem

Az adatok nem MathML alakó állapotának megtartására használhatjuk.

A megjelenítő mód legfontosabb token elemei

<mi>

Azonosítók ("mathematical identifier")
Dőlt: (x)

<mn>

Számok ("mathematical number")
álló betütípus: (2)

<mo>

Műveleti jelek ("mathematical operator")
Szélesebb szóköz: (+)

A három alap megjelenítő elemen kívül a legtöbb: üres megjelenítő elem. Ezeket leginkább sorbarendezéssel kapcsolatban használjuk.

A tartalmi jelölés token elemei

<ci>

Azonosítók ("content identifier")

<cn>

Számok ("content number")

Kifejezés-fák és token-elemek

A MathML elemeknek általában van nyitó és záró tagjuk, melyek körbeveszik a tartalmukat.

- token-elemeknél a tartalom karakteres adat
- többi esetben valamilyen utód elem
- üres elemeknél <elemnév/> Például: <plus/>

Megjelenítő mód

Példa: $(a+b)^2$

```
<msup>
  <mfenced>
    <mrow>
      <mi>a</mi>
      <mo>+</mo>
      <mi>b</mi>
    </mrow>
  </mfenced>
  <mn>2</mn>
</msup>
```

Tartalmi jelölés

Példa: $(a+b)^2$

```
<apply>
    <power>
    <apply>
      <plus/>
      <ci>a</ci>
      <ci>b</ci>
    </apply>
    <cn>2</cn>
</apply>
```

Kifejezés-fák és token-elemek

Megjelenítő jelölés

A szerkezeti séma több osztályba sorolható:

- Írásformákkal áll kapcsolatban például: <**msub**>, <**msup**>, <**munder**>
- általánosan írja le a szerkezetet, például: <**mrow**>, <**mfrac**>
- táblázatokkal áll kapcsolatban
- <**maction**> önmagában külön kategória, a jelöléssel kapcsolatos vezérléseket jelöli.

Példa

A legtöbb szerkezeti sémánál fontos az utódok sorrendje.
Az <**mfrac**> elem első utóda a számláló, a második a nevező.
Ekkor mint argumentumaira hivatkozunk a megfelelő sorrendben, és tudjuk, hogy az <**mfrac**> elem egy konstruktor.

$$a/b$$

```
<math>
  <mfrac>
    <mi>a</mi>
    <mi>b</mi>
  </mfrac>
</math>
```

Speciális jelek

A speciális jelek segítenek a speaciális karakterek ábrázolásában és a könnyebb hangos felolvashatóság segítésében.
Néhány a MathML által használt speciális jel:

- 	: tabulátor
-
: új sor
- &IndentingNewLine;: új bekezdés
- &Space;: szóköz (ennek több fajtája is van, hosszát tekintve)
- ⁢: láthatatlan szorzás
- ±: ± a tartalmi jelölőrendszerben erre nincs elem, így az <**fn**> elemet vezettük be helyette
- &false;: logikai "hamis"
- &true;: logikai "igaz"

Láthatatlan jelek

- ⁢ vagy ⁢ - láthatatlan szorzás
- ⁡ vagy ⁡ - függvények ábrázolásánál láthatatlan szóköz
- ⁣ vagy ⁣ - láthatatlan szóköz
- ⁤: - láthatatlan összeadás

Egyenlet

$$x^2 + 5x + 6 = 0$$

```
<mrow>
  <mrow>
  <msup>
    <mi>x</mi>
    <mn>2</mn>
  </msup>
    <mo>+</mo>
```

Egyenlet

```
    <mrow>
      <mn>5</mn>
<mo>&InvisibleTimes;</mo>
      <mi>x</mi>
    <mrow>
    <mo>+</mo>
    <mn>6</mn>
  </mrow>
  <mo>=</mo>
  <mn>0</mn>
</mrow>
```

Intervallum

- $(0;1)$ - `<mfenced><mn>0</mn><mn>1</mn></mfenced>`
- $[0;1]$ -
  `<mfenced open="[" close="]"><mn>0</mn><mn>1</mn></mfence`
- $(0;1]$ -
  `<mfenced open="(" close="]"><mn>0</mn><mn>1</mn></mfence`
- $[0;1)$ -
  `<mfenced open="[" close=")"><mn>0</mn><mn>1</mn></mfence`

Függvény

$$f(x)$$

```
<mrow>
  <mi> f </mi>
  <mo> &ApplyFunction; </mo>
  <mrow>
    <mo> ( </mo>
    <mi> x </mi>
    <mo> ) </mo>
  </mrow>
</mrow>
```

Inverz függvény

$$f^{(-1)}$$

```
<msup>
  <mi>f</mi>
<mrow>
  <mo>(</mo>
<mn>-1</mn>
<mo>)</mo>
  </mrow>
</msup>
```

Másodfokú egyenlet megoldása

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
<mrow>
 <mi>x</mi>
  <mo>=</mo>
  <mfrac>
    <mrow>
      <mrow>
        <mo>-</mo>
        <mi>b</mi>
    </mrow>
    <mo>&PlusMinus;</mo>
```

Másodfokú egyenlet megoldása

```
<msqrt>
  <mrow>
    <msup>
      <mi>b</mi>
      <mn>2</mn>
    </msup>
    <mo>-</mo>
    <mrow>
      <mn>4</mn>
      <mo>&InvisibleTimes;</mo>
      <mi>a</mi>
      <mo>&InvisibleTimes;</mo>
      <mi>c</mi>
    </mrow>
```

Másodfokú egyenlet megoldása

```
        </mrow>
      </msqrt>
    </mrow>
    <mrow>
      <mn>2</mn>
      <mo>&InvisibleTimes;</mo>
      <mi>a</mi>
    </mrow>
  </mfrac>
</mrow>
```

Alapfüggvények

Trigonometrikus függvények példa

$$\sin(x+1)$$

A forráskód:

```
<mrow>
  <mi>sin</mi>
  <mo> &ApplyFunction; </mo>
  <mrow>
  <mo>(</mo>
<mi>x</mi>
<mo>+</mo>
    <mn>1</mn>
<mo>)</mo>
  </mrow>
</mrow>
```

Törtfüggvények

Törtfüggvény

$$\frac{1}{x^2 + \frac{x}{5}}$$

A forráskódja:

```
<mfrac>
  <mn> 1 </mn>
  <mrow>
    <msup><mi> x </mi><mn> 2 </mn></msup>
    <mo> + </mo>
    <mfrac><mi> x </mi><mn> 5 </mn></mfrac>
  </mrow>
</mfrac>
```

Törtfüggvények

Binomiális együttható

$$\binom{n}{k}$$

A forráskódja:

```
<mrow>
  <mo> ( </mo>
  <mfrac linethickness="0">
    <mi> a </mi>
    <mi> b </mi>
  </mfrac>
  <mo> ) </mo>
</mrow>
```

Sorösszeg

Sorösszeg megjelenítése:

$$\sum_{x=a}^{b} f(x)$$

Forráskódja

```
<mrow>
<munderover>
  <mo>&#x2211;</mo>
  <mrow><mi>x</mi><mo>=</mo><mi>a</mi></mrow>
  <mi>b</mi>
</munderover>
<mrow><mi>f</mi><mo>&#x2061;</mo>
  <mfenced><mi>x</mi></mfenced></mrow>
</mrow>
```

Integrálszámítás

Integrál példa!

$$\int_0^2 e^x \, \mathrm{d}x$$

Forráskódja

```
<mrow>
  <msubsup>
    <mo> &int; </mo>
    <mn> 0 </mn>
    <mn> 2 </mn>
  </msubsup>
```

Integrálszámítás

```
<mrow>
  <msup>
    <mi> &ExponentialE; </mi>
    <mi> x </mi>
  </msup>
  <mo> &InvisibleTimes; </mo>
  <mrow>
    <mo> &DifferentialD; </mo>
    <mi> x </mi>
  </mrow>
</mrow>
</mrow>
```

Mátrixok

3×3-as egységmátrix

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

A képletet leíró formula

```
<mrow>
<mo> ( </mo>
<mtable>
<mtr>
<mtd> <mn>1</mn> </mtd>
<mtd> <mn>0</mn> </mtd>
<mtd> <mn>0</mn> </mtd>
</mtr>
```

Mátrixok

```
<mtr>
<mtd> <mn>0</mn> </mtd>
<mtd> <mn>1</mn> </mtd>
<mtd> <mn>0</mn> </mtd>
</mtr>
<mtr>
<mtd> <mn>0</mn> </mtd>
<mtd> <mn>0</mn> </mtd>
<mtd> <mn>1</mn> </mtd>
</mtr>
</mtable>
<mo> ) </mo>
</mrow>
```

Mátrix zárójelezés

$$A = \begin{bmatrix} x & y \\ z & w \end{bmatrix}$$

```
<mrow>
   <mi>A</mi>
   <mo>=</mo>
   <mfenced open="[" close="]">
      <mtable>
         <mtr>
            <mtd><mi>x</mi></mtd>
            <mtd><mi>y</mi></mtd>
         </mtr>
```

Mátrix zárójelezés

```
            <mtr>
               <mtd><mi>z</mi></mtd>
               <mtd><mi>w</mi></mtd>
            </mtr>
         </mtable>
      </mfenced>
</mrow>
```

Matematikai programcsomagok
A MathMl szabvány - Tartalmi jelölés

Dr. Hartung Ferenc - Dr. Virágh János - Pozsgai Tamás

2014. május 3.

Tartalomjegyzék

Kifejezés-fák és token-elemek

Tartalmi jelölés

Körülbelül 75 tartalmi jelölő elem van, amelyek közel egy tucat attribútummal rendelkeznek. A következőképpen csoportosíthatjuk őket:

- üres elemek (<partialdiff/>, <leq/>, <tan/>)
- matematikai attribútumok kódolására használt elemek (<matrix>, <set>)
- új matematikai objektumok létrehozására használt elemek (<apply>)

A tartalmi jelölés token elemei

<ci>

Azonosítók ("content identifier")

<cn>

Számok ("content number")

Az apply elem

Az <apply> elemhasználatával argumentumok csoportjából függvényt alkothatunk. Az utódok sorrendje fontos. Az első jelzi a függvényt, a többi annak argumentumait.
Az alábbi kód az **a-b** kifejezést írja le.

```
<apply>
  <minus/>
    <ci>a</ci>
    <ci>b</ci>
</apply>
```

MathML alapjai

Mennyiségek előre meghatározása:

- <**bvar**>
- <**lowlimit**>

Ezek mellé használjuk:

- <**diff**/>
- <**int**/>

A <declare> elemet akkor használjuk, ha egy változó típusát előre megakarjuk adni. Azokban az esetekben használjuk, ha egy kifejezést ki szeretnénk értékelni.

Hatványfüggvény

$$x^3$$

Forráskódja tartalmi jelöléssel:

```
<apply><power/><ci>x</ci><cn>3</cn></apply>
```

Összehasonlításként a megjelenítő jelölés:

```
<msup><mi>x</mi><mn>3</mn></msup>
```

Gyökfüggvény

$$\sqrt[n]{x}$$

Forráskódja:

```
<apply><root/>
  <degree>
  <ci type="integer">n</ci>
</degree>
  <ci>x</ci>
</apply>
```

Egyenlet

$$x^2 + 2x + 1 = 0$$

Forráskódja:

```
<reln>
  <eq/>
  <apply>
    <plus/>
    <apply>
      <power/>
      <ci>x</ci>
      <cn>2</cn>
```

Egyenlet folytatása

```
    </apply>
    <apply>
      <times/>
      <cn>2</cn>
      <ci>x</ci>
    </apply>
    <cn>1</cn>
  </apply>
  <cn>0</cn>
</reln>
```

A <**reln**> elem helyett általában használható az <**apply**> elem is, kivéve abban az estben, amikor a műveleti jel valamilyen tényleges relációt ír le.

Másodfolú megoldóképlet

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
<reln>
   <eq/>
   <ci>x</ci>
   <apply>
      <divide/>
      <apply>
         <fn>
            <mo>\&PlusMinus;</mo>
         </fn>
         <apply>
```

Másodfokú megoldóképlet folytatása

```
      <minus/>
      <ci>b</ci>
   </apply>
   <apply>
      <root/>
      <apply>
         <minus/>
         <apply>
            <power/>
            <ci>b</ci>
            <cn>2</cn>
         </apply>
         <apply>
            <times/>
```

Másodfokú megoldóképlet folytatása

```
                <cn>4</cn>
                <ci>a</ci>
                <ci>c</ci>
              </apply>
            </apply>
            <cn>2</cn>
          </apply>
        </apply>
        <apply>
          <times/>
          <cn>2<cn>
          <ci>a</>
        </apply>
      </apply>
```

Esetszétválasztásos függvény

$$f(x) = \begin{cases} -x & \text{if x<0} \\ 0 & \text{if x=0} \\ x & \text{if x>0} \end{cases}$$

Esetszétválasztásos függvény

A forráskód:

```
<piecewise>
  <piece>
      <apply><minus/><ci>x</ci></apply>
      <apply><lt/><ci>x</ci><cn>0</cn></apply>
  </piece>
  <piece>
      <cn>0</cn>
      <apply><eq/><ci>x</ci><cn>0</cn></apply>
  </piece>
  <piece>
      <ci>x</ci>
      <apply><gt/><ci>x</ci><cn>0</cn></apply>
  </piece>
```

Abszolútérték függvény

$$|x|$$

A forráskód:

```
<apply>
  <abs/>
  <ci>x</ci>
</apply>
```

Logaritmus függvény

$$\log_2(x)$$

A forráskód:

```
<apply>
        <log/>
        <logbase>
              <cn> 2 </cn>
        </logbase>
        <ci> x </ci>
</apply>
```

Az e^x függvény

Az e^x függvényt leíró utasítások:

```
<apply>
  <exp/>
  <ci>x</ci>
</apply>
```

Összetett függvény

$$(f \circ g)(x) = f(g(x))$$

A forráskód:

```
<apply><eq/>
  <apply>
    <apply><compose/>
  <ci>f</ci><ci>g</ci>
    </apply>
    <ci>x</ci>
  </apply>
  <apply><ci>f</ci>
  <apply><ci>g</ci><ci>x</ci>
    </apply>
  </apply>
```

Trigonometrikus függvény

$$\sin(\cos x + x^2)$$

A forráskód:

```
<apply><sin/>
  <apply><plus/>
    <apply><cos/>
  <ci>x</ci>
</apply>
    <apply><power/>
  <ci>x</ci>
<cn>2</cn>
</apply>
  </apply>
</apply>
```

Határértékszámítás

Függvényhatárérték

$$\lim_{x \to 0} \sin(x)$$

A forráskód

```
<apply>
  <limit/>
    <bvar><ci>x</ci></bvar>
    <lowlimit><cn>0</cn></lowlimit>
    <apply>
  <sin/>
  <ci>x</ci>
</apply>
</apply>
```

Határértékszámítás

Függvényhatárérték

$$\lim_{x \to 0^+} \sin(x)$$

A forráskód

```
<apply><limit/>
  <bvar><ci>x</ci></bvar>
  <condition>
    <apply><tendsto type="above"/>
  <ci>x</ci><cn>0</cn>
</apply>
  </condition>
  <apply>
  <sin/><ci>x</ci>
</apply>
</apply>
```

Sorösszegszámítás

Sorösszeg

$$\sum_{i=2}^{\infty} i^3$$

A képletet leíró formula

```
<apply>
<sum/>
  <bvar><ci>i</ci></bvar>
  <lowlimit><cn>2</cn></lowlimit>
  <uplimit>
  <ci><infinity/></ci>
  </uplimit>
  <apply>
  <power/>
<ci>i</ci><cn>3</cn>
```

Integrálszámítás

4 Határozott integrál megjelenítése.

$$\int_0^1 2^x \mathrm{d}x$$

A képletet leíró formula:

```
<apply><int/>
   <bvar><ci>x</ci></bvar>
   <lowlimit><cn>0</cn></lowlimit>
   <uplimit><cn>1</cn></uplimit>
   <apply><power/><cn>2</cn><ci>x</ci></apply>
</apply>
```

Vektorok

$$\begin{pmatrix} 2 \\ 3 \\ 7 \end{pmatrix}$$

A képletet leíró formula

```
<vector>
   <cn>2</cn>
   <cn>3</cn>
   <cn>7</cn>
</vector>
```

Mátrix

$$A = \begin{pmatrix} x & y \\ z & w \end{pmatrix}$$

MathML kód:

```
<reln>
   <eq/>
   <ci>A</ci>
   <matrix>
      <matrixrow>
         <ci>x</ci>
         <ci>y</ci>
      </matrixrow>
```

MathML alapjai
Példák: tartalmi jelölés

```
        <matrixrow>
            <ci>z</ci>
            <ci>w</ci>
        </matrixrow>
    </matrix>
</reln>
```

A mátrix elem tartalmazza a zárójeleket, ezért azokat nem kell külön deklaráni.

Mátrixok

3×3-as egységmátrix

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

A képletet leíró formula

```
<matrix>
    <matrixrow>
        <cn> 0 </cn>
        <cn> 1 </cn>
        <cn> 0 </cn>
    </matrixrow>
    <matrixrow>
        <cn> 0 </cn>
        <cn> 0 </cn>
```

Az és és a vagy reláció

Az $a \vee b$ relációt leíró utasítások:

```
<apply><and/>
  <ci>a</ci><ci>b</ci>
</apply>
```

Az $a \wedge b$ relációt leíró utasítások: I

```
<apply><or/>
  <ci>a</ci><ci>b</ci>
</apply>
```

A negáció és az implikáció reláció

A $\neg a$ relációt leíró utasítások:

```
<apply><not/>
  <ci>a</ci>
</apply>
```

Az $a \Rightarrow b$ relációt leíró utasítások: I

```
<apply><implies/>
  <ci>a</ci><ci>b</ci>
</apply>
```

Halmaz megadása

Az $\{x|(x < 5 \vee x \in \mathbb{N}\}$ halmazt leíró utasítások:

```
<set>
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply><and/>
      <apply><lt/><ci>x</ci><cn>5</cn></apply>
      <apply><in/>
  <ci>x</ci><naturalnumbers/>
</apply>
    </apply>
  </condition>
    <ci>x</ci>
</set>
```

Részhalmaz

Az $A \subset B$-t leíró utasítások:

```
<apply><subset/>
  <ci>A</ci><ci>B</ci>
</apply>
```

Az $A \not\subset B$-t leíró utasítások:

```
<apply><notsubset/>
  <ci>A</ci><ci>B</ci>
</apply>
```

Halmazok uniója és metszete

Az $A \cap B$-t leíró utasítások:

```
<apply><union/>
  <ci>A</ci><ci>B</ci>
</apply>
```

Az $A \cup B$-t leíró utasítások:

```
<apply><intersect/>
  <ci>A</ci><ci>B</ci>
</apply>
```

Halmazok különbsége

Az $A \setminus B$-t leíró utasítások:

```
<apply><setdiff/>
<ci type="set">A</ci>
<ci type="set">B</ci>
</apply>
```

Jelölések ötvözése

$$\int_0^t \frac{\mathrm{d}x}{x}$$

```
<sematics>
<mrow>
    <msubsup>
        <mo>\&int;</mo>
        <mn>0</mn>
        <mi>t</mi>
    </msubsup>
    <mfrac>
```

Jelölések ötvözése folytatás

```
        <mrow>
            <mo>\&dd;</mo>
            <mi>x</mi>
        </mrow>
        <mi>x</mi>
    </mfrac>
</mrow>
<annotation-xml encoding="MathML-Content">
    <apply>
        <int/>
        <bvar>
            <ci>x</ci>
        </bvar>
```

Jelölések ötvözése folytatás

```
            <lowlimit>
               <cn>0</cn>
            </lowlimit>
            <uplimit>
               <ci>t</ci>
            </uplimit>
            <apply>
               <divide/>
               <cn>1</cn>
               <ci>x</ci>
            </apply>
         </apply>
      </annotation-xml>
   </sematics>
```

<semantics> elem

<semantics>

Jelzi, hogy a megjelenítő jelöléshez tartozik egy szemantikai megjegyzés ami tartalmi. Első utóda az a kifejezés, amihez a megjegyzést fűzzük, a másik maga a megjegyzés.

A megjegyzés bármilyen típusú lehet, így az <annotation> elemben meg kell adni annak típusát.

Matematikai programcsomagok
9. előadás
Az OpenMath szabvány

Dr. Hartung Ferenc - Dr. Virágh János - Pozsgai Tamás

2014. május 2.

OpenMath

Az **OpenMath** egy leíró nyelv, amellyel matematikai formulákat jeleníthetünk meg webes felületen.

- Kifejlesztése 1993-ban indult.
- Európai fejlesztésű nyelv.
- Létrejött az *OpenMath Society* ami az első specifikációt adta ki.
- Első kiadás: 1996.
- 1997-ben vált az EU egyik szabványává (EU Negyedik Keretprogramjának hatására).
- 2002 1.1 verzió
- 2004 2.0 verzió, jelenleg is érvényes

Kódolás

Az *OpenMath* nyelvnek kétféle kódolási formája van:

- XML formátumú kódolás
- ún. bájtfolyam típusú

Az XML formátumú kódolás.

- emberi megértésre alkalmas
- hátránya: elég terjedelmes

XML mód

```
<OMOBJ>
  <OMA>
    <OMS cd="arith" name="times"/>
    <OMA>
      <OMS cd="arith" name="plus"/>
      <OMV name="x"/>
      <OMV name="y"/>
    </OMA>
    <OMA>
      <OMS cd="arith" name="plus"/>
      <OMV name="x"/>
      <OMV name="z"/>
    </OMA>
  </OMA>
</OMOBJ>
```

Bájtfolyam mód

- Minden XML-kódolású *OpenMath* szimbólumnak van bájtokban leírt megfelelője.
- Rövidebb mint az XML kód.
- Hátránya: az ember számára kevésbé könnyen érhető formátum.

Az előző példa bájtfolyam típussal:

```
18 10 08 05 05 61 72 69 74 68 74 69 6d 65 73 10 08 05 04 61
72 69 74 68 70 6c 75 73 05 01 78 05 01 79 11 10 48 01 45 00
05 01 7a 11 11 19
```

Bájtfolyam mód

A példa szerint látjuk, hogy a bájtfolyam típussal rövidebben írhatjuk le a matematikai formulákat. Minden számnak külön jelentése van, amiből később az *OpenMath* képletet jelenít meg. A kód elején lévő 18-as, például az objektum kezdetét jelöli, míg a 19-es a kódsorozat végén az objektum végét.
Az eredeti képletünk:

$$(x + y)(x + z)$$

MathML vs *OpenMath*

- Az *OpenMath* sokkal kevesebb elemet használ.
- A MathMl-ben megtalálható elemek többségét attribútumként adja meg.
- Az *OpenMath* case-sensitive, azaz kis- és nagybetűérzékeny.
- A *Content Dictionary (CD)*, amiben az elemeknél megadott attribútumok vannak meghatározva.
- Mindkettő fastruktúrán alapul.

frametitleAz OpenMath alapjai

Alapobjektumok:

- egészek
- lebegőpontos számok
- szimbólumok: CD-ban meghatározott elem, név szerint hivatkozunk rá (csak betűk, számok és _ jel)
- változók: programozási nyelvekhez hasonlóan használhatjuk őket, nevük speciális jeleket is tartalmazhat (pl. α)
- sztringek: Unicode karakterek
- bájttömbök: nem kell értelmezhetőnek lennie

Az OpenMath alapjai

Összetett objektumok:

- alkalmazás (application): első rész - **fejobjektum**, második rész - **argumentumobjektum** (matematikai függvényeknek felel meg)
- kapocs (binding): min. két rész; első rész - **kötő objektum**, többi rész - **argumentumobjektum**
- tulajdonság (attribution): szimbólum (adott obj. attribútuma) és valamilyen más obj. párok
- hiba (error): nincs közvetlen matematikai jelentése, alkalmazásokkal történő feldolgozások közben történő hibák kiértékelésére szolgál

A Content Dictionary és az OmDoc

A Content Dictionary (CD) egy fogalomtár, amiben összegyűjtve találjuk meg a különböző matematikai fogalmakat, ami nagyban megkönnyíti az OpenMath alkalmazások használatát. Az alkalmazásokat másképpen akár matematikai függvényeknek is nevezhetjük.
Az *OpenMath* másik jelentős dokumentációja az OmDoc, ami tulajdonképpen az *OpenMath* egyik bővítménye, ami az általános kommunikációt támogatja, beleértve a matematikai funkciók megjelenítését, például levelekben, e-könyvekben, e-mailekben matematikai/számítási szolgáltatások között, és így tovább... Leíró nyelvként, az OmDoc támogatást nyújt számos olyan struktúrához, amelyeket matematikusok használnak könyvekben, leírásokban tételek, bizonyítások megfogalmazásra.

Tokenelemek *OpenMath*-ban

Számok

OpenMath-ban TEX-hez és MathMl-hez képest több lehetőségünk van a számok ábrázolására.

Példák számok megjelenítésére

27732.3

```
<OMF dec="27732.3"/>
```

27732

```
<OMI>27732</OMI>
```

$BCF4_{16}$

```
<OMI>xBCF4</OMI>
```

π

```
<OMS cd="nums" name="pi"/>
```

Azonosítók

Valamilyen matematikai elem azonosítói. *OpenMath*-ban az <OMV/> elemet használjuk az azonosítók megadására.

Példák azonosítók megjelenítésére

z

```
<OMV name="z"/>
```

k_n

```
<OMA>
  <OMATTR>
  <OMATP>
 <OMS cd="presentation" name="style"/>
 <OMSTR> subscripted <OMSTR>
</OMATP>
<OMV name="k"/>
</OMATTR>
<OMV name="n"/>
</OMA>
```

Tartalmi elemek csoportosítása

Egy elem, egy függvény vagy művelet argumentumaival való megjelenítsére szolgál az <OMA> elem.

- első argumentum: az a művelet, amit megvalósítanánk
- másodiktól: a művelet argumentumai

Példa:
7!

```
<OMA>
   <OMS cd="arith" name="factorial"/>
 <OMI>7</OMI>
</OMA>
```

Deklarációk *OpenMath*-ban

A deklarációs elemeknek két fontos szerepe van:

- megváltoztassuk, megadjuk az alapértelmezését egy edogg kifejezésnek
- kapcsolatot létesítsünk egy név és objektum között.

Tipikusan egy számítógépes algebra esetén használhatjuk. <OMS/> és <OMV/> elemek használatosak erre a célra
Példa:
$f(x) = \ln x$

Példa deklarációra

```
<OMA>
<OMS cd="relation" name="eq"/>
<OMA>
<OMS cd="arith" name="times"/>
<OMV name="f"/>
<OMMATR>
<OMATP>
<OMS cd="presentation" name="left"/>
<OMSTR>(</OMSTR>
<OMS cd="presentation" name="right"/>
<OMSTR>)</OMSTR>
</OMATP>
<OMV name="x"/>
</OMATTR>
</OMA>
```

Példa deklarációra (folyt.)

```
<OMA>
<OMS cd="transc" name="ln"/>
<OMV name="x"/>
</OMA>
</OMA>
<OMA>
```

Aritmetika, algebra, logika

Alapműveletek

OpenMath-ban az alapműveleteket a <**OMS**/> elem megfelelő paraméterezésével helyettesítjük, a "name" attribútum "minus", "plus", "times", "divide" paraméterekre állításával. A szükséges CD minden esetben az "arith".

Példa:

$x - y - z$

```
<OMA>
<OMS cd="arith" name="minus"/>
<OMV name="x"/>
<OMV name="y"/>
<OMV name="z"/>
</OMA>
```

További példák

$y + z$

```
<OMA>
<OMS cd="arith" name="plus"/>
<OMV name="z"/>
<OMV name="x"/>
</OMA>
```

$c \cdot d$

```
<OMA>
<OMS cd="arith" name="times"/>
<OMV name="c"/>
<OMV name="d"/>
</OMA>
```

Aritmetika, algebra, logika

Egyéb műveletek

<OMS/> elem megfelelő paraméterezésével adhatjuk meg őket

Jelölések: **'power'** - hatványozás, **'root'** - n-edik gyök, **'quotient'** - osztás egészrésze, **'abs'** - abszolútérték, **'gcd'** - legnagyobb közös osztó, **'mod'** - maradékos osztás

Példa:

m^n

```
<OMA>
<OMS cd="arith" name="power"/>
<OMV name="m"/>
<OMI> n </OMI>
</OMA>
```

Aritmetika, algebra, logika

Algebrai, logikai és aritmetikai összefüggések
<OMS/> elem megfelelő paraméterezésével adhatjuk meg őket
Jelölések: **'max'** - maximum függvény, **'min'** - minimum függvény, **'ln'** - logaritmus naturalis függvény, **'log'** - 'a' alapú logaritmusfüggvény, **'exp'** - exponenciális függvény
Példa:
ln a

```
<OMA>
<OMS cd="transc" name="ln"/>
<OMV name="a"/>
</OMA>
```

Aritmetika, algebra, logika

Logikai műveletek

<OMS/> elem megfelelő paraméterezésével adhatjuk meg őket

Jelölések: **'and'** - és, **'or'** - vagy, **'xor'** - kizáró vagy, **'not'** - nem, **'implies'** - implikáció, **'forall'** - minden -ra/re kifejezéshez, **'exists'** - "létezik, hogy..."

Példa:

$A \Rightarrow B$

```
<OMA>
<OMS cd="logic" name="implies"/>
<OMV name="A"/>
<OMV name="B"/>
</OMA>
```

Aritmetika, algebra, logika

Relációk

<OMS/> elem megfelelő paraméterezésével adhatjuk meg őket
Jelölések: **'eq'** - egyenlő, **'neq'** - nem egyenlő, **'gt'** - nagyobb, mint.., **'lt'** - kisebb, mint..., **'geq'** - nagyobb vagy egyenlő, **'leq'** - kisebb vagy egyenlő
Példa:
$c < d$

```
<OMA>
<OMS cd="relation" name="lt"/>
<OMV name="c"/>
<OMV name="d"/>
</OMA>
```

Függvénytan

Felhasználói függvények

OpenMath-ban egy <OMA> és egy <OMS/> elem kapcsolatával fejezhetjük ki, ahol az <OMS/> elemben írjuk le magát a felhasználói függvényt. Több, egymásba ágyazott függvény esetén az <OMA> elem helyett használhatjuk az <OMBIND>, <OMBVAR> elempárt is, mely magát az egymásba ágyazást jelöli.

Példa felhasználói függvényre

$f + g$

```
<OMA>
  <OMS cd="arith" name="plus"/>
  <OMA>
    <OMS cd="ecc" name="type"/>
    <OMS cd="ecc" name="function"/>
    <OMV name="f"/>
  </OMA>
  <OMA>
    <OMS cd="ecc" name="type"/>
    <OMS cd="ecc" name="function"/>
    <OMV name="g"/>
  </OMA>
</OMA>
```

Függvénytan

A lambda függvény

A lambda elemet egy olyan felhasználó által meghatározott függvény esetében használjuk, amelyet egy adott kifejezésből és egy vagy több független változóból készítünk.

A lambdát *OpenMath*-ban az <OMBIND> - <OMBVAR/> elempárossal írhatjuk le, a szükséges CD, az "fns".

Példa:

λ

```
<OMBIND>
<OMS cd="fns" name="lambda"/>
</OMBIND>
```

Függvénytan

Függvényekkel végzett műveletek

Megadhatjuk a függvények inverzét, ekkor az <OMS/> elem megfelelő paramétereihez az "inverse" parancsot kell megadnunk.

Példa:

f^{-1}

```
<OMA>
<OMS cd="fns" name="'inverse"/>
<OMV name="f"/>
</OMA>
```

Függvénytan

Függvényekkel végzett műveletek

Megadhatjuk még a függvények kompozícióját, vagy összetételét is. Ehhez is az <OMS/> elemet kell használnunk, a "left_compose" beállítással.

Példa:

$$(f \circ g)(x)$$

Függvénytan

```
<OMA>
  <OMS cd="fns" name="compose"/>
  <OMA>
    <OMATTR>
      <OMATP>
        <OMS cd="ecc" name="type"/>
        <OMS cd="ecc" name="function"/>
      </OMATP>
      <OMV name="f"/>
    </OMATTR>
  </OMA>
  <OMA>
    <OMATTR>
      <OMATP>
        <OMS cd="ecc" name="type"/>
        <OMS cd="ecc" name="function"/>
      </OMATP>
      <OMV name="g"/>
    </OMATTR>
  </OMA>
  <OMV name="x"/>
```

Integrál- és differenciálszámítás

Integrálszámítás

OpenMath-ban az integrál és intervallumának leírására az <OMS/> elemet használhatjuk, a megfelelő paraméterezéssel (a name attribútum defint, interval; halmazként való megadás esetén lambda beállításával).

Példa:

$$\int_a^b f(x)\,\mathrm{d}x$$

Integrál- és differenciálszámítás

```
<OMA>
  <OMS cd="calculus" name="defint"/>
  <OMA>
    <OMS cd="interval" name="interval"/>
    <OMV name="a"/>
    <OMV name="b"/>
  </OMA>
  <OMA>
    <OMATTR>
      <OMATP>
        <OMS cd="ecc" name="type"/>
        <OMS cd="ecc" name="function"/>
      </OMATP>
      <OMV name="f"/>
    </OMATTR>
    <OMV name="x"/>
  </OMA>
```

Differenciálszámítás

Az <OMS/> elem megfelelő paraméterezésével (a name attribútum diff vagy partialdiff beállításával)
Példa: $\frac{d}{dx} f(x)$

```
<OMA>
  <OMS cd="calculus" name="diff"/>
  <OMV name="x"/>
  <OMA>
    <OMATTR>
      <OMATP>
        <OMS cd="ecc" name="type"/>
        <OMS cd="ecc" name="function"/>
      </OMATP>
      <OMV name="f"/>
    </OMATTR>
    <OMV name="x"/>
  </OMA>
```

Határérték

Hatérérték számítása

Az <OMS/> elem megfelelő paraméterezésével (a name attribútum limit, sum, product beállításával) végezzük el.
Példa: $\lim_{x \to 0} \sin x$

```
<OMA>
  <OMS cd="limit" name="limit"/>
  <OMA>
    <OMS cd="transc" name="sin"/>
    <OMV name="x"/>
  </OMA>
</OMA>
```

Halmazelmélet

Halmazok megadása

Az <OMS/> elem megfelelő paraméterezésével (a name attribútum set, list, in, notin beállításával) végezzük el.
Példa: $\{a, b\}$

```
<OMATTR>
<OMATP>
<OMS cd="presentation" name="left"/>
<OMSTR>{</OMSTR>
<OMS cd="presentation" name="right"/>
<OMSTR>}</OMSTR>
</OMATP>
<OMA>
<OMS cd = "set" name="set"/>
<OMV name="a"/>
<OMV name="b"/>
</OMA>
```

Halmazműveletek

Az <OMS/> elem megfelelő paraméterezésével (a name attribútum union, intersect, subset, notsubset, prsubset, notprsubset, setdiff beállításával) végezzük el.
Példa: $A \cup B$

```
<OMA>
<OMS cd="set" name="union"/>
<OMV name="A"/>
<OMV name="B"/>
</OMA>
```

Trigonometria

Trigonometria

Az <OMS/> elem attribútumában definiálhatjuk őket
Példa: $\cos x$

```
<OMA>
  <OMS cd="transc" name="cos"/>
  <OMV name="x"/>
</OMA>
```

OpenMath HTML-be ágyazása

Egyetlen elem van a beágyazáshoz: <OMOBJ/>, ez zár közre minden OpenMath nyelvű szövegrészt az adott HTML dokumentumban

- az <OMOBJ/> nem egymásba ágyazható elem
- korlátlan argumentumai lehetnek
- a META elem beállítása HTML oldalaknál erősen ajánlott (<META content-math-type="text/OpenMath">)

Az OpenMath előnyei, hátrányai

- könnyen, egyszerűen bővíthetők a nyelv funkciói (CD)
- a kevés elemet használó OpenMath nyelv ezáltal bonyolultá válik
- számos funkció nehezen megoldható lesz (pl. indexelés)
- nincs lehetőség stíluslapok használtára
- ábrák, képek beillesztésére nincs lehetőség

Matematikai programcsomagok (Maple)

Virágh János

2014. február 28.

7. Analízis alkalmazások
   - Határértékszámítás
   - Differenciálás
   - Függvénydiszkusszió
   - Integrálás
   - Optimalizálás
   - Differenciálegyenletek

8 Programozási alapok
- Táblák és rekordok
- Maple utasítások
- Maple eljárások
- Modulok
- Csomagkészítés

10. előadás

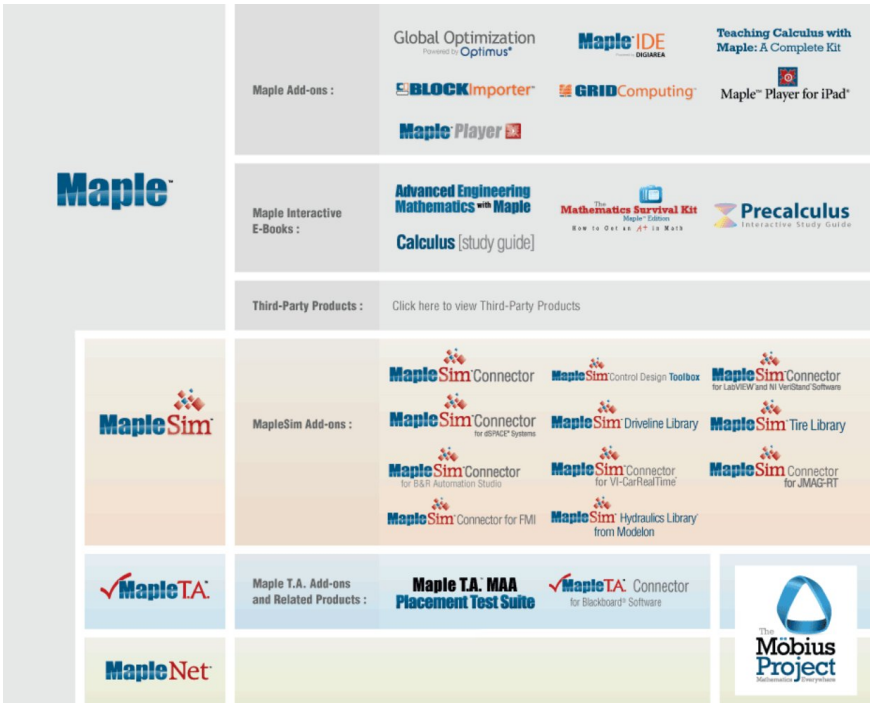
1. Dokumentumok készítése a Maple segítségével

A következő öt előadásban a Maple számítógépes algebrai rendszer használatához szükséges legfontosabb tudnivalókat vesszük sorra.
A leírások, képernyőképek a legújabb változat, a Maple 17 futtatásával készültek.
A program elérhető a Windows operációs rendszer különböző változatain, Linuxon, illetve OS X-en is.
Létezik „magyarított”, magyar menürendszerű változata. Az előadásokban az angol változatot használjuk, mivel a legtöbb elérhető dokumentáció (beleértve a rendszer beépített Helpjét) angol nyelvű, s a régebbi leírások is az angol változathoz idomultak.
A programot a Maplesoft kanadai cég forgalmazza, ezzel kapcsolatban további részleteket a cég weboldalán tudhatunk meg.

Bevezetés I.

A Maple fejlesztését az 1980-as évek elején kezdték a kanadai Waterloo egyetemen. Később a program üzleti forgalmazására, továbbfejlesztésére megalakult a Maplesoft cég, amely a Maple-re, mint „matematikai motorra” támaszkodva egyre bővülő termékpalettát kínál. Ennek fontosabb részeit láthatjuk a cég által készített összefoglalón:

Maple	szimbolikus számításokat végző („Computer Algebra”, CAS) rendszer;
MapleSim	ipari alkalmazásokhoz szükséges szimulációs rendszer;
MapleTA	matematika oktatást és számonkérést segítő rendszer;
MapleNet	a Maple webes (böngészőből való) elérése .



Bevezetés II.

Az Application Center weboldalon rengeteg hasznos információt, letölthető demót és oktatási segédanyagot találunk.

Bevezetés III.

A bőséges idegen nyelvű szakirodalom mellett több magyar Maple szakkönyv is megjelent. Bár ezek általában a program korábbi változataira támaszkodnak, a bennük leírtak legnagyobb része érvényes a mostani Maple-re is.

Heck Bevezetés a Maple használatába, JGYTF Kiadó, 1999, ISBN 9-639-16716-9;

Klincsik–Maróti Maple 8 tételben (a matematikai problémamegoldás művészetéről), Livermore Kft, 2006, ISBN 9-630-60571-6

Molnárka–Gergő–Wettl–Horváth–Kallós A Maple V. és alkalmazásai, Springer, 1996, ISBN 9-688-45589-6

Bevezetés IV.

Az előadások megértéséhez, illetve a tárgyalt anyag Maple-ben való „éles” kipróbálásához néhány technikai megjegyzés:

- az előadások elkészítéséhez a linuxon futó Maple 17 verziót használtuk;

Bevezetés IV.

Az előadások megértéséhez, illetve a tárgyalt anyag Maple-ben való „éles” kipróbálásához néhány technikai megjegyzés:

- az előadások elkészítéséhez a linuxon futó Maple 17 verziót használtuk;
- a példákban szereplő Maple inputokat egérrel átmásolva egy Maple munkalap input szekciójába és végrehajtatva az itt látható eredményeket kell kapnunk (az input elöl elhagytuk a > prompt jelet);

Bevezetés IV.

Az előadások megértéséhez, illetve a tárgyalt anyag Maple-ben való „éles” kipróbálásához néhány technikai megjegyzés:

- az előadások elkészítéséhez a linuxon futó Maple 17 verziót használtuk;
- a példákban szereplő Maple inputokat egérrel átmásolva egy Maple munkalap input szekciójába és végrehajtatva az itt látható eredményeket kell kapnunk (az input elöl elhagytuk a > prompt jelet);
- a bemutatóban alkalmazott LaTeX technológia miatt a Maple által kiírt eredmények néha eltérő fontokat és formázást használhatnak, de a matematikai tartalom azonos a diákon találhatóval;

Bevezetés IV.

Az előadások megértéséhez, illetve a tárgyalt anyag Maple-ben való „éles” kipróbálásához néhány technikai megjegyzés:

- az előadások elkészítéséhez a linuxon futó Maple 17 verziót használtuk;
- a példákban szereplő Maple inputokat egérrel átmásolva egy Maple munkalap input szekciójába és végrehajtatva az itt látható eredményeket kell kapnunk (az input elöl elhagytuk a > prompt jelet);
- a bemutatóban alkalmazott LaTeX technológia miatt a Maple által kiírt eredmények néha eltérő fontokat és formázást használhatnak, de a matematikai tartalom azonos a diákon találhatóval;
- Azokban a példákban, amelyek fájlokat olvasnak vagy írnak, a `viragh` nevű felhasználó (linuxos) `home` könyvtárában létrehozott `Maple` alkönyvtárat, vagyis a `/home/viragh/Maple` elérési utat használjuk. Windowson vagy más környezetben futtatva ezt értelemszerűen módosítani kell.

Bevezetés IV.

Az előadások megértéséhez, illetve a tárgyalt anyag Maple-ben való „éles” kipróbálásához néhány technikai megjegyzés:

- az előadások elkészítéséhez a linuxon futó Maple 17 verziót használtuk;
- a példákban szereplő Maple inputokat egérrel átmásolva egy Maple munkalap input szekciójába és végrehajtatva az itt látható eredményeket kell kapnunk (az input elöl elhagytuk a > prompt jelet);
- a bemutatóban alkalmazott LaTeX technológia miatt a Maple által kiírt eredmények néha eltérő fontokat és formázást használhatnak, de a matematikai tartalom azonos a diákon találhatóval;
- Azokban a példákban, amelyek fájlokat olvasnak vagy írnak, a `viragh` nevű felhasználó (linuxos) `home` könyvtárában létrehozott `Maple` alkönyvtárat, vagyis a `/home/viragh/Maple` elérési utat használjuk. Windowson vagy más környezetben futtatva ezt értelemszerűen módosítani kell.
- Előfordulhat, hogy egyes példák lefuttatása előtt szükséges egy `restart` utasítást kiadni – ezt helytakarékosságból néhol kihagytuk a példák elejéről.

Bevezetés V.

A Maple munkalapok szokásos beállításainak megfelelően a következő színkódokat alkalmaztuk:

- a Maple inputok piros színűek:

```
x := a + b - 1/2*sin(x)
```

Bevezetés V.

A Maple munkalapok szokásos beállításainak megfelelően a következő színkódokat alkalmaztuk:

- a Maple inputok piros színűek:
  ```
  x := a + b - 1/2*sin(x)
  ```
- az eredmények kék színben, a szokásos matematikai formázás szerint láthatók:

$$\int_0^{\pi} \sin(x)dx = 2$$

Bevezetés V.

A Maple munkalapok szokásos beállításainak megfelelően a következő színkódokat alkalmaztuk:

- a Maple inputok piros színűek:
  ```
  x := a + b - 1/2*sin(x)
  ```
- az eredmények kék színben, a szokásos matematikai formázás szerint láthatók:

$$\int_0^{\pi} \sin(x)dx = 2$$

- a szöveges formában kiírt Maple eredmények szintén kék színűek:
  ```
  Az egyenlet gyökei kerekítve: x1 = 0.6 és x2 = 1.5
  ```

Bevezetés V.

A Maple munkalapok szokásos beállításainak megfelelően a következő színkódokat alkalmaztuk:

- a Maple inputok piros színűek:
  `x := a + b - 1/2*sin(x)`
- az eredmények kék színben, a szokásos matematikai formázás szerint láthatók:

$$\int_0^{\pi} \sin(x)dx = 2$$

- a szöveges formában kiírt Maple eredmények szintén kék színűek:
  `Az egyenlet gyökei kerekítve: x1 = 0.6 és x2 = 1.5`
- a Maple hibajelzések lila színben, aláhúzva jelennek meg:
  `Error, numeric exception: division by zero`

Bevezetés V.

A Maple munkalapok szokásos beállításainak megfelelően a következő színkódokat alkalmaztuk:

- a Maple inputok piros színűek:
  `x := a + b - 1/2*sin(x)`
- az eredmények kék színben, a szokásos matematikai formázás szerint láthatók:

$$\int_0^{\pi} \sin(x)dx = 2$$

- a szöveges formában kiírt Maple eredmények szintén kék színűek:
  `Az egyenlet gyökei kerekítve: x1 = 0.6 és x2 = 1.5`
- a Maple hibajelzések lila színben, aláhúzva jelennek meg:
  `Error, numeric exception: division by zero`
- a parancsok szintaxisának ismertetésénél a Maple kulcsszavakat pirossal, a magyar nyelvű szintaktikus kategóriákat félkövér lila színnel adtuk meg:
  `solve(`**`egyenletek, ismeretlenek`**`)`

Bevezetés V.

A Maple munkalapok szokásos beállításainak megfelelően a következő színkódokat alkalmaztuk:

- a Maple inputok piros színűek:
  `x := a + b - 1/2*sin(x)`
- az eredmények kék színben, a szokásos matematikai formázás szerint láthatók:

$$\int_0^{\pi} \sin(x)dx = 2$$

- a szöveges formában kiírt Maple eredmények szintén kék színűek:
  `Az egyenlet gyökei kerekítve: x1 = 0.6 és x2 = 1.5`
- a Maple hibajelzések lila színben, aláhúzva jelennek meg:
  `Error, numeric exception: division by zero`
- a parancsok szintaxisának ismertetésénél a Maple kulcsszavakat pirossal, a magyar nyelvű szintaktikus kategóriákat félkövér lila színnel adtuk meg:
  `solve(`**`egyenletek, ismeretlenek`**`)`
- a Maple dokumentációban és az angol nyelvű leírásokban használt megfelelő fogalmakat a Help formázásával összhangban lila betűkkel szedtük:
  `solve(equations,variables)`

A program működése I.

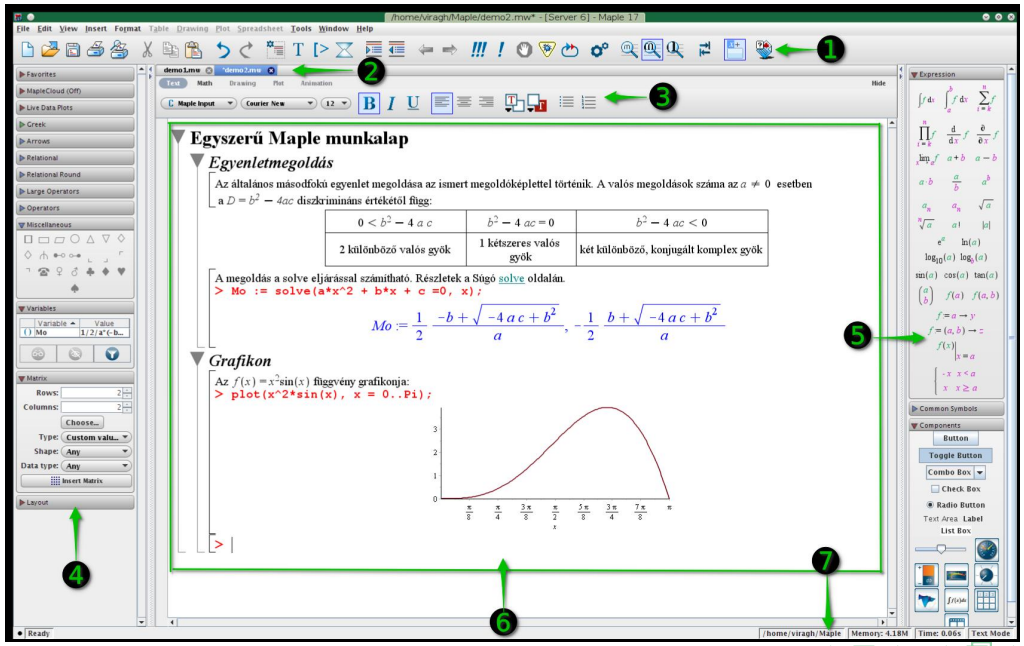
A Maple rendszer moduláris felépítésű. A rendszer „matematikai motorját”, a Maple interpretert, továbbá a működését biztosító alapvető szimbolikus átalakításokat, be- és kimenetet megvalósító segédeljárásokat C nyelven írták meg. Legelőször az ezeknek megfelelő gépi kód töltődik be a rendszer elindításakor.
A terminálból kiadott `maple` utasítás hatására a program parancssoros (konzolos) felületű változata indul el. Ezt látjuk az alábbi ábrán.

```
[viragh@localhost Maple]$ maple
    |\^/|     Maple 17 (X86 64 LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2013
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> solve(x^2-5*x-2=0);
                                                    1/2         1/2
                                                  33          33
                                          5/2 + -----, 5/2 - -----
                                                  2           2

> is( 2*sin(x)*cos(x) = sin(2*x) );
                                                     true

> Sum( j^2, j = 1..n) = sum( j^2, j = 1..n);
                                                  n
                                                -----
                                                 \     2   (n + 1)    (n + 1)
                                                  )   j  = -------- - -------- + n/6 + 1/6
                                                 /           3          2
                                                -----
                                                j = 1

> quit
memory used=3.3MB, alloc=8.3MB, time=0.11
[viragh@localhost Maple]$
```

A program működése II.

Az `xmaple` utasítás végrehajtásakor Java technológiával készült munkalapos (grafikus) felület töltődik be. Fontosabb elemeit az alábbi ábrán láthatjuk.

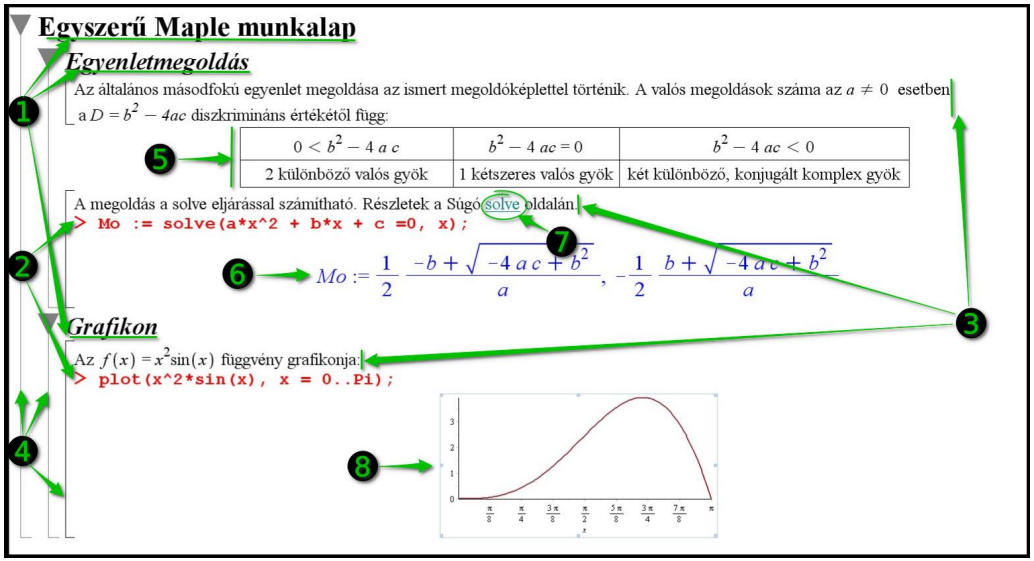
1. főmenü sor
2. a megnyitott munkalapoknak megfelelő „fülek”
3. formázási menüsor
4. bal oldali segédpaletták
5. jobb oldali segédpaletták
6. az aktív munkalap
7. állapotsor



A program működése III.

Az alábbi ábra a munkalapok legfontosabb részeit szemlélteti. A megnyitott demo munkalap több alfejezetből áll, szöveget, táblázatot, matematikai képleteket és grafikont is tartalmaz. Figyeljük meg a számozással kiemelt felületelemeket.

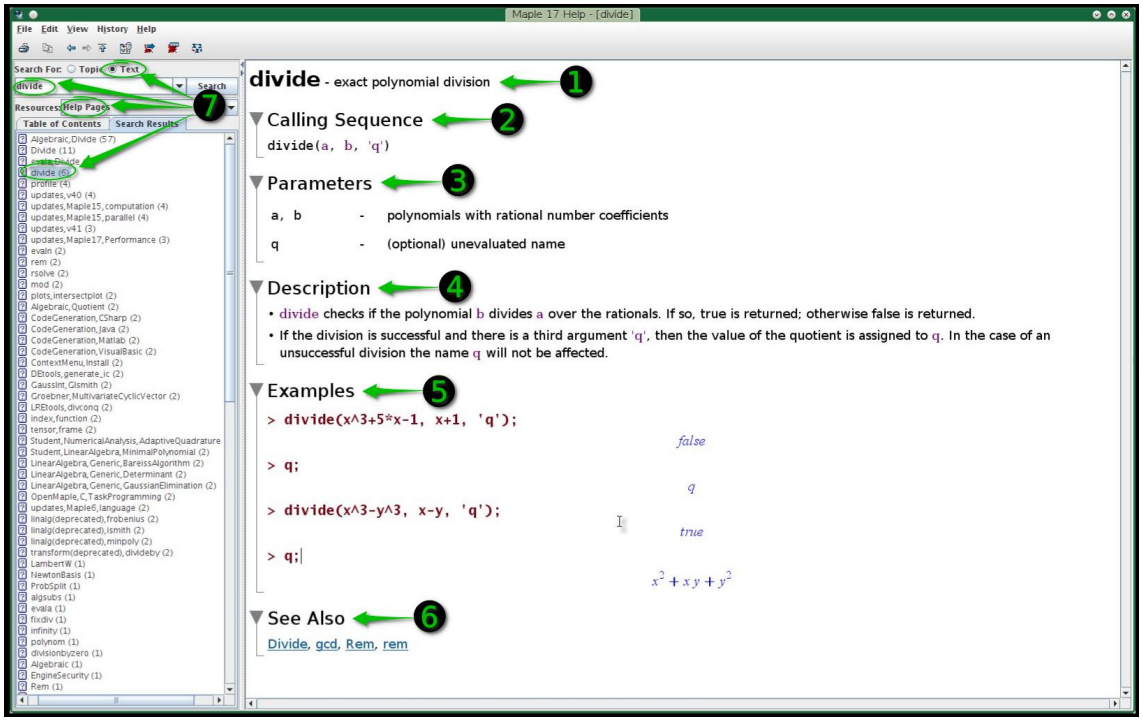
1. címsorok (fejezet - alfejezet);
2. Maple 1D input sorok (a > prompt után);
3. szöveg (paragrafusok), esetleges beágyazott 2D matematikai képletekkel, képekkel, grafikonokkal, stb.
4. a munkalap elemeinek összetartozását és elhatárolását segítő függőleges vonalak
5. táblázat;
6. az input sorok eredménye (2D matematika)
7. szövegbe ágyazott hiperlink
8. a plot parancs eredményeként kapott grafikon



A program működése IV.

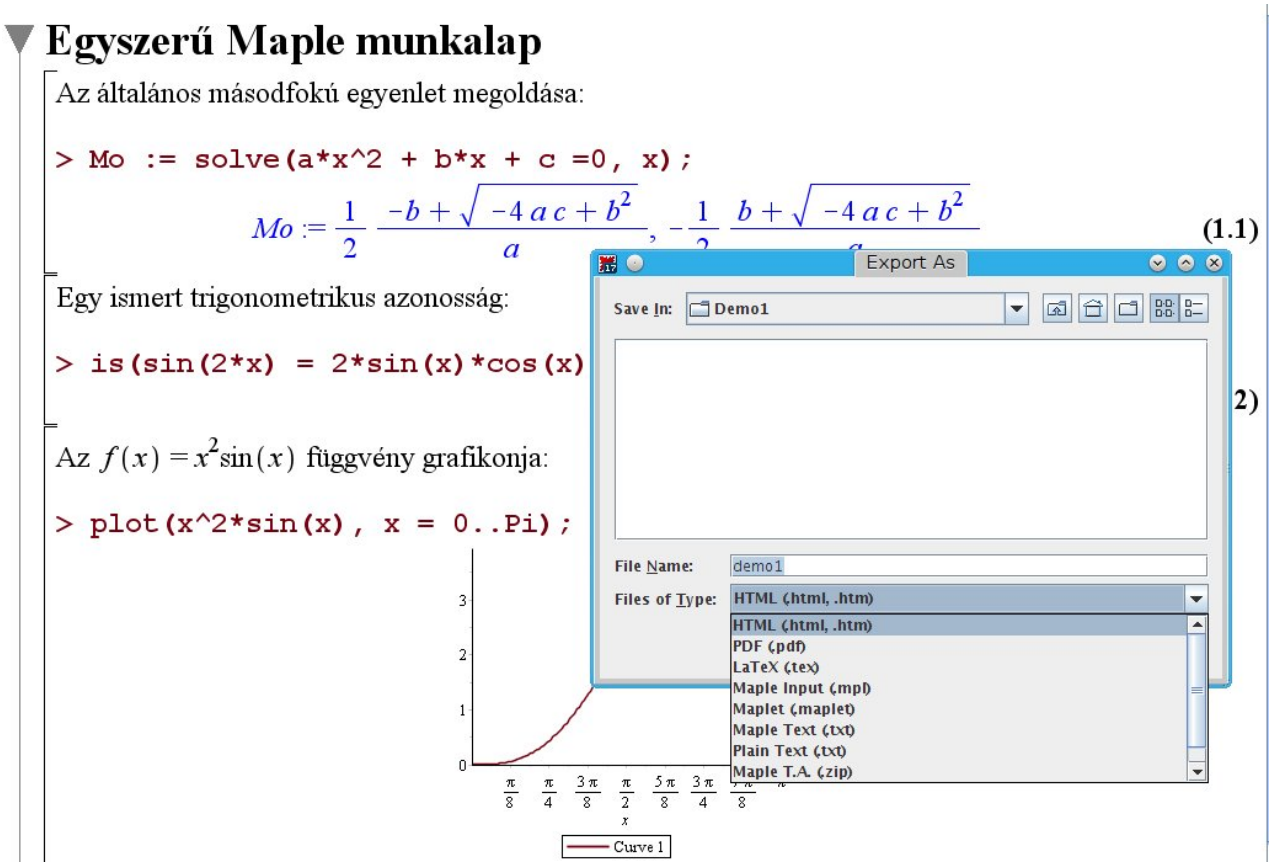
A program hasznos kiegészítője az interaktív, környezetfüggő Súgó (Help). Itt a racionális együtthatós polinomok maradékos osztását végző divide eljárást bemutató oldalt látjuk.

1. az eljárás neve és tömör leírása;
2. hívás módja;
3. a paraméterek leírása, típusa;
4. részletes ismertetés;
5. példák;
6. kapcsolódó Help oldalak;
7. teljes szöveges keresés a divide kulcsszóra, találat kiválasztása.



A program működése V.

Maple munkalapjainkat különböző formátumokba (HTML, PDF, LaTeX, mpl, TXT, RTF) exportálhatjuk a **File/Export As** menüpont segítségével. Ezek közül a fontosabbakat tekintjük át röviden. Az alábbi képen látható egyszerű munkalapot mentjük el többféle módon.



A program működése VI.

A munkalapokat HTML-be exportálva elvileg tetszőleges webböngészőben megtekinthetjük tartalmukat. Néhány technikai probléma: alapértelmezésben minden matematikai formula GIF képként kerül be a HTML fájlba, ami elég csúnya és nehezen olvasható szöveget eredményezhet. Ha MathML formátumban mentjük el a képleteket, akkor meg Java pluginre van szükség megjelenítésükhöz. Az alábbi ábra munkalapunk GIF-képes exportját mutatja.

Egyszerű Maple munkalap

Az általános másodfokú egyenlet megoldása:

```
> Mo := solve(a*x^2 + b*x + c =0, x);
```

$$Mo := \frac{1}{2}\,\frac{-b+\sqrt{-4ac+b^2}}{a},\ -\frac{1}{2}\,\frac{b+\sqrt{-4ac+b^2}}{a} \qquad (1.1)$$

Egy ismert trigonometrikus azonosság:

```
> is(sin(2*x) = 2*sin(x)*cos(x));
```

true (1.2)

Az $f(x) = x^2\sin(x)$ függvény grafikonja:

```
> plot(x^2*sin(x), x = 0..Pi);
```

A program működése VII.

A PDF export sajnos hibásan kezeli a magyar ékezetes karaktereket, helyette a **File/Print...** menü **Print To File** opciója ajánlott, amely .ps kiterjesztésű PostScript fájlba menti a munkalapot. Ennek eredményét látjuk alant:

Egyszerű Maple munkalap

Az általános másodfokú egyenlet megoldása:

```
> Mo := solve(a*x^2 + b*x + c =0, x);
```

$$Mo := \frac{1}{2}\,\frac{-b+\sqrt{-4\,a\,c+b^2}}{a},\ -\frac{1}{2}\,\frac{b+\sqrt{-4\,a\,c+b^2}}{a} \qquad (1.1$$

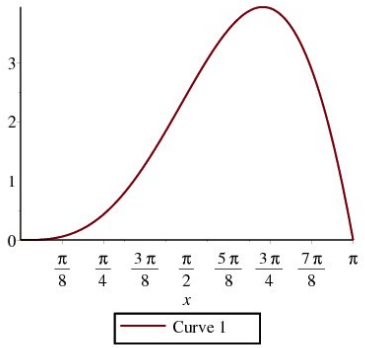
Egy ismert trigonometrikus azonosság:

```
> is(sin(2*x) = 2*sin(x)*cos(x));
```

true (1.2

Az $f(x) = x^2 \sin(x)$ függvény grafikonja:

```
> plot(x^2*sin(x), x = 0..Pi);
```



A program működése VIII.

A Maple Input (.mpl) formátumba történő export olyan szövegfájlt állít elő, amelyet végrehajtathatunk a Maple parancssoros változatával is. Ekkor kimarad a munkalapból minden output mező, s a # jel utáni Maple kommentárként kerül be minden nem-input tartalom. A parancssoros végrehajtás eredménye:

```
[viragh@localhost Demo1]$ maple demo1.mpl
    |\^/|     Maple 17 (X86 64 LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2013
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
# Egyszerű Maple munkalap
# Az általános másodfokú egyenlet megoldása:
#
> Mo := solve(a*x^2 + b*x + c =0, x);
                                        2 1/2                    2 1/2
                            -b + (-4 a c + b )       b + (-4 a c + b )
                     Mo := --------------------, - -------------------
                                   2 a                      2 a

# Egy ismert trigonometrikus azonosság:
#
> is(sin(2*x) = 2*sin(x)*cos(x));
                                          true

# Az  f(x) = x^2*sin(x); függvény grafikonja:
#
> plot(x^2*sin(x), x = 0..Pi);

 4+
  |                                                    HHHHHHHHHHHH
  +                                                 HHHH         HHH
  +                                               HH               HH
  |                                             HH                   HH
  +                                           HHH                     HH
  |                                         HH                        HH
  +                                        HH                         H
 3+                                      HH                           H
  |                                     HH                            HH
  +                                    HH                              HH
  +                                   HH                               H
  |                                  H                                  H
```

A program működése IX.

További export lehetőségek

Maple Text A Maple Input (.mpl) fájlokhoz hasonló, de ez tartalmazza az outputot is 1D formátumban.

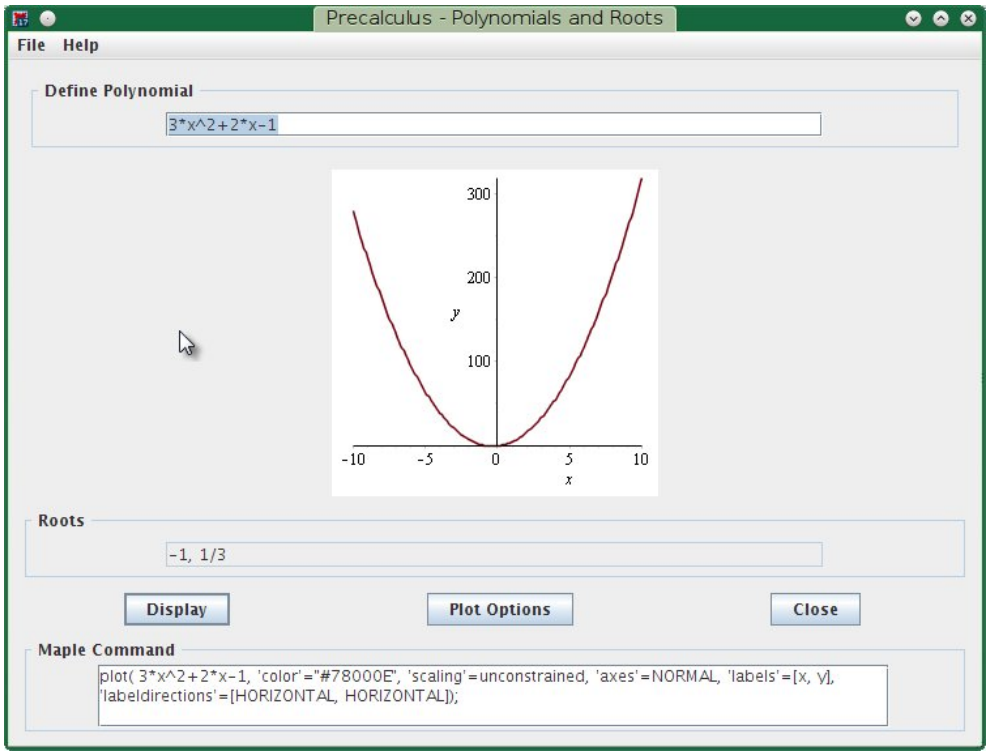
LaTeX A teljes munkalapot LaTeX formátumra konvertálja, sajnos ez az újabb Maple változatokban gyakorlatilag használhatatlan.

RTF Rich TexT Format (.rtf), ezt viszont csak az MS Office tudja helyesen megjeleníteni.

A program működése X.

A munkalapokon kívül lehetőségünk van önálló grafikus kisalkalmazások „mapletek” létrehozására és használatára.
Ezeket Java technológia segítségével futtatja a Maple, s a rajtuk található gombok, szövegdobozok, beviteli mezők segítségével interaktív, bonyolult matematikai műveleteket elvégző programok készítését teszik lehetővé.
A különböző Maple csomagok „interactive” eljárásai, illetve a menüből megnyitható bemutatók és demók nagy része így működik.

```
Student[Precalculus][PolynomialTutor]();
```



Az alábbi táblázatban összefoglaljuk a Maple leggyakrabban használt jelöléseit és konstrukcióit. Egy részükkel a későbbi előadásokban foglalkozunk részletesen.

`a+b`, `a-b`, `a*b`, `a/b`	aritmetikai műveletek: összeadás, kivonás, szorzás, osztás
`a^b`	hatványozás
`a union b`, `a minus b`, `a intersect b`	halmaz műveletek: egyesítés, különbség, metszet
`a=b, a<>b, a<b, a<=b, a>b, a>= b`	relációjelek
`not a`, `a and b`, `a or b`, `a xor b`, `a implies b`	logikai műveletek: tagadás, és, vagy, kizáró vagy, implikáció
`'a'`	kiértékeletlen kifejezés
`a[b]`	indexelt név
`a__b`	index literál
`a!`	faktoriális
`a mod b`	a modulo b
`f@g`	függvénykompozíció
`"abc"`	sztring megadása
`a||b`	sztring konkatenáció
`a:-b`	modul vagy rekord elemére hivatkozás
`a:=b`	értékadás
`a..b`	tartomány
`a::b`	típus megadása
`;`, `:`	utasítás végjelek
`%`, `%%`, `%%%`	hivatkozás előző eredményekre
`a -> b`	a nyíl operátor (függvény megadása)
`a,b,c`	kifejezéssorozat
`{a,b,c}`	halmaz
`[a,b,c]`	lista
`(a+b)*c`	zárójelezett kifejezés
`f(a,b,c)`	függvény vagy eljáráshívás
`#`	ettől kezdve a sor végéig minden megjegyzésnek számít
`?a`	az a-ra vonatkozó help oldal megnyitása

Tudnivalók I.

A továbbiakban leírtak többsége mind a parancssoros, mind a grafikus (munkalapos) felület esetén érvényesek.
A Maple parancsokat az input prompt (>) utáni sorokban adjuk meg. Egy parancsnak számít például egy matematikai kifejezés vagy egy értékadás. Mindkét felületen használható a hagyományos (1D) formátumú input, melynél a parancsokat és a matematikai kifejezéseket – a programozási nyelvekből is ismert – „linearizált” írásmódban, karakterek egymásutánjaként visszük be.
A parancsokat `;` vagy `:` zárja le, az előbbi esetben a Maple kiírja az elvégzett számítások eredményét, az utóbbinál nem.
A munkalapos felületen az eredmények a matematikában megszokott 2D formázású, kék színű output mezőkben jelennek meg.
Egy parancssor több parancsból is állhat, a végrehajtás az <Enter> megnyomásával történik. Több input sort tartalmazó végrehajtási csoportokat (`execution group`) is létrehozhatunk, ekkor az <Enter> hatására a csoport minden utasítása végrehajtódik. A menü **!!!!** ikonjára kattintva a munkalap összes parancsát végrehajtja a Maple.

```
2^3-11; b:=3: c:= 2*a*b:
```

$$-3$$

```
a; b; c;
```

$$a$$
$$3$$
$$6a$$

Tudnivalók II.

Hosszabb parancsok sokszor nem férnek ki egy sorban. A <Shift>+<Enter> billentyűkombináció anélkül kezd új sort az inputban, hogy az addigi parancsokat végrehajtaná, lényegében szóköz („üreshely”) karakterrel ekvivalens.

```
a + b; c
+ d;
```

$$3$$
$$3 + d$$

Egy szintaktikai egységen (például számon) belül \+<Enter>-rel folytathatjuk új sorban a parancsot.

```
1\
0 / 23;
```

$$\frac{10}{23}$$

Tudnivalók III.

A # karakter a megjegyzések (comment) kezdetét jelzi, az utána álló karaktereket a sor végéig figyelmen kívül hagyja a Maple interpreter.

```
a := 1; b := 2; c := 3; #d := 4;
```

$$a := 1$$
$$b := 2$$
$$c := 3$$

A %, %% és %%% speciális szimbólumokban a Maple az előző, az azt megelőző és a még eggyel előbbi parancs eredményét tárolja, ezekkel hivatkozhatunk a korábbi eredményekre.

```
%+1; %%+1; %%%+1;
```

$$4$$
$$5$$
$$6$$

Tudnivalók IV:

Ismerjünk meg néhány hasznos Maple parancsot. A `restart` utasítás hatására a Maple alapállapotba kerül, „elfelejt” minden addigi számítási eredményt.

```
a; restart: a;
```

$$1$$
$$a$$

Parancssoros felület esetén a kérdéses kulcsszó (többnyire utasítás vagy eljárásnév) elé írt kérdőjellel hívhatjuk segítségül a Helpet.
Másik formája `help(`**kulcsszó**`)` . A grafikus felületen menüből is hívható - vagy egyszerűen az <F2> billentyűvel is - utóbbi esetben arra a kulcsszóra vonatkozó információt kapunk, amin épp áll a kurzor. ?sin help(sin)

```
?sin
help(sin)
```

Tudnivalók V.

A `save` paranccsal a megadott változók értékét fájlba menthetjük. A `.mpl`-re végződő fájlnevek esetén a fájlba a Maple szintaxisa szerinti, a változók pillanatnyi értékének megfelelő értékadó utasítások kerülnek. Ha `.m` vagy `.M` kiterjesztésű fájlba mentünk, "bináris", a Maple belső ábrázolásának megfelelő fájlokat kapunk.
A `read` parancs a `save` „inverze” abban az értelemben, hogy ezzel egy korábban a `save`-vel elmentett fájlból visszaolvastathatjuk az elmentett változók értékét. Részletek: `?read` `?save`.
Néhány példa a két parancs használatára:

```
a := 1; b := c*d; x := 333;
save a,b,x, "/home/viragh/Maple/mentes.mpl";
```

$$a := 1$$
$$b := cd$$
$$x := 333$$

```
restart;
a;b;x;
```

$$a$$
$$b$$
$$x$$

```
read "/home/viragh/Maple/mentes.mpl":
a;b;x;
```

$$1$$
$$cd$$
$$333$$

Tudnivalók VI.

A munkalapos felületen használható a 2D matematikai input is, melynél a segédpaletták segítségével, egérrel is „összeklikkelhetünk” bonyolult kifejezéseket.

Az előadásokban ezt nem használjuk, mivel gyakran elfödi a hibákat, nehezen javítható.

Nézzük meg alaposan az alábbi ábra számításait .

1D input, kell szorzásjel!

```
> 2*(x+3);
```

$$2x + 6$$

2 D input, nem kell szorzásjel, de lehet is ...

```
> 2 (x + 3),  2·(x + 3)
```

$$2x + 6, 2x + 6$$

2D input, ezek vajon miért nem jók? Miért nincs output?

```
> 2 (x + 3), 2(x + 3)
```

```
> z := Pi/2;
```

$$z := \frac{1}{2}\pi$$

1D input, függvényhívás

```
> sin (z), sin(z);
```

$$1, 1$$

2D input, vajon ez hogyan jött ki? Mik ezek a kifejezések?

```
> sin (z), sin(z), sin z
```

$$\frac{1}{2}\sin\pi, 1, \frac{1}{2}\sin\pi$$

Tudnivalók VII.

Egész (`integer`) és racionális (`fraction`) számokból álló aritmetikai kifejezések kiszámításakor a Maple nagy teljesítményű, intelligens kalkulátorként működik. A számolásban nem korlátozza a számítógép beépített aritmetikája, mivel szükség esetén saját szoftveresen emulált, gyakorlatilag „végtelen pontosságú” aritmetikáját használja.

```
55!;
2^222/((3^3)^3)^3;
```

$$126964033536582759259651008475665169595803210514494367622758400 0000000000$$
$$\frac{6739986666787659948666753771754907668409286105635143120275902562 304}{7625597484987}$$

Figyeljük meg, hogy számolás közben a Maple sok automatikus egyszerűsítést végez, például a racionális számok nevezőjét mindig pozitívvá, a számlálót és a nevezőt pedig relatív prímmé alakítja,
Ügyeljünk a műveletek végrehajtási sorrendjére!

```
3*(21+57)/6*(2-505);
3*(21+57)/(6*(2-505));
```

$$-19617$$
$$\frac{-39}{503}$$

Tudnivalók VIII.

Ennél sokkal többet is tud a rendszer. Tud számolni gyökös kifejezésekkel, komplex számokkal, az ismert matematikai függvényekkel (a függvények angolul szokásos neveit használja).
Ismeri a fontos matematikai konstansokat: `Pi`, `I` (képzetes egység), `exp(1)` (a természetes alapú logaritmus alapszáma, e).

```
1+1/abs(1+sqrt(1/(1-exp(1))));
ln(exp(1))-2*arcsin(sin(1/2));
sin(Pi/2)*sqrt(-26);
tan(Pi/3)*(2/5)^(-1/3);
(3^(1/2)-1)/(sqrt(3) -1)^(5/2);
```

$$1+\frac{1}{\sqrt{1+(-1+e^{1})^{-1}}}$$

$$0$$

$$I\sqrt{26}$$

$$1/2\ \sqrt{3}2^{2/3}\sqrt[3]{5}$$

$$\left(\sqrt{3}-1\right)^{-3/2}$$

Tudnivalók IX.

A komplex számoknak megfelelő Maple adattípus neve `complex`, amely két komponensből (a valós és a komplex részből álló) összetett adattípus. A valós számokra az előbb alkalmazott műveletek, illetve a beépített függvények komplex számokra is használhatók.
Figyeljük meg az `evalc` eljárást, amely kanonikus alakra hozza a komplex számokat.

```
z1 := 1+2*I;
z2 := 2-2*I;
```

$$z1 := 1 + 2I$$
$$z2 := 2 - 2I$$

```
z1+z2;  z1-z2;  z1*z2;  z1/z2;
```

$$3$$
$$-1 + 4I$$
$$6 + 2I$$
$$-1/4 + 3/4I$$

```
z1^(-(1/2)); evalc(%);
```

$$\frac{1}{\sqrt{1+2\,I}}$$
$$1/10\,\sqrt{2+2\,\sqrt{5}}\,\sqrt{5} - 1/10\,I\,\sqrt{-2+2\,\sqrt{5}}\,\sqrt{5}$$

Tudnivalók X

A `Re`, `Im` és `conjugate` függvények szolgálnak a valós rész, a képzetes rész és a komplex konjugált kiszámítására. Az `abs` a szám abszolút értékét, az `argument` a trigonometrikus alakban szereplő szöget adja vissza. A `polar` eljárás adja trigonometrikus a alakot.

```
z := 2-3*I;
Re(z);
Im(z);
conjugate(z);
abs(z);
argument(z);
polar(z);
```

$$z := 2 - 3I$$
$$2$$
$$-3$$
$$2 + 3I$$
$$\sqrt{13}$$
$$-\arctan\left(\frac{3}{2}\right)$$
$$polar\left(\sqrt{13}, -\arctan\left(\frac{3}{2}\right)\right)$$

Tudnivalók XI.

A Maple a $\pm$ végtelennel kiterjesztett aritmetikában is tud számolni (a végtelen konstans jele `infinity`). Ekkor bizonyos műveleteknél az `undefined` értéket kapjuk.

```
1-infinity;
infinity/2*infinity;
infinity/infinity;
exp(-infinity); sqrt(infinity);
```

$$-\infty$$
$$\infty$$
$$undefined$$
$$0$$
$$\infty$$

Tudnivalók XII.

Figyeljük meg, hogy az összes eddigi példában szimbolikusan számolt a Maple: nem kerekített, nem írt ki közelítő függvényértékeket, nem használt tizedes törteket.
Ha ilyen eredményeket szeretnénk látni, használhatjuk az `eval` eljárást, amely az argumentumában megadott kifejezés lebegőpontos (közelítő) értékét adja vissza.

```
evalf(Pi);  evalf( exp(1) ); evalf( sqrt(3) );
```

$$3.141592654$$
$$2.718281828$$
$$1.732050808$$

Ha egy kifejezésben előfordul lebegőpontos szám is, akkor a Maple eleve lebegőpontosra értékeli ki a kifejezést:

```
sin(2.0); sqrt(1. + 3);
```

$$0.9092974268$$
$$2.000000000$$

Tudnivalók XIII.

Az itt látott lebegőpontos számok a Maple lebegőpontos (`float`) típusába tartoznak, amely magába foglalja a számítógépes aritmetika lebegőpontos típusát (`hfloat`) és a korábban említett szoftveres emulációval megvalósított `sfloat` típust.
Ezeket a Maple normalizált decimális alakban tárolja, s akár több százezer jegyű mantisszát és karakterisztikát is tartalmazhatnak.
A lebegőpontos számítások pontosságát a `Digits` rendszerváltozó szabja meg (alapértelmezetten 10). Az `evalf` második paramétereként – a `Digits` értékétől függetlenül – előírhatjuk, hogy milyen pontosságú értéket szeretnénk.

```
Digits := 10;
evalf(1/3);
Digits := 50;
evalf(1/3);
evalf(1/3,5);
```

$$10$$
$$0.3333333333$$
$$50$$
$$0.33333333333333333333333333333333333333333333333333$$
$$0.33333$$

Tudnivalók XIV.

Egy érdekesség: az `identify` eljárás tetszőleges `x` lebegőpontos konstans input esetén olyan pontos (egzakt) matematikai kifejezést próbál keresni, aminek értéke a megadott konstans (ha például `x=1.414213562`, akkor ez a kifejezés a `sqrt(2)`), lásd `?identify`.
Lássuk, mit tud az `identify`! Elég bátran találgat...

```
kif := 1/2*(Pi-sqrt(2) -exp(1));
evalf(kif,4); identify( evalf(kif,4));
evalf(kif,6); identify( evalf(kif,6));
evalf(kif,9); identify( evalf(kif,9));
```

$$kif := 1/2\,\pi - 1/2\,\sqrt{2} - 1/2\,e$$
$$-.4950$$
$$-\frac{7}{20}\,\sqrt{2}$$
$$-.495445$$
$$-\tan\left(\frac{23}{50}\right)$$
$$-.495451380$$
$$1/2\,\pi - 1/2\,\sqrt{2} - 1/2\,e$$

Tudnivalók XV.

Részben az előzőekhez kapcsolódik a számok kerekítése; erre a Maple a következő függvényeket biztosítja:

- `trunc` : kerekítés a 0 felé legközelebb eső egészre
- `round` : kerekítés a legközelebbi egészre
- `floor` : az adott számnál kisebb vagy egyenlő legnagyobb egész szám
- `ceil` : az adott számnál nagyobb vagy egyenlő legkisebb egész szám
- `frac` : az adott szám tört része

Figyeljük meg, hogy míg az első négy függvény visszaadott értéke mindig egy egész szám, a `frac` eredménye lehet szimbolikus kifejezés is.

```
trunc(2.4), trunc(16/7), trunc(Pi), trunc(sqrt(2)), trunc(-2.5);
round(2.4), round(16/7), round(Pi), round(sqrt(2)), round(-2.5);
floor(2.4), floor(16/7), floor(Pi), floor(sqrt(2)), floor(-2.5);
ceil(2.4),  ceil(16/7),  ceil(Pi),  ceil(sqrt(2)),  ceil(-2.5);
```

$$2, 2, 3, 1, -2$$
$$2, 2, 3, 1, -3$$
$$2, 2, 3, 1, -3$$
$$3, 3, 4, 2, -2$$

```
frac(2.4), frac(16/7), frac(Pi), frac(sqrt(2)), frac(-2.5);
```

$$0.4, 2/7, \pi - 3, \sqrt{2} - 1, -0.5$$

Tudnivalók XVI.

A Maple ismeri a `true` és a `false` logikai konstansokat, valamint a szokásos logikai műveleteket. Valójában ezek kiterjesztésével, a `{true, false, FAIL}` háromértékű logikával dolgozik. Lásd `?FAIL`.

```
not true;
false or true;
(false xor false) or true;
not FAIL; true and FAIL; false or FAIL;  FAIL or true;
```

$$false$$
$$true$$
$$true$$
$$FAIL$$
$$FAIL$$
$$FAIL$$
$$true$$

Relációkat az `evalb` („eval to boolean”) eljárással kiértékeltetve logikai értéket kapunk:

```
evalb(1=2); evalb(1<2);evalb(2 <> -2);
```

Tudnivalók XVII.

A Maple szimbolikus neveket (`name`) használ a kifejezések építőelemeiként és bizonyos utasítások részeiként. Más rendszerekben inkább a *változónév* vagy a *változó* elnevezés használatos. A továbbiakban néha mi is használjuk ezeket. A nevek képzésének szintaktikus szabálya (lásd `?name`):
a név lehet egy betű, vagy egy betű és utána tetszőleges számú betű, számjegy vagy _ karakter.
A nevek hossza akár több millió karakter is lehet. A program a nevekben is, mint mindenütt, megkülönbözteti a kis és a nagy betűket. A neveknek két fajtája van, az *indexelt nevek* és a *szimbólumok* (nem-indexelt nevek).
A rendszer belső változóinak neve az _ karakterrel és sokszor az `_Env` prefixszel kezdődik – ilyen neveket lehetőleg ne használjunk.
A nevek jelenthetik önmagukat, minden hozzájuk rendelt érték nélkül, vagy értékadó utasítással (esetleg más módon) értékül adhatunk nekik valamilyen Maple kifejezést. *Nem* csak matematikai kifejezést, hanem a később tárgyalt sztringeket, listát, táblát, stb. is!
A Maple a változó nevét egy mutatóval (pointer) együtt tárolja. Ha a változónak érteket adunk, akkor ez a mutató a memóriának arra a részére fog mutatni, ahol a program a változóhoz rendelt érteket tárolja. Ha a változóhoz még nincs érték rendelve, akkor a pointer őrá mutat.

Tudnivalók XVIII.

A Maple interpreter a kifejezések értékének kiszámításához megpróbálja rekurzívan meghatározni az egyes részkifejezésekhez és változókhoz rendelt értékeket.
Általában teljes kiértékelést (`full evaluation`) végez: a változónévtől elindulva addig követi a mutatókat, míg el nem jut egy olyan kifejezésig vagy változóig, ahonnan már nem vezet tovább mutató.
Ha például az alábbihoz hasonló értékadási lánccal találkozik, végig megy rajta, és ennek megfelelő értéket rendel a változókhoz - most mindegyikhez az 1 értéket.

```
restart;
a := b; b := c; c:=d; d := 1;
a; b; c; d;
```

$$a := b$$
$$b := c$$
$$c := d$$
$$d := 1$$
$$1$$
$$1$$
$$1$$
$$1$$

Tudnivalók XIX.

Az `eval` eljárás alkalmazásával mi is vezérelni tudjuk a kiértékelések végrehajtását. Második paraméterének pozitív egész számot megadva a Maple csak az adott számú lépésben (a megadott mélységig) végzi el a kiértékelést.

```
restart;
a := b: b := c: c:=d: d := 1:
eval(a,1); eval(a,2);eval(a,3);eval(a,4);
```

$$b$$
$$c$$
$$d$$
$$1$$

```
eval(a+b+c,3);
```

$$d+2$$

Az `evaln` eljárás mindig névre értékel ki, az argumentumában megadott változó nevét adja vissza.

```
evaln(a);
```

$$a$$

Tudnivalók XX.

Vannak olyan nagy, összetett Maple objektumok, amelyeknél a teljes kiértékelés nem lenne célszerű. Ilyenkor a rendszer csak az utolsó névig végzi el a kiértékelést (`last name evaluation`). Később látni fogjuk, hogy ez történik tömbök és táblák, vagy eljárások (függvények) esetében.
Egy egyszerű példa: az `f` névhez értékként az `x -> x^2+1` függvényt rendeljük. Az utolsó névre való kiértékelés miatt az `f` kifejezés csak önmagára értékelődik ki. Az `eval` hívásával kényszerítjük ki az `f`-hez rendelt tényleges értékre (a hozzá rendelt függvényre) való kiértékelést.

```
f := x -> x^2 +1;
f;  evaln(f);
eval(f);
```

$$f := x \to x^2 + 1$$
$$f$$
$$f$$
$$x \to x^2 + 1$$

Tudnivalók XXI.

Néha pont azt szeretnénk, hogy valamit ne értékeljen ki a program. Az egyszeres álló idézőjelek (<Shift>+<1>) közé zárt kifejezéseket nem értékeli ki a Maple, csak leveszi az idézőjeleket, és az így kapott kifejezést tekinti a kiértékelés eredményének.
Ha többszörös, skatulyázott idézőjeleket használunk, több lépéssel késleltejük a kifejezés kiértékelését.
Figyeljük meg, az alábbi példában hogyan fogynak el az idézőjelek.

```
restart;
a := 1;
eval('''a'+ 1'',1);
eval('''a'+ 1'',2);
eval('''a'+ 1'',3);
eval('''a'+ 1'',4);
```

$$a := 1$$
$$''a' + 1'$$
$$'a' + 1$$
$$a + 1$$
$$2$$

Hasonló eredménye lesz a % operátor ismételt alkalmazásának is, mivel ekkor minden lépésben eggyel több kiértékelést hajt végre a rendszer:

```
'''a'+ 1''; %; %; %;
```

$$''a' + 1'$$
$$'a' + 1$$
$$a + 1$$
$$2$$

Tudnivalók XXII.

Az idézőjeleket gyakran használjuk változók értékének törlésére (a változók „tisztítására”):

```
a := 1;
a := 'a';
a ;
```

$$1$$
$$a$$
$$a$$

Tudnivalók XXIII.

A Maple értékadó utasításának több változata van. Legegyszerűbb a már látott **név** := **kifejezés** alak.
Egy fokkal általánosabb a **névsorozat** := **kifejezéssorozat** alakú többszörös értékadás, ahol a két sorozat hossza azonos, és a bal oldali nevek értékét a jobb oldali kifejezéssorozat megfelelő sorszámú tagja adja.
Az olvashatóság miatt mindkét sorozatot `(...)` zárójelbe tehetjük, de ez nem kötelező.

```
restart;
i := 3; j := 4; k := 5;
i; j; k;
```

$$i := 3$$
$$j := 4$$
$$k := 5$$
$$3$$
$$4$$
$$5$$

```
restart;
(i, j, k) := (3, 4, 5);
i; j; k;
```

$$i,j,k := 3,4,5$$
$$3$$
$$4$$
$$5$$

Tudnivalók XXIV.

Az értékadás másik módja az `assign` eljárás használata. Hívása: `assign(`**név, kifejezés**`)`, vagy `assign(`**név**`=`**kifejezés**`)`, vagy `assign(`**névsorozat**`=`**kifejezéssorozat**`)`. Az első kettő a **név** `:=` **kifejezés** értékadáshoz, a harmadik a **névsorozat** `:=` **kifejezéssorozat** többszörös értékadáshoz hasonló – egy lényeges különbséggel.
Míg a közönséges értékadásnál a Maple csak a jobb oldalt értékeli ki, itt mindkét oldal kiértékelődik.
Hibának számít, ha a kiértékeléssel az első argumentum értékeként nem nevet kap a Maple. Ez elkerülhető, ha az első argumentumban szereplő minden nevet idézőjelek közé teszünk. Az outputból az is látszik, hogy az `assign` nem ír ki semmit.

```
restart;
a := 1; b := 2;
assign( a,b = 3,4 );
```

$$1$$
$$2$$

```
Error, invalid left hand side in assignment
```

```
restart;
a := 1; b := 2;
assign( ('a','b') = (3,4) );
a; b;
```

$$a := 1$$
$$b := 2$$
$$3$$
$$4$$

Tudnivalók XXV.

Vigyázat! a zárójelek elhagyása itt mást eredményt ad:

```
restart;
a := 1; b := 2;
assign( 'a','b' = 3,4 );
a; b;
```

$$a := 1$$
$$b := 2$$
$$2 = 3, 4$$
$$2$$

Még egy tanulságos példa:

```
restart;
a := b;
a := 2;
a,b;
```

$$a := b$$
$$a := 2$$
$$2, b$$

```
restart;
assign(a=b);
assign(a=2);
a,b;
```

$$2, 2$$

Tudnivalók XXVI.

Az `assigned` eljárással megkérdezhetjük, hogy egy névhez van-e érték rendelve. Ez kiderül az `about` eljárás outputjából is.

```
restart;
a := 1;
assigned(a);
about(a);
```

$$1$$

$$\textit{true}$$

```
1:
  All numeric values are properties as well as objects.
  Their location in the property lattice is obvious,
  in this case integer.
```

Tudnivalók XXVII.

Az `unassign` eljárás törli az argumentumában megadott változók értékét.
Az idézőjeles értékadáson és az `evaln` hívásán túl ez már a harmadik módszer változók tisztítására.

```
restart;
a:=1; b:=1; c := 1;
```

$$a := 1$$
$$b := 1$$
$$c := 1$$

```
a:='a': unassign('b'): c := evaln(c):
a; b; c;
```

$$a$$
$$b$$
$$c$$

Tudnivalók XXVIII.

Azon túl, hogy a megfelelő menüpontok használatával fájlokat nyithatunk meg, illetve eredményeinket fájlokba menthetjük, a Maple több szintű hozzáférést is biztosít a host gép operációs rendszeréhez. Az ehhez szükséges eljárásokból csak kettőt említünk meg.
A `system` és az `ssystem` eljárás segítségével parancsokat adhatunk át az operációs rendszernek, lásd `?system`, `?ssystem`.
Pontosabban az eljárások első paramétereként megadott sztringet az operációs rendszer parancsértelmezője (shell) kapja meg és hajtja végre.
Míg a `system` esetében a shell output egy külön felugró ablakban látható, az `ssystem` ezt egy Maple sztringben adja vissza, s így további számításainkban is felhasználhatjuk.

Tudnivalók XXIX.

Ezek a lehetőségek csak az operációs rendszer és a Maple megfelelő biztonsági beállításai mellett működnek.

```
system("date");
ssystem("ls /home/viragh/Maple");
```

$$0$$

$$[0,"abra1.eps\ abra2.eps\ abra3.jpg\ abrak.mpl\ demo1.mw\ demo2.mw\ mentes.mpl"]$$

A `system[launch](`**programnév, paraméterek**`)` alakú hívással külső programokat – itt a firefoxot – indíthatunk el. A visszaadott érték az elindított linux processz azonosítószáma (PID).

```
system[launch]("firefox", "www.maplesoft.com");
```

$$15974$$

11. előadás

Ebben az előadásban az alábbi három területet tekintjük át:

- a Maple legfontosabb adattípusai;

Ebben az előadásban az alábbi három területet tekintjük át:

- a Maple legfontosabb adattípusai;
- számolás algebrai kifejezésekkel;

Ebben az előadásban az alábbi három területet tekintjük át:

- a Maple legfontosabb adattípusai;
- számolás algebrai kifejezésekkel;
- egyenletek megoldása

2. Adattípusok

Adattípusok

A matematikai kifejezésekkel való szimbolikus számításon túl a Maple támogatja a más programnyelvekben is használt alapvető adattípusok használatát.
Ebben a fejezetben a következő adattípusokkal foglalkozunk (hasonló témákra visszatérünk még a 14. Előadás megfelelő részében):

- sztringek (**string**);

Adattípusok

A matematikai kifejezésekkel való szimbolikus számításon túl a Maple támogatja a más programnyelvekben is használt alapvető adattípusok használatát.
Ebben a fejezetben a következő adattípusokkal foglalkozunk (hasonló témákra visszatérünk még a 14. Előadás megfelelő részében):

- sztringek (**string**);
- kifejezésssorozatok (**exprseq**);

Adattípusok

A matematikai kifejezésekkel való szimbolikus számításon túl a Maple támogatja a más programnyelvekben is használt alapvető adattípusok használatát.
Ebben a fejezetben a következő adattípusokkal foglalkozunk (hasonló témákra visszatérünk még a 14. Előadás megfelelő részében):

- sztringek (**string**);
- kifejezésssorozatok (**exprseq**);
- listák (**list**) és

Adattípusok

A matematikai kifejezésekkel való szimbolikus számításon túl a Maple támogatja a más programnyelvekben is használt alapvető adattípusok használatát.
Ebben a fejezetben a következő adattípusokkal foglalkozunk (hasonló témákra visszatérünk még a 14. Előadás megfelelő részében):

- sztringek (**string**);
- kifejezésssorozatok (**exprseq**);
- listák (**list**) és
- halmazok (**set**).

Típuskezelés I.

A Maple számos különböző adattípusba tartozó objektummal dolgozik (lásd `type`). Sőt a típusok rendszere bővíthető - normál felhasználói szinten is - újabb típusok definiálásával.
Az alábbi példákban olyan matematikai kifejezésekkel kapcsolatos típusok szerepelnek, amelyekkel már korábban találkoztunk.
Több lehetőségünk is van egy Maple kifejezés típusának megállapítására. A `whattype` eljárás a vizsgált objektum típusát adja vissza. A `whattype` csak az alapvető adattípusokat ismeri, de ezek száma is közel 30.

```
whattype(a);
whattype(3);
whattype(3.0);
whattype(1/3);
whattype(2+3*I);
whattype(cos(x));
whattype(x=2);
```

$$
\begin{gathered}
symbol \\
integer \\
float \\
fraction \\
complex(extended_numeric) \\
function \\
`=`
\end{gathered}
$$

Típuskezelés II.

Az alábbi példákból látható, hogy az adattípust a kifejezés „legkülső” művelete határozza meg, tehát az eredmény a kifejezés szintaktikus elemzésével kapott szintaxisfa gyökerénél található típus lesz.

```
x*sin(x)- cos(x))^2 ;
whattype( (x*sin(x)- cos(x))^2 );
 x^2+cos(x)-2;
whattype( x^2+cos(x)-2 );
 y*(x*sin(x)- cos(x))^2
whattype( y*(x*sin(x)- cos(x))^2 );
```

$$(x\sin(x) - \cos(x))^2$$
$$\text{``^``}$$
$$x^2 + \cos(x) - 2$$
$$\text{`+`}$$
$$y(x\sin(x) - \cos(x))^2$$
$$\text{`*`}$$

Típuskezelés III.

A `type` eljárással azt kérdezhetjük meg - és a visszaadott eredmény egy logikai érték -, hogy egy kifejezés adott típusú-e. Ez a parancs száznál több típust ismer fel.
A második paraméterként megadandó típusneveket a Help megfelelő oldalain találhatjuk meg.

```
type(a + b, integer);
type(a + b, `+`);
type(2+3*I, `+`);
type(2+3*I, complex);
type(2+3*I, float);
type(sin(cos(x)),trig);
```

false
true
false
true
false
true

Típuskezelés IV.

Az alábbi példákból látható, hogy a Maple gazdag eszközrendszere megengedi általánosabb típus kifejezésekkel (`type expression`), például a típusokból álló halmazokkal: `{polynom, sin}` vagy az összetett típusokkal való típusmegadást is.

```
> type(a + b, polynom(name,name));
> type(sin(cos(x)), {polynom, sin});
> type(sin(x^3-2*x), function(polynom));
> type(sin(cos(x-1)),function(function(name)));
```

$$true$$
$$false$$
$$true$$
$$false$$

Sztringek I.

A sztring (karakterlánc) adattípus neve a Maple-ben `string`. A sztringeket a `"` kettős idézőjelek közé írt tetszőleges karaktersorozattal adjuk meg. Az *üres sztring* a `""` jelsorozat.

```
"ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP CSAPPANTYÚK!?";
"";
```

"ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP CSAPPANTYÚK!?"
""

A Maple-ben a `symbol` típusú (változó)nevek, és a `string` típusú sztringek egészen mást jelentenek. A sztringek valójában sztring konstansok, tehát nem állhatnak pl. értékadás bal oldalán (de a jobb oldalon igen):

```
kifli; whattype(kifli);
```

kifli
symbol

```
"kifli"; whattype("kifli");
```

"kifli"
string

```
kifli := "sós";
```

kifli := "sós"

```
"kifli" := "sós";
```

```
Error, invalid left hand side of assignment
```

A `convert` eljárással sztringet névvé, nevet sztringgé konvertálhatunk.

```
convert("kifli", symbol);
convert(sós, string);
```

kifli
"sós"

Sztringek II.

Sztringek és nevek összefűzésére (konkatenációjára) a `cat` eljárás, illetve a `||` művelet szolgál, lásd `?cat`. A Help szerint inkább a `cat` használata ajánlott.
A konkatenáció eredményének típusa (`symbol` vagy `string`) a bal oldali operandus típusától függ: ha ez név, akkor az eredmény is az lesz; ha nem, akkor mindenképpen sztringet kapunk.

```
restart;
cat(a, b), whattype(cat(a, b));
cat(a, "barack"), whattype(cat(a, "barack"));
cat("alma", b), whattype(cat("alma", b));
cat("alma", "barack"), whattype(cat("alma", "barack"));
a||b, a||"barack", "alma"||b, "alma"||"barack";
```

$$ab, \mathit{symbol}$$
$$\mathit{abarack}, \mathit{symbol}$$
$$\text{"almab"}, \mathit{string}$$
$$\text{"almabarack"}, \mathit{string}$$
$$ab, \mathit{abarack}, \text{"almab"}, \text{"almabarack"}$$

Sztringek III.

A `cat` eljárás és a `||` művelet más típusokkal (például egész konstansokkal, vagy egész értékű tartományokkal) is használható.
Figyeljük meg az alábbi példákat. Tartomány (`range`) hozzáfűzésével könnyen generálható változónevek sorozata.

```
cat(a,1); cat(a,1..3);
cat(1,a); cat(1..3,a);
```

$$a1$$
$$a1, a2, a3$$
$$1a$$
$$1a, 2a, 3a$$

A változónevek generálásával párhuzamosan rögtön értéket is adhatunk nekik:

```
cat('a', 1..3) := (666\$3);
```

$$a1, a2, a3 := 666, 666, 666$$

A `||` művelet kicsit másképpen viselkedik.

```
restart;
a||1; a||1..3;
```

$$a1$$
$$a1..3$$

```
1||a;
Error, '||' unexpected
```

Kiértékeletlen konkatenációt kapunk, ha a második operandus nem `integer`, `symbol`, `range` vagy `string` típusú.

```
cat(a, p+q), a||(p+q);
```

$$a||(p+q), a||(p+q)$$

Sztringek IV.

Tetszőleges sztring vagy név hosszát a `length` eljárással számíthatjuk ki, összefüggő „szeleteiket” a `substring` eljárás adja. (Mint általában, a pozitív számok elölről, a negatívak pedig hátulról számolják a pozíciókat.
A több bájtos UTF8 karakterkódolás miatt ékezetes vagy egyéb speciális karaktereket tartalmazó sztringek esetében a `length` és a `substring` eredménye első látásra furcsa lehet:

```
S1:="Sűrű, sötét az éj";
length(S1);
Nev1:= vakablak77;
length(Nev1);
```

$$S1 := \text{"Sűrű, sötét az éj"}$$
$$22$$
$$Nev1 := vakablak77$$
$$10$$

```
substring(S1,2..6), whattype(substring(S1,2..6));
substring(Nev1,-5..-2), whattype(substring(Nev1,-5..-2));
```

$$\text{"űrű"}, string$$
$$lak7, symbol$$

Sztringek V.

További sztringeljárások a `StringTools` csomagban találhatók. Példa keresésre és rendezésre a csomag eljárásaival:

```
with(StringTools):
S1;
poz := Search("ö", S1); # UTF8 tárolás!
```

"Sűrű, sötét az éj"

$poz := 10$

```
Nev1;
Sort(Nev1);
```

vakablak77

"77aaabkklv"

Sorozatok I.

A Maple legegyszerűbb összetett adatstruktúrája a kifejezés sorozat (expression sequence), röviden sorozat. Tagjai nem csak számok, hanem tetszőleges kifejezések is lehetnek. Azonos elemek többször is előfordulhatnak, a sorozat tagjai lehetnek különböző típusúak is.
Sorozatot megadni vesszővel elválasztott elemeinek felsorolásával lehet. Az üres sorozatot a NULL konstans jelöli.
Bár a Maple-ben nincs „expression sequence” *típus*, a `whattype` eljárás az `exprseq` értéket adja vissza azokra a Maple objektumokra, melyekhez rendelt érték sorozat.

```
restart;
piros, feher, "zöld", 2, sin(x)+1;
s := %;
whattype(s);
uressorozat:= NULL;
```

$$piros, feher, \text{"zöld"}, 2, \sin(x)+1$$
$$s := piros, feher, \text{"zöld"}, 2, \sin(x)+1$$
$$exprseq$$
$$uressorozat :=$$

A sorozatokat egymás után írással össze lehet fűzni:

```
s2 := "második", Pi, x^3;
s3 := s,s2;
```

$$s2 := \text{"második"}, \pi, x^3$$
$$s3 := piros, feher, \text{"zöld"}, 2, \sin(x)+1, \text{"második"}, \pi, x^3$$

Sorozatok II.

A sorozatokat elemeik felsorolása helyett sokszor könnyebb valamilyen matematikai szabállyal definiálni. A Mapleben ezt az `seq` eljárással tehetjük meg. Az `seq` az első paraméterében szereplő képletet alkalmazza a második paraméterében megadott esetekre, és az így kapott értékekből álló sorozatot adja vissza. Két változata:

`seq(` **kifejezés, név = kezdőérték..végérték, növekmény** `)`
`seq(` **kifejezés, név = objektum** `)`

Az első alakban a név változó értéke a kezdőértéktől a végértékig fut növekmény lépésekben (A három paraméter csak numerikus érték lehet), miközben kiszámolja a kifejezés megfelelő értékeit. A második alakban a név változó sorra az objektum egyes elemeit (operandusait) veszi fel értékül.

```
seq( i^2, i=0..5);
```

$$0, 1, 4, 9, 16, 25$$

```
n := -1; m := 9/2;
seq( i^2, i= n..m, 2 );
```

$$n := -1$$
$$m := 9/2$$
$$1, 1, 9$$

Az objektum lehet sorozat,(zárójelbe kell tenni!), sztring vagy kifejezés is.

```
seq( i, i=(2,3,5,7,11) );
seq( i, i= "vakablak" );
seq( c, c = 3*x^2 + x -1 );
```

$$2, 3, 5, 7, 11$$
$$\text{"v"}, \text{"a"}, \text{"k"}, \text{"a"}, \text{"b"}, \text{"l"}, \text{"a"}, \text{"k"}$$
$$3x^2, x, -1$$

Sorozatok III.

Explicit módon megadott véges sorozat esetén a sorozat tagjainak összeadására az `add`, összeszorzására pedig a `mul` eljárást lehet használni. Lásd `add`, `mul`.
A paraméterek megadása hasonlít az `seq` eljáráséhoz.

```
n := 1; m:= 4;
add(i^3,i=n..m);
mul(i^3,i={-1,1,3});
add(a^k*b^(3-k), k=0..3);
```

$$n := 1$$
$$m := 4$$
$$100$$
$$a^3 + a^2 b + a b^2 + b^3$$

```
add(sin((2*k)*Pi/5), k= -3..3);
```

$$0$$

Itt „csalunk”, mert a `sin` függvény kiértékelését megakadályozzuk:

```
add('sin'((2*k)*Pi/5), k= -3..3);
```

$$\sin\left(-\frac{6\pi}{5}\right) + \sin\left(-\frac{4\pi}{5}\right) + \sin\left(-\frac{2\pi}{5}\right) + \sin(0) + \sin\left(\frac{2\pi}{5}\right) + \sin\left(\frac{4\pi}{5}\right) + \sin\left(\frac{6\pi}{5}\right)$$

Sorozatok IV.

Szimbolikus összeget, illetve szorzatot a `sum`, illetve a `product` eljárásokkal kaphatunk meg. A megfelelő inert `Sum` és `Product` eljárások csak kijelölik az elvégzendő műveleteket.

```
Sum(i^4, i = 1..k) = sum(i^4,i = 1..k);
```

$$\sum_{i=1}^{k} i^4 = 1/5\,(k+1)^5 - 1/2\,(k+1)^4 + 1/3\,(k+1)^3 - 1/30\,k - 1/30$$

```
Product(exp(i),i=1..k) = product(exp(i),i=1..k);
```

$$\prod_{i=1}^{k} \mathrm{e}^{i} = \mathrm{e}^{1/2\,(k+1)^2 - 1/2\,k - 1/2}$$

Ha a Maple nem tudja meghatározni a kért összeget vagy szorzatot, az inert alakhoz hasonló kiértékeletlen eredményt ad vissza

```
sum(k!/sin(k*x), k = 1..t);
```

$$\sum_{k=1}^{t} \frac{k!}{\sin(kx)}$$

Sorozatok V.

Célszerű az `add` és a `sum` paramétereit kiértékeletlenül (idézőjelek között) megadni, így az sem zavar, ha korábban már valamilyen értéket kaptak. Figyeljük meg a három sum eljárás különböző eredményét.

```
j:= 0;
sum(j^2, j = 1..t);
```

$$j := 0$$

```
Error, (in sum) summation variable previously assigned,
second argument evaluates to 0 = 1 .. t
```

```
sum(j^2, 'j' = 1..t);
sum('j^2', 'j' = 1..t);
```

$$0$$

$$1/3\,(n+1)^3 - 1/2\,(n+1)^2 + 1/6\,n + 1/6$$

Sorozatok VI.

A `sum` és a `product` eljárások képesek végtelen összegek és szorzatok kezelésére is, összhangban ezek matematikai definíciójával (a részletszorzatok, illetve részletösszegek határértéke).

```
restart;
Sum(1/k, k= 1..infinity)= sum(1/k, k= 1..infinity);
Product(1/k, k = 1..infinity)= product(1/k, k = 1..infinity);
```

$$\sum_{k=1}^{\infty} k^{-1} = \infty$$

$$\prod_{k=1}^{\infty} k^{-1} = 0$$

Sorozatok VII.

A Maple is ismeri a négyzetszámok reciprokaiból álló sor összegét. Sőt még ennél jóval többet is tud.

```
a := n -> 1/n^k;
seq(Sum(a(n), n=1..infinity) = sum(a(n), n=1..infinity), k=1..6);
```

$$n \to \left(n^k\right)^{-1}$$

$$\sum_{n=1}^{\infty} n^{-1} = \infty, \sum_{n=1}^{\infty} n^{-2} = 1/6\,\pi^2, \sum_{n=1}^{\infty} n^{-3} = \zeta(3), \sum_{n=1}^{\infty} n^{-4} = \frac{1}{90}\,\pi^4, \sum_{n=1}^{\infty} n^{-5} = \zeta(5), \sum_{n=1}^{\infty} n^{-6} = \frac{1}{945}\,\pi^6$$

```
seq( Sum(a(n), n=1..infinity) = evalf(sum(a(n), n=1..infinity)), k=1..5);
```

$$\sum_{n=1}^{\infty} n^{-1} = Float(\infty), \sum_{n=1}^{\infty} n^{-2} = 1.644934068, \sum_{n=1}^{\infty} n^{-3} = 1.202056903, \sum_{n=1}^{\infty} n^{-4} = 1.082323234, \sum_{n=1}^{\infty} n^{-5} = 1.03692775$$

Listák I.

A listákat szögletes zárójelek között, egymástól vesszővel elválasztott elemeinek felsorolásával adhatjuk meg. Az elemek - a sorozatokhoz hasonlóan - lehetnek eltérő típusúak is. Úgy is mondhatjuk, hogy sorozatot szögletes zárójelekbe téve listát kapunk. Üres listát a `[]` jelöléssel hozhatunk létre. A lista típus neve `list`.

```
restart;
[egy,ketto,harom,negy,ot];
s := a, 22, "valami";
L := [s];
whattype(L);
ureslista:=[];
whattype(ureslista);
```

$$[egy, ketto, harom, negy, ot]$$
$$s := a, 22, \text{"valami"}$$
$$L := [a, 22, \text{"valami"}]$$
$$list$$
$$ureslista := []$$
$$symbol$$

A lista elemeit a `[]` indexkiválasztó operátorral kapjuk. Az összes elemet adja az `[]` üres indexkiválasztó kifejezés:

```
L[1];
L[-2]
L[];
```

$$a$$
$$22$$
$$a, 22, \text{"valami"}$$

Listák II.

Lista elemeinek rendezésére a `sort` eljárás használható. Megadott feltételt teljesítő elemeit kiválasztására vagy törlésére a `select`, illetve a `remove` eljárás szolgál. A három eljárás szintaxisát és működését az alábbi példák illusztrálják. A `select` és a `remove` első paraméterként olyan eljárást kell megadni, amely bármelyik listaelemre alkalmazva a `true` vagy a `false` logikai értéket adja vissza. Lásd `?select`, `?remove`.

```
sort([13,-5, 4]);
sort([i,a,b,f,e]);
```

$$[-5, 4, 13]$$
$$[a, b, e, f, i]$$

```
L:=[0,2,25,100,13];
select(isprime,L);
select(isprime,L);
select( x -> x < 10, L);
remove(isprime,L);
remove(x -> x mod 25 = 0, L);
```

$$L := [0, 2, 25, 100, 13]$$
$$[2, 13]$$
$$[0, 2]$$
$$[0, 25, 100]$$
$$[2, 13]$$

Listák III.

További fontos listaműveleteket a `ListTools` csomag tartalmaz. Ebből mutatunk néhány példát listában való keresésre (`Search`, `SearchAll`) és listák megfordítására (`Reverse`).

```
with(ListTools);
```

$$[BinaryPlace, BinarySearch, Categorize, Classify, DotProduct, Enumerate, FindMaximalElement, FindMinimalElement, FindRepetitions, Flatten, FlattenOnce, Group, Interleave, Join, JoinSequence, LengthSplit, MakeUnique, Occurrences, Pad, PartialSums, Reverse, Rotate, Search, SearchAll, SelectFirst, SelectLast, Sorted, Split, Transpose]$$

```
L := [0, 3, 2, 25, 100, 3, 13];
ind := Search(25, L);
ind := SearchAll(3, L);
Reverse(L);
```

$$[0, 3, 2, 25, 100, 3, 13]$$
$$ind := 4$$
$$ind := 2, 6$$
$$[13, 3, 100, 25, 2, 3, 0]$$

Halmazok I.

A halmaz típus neve `set`. A halmazokat `{}` zárójelek között, egymástól vesszővel elválasztott elemeikkel adjuk meg. Természetesen a halmazok elemei is lehetnek eltérő típusúak. Úgy is mondhatjuk, hogy sorozatot kapcsos zárójelekbe téve halmazt kapunk - persze a sorozat esetleges ismétlődő elemei a halmazban csak egyszer fordulnak elő.

```
A:={Pi,1,valami,x+y,Pi};
B:={"valami",Pi,2};
```

$$1, \pi, valami, x+y$$
$$2, \text{"valami"}, \pi$$

Látható, hogy a megadott halmazok elemeit nem az általunk írt sorrendben írja ki a Maple - persze a halmazoknál a sorrend nem is számít! Az üres halmaz jelölése `{}`.

```
ureshalmaz:= {};whattype(A);
```

$$ureshalmaz := set$$

A szokásos halmazműveletek: egyesítés (`union`), metszet (`intersect`) és különbség (`minus`).

```
'A' = A; 'B' = B;
'A union B' = A union B;
'A intersect B' = A intersect B;
'A minus B' = A minus B;
```

$$A = 1, \pi, valami, x+y$$
$$B = 2, \text{"valami"}, \pi$$
$$A \cup B = 1, 2, \text{"valami"}, \pi, valami, x+y$$
$$A \cap B = \pi$$
$$A \backslash B = 1, valami, x+y$$

Átalakítások I.

Tekintsük át, hogy milyen kapcsolat van az előzőekben vizgált három adattípus között.

```
sorozat:= egy,ketto,harom,negy;
halmaz:= {egy,ketto,harom,negy};
lista := [egy,ketto,harom,negy];
```

$$egy, ketto, harom, negy$$
$$\{egy, harom, ketto, negy\}$$
$$[egy, ketto, harom, negy]$$

Mindhárom adattípus esetén az elemekre való hivatkozás az elemek indexének szögletes zárójelek közt való megadásával történik. Az elemek indexelése 1-gyel kezdődik. Negatív index az utolsó elemtől való visszaszámolást jelenti.

```
sorozat[2];
halmaz[1];
lista[-1];
```

$$ketto$$
$$egy$$
$$negy$$

Kijelölhetünk ilyen módon résztartományt is (az output ugyanolyan típusú lesz, mint az input volt)

```
sorozat[2..4];
halmaz[2..-2];
lista[-3..-1];
```

$$ketto, harom, negy$$
$$\{harom, ketto\}$$
$$[ketto, harom, negy]$$

Átalakítások II.

Bármelyik típust a másikba tudjuk alakítani.

```
sorozat;
[sorozat];
{sorozat};
```

$$egy, ketto, harom, negy$$
$$[egy, ketto, harom, negy]$$
$$\{egy, harom, ketto, negy\}$$

```
halmaz;
halmaz[];
[halmaz[]];
```

$$\{egy, harom, ketto, negy\}$$
$$egy, harom, ketto, negy$$
$$[egy, harom, ketto, negy]$$

```
lista;
lista[];
{lista[]};
```

$$[egy, ketto, harom, negy]$$
$$egy, ketto, harom, negy$$
$$\{egy, harom, ketto, negy\}$$

Átalakítások III.

A `convert` eljárással is végeztethetünk hasonló konverziókat. Ennek első paramétere a konvertálandó objektum, a második meg a célobjektum (a konvertálás eredménye) típusa.

```
convert(lista,set);
convert(halmaz,list);
convert([sorozat],set);
convert([sorozat],list);
```

$$\{egy, harom, ketto, negy\}$$
$$[egy, harom, ketto, negy]$$
$$\{egy, harom, ketto, negy\}$$
$$[egy, ketto, harom, negy]$$

Az utolsó két esetben azért kellett a `[sorozat]` alak, mert különben a sorozat minden eleme a `convert` függvény egy-egy argumentumaként szerepelt volna.

Átalakítások IV.

A sorozatok egyszerűen bővíthetők:

```
sorozat;
"kek",sorozat,123;
```

$$egy, ketto, harom, negy$$
$$\text{"kek"}, egy, ketto, harom, negy, 123$$

A listák bővítése nehézkesebb (először a listát sorozattá konvertáljuk, így bővítjük, majd visszaalakítjuk listává)

```
lista := [1,2,3];
ujlista:=[elolbovit, lista[], hatulbovit];
```

$$[1, 2, 3]$$
$$ujlista := [elolbovit, 1, 2, 3, hatulbovit]$$

Hasonlóan járhatunk el halmazok esetében is. (Még egyszer kiemeljük, hogy halmazban ugyanaz az elem csak egyszer szerepelhet!)

```
H:={4,1,2,4,4};
ujH := {H[], ujelem};
```

$$H := \{1, 2, 4\}$$
$$ujH := \{1, 2, 4, ujelem\}$$

Átalakítások V.

A `member` eljárással megkérdezhetjük, hogy egy adott elem előfordul-e egy halmazban, listában vagy sorozatban, és a harmadik paraméterként megadott változóban azt is visszakapjuk, hogy hányadik helyen szerepel:

```
halmaz := {egy, ketto, harom, negy};
member(egy, halmaz, 'poz1');
poz1;
```

$$halmaz := \{egy, harom, ketto, negy\}$$
$$true$$
$$1$$

```
member(5, halmaz, 'poz2');
```

$$false$$
$$poz2$$

```
member(Pi,[1,2,Pi,x^2], 'poz3');
poz3;
```

$$true$$
$$3$$

Sorozatoknál a szokásos trükköt alkalmazzuk

```
s := 11,12,f(x);
member(12,{s}, 'poz4');
poz4;
```

$$s := 11, 12, f(x)$$
$$true$$
$$2$$

Átalakítások VI.

Indexelhető adatttípusok esetében (ilyenek például a sztring, a lista és a halmaz, lásd `?type,indexable`) az elemek (komponensek) számát megkaphatjuk a `numelems` eljárással.
Sztringeknél az UTF8 kódolás miatt néha első látásra meglepő az eredmény:

```
numelems("abcxw"), numelems("rózsa");
numelems([a,2,Pi,sqrt(2)]);
numelems({a,2,Pi});
```

$$5, 6$$
$$4$$
$$3$$

Az op függvénycsalád I.

Kifejezések részkifejezéseinek (operandusainak) kezelése az `op` függvénycsalád segítségével lehetséges.Lásd `?op`.

```
kif := 3*x^2 - Pi*x + sin(x);
```

$$kif := 3x^2 - \pi x + \sin(x)$$

Az operandusok számát a `nops` eljárás adja meg:

```
nops(kif);
```

$$3$$

Az összes operandus megkapható – kifejezéssorozatként – az `op` alkalmazásával:

```
op(kif);
```

$$3x^2, -\pi x, \sin(x)$$

Az op függvénycsalád II.

Az `op` első paramétereként egy természetes számot megadva a kifejezés adott sorszámú operandusát (például a nulladikat, ami a kifejezés típusa) kapjuk eredményül

```
op(0,kif); op(1,kif); op(2,kif);op(3,kif);
```

$$+$$
$$3x^2$$
$$-\pi x$$
$$sin(x)$$

Egy **tartomány** (*range*) típusú kifejezéssel az operandusok egy bizonyos részét kérhetjük, az eredmény kifejezéssorozat:

```
op(0..2, kif);
```

$$+, 3x^2, -\pi x$$

```
op(0..nops(kif), kif);
```

$$+, 3x^2, -\pi x, \sin(x)$$

Az op függvénycsalád III.

A `subsop` segítségével az utolsó paramétereként megadott kifejezés előtte fölsorolt sorszámú operandusait helyettesíthetjük a megfelelő helyettesítő kifejezéssel (a `NULL` helyettesítése az adott operandus törlését jelenti). Az eredeti kifejezés változatlan marad.

```
kif;
subsop(1 = a^2, 3=b^2, kif);
kif;
```

$$3x^2 - \pi x + \sin(x)$$
$$-\pi x + a^2 + b^2$$
$$3x^2 - \pi x + \sin(x)$$

Az op függvénycsalád IV.

Mivel a Maple-ben „minden” kifejezés, az előző függvényeket alkalmazhatjuk mindenféle adatstruktúrára is:

```
L := [ 1, ketto, "három", Pi ];
op(L);
H := { elem1, elem2, elem3 };
op(0..nops(H), H);
subsop(1 = NULL, H);
```

$$L := [1, ketto, \text{"három"}, \pi]$$
$$1, ketto, \text{"három"}, \pi$$
$$H := \{elem1, elem2, elem3\}$$
$$set, elem1, elem2, elem3$$
$$\{elem2, elem3\}$$

A select és a remove eljárások

A listáknál korábban említett `select` és `remove` eljárások tetszőleges Maple kifejezések komponenseinek kiválogatására vagy eltávolítására használhatók. A harmadik paraméterben adjuk meg, hogy milyen részkifejezéseket kell figyelembe venni a kiválasztásnál vagy az eltávolításnál. Lásd `?select`, `?remove`.

```
select(has, {cos(x),x+1,sqrt(x+y)}, cos);
remove(has, {cos(x),x+1,sqrt(x+y)}, cos);
```

$$\{\cos(x)\}$$
$$\{\sqrt{x+y}, x+1\}$$

```
select(has, sin(x) + cos(y) + exp(x^2) +1, x);
remove(has, sin(x) + cos(y) + exp(x^2) +1, x);
```

$$\sin(x) + \exp(x^2)$$
$$1 + \cos(y)$$

Helyettesítések I.

A korábban használt `subsop` eljárás tetszőleges kifejezés adott sorszámú részkifejezését/kifejezéseit helyettesíti a megadott kifejezéssel/kifejezésekkel.

```
subsop(1 = y^2, 2=abs(cos(z)), c + abs(a+b)+ sin(\sqrt(a+b)));
```

$$y^2 + |\cos(z)| + \sin(\sqrt{a+b})$$

Hasonlóan működik a `subs` eljárás. Ez is szintaktikus helyettesítést végez: a megadott helyettesítendő kifejezés(ek) *minden* előfordulási helyén lecseréli őket a megfelelő helyettesítő kifejezésre. A helyettesítés *szekvenciálisan* történik: először az első, majd a második, . . . , végül az utolsó minta szerinti helyettesítéseket végzi el a Maple. Lásd `?subs`

```
subs(c = y^2, a + b = cos(z), c + abs(a+b)+ sin(\sqrt(a+b)));
```

$$y^2 + |\cos(z)| + \sin(\sqrt{\cos(z)})$$

Helyettesítések II.

Az `algsubs` („algebrai helyettesítés") figyelembe veszi a műveletek algebrai tulajdonságait (pl. kommutativitás, asszociativitás) is, ezért néha jobban használható. Lásd `?algsubs`

```
subs(a*b=x, b*c*a);
algsubs(a*b=x, b*c*a);
```

$$bca$$
$$cx$$

Sajnos az `algsubs` fontos korlátja, hogy a helyettesítendő változók a megadott mintában csak racionális kifejezésben szerepelhetnek. Itt a `subs` működik, az `algsubs` nem:

```
subs(sqrt(a + b) = cos(z), c + abs(a+b)+ sin(sqrt(a+b)));
algsubs(sqrt(a + b) = cos(z), c + abs(a+b)+ sin(sqrt(a+b)));
```

$$c + |a + b| + \sin(\cos(z))$$

```
Error, (in algsubs) no variables appearing rationally in pattern
```

Helyettesítések III.

A `simplify` is használható helyettesítésre alkalmas mellékfeltételek megadásával (ezeknek polinomiálisaknak kell lenni):

```
simplify(b*c*a, {a*b = x});
simplify(c + abs(a+b)+ sin(sqrt(a+b)), {a+b=cos(z), c=y^2});
simplify(c + sqrt(a+b) + ln(a+b), {sqrt(a+b)=cos(z)});
```

$$cx$$

$$y + |\cos(z)| + \sin(\sqrt{\cos(z)})$$

```
Error, (in simplify/siderels`:-simplify/siderels) side relations

must be polynomials in (name or function) variables
```

Helyettesítések IV.

Bizonyos esetekben a `subs` helyett az `eval` eljárás használata szükséges, mivel a formális (szintaktikus) helyettesítés nem ad elfogadható eredményt (néha az `algsubs` is segíthet).

```
subs(x = infinity,exp(-x^2));
algsubs(x = infinity,exp(-x^2));
eval(exp(-x^2), x = infinity);
```

$$e^{-\infty}$$
$$0$$
$$0$$

```
subs(x=0,sin(x)/x); algsubs(x=0,sin(x)/x); eval(sin(x)/x,x=0);
Error, numeric exception: division by zero
```

$$1$$

```
Error, numeric exception: division by zero
```

Helyettesítések V.

Ha a `subs` helyettesítési mintáit nem kifejezéssorozatként, hanem listában vagy halmazban adjuk meg, akkor a Maple *párhuzamosan* (szimultán) végzi el a helyettesítéseket. Figyeljük meg alaposan az alábbi hívások eredményét.

```
subs( y=x*y, x=a,    x*y^2);
subs( x=a, y=x*y,    x*y^2);
subs( [y=x*y, x=a], x*y^2);
subs( {x=a, y=x*y}, x*y^2);
```

$$a^3y^2$$
$$ax^2y^2$$
$$ax^2y^2$$
$$ax^2y^2$$

3. Algebrai műveletek

A Maple eredendően szimbolikus számításokra tervezett rendszer, ezért bonyolult matematikai kifejezések kezelésére, rajtuk különböző átalakítások elvégzésére is képes. Ezeket a lehetőségeket tekintjük át ebben a fejezetben.

Először a polinomokon és a racionális törtkifejezéseken elvégezhető legfontosabb átalakításokat vizsgáljuk. Ehhez az `expand`, `factor`, `combine`, `collect`,`sort`, `normal` és `convert` eljárásokat használjuk föl.
Vegyük figyelembe, hogy a Maple a beírt kifejezéseket és a számítások részeredményeit sokszor automatikusan egyszerűsíti, normálalakra hozza. Emiatt az output formátuma nem mindig felel meg várakozásainknak, vagy a matematikában leggyakoribb írásmódnak.

```
restart;
kif := (x-2)*(x-2)^3/(x+3*x);
```

$$\frac{1}{4}\,\frac{(x-2)^4}{x}$$

Műveletek polinomokkal és racionális törtkifejezésekkel Ia.

Az `expand` eljárás kifejti a kifejezésekben található beszorzásokat és a többtagúak hatványait, ezzel általában egy „rövidebb” kifejezésből „hosszabbat” hoz létre.

```
kif1:= (a-b)^5 - (a+b)^5;
kif2 := (x-y)*(x+2)^2/(1+(x-y)*(x+y));
```

$$kif1 := (a-b)^5 - (a+b)^5$$

$$kif2 := \frac{(x-y)(x+2)^2}{1+(x-y)(x+y)}$$

```
kif1e := expand(kif1);
kif2e := expand(kif2);
```

$$kif1e := -10a^4b - 20a^2b^3 - 2b^5$$

$$kif2e := \frac{x^3}{x^2-y^2+1} - \frac{x^2y}{x^2-y^2+1} + 4\frac{x^2}{x^2-y^2+1} - 4\frac{xy}{x^2-y^2+1} + 4\frac{x}{x^2-y^2+1} - 4\frac{y}{x^2-y^2+1}$$

Műveletek polinomokkal és racionális törtkifejezésekkel Ib.

Paraméterként meg lehet adni azokat a részkifejezéseket, amikre *nem* akarjuk az `expand`-et alkalmazni.

```
kif2ex := expand(kif2,x-y);
```

$$kif2ex := \frac{(x-y)\,x^2}{(x-y)\,x+(x-y)\,y+1} + 4\,\frac{(x-y)\,x}{(x-y)\,x+(x-y)\,y+1} + 4\,\frac{x-y}{(x-y)\,x+(x-y)\,y+1}$$

Műveletek polinomokkal és racionális törtkifejezésekkel IIa.

A `combine` és a `factor` eljárások lényegében „visszafelé” hajtják végre az `expand`-nél látott azonos átalakításokat, ezért általában „hosszabból” „rövidebb” kifejezéseket hoznak létre.
Általában nem igaz, hogy az `expand` eredményére alkalmazva rögtön visszadják a kifejezés eredeti (`expand` előtti) alakját.

```
kif3  := (x-2)^m*(x-2)^2;
kif3c := combine(kif3);
```

$$kif3 := (x-2)^m (x-2)^2$$
$$kif3c := (x-2)^{m+2}$$

Műveletek polinomokkal és racionális törtkifejezésekkel IIb.

A `factor` szorzattá alakítja a kifejezést.

```
'kif1' = kif1;
kif1f := factor(kif1e);
'kif2' = kif2;
kif2f := factor(kif2e);
```

$$kif1 = (a-b)^5 - (a+b)^5$$
$$kif1f := -2\,b\left(5\,a^4 + 10\,b^2a^2 + b^4\right)$$
$$kif2 = \frac{(x-y)\,(x+2)^2}{1+(x-y)\,(x+y)}$$
$$kif2f := \frac{(x-y)\,(x+2)^2}{x^2-y^2+1}$$

Műveletek polinomokkal és racionális törtkifejezésekkel III.

Racionális törtkifejezések esetén a `factor` külön-külön faktorizálja a számlálót és a nevezőt, majd elvégzi a lehetséges egyszerűsítéseket. Ez figyelhető meg az alábbi példán is.

```
kif4  := (x^2-1)/(x^2+3*x-4);
kif4f := factor(kif4);
kif4num := factor(numer(kif4));
kif4denom := factor(denom(kif4));
```

$$kif4 := \frac{x^2-1}{x^2+3x-4}$$
$$kif4f := \frac{x+1}{x+4}$$
$$kif4num := (x-1)(x+1)$$
$$kif4denom := (x+4)(x-1)$$

```
kif5  := 1/x + 2/(x^2+1) - (x+1)/(x-1)^2;
kif5f := factor(kif5);
```

$$kif5 := \frac{1}{x} + \frac{2}{x^2+1} - \frac{x+1}{(x-1)^2}$$
$$kif5f := -\frac{(x+1)\left(x^2+2x-1\right)}{x\left(x^2+1\right)(x-1)^2}$$

Műveletek polinomokkal és racionális törtkifejezésekkel IV.

A `collect` eljárással összegyűjthetjük valamelyik változó szerint az egynemű tagokat. Alkalmazzuk a `collect`-et a korábban definiált kif2 kifejezésre.
A Maple itt is külön-külön végzi a számlálóban és a nevezőben a megfelelő átalakításokat.

```
kif2;
kif2cx = collect(kif2, x);
kif2cy = collect(kif2, y);
```

$$\frac{(x-y)(x+2)^2}{1+(x-y)(x+y)}$$

$$kif2cx = \frac{(x-y)\left(x^2+4x+4\right)}{x^2-y^2+1}$$

$$kif2cy = \frac{(x-y)(x+2)^2}{x^2-y^2+1}$$

Ha az x hatványok szerint rendezett, „szebb” számlálót szeretnénk, például így segíthetünk:

```
kif2cxx := collect(expand(numer(kif2))/denom(kif2),x);
```

$$kif2cxx := \frac{x^3+(-y+4)x^2+(-4y+4)x-4y}{x^2-y^2+1}$$

Ha a kif2 `expand`-del kifejtett alakját vesszük, nem sokat segít a `collect`:

```
kif2ecx = collect(kif2e, x);
```

$$kif2ecx = \frac{x^3}{x^2-y^2+1} - \frac{x^2y}{x^2-y^2+1} + 4\frac{x^2}{x^2-y^2+1} - 4\frac{xy}{x^2-y^2+1} + 4\frac{x}{x^2-y^2+1} - 4\frac{y}{x^2-y^2+1}$$

Műveletek polinomokkal és racionális törtkifejezésekkel V.

A `sort` eljárás a megadott kifejezést az adott változó hatványai szerint rendezi. Vegyük a korábbi kif1 kifejezés kifejtett alakját.

```
'kif1e' = kif1e;
kif1sa  := sort(kif1e, a);
kif1sb  := sort(kif1e, b);
```

$$kif1e = -10\,a^4b - 20\,a^2b^3 - 2\,b^5$$
$$kif1sa := -10\,ba^4 - 20\,b^3a^2 - 2\,b^5$$
$$kif1sb := -2\,b^5 - 20\,a^2b^3 - 10\,a^4b$$

Műveletek polinomokkal és racionális törtkifejezésekkel VI.

A `normal` eljárás faktorizált normál formára (`factored normal form`) hozza a kifejezéseket. Ezekben a számláló és a nevező egész együtthatós relatív prím polinomok.
Az eddig vizsgált kifejezések normálformája:

```
kif1 = normal(kif1);
kif2 = normal(kif2);
kif3 = normal(kif3);
kif4 = normal(kif4);
kif5 = normal(kif5);
```

$$(a-b)^5-(a+b)^5=-2\,b^5-20\,a^2b^3-10\,a^4b$$

$$\frac{(x-y)\,(x+2)^2}{1+(x-y)\,(x+y)}=\frac{(x-y)\,(x+2)^2}{x^2-y^2+1}$$

$$(x-2)^m\,(x-2)^2=(x-2)^m\,(x-2)^2$$

$$\frac{x^2-1}{x^2+3\,x-4}=\frac{x+1}{x+4}$$

$$\frac{1}{x}+\frac{2}{x^2+1}-\frac{x+1}{(x-1)^2}=-\frac{x^3+3\,x^2+x-1}{x\,(x^2+1)\,(x-1)^2}$$

Műveletek polinomokkal és racionális törtkifejezésekkel VII.

Az `expanded` opcióval kifejtett normálformát kapunk, vagyis mind a számláló, mind a nevező kifejtett polinomokból áll.

```
'kif5' = kif5;
kif5ne := normal(kif5,expanded);
```

$$kif5 = \frac{1}{x} + \frac{2}{x^2+1} - \frac{x+1}{(x-1)^2}$$

$$kif5ne := \frac{-x^3 - 3x^2 - x + 1}{x^5 - 2x^4 + 2x^3 - 2x^2 + x}$$

Műveletek polinomokkal és racionális törtkifejezésekkel VIII.

A convert eljárással kifejezéseket különböző formába konvertálhatunk. A `parfrac` opció a kifejezés parciális törtekre bontott alakját adja. Néhány példa:

```
kif6 := (x^2+2*x-3)/(x^4+6*x^3+13*x^2+12*x+4);
'kif2' = kif2;
```

$$kif6 := \frac{x^2+2x-3}{x^4+6x^3+13x^2+12x+4}$$

$$kif2 = \frac{(x-y)(x+2)^2}{1+(x-y)(x+y)}$$

```
kif6p := convert(kif6,parfrac);
kif2px = convert(kif2,parfrac,x);
kif2py = convert(kif2,parfrac,y);
```

$$kif6p := \frac{8}{x+1} - \frac{3}{(x+2)^2} - \frac{8}{(x+2)} - 4\frac{1}{(x+1)^2}$$

$$kif2px = x - y + 4 + \frac{xy^2 - y^3 - 4xy + 4y^2 + 3x - 3y - 4}{x^2 - y^2 + 1}$$

$$kif2py = \frac{-x^3 + x^2y - 4x^2 + 4xy - 4x + 4y}{-x^2 + y^2 - 1}$$

Műveletek polinomokkal és racionális törtkifejezésekkel IX.

A `confrac` opcióval a `convert` a megadott kifejezés lánctört előállítását adja vissza.

```
kif6 = convert(kif6, confrac,x);
```

$$\frac{x^2+2x-3}{x^4+6x^3+13x^2+12x+4} = \cfrac{1}{x^2+4x+8+\cfrac{8}{x-\frac{3}{2}+\cfrac{9}{4\left(x+\frac{7}{2}\right)}}}$$

Az előző alfejezet átalakításait a racionális törtkifejezéseknél általánosabb, például gyök, trigonometrikus, vagy exponenciális függvényeket, illetve ezek inverzeit tartalmazó matematikai kifejezésekre is el tudja végezni a Maple. Az `expand`, a `collect`, a `factor` és a többi eljárás ilyenkor a megfelelő gyökös, trigonometrikus, exponenciális, stb. azonosságokat alkalmazza. Rendezési vagy konvertálási opciókban (a `sort` és a `convert` esetében) már általánosabb kifejezések is megadhatók.

Műveletek tetszőleges matematikai kifejezésekkel I.

Vizsgáljuk meg a kif1 trigonometrikus kifejezés lehetséges átalakításait.

```
restart;
kif1 := cos(3*x)*sin(x);
```

$$kif1 := \cos(3x)\sin(x)$$

```
'kif1' = expand(kif1);
'kif1' = collect(expand(kif1),sin(x));
'kif1' = factor(expand(kif1));
'kif1' = combine(kif1);
'kif1' = normal(kif1);
'kif1' = sort(combine(kif1), sin(2*x));
'kif1' = convert(kif1, sin);
```

$$kif1 = 4\sin(x)(\cos(x))^3 - 3\sin(x)\cos(x)$$
$$kif1 = \left(4(\cos(x))^3 - 3\cos(x)\right)\sin(x)$$
$$kif1 = \sin(x)\cos(x)\left(4(\cos(x))^2 - 3\right)$$
$$kif1 = -1/2\sin(2x) + 1/2\sin(4x)$$
$$kif1 = \cos(3x)\sin(x)$$
$$kif1 = -1/2\sin(2x) + 1/2\sin(4x)$$
$$kif1 = \sin\left(3x + \frac{1}{2}\pi\right)\sin(x)$$

Műveletek tetszőleges matematikai kifejezésekkel IIa.

Mivel a kifejezésekben szereplő függvények néha nem mindenütt értelmezettek, vagy nem mindenütt adnak valós értéket, sokszor az `assume` eljárással segítenünk kell a Maple-nek. Meg kell adnunk, hogy mit tételezünk fel az egyes változókról vagy részkifejezésekről.
A kif2 kifejezésben szereplő logaritmusokkal például nem tud mit kezdeni a Maple. Az ismert azonosságok csak pozitív számokra teljesülnek!

```
restart;
kif2 := 2*ln(x^3) -3*ln(y) + ln(x*y);
'kif2' = combine(kif2);
```

$$kif2 := 2\ln\left(x^3\right) - 3\ln\left(y\right) + \ln\left(xy\right)$$
$$kif2 = 2\ln\left(x^3\right) - 3\ln\left(y\right) + \ln\left(xy\right)$$

Műveletek tetszőleges matematikai kifejezésekkel IIb.

Ha föltesszük, hogy az ln függvény argumentumai pozitívak (a hozzárendelt föltétel létezését mutatja a ~ karakter), már alkalmazható több azonosság:

```
assume(x::positive, y::positive);
'kif2' = combine(kif2);
'kif2' = expand(kif2);
'kif2' = factor(kif2);
'kif2' = normal(kif2);
```

$$kif2 = \ln\left(\frac{x\sim^7}{y\sim^2}\right)$$
$$kif2 = 7\ln(x\sim) - 2\ln(y\sim)$$
$$kif2 = 6\ln(x\sim) - 3\ln(y\sim) + \ln(x\sim\ y\sim)$$
$$kif2 = 6\ln(x\sim) - 3\ln(y\sim) + \ln(x\sim\ y\sim)$$

Műveletek tetszőleges matematikai kifejezésekkel III.

Ebben a példában még több azonos átalakítást vet be a Maple. Hasonlítsuk össze az egyes parancsok eredményeit.

```
restart;
kif3 := sqrt(ln(x^2))*sqrt((sin(x))^2*cos(x)^2)+ sqrt(x^2-1)/sqrt(x-1);
assume (x>1);
'kif3' = expand(kif3);
'kif3' = factor(kif3);
'kif3' = combine(kif3);
'kif3' = normal(kif3);
```

$$kif3 := \sqrt{\ln(x^2)}\ \sqrt{\sin(x)^2\cos(x)^2} + \frac{\sqrt{x^2-1}}{\sqrt{x-1}}$$

$$kif3 = \sqrt{2}\ \sqrt{\ln(x\sim)}\ \sqrt{\sin(x\sim)^2\cos(x\sim)^2} + \frac{\sqrt{x\sim^2-1}}{\sqrt{x\sim-1}}$$

$$kif3 = \frac{\sqrt{2}\ \sqrt{\ln(x\sim)}\ \sqrt{\sin(x\sim)^2\cos(x\sim)^2}\ \sqrt{x\sim-1} + \sqrt{(x\sim-1)\ (x\sim+1)}}{\sqrt{x\sim-1}}$$

$$kif3 = \frac{1}{4}\ \frac{2\sqrt{\ln(x\sim)\ (1-\cos(4\,x\sim))\ (x\sim-1)} + 4\sqrt{x\sim^2-1}}{\sqrt{x\sim-1}}$$

$$kif3 = \frac{\sqrt{2}\ \sqrt{\ln(x\sim)}\ \sqrt{\sin(x\sim)^2\cos(x\sim)^2}\ \sqrt{x\sim-1} + \sqrt{x\sim^2-1}}{\sqrt{x\sim-1}}$$

Műveletek tetszőleges matematikai kifejezésekkel IV.

A `simplify` eljárás az ismert matematikai azonosságok szerinti egyszerűsítéseket hajt végre tetszőleges matematikai kifejezéseken.

```
restart;
kif1 := (x-b)*(x-c)/(a-b)*(a-c)+(x-c)*(x-a)/(b-c)*(b-a)+(x-a)*(x-b)/(c-a)*(c-b);
kif2 := exp(1-x)*exp(2*x)/exp(x);
kif3 := cos(3*x)/cos(x);
```

$$kif1 := \frac{(x-b)(x-c)}{(a-b)(a-c)} + \frac{(x-c)(x-a)}{(b-c)(b-a)} + \frac{(x-a)(x-b)}{(c-a)(c-b)}$$

$$kif2 := \frac{e^{1-x}e^{2x}}{e^{x}}$$

$$kif3 := \frac{\cos(3x)}{\cos(x)}$$

```
'kif1' = simplify(kif1);
'kif2' = simplify(kif2);
'kif3' = simplify(kif3);
```

$$kif1 = 1$$

$$kif2 = e$$

$$kif3 = 4\cos(x)^2 - 3$$

Műveletek tetszőleges matematikai kifejezésekkel V.

Mivel a Maple a beépített függvényeket alapjában komplex változós és komplex értékű függvényekként kezeli, néha (teljesen logikusan) nem az általunk várt eredményeket kapjuk.
A `symbolic` opcióval szimbolikusan (szintaktikusan) egyszerűsít a Maple, nem zár ki bizonyos speciális esetekben nem teljesülő azonosságokat sem. Emiatt elképzelhető, hogy az általa elvégzett egyszerűsítés bizonyos körülmények között nem érvényes.

```
kif1 := (x^2)^(1/2);
kif2 := 2*ln(x^3) -3*ln(y) + ln(x*y);
```

$$kif1 := \sqrt{x^2}$$
$$kif2 := 2\ln\left(x^3\right) - 3\ln\left(y\right) + \ln\left(xy\right)$$

Mi a véleményünk a kapott eredményekről?

```
'kif1' = simplify(kif1, symbolic);
'kif2' = simplify(kif2,symbolic);
```

$$kif1 = x$$
$$kif2 = 7\ln\left(x\right) - 2\ln\left(y\right)$$

Műveletek tetszőleges matematikai kifejezésekkel VI.

Ha eleve ismertek a kifejezésben szereplő változónevekre vonatkozó megkötések, akkor a `simplify` megsegítésére két formában is használhatjuk az `assume` lehetőségeit. Vagy magába a `simplify`-ba beépített opciókként, vagy még a `simplify` hívása előtt külön felsorolva.

```
restart; kif1 := (x^2)^(1/2);
'kif1' = simplify(kif1, assume=positive);
assume(x>0);
'kif1' = simplify(kif1);
```

$$kif1 := \sqrt{x^2}$$
$$kif1 = x$$
$$kif1 = x\sim$$

```
restart; kif2 := 2*ln(x^3) -3*ln(y) + ln(x*y);
'kif2' = simplify(kif2, assume=[x>0,y>0]);
assume(x>0,y>0);
'kif2' = simplify(kif2);
```

$$kif2 := 2\ln\left(x^3\right) - 3\ln(y) + \ln(xy)$$
$$kif2 = 7\ln(x) - 2\ln(y)$$
$$kif2 = 7\ln(x\sim) - 2\ln(y\sim)$$

Műveletek tetszőleges matematikai kifejezésekkel VII.

A kifejezésben szereplő változókra vonatkozó kiegészítő feltételeket (azonosságokat) is megadhatunk egyenletek formájában.

```
restart;  kif1 := a^3+b^3;
'kif1' = simplify(kif1, {a+b=1});
```

$$kif1 := a^3 + b^3$$
$$kif1 = 3\,a^2 - 3\,a + 1$$

Itt mi magunk írjuk elő, hogy teljesüljenek a logaritmus azonosságai:

```
restart;  kif2 := 2*ln(x^3) -3*ln(y) + ln(x*y);
'kif2' = simplify(kif2, {ln(x*y)= ln(x) +ln(y)});
'kif2' = simplify(kif2, {ln(x^3)= 3*ln(x),ln(x*y)= ln(x) +ln(y)});
```

$$kif2 := 2\,\ln\left(x^3\right) - 3\,\ln\left(y\right) + \ln\left(xy\right)$$
$$kif2 = 2\,\ln\left(x^3\right) - 2\,\ln\left(y\right) + \ln\left(x\right)$$
$$kif2 = -2\,\ln\left(y\right) + 7\,\ln\left(x\right)$$

Műveletek tetszőleges matematikai kifejezésekkel VIII.

A `?simplify` Help oldal szerint különböző kulcsszavakkal is megadhatjuk, hogy milyen jellegű egyszerűsítéseket alkalmazzon a Maple, például `size` (méret szerint), `ln` (logaritmusos), `trig` (trigonometrikus), `radical` (gyökös) stb. Melyik a „legjobb” eredmény?

```
restart;  kif4:= sin(x)^2+ln(2*x)+cos(x)^2+ (x^2-1)*(x+1)^2;
'kif4e'     = simplify(kif4);
'kif4esize' = simplify(kif4, size);
'kif4eln'   = simplify(kif4,ln);
'kif4etrigln' = simplify(kif4,trig,ln);
```

$$kif4 := \sin(x)^2 + \ln(2x) + \cos(x)^2 + \left(x^2-1\right)(x+1)^2$$
$$kif4e = x^4 + 2x^3 + \ln(2) + \ln(x) - 2x$$
$$kif4esize = x^4 + 2x^3 + \sin(x)^2 + \cos(x)^2 + \ln(2x) - 2x - 1$$
$$kif4eln = \sin(x)^2 + \ln(2) + \ln(x) + \cos(x)^2 + \left(x^2-1\right)(x+1)^2$$
$$kif4etrigln = x^4 + 2x^3 + \ln(2) + \ln(x) - 2x$$

Műveletek tetszőleges matematikai kifejezésekkel IX.

A Maple kifejezések egyenlőségének eldöntésére tartalmaz egy `testeq` nevű random polinomiális idejű tesztelő eljárást.
Ha ez hamisat (`false`) ad vissza, a két kifejezés biztosan nem egyenlő, de az igaz (`true`) érték esetén csak „nagy valószínűséggel” egyeznek meg.
Bonyolultabb kifejezéseknél néha a `FAIL` értéket adja vissza („nem tud dönteni”).

```
'kif4' = kif4;
kif4rovid := sort(combine( simplify(kif4, trig) ));
```

$$kif4 = \sin(x)^2 + \ln(2x) + \cos(x)^2 + \left(x^2 - 1\right)(x+1)^2$$
$$kif4rovid := x^4 + 2x^3 - 2x + \ln(2x)$$

```
testeq(kif4,kif4rovid);
testeq(kif4,kif4rovid -1);
testeq((abs(kif4))^2,(abs(kif4rovid))^2);
```

$$true$$
$$false$$
$$FAIL$$

4. Egyenletek és egyenletrendszerek megoldása

A Maple képes különböző típusú – nemcsak algebrai – egyenletek, egyenlőtlenségek, egyenletrendszerek és egyenlőtlenség rendszerek pontos (szimbolikus) megoldására a `solve` eljárás használatával. Speciális alakú feladatokra több speciális megoldó eljárás is létezik, így például

`LinearSolve` : lineáris egyenletrendszerek megoldására, a `LinearAlgebra` csomagban

`PolynomialSystem` : algebrai egyenletrendszerek megoldására, a `SolveTools` csomagban

`roots` : algebrai egyenletek adott számtest fölötti gyökeinek meghatározására

Ha az előző eszközökkel mégsem sikerült megtalálni a megoldásokat, vagy eleve elegendő csak közelítő megoldások meghatározása, akkor használhatjuk az `fsolve` eljárást, amely numerikus módszerekkel közelíti a gyököket.
Ebben a fejezetben a következő témaköröket vizsgáljuk:

- egyenletek szimbolikus és közelítő megoldása

A Maple képes különböző típusú – nemcsak algebrai – egyenletek, egyenlőtlenségek, egyenletrendszerek és egyenlőtlenség rendszerek pontos (szimbolikus) megoldására a `solve` eljárás használatával. Speciális alakú feladatokra több speciális megoldó eljárás is létezik, így például

`LinearSolve` : lineáris egyenletrendszerek megoldására, a `LinearAlgebra` csomagban

`PolynomialSystem` : algebrai egyenletrendszerek megoldására, a `SolveTools` csomagban

`roots` : algebrai egyenletek adott számtest fölötti gyökeinek meghatározására

Ha az előző eszközökkel mégsem sikerült megtalálni a megoldásokat, vagy eleve elegendő csak közelítő megoldások meghatározása, akkor használhatjuk az `fsolve` eljárást, amely numerikus módszerekkel közelíti a gyököket.
Ebben a fejezetben a következő témaköröket vizsgáljuk:

- egyenletek szimbolikus és közelítő megoldása
- egyenletrendszerek szimbolikus és közelítő megoldása

A Maple képes különböző típusú – nemcsak algebrai – egyenletek, egyenlőtlenségek, egyenletrendszerek és egyenlőtlenség rendszerek pontos (szimbolikus) megoldására a `solve` eljárás használatával. Speciális alakú feladatokra több speciális megoldó eljárás is létezik, így például

`LinearSolve` : lineáris egyenletrendszerek megoldására, a `LinearAlgebra` csomagban

`PolynomialSystem` : algebrai egyenletrendszerek megoldására, a `SolveTools` csomagban

`roots` : algebrai egyenletek adott számtest fölötti gyökeinek meghatározására

Ha az előző eszközökkel mégsem sikerült megtalálni a megoldásokat, vagy eleve elegendő csak közelítő megoldások meghatározása, akkor használhatjuk az `fsolve` eljárást, amely numerikus módszerekkel közelíti a gyököket.
Ebben a fejezetben a következő témaköröket vizsgáljuk:

- egyenletek szimbolikus és közelítő megoldása
- egyenletrendszerek szimbolikus és közelítő megoldása
- egyenlőtlenségek megoldása

A Maple képes különböző típusú – nemcsak algebrai – egyenletek, egyenlőtlenségek, egyenletrendszerek és egyenlőtlenség rendszerek pontos (szimbolikus) megoldására a `solve` eljárás használatával. Speciális alakú feladatokra több speciális megoldó eljárás is létezik, így például

`LinearSolve` : lineáris egyenletrendszerek megoldására, a `LinearAlgebra` csomagban

`PolynomialSystem` : algebrai egyenletrendszerek megoldására, a `SolveTools` csomagban

`roots` : algebrai egyenletek adott számtest fölötti gyökeinek meghatározására

Ha az előző eszközökkel mégsem sikerült megtalálni a megoldásokat, vagy eleve elegendő csak közelítő megoldások meghatározása, akkor használhatjuk az `fsolve` eljárást, amely numerikus módszerekkel közelíti a gyököket.
Ebben a fejezetben a következő témaköröket vizsgáljuk:

- egyenletek szimbolikus és közelítő megoldása
- egyenletrendszerek szimbolikus és közelítő megoldása
- egyenlőtlenségek megoldása
- rekurzív egyenletek megoldása

Egyenletek I.

Egyenletet (egyenlőtlenséget) úgy kapunk, hogy egyenlőség vagy egyenlőtlenségjellel kapcsolunk össze két kifejezést.
A Maple képes az egyenletek és egyenlőtlenségek, mint önálló objektumok kezelésére. Ez azt is jelenti, hogy egyszerű értékadással változókhoz értékként egyenleteket (egyenlőtlenségeket) rendelhetünk.
Az `lhs` és rhs segédeljárásokat egyenletekre (egyenlőtlenségekre) alkalmazva az egyenlet (egyenlőtlenség) bal, illetve jobb oldalán álló kifejezést kapjuk.

```
eq1 := sqrt(x^2-1) = x^2+2*x+2;
op(0..nops(eq1), eq1);
lhs(eq1), rhs(eq1);
eq1uj := lhs(eq1) - rhs(eq1) = 0;
```

$$eq1 := \sqrt{x^2-1} = x^2+2x+2$$
$$`=`, \sqrt{x^2-1}, x^2+2x+2$$
$$\sqrt{x^2-1}, x^2+2x+2$$
$$\sqrt{x^2-1} - x^2 - 2x - 2 = 0$$
$$eq1uj := \sqrt{x^2-1} - x^2 - 2x - 2 = 0$$

```
eq2 := sin(x)^2+cos(x)^2 + cos(2*x) <= ln(2*x);
op(0..nops(eq2), eq2);
lhs(eq2), rhs(eq2);
eq2uj := lhs(eq2) - rhs(eq2) <= 0;
```

$$eq2 := \sin(x)^2 + \cos(x)^2 + \cos(2x) \le \ln(2x)$$
$$`<=`, \sin(x)^2 + \cos(x)^2 + \cos(2x), \ln(2x)$$
$$eq2uj := \sin(x)^2 + (\cos(x))^2 + \cos(2x) - \ln(2x) \le 0$$

Egyenletek II.

A korábban látott kifejezés-átalakító eljárások általában egyenletekre is alkalmazhatók: a szóban forgó átalakítást a MAPLE az egyenlet mindkét oldalán végrehajtja.
Hasonlítsuk össze az alábbi hívások eredményeit.

```
eq3:= cos(3*x)*sin(x) = x*(x^2+1);
```

$$eq3 := \cos(3x)\sin(x) = x\left(x^2+1\right)$$

```
expand(eq3);
collect(expand(eq3),sin(x));
factor(expand(eq3));
combine(eq3);
simplify(eq3);
normal(eq3);
```

$$4\sin(x)(\cos(x))^3 - 3\sin(x)\cos(x) = x^3 + x$$
$$\left(4(\cos(x))^3 - 3\cos(x)\right)\sin(x) = x^3 + x$$
$$\sin(x)\cos(x)\left(4(\cos(x))^2 - 3\right) = x\left(x^2+1\right)$$
$$1/2\sin(4x) - 1/2\sin(2x) = x\left(x^2+1\right)$$
$$\sin(x)\cos(x)\left(4(\cos(x))^2 - 3\right) = x\left(x^2+1\right)$$
$$\cos(3x)\sin(x) = x\left(x^2+1\right)$$

Egyenletek III.

Megfelelő helyettesítések, például a `subs` alkalmazásával, sokszor egyszerűbb feladatot kapunk:

```
eq4 := (ln(x))^2 - 2*ln(x) +1 = 0;
```

$$eq4 := (\ln(x))^2 - 2\ln(x) + 1 = 0$$

```
eq5 := subs(ln(x) = y, eq4);
```

$$eq5 := y^2 - 2y + 1 = 0$$

```
factor(eq5);
```

$$(y-1)^2 = 0$$

A `map` függvénnyel tetszőleges, általunk definiált átalakításokat is végezhetünk az egyenleteken. Az alábbi lépések a gyökös egyenletek „szokásos” megoldási menetét követik.

```
eq1;
map(z -> z^2, eq1);
expand(%);
lhs(%) -rhs(%) = 0;
```

$$\sqrt{x^2-1} = x^2 + 2x + 2$$
$$x^2 - 1 = \left(x^2 + 2x + 2\right)^2$$
$$x^2 - 1 = x^4 + 4x^3 + 8x^2 + 8x + 4$$
$$-x^4 - 4x^3 - 7x^2 - 8x - 5 = 0$$

Egyenletek IV.

Az egyenletekkel való számolási szabályok áttekintése után lássuk a `solve` használatát!
Lehetséges hívásai:
`solve(`**kifejezés**`)`
`solve(`**egyenlet**`)`
`solve(`**egyenlet**, **ismeretlen**`)`
Az első változat azzal ekvivalens, mintha a 0-ra redukált, **kifejezés** `= 0` alakú egyenletet oldatnánk meg.
A harmadik változatra valójában csak paraméteres egyenletek megoldásakor van szükség, mert ekkor nem tudja a Maple, hogy mit tekintsen ismeretlennek.
A `solve` visszaadott értéke általában a megoldásokból álló kifejezéssorozat. Az alábbi példákban a `solve` visszaadott értékét a {} halmazzárójelbe tesszük – így kapjuk az egyenlet megoldáshalmazát.

```
restart;
eq1 := cos(x) = sin(x);       M := {solve(eq1)};
eq2 := x^3 + 3*a*x  = 0;      M := {solve(eq2,x)};
eq3 := abs(x^2-10*x+20) = 4; M := {solve(eq3)};
```

$$eq1 := \cos(x) = \sin(x)$$
$$M := \{1/4\,\pi\}$$
$$eq2 := x^3 + 3\,ax = 0$$
$$M := \left\{0,\ \sqrt{-3\,a},\ -\sqrt{-3\,a}\right\}$$
$$eq3 := \left|x^2 - 10\,x + 20\right| = 4$$
$$M := \{2, 4, 6, 8\}$$

Ha az `_EnvAllSolutions` rendszerváltozó értéke `true`, akkor a `solve` törekszik valóban az összes megoldás megkeresésére. Figyeljük meg, hogy most az `eq1` trigonometrikus egyenlet összes megoldását megadta (`_Z1` egész értékű paramétert jelöl), ellentétben a korábbi egyetlen megoldással.

```
_EnvAllSolutions:=true: eq1 := cos(x) = sin(x); M := {solve(eq1)};
```

$$\cos(x) = \sin(x)$$
$$M := \{1/4\,\pi + \pi\_Z1\sim\}$$

Egyenletek V.

Előfordulhat, hogy a `solve` nem talál megoldást, ezt látjuk az `eq4` egyenletnél. Az `eq5` és `eq6` esetében csak implicit alakban, `RootOf`-os kifejezésként kapjuk a megoldást. Az ezekkel való számoláshoz lásd `?RootOf`.
A $RootOf(\_Z^4 + \_Z^3 + 2\_Z^2 + 2\_Z + 1, index = 1)$ jelentése: az $\_Z$ segédváltozóval fölírt (negyedfokú, algebrai) egyenlet első gyöke.
Az `allvalues` eljárás a paramétereként kapott `RootOf`-os kifejezés lehetséges értékeit számolja ki, és vagy explicit gyökös kifejezésekként adja vissza, mint az `eq5` esetében, vagy minden lehetséges értékre egy lebegőpontos közelítést határoz meg, mint az `eq6` példában.

```
eq4 := sqrt(x-8)+sqrt(x) = 2;      M := {solve(eq4,x)};
eq5 := x^2 + x + 1/(x^2+2) = 0;    M := {solve(eq5)}; Mexpl := allvalues(M); Mkoz := evalf(%);
eq6 := sin(x)=3*x/Pi;              M := {solve(eq6)}; Mexpl := allvalues(M);
```

$$eq4 := \sqrt{x-8} + \sqrt{x} = 2$$
$$M := \{\}$$
$$eq5 := x^2 + x + \left(x^2+2\right)^{-1} = 0$$
$$M := \left\{RootOf\left(\_Z^4 + \_Z^3 + 2\_Z^2 + 2\_Z + 1, index = 1\right), RootOf\left(\_Z^4 + \_Z^3 + 2\_Z^2 + 2\_Z + 1, index = 2\right),\right.$$
$$\left. RootOf\left(\_Z^4 + \_Z^3 + 2\_Z^2 + 2\_Z + 1, index = 3\right), RootOf\left(\_Z^4 + \_Z^3 + 2\_Z^2 + 2\_Z + 1, index = 4\right)\right\}$$
$$Mexpl := \left\{-1/4 - 1/4\,i\sqrt{3} - 1/4\,\sqrt{-10 - 6\,i\sqrt{3}}, -1/4 - 1/4\,i\sqrt{3} + 1/4\,\sqrt{-10 - 6\,i\sqrt{3}},\right.$$
$$\left. -1/4 + 1/4\,i\sqrt{3} - 1/4\,\sqrt{-10 + 6\,i\sqrt{3}}, -1/4 + 1/4\,i\sqrt{3} + 1/4\,\sqrt{-10 + 6\,i\sqrt{3}}\right\}$$
$$Mkoz := \{-0.6217444142 - 0.4405969990\,i, -0.6217444142 + 0.4405969990\,i,$$
$$0.1217444142 - 1.306622403\,i, 0.1217444142 + 1.306622403\,i\}$$
$$eq6 := \sin(x) = 3\,\frac{x}{\pi}$$
$$M := \{RootOf\left(-\sin\left(\_Z\right)\pi + 3\,\_Z\right)\}$$
$$Mexpl := \{RootOf\left(-\sin\left(\_Z\right)\pi + 3\,\_Z, 0.5235987756\right)\}, \{RootOf\left(-\sin\left(\_Z\right)\pi + 3\,\_Z, -0.5235987756\right)\},$$
$$\{RootOf\left(-\sin\left(\_Z\right)\pi + 3\,\_Z, 0.0\right)\}$$

Egyenletek VI.

Figyeljük meg, hogy a `solve` paramétereit a [] lista vagy a {} halmaz zárójelek között is megadhatjuk, de ekkor az előzőektől eltérő, halmaz vagy lista alapú eredményeket kapunk.
Röviden: ha az ismeretlent (egy elemű) listában adjuk meg, akkor az eredményt is (egy elemű) listák listájaként kapjuk, a többi esetben meg mindig egy elemű halmazokból álló kifejezéssorozatban kapjuk a megoldást.
Később látni fogjuk, hogy egyenletrendszereknél is hasonló hatása van az eltérő szintaxisnak.

```
M := solve(x^2-x-2,[x]);
M := solve([x^2-x-2],[x]);
```

$$[[x = 2], [x = -1]]$$
$$[[x = 2], [x = -1]]$$

```
M := solve({x^2-x-2}, x );
M := solve([x^2-x-2], x );
M := solve( x^2-x-2, {x});
M := solve({x^2-x-2},{x});
M := solve([x^2-x-2],{x});
```

$$\{x = 2\}, \{x = -1\}$$
$$\{x = 2\}, \{x = -1\}$$
$$\{x = 2\}, \{x = -1\}$$
$$\{x = 2\}, \{x = -1\}$$
$$\{x = 2\}, \{x = -1\}$$

Egyenletek VII.

Ne feledkezzünk meg arról sem, hogy bármennyire is bízunk a Maple-ben, célszerű az egyenletek `solve`-val kapott megoldásait behelyettesítéssel ellenőrizni.
Az ellenőrzésre több lehetőségünk is van, részben attól függően, hogy milyen formában (milyen adattípusban) kaptuk meg a megoldásokat. Néhány példa:

```
restart; eq3 := abs(x^2-10*x+20) = 4; M := {solve(eq3)};
```

$$|x^2 - 10x + 20| = 4$$
$$M := \{2, 4, 6, 8\}$$

```
subs(x = M[1],eq3);
simplify(%);
eval(eq3, x= M[2]);
```

$$|4| = 4$$
$$4 = 4$$
$$4 = 4$$

```
x := M[4]; eq3;
x := 'x': # ez célszerű a további számolás miatt
assign(x = M[2]); eq3;
x := 'x': # ez célszerű a további számolás miatt
```

$$x := 8$$
$$4 = 4$$
$$4 = 4$$

```
M := solve({eq3},x);
subs(M[1],eq3); simplify(%);
eval(eq3, M[2]);
```

$$M := \{x = 2\}, \{x = 8\}, \{x = 4\}, \{x = 6\}$$
$$|4| = 4$$
$$4 = 4$$
$$4 = 4$$

Egyenletek VIII.

Egyenletek megoldásán túl a solve segítségével azonosságok teljesülését is vizsgálhatjuk. A `solve(identity(`**kif1** = **kif2**`, x), a1,a,2,...ak)` alakú hívásnál a Maple azt számolja ki, hogy a **kif1** és **kif2** kifejezésekben szereplő `a1, a2, ..., ak` paraméterek mely értékeire lesz **kif1**=**kif2** (vagyis mikor teljesül bármely x-re az egyenlőség.)
Két egyszerű példa:

```
solve(identity(a*(x+1)^2 + b*x = x^2 + 2*x + 1, x), [a,b]);
```

$$a = 1, b = 0$$

```
solve(identity(sin(m*x) = n*sin(x)*cos(x), x), [m,n]);
```

$$m = -2, n = -2, m = 0, n = 0, m = 2, n = 2$$

Egyenletek IX.

Egyenletek gyökeinek közelítő (numerikus) meghatározására az `fsolve` használható. Jegyezzük meg, hogy a `solve` is rögtön közelítő módszerekre vált át, ha a megoldandó egyenlet lebegőpontos számot is tartalmaz. Eltérések a `solve`-hoz képest:

- algebrai egyenletek esetében az `fsolve` általában minden gyökre kiszámol egy közelítést, de egyébként csak egy (a `complex` opció megadását kivéve) valós megoldást ad vissza;
- a megoldandó egyenlet nem tartalmazhat paramétereket;
- különböző opciókkal megadhatjuk, hogy hol keresse az `fsolve` a gyököket;
- ha nem talál a hívás feltételeinek megfelelő gyököt, vagy üres kifejezéssel, vagy a kiértékeletlen eljáráshívással tér vissza.

```
M         := solve(x^4 + 4*x^2+ 1 = 0);
Mfloat    := solve(x^4 + 4*x^2+ 1.0 = 0);
Mf        := fsolve(x^4 + 4*x^2+ 1 = 0);
Mfcomplex := fsolve(x^4 + 4*x^2+ 1 = 0, complex);
```

$$M := 1/2\,i\left(\sqrt{6}+\sqrt{2}\right), -1/2\,i\left(\sqrt{6}+\sqrt{2}\right), 1/2\,i\left(\sqrt{6}-\sqrt{2}\right), -1/2\,i\left(\sqrt{6}-\sqrt{2}\right)$$

$$Mfloat := 1.931851653\,I, -1.931851653\,I, 0.5176380902\,I, -0.5176380902\,I$$

$$Mf :=$$

$$Mfcomplex := -1.931851652578136 62\,I, -0.517638090205041590\,I,$$
$$0.517638090205041590\,I, 1.931851652578136 62\,I$$

Egyenletek X.

Az fsolve további opcióiról a Helpből tájékozódhatunk, lásd `?fsolve,details`. A legfontosabbak:

- `fsolve(`**egyenlet, ismeretlen,** `fulldigits )` a számítások minden részletét a Digits változó értékének megfelelő pontossággal végezze;
- `fsolve(`**egyenlet, ismeretlen, alsóhatár..felsőhatár**`)` a megadott intervallumban keresse a gyököt;
- `fsolve(`**egyenlet, ismeretlen** = **alsóhatár..felsőhatár**`)` mint az előző;
- `fsolve(`**egyenlet, ismeretlen** = **kezdőérték**`)` a kezdőérték környezetében keressen gyököt;
- `fsolve(`**egyenlet, ismeretlen ,** `avoid = ... )` hol *ne* keressen gyököt.

```
restart;  eq := sqrt(1+2*x^2) -sqrt(1+x^2) = 1;  solve(eq);
```

$$eq := \sqrt{2x^2+1} - \sqrt{x^2+1} = 1$$
$$\sqrt{3+2\sqrt{3}}, -\sqrt{3+2\sqrt{3}}$$

```
fsolve(eq, x);
fsolve(eq, x,  -10..10);
fsolve(eq, x = -10..10);
fsolve(eq, x = -1..1); # itt nincs gyök!
fsolve(eq, x = -5);
fsolve(eq, x = -10..10, avoid = {x=2.5} );
fsolve(eq, x, avoid = {x=2.5} );
```

$$2.542459757$$
$$2.542459757$$
$$2.542459757$$
$$fsolve\left(\sqrt{2x^2+1} - \sqrt{x^2+1} = 1, x, -1\ldots 1\right)$$
$$-2.542459757$$
$$2.542459757$$
$$2.542459757$$

Egyenletrendszerek I.

A `solve` eljárás képes egyenletrendszerek megoldására is. Ehhez a megoldandó rendszert egyenletek halmazaként vagy listájaként kell felírni.
Mindig szükség van az ismeretlenek megadására, ez történhet szintén listában vagy halmazban.
A visszaadott megoldások adattípusa halmazokból álló kifejezéssorozat, vagy listák listája, egy megoldást egy (egyenlőségekből álló) halmaz, vagy egy listaelem ír le.
A `RootOf`-os eredmények miatt itt is sokszor hasznos lehet az `allvalues` alkalmazása.

Egyenletrendszerek II.

Tekintsük a következő egyszerű algebrai egyenletrendszert, mint egyenletek listáját.

```
restart;
Eq1 := x^2+2*x*y+5*y^2=32: Eq2 := x+3*y=2: Erendszer := [Eq1, Eq2];
```

$$Erendszer := [x^2 + 2\,xy + 5\,y^2 = 32, x + 3\,y = 2]$$

Megoldatjuk a rendszert a `solve` eljárással, s tájékozódásul lebegőpontosra konvertáljuk a kapott két megoldást.

```
M := solve(Erendszer, {x,y});
```

$$M := \left\{x = -3\,RootOf\left(2\_Z^2 - 2\_Z - 7\right) + 2, y = RootOf\left(2\_Z^2 - 2\_Z - 7\right)\right\}$$

```
M := allvalues(M);
```

$$M := \left\{x = 1/2 - 3/2\,\sqrt{15}, y = 1/2 + 1/2\,\sqrt{15}\right\}, \left\{x = 1/2 + 3/2\,\sqrt{15}, y = 1/2 - 1/2\,\sqrt{15}\right\}$$

```
Mfloat := evalf(M);
```

$$Mfloat := \{x = -5.309475019, y = 2.436491673\}, \{x = 6.309475019, y = -1.436491673\}$$

Egyenletrendszerek III.

Az egyenletrendszerbe való behelyettesítéssel, a `subs` vagy az `eval` használatával ellenőrizhetjük a kapott megoldások helyességét

```
M;
```

$$\left\{x = 1/2 - 3/2\ \sqrt{15}, y = 1/2 + 1/2\ \sqrt{15}\right\}, \left\{x = 1/2 + 3/2\ \sqrt{15}, y = 1/2 - 1/2\ \sqrt{15}\right\}$$

```
subs(M[1], Erendszer);
simplify(%);
```

$$\left[\left(1/2 - 3/2\ \sqrt{15}\right)^2 + 2\left(1/2 - 3/2\ \sqrt{15}\right)\left(1/2 + 1/2\ \sqrt{15}\right) + 5\left(1/2 + 1/2\ \sqrt{15}\right)^2 = 32, 2 = 2\right]$$
$$[32 = 32, 2 = 2]$$

```
eval(Erendszer, M[2]);
simplify(%);
```

$$\left[\left(1/2 + 3/2\ \sqrt{15}\right)^2 + 2\left(1/2 + 3/2\ \sqrt{15}\right)\left(1/2 - 1/2\ \sqrt{15}\right) + 5\left(1/2 - 1/2\ \sqrt{15}\right)^2 = 32, 2 = 2\right]$$
$$[32 = 32, 2 = 2]$$

Egyenletrendszerek IV.

Ha az ismeretleneket listában adjuk meg, a `solve` listákból álló kifejezésssorozatot ad vissza – ez elég nehézkesen kezelhető.

```
Eq1 := x^2+2*x*y+5*y^2=32: Eq2 := x+3*y=2:
Erendszer := [Eq1, Eq2];
M := solve(Erendszer, [x,y]);
M := allvalues(M);
```

$$Erendszer := [x^2 + 2\,xy + 5\,y^2 = 32, x + 3\,y = 2]$$
$$M := \left[\left[x = -3\,RootOf\left(2\_Z^2 - 2\,\_Z - 7\right) + 2, y = RootOf\left(2\_Z^2 - 2\,\_Z - 7\right)\right]\right]$$
$$M := \left[\left[x = 1/2 - 3/2\,\sqrt{15}, y = 1/2 + 1/2\,\sqrt{15}\right]\right], \left[\left[x = 1/2 + 3/2\,\sqrt{15}, y = 1/2 - 1/2\,\sqrt{15}\right]\right]$$

Mivel algebrai egyenletrendszerünk van, használhatjuk a `PolynomialSystem` eljárást. Az előzőekkel megegyező megoldásokat kapunk.

```
SolveTools[PolynomialSystem](Erendszer, {x,y});
allvalues(%);
```

$$\left\{x = -3\,RootOf\left(2\_Z^2 - 2\,\_Z - 7\right) + 2, y = RootOf\left(2\_Z^2 - 2\,\_Z - 7\right)\right\}$$
$$\left\{x = 1/2 - 3/2\,\sqrt{15}, y = 1/2 + 1/2\,\sqrt{15}\right\}, \left\{x = 1/2 + 3/2\,\sqrt{15}, y = 1/2 - 1/2\,\sqrt{15}\right\}$$

Egyenletrendszerek V.

Persze nem csak algebrai egyenletrendszereket tud megoldani a Maple. Itt csak 2 megoldást ad vissza a `solve`, de az `_EnvAllSolutions:= true` értékadás után megkaphatjuk paraméteresen az összes (periodikus) megoldást.

```
Eq1:= cos(x) + cos(y) = 1/2: Eq2:= sin(x) + sin(y) = 1:
Erendszer2 := {Eq1,Eq2};
```

$$Erendszer2 := \{\cos(x) + \cos(y) = 1/2, \sin(x) + \sin(y) = 1\}$$

```
M := solve(Erendszer2, {x,y} );
M := allvalues(%);
Mfloat := evalf(M);
```

$$M := \{x = \arctan\left(1/4\, RootOf\left(5\_Z^2 - 20\_Z + 9\right), 5/4 - 1/2\, RootOf\left(5\_Z^2 - 20\_Z + 9\right)\right),$$
$$y = \arctan\left(-1/4\, RootOf\left(5\_Z^2 - 20\_Z + 9\right) + 1, -3/4 + 1/2\, RootOf\left(5\_Z^2 - 20\_Z + 9\right)\right)\}$$

$$M := \left\{x = \arctan\left(\frac{1/2 - 1/20\sqrt{55}}{1/4 + 1/10\sqrt{55}}\right), y = \arctan\left(\frac{1/2 + 1/20\sqrt{55}}{1/4 - 1/10\sqrt{55}}\right) + \pi\right\},$$
$$\left\{x = \arctan\left(\frac{1/2 + 1/20\sqrt{55}}{1/4 - 1/10\sqrt{55}}\right) + \pi, y = \arctan\left(\frac{1/2 - 1/20\sqrt{55}}{1/4 + 1/10\sqrt{55}}\right)\right\}$$

$$Mfloat := \{x = .1295521671, y = 2.084745269\}, \{x = 2.084745269, y = .1295521671\}$$

Egyenletrendszerek VI.

Figyeljük meg, hogy ha az előző rendszer jobboldali konstansait lebegőpontos számként adjuk meg, akkor az egyenleteknél említettekhez hasonlóan itt is lebegőpontos (közelítő) eredményt ad a `solve`.

```
 Eq1:=cos(x)+cos(y) = 0.5: Eq2:=sin(x) + sin(y) = 1.:
 Erendszer3 := [Eq1,Eq2];
M := solve(Erendszer3, {x,y} );
```

$$Erendszer3 := [\cos(x) + \cos(y) = 0.5, \sin(x) + \sin(y) = 1.0]$$
$$M := \{x = 0.1295521671, y = 2.084745268\}, \{x = 2.084745268, y = 0.1295521671\}$$

Egyenletrendszerek közelítő megoldását az `fsolve` végzi, így csak egy közelítő megoldást kapunk.

```
Eq1 := cos(x)+cos(y) = 1/2: Eq2 := sin(x) + sin(y) = 1:
Erendszer2 := [Eq1,Eq2];
fsolve(Erendszer2, {x,y} );
```

$$Erendszer2 := [\cos(x) + \cos(y) = 1/2, \sin(x) + \sin(y) = 1]$$
$$\{x = 134.0316367, y = 119.5100730\}$$

Egyenletrendszerek VII.

Az `fsolve` egyenleteknél megismert opcióit használva további megoldásokat is találhatunk. (Kérdés: ezek mennyire „új” megoldások, illetve mi következik az egyenletrendszer függvényeinek periodicitásából?)

```
fsolve(Erendszer2, {x=0,y=2});
fsolve(Erendszer2, {x,y}, 0..3);
fsolve(Erendszer2, {x,y}, avoid={x=0,y=2});
```

$$\{x = .1295521671, y = 2.084745268\}$$
$$\{x = 2.084745268, y = .1295521671\}$$
$$\{x = 134.0316367, y = 119.5100730\}$$

Egyenlőtlenségek I.

A `solve` felhasználható egyenlőtlenségek és egyenlőtlenség rendszerek megoldására is. A paraméterek használata hasonló az előzőekhez, a visszaadott megoldások viszont legtöbbször (valós) intervallumok, ezekre a Maple a `RealRange(a,b)` jelölést használja, amely az $[a, b]$ zárt intervallumnak felel meg. `Open(a)`, `Open(b)` a megfelelő oldalról nyitott intervallumot jelöl.

Egyenlőtlenségek II.

Hasznos trükk lehet a következő: mivel a `solve`-nak nincs külön opciója a gyökök szétválasztására (például ha csak a negatív gyökökre lennénk kíváncsiak), hozzávéve az egyenlethez az $x < 0$ egyenlőtlenséget, a rendszer megoldása a kívánt gyököt/gyököket adja.

```
M := solve(1/4*x - sqrt(x+2) +1/2 = 0);
M := solve({1/4*x - sqrt(x+2) +1/2 = 0, x <0});
```

$$M := -2, 14$$
$$M := \{x = -2\}$$

Egyenlőtlenségek III.

Néhány egyszerű példa következik. Ha halmazként adjuk meg a paramétereket, az output is halmaz lesz.
Az utolsó példánál csak a lebegőpontosra konvertált eredményt mutatjuk.

```
M1 := solve(abs(6*x-8)> 4*x+2,x);
M2 := solve(x^2>4,x);
M3 := solve({x^2-4<=0},x);
M4 := solve(sin(2*x)>4*sin(x)+2, x):
evalf(%);
```

$$RealRange\left(Open\left(5\right),\infty\right),RealRange\left(-\infty,Open\left(3/5\right)\right)$$
$$RealRange\left(-\infty,Open\left(-2\right)\right),RealRange\left(Open\left(2\right),\infty\right)$$
$$\{-2\le x,x\le 2\}$$
$$RealRange\left(Open\left(-9.077764689\right),Open\left(-7.128537614\right)\right),$$
$$RealRange\left(Open\left(-2.794579381\right),Open\left(-0.8453523065\right)\right),$$
$$RealRange\left(Open\left(3.488605927\right),Open\left(5.437833002\right)\right)$$

Egyenlőtlenségek IV.

Egyismeretlenes egyenlőtlenség-rendszert is megadhatunk. Figyeljük meg, hogy a különböző jelölések milyen hatással vannak az eredmény formátumára. (A helyzet hasonló, mint az egyenleteknél és az egyenletrendszereknél.)

```
solve([x^2-4<=0, abs(x-1)> abs(x-2)]);
solve([x^2-4<=0, abs(x-1)> abs(x-2)],x);
solve({x^2-4<=0, abs(x-1)> abs(x-2)},{x});
solve({x^2-4<=0, abs(x-1)> abs(x-2)},[x]);
```

$$\{x \le 2, 3/2 < x\}$$
$$\{x \le 2, 3/2 < x\}$$
$$\{x \le 2, 3/2 < x\}$$
$$[[x \le 2, 3/2 < x]]$$

Egyenlőtlenségek V.

Több ismeretlenes egyenlőtlenség rendszereket is megoldathatunk a `solve` segítségével. Például egy kétismeretlenes lineáris egyenlőtlenség rendszer megoldása így történhet:

```
solve({x-2*y <1, 3*x + y> 1}, {x,y});
```

$$\{-2/7 < y, x < 2\,y + 1, 1/3 - 1/3\,y < x\}$$

A `simplex` csomag eljárásai (lásd `?simplex`) általánosabb, lineáris egyenlőtlenség rendszerekből és célfüggvényből álló optimalizációs feladatokat kezelnek.

Rekurzív egyenletek I.

Rekurzív egyenleteket, így például a mértani sorozat n-dik tagját definiáló $a(n) = a(0)q^n$, vagy a Fibonacci sorozatot leíró $f(n+2) = f(n+1) + f(n)$ rekurziót, a Maple `rsolve` utasításával tudunk megoldani.
A megoldás egy végtelen sorozat, vagy másképpen a természetes számok halmazán definiált függvény. Az `rsolve` egy **kifejezést**, ennek a függvénynek a képletét adja vissza. Az egyértelmű megoldás előállításához általában szükséges néhány peremföltétel, a rekurzív sorozat első néhány tagjának megadása.
Az `rsolve` első paramétere a rekurzív definíció, vagy a rekurzív definícióból és a peremföltételekből álló halmaz, a második az ismeretlen függvény - ezt érdemes a kiértékelést gátló egyszeres aposztrófok között átadni, így az sem okoz gondot, ha korábban már kapott értéket.

Rekurzív egyenletek II.

Először egy mértani sorozatot állítunk elő. Az első változatban nem írjuk elő $a(0)$ értékét, ez a megoldás paramétere marad. az `a := n -> Mo;` sor állítja elő az $a(n)$ sorozatot, mint függvényt.

```
restart;
Mo := rsolve(a(n+1) = q*a(n), 'a');
Mo := rsolve({a(n+1) = q*a(n), a(0) = -2}, 'a');
seq(subs(n=j,Mo),j=0..10); # ez nehézkesebb
a := n->Mo;
seq(a(n), n=0..10); # ez elegánsabb
```

$$Mo := a(0)\,q^n$$
$$Mo := -2\,q^n$$
$$-2, -2\,q, -2\,q^2, -2\,q^3, -2\,q^4, -2\,q^5, -2\,q^6, -2\,q^7, -2\,q^8, -2\,q^9, -2\,q^{10}$$
$$a := n \to -2\,q^n$$
$$-2, -2\,q, -2\,q^2, -2\,q^3, -2\,q^4, -2\,q^5, -2\,q^6, -2\,q^7, -2\,q^8, -2\,q^9, -2\,q^{10}$$

Rekurzív egyenletek III.

A második példa egy nemlineáris rekurzióval definiált egyszerű sorozat.

```
sorozat := rsolve({b(n)*b(n-1) + b(n) - b(n-1) = 0, b(0)=alpha}, 'b(k)');
bfun := k -> sorozat;
seq(bfun(k), k=0..10);
```

$$sorozat := \frac{\alpha}{\alpha\, k + 1}$$

$$bfun := k \rightarrow sorozat$$

$$\alpha, \frac{\alpha}{\alpha + 1}, \frac{\alpha}{2\,\alpha + 1}, \frac{\alpha}{3\,\alpha + 1}, \frac{\alpha}{4\,\alpha + 1}, \frac{\alpha}{5\,\alpha + 1}, \frac{\alpha}{6\,\alpha + 1}, \frac{\alpha}{7\,\alpha + 1}, \frac{\alpha}{8\,\alpha + 1}, \frac{\alpha}{9\,\alpha + 1}, \frac{\alpha}{10\,\alpha + 1}$$

Rekurzív egyenletek IV.

Határozzuk meg az $f(n+2) = f(n+1) + f(n), f(0) = 1, f(1) = 1$ képlettel definiált Fibonacci-sorozat explicit képletét és állítsuk elő a sorozat első 11 tagját tartalmazó listát.

```
Fib := rsolve({f(n+2)=f(n+1)+f(n), f(0)=1, f(1)=1},f(n));
F := n -> Fib;
[seq(F(n), n=0..10)]: # a sok gyökjel miatt nem íratjuk ki
simplify(%);
```

$$Fib := \left(1/10\sqrt{5} + 1/2\right)\left(1/2\sqrt{5} + 1/2\right)^n + \left(1/2 - 1/10\sqrt{5}\right)\left(-1/2\sqrt{5} + 1/2\right)^n$$

$$F := n \to Fib$$

$$[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]$$

Rekurzív egyenletek V.

Az alábbi sorozat képlete még érdekesebb: képzetes tagok hatványaiból áll, holott a sorozat minden tagja valós szám.

```
 restart;
s := rsolve({x(n)-10*x(n-1)+50*x(n-2) = 0, x(0) = 0, x(1) = 10}, 'x(n)');
# Maple Magic:
s := evalc(Re(convert(s, expln)));
[seq(subs(n=j,s),j=0..10)]:
simplify((%));
```

$$s := I\,(5 - 5\,I)^n - I\,(5 + 5\,I)^n$$

$$s := 2\,\mathrm{e}^{1/2\,\ln(50)n}\sin\left(1/4\,\pi\,n\right)$$

$$[0, 10, 100, 500, 0, -25000, -250000, -1250000, 0, 62500000, 625000000]$$

```
x := n -> s;
seq(x(n), n = 0..10);
```

$$x := n \to s$$

$$0, 10, 100, 500, 0, -25000, -250000, -1250000, 0, 62500000, 625000000$$

12. előadás

5. Függvények

A Maple, mint komputeralgebra rendszer, eredendően matematikai **kifejezésekkel** való szimbolikus számolásra készült. Az előző előadás 3. fejezetében már bemutattuk az ehhez kapcsolódó lehetőségeket.
A matematikában használatos **függvények** teljesen más szintaktikus kategóriát alkotnak, őket másként kezeli a Maple. Ebben a fejezetben a függvények definiálásának és használatának szabályait tekintjük át.
A rendszer által ismert függvények listáját a `?inifcns` Help oldalon találjuk. (A Maple csomagok ezeken túl még további speciális függvényeket is definiálnak.)
Ezekkel a függvényekkel a Maple az analízisből ismert módon számol, tudja differenciálni és integrálni őket, ismeri a rájuk vonatkozó azonosságokat és egyszerűsítési szabályokat.

A legfontosabb függvények Maple jelöléseit az alábbi táblázat tartalmazza.

trigonometrikus függvények	sin, cos, tan, cot, sec, csc
trigonometrikus függvények inverzei	arcsin, arccos, arctan, arccot, arcsec, arccsc
hiperbolikus függvények	sinh, cosh, tanh, coth, sech, csch
hiperbolikus függvények inverzei	arcsinh, arccosh, arctanh, arccoth, arcsech, arccsch
exponenciális függvény	exp
természetes alapú logaritmus függvény	ln
négyzetgyök függvény	sqrt
abszolút érték	abs
előjel függvény	signum
maximum és minimum	max, min

Függvények I.

Az egyes függvények Help oldalait, például a `?sin` oldalt is érdemes megnézni.
A Maple beépített matematikai függvényeiről részletes információkat kaphatunk a `FunctionAdvisor` eljárás hívásával, lásd `?FunctionAdvisor`. A `describe` opció rövid leírást nyújt, a `display` kiír „mindent”, amit a Maple a megadott függvényről tud.
A tangens függvényről az alábbiakat tudjuk meg (a `display` eredményének csak az első pár sorát mutatjuk). A leírtak értelmezéséhez vegyük figyelembe, hogy a Maple a tangenst (és a többi elemi függvényt) mint komplex változós, komplex értékű függvényt tekinti.

```
FunctionAdvisor(describe, tan);
```

tan = *tangent function*

Függvények II.

```
FunctionAdvisor(display, tan);
```

```
The system is unable to compute the "asymptotic_expansion" for tan
tan belongs to the subclass "trig" of the class "elementary" and so,
in principle, it can be related to various of the 26 functions of those
classes - see FunctionAdvisor("trig"); and FunctionAdvisor("elementary");
```

$$describe = (\tan = tangent\ function)$$

$$classify\_function = (trig, elementary)$$

$$definition = \left[\tan(z) = -\frac{I\left(\left(e^{Iz}\right)^2 - 1\right)}{\left(e^{Iz}\right)^2 + 1},\ with\ no\ restrictions\ on\ (z)\right]$$

$$singularities = \left[\tan(z),\ z = \infty + \infty\, I,\ \left(\frac{z}{\pi} - \frac{1}{2}\right)::integer\right]$$

$$branch\_points = [\tan(z),\ No\ branch\ points]$$

$$branch\_cuts = [\tan(z),\ No\ branch\ cuts]$$

$$special\_values = \left[\tan\left(\frac{1}{6}\pi\right) = \frac{1}{3}\sqrt{3},\ \tan\left(\frac{1}{4}\pi\right) = 1,\ \tan\left(\frac{1}{3}\pi\right) = \sqrt{3},\ \tan(\infty) = undefined,\ \tan(\infty\, I) = I,\ [\tan(\pi n) = 0, \mathrm{And}(n::integer)],\ \left[\tan\left(\frac{1}{2}(2n+1)\pi\right) = \infty + \infty\, I, \mathrm{And}(n::integer)\right]\right]$$

$$identities = \left[\tan(z) = -\tan(-z),\ \tan(z) = \frac{2\tan\left(\frac{1}{2}z\right)}{1 - \tan\left(\frac{1}{2}z\right)^2},\ \tan(z) = \frac{\sin(z)}{\cos(z)},\ \tan(z) = \frac{1}{\cot(z)},\ \tan(z) = \frac{2\cot\left(\frac{1}{2}z\right)}{\cot\left(\frac{1}{2}z\right)^2 - 1},\ \tan(z)\right.$$

Függvények III.

Egyváltozós függvényeket a Maple-ben legegyszerűbben a matematikában is használatos -> nyíl operátorral adhatunk meg. A szintaxis:
függvénynév := **változónév** -> **kifejezés**.
A változónevet kerek zárójelek közé is tehetjük, de ez nem kötelező. A kifejezés – ami általában tartalmazza a változónevet is – a függvény hozzárendelési szabályát definiálja.
A függvények speciális Maple **eljárások** (`procedure`), erre még visszatérünk a 14. Előadásban.

```
restart;
f := x -> 1/2*sin(x) + 1;
```

$$f := x \to \frac{1}{2}\sin(x) + 1$$

```
whattype(f), type(f,procedure);
```

$$symbol, true$$

Függvények IV.

Számoljuk ki különböző helyeken az $f(x)$ függvény helyettesítési értékét. Mivel a Maple szimbolikus számítási rendszer, függvény-argumentumként megadhatunk szimbolikus konstansokat és kifejezéseket is. A szemléletesség kedvéért az egyenlőségek bal oldalán föltüntettük a kiértékeletlen függvényhívást.

```
'f(1)'    = f(1);
'f(Pi/3)' = f(Pi/3);
'f(0.5)'  = f(0.5);
'f(z-2)'  = f(z-2);
```

$$f(1) = 1/2\ \sin(1) + 1$$
$$f\left(\frac{\pi}{3}\right) = 1/4\ \sqrt{3} + 1$$
$$f(0.5) = 1.239712769$$
$$f(z-2) = 1/2\ \sin(z-2) + 1$$

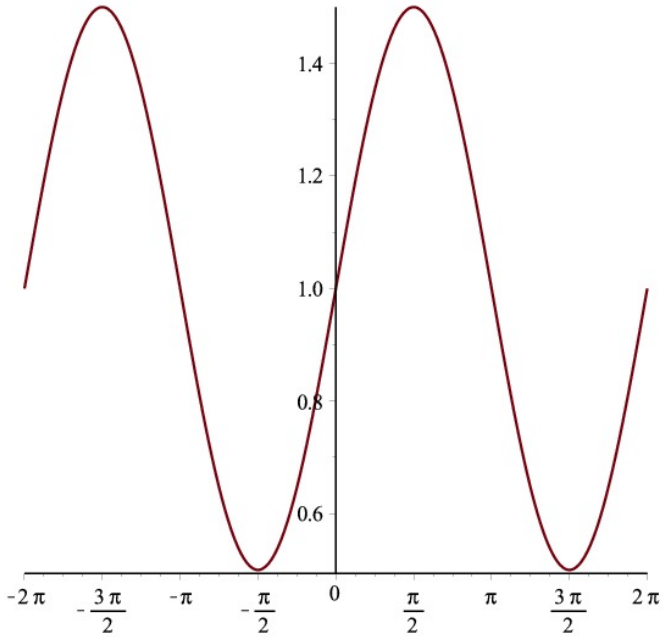
Függvények V.

Ábrázoljuk függvényünket! Ezt legegyszerűbben a `plot` eljárással végezhetjük. Hívása:

`plot(`**függvénynév, tartomány**`)`

ahol a tartomány kifejezés a változó ábrázolási tartományát adja meg. A `plot` használatára a 2D grafikáról szóló fejezetben még visszatérünk.

```
plot(f, -2*Pi..2*Pi);
```



Függvények VI.

Konstansokat és konstans függvényeket is definiálhatunk a `->` operátorral. Az `fcons` konstanst megadhatjuk nulla változós függvényként.
A Maple szintaktikus szabályaiból következik, hogy az `fcons` függvénynek adhatunk egy vagy több argumentumot is, az eredmény mindig 2 lesz.

```
fcons := () ->2;
fcons(),fcons(3),fcons(a,b);
```

$$fcons := () \to 2$$
$$2, 2, 2$$

A `gcons` egyváltozós konstans függvény bármely argumentumra a 2 értéket adja.

```
gcons := (x) ->2;
gcons(), gcons(3), gcons(a,b);
```

$$gcons := (x) \to 2$$
$$2, 2, 2$$

```
h := x -> sin(x);
h(fcons()), fcons(h), fcons(h(a),b),  fcons(h(a,b));
h(gcons()), gcons(h), gcons(h(a),b), gcons(h(a,b));
```

$$h := x \to \sin(x)$$
$$\sin(2), 2, 2, 2$$
$$\sin(2), 2, 2, 2$$

Nem kötelező függvényeinknek nevet adni. Használhatunk a felhasználás helyén „röptében” definiált függvényeket is:

```
( x -> x^2+1 )(2), ( x -> x/(x+sin(x)) )(a);
```

$$5, \frac{a}{a + \sin(a)}$$

Függvények VII.

A többváltozós függvényeket szintén a -> nyíl operátorral definiálhatjuk. A szintaxis: **függvénynév** := (**változónévlista**) -> **kifejezés**. A kifejezés – ami általában tartalmazza a változóneveket is – a függvény hozzárendelési szabályát definiálja. Most kötelező a változókat zárójelbe tenni.
Adjunk meg egy 2 változós g(x,y) függvényt. Számítsuk ki helyettesítési értékét különböző helyeken.

```
g := (x,y) -> abs(sin(x)*cos(y));
```

$$g := (x, y) \to \left|\sin(x)\cos(y)\right|$$

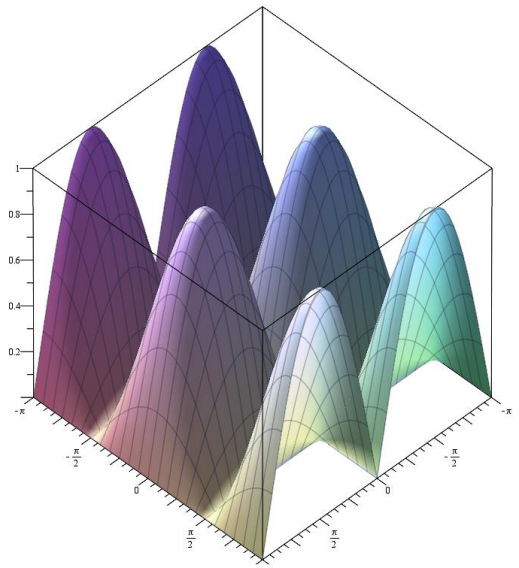
```
'g(0,0)'       = g(0,0);
'g(Pi/3,Pi/3)' = g(Pi/3,Pi/3);
'g(0.2,0.1)'   = g(0.2,0.1);
'g(2*a-1,s)'   = g(2*a-1,s);
```

$$g(0,0) = 0$$
$$g\left(\frac{\pi}{3}, \frac{\pi}{3}\right) = 1/4\ \sqrt{3}$$
$$g(0.2, 0.1) = 0.1976768117$$
$$g(2a-1, s) = \left|\sin\left(2\,a-1\right)\cos\left(s\right)\right|$$

Függvények VIII.

Kétváltozós függvényeket legegyszerűbben a `plot3d` függvénnyel ábrázolhatunk. A legegyszerűbb hívása:
`plot3d(`**`függvénynév, tartomány1, tartomány2`**`)`
ahol a tartomány1 és tartomány2 `range` kifejezések az első és a második változó ábrázolási tartományát adják meg.
A `plot3d` használatára a 3D grafikáról szóló fejezetben még visszatérünk.

```
plot3d(g, -Pi..Pi, -Pi..Pi);
```



Függvények IX.

A Maple ismeri a függvénykalkulus műveleteit, az összeg-, különbség-, szorzat- és hányadosfüggvénnyel, valamint az összetett függvénnyel való számolás – a függvénykompozíció – szabályait.
A 13. Előadásban látni fogjuk, hogy derivált- és primitív függvényeket is meg tud határozni.

```
f:= x ->x^2 + sin(x); g := x -> ln(x+1);
```

$$f := x \to x^2 + \sin(x)$$
$$g := x \to \ln(x+1)$$

```
h := f + g; 'h(2)' = h(2);
```

$$h := f + g$$
$$h(2) = 4 + \sin(2) + \ln(3)$$

```
s := f - g; 's(2)' = s(2);
```

$$s := f - g$$
$$s(2) = 4 + \sin(2) - \ln(3)$$

```
t := f * g; 't(2)' = t(2);
```

$$t := fg$$
$$t(2) = (4 + \sin(2)) * \ln(3)$$

```
u := f/g; 'u(2)' = u(2);
```

$$u := f/g$$
$$u(2) = \frac{4+\sin(2)}{\ln(3)}$$

Függvények X.

Az $f(g(x))$ függvénykompozícióra használhatjuk az `(f@g)(x)` jelölést. Az $f(f(...f(x))...)$ n-szeres kompozíció `(f@@n)(x)` alakban is írható.

```
v := f@g;   'v(2)' = v(2);
u := g@f;   'u(2)' = u(2);
w := g@@4;  'w(2)' = w(2);
```

$$v := f@g$$
$$v(2) = (\ln(3))^2 + \sin(\ln(3))$$
$$u := g@f$$
$$u(2) = \ln(5 + \sin(2))$$
$$w := g^{(4)}$$
$$w(2) = \ln(\ln(\ln(\ln(3) + 1) + 1) + 1)$$

Vigyázat! `f@g(x) <> (f@g)(x)`, mivel a `@` művelet prioritása kisebb, mint a függvényalkalmazásé.

```
f@g(x) <> (f@g)(x);
```

$$f@ln(x+1) \neq ln(x+1)^2 + sin(ln(x+1))$$

Függvények XI.

A többváltozós analízisben sokszor elegánsabb (sőt néha elengedhetetlen) a függvények változóinak neveire az $x, y, z, \ldots$ helyett $x_1, x_2, x_3, \ldots$ alakú indexelt változóneveket használni. Ezt a jelölést azonban nem támogatja a Maple. További információ hiányában az `x[1]` kifejezés a Maple szerint egy `x` nevű tábla első elemére való hivatkozás, s mint ilyen, nem állhat függvény definícióban a `->` nyil bal oldalán.

```
restart;
f := (x[1], x[2]) -> x[1]*sin(x[2]);
Error, invalid parameter; functional operators require
 their parameters to be of type symbol or (symbol::type)
```

Függvények XII.

A megoldás: literál indexek (`literal subscripts`) használata. Ez a lehetőség hiányzik a régebbi Maple változatokból.
Ha ilyenekkel indexezünk egy változónevet, az eredmény típusa szimbólum (`symbol`) és nem indexelt név.
Az 1D jelölésben a literál indexeket a __ dupla aláhúzás karakter jelöli. A 2D matematikai outputban az így indexelt változók lila színben jelennek meg, így lehet a közönséges indexelt változóktól megkülönböztetni őket.

```
x,     whattype(x);
x[1],  whattype(x[1]);
x__1,  whattype(x__1);
```

$$x, symbol$$
$$x_1, indexed$$
$$x_1, symbol$$

Függvények XIII.

Az f függvény két literál-indexszel megadott változója, x_1 és x_2 pontosan úgy viselkedik, mintha helyettük például az x és y szimbólumokat használtuk volna. A különbséget a munkalapokon az eltérő (lila) színű megjelenítés mutatja.

```
restart;
f := (x__1, x__2) -> x__1*sin(x__2);
```

$$f := (x_1, x_2) \to x_1 \sin(x_2)$$

```
'f(1,Pi/4)'=f(1,Pi/4),'f(a*s+b,k)'=f(a*s+b,k);
```

$$f(1, \pi/4) = 1/2\ \sqrt{2},\ f(as+b, k) = (as+b)\sin(k)$$

```
x__1, x__2 := exp(1), Pi/3;
f(x__1, x__2);
```

$$x_1, x_2 := e, \frac{\pi}{3}$$

$$\frac{1}{2} e \sqrt{3}$$

Függvények XIV.

A matematikában gyakran találkozunk olyan függvényekkel, amelyek az értelmezési tartományuk egyes részein más- és más képletekkel adhatók meg. Ilyen például az $\mathrm{abs}(x)$ vagy a $\mathrm{signum}(x)$ függvény.
Hasonló – szakaszonként folytonos – függvények előállítására a Maple-ben kényelmes megoldás a `piecewise` eljárás használata. Szintaxisa:
`piecewise(`**felt11 , kif1 , felt2 , kif2 , ... , feltn , kifn , kifdefault** `)`
Vigyázat! A visszaadott érték *nem* **függvény**, hanem a szakaszonként folytonos függvényt definiáló speciális Maple **kifejezés**, melyet adott helyen az `eval` eljárás segítségével értékelhetjük ki.

```
fkif:= piecewise(x<-1, x^2, x<=1, 1, x^3);
```

$$fkif := \begin{cases} x^2 & x < -1 \\ 1 & x \le 1 \\ x^3 & otherwise \end{cases}$$

```
eval(fkif,x=-2),eval(fkif,x=-1),eval(fkif,x=1),eval(fkif,x=2);
```

$$4, 1, 1, 8$$

Függvények XV.

Ha valódi **függvényt** akarunk definiálni, használjuk a `->` operátort.

```
f := x -> piecewise(x<-1, x^2, x<=1, 1, x^3);
'f(-2)'=f(-2), 'f(-1)'=f(-1), 'f(sqrt(2))'=f(sqrt(2)), 'f(2)'=
```

$$f := x \to piecewise(x < -1, x^2, x \le 1, 1, x^3)$$
$$f(-2) = 4, f(-1) = 1, f\left(\sqrt{2}\right) = 2\sqrt{2}, f(2) = 8$$

Ha f argumentumában változók (paraméterek) szerepelnek, a `piecewise` nem tudja eldönteni, hogy melyik feltételek teljesülnek.

```
'f(a/b)' = f(a/b);
```

$$f\left(\frac{a}{b}\right) = \begin{cases} \frac{a^2}{b^2} & \frac{a}{b} < -1 \\ 1 & \frac{a}{b} \le 1 \\ \frac{a^3}{b^3} & otherwise \end{cases}$$

Az apply eljárás

Az `apply` eljárás az első argumentumában megadott függvényt (eljárást) hívja a további paraméterekkel (ha vannak). Használhatjuk általunk definiált függvényekkel is. Lásd `?apply`.

```
restart;
apply(f);
apply(sqrt, 4);
apply(max,2,13,4);
myfun1 := (x,y) -> x^2 + y^2 +1;
apply(myfun1,2,3);
```

$$f()$$
$$2$$
$$13$$
$$myfun1 := (x,y) -> x^2 + y^2 + 1$$
$$14$$

Az unapply eljárás

Az `unapply` eljárás **kifejezésből függvényt** (eljárást) csinál. Első paramétere a függvényt definiáló kifejezés. Ezután a létrehozandó függvény változóit soroljuk föl. (A változókat listában is megadhatjuk.)
Teljesül az
`apply(unapply(f(x,y,...),x,y,...),a1,a2...)=f(a1,a2,...)`
azonosság. Lásd `?unapply`.

```
myfun2 := unapply(a*x^2 + b*y^2 +1,x,y);
myfun3 := unapply(a*x^2 + b*y^2 +1,[x,y]);
```

$$myfun2 := (x,y) -> ax^2 + by^2 + 1$$
$$myfun3 := (x,y) -> ax^2 + by^2 + 1$$

```
myfun2(2,3);
apply(myfun3,2,3);
apply(unapply(myfun2(x,y),x,y),2,3 );
```

$$4a + 9b + 1$$
$$4a + 9b + 1$$
$$4a + 9b + 1$$

A map eljárás I.

A `map` eljárás az első argumentumában megadott függvényt alkalmazza a második argumentumaként megadott kifejezés minden komponensére (operandusára). Az eredeti kifejezés változatlan marad.
A függvény az esetleges további argumentumokat egy-az-egyben továbbadja az alkalmazott függvénynek.
A `map2` eljárás is hasonló, de ott a második argumentum beíródik a megadott függvény minden hívásánál első argumentumként.

```
restart;
L := [1,2,3,4,5,6];
map(isprime, L);
L;
```

$$L := [1, 2, 3, 4, 5, 6]$$
$$[false, true, true, false, true, false]$$
$$[1, 2, 3, 4, 5, 6]$$

A map eljárás II.

A `map2` eljárás is hasonló, de ott a második argumentum beíródik a megadott függvény minden hívásánál első argumentumként.

```
map(f, {a,b,c});
map(x -> 5*x^2, a+ sin(b-c));
map(diff, [sin(x), cos(x), exp(x)], x);
map2(`+`,2, [sin(x), cos(x), exp(x)]);
```

$$\{f(a),f(b),f(c)\}$$
$$5a^2 + 5\sin(b-c)^2$$
$$[\cos(x), -\sin(x), e^x]$$
$$[2+\sin(x), 2+\cos(x), 2+e^x]$$

6. Grafikai lehetőségek

Maple grafika

A Maple segítségével látványos két- és háromdimenziós vizualizációkat készíthetünk. Ebben a fejezetben a következő három területet tekintjük át:

- síkbeli ábrázolások (2D grafika);

Maple grafika

A Maple segítségével látványos két- és háromdimenziós vizualizációkat készíthetünk. Ebben a fejezetben a következő három területet tekintjük át:

- síkbeli ábrázolások (2D grafika);
- térbeli ábrázolások (3D grafika);

Maple grafika

A Maple segítségével látványos két- és háromdimenziós vizualizációkat készíthetünk. Ebben a fejezetben a következő három területet tekintjük át:

- síkbeli ábrázolások (2D grafika);
- térbeli ábrázolások (3D grafika);
- animációk.

Bevezetés

A Maple 2D ábrák készítéséhez különböző lehetőségeket kínál. Ábrázolni tud például

- grafikus primitíveket (pontsorozatok, egyenes szakaszok, sokszögek, körök, stb.)
- többféle módon formázott és pozicionált szövegeket
- különböző formában definiált (explicit, implicit, paraméteres) függvény-grafikonokat.

Sőt többféle képtranszformációra, például elforgatásokra és eltolásokra is képes. A föntiek nagy része végrehajtható a korábban is használt `plot` eljárással. A Help részletes katalógust tartalmaz a használható grafikonfajtákról, lásd `?worksheet,reference,PlottingGuide`. A speciálisabb megoldásokhoz a `plots` és a `plottools` csomag eljárásai szükségesek, ezért a továbbiakban föltesszük, hogy ezeket betöltöttük:

```
with(plots): with(plottools):
```

2D grafika I.

A munkalapos felületen az ábrák alapértelmezés szerint a munkalapba beágyazva jelennek meg. Ehelyett azonban eleve fájlokba menthetők az összes fontosabb grafikus formátumban. A szükséges beállításokat megadhatjuk a `plotsetup` eljárással. Általános szintaxisa:
`plotsetup(`**grafikusmeghajtó**`,plotoutput=`**fájlnév**`,plotoptions=`**opciók**`)`

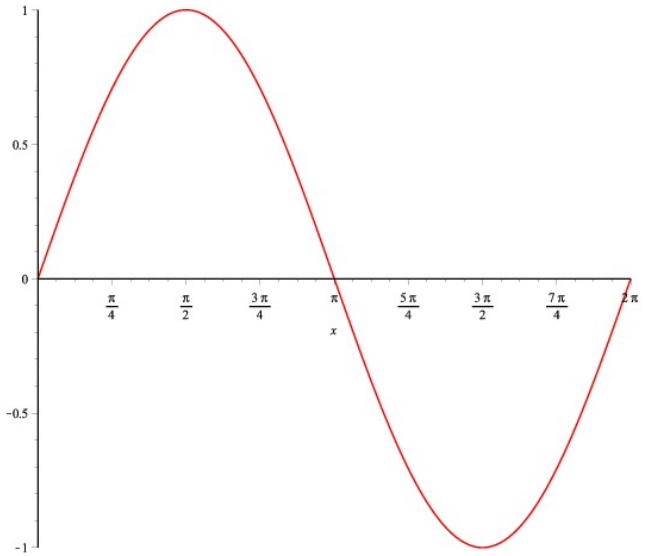
- A **grafikusmeghajtó** gyakori értékei: `char` (karakteres), `default` (alapértelmezett), `inline` (munkalapba ágyazott), `jpeg` (JPEG kép), `gif` (GIF kép), `ps` (PostScript), `cps` (színes PostScript).
- A **fájlnév** nevű (sztringként vagy változóként megadott) fájlba menti a Maple a képet.
- Az **opciók** az ábrákhoz tartozó további beállításokat adhat meg, így például keretezést (`noborder`), méretet (`width=4cm,height=4cm`), színmodellt (`color=rgb`), stb.

A további részletekre vonatkozóan lásd `?plotsetup`.
A `plotsetup` hívása után az ábrákat generáló eljárások a megadott paraméterek szerinti outputot generálnak – amíg ezeket meg nem változtatjuk.

2D grafika II.

Figyeljük meg az alábbi példákban az eltérő grafikus meghajtók és opciók hatását. A generált fájlokat a `/home/viragh/Maple` alkönyvtár tartalmazza.

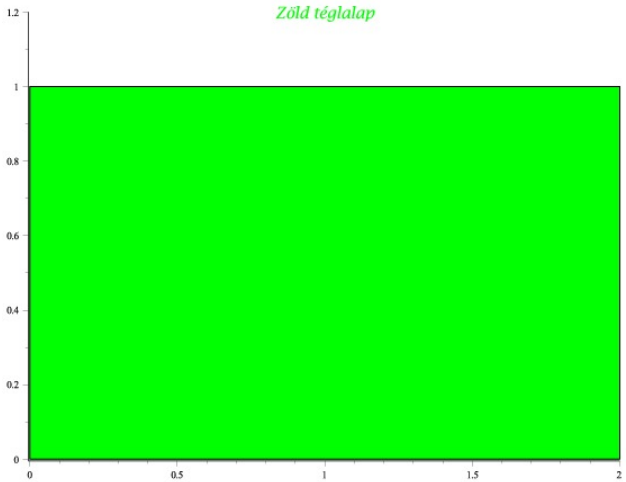
```
plotsetup(ps,plotoutput="/home/viragh/Maple/abra1.eps",
          plotoptions=",noborder,width=4cm,height=4cm,color=gray");
plot(sin(x), x=0..2*Pi, color=red);
plotsetup(ps,plotoutput="/home/viragh/Maple/abra2.eps",
          plotoptions="noborder,width=8cm,height=8cm,color=rgb");
plot(sin(x), x=0..2*Pi, color=red);
plotsetup(jpeg,plotoutput="/home/viragh/Maple/abra3.jpg");
plot(sin(x), x=0..2*Pi, color=red);
plotsetup(default);
plot(sin(x), x=0..2*Pi, color=red);
```



2D grafika III.

A `plotsetup` leírásánál felsorolt lehetőségek azon alapulnak, hogy a Maple nem közvetlenül rajzol a képernyőre, hanem először speciális `PLOT` és `PLOT3D` belső adatstruktúrákat hoz létre.
Az éppen használatban lévő grafikus meghajtó ezeket értelmezve generál képernyőképet, vagy speciális formátumú (JPEG, GIF, EPS) képfájlokat. A `PLOT` struktúrák leírása megtalálható a `?plot,structure` Help oldalon.
A leírás alapján mi is készíthetünk és megjeleníthetünk ilyen struktúrákat, de ehelyett inkább a továbbiakban ismertetett felhasználói szintű eljárások és csomagok használata ajánlott.
Amikor változóban elmentjük a `plot` hívás eredményét, valójában ilyen struktúrákat tárol a Maple. Egy egyszerű példa:

```
P1 := PLOT(
      POLYGONS([[0,0],[2,0],[2,1],[0,1]], COLOR(RGB,0,1,0)),
      TEXT([1.0,1.2], "`Zöld téglalap`", COLOR(RGB,0,1,0),FONT(Times,16)),
      AXESSTYLE(FRAME)):

P1;
```



2D grafika IV.

A `plot` ezernyi paramétere és opciója közül a legfontosabb az első kettő: *mit* és *hol* akarunk ábrázolni. Harmadik paraméterként megadhatjuk a függő változó ábrázolási tartományát is, bár ez nem kötelező.
Ha **függvényt** ábrázolunk, a
`plot(`**függvénynév, balvégpont..jobbvégpont, plotopciók**`)`
alakot használjuk, például

```
f := x -> x - sin(x):
plot(f, 0..2*Pi); # vagy: plot(f, 0..2*Pi, y=0..6.3);
```

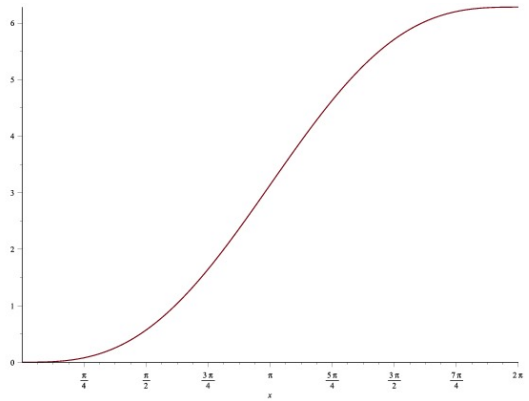
Ha kifejezéssel adjuk meg az ábrázolandó függvényt, a
`plot(`**kifejezés, x=balvégpont..jobbvégpont, plotopciók**`)`
alak a helyes:

```
plot(x - sin(x), x=0..2*Pi); #vagy:  plot(x - sin(x), x=0..2*Pi, y=0..6.3);
```

Mivel az `f(x)` függvényhívás már kifejezésnek számít, ilyenkor az előző alakot kell vennünk.

```
plot(f(x), x=0..2*Pi);  #vagy: plot(f(x), x=0..2*Pi, y=0..6.3);
```

Mindegyik plot hívás az alábbi grafikont hozza létre:



2D grafika V.

A legfontosabb `plot` opciók az alábbi táblázatban láthatók. A `?plot,options` Help oldal tartalmazza a teljes leírást.

Opciónév	**Jelentés**	**Típus**	**Példa**
legend	jelmagyarázat	sztring	"Az f(x) grafikonja"
caption	ábra aláírás	sztring	"A két függvény képe"
title	ábra cím	sztring	"Függvényvizsgálat"
color	grafikonszíne	színnév	red
thickness	vonalvastagság	pozitív egész	3
font	betűtípus	3 elemű lista	[Times, bold, 16]
axes	tengelyek	boxed, frame, none, vagy normal	boxed
discont	szakadások kezelése	logikai	true
linestyle	vonalstílus	solid, dot, dash, stb.	dot
style	az ábra stílusa	line, point, polygon, polygonoutline	point
adaptive	adaptív mintapontválasztás	logikai	true
numpoints	mintapontok száma	pozitív egész	4000
scaling	a koordináták méretaránya	constrained, unconstrained	constrained

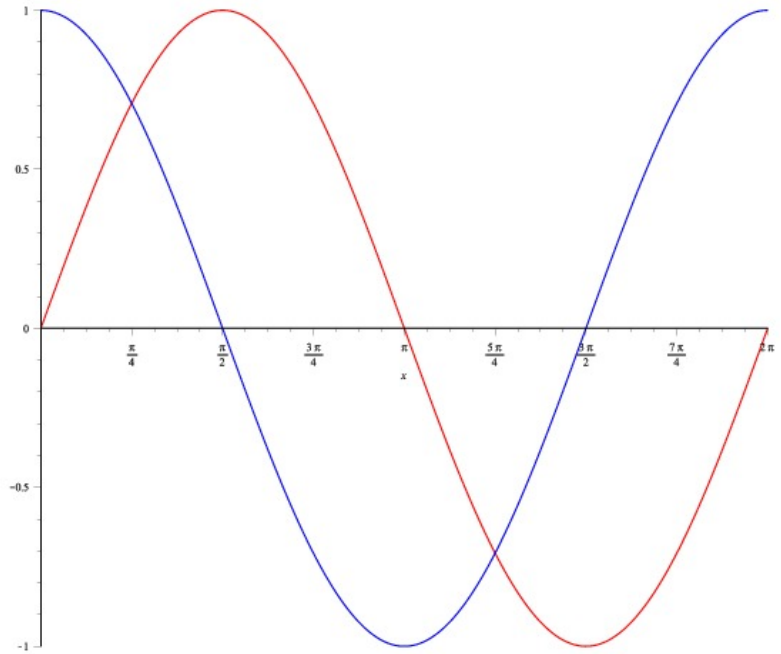
2D grafika VI.

A további példák bemutatásához szükségünk lesz arra, hogy több függvény grafikonját megjelenítsük

1. egyetlen közös ábrán, egyszerre;
2. egyetlen közös ábrán, de lépésenként, egymás után;
3. egymás mellé, táblázatosan elhelyezett ábrákon.

A `plot` első paramétereként az ábrázolandó függvények megadásánál használhatunk listákat vagy halmazokat is, ezzel megoldható az 1. feladat.

```
plot([sin(x),cos(x)], x= 0..2*Pi, color=[red,blue]);
```



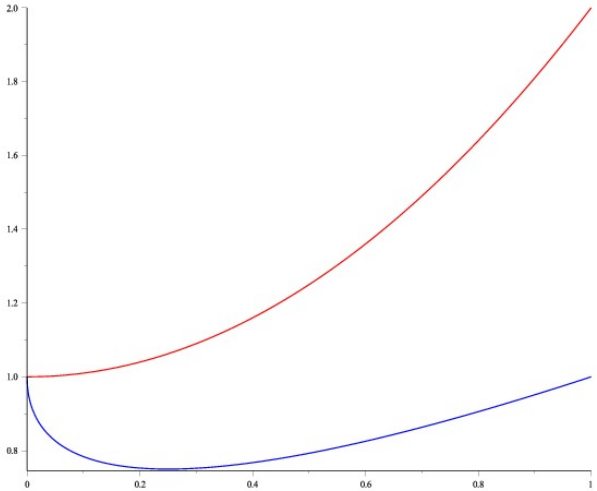
2D grafika VII.

A `display` eljárással még hatékonyabban oldható meg mindhárom feladat. Hívása
`display(L, insequence=boolean,` **plotopciók**`)`
vagy
`display(A,` **plotopciók**`)`
Az első változatnál az L listában lévő `PLOT` struktúrákat a `display` egyetlen közös ábrán jeleníti meg; az `insequence=false` alapértelmezett opcióval egyidejűleg, az `insequence=true` esetben pedig lépésenként, egymás után (erre még visszatérünk az animációknál).

```
f := x -> x^2 +1; g := x -> x-sqrt(x) +1;
```

$$f := x \to x^2 + 1$$
$$g := x \to x - \sqrt{x} + 1$$

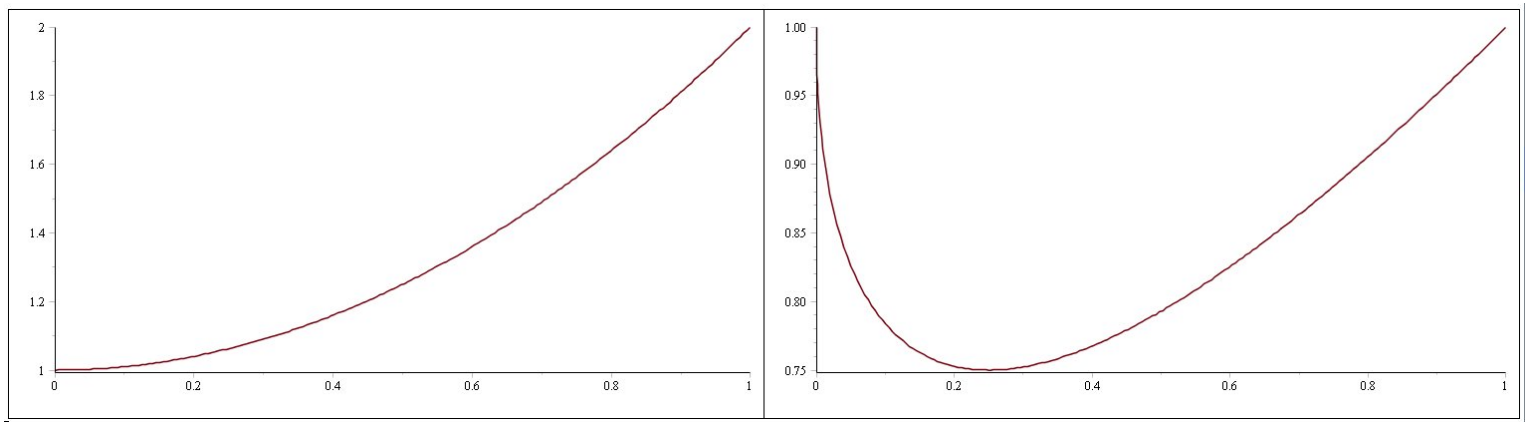
```
L1 := plot(f, 0..1,color=red):
L2 := plot(g, 0..1,color=blue):
display([L1,L2]);
```



2D grafika VIII.

A `display` a második változat hatására az A tömbben elhelyezett `PLOT` struktúrákat egymás mellett, táblázatos elrendezésben mutatja meg.

```
A := Array(1..2):
A[1] := plot(f, 0..1):
A[2] := plot(g, 0..1):
display(A);
```



2D grafika IX.

Az alábbi három `plot` parancsban a `legend`, `caption`, `title`, `color` és a `thickness` opciók megadását tanulmányozhatjuk.
Sztringekbe a `typeset` eljárással illeszthetünk be 2D formázott matematikai képleteket, erről lásd `?typeset`. Itt a jelmagyarázatok, képaláírások és a címek elkészítésénél használtuk föl.

```
P := Array(1..3): f := x -> x^2 +1;
```

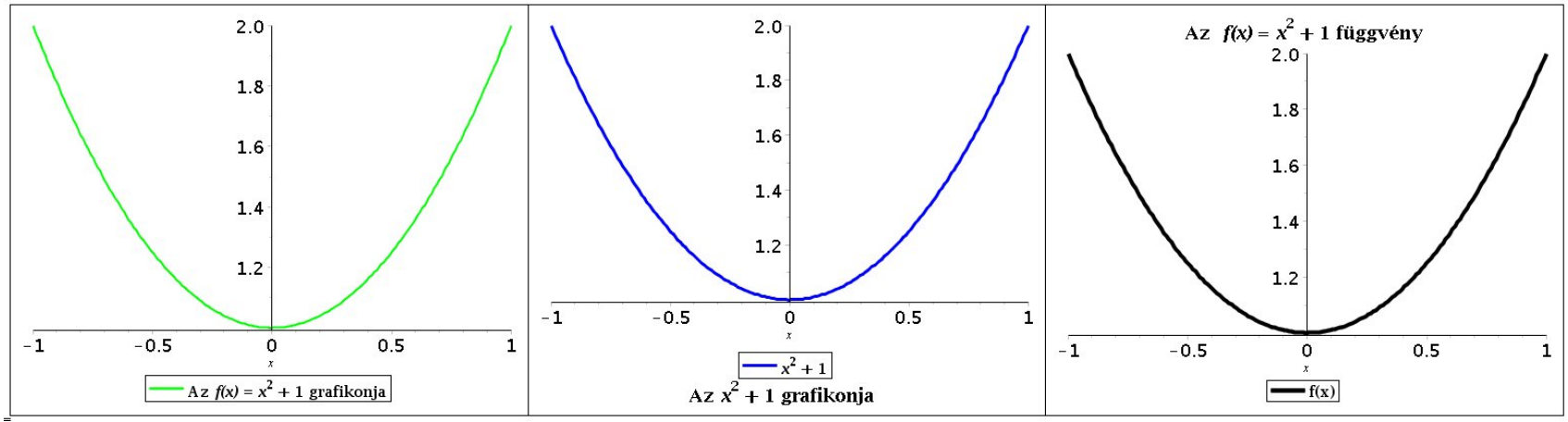
$$f := x \to x^2 + 1$$

```
P[1] := plot( f(x), x=-1..1,
              legend=typeset("Az %1 grafikonja",`f(x)`=f(x)),
              color=green,thickness=2):
P[2] := plot( f(x), x=-1..1,
              caption=typeset("Az %1 grafikonja",f(x)),
              legend=f(x),color=blue,thickness=3):
P[3] := plot( f(x), x=-1..1,
              title=typeset("Az  ",`f(x)`=f(x), "függvény"),
              legend = "f(x)",color=black,thickness=4):
```

2D grafika X.

Az előbb létrehozott és a P tömbben tárolt 3 grafikont jelenítjük meg a `display` eljárással.
A `display` opciójaként még itt is megadhatjuk az ábrákon használandó fontokat. Figyeljük meg, hogy az általános font mellett a tengelyekre, illetve a magyarázatra vonatkozó speciális font opciókat is alkalmaztunk.

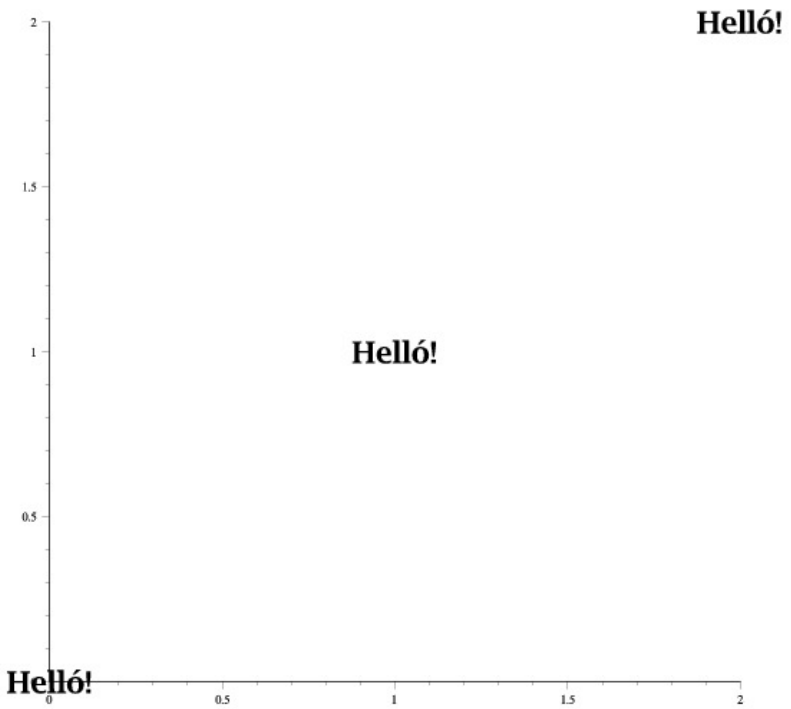
```
display(P,
        font=[Times, bold,14],
        axesfont=[Helvetica,roman,14],
        legendstyle=[font=[Times,bold, 12]]);
```



2D grafika XI.

Az ábrákon az eddig látott opciókon túl is elhelyezhetünk tetszőleges szövegeket a `textplot` eljárással, lásd `?textplot`. Hívása:
`textplot(L, plotopciók)`
ahol L vagy egy feliratot meghatározó [xkoordináta,ykoordináta, sztring] típusú 3 elemű lista, vagy – több szöveg esetén – ilyen 3 elemű listák listája.
Az eljárás úgy írja ki a megadott szövege(ke)t, hogy a szöveg közepe a megadott koordinátájú pontban legyen..

```
textplot([ [0,0,"Helló!"], [1,1,"Helló!"], [2,2,"Helló!"] ], font=[Times,bold,24]);
```



2D grafika XIIa.

Még egy példa a különböző 2D plot opciók használatára az alábbi három függvény közös grafikonjával.

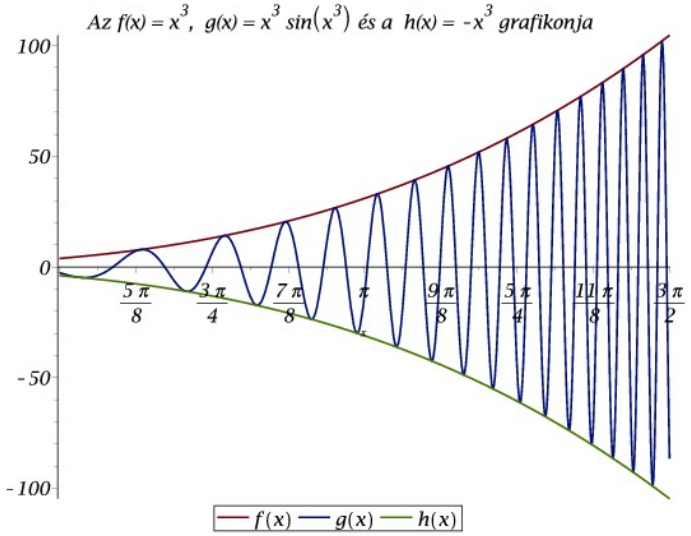
```
f := x -> x^3;
g := x -> x^3*sin(x^3);
h := x -> -x^3;
```

$$f := x \to x^3$$
$$g := x \to x^3 \sin\left(x^3\right)$$
$$h := x \to -x^3;$$

2D grafika XIIb.

Az első plot hívásnál megadtuk a `resolution` opcióval az eszköz (a képernyő) vízszintes felbontását, a `tickmarks` opcióval pedig az X tengelyen látható osztáspontok eloszlását. A cím, aláírás vagy magyarázat opciók helyett a `textplot` eljárással közvetlenül helyeztünk el egy `typeset`-tel formázott szöveget az ábrán – ennek pozicionálása „kézi erővel” néha sok próbálgatást igényel.

```
p1 := plot([f(x), g(x), h(x)], x=Pi/2..3/2*Pi,
          resolution=1600,tickmarks=[spacing(Pi/4), default]):
p2 := textplot([3.1, 100,
              typeset("Az %1, %2 és a %3 grafikonja",f(x),g(x),h(x))]):
display([p1,p2], font=[Times, bold, 13]);
```



2D grafika XIII.

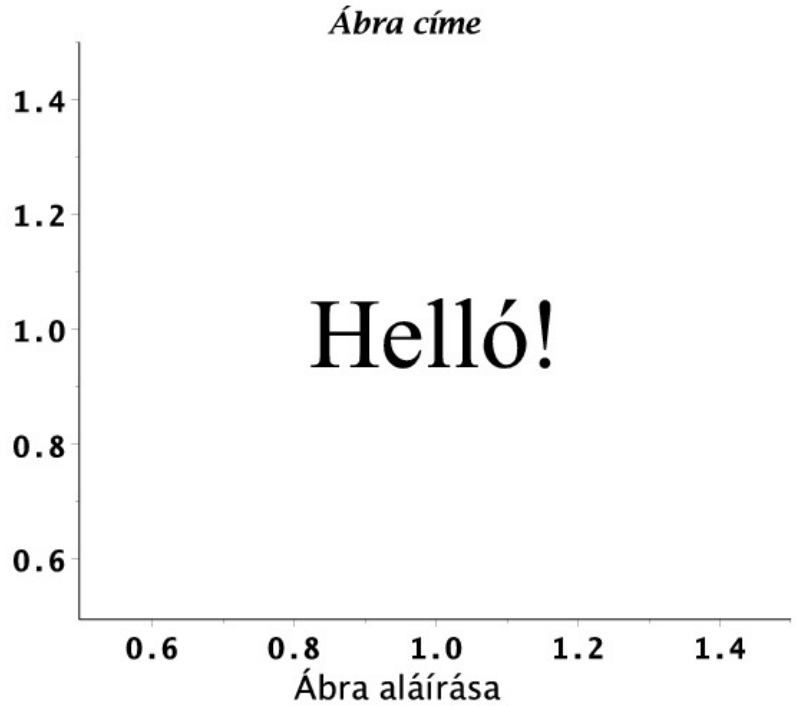
Néhány szó a Maple fontkezeléséről. Sok Maple megjelenítő eljárás, így a `plot` is, lehetővé teszi a használandó fontok specifikálását a `font = l` opcióval, ahol `l = [`**fontnév, változat, méret**`]` alakú három elemű lista.

- A fontnév valamely fontcsalád neve, például `Times`, `Courier`, `Helvetica`, vagy `Symbol`. Ezeket mindenképpen támogatja a Maple, de ha talál a gépen más, általa használható fontokat is (például `Tahoma`, `Lucida`, stb.), akkor ezeket is megadhatjuk. A fontnév mindig nagybetűvel kezdődik.
- A változat lehet `roman` (normál), `bold` (félkövér), `italic` (kurzív), `bolditalic` (fékövér kurzív), `oblique` (dőlt) vagy `boldoblique` (félkövér dőlt) – már ha az adott fontcsalád tartalmaz ilyen változatot (például a `Symbol` fontnak egyetlen változata van). Ez az elem el is hagyható.
- A méret pozitív egész szám, a betűk pontokban megadott (tervezési) mérete.

2D grafika XIV.

A `plot` esetében az általános `font=l` opción kívül külön-külön is beállítható az ábrák aláírásához (`captionfont=l`), címéhez (`titlefont=l`), a tengelyek feliratainak (`axesfont=`l) és az ábramagyarázatokhoz (`legendstyle=[font=l]`) rendelt font. Az alábbi példán a `textplot` hívásával demonstráljuk az előbbieket.

```
textplot([1,1,"Helló!"], font=[Lucida,roman, 60],
   captionfont=[Helvetica, 20], caption="Ábra aláírása",
   axesfont=[Courier, bold,18],
   titlefont=[Times,bolditalic, 18],  title = "Ábra címe");
```



2D grafika XV.

Néhány szó a Maple színkezeléséről. Ahol színt lehet megadni, mindenütt alkalmazhatók az alapszínek (angol) nevei, például `white`, `black`, `red`, `blue`, `green`, `yellow`, `brown`, `magenta`, stb.
A Maple 16-os változata óta hivatalosan sztringekként kell megadni a színeket, bár az előző színek esetében még többnyire elfogadja a rendszer az idézőjel nélküli írásmódot.
Használhatjuk a HTML szabványból ismert színeket is, ezek nagybetűs angol neve megtalálható a `?plot,colornames` Help oldalon. Ha ezeket választjuk, a `plot` opcióinál már kötelező az idézőjel használata.

```
plot(sin(x), color = Green);
Error, (in plot) color name must be a string
```

2D grafika XVI.

A `ColorTools` csomag a színkezeléssel kapcsolatos több tucat eljárást tartalmaz. A `GetColorNames()` például visszaadja a rendszer által ismert színek elnevezését. A `GetPalette(palettanév)` az adott paletta színeit listázza ki. A "HTML" paletta a HTML színeket tartalmazza.
A rendszer alapértelmezésben a grafikonokon a "Niagara" palettát használja.

```
with(ColorTools):
GetPalette("Niagara");
```

(Palette Niagara: Burgundy Navy LeafGreen Azure Purple BlueGreen Violet DeepBlue Cinnamon PaleRed GreenishBlue DarkRose BluishGreen DarkOrchid BluishPurple Olive)

2D grafika XVII.

A `Color` eljárással új színeket hozhatunk létre, vagy megnézhetünk egy létező színt. Outputja az illető szín RGB komponenseinek értéke az adott színű téglalapban. Figyeljük meg a nevek különböző írásmódját, valamint a `green` és a `Green` szín eltérő voltát.

```
zöld1 = Color(green), zöld2 = Color("green");
zöld3 = Color(Green), zöld4 = Color("Green");
```

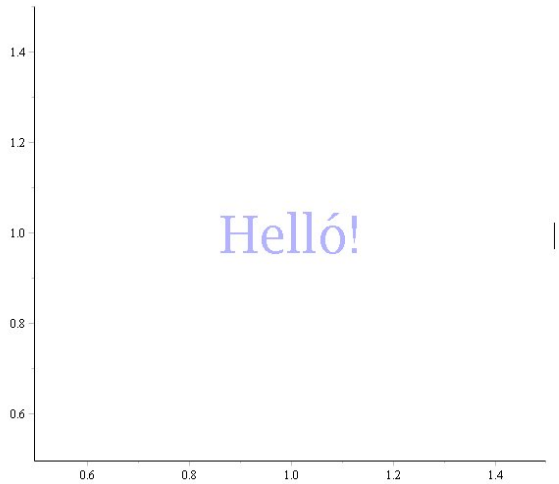
$$zöld1 = \langle \text{RGB } 0\ 1\ 0 \rangle, zöld2 = \langle \text{RGB } 0\ 1\ 0 \rangle$$
$$zöld3 = \langle \text{RGB } 0\ 0.502\ 0 \rangle, zöld4 = \langle \text{RGB } 0\ 0.502\ 0 \rangle$$

2D grafika XVIII.

Új szín megadásához a három RGB komponens erősségének 0 és 1 közti értékét kell listában megadni. Rögtön használhatjuk is az így létrehozott színt. A `Halványkék` Maple név (*nem* sztring), ezért nem kell az idézőjel.

```
Halványkék := Color([0.7, 0.7, 1]);
plots:-textplot([1,1,"Helló!"],
      font=[Times,roman, 40], color=Halványkék);
```

Halványkék := ⟨RGB 0.7 0.7 1⟩



A Maple belső `PLOT` struktúráiban használt színmegadás szintaxisa kicsit eltérő, ezt lehetőleg ne használjuk.

```
Halványkék2:=COLOR(RGB,0.7, 0.7, 1);
Color(Halványkék2);
```

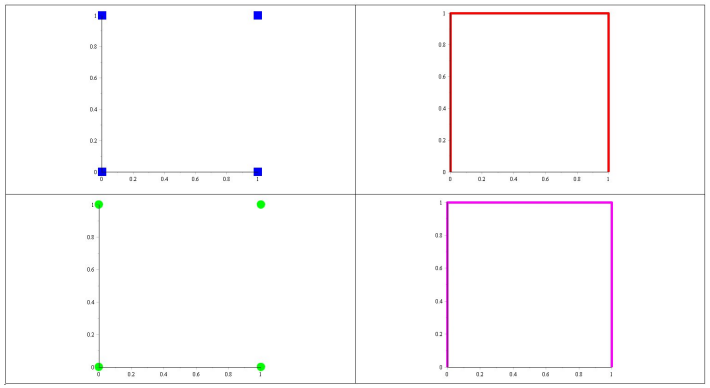
2D grafika XIX.

Egy L listában két koordinátájukkal megadott síkbeli pontokat ábrázolhatunk a plot segítségével, de léteznek erre speciális eljárások, például a `pointplot` és a `listplot`, lásd `?pointplot`, `?listplot`.
Az alábbiakban az egységnégyzetről készíttetünk 4 ábrát a `plot`, a `pointplot` és a `listplot` különböző opcióival.

```
L := [[0,0],[0,1],[1,1],[1,0]];
```

$$L := [[0,0],[0,1],[1,1],[1,0]]$$

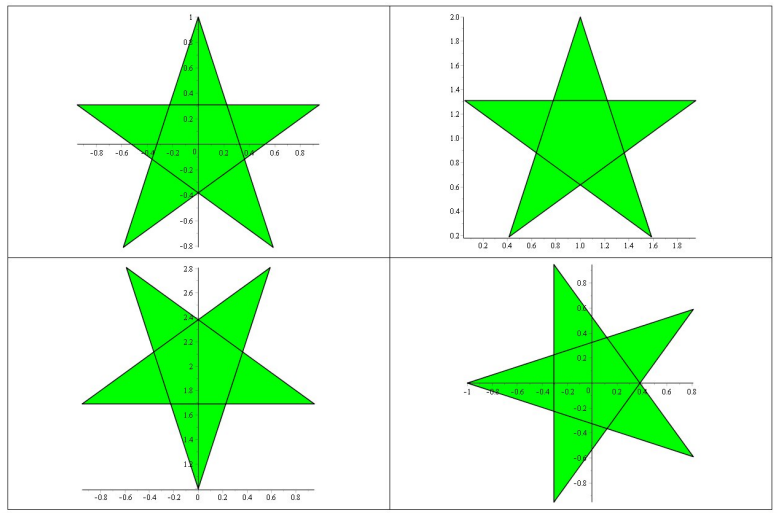
```
p := Array(1..2,1..2):
p[1,1] := plot(L, style=point, symbol=solidbox, color=blue, scaling=constrained):
p[1,2] := plot(L, style=line, color=red, scaling=constrained):
p[2,1] := pointplot(L,symbol=solidcircle, color=green, scaling=constrained):
p[2,2] := listplot(L, connect=true, color=magenta, scaling=constrained):
display( p, symbolsize=24,thickness=5 );
```



2D grafika XX.

Csúcspointjaik koordinátáival megadott sokszögeket ábrázolhatunk a `polygonplot` eljárással, lásd `?poligonplot`. A `plottools` csomag `translate`, `reflect` és `rotate` eljárásaival ábrákat (pontosabban 2D `PLOT` struktúrákat) eltolhatunk, tükrözhetünk vagy elforgathatunk. Ezt mutatjuk be az alábbi 4 ábrán. A csillag változóban tárolt `PLOT` struktúrát és ennek transzformáltjait P-ben tároljuk, majd a display eljárással megjelenítjük.

```
L_ötszög:=[seq( [sin(4*n*Pi/5), cos(4*n*Pi/5)], n=0..4)]:
csillag:=polygonplot(L_ötszög):
P := Array(1..2,1..2):
P[1,1]:=csillag:                              P[1,2]:=translate(csillag,1,1):
P[2,1]:=reflect(csillag,[[0,1],[1,1]]):  P[2,2]:=rotate(csillag,Pi/10):
display(P, color= green, scaling=constrained);
```



2D grafika XXI.

Ha az $r = f(\theta)$ polárkoordinátás egyenlettel adott síkgörbét akarjuk ábrázolni, használjuk a `polarplot` eljárást, lásd `?polarplot`. Szintaxisa:

`polarplot(f(theta), theta =` **kezdőérték..végérték, plotopciók**`)`

Készítsük el az alábbi görbék polárkoordinátás ábrázolását.

lemniszkáta	$r^2 = a^2\cos(2\theta)$	Fermat spirál	$r^2 = a^2\theta$
kör	$r = a$	hiperbolikus spirál	$r = \frac{a}{\theta}$

A grafikonokat a P tömbben generáljuk.

```
a := 'a':
f1:=theta -> sqrt(a^2*cos(2*theta));  f2:=theta -> a*sqrt(theta);
f3:=theta -> a;                       f4:=theta -> a/theta;
```

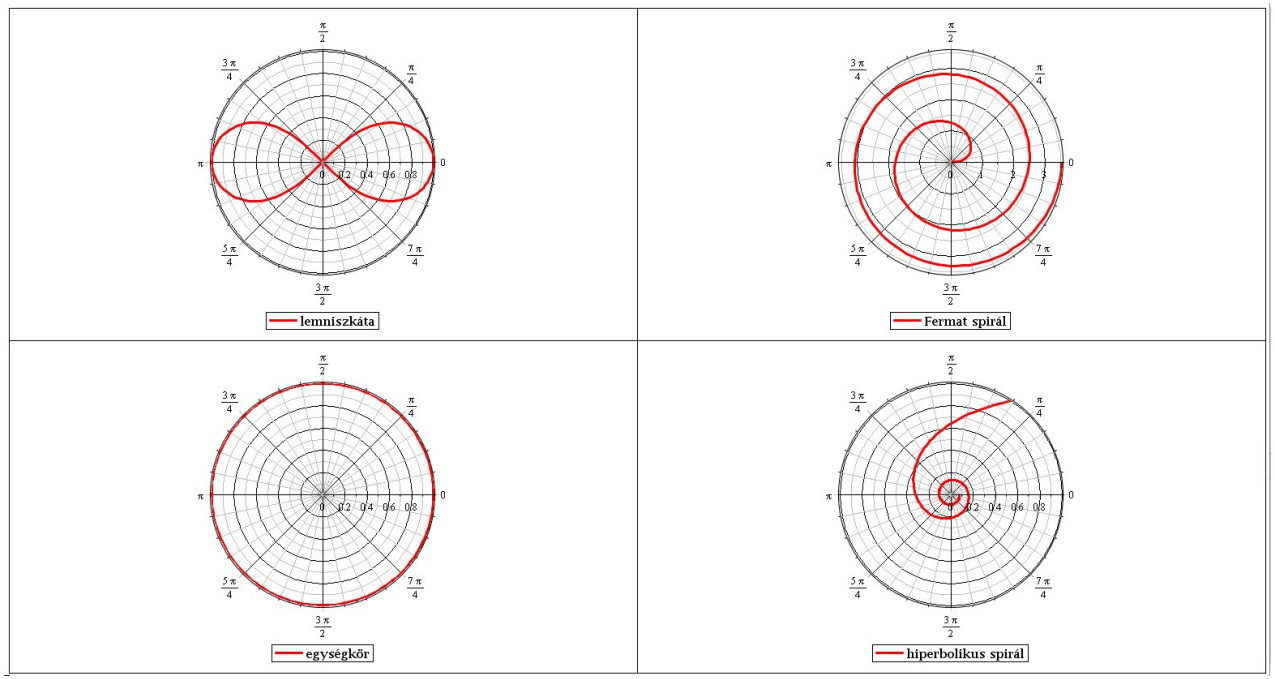
$$f1 := \theta \to \sqrt{a^2\cos(2\theta)}$$
$$f2 := \theta \to a\sqrt{\theta}$$
$$f3 := \theta \to a$$
$$f4 := \theta \to \frac{a}{\theta}$$

```
a := 1: P := Array(1..2,1..2):
P[1,1] := polarplot(f1(theta), theta=0..2*Pi,color = red,
                    legend="Lemniszkáta"):
P[1,2] := polarplot(f2(theta), theta=0..4*Pi,color = red,
                    legend="Fermat spirál"):
P[2,1] := polarplot(f3(theta), theta=0..2*Pi,color = red,
                    legend="Egységkör"):
P[2,2] := polarplot(f4(theta), theta=1..4*Pi,color = red,
                    legend="Hiperbolikus spirál"):
```

2D grafika XXII.

Az előző lapon elkészült ábra megjelenítése a display segítségével:

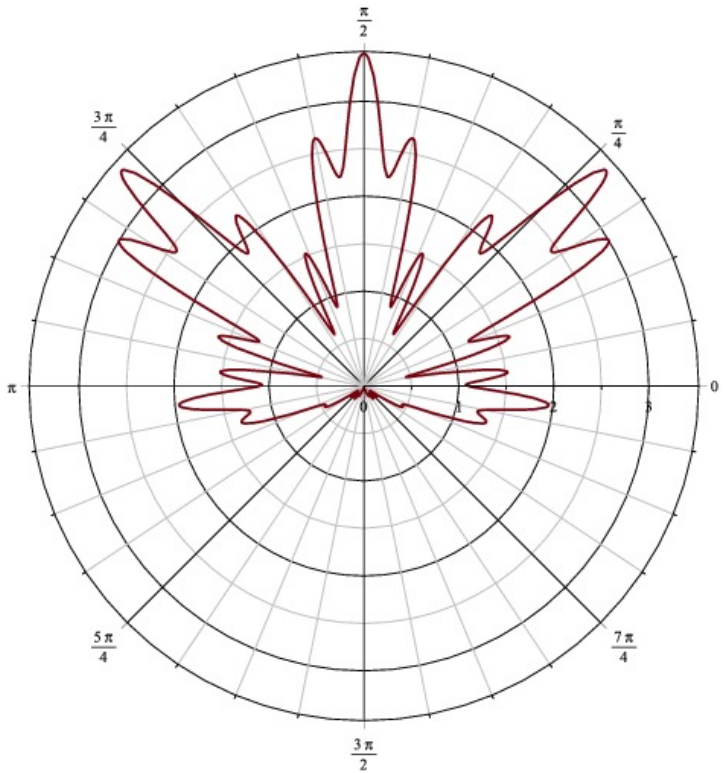
```
display(P, thickness=3, legendstyle=[font=[Times,bold, 12]]);
```



2D grafika XXIII.

Az alábbi függvény polárkoordinátás grafikonja juharlevelet formáz, így a program angol elnevezésére emlékeztet.

```
maple_leaf:=t->100/(100+(t-Pi/2)^8)*(2-sin(7*t)-cos(30*t)/2);
polarplot(maple_leaf(t), t=-Pi/2..3*Pi/2, thickness=2);
```



2D grafika XXIVa.

Ha az ábrázolandó síkgörbe $x = f(t), y = g(t)$ paraméteres alakban adott, akkor a plot szintaxisa:

`plot([f(t), g(t), t=`**tkezd..tveg**`], `**plotopciok**`)`

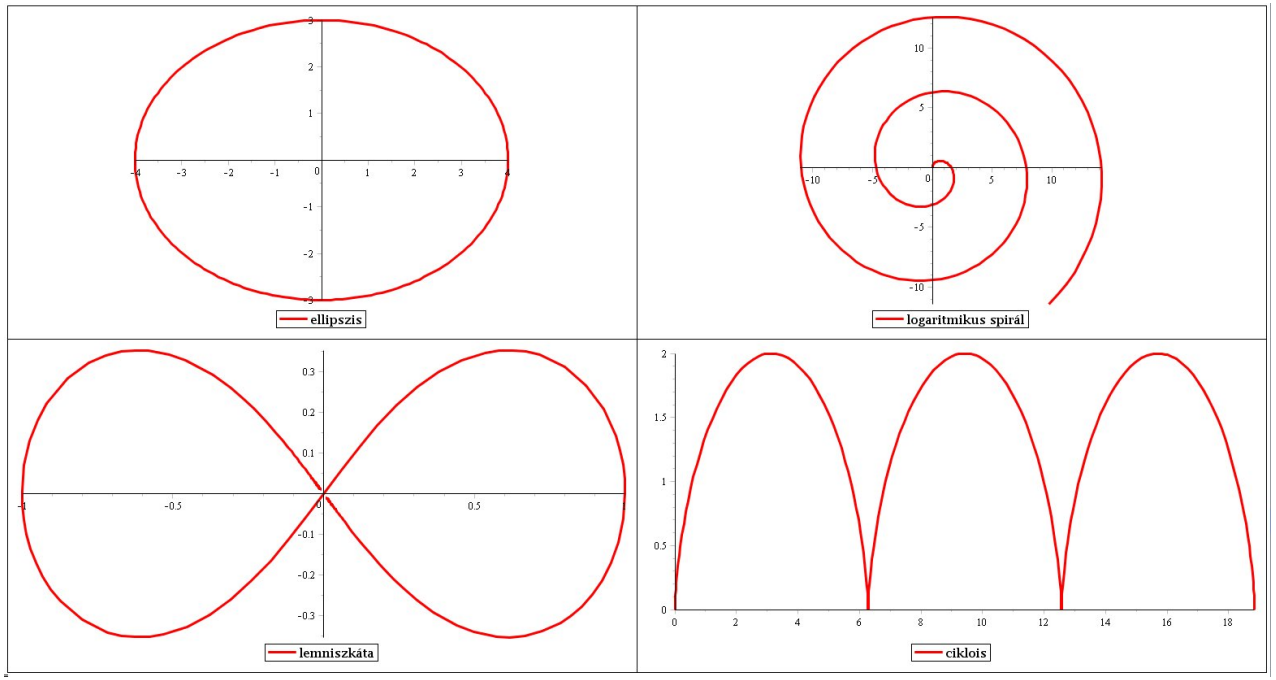
Az alábbi ábrák 4 síkgörbe, az ellipszis, a logaritmikus spirál, a lemniszkáta és a ciklois paraméteres előállítása alapján készültek

```
P := Array(1..2,1..2):
P[1,1]:=plot([4*sin(t),3*cos(t),t=0..2*Pi],
   legend="ellipszis",scaling=constrained,color=red):
P[1,2]:=plot([t*sin(t),t*cos(t),t=0..15],
   legend="logaritmikus spirál",scaling=constrained,color=red):
P[2,1]:=plot([t*(1+t^2)/(1+t^4),t*(1-t^2)/(1+t^4),t=-100..100],
   legend="lemniszkáta",color=red):
P[2,2]:=plot([(t-sin(t)),(1-cos(t)),t=0..6*Pi],
   legend="ciklois",color=red):
```

2D grafika XXIVb.

A P-ban tárolt ábrák megjelenítése:

```
display(P, thickness=3,legendstyle=[font=[Times,bold, 12]]);
```



2D grafika XXVa.

Az $f(x,y) = 0$ implicit egyenlettel definiált síkgörbéket az `implicitplot` eljárással ábrázolhatjuk. Opciói hasonlóak a `plot`-hoz, most az `y =` **kezdőérték..végérték** paraméter kötelező.

```
f := (x,y) -> x^2/4 + y^2 - 1;
implicitplot(f, -5..5);
```

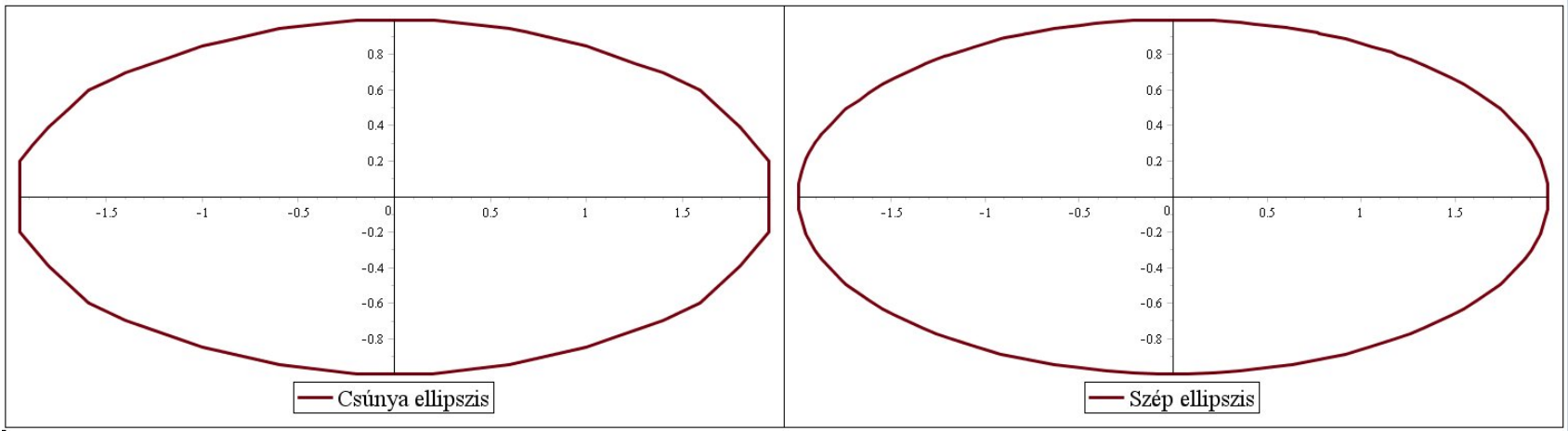
$$f := (x,y) \to \frac{x^2}{4} + y^2 - 1$$

```
Error, (in plots:-implicitplot) invalid input: `plots/implicitplot`
uses a 3rd argument, yin, which is missing
```

2D grafika XXVb.

Általában szükség van a `numpoints=`**mintapontszám** opcióra is, enélkül torz ábrát kaphatunk. Az $f(x,y)$ ellipszis bal oldali grafikonja még elég „darabos”.

```
P := Array(1..2):
P[1] := implicitplot(f, -5..5, -5..5,
    legend="Csúnya ellipszis"):
P[2] := implicitplot(f, -5..5, -5..5, numpoints=5000,
    legend="Szép ellipszis"):
display(P, thickness=3, legendstyle=[font=[bold, 18]]);
```



2D grafika XXVI.

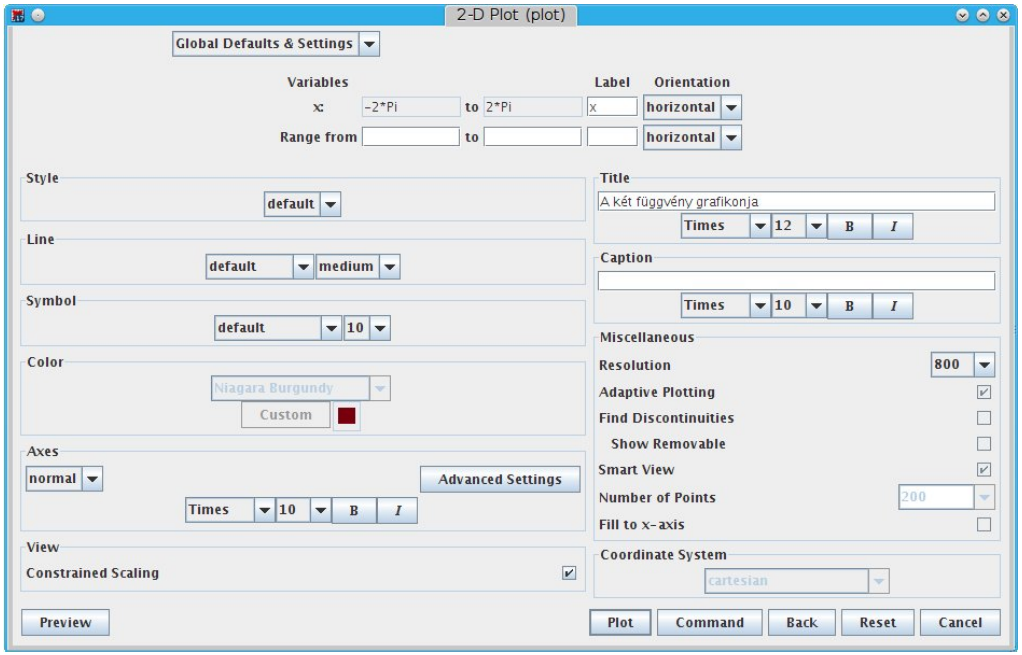
Ha nem ismerjük a plot ezernyi paraméterét és opcióját, több könnyített lehetőséget is használhatunk.
A `smartplot` eljárás paramétere(i) az ábrázolandó kifejezés(ek). Az eljárás maga határozza meg a többi szükséges paramétert.
Főleg olyan elnagyolt grafikák készítésére szánták, melyeket azután a rendszer interaktív grafikonkészítő segédletével finomítunk tovább (jobbklikk az ábrára, és a megfelelő menüpontok kiválasztása).
Létezik 3D változata is, a smartplot3D. Lehetséges hívása például

```
smartplot(sin(x),x*cos(x));
```

2D grafika XXVII.

Ha teljesen „0-ról akarunk indulni”, hívhatjuk a `plots` csomag `interactive` nevű grafikonkészítő eljárását, mellyel mapletes grafikus felületen lépésről-lépésre haladva készíthetjük el a kívánt grafikont.
Ugyanide vezet a **Tools/Assistants/Plot Builder** menüpont is. A lenti ábrán az interaktívan választható opciók egy része látható.

```
interactive(sin(x),x*cos(x));
```



Bevezetés

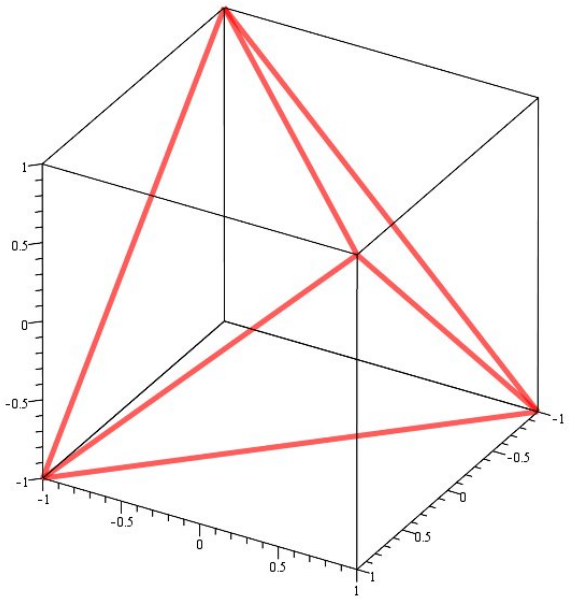
3D ábrák készítésekor hasonló opciókat és eljárásokat használhatunk, mint az előző alfejezet síkbeli grafikonjainál, csak legtöbbször az ott megismert eljárások 3D változatát kell alkalmazni: `plot` - `plot3d`, `pointplot` - `pointplot3d`, `textplot` - `textplot3d`, `implicitplot` - `implicitplot3d`, stb.
Itt is használható a `numpoint` opció, de a mintapontok vételezését szabályozhatjuk a `grid = [`**Xosztás, Yosztás**`]` opcióval is, amely azt adja meg, hogy milyen beosztású négyzetrács pontjait vegye a Maple (alapértelmezett értéke [49,49]).
Az `orientation = [theta, phi, psi]` opció azt mondja meg, hogy az elkészült ábrát az X, majd az Y és végül a Z tengely körül milyen szögekkel elforgatva szeretnénk látni. A harmadik szög alapértelmezett értéke 0, ez el is hagyható.
A legtöbb opció értéke – a 2D esethez hasonlóan – a jobb egérgombbal a megjelenített grafikára kattintva a fölbukkanó menüből interaktívan változtatható.
Speciális ábrák készítéséhez itt is a `plots` és a `plottools` csomag eljárásai szükségesek, ezért a továbbiakban föltesszük, hogy ezeket betöltöttük:

```
with(plots): with(plottools):
```

3D grafika I.

3D grafikát készíthetünk a Maple belső ábrázolása szerinti `PLOT3D` struktúrák létrehozásával, de az előző alfejezethez hasonlóan itt is célszerűbb a felhasználói szintű eljárásokat és csomagokat használni.
Az alábbi példa egy szabályos tetraéder előállítását mutatja PLOT3D parancsokkal.

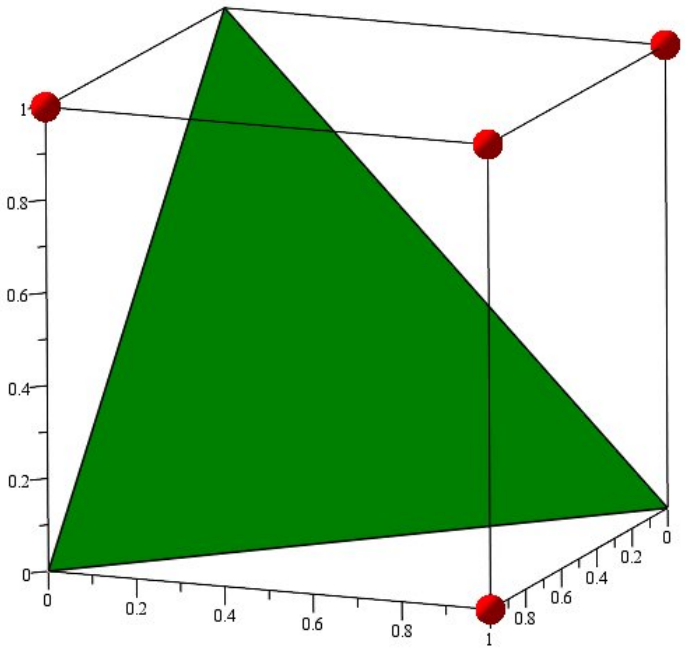
```
PLOT3D(POLYGONS([[1,1,1], [-1,-1,1], [-1,1,-1]],
  [[1,1,1], [-1,-1,1], [1,-1,-1]],
  [[-1,1,-1], [1,-1,-1], [-1,-1,1]],
  [[-1,1,-1], [1,-1,-1], [1,1,1]]),
  STYLE(LINE), THICKNESS(5), COLOR(RGB,1,0,0), AXESSTYLE(BOX),
  ORIENTATION(30,60), SCALING(CONSTRAINED));
```



3D grafika II.

A `pointplot3d` és a `polygonplot3` nagyjából úgy működnek, mint 2D változataik, csak most a térbeli pontokat 3 koordinátájukkal kell megadni. Egy példa:

```
p1 := pointplot3d([[1,1,1],[0,1,1],[1,0,1],[1,1,0]], style=point,
color=red, symbol=solidcircle, symbolsize=32):
p2 := polygonplot3d([[0,0,1],[0,1,0],[1,0,0]], color=green):
display(p1,p2);
```



3D grafika III.

A `plot3d` eljárás opcióinak nagy része megegyezik a megfelelő `plot` opciókkal (bár például a `legend` itt nem használható).
Az ábrázolás stílusát a `style` opció definiálja. Lehetséges értékei (zárójelben az alternatív elnevezések): `surface` (`patchnogrid`), `surfacewireframe` (`patch`), `contour`, `surfacecontour` (`patchcontour`), `wireframe` (`line`), `wireframeopaque` (`hidden`) és `point`.
Készítünk néhány ábrát, melyek a g(x,y) függvényt különböző stílusokban jelenítik meg.

```
g := (x,y) -> sin(x)*sin(y);
intv := -2*Pi..2*Pi;
P := Array(1..2,1..3):
```

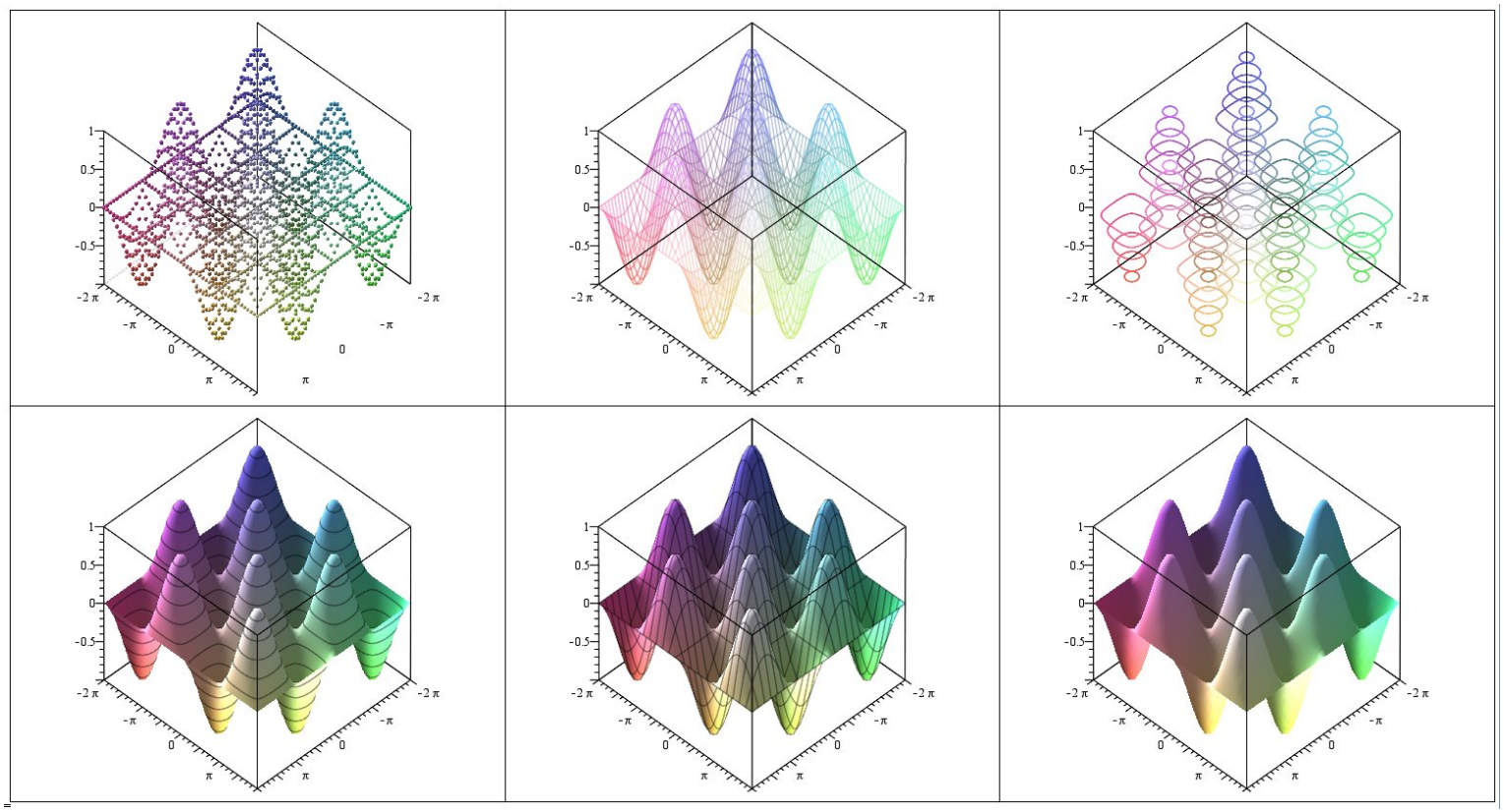
$$g(x,y) := (x,y) \to \sin(x)\sin(y)$$
$$intv := -2\pi \ldots 2\pi$$

```
P[1,1] := plot3d(g, intv, intv, style=point,symbol=solidcircle):
P[1,2] := plot3d(g, intv, intv, style=wireframe):
P[1,3] := plot3d(g, intv, intv, style=contour):
P[2,1] := plot3d(g, intv, intv, style=patchcontour):
P[2,2] := plot3d(g, intv, intv, style=surfacewireframe):
P[2,3] := plot3d(g, intv, intv, style=surface):
```

3D grafika IV.

Az előző oldalon elkészített grafikonok megjelenítése:

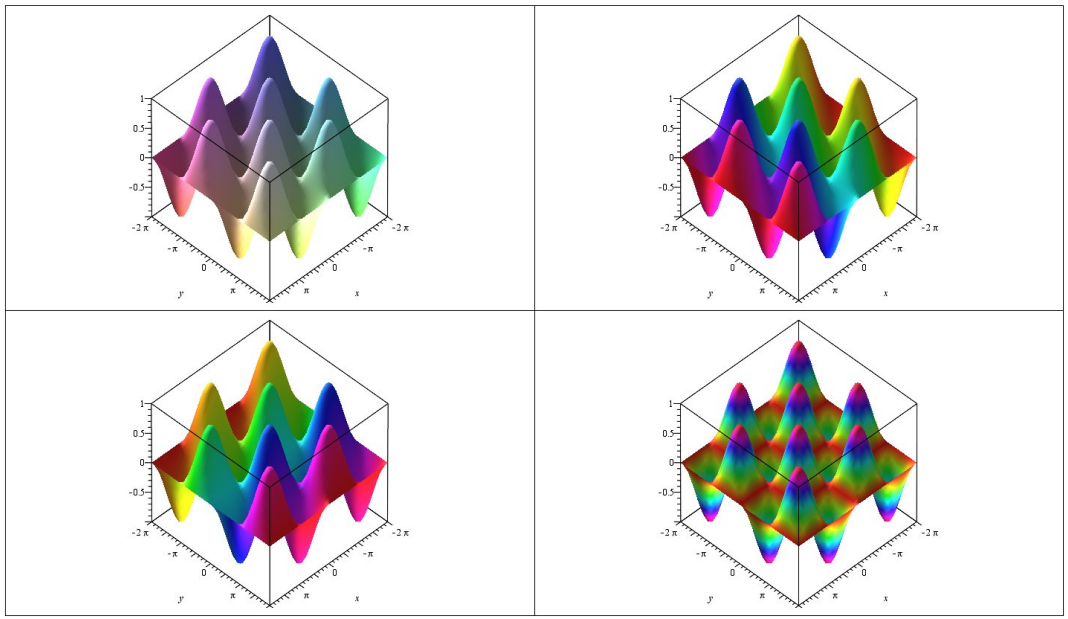
```
display(P);
```



3D grafika V.

A 3D grafikonok szinezését nem csak színkonstansokkal, hanem színfüggvényekkel is megadhatjuk.
Ezekben szerepelhet tetszőleges, a változóktól – itt x-től és y-tól – függő kifejezés.
Hasonlítsuk össze az első, alapértelmezett színeket használó grafikont a többivel.

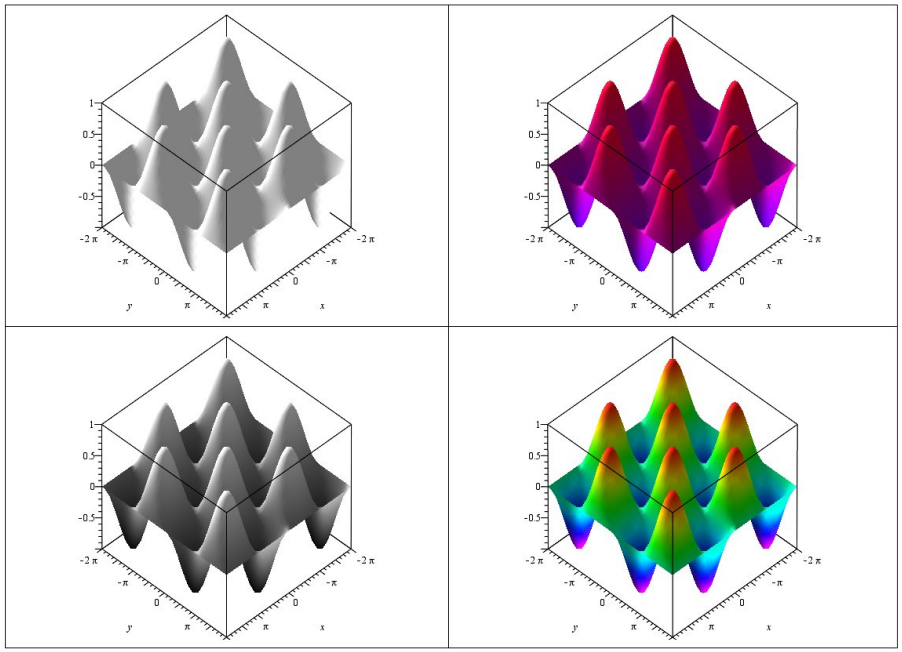
```
P := Array(1..2,1..2): kif := sin(x)*sin(y): intv := -2*Pi..2*Pi:
P[1,1] := plot3d( kif, x=intv, y=intv, style=surface ):
P[1,2] := plot3d( kif, x=intv, y=intv, style=surface, color= x ):
P[2,1] := plot3d( kif, x=intv, y=intv, style=surface, color= sqrt(x^2+y^2) ):
P[2,2] := plot3d( kif, x=intv, y=intv, style=surface, color= abs(kif) ):
display(P);
```



3D grafika VI.

Részben hasonló szerepű a (színezett) árnyékolást megadó `shading` opció, lehetséges értékei: `none, xy, xyz, xyz, z, zgreyscale, zhue`. A `color` opciónál megadott színezésnek nagyobb a prioritása ennél, tehát fölülírhatja a árnyékolásnál használt színeket.
Az árnyékolási opciók hatása látható az alábbi ábrán.

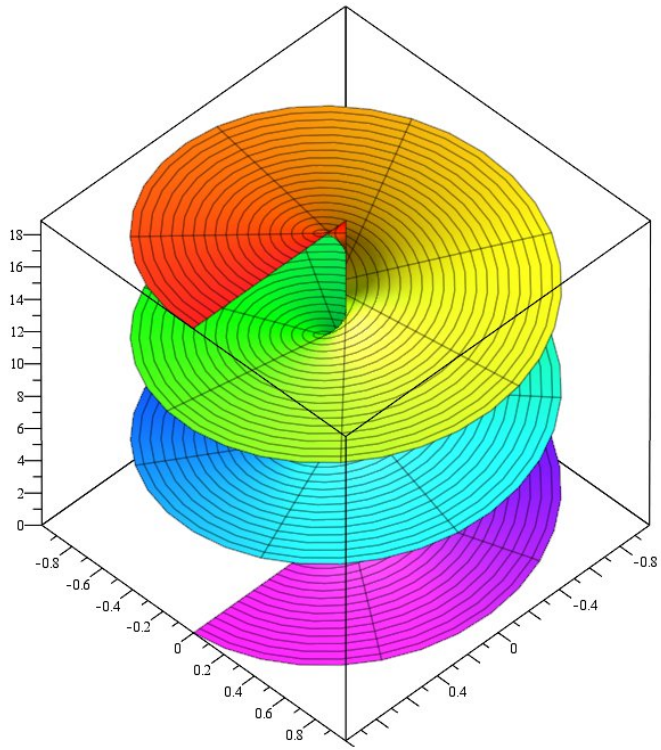
```
P := Array(1..2,1..2): kif := sin(x)*sin(y): intv := -2*Pi..2*Pi:
P[1,1] := plot3d( kif, x=intv, y=intv, style=surface, shading=none):
P[1,2] := plot3d( kif, x=intv, y=intv, style=surface, shading=z ):
P[2,1] := plot3d( kif, x=intv, y=intv, style=surface, shading=zgreyscale):
P[2,2] := plot3d( kif, x=intv, y=intv, style=surface, shading=zhue):
display(P);
```



3D grafika VII.

Megadhatunk paraméteres felületeket is, ekkor a 2D esethez hasonlóan listában soroljuk föl a paraméteres kifejezéseket:

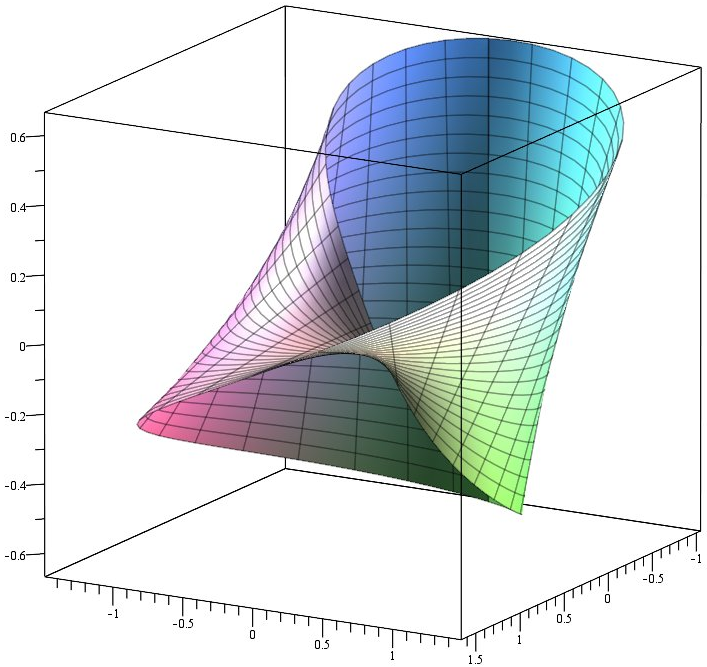
```
plot3d([u*cos(theta), u*sin(theta), theta],
       u=0..1, theta=0..6*Pi, shading=zhue, grid=[100, 100]);
```



3D grafika VIII.

A geometriából ismert Möbius szalagot paraméteres egyenletével így ábrázolhatjuk:

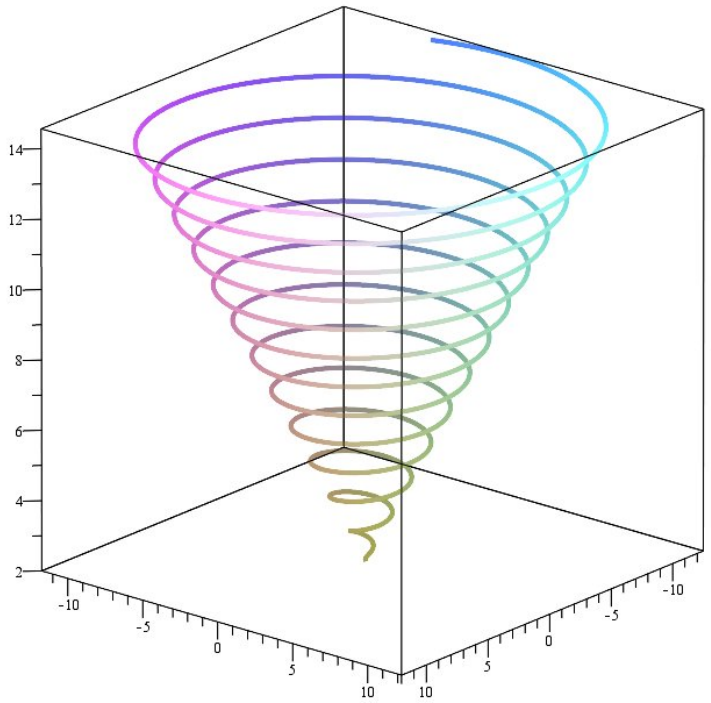
```
plot3d([(1 + u *cos(1/2*theta))*cos(theta),
        (1+u*cos(1/2*theta))*sin(theta),
         u*sin(1/2*theta)],
       u = -2/3..2/3, theta=-Pi..Pi, orientation=[30,75]);
```



3D grafika IX.

Három dimenziós térgörbéket a `spacecurve` eljárással rajzoltathatunk föl. (A paraméteres felületekhez képest annyi a különbség, hogy most csak egy paraméter van.)

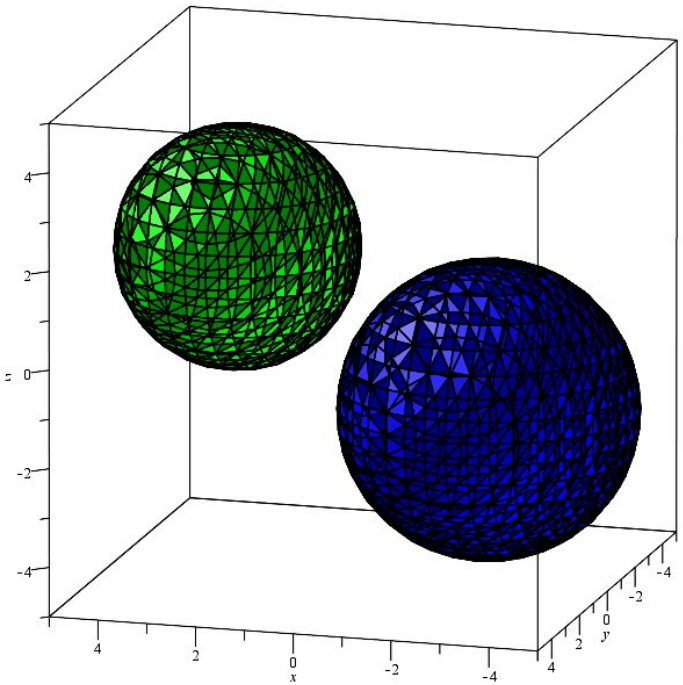
```
spacecurve([t*cos(2*Pi*t),t*sin(2*Pi*t),2+t], t=0..4*Pi,
           orientation=[40,70], thickness=4, numpoints=5000);
```



3D grafika X.

Az `implicitplot3d` kétdimenziós változatához hasonlóan működik, csak most három változóval (az alábbi példában x,y és z) dolgozunk.
Az ábrán két, implicit egyenletével megadott gömböt látunk.

```
implicitplot3d([(x+2)^2+(y+2)^2+(z+2)^2=9, (x-2)^2+(y-2)^2+(z-2)^2=6],
               x=-5..5, y=-5..5, z=-5..5, color=[blue,green],
               numpoints=10000,orientation=[106, 78]);
```



3D grafika XI.

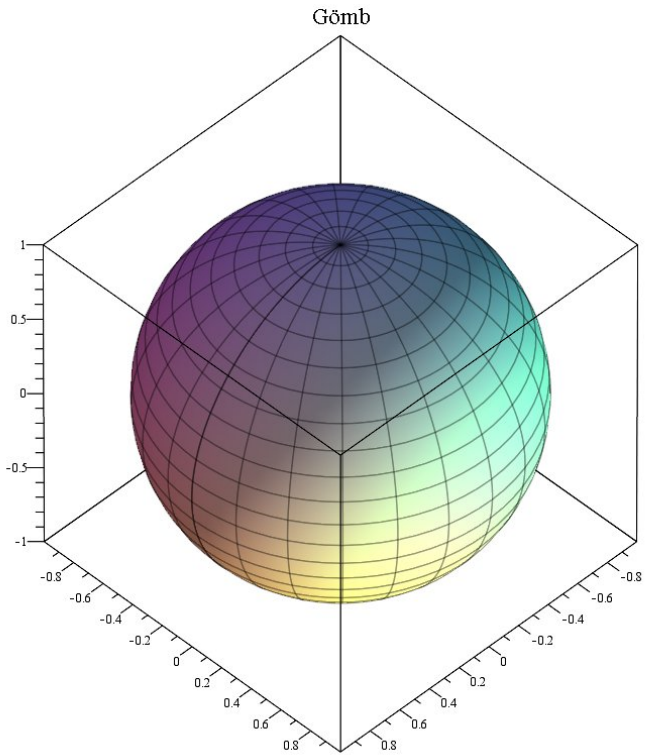
A 3D ábrázolások során használhatunk különböző koordináta-rendszereket. Eddig az alapértelmezett Descartes-féle (`rectangular`) koordinátarendszerrel dolgoztunk.
A választható koordinátarendszerek felsorolását a `?coords` Help oldalon találjuk. Ezek közül mutatunk be néhányat.
A Maple régebbi változataiban a különböző koordináta-rendszerek szerinti megjelenítést külön-külön speciális eljárások végezték, például a `sphereplot`, `cylinderplot`, stb. Ezek elavultnak számítanak, mindegyik helyett a `plot3d` használandó a `coords=...` opcióval.

3D grafika XII.

Első példánkban gömbkoordinátákat alkalmazunk, melyek a Descartes-féle koordinátákkal így függenek össze:

$x = u\cos(\theta)\sin(\psi), y = u\sin(\theta)\sin(\psi), z = u\cos(\psi)$

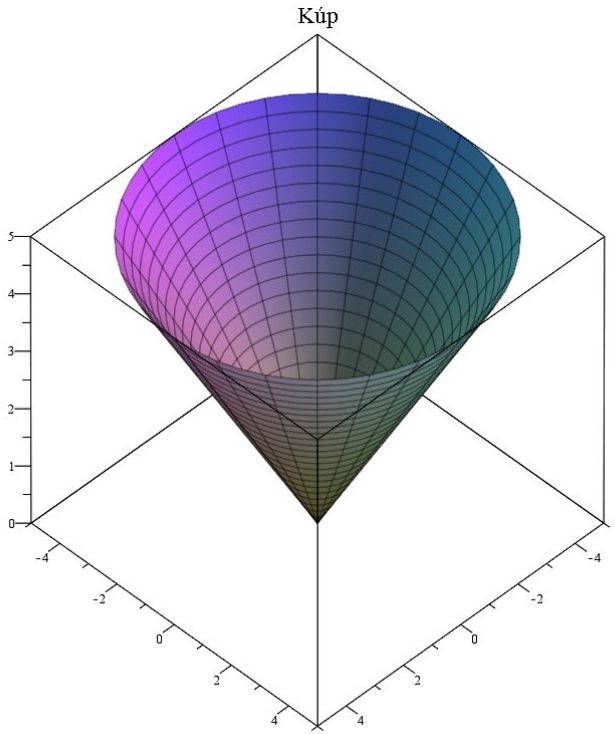
```
plot3d(1, theta=0..2*Pi, psi=0..Pi, coords=spherical,
       titlefont=[bold,18], title="Gömb", numpoints=5000);
```



3D grafika XIII.

Következő példáinkban hengerkoordinátákat használunk, ezekből a Descartes-féle koordináták így adódnak:
$x = u\cos(\theta), y = u\sin(\theta), z = w.$

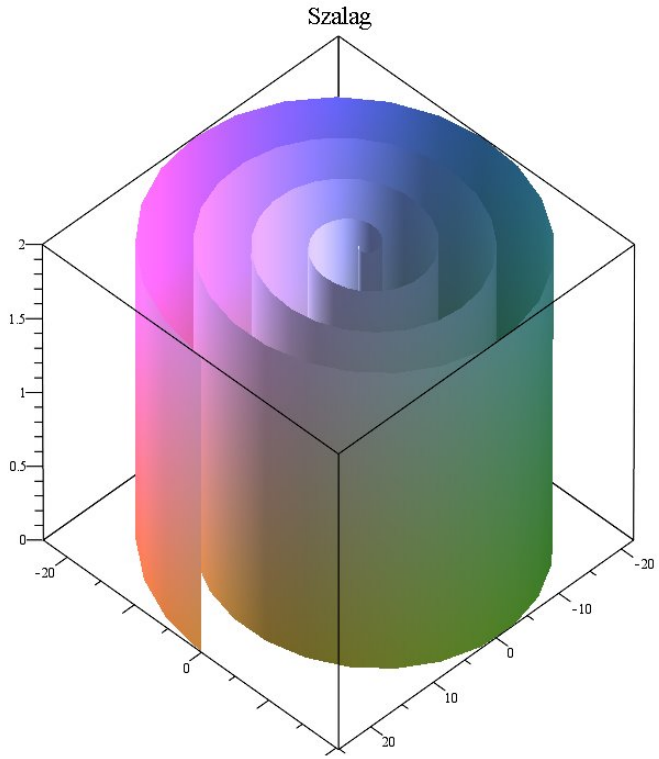
```
plot3d(u, theta=0..2*Pi, u=0..5, coords=cylindrical,
       titlefont=[bold,18], title="Kúp");
```



3D grafika XIV.

Még egy hengerkoordinátás példa:

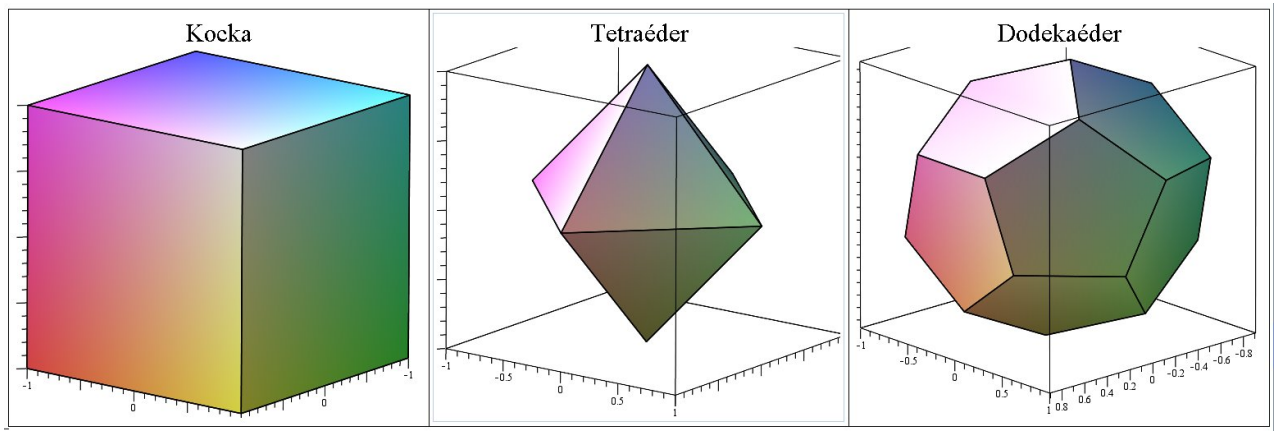
```
plot3d(theta, theta=0..8*Pi, u=0..2, coords=cylindrical,
       style=surface, titlefont=[bold,18], title="Szalag",
       grid=[100,100]);
```



3D grafika XV.

A `polyhedraplot` eljárás szabályos poliédereket ábrázol:

```
P := Array(1..3):
P[1] := polyhedraplot([0,0,0], polytype=hexahedron,
                      title="Kocka", titlefont=[bold,24]):
P[2] := polyhedraplot([0,0,0], polytype=octahedron,
                      title="Tetraéder", titlefont=[bold,24]):
P[3] := polyhedraplot([0,0,0], polytype=dodecahedron,
                      title="Dodekaéder", titlefont=[bold,24]):
display(P);
```



Animáció I.

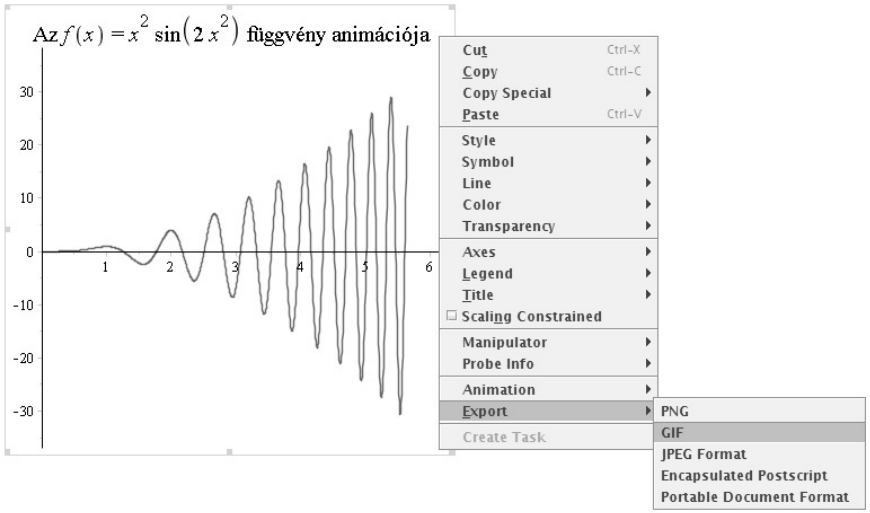
Animációk készítéséhez a `plots` csomagban több segédeljárást is találunk: `animate`, `animate3d`, `animatecurve`, `display`.
Az ábrák csak az animációk egy adott fázisát mutatják. A példákat lefuttatva a Maple alább látható animációs menüje segítségével lépésenként követhetjük a megjelenítés folyamatát.

Animáció II.

Függvények görbéjét animálva ábrázolja az `animatecurve`. Paraméterezése a `plot`-hoz hasonló, de nem „egyből" rajzolja ki a görbét, hanem fokozatosan jeleníti meg.
A `frames=50` paraméterrel adjuk meg, hogy hány lépésben történjen az animáció.
Az animációkat animált GIF fájlokként menthetjük el az ábrákon a jobb klikk után felbukkanó menű segítségével.

```
restart: with(plots):
f := x -> x^2*sin(2*x^2);
animatecurve(f, 0..2*Pi, numpoints=1000,
  title=typeset("Az ",'f(x)' = x^2*sin(2*x^2), " függvény animációja"),
   titlefont=[Times, 14], frames=50 );
```

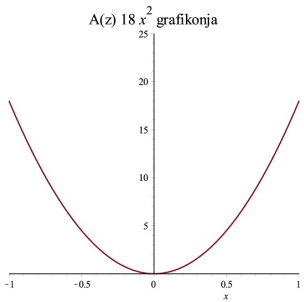
$$f := x \to x^2 \sin(2x^2);$$



Animáció III.

Az animációk legegyszerűbb létrehozási módja, hogy a „kockánként” (például alkalmas `plot` parancsokkal) elkészített képeket először elmentjük, majd a `display` eljárással sorban, egymás után megjelenítjük.
A `display` első paramétere a képelemeket tartalmazó *lista*, második paramétereként az `insequence=true` opcióval jelöljük, hogy egymás után szeretnénk látni a képeket. (A `display` az `insequence=false` alapértelmezett opció esetén egyetlen ábrán, egyszerre jeleníti meg a lista tartalmát.)
A `plot`-nál megismert legtöbb opció – kivéve pl. `legend`, `discont` és `adaptive` – használható a `display` esetében is.
Az alábbi példában a jx^2 parabola-sereget animáljuk 25 lépésben. A `P` struktúra *tábla*, ebből a `convert(P,list)` csinál listát.

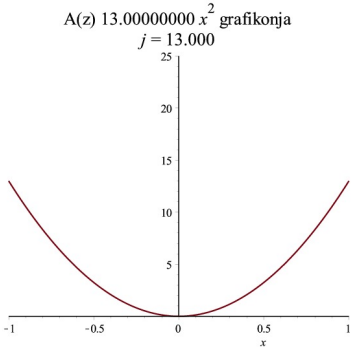
```
j := 'j':
for j from 1 to 25 do
  P[j] := plot(j*x^2, x = -1..1,
               title = typeset("A(z) %1 grafikonja", j*x^2 ),
               titlefont = ["Roman", 16]):
end do:
display(convert(P,list), insequence=true);
```



Animáció IV.

Az `animate` eljárás egy fokkal könnyebbé teszi az előzőek végrehajtását.
Első paramétere az egyes kockákat generáló eljárás neve, például `plot`, `plot3d`, stb. Második paramétereként a generáló eljárás paramétereit adjuk meg egy listában.
A harmadik paraméter az „animációs változó”, amelynek a megadott értéktartományon belüli változása generálja az egymás utáni képkockákat.
Végül itt is használhatjuk a `frames` opciót a lépések számának megadására (alapértelmezett értéke 25.) Ezzel az előző megoldáshoz hasonló animációt kapunk.

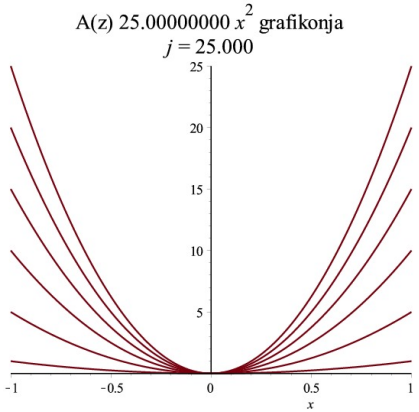
```
j := 'j':
animate(plot,
  [ j*x^2, x=-1..1,
    title = typeset("A(z) %1 grafikonja", j*x^2 ),
    titlefont = ["Roman", 16] ],
  j =1..25, frames=50 );
```



Animáció V.

A `trace = [...]` opcióval azt írhatjuk elő, hogy a megjelenítés közben csak a listában megadott sorszámú képkockákat szeretnénk látni (nem az összeset), de ezek „egymásra vetítve” jelenjenek meg, tehát a régebbiek is megmaradjanak az újak megjelenése után.

```
j := 'j':
animate(plot,
   [ j*x^2, x =-1..1, title = typeset("A(z) %1 grafikonja", j*x^2 ),
     titlefont = ["Roman", 16]  ],
   trace = [1,5,10,15,20,25],
   j=1..25);
```

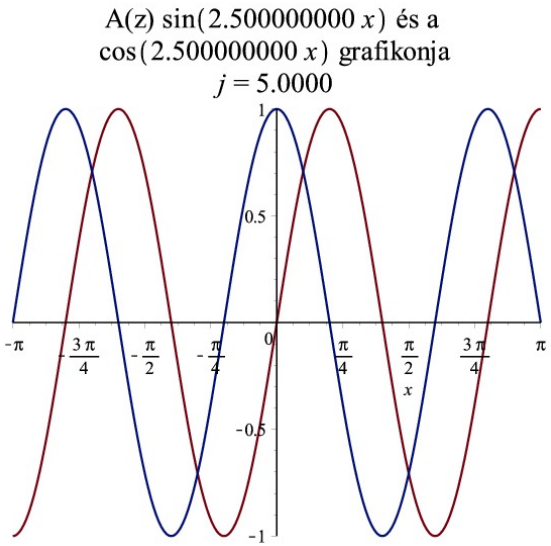


Animáció VI.

Az `animate` – megfelelő `plot` listával vagy halmazzal – több függvény grafikonját is tudja egyszerre animálni.

Itt a $\left[\sin\left(\frac{k}{2}x\right), \cos\left(\frac{k}{2}x\right)\right]$ függvénycsalád elemeit ábrázoljuk k=1-től 25-ig.

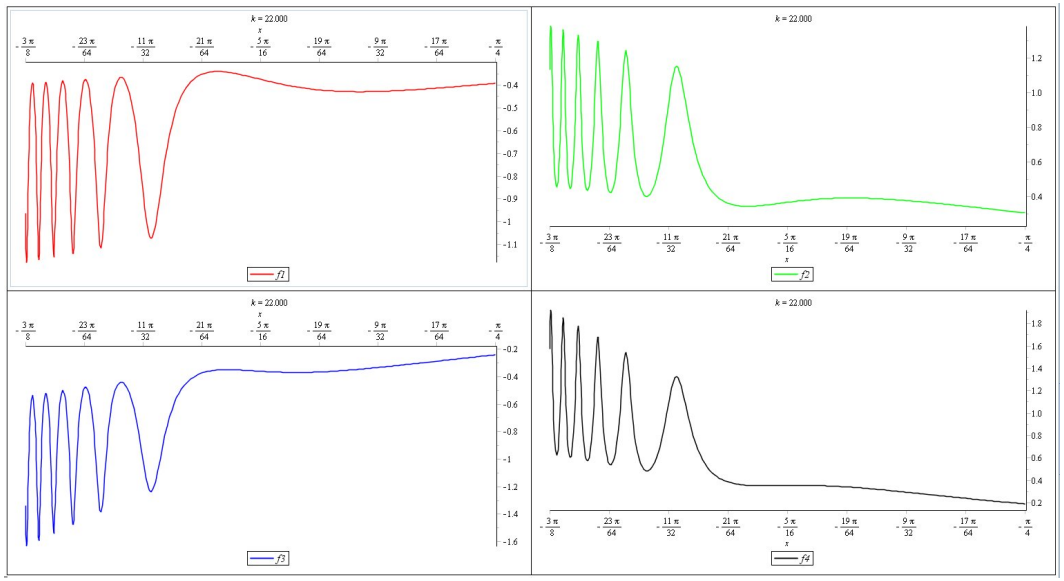
```
animate(plot,
   [[sin(j/2*x), cos(j/2*x)], x =-Pi..Pi,
    title = typeset("A(z) %1 és a %2 grafikonja",sin(j/2*x), cos(j/2*x)),
    titlefont = ["Roman", 16] ],
  j =1..25);
```



Animáció VII.

Az `animate`-tel készült több animációt tömbben (`Array`) elmentve a `display` eljárás segítségével ezeket egymás mellé helyezve, párhuzamosan futó animációkként is megjeleníthetjük.

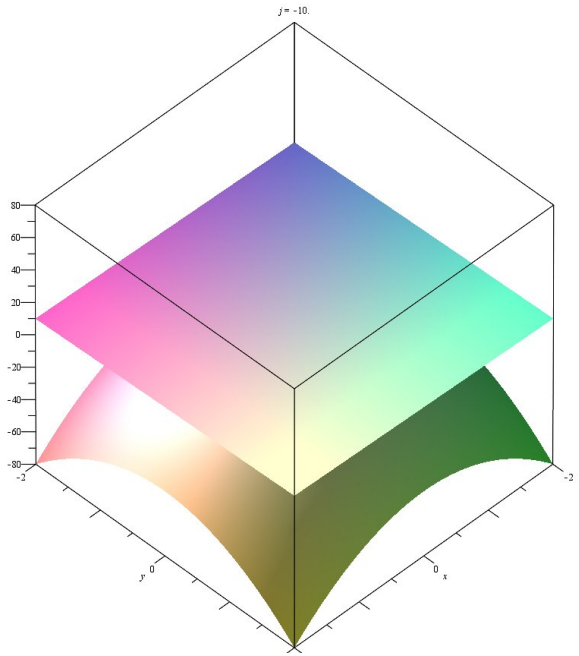
```
f1:= x/(2+sin(x^k)): f2:=x^2/(2+sin(x^k)): f3:=x^3/(2+sin(x^k)): f4:=x^4/(2+sin(x^k)):
a,b := -Pi/4, -3/8*Pi:
Plotanim := Array(1..2,1..2):
Plotanim[1,1] := animate(plot, [f1, x=a..b, color=red,   legend=['f1'] ], k=1..25):
Plotanim[1,2] := animate(plot, [f2, x=a..b, color=green, legend=['f2'] ], k=1..25):
Plotanim[2,1] := animate(plot, [f3, x=a..b, color=blue,  legend=['f3'] ], k=1..25):
Plotanim[2,2] := animate(plot, [f4, x=a..b, color=black, legend=['f4'] ], k=1..25):
display(Plotanim);
```



Animáció VIII.

Bár létezik speciális `animate3d` eljárás (lásd `?animate3d`), ez elavultnak számít, helyette az `animate`-et használjuk 3D animációk elkészítésére is.
Ekkor a képkockákat szükségképpen valamely 3D ábrákat előállító Maple eljárással generáltatjuk.

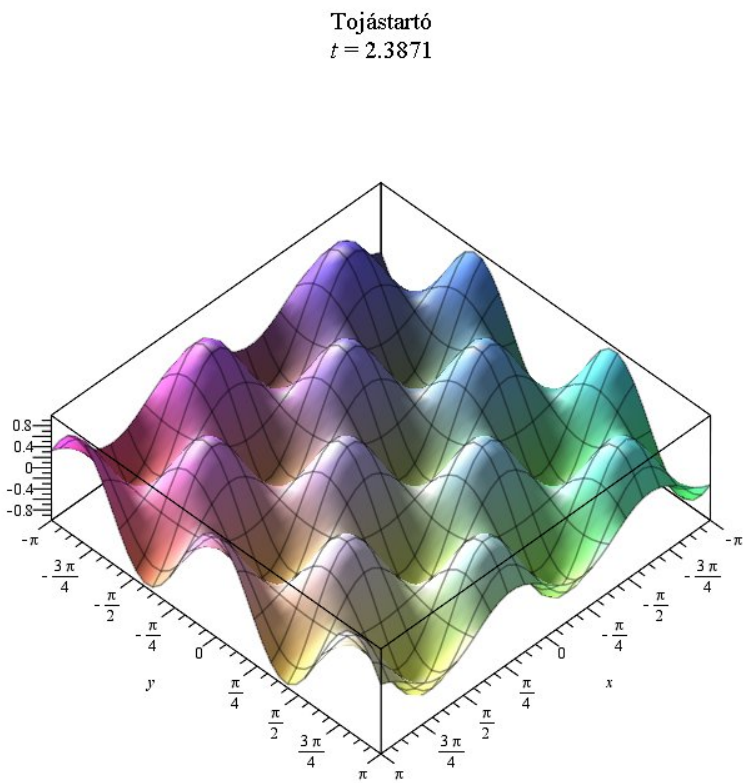
```
animate(plot3d,
     [[j*x^2 + j*y^2, x+y-j], x= -2..2, y= -2..2, style=surface,
      title="3D animáció", titlefont = ["ROMAN", 16]],
      j = -10..10);
```



Animáció IX.

Az alábbi 3D animáció tojástartóra emlékeztető felületeket jelenít meg.

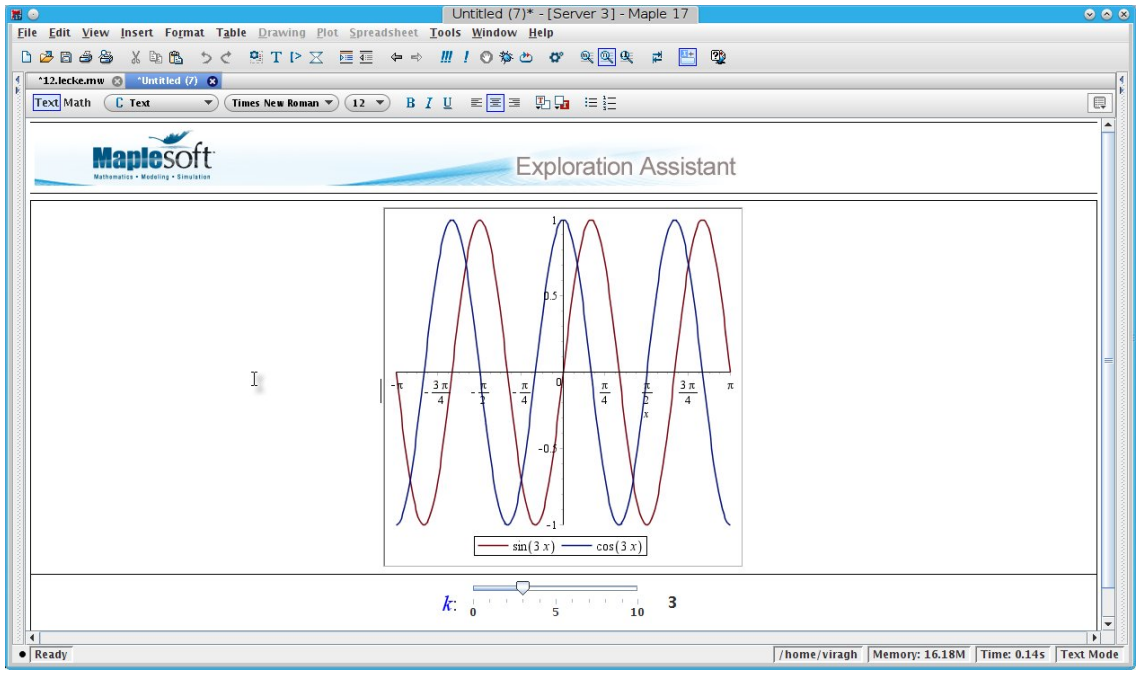
```
animate(plot3d,
[sin(t*x) * cos(t*y),x=-Pi..Pi,y=-Pi..Pi, axes = "boxed",grid=[40,40],
scaling=constrained, titlefont=[bold, 16], title="Tojástartó" ],
t = 2..5, frames=32);
```



Animáció X.

Az `Explore` eljárás használatával is elérhetünk az előző animációkhoz hasonló effektusokat. Ekkor – a `newsheet` opcióval – külön ablakban futtatható a grafikont készítő eljárás, és a benne szereplő paraméterek interaktívan, csúszkák és gombok segítségével változtathatók. Az első példa hasonlít az `animate`-tel korábban megadottra.

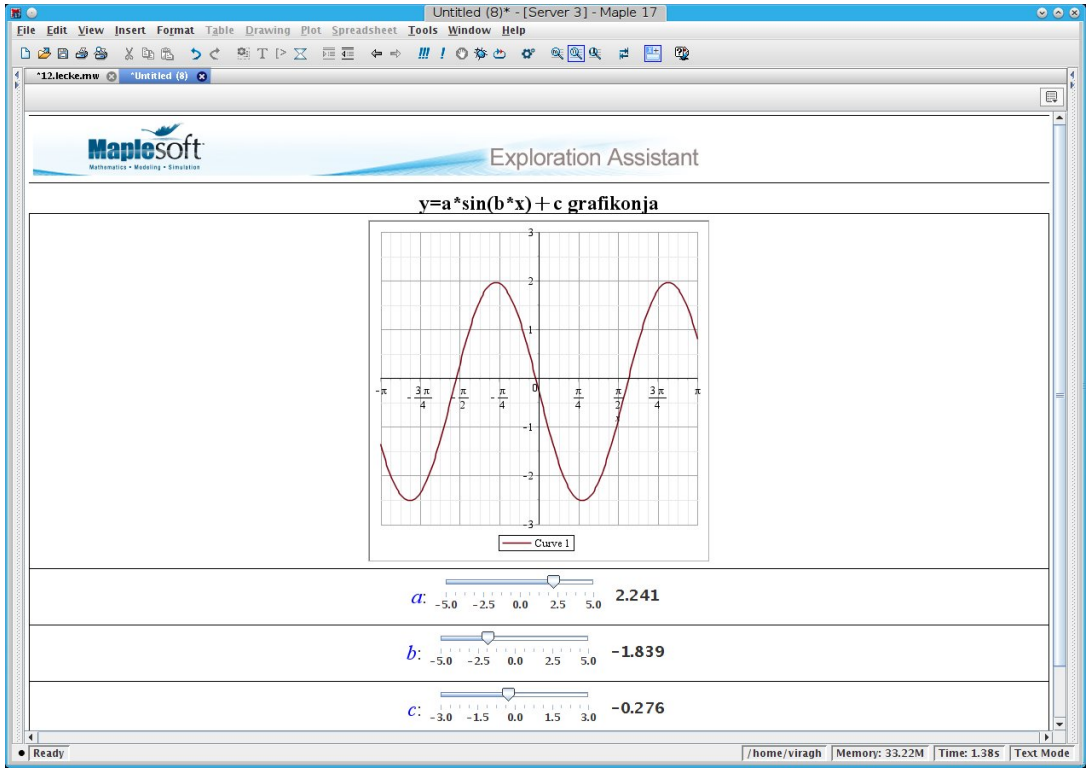
```
restart;
Explore(plot([sin(k*x),cos(k*x)], x=-Pi..Pi,
            legend=[typeset(sin(k*x)),typeset(cos(k*x))]),
        newsheet);
```



Animáció XI.

Második példánk részletesebben mutatja az `Explore` lehetséges opcióit.

```
Explore( plot(a*sin(b*x)+c, x=-Pi..Pi,view=-3..3,gridlines),
         parameters     = [a=-5. .. 5., b=-5. .. 5., c=-3. .. 3.],
         initialvalues  = [a=-3., b=-3., c=-1.],
         title          = "y=a*sin(b*x)+c grafikonja",
         newsheet);
```



13. előadás

7. Analízis alkalmazások

Analízis alkalmazások

Az egy- és többváltozós analízis kurzusok számos olyan témát tárgyalnak, melyeknél mind az elmélet megértésében, mind a gyakorlati alkalmazásokban hasznos segédeszköz lehet a Maple. A fejezet további részében ezeket a témaköröket tekintjük át röviden.

- Határértékszámítás
- Differenciálás
- Integrálás
- Optimalizálás
- Differenciálegyenletek megoldása

A `Student` csomag `Precalculus`, `Calculus1` és `Multivariate Calculus` részcsomagjai mind az analízis előadásokhoz, mind a hallgatók otthoni munkájához, különböző számolási feladatok megoldásához hasznos segédeljárásokat, demókat tartalmaznak. Közülük csak néhányat említünk a továbbiakban, de érdemes behatóbban is megismerkedni velük.

Határérték I.

Kifejezések (vagy függvények) adott helyen vett határértékét a `limit` eljárással kaphatjuk meg. Általános alakja:
`limit(f, x=a)`, jelentése az `f` **kifejezésnek** az `x=a` helyen vett határértéke, azaz $\lim_{x\to a} f$.
`limit(f, x=a, irány)` jelentése az `f` **kifejezésnek** az `x=a` helyen vett **irány** szerint (`left` vagy `right`) határértéke, azaz $\lim_{x\to a^-} f$ vagy $\lim_{x\to a^+} f$.
A `limit` nem csak véges `a` értékeket kezel, az `x = infinity` opció például a (plusz) végtelenben vett határértéket számolja ki.
Az eljárás inert változata, a `Limit` kiértékeletlenül, de 2D formában adja vissza argumentumait, s így segíthet az eredmények olvashatóbbá tételében, vagy bonyolultabb határértékek esetén bizonyos „számítási trükkök” megvalósításában.

```
Limit(x^3/sin(x^3), x =0) = limit(x^3/sin(x^3), x =0);
```

$$\lim_{x\to 0} \frac{x^3}{\sin\left(x^3\right)} = 1$$

Határérték II.

A Maple-nek természetesen nem okoznak gondot az ismert határérték példák.

```
Limit(1/x,x=infinity)= limit(1/x,x=infinity);
Limit(1/x^2,x=0)= limit(1/x^2,x=0);
Limit(sin(x)/x,x=0)= limit(sin(x)/x,x=0);
Limit((1+ x/n)^n, n=infinity) = limit ((1+ x/n)^n, n=infinity);
```

$$\lim_{x\to\infty} x^{-1} = 0$$
$$\lim_{x\to 0} x^{-2} = \infty$$
$$\lim_{x\to 0} \frac{\sin(x)}{x} = 1$$
$$\lim_{n\to\infty} \left(1+\frac{x}{n}\right)^n = \mathrm{e}^x$$

Határérték III.

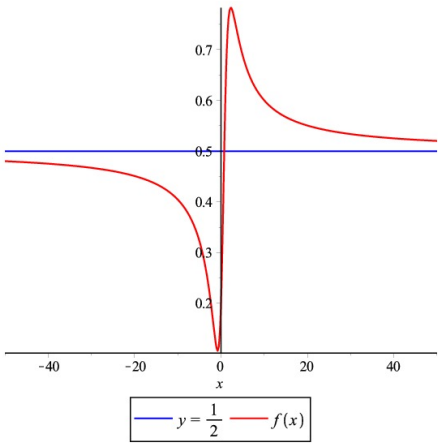
Az ábra alapján is sejthető, hogy az alábbi racionális törtfüggvény $\pm\infty$-ben vett határértéke $\frac{1}{2}$.

```
f:=x->(x^2+x+1)/(2*x^2-2*x+5);
Limit(f(x),x=-infinity)=limit(f(x),x=-infinity);
Limit(f(x),x=infinity)=limit(f(x),x=infinity);
plot([1/2,f(x)],x=-50..50, color=[blue,red], legend=['y=1/2','f(x)']);
```

$$f := x \to \frac{x^2+x+1}{2x^2-2x+5}$$

$$\lim_{x\to-\infty} \frac{x^2+x+1}{2x^2-2x+5} = \frac{1}{2}$$

$$\lim_{x\to\infty} \frac{x^2+x+1}{2x^2-2x+5} = \frac{1}{2}$$



Határérték IV.

Ha a `limit` visszaadott értéke egy **tartomány**, ez úgy értendő, hogy *ha* létezik a keresett határérték, *akkor* az benne van a megadott tartományban.

```
limit(sin(x),x=infinity);
```

$$-1..1$$

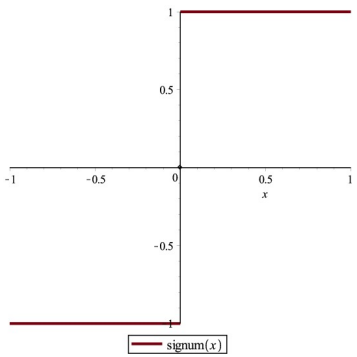
Határérték V.

Nem folytonos függvények esetében – ilyen például a Maple-ben `signum`-mal jelölt előjelfüggvény – előfordulhat, hogy a bal- és a jobboldali határértékek különböznek, vagy akár nem is létezik közülük valamelyik. Az ábra alapján ellenőrizhető a Maple válaszainak helyessége.

```
limit(signum(x), x = 0);
limit(signum(x), x = 0, left);
limit(signum(x), x = 0, right);
```

$$undefined$$
$$-1$$
$$1$$

```
plot(signum(x),x = -1..1,discont=true,thickness=3,legend='signum(x)');
```



Határérték VI.

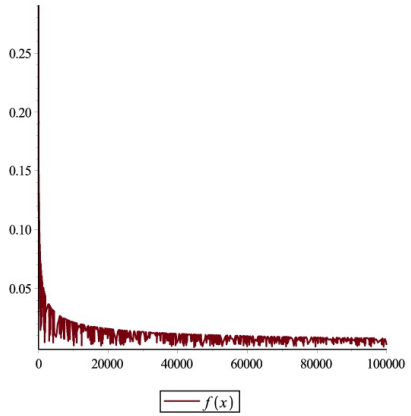
Az alábbi példában egy `f(x)` függvény végtelenben vett határértékét határozzuk meg.

```
f := x -> sqrt(abs(sin(x)*cos(x))*ln(x)/x);
Limit(f(x), x=infinity) = limit (f(x), x=infinity);
```

$$f := x \to \sqrt{\frac{|\sin(x)\cos(x)|\ln(x)}{x}}$$

$$\lim_{x\to\infty} \sqrt{\frac{|\sin(x)\cos(x)|\ln(x)}{x}} = 0$$

```
plot(f, 0..10^5, legend='f(x)');
```



A grafikonon látottak is alátámasztják az eredmény helyességét.

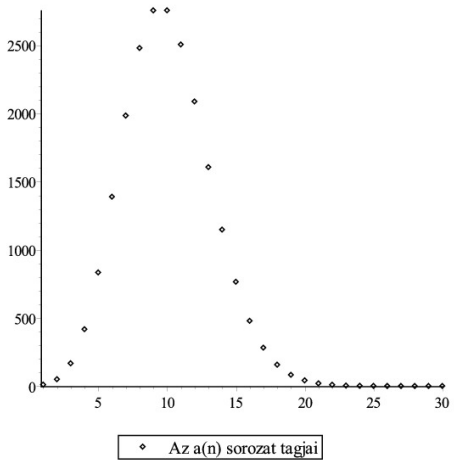
Határérték VII.

Nézzünk néhány példát (végtelen) sorozatok határértékével és összegével kapcsolatban. Az a_n sorozatot `a := n -> kif` alakban, mint a természetes számokon értelmezett függvényt adjuk meg.
A kalkulusból ismert, hogy az alábbi 0-sorozat kezdetben elég drasztikusan növekszik. Ezt mutatja a grafikon is.

```
a := n -> 10^n/n!;
Limit(a(n), n = infinity) = limit(a(n), n = infinity);
plot([seq([k,a(k)], k = 1..30)], style = point,
     legend="Az a(n) sorozat tagjai");
```

$$a := n \to \frac{10^n}{n!}$$

$$\lim_{n\to\infty} \frac{10^n}{n!} = 0$$



Határérték VIII.

A végtelen mértani sor összegének meghatározása a $|q| < 1$ esetben a következő módon történhet (a bal oldalon a `Limit` és a `Sum` inert alakot használtuk):

```
assume(abs(q)<1);
Limit(Sum(a[1]*q^k, k=0..n),n=infinity)  =
limit(sum(a[1]*q^k, k=0..n), n=infinity);
```

$$\lim_{n\to\infty}\left(\sum_{k=0}^{n} a_1 q\sim^k\right) = -\frac{a_1}{q\sim -1}$$

A teljes igazság az, hogy a `limit` használata itt szükségtelen, mivel a `sum` eleve képes a határérték kiszámítására:

```
Sum(a[1]*q^n, n=0..infinity) = sum(a[1]*q^n, n=0..infinity);
```

$$\sum_{n=0}^{\infty} a_1 q\sim^n = -\frac{a_1}{q\sim -1}$$

Differenciálás I.

A Maple több eljárással támogatja a közönséges és a parciális differenciálhányadosok meghatározását, sőt a differenciáloperátorokkal való számolást is.
Az alább ismertetetteken túl hasznos lehet még a `Student` csomag `Calculus1` részcsomagja és a `VectorCalculus` csomag.
Ügyeljünk arra, hogy az (algebrai) **kifejezések** és a **függvények** a Maple-ben két különböző szintaktikus kategóriát alkotnak, ezért mást jelent egy *függvényt* deriválni (az eredmény a *derivált függvény*) és megint mást egy *kifejezést* differenciálni (az eredmény egy *kifejezés*).

Differenciálás II.

A differenciálhányados matematikai definíciója szerint $f'(x) = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h}$. Az $f(x)$ függvény x helyen vett differenciálhányadosát, illetve az $f'(x)$ derivált függvényt a Maple limit eljárását felhasználva így számíthatjuk:

```
fder:= x->limit((f(x+h)-f(x))/h,h=0);
```

$$fder := x \to \lim_{h\to 0} \frac{f(x+h)-f(x)}{h}$$

```
f := x -> x^2 -2*x;
'fder(z)' = fder(z);
'fder(0)' = fder(0);
```

$$f := x \to x^2 - 2x$$
$$fder(z) = -2 + 2z$$
$$fder(0) = -2$$

```
f:= x-> x^5*ln(x)+x^2*tan(x);
'fder(t)' = fder(t);
'fder(1)' = fder(1);
```

$$x \to x^5 \ln(x) + x^2 \tan(x)$$
$$fder(t) = 2\tan(t)t + (\tan(t))^2 t^2 + t^2 + 5\ln(t)t^4 + t^4$$
$$fder(1) = 2 + (\tan(1))^2 + 2\tan(1)$$

Differenciálás III.

Maple kifejezéseket a bennük szereplő (változó)nevek szerint a beépített `diff` eljárással differenciálhatunk, az eredmény *kifejezés*. A `diff` első paramétere a differenciálandó kifejezés, a második azt mondja meg, hogy mi szerint kell deriválni. Lásd `?diff`.
A `diff` inert párja, a `Diff` függvény nem számol semmit, csak 2D formában visszhangozza argumentumait. Ezt többször felhasználjuk az alábbi példákban.
Maple *függvényeket* a `D` operátorral differenciálhatunk, az eredmény a *derivált függvény*. Lásd `?D`.

```
poli := x^2+x+1; fpoli := x -> x^2+x+1;
#ezzel ekvivalens fpoli := unapply(poli, x);
fexp:= x^2*sin(x*y); ffun := (x,y) -> x^2*sin(x*y);
```

$$poli := x^2 + x + 1$$
$$fpoli := x \to x^2 + x + 1$$
$$fexp := x^2 \sin(xy)$$
$$ffun := (x, y) \to x^2 \sin(xy)$$

```
Diff(poli, x) = diff(poli, x);
Diff(fexp, x) = diff(fexp, x);
```

$$\frac{d}{dx}\left(x^2 + x + 1\right) = 2x + 1$$
$$\frac{\partial}{\partial x}\left(x^2 \sin(xy)\right) = 2x \sin(xy) + x^2 \cos(xy)\, y$$

```
# egyváltozós függvényeknél a D(fpoli) jelölés is elég
Diff(fpoli, x) = D[1](fpoli);
Diff(ffun, x) = D[1](ffun);
```

$$\frac{\partial}{\partial x} fpoli = (x \to 2x + 1)$$
$$\frac{\partial}{\partial x} ffun = (x, y) \to (2x \sin(xy) + x^2 \cos(xy)\, y)$$

Differenciálás IV.

A `D` input paramétere mindig *függvény*, outputja pedig a *derivált függvény*. Néhány variáció a lehetséges jelölésekre:

```
sin_der := D(sin);
poli_der := D(x -> x^2 +1);
f:= x -> 3*ln(x)-x;
fder := D(f);
```

$$sin\_der := \cos$$
$$poli\_der := x \to 2\,x$$
$$f := x \to 3\,\ln(x) - x$$
$$fder := x \to 3\,x^{-1} - 1$$

A derivált függvény különböző helyeken fölvett értékeit a szokásos függvényalkalmazással kapjuk. Az eredmények már *kifejezések*, nem *függvények*.

```
sin_der(x); fder(x); D(f)(x);
```

$$cos(x)$$
$$\frac{3}{x} - 1$$
$$\frac{3}{x} - 1$$

A derivált függvény értéke az 1 helyen:

```
sin_der(0); fder(1); D(f)(1);
```

$$1$$
$$2$$
$$2$$

Differenciálás V.

A `D` operátor az ismert deriválási szabályok szerint kezeli a függvényeket:

```
restart;
'D(f+g)' = D(f+g); 'D(f+g)(z)' = D(f+g)(z);
'D(f*g)' = D(f*g);'D(f*g)(z)' = D(f*g)(z);
'D(f/g)' = D(f/g);'D(f/g)(z)' = D(f/g)(z);
'D(f@g)' = D(f@g);'D(f@g)(z)' = D(f@g)(z);
```

$$D(f+g) = \mathsf{D}(f) + D(g)$$
$$D(f+g)(z) = D(f)(z) + D(g)(z)$$
$$D(f*g) = D(f)\,g + f D(g)$$
$$D(f*g)(z) = D(f)(z)\,g(z) + f(z)\,D(g)(z)$$
$$D\left(\frac{f}{g}\right) = \frac{D(f)}{g} - \frac{f D(g)}{g^2}$$
$$D\left(\frac{f}{g}\right)(z) = \frac{D(f)(z)}{g(z)} - \frac{f(z)\,D(g)(z)}{(g(z))^2}$$
$$D(f@g) = (D(f)@g)\,D(g)$$
$$D(f@g)(z) = D(f)(g(z))\,D(g)(z)$$

Differenciálás VI.

Többszörös deriválás esetén a `diff` ismételt rekurzív hívása helyett alkalmazhatunk egyszerűbb jelölést is: a `diff` második argumentuma lehet névsorozat vagy nevek listája, s így adhatjuk meg, hogy hányszor és mi szerint történjen a deriválás.
Az egyváltozós függvények magasabb rendű deriváltjainak, illetve a többváltozós függvények különböző parciális deriváltjainak kiszámítása hasonló szintaxissal történik, kivéve, hogy egy változó esetén a `diff` megfelelő argumentumában mindig csak ugyanaz a változó szerepelhet.

```
Diff(cos(x^2), x,x,x) = diff(diff(diff(cos(x^2), x),x),x);
Diff(cos(x^2), x,x,x) = diff(cos(x^2), x,x,x);
Diff(cos(x^2), x,x,x) = diff(cos(x^2), [x,x,x]);
```

$$\frac{d^3}{dx^3}\cos\left(x^2\right) = 8\,\sin\left(x^2\right)x^3 - 12\,\cos\left(x^2\right)x$$
$$\frac{d^3}{dx^3}\cos\left(x^2\right) = 8\,\sin\left(x^2\right)x^3 - 12\,\cos\left(x^2\right)x$$
$$\frac{d^3}{dx^3}\cos\left(x^2\right) = 8\,\sin\left(x^2\right)x^3 - 12\,\cos\left(x^2\right)x$$

```
fexp := x^2*sin(x*y);
Diff(fexp, x,y,y) = diff(diff(diff(fexp, x),y),y);
Diff(fexp, x,y,y) = diff(fexp, x,y,y);
Diff(fexp, x,y,y) = diff(fexp, [x,y,y]);
```

$$fexp := x^2\sin(xy)$$
$$\frac{\partial^3}{\partial y^2\partial x}\left(x^2\sin\left(xy\right)\right) = -4\,x^3\sin\left(xy\right) - x^4\cos\left(xy\right)y$$
$$\frac{\partial^3}{\partial y^2\partial x}\left(x^2\sin\left(xy\right)\right) = -4\,x^3\sin\left(xy\right) - x^4\cos\left(xy\right)y$$
$$\frac{\partial^3}{\partial y^2\partial x}\left(x^2\sin\left(xy\right)\right) = -4\,x^3\sin\left(xy\right) - x^4\cos\left(xy\right)y$$

Differenciálás VII.

Függvények esetében a `D` operátor használatakor a `D` utáni, a változók sorszámait tartalmazó listában adhatjuk meg a szükséges deriválásokat. Az egyváltozós függvények magasabb rendű deriváltjainak, illetve a többváltozós függvények különböző parciális deriváltjainak kiszámítása hasonló szintaxissal történik, kivéve, hogy egy változó esetén a `D` megfelelő argumentumában mindig csak az 1 sorszám szerepelhet, illetve az 1 el is hagyható.

```
f := x -> cos(x^2);
Diff(f, x,x,x) = D[1](D[1](D[1](f)));
Diff(f, x,x,x) = D(D(D(f)));
Diff(f, x,x,x) = (D@@3)(f);
Diff(f, x,x,x) = D[1,1,1](f);
```

$$f := x -> \cos(x^2)$$
$$\frac{\partial^3}{\partial x^3} f = (x \to 8 \sin\left(x^2\right) x^3 - 12 \cos\left(x^2\right) x)$$
$$\frac{\partial^3}{\partial x^3} f = (x \to 8 \sin\left(x^2\right) x^3 - 12 \cos\left(x^2\right) x)$$
$$\frac{\partial^3}{\partial x^3} f = (x \to 8 \sin\left(x^2\right) x^3 - 12 \cos\left(x^2\right) x)$$
$$\frac{\partial^3}{\partial x^3} f = (x \to 8 \sin\left(x^2\right) x^3 - 12 \cos\left(x^2\right) x)$$

```
ffun := (x,y) -> x^2*sin(x*y);
Diff(ffun, x,y,y) = D[2](D[2](D[1](ffun)));
Diff(ffun, x,y,y) = D[1,2,2](ffun);
```

$$ffun := (x, y) \to x^2 \sin(xy)$$
$$\frac{\partial^3}{\partial y^2 \partial x} ffun = ((x, y) \to -4\, x^3 \sin(xy) - x^4 \cos(xy)\, y)$$
$$\frac{\partial^3}{\partial y^2 \partial x} ffun = ((x, y) \to -4\, x^3 \sin(xy) - x^4 \cos(xy)\, y)$$

Differenciálás VIII.

Listában vagy halmazban megadott kifejezéseket differenciáltathatunk egyetlen `diff` paranccsal:

```
Flist := [x^2 + 2*y^2,  exp(y^2)*sin(x)];
Fset := {x^2 + 2*y^2,  exp(y^2)*sin(x)};
```

$$Flist := \left[x^2 + 2\,y^2, \mathrm{e}^{y^2}\sin\left(x\right)\right]$$
$$Fset := \left\{\mathrm{e}^{y^2}\sin\left(x\right), x^2 + 2\,y^2\right\}$$

```
diff(Flist, x );
diff(Fset, x );
```

$$\left[2\,x, \mathrm{e}^{y^2}\cos\left(x\right)\right]$$
$$\left\{\mathrm{e}^{y^2}\cos\left(x\right), 2\,x\right\}$$

```
diff(Flist, [x,y] );
diff(Fset, [x,y] );
```

$$\left[0, 2\,y\mathrm{e}^{y^2}\cos\left(x\right)\right]$$
$$\left\{0, 2\,y\mathrm{e}^{y^2}\cos\left(x\right)\right\}$$

Differenciálás IX.

Az előzőek alapján előállíthatjuk az `Flist` listában megadott függvényrendszerhez tartozó Jacobi mátrixot.
Erre van egy `Jacobian` nevű beépített eljárás is a `VectorCalculus` csomagban.
Lásd `?VectorCalculus,Jacobian`.

```
JacobiM1 := Matrix(
   [[diff(Flist[1], x),diff(Flist[1], y)],[diff(Flist[2],
x),diff(Flist[2], y)]] );
JacobiM2 := VectorCalculus[Jacobian](Flist,[x,y]);
```

$$JacobiM1 := \begin{bmatrix} 2x & 4y \\ e^{y^2}\cos(x) & 2ye^{y^2}\sin(x) \end{bmatrix}$$

$$JacobiM2 := \begin{bmatrix} 2x & 4y \\ e^{y^2}\cos(x) & 2ye^{y^2}\sin(x) \end{bmatrix}$$

Differenciálás X.

A $g(x,y)$ kétváltozós függvény Hesse mátrixa is könnyen felírható.

```
HesseM := Matrix(
   [[Diff(g, x,x),Diff(g, x,y)],[Diff(g, y,x),Diff(g, y,y)]] );
g := sin(x)*y^2;
Hg = map(value, HesseM);
```

$$HesseM := \begin{bmatrix} \frac{\partial^2}{\partial x^2} g & \frac{\partial^2}{\partial y \partial x} g \\ \frac{\partial^2}{\partial x \partial y} g & \frac{\partial^2}{\partial y^2} g \end{bmatrix}$$

$$g := \sin(x)\, y^2$$

$$Hg = \begin{bmatrix} -\sin(x)\, y^2 & 2\cos(x)\, y \\ 2\cos(x)\, y & 2\sin(x) \end{bmatrix}$$

Ugyanezt számolja a VectorCalculus csomag Hessian eljárása:

```
H := VectorCalculus[Hessian](g, [x,y]);
```

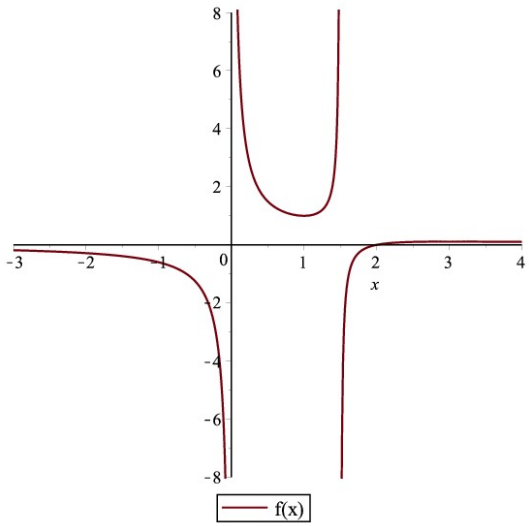
$$\begin{bmatrix} -\sin(x)\, y^2 & 2\cos(x)\, y \\ 2\cos(x)\, y & 2\sin(x) \end{bmatrix}$$

Függvénydiszkusszió I.

Eddigi ismereteinket felhasználva a Maple-lel teljes függvénydiszkussziót végeztethetünk. Megállapíthatjuk az $f(x)$ függvény zérushelyeit, határértékét a $\pm\infty$-ben, szakadási-, szélsőérték- és inflexiós helyeit.
Nézzünk erre egy példát, az alábbi racionális törtfüggvény vizsgálatát. Ábrázoljuk is függvényünket.

```
f:=x->(x-2)/(2*x^2-3*x);
plot(f(x),x= -3..4, discont=true, legend ="f(x)");
```

$$f := x \to \frac{x-2}{2x^2-3x}$$



Függvénydiszkusszió II.

Páros vagy páratlan függvény-e f(x)?

```
evalb(f(x)=f(-x));
evalb(f(x)=-f(-x));
```

$$false$$
$$false$$

Mik a zérushelyei?

```
solve(f(x));
```

$$2$$

Mik a határértékei a +-végtelenben:

```
Limit(f(x),x=infinity)=limit(f(x),x=infinity);
Limit(f(x),x=-infinity)=limit(f(x),x=-infinity);
```

$$\lim_{x\to\infty} \frac{x-2}{2x^2-3x} = 0$$
$$\lim_{x\to-\infty} \frac{x-2}{2x^2-3x} = 0$$

Függvénydiszkusszió III.

Szakadási helyei ott lehetnek (és ténylegesen vannak is), ahol a nevező 0. Ugyanezt mondja a `discont` Maple eljárás is, lásd `?discont`.

```
S := solve({denom(f(x))} );
T := discont(f(x), x);
```

$$S := \{x = 0\}, \left\{x = \frac{3}{2}\right\}$$

$$T := \left\{0, \frac{3}{2}\right\}$$

A szakadási helyeken a bal- és jobboldali határértéke:

```
Limit(f(x),x=0,left)=limit(f(x),x=0,left);
Limit(f(x),x=0,right)=limit(f(x),x=0,right);
Limit(f(x),x=3/2,left)=limit(f(x),x=3/2,left);
Limit(f(x),x=3/2,right)=limit(f(x),x=3/2,right);
```

$$\lim_{x\to 0^-} \frac{x-2}{2x^2-3x} = -\infty$$

$$\lim_{x\to 0^+} \frac{x-2}{2x^2-3x} = \infty$$

$$\lim_{x\to 3/2^-} \frac{x-2}{2x^2-3x} = \infty$$

$$\lim_{x\to 3/2^+} \frac{x-2}{2x^2-3x} = -\infty$$

Függvénydiszkusszió IV.

A kritikus pontok meghatározásához először képezzük $f(x)$ deriváltfüggvényét, majd keressük meg ennek zérushelyeit.
A derivált viselkedéséből látható, hogy $x = 1$ lokális minimum, $x = 3$ pedig lokális maximumhely.

```
fd := D(f);
K := solve(fd(x), {x});
solve(fd(x)>0,{x});
solve(fd(x)<0,{x});
```

$$fd := x \to \frac{1}{2x^2 - 3x} - \frac{(-2+x)(4x-3)}{(2x^2-3x)^2}$$

$$K := \{x = 1\}, \{x = 3\}$$

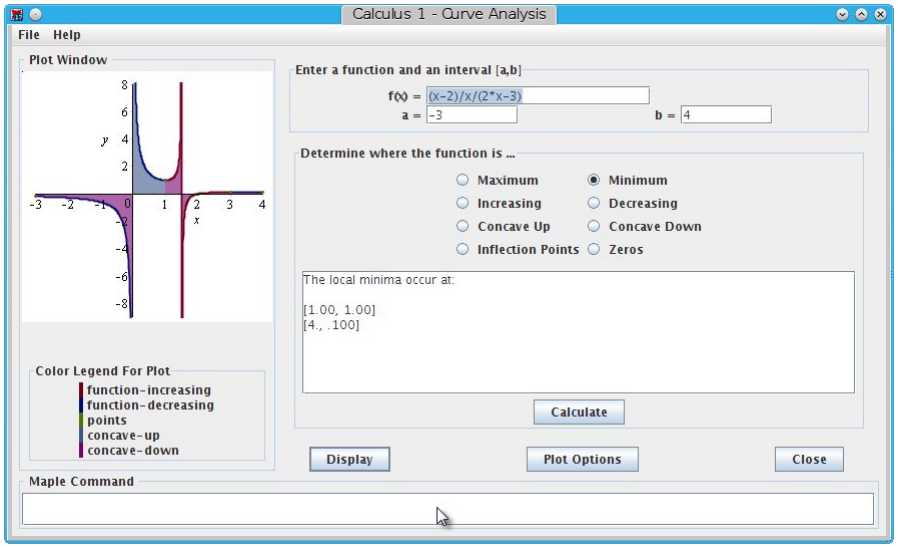
$$\{1 < x, x < 3/2\}, \{3/2 < x, x < 3\}$$

$$\{x < 0\}, \{0 < x, x < 1\}, \{3 < x\}$$

Függvénydiszkusszió V.

Az előző feladat megoldható a `Student` csomag `Calculus1` alcsomagjának `CurveAnalysisTutor` eljárásával, amely meghatározza a függvény zérushelyeit, szélsőértékeit, inflexiós pontjait, növekedési és csökkenési intervallumait, stb.
Paraméterek nélküli hívásakor rögtön elindul a mapletes grafikus felület, és betöltődik az alapértelmezett $f(x) = x\cos(x)$ függvény grafikonja a $[-2\pi, 2\pi]$ intervallumon. Mi is megadhatjuk a hívás első paraméterként a vizsgálandó függvény képletét (tehát **kifejezést**), másodikként pedig az intervallumot `CurveAnalysisTutor(f(x), x=a..b)` alakban.
Alkalmazzuk az eljárást előző feladatunkra! A megfelelő maplet ablak az ábrán látható.

```
with(Student[Calculus1]):
CurveAnalysisTutor((x-2)/(2*x^2-3*x), x=-3..4);
```



Integrálás I

Ebben az alfejezetben két feladatot vizsgálunk:

- határozatlan integrálok (primitív függvények) meghatározása
- határozott integrálok kiszámítása

Ezek az `int` eljárás, illetve nagybetűs (inert) `Int` változatának használatával oldhatók meg.
A Maple képes többváltozós függvények integráljainak meghatározására, mi csak az egyváltozós esettel foglalkozunk.
A korábbiakhoz hasonlóan, bár *függvényekről* beszélünk, legtöbbször a kényelmesebb szintaktika miatt Maple *kifejezésekkel* dolgozunk. Néha utalunk rá, hogy a számítások hogyan végezhetők el „igazi” Maple függvényekkel.

Integrálás II

A Maple integrálással kapcsolatos eljárásai az `int` és ennek inert változata, az `Int`. Paramétereik megadása a `diff`-hez hasonló. Lásd `?int`.
Határozatlan integrál (primitív függvény) kiszámításakor a Maple nem írja ki a C integrációs konstanst, vagy másképpen a $C = 0$ értéket veszi. A primitív függvényre kapott eredmény helyességét differenciálással ellenőrizhetjük.

```
Int(sin(x),x)=int(sin(x),x);
diff(%,x);
```

$$\int \sin(x)\,dx = -\cos(x)$$
$$\sin(x) = \sin(x)$$

Az `Int`hez hasonló inert függvények kiértékelését a `value` eljárással végezhetjük. Lásd `?value`.

```
Int( x/(x^3-1), x ) = value(Int( x/(x^3-1), x ));
```

$$\int \frac{x}{x^3-1}\,dx = -\frac{1}{6}\ln\left(x^2+x+1\right) + \frac{1}{3}\sqrt{3}\arctan\left(\frac{1}{3}(2x+1)\sqrt{3}\right) + \frac{1}{3}\ln(x-1)$$

A Maple ismeri a primitív függvényre vonatkozó azonosságot:

```
Int(D(f)(x),x)=int(D(f)(x),x);
```

$$\int \mathrm{D}(f)(x)\,dx = f(x)$$

A következő primitív függvényben szereplő `erf(x)`, a Gauss féle hibafüggvény nem elemi függvény, de a Maple ismeri, tud vele számolni.

```
Int(exp(-x^2),x)=int(exp(-x^2),x);
```

$$\int e^{-x^2}\,dx = 1/2\,\sqrt{\pi}\,\mathrm{erf}(x)$$

Integrálás III

Határozott integrálok kiszámításánál második paraméterként a határokat kell tartományként megadni **változónév=alsóhatár..felsőhatár** formában. Szemléltetésül itt is használhatjuk az `Int` inert alakot.

```
Int(x*sin(x), x= 0..Pi) = int(x*sin(x), x= 0..Pi);
```

$$\int_0^{\pi} x \sin(x)\, dx = \pi$$

```
Int(x*sin(x), x= 0..Pi) = value( Int(x*sin(x), x= 0..Pi) );
```

$$\int_0^{\pi} x \sin(x)\, dx = \pi$$

Integrálás IV

Eddig kifejezésként adtuk meg az integrálandó függvényt, a továbbiakban is többnyire így teszünk, bár lehetne „igazi függvényeket" is használni. Figyeljük meg a szintakszist!

```
f := x ->x*sin(x);
```

$$f := x \to x \sin(x)$$

Így csúnya az `Int` formátuma, de az eredmény korrekt:

```
Int(f,0..Pi) = int(f,0..Pi);
value(%);
```

$$Int(f, 0..\pi) = \pi$$
$$\pi = \pi$$

Így már az `Int` is jobban néz ki, függvények esetében ez a legtermészetesebb megoldás:

```
Int(f(x),x=0..Pi) = int(f(x), x=0..Pi);
```

$$\int_0^{\pi} x \sin(x)\, dx = \pi$$

Integrálás V

A Maple jól kezeli a különböző típusú (véges vagy végtelen intervallumon vett) improprius integrálokat

```
Int(1/x^2,x=0..1)=int(1/x^2,x=0..1),
Int(1/sqrt(x),x=0..1)=int(1/sqrt(x),x=0..1),
Int(1/sqrt(1-x^2),x=-1..1)=int(1/sqrt(1-x^2),x=-1..1);
```

$$\int_0^1 x^{-2}dx = \infty, \int_0^1 \frac{1}{\sqrt{x}}dx = 2, \int_{-1}^1 \frac{1}{\sqrt{-x^2+1}}dx = \pi$$

```
Int(1/x^2,x=1..infinity)=int(1/x^2,x=1..infinity),
Int(exp(-x^2),x=-infinity..infinity)=int(exp(-x^2),x=-infinity..infinity)
```

$$\int_1^\infty x^{-2}dx = 1, \int_{-\infty}^\infty \mathrm{e}^{-x^2}dx = \sqrt{\pi}$$

Integrálás VI

Előfordulhat, hogy a Maple nem tudja meghatározni a primitív függvényt. Ilyenkor az `Int`-hez hasonlóan a kiértékeletlen eljáráshívást adja vissza.

```
f := x -> (1-2*ln(x))/ln(x)^(3/2);
int(f(x),x);
```

$$f := x \to \frac{1 - 2\ln(x)}{(\ln(x))^{3/2}}$$

$$\int \frac{1 - 2\ln(x)}{(\ln(x))^{3/2}} dx$$

Holott f(x) primitív függvénye létezik, az alábbi $F(x)$-re teljesül $f(x) = F'(x)$.
Érdekes, hogy $f(x)$ határozott integráljának kiszámítása már nem okoz problémát.

```
F:= x -> -2*x/ln(x)^(1/2);
f(x) - D(F)(x);
simplify(%);
Int(f(x),x=2..4) = int(f(x),x=2..4);
```

$$F := x \to -2\frac{x}{\sqrt{\ln(x)}}$$

$$\frac{1 - 2\ln(x)}{(\ln(x))^{3/2}} + 2\frac{1}{\sqrt{\ln(x)}} - (\ln(x))^{-3/2}$$

$$0$$

$$\int_2^4 \frac{1 - 2\ln(x)}{(\ln(x))^{3/2}} dx = -4\frac{\sqrt{2} - 1}{\sqrt{\ln(2)}}$$

Integrálás VII

Határozott integrálok meghatározásához, például ha nem ismert a primitív függvény, szükség lehet numerikus (közelítő) integrálási módszerekre is.
Numerikus integrálást az `evalf(Int(...))` vagy `int(..., numeric = true, ...)` alakú hívásokkal kérhetünk a Maple-től. Lásd `?int,numeric`.
Ilyenkor az alkalmazandó numerikus módszert is megadhatjuk a `method = ...` opcióval, illetve a kért pontosságot az `epsilon = ...` opcióval. (Ha a Maple nem tud pontosan kiintegrálni valamit, akkor az `evalf(int(...))` konstrukció is ugyanazt adja, mint az előzőek.)

```
evalf( Int(x*sin(x), x= 0..Pi, epsilon = 10^(-8) ));
int(x*sin(x), x= 0..Pi, numeric = true, epsilon = 10^(-8));
```

$$3.141592654$$
$$3.141592654$$

Ez vajon most mit jelent?

```
evalf( int(x*sin(x), x= 0..Pi) );
```

$$3.141592654$$

Numerikusan „bármilyen bonyolult" függvényeket is tud integrálni a Maple, még ha nem is ismeri a primitív függvényt:

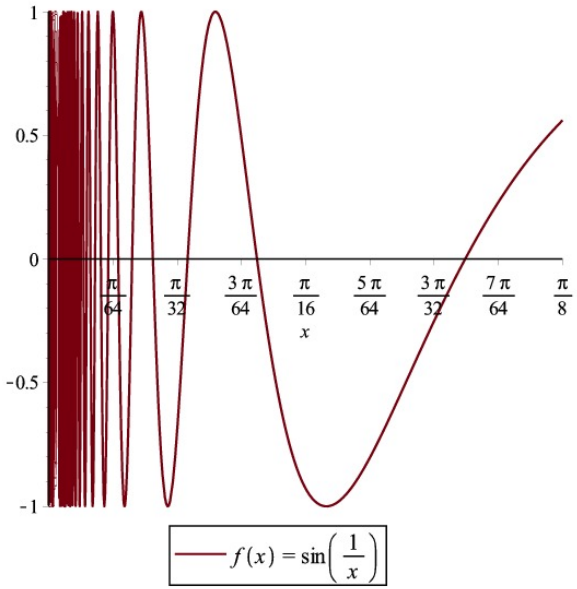
```
Int( exp(-x^2)*ln(x), x) = int( exp(-x^2)*ln(x), x);
Int( exp(-x^2)*ln(x), x=1..2) = evalf(Int( exp(-x^2)*ln(x), x=1..2) );
```

$$\int e^{-x^2} \ln(x)\, dx = \int e^{-x^2} \ln(x)\, dx$$
$$\int_1^2 e^{-x^2} \ln(x)\, dx = 0.03261204236$$

Integrálás VIII

Lássunk egy „nehezebb" függvényt is:

```
plot(sin(1/x), x= 0..Pi/8, legend=['f(x)'=sin(1/x)]);
```



Ennek határozatlan integrálját pontosan is meg tudja adni a Maple, az eredményben látható $Ci(x)$ nem elemi függvény definíciója: $Ci(x) = \gamma + \ln(x) + \int_{t=0}^{x} \frac{cos(t)-1}{t} dt$

```
Int(sin(1/x), x) =int(sin(1/x), x);
```

$$\int \sin\left(\frac{1}{x}\right) dx = \sin\left(\frac{1}{x}\right) x - Ci\left(\frac{1}{x}\right)$$

Integrálás IX

A határozott integrál pontos értéke:

```
Int(sin(1/x), x= 0..Pi/2) = int(sin(1/x), x= 0..Pi/2);
```

$$\int_0^{1/2\,\pi} \sin\left(\frac{1}{x}\right) dx = \pi\,\sin\left(\frac{1}{\pi}\right)\cos\left(\frac{1}{\pi}\right) - Ci\left(\frac{2}{\pi}\right)$$

Ami lebegőpontosan kb. ennyi:

```
evalf(rhs(%),10);
```

$$0.9078007610$$

Próbáljuk numerikusan közelíttetni az előző integrált 10 tizedes jegy pontossággal. Az eredmény megegyezik a pontos megoldás előbbi kerekített értékével.

```
Digits := 11;
int(sin(1/x), x= 0..Pi/2, numeric = true, epsilon = 10^(-10));
```

$$11$$
$$0.90780076100$$

Integrálás X

Az integrálszámítás egyik hasznos alkalmazása az $y = f(x)$ függvény görbéjének X (vagy Y) tengely körüli elforgatásával kapott forgástest térfogatának meghatározása. Az X tengely körül elforgatva a görbe $x = a$ és $x = b$ közti ívét az ismert képlet szerint $V = \int_a^b \pi f(x)^2\,dx$ lesz.

Ezt megkaphatjuk az `int` hívásával, de a `Student[Calculus1]` csomag `VolumeOfRevolution` eljárása ennél többet is nyújt. Az `output` opciónál például kérhetjük csak a végeredményt (ez az alapértelmezett), a kiszámolandó integrált (`output=integral`) vagy a kapott forgástest 3D ábrázolását (`output=plot`).
Lásd `?Student[Calculus1][VolumeOfRevolution]`. Az alábbiakban az $y = sin(x)$ függvény görbéjével számolunk.

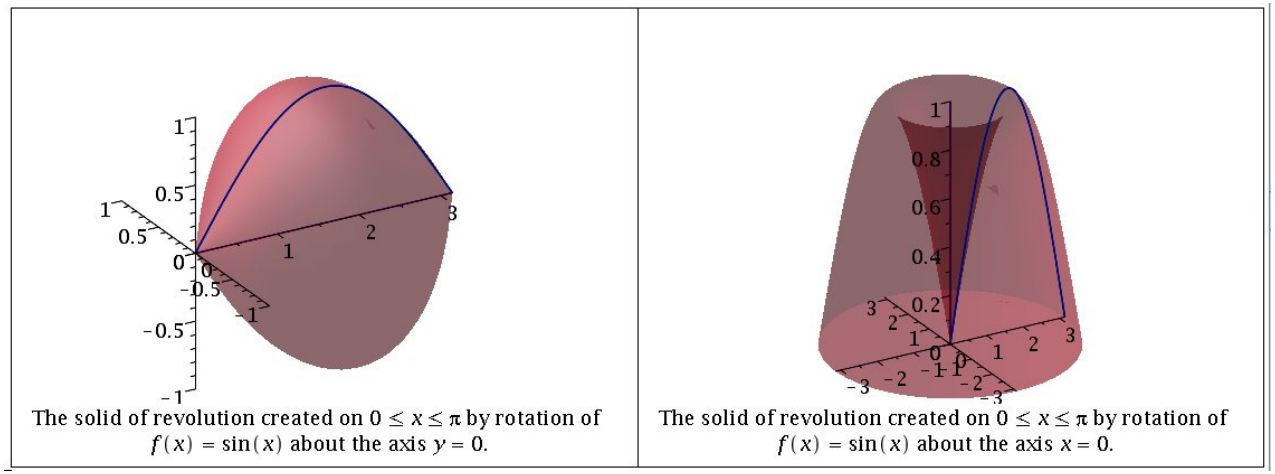
```
with(Student[Calculus1]):
f:= x -> sin(x);
V1 := VolumeOfRevolution(f(x), x=0..Pi);
V2 := VolumeOfRevolution(f(x), x=0..Pi,axis = vertical);
V1 := VolumeOfRevolution(f(x), x=0..Pi, output=integral);
V2 := VolumeOfRevolution(f(x), x=0..Pi, output=integral, axis = vertical);
```

$$f := x \to \sin(x)$$
$$V1 := 1/2\,\pi^2$$
$$V2 := 2\,\pi^2$$
$$V1 := \int_0^{\pi} \pi\,(\sin(x))^2 dx$$
$$V2 := \int_0^{\pi} 2\,\pi\,x\sin(x)\,dx$$

Integrálás XI

Az előzőekben leírt feladatnál az X és az Y tengely körüli elforgatással kapott két forgástestet mutatja az alábbi ábra.

```
Plotok := Vector[row](2):
Plotok[1] := VolumeOfRevolution(f(x), x=0..Pi, output=plot,
   font=[Times, 16]):
Plotok[2] := VolumeOfRevolution(f(x), x=0..Pi, output=plot,
   axis = vertical, font=[Times, 16]):
plots:-display(Plotok);
```



Integrálás XII

Az integrálszámítás további fontos gyakorlati alkalmazása: határozzuk meg az $y = f(x)$ görbe $x = a$ és $x = b$ közti darabjának ívhosszát. Az ismert képlet szerint $L = \int_a^b \sqrt{(f'(x))^2 + 1}\, dx$. Ez kiszámolható az `int` eljárás hívásával, de a `Student[Calculus1]` csomag `ArcLength` eljárása még kényelmesebb megoldás. Lásd `?Student[Calculus1][ArcLength]`.
Határozzuk meg ezzel is az egységkör félkerületét, vagyis az $y = \sqrt{1 - x^2}$ függvény ívhosszát a $[-1, 1]$ intervallumon.

```
with(Student[Calculus1]):
f := x -> sqrt(1-x^2); fder := D(f);
```

$$f := x \to \sqrt{1 - x^2}$$
$$fder := x \to -\frac{x}{\sqrt{-x^2 + 1}}$$

```
int(sqrt(fder(x)^2+1), x=-1..1);
ArcLength(f(x), x=-1..1);
ArcLength(f(x), x=-1..1, output=integral);
```

$$\pi$$
$$\pi$$
$$\int_{-1}^{1} \frac{1}{\sqrt{-x^2 + 1}}\, dx$$

Optimalizálás I

Optimalizálási feladatok megoldására (egy- és többváltozós függvények globális, lokális vagy feltételes szélsőértékeinek meghatározására) a Maple több beépített eljárást és csomagot is ajánl. Ezeket tekintjük át röviden a továbbiakban.
Bár most is *függvények* szélsőértékeiről beszéltünk, sokszor kényelmesebb, ha nem függvényekkel, hanem kifejezésekkel (ha úgy tetszik, a függvényeinket definiáló kifejezésekkel) dolgozunk. Ügyeljünk a szintaktikus eltérésekre!

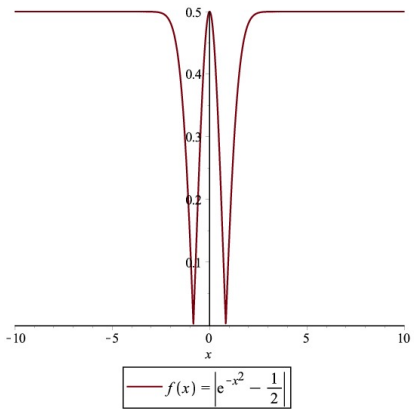
Optimalizálás II

A `minimize` és `maximize` eljárások egy- vagy többváltozós függvények globális szélsőértékeit határozzák meg. A `maximize(f, ...)` hívás lényegében a `minimize(-f, ...)` hívással egyenértékű. (Lásd `?minimize`.) Figyeljük meg, hogy ha `f`-et függvényként definiáljuk, másképpen kell megadni az első argumentumot. A `location` opció használatakor a minimum értéke mellett a minimumhely(ek)et is visszaadja az eljárás a példán látható listás-halmazos formátumban.

```
f1 := abs(exp(-x^2)-1/2); # f1 kifejezés!
f1func := x -> abs(exp(-x^2)-1/2); # f1func függvény!
plot(f1, x= -10..10,numpoints=1000,legend=['f(x)' = f1]);
minimize(f1); minimize(f1func(x));
minimize(f1, location);
```

$$f1 := \left|e^{-x^2} - 1/2\right|$$

$$f1func := x \to \left|e^{-x^2} - 1/2\right|$$



$$0$$

$$0$$

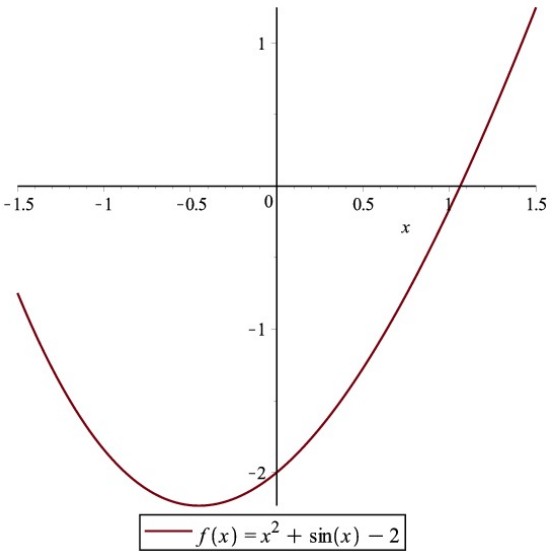
$$0, \left\{\left[\left\{x = \sqrt{\ln(2)}\right\}, 0\right], \left[\left\{x = -\sqrt{\ln(2)}\right\}, 0\right]\right\}$$

Optimalizálás III

Az alábbi `f2` függvénnyel már nem boldogul a `minimize`: nem találja meg az (egyetlen, létező) globális minimumot. Ilyenkor az output a kiértékeletlen eljáráshívásból áll.

```
f2 := x^2+sin(x)-2;
plot(f2,x=-3/2..3/2,legend=['f(x)' = f2]);
```

$$x^2 + \sin(x) - 2$$




```
> minimize(f2);
```

$$minimize(x^2 + \sin(x) - 2)$$

Optimalizálás IV

Ha azonban csak egy véges tartományon kerestetjük a minimumot, itt az `x = -1..0` opcióval, már jobban boldogul a Maple. Az első parancs `RootOf`-os eredménye nem sokat mond, ezért ki sem írattuk, de az `evalf` eredményéből látszik a megoldás helyessége.

```
minimize(x^2-2+sin(x),x=-1..0,location):
evalf(%);
```

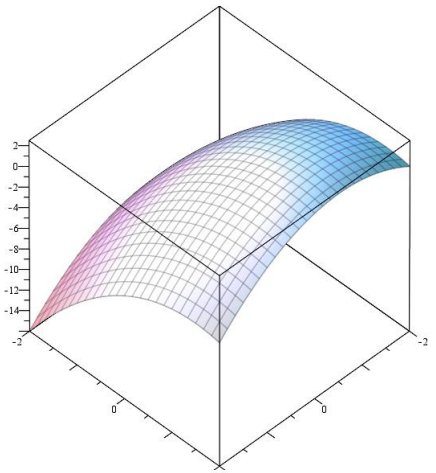
$$-2.232465575, [x = -.4501836113, -2.232465575]$$

Optimalizálás V

A `maximize` eljárás hasonlóan alkalmazható – itt egy kétváltozós példát mutatunk be:

```
f3:= -x^2-x -y^2+3*y;
plot3d(f3, x=-2..2,y=-2..2);
Maximum := maximize(f3, location);
```

$$f3 := -x^2 - x - y^2 + 3 * y;$$



$$Maximum := 5/2, \{[\{x = -1/2, y = 3/2\}, 5/2]\}$$

Optimalizálás VI

Az előző példákból sejthető, hogy a `minimize` vagy a `maximize` használata sokszor nem vezet eredményre. Ilyenkor próbálkozhatunk a beépített `Optimization` csomag eljárásaival is. Lásd `?Optimization`. Ez a csomag kizárólag *lokális* szélsőértékkereső *numerikus* módszereket használ, ezért globális szélsőértékek meghatározására nem alkalmas (hacsak nincs valami előzetes információnk az illető függvény lokális és globális szélsőértékeinek kapcsolatáról). Az eljárások feltételes szélsőértékfeladatokat is kezelnek, a feltételek megadásának szintaktikájára a Helpben találunk példákat. Az `infolevel[Optimization]` értékét megnövelve részletesen követhető az eljárások működése. A csomag fontosabb eljárásai:

`Minimize`	minimumkereső eljárás
`Maximize`	maximumkereső eljárás
`NLPSolve`	nemlineáris optimalizációs feladatok megoldása
`QPSolve`	kvadratikus optimalizációs feladatok megoldása
`LPSolve`	lineáris programozási feladatok megoldása
`Interactive`	interaktív grafikus felület (maplet) optimalizálási feladatokhoz

```
with(Optimization);
infolevel[Optimization] := 10;
```

$$[ImportMPS, Interactive, LPSolve, LSSolve, Maximize, Minimize, NLPSolve, QPSolve]$$
$$infolevel_{Optimization} := 10$$

Optimalizálás VII

Oldjuk meg a `Maximize` eljárással is az előző feladatot. A vizsgált kifejezés kvadratikus, ezért a `QPSolve` számolja a közelítéseket. A numerikus algoritmusok miatt mindig lebegőpontos eredményeket kapunk.

```
f3:= -x^2-x -y^2+3*y;
Maximize(f3);
```

$$f3 := -x^2 - x - y^2 + 3y$$

```
QPSolve: calling QP solver
QPSolve: number of problem variables 2
QPSolve: number of general linear constraints 0
QPSolve: feasibility tolerance set to 0.1053671213e-7
QPSolve: iteration limit set to 50
QPSolve: infinite bound set to 0.10e21
QPSolve: number of iterations taken 1
QPSolve: final value of objective -2.49999999999999956
```

$$[2.50000000000000, [x = -.500000000000000, y = 1.50000000000000]]$$

Optimalizálás VIII

A függvényes alakkal is dolgozhatunk. Érdekes, hogy ekkor a `Maximize` az `NLPSolve` eljárást hívja. Az eredmény más formátumú, de ekvivalens a korábban kapottakkal.

```
f3func := (x,y) -> -x^2-x -y^2+3*y;
Maximize(f3func);
```

$$f3func := (x, y) -> -x^2 - x - y^2 + 3y$$

```
NLPSolve: calling NLP solver
NLPSolve: using method=pcg
NLPSolve: number of problem variables 2
NLPSolve: optimality tolerance set to 0.3256082241e-11
NLPSolve: iteration limit set to 50
NLPSolve: trying evalhf mode
NLPSolve: trying evalf mode
attemptsolution: number of major iterations taken 2
```

$$\left[2.50000000000000000, \left[\begin{array}{c} -0.500000000000000000 \\ 1.500000000000000000 \end{array}\right]\right]$$

Optimalizálás IX

Utolsó feltételes szélsőértékfeladatunkban az egységkörön keressük a maximumot, az iterációt a megadott kezdőpontból indítjuk.

```
f4 := sin(x) + sin(y);
Maximize(f4, {x^2+y^2 = 1}, initialpoint=[x=0.7,y=0.7]);
```

$$f4 := \sin(x) + \sin(y)$$

```
NLPSolve: calling NLP solver
NLPSolve: using method=sqp
NLPSolve: number of problem variables 2
NLPSolve: number of nonlinear inequality constraints 0
NLPSolve: number of nonlinear equality constraints 1
NLPSolve: number of general linear constraints 0
NLPSolve: feasibility tolerance set to 0.1053671213e-7
NLPSolve: optimality tolerance set to 0.3256082241e-11
NLPSolve: iteration limit set to 50
NLPSolve: infinite bound set to 0.10e21
NLPSolve: trying evalhf mode
attemptsolution: number of major iterations taken 3
```

$$[1.29927387816012496, [x = .707106781186548, y = .707106781186548]]$$

```
# egy kis Maple Magic - ez a pontos szélsőértékhely koordinátája:
identify(0.707106781186548);
```

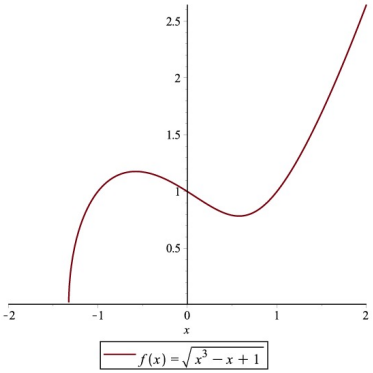
$$\frac{1}{2}\sqrt{2}$$

Optimalizálás X

Algebrai függvények feltételes szélsőértékeit az `extrema` eljárással is meghatározhatjuk. Az eljárás a Lagrange-multiplikátoros módszerrel dolgozik. Lásd `?extrema`.

```
f := sqrt(x^3-x+1);  plot(f, x = -2..2,legend=['f(x)' = f]);
```

$$f := \sqrt{x^3 - x + 1}$$



Az alábbi példában az `abs(x)=x` feltétel valójában az `x>0` helyett áll, mivel az utóbbi szintaktika nem megengedett.
Az `mhely` változóban a minimumhely(ek)et, az `mertek` változóban a minimum értékét kapjuk.

```
mertek := extrema(f,{abs(x)=x},x,'mhely'):
'minhely' = op(1,mhely), minertek = evalf(mertek);
```

$$minhely = \left\{x = \frac{1}{3}\sqrt{3}\right\}, minertek = \{.7842829976\}$$

Differenciálegyenletek I

Differenciálegyenletek megoldására a Maple a beépített `dsolve` eljárást kínálja. Ezzel mind egzaktul (szimbolikusan), mind numerikusan megoldható a differenciálegyenletekkel kapcsolatos számos feladat.
Alkalmas első vagy magasabb rendű közönséges és parciális differenciálegyenletek, sőt egyenletrendszerek megoldására is.
Ebben az előadásban csak közönséges differenciálegyenleteket vizsgálunk, azok közül is főleg első rendűeket.

Differenciálegyenletek II

Ha differenciálegyenletünkben az ismeretlen függvényt `y(x)` jelöli (`x` a független változó), akkor `y(x)` deriváltjait a `diff` függvénnyel vagy a `D` operátorral képezhetjük.
A változót mindig ki kell írni, tehát mindig az `y(x)` teljes alakot kell használni. Az alábbi két forma ugyanazt az elsőrendű közönséges differenciálegyenletet definiálja.

```
de1a := diff(y(x),x)-sin(x)*y(x)=sin(x);
de1b := D(y)(x)-sin(x)*y(x)=sin(x);
```

$$de1a := \frac{d}{dx}y(x) - \sin(x)\,y(x) = \sin(x)$$
$$de1b := \mathrm{D}(y)(x) - \sin(x)\,y(x) = \sin(x)$$

Másodrendű egyenleteknél hasonló jelöléseket használhatunk.

```
de2a := diff(y(x),x,x)-k*y(x) = 0;
de2b := (D@@2)(y)(x)-k*y(x)=0;
```

$$de2a := \frac{d^2}{dx^2}y(x) - k\,y(x) = 0$$
$$de2b := D^{(2)}(y)(x) - k\,y(x) = 0$$

Differenciálegyenletek III

Ha az előző egyenleteket a `dsolve`-val akarjuk megoldatni, akkor elegendő első paraméterként a differenciálegyenletet, másodikként pedig az ismeretlen függvényt (most `y(x)`-et) megadni. Lásd `?dsolve`. Így az egyenletek (pontos,egzakt) általános megoldását kapjuk, amely elsőrendű egyenleteknél egy, másodrendűeknél két tetszőlegesen választható integrációs konstanst tartalmaz. A Maple ezeket _$C1$-gyel, illetve _$C2$-vel jelöli. Az outputból látszik, hogy a `diff`, illetve a D operátoros jelölés valóban ekvivalens egyenleteket adott meg.

```
dsolve(de1a,y(x));
dsolve(de1b,y(x));
dsolve(de2a,y(x));
dsolve(de2b,y(x));
```

$$y(x) = -1 + \mathrm{e}^{-\cos(x)}\_C1$$
$$y(x) = -1 + \mathrm{e}^{-\cos(x)}\_C1$$
$$y(x) = \_C1\,\mathrm{e}^{\sqrt{k}x} + \_C2\,\mathrm{e}^{-\sqrt{k}x}$$
$$y(x) = \_C1\,\mathrm{e}^{\sqrt{k}x} + \_C2\,\mathrm{e}^{-\sqrt{k}x}$$

Differenciálegyenletek IV

A `dsolve` által kiszámolt megoldások az eredeti differenciálegyenletbe való visszahelyettesítéssel ellenőrizhetők. Erre – például a korábban tanult `subs` és `eval` eljárások fölhasználásával – több módszer is van.

```
de := diff(y(x),x) = 1 - 2*y(x); Mo := dsolve(de);
```

$$de := \frac{d}{dx}y(x) = 1 - 2\,y(x)$$
$$Mo := y(x) = \frac{1}{2} + \mathrm{e}^{-2x}\_C1$$

```
subs(Mo,de); simplify(%); rhs(%) -lhs(%);
```

$$\frac{\partial}{\partial x}\left(\frac{1}{2} + \mathrm{e}^{-2x}\_C1\right) = \frac{1}{2} + \mathrm{e}^{-2x}\_C1$$
$$-2\,\mathrm{e}^{-2x}\_C1 = -2\,\mathrm{e}^{-2x}\_C1$$
$$0$$

```
eval(de, Mo); rhs(%)-lhs(%);
```

$$-2\,\mathrm{e}^{-2x}\_C1 = -2\,\mathrm{e}^{-2x}\_C1$$
$$0$$

A legelegánsabb az `odetest` eljárás használata, melynek első paramétere a `dsolve` által visszaadott megoldás, a második meg a vizsgált differenciálegyenlet. Lásd `?odetest`. . A teszt sikeres, ha az `odetest` által visszaadott érték 0.

```
odetest(Mo,diffeq);
```

$$0$$

Differenciálegyenletek V

Egyértelmű megoldást kaphatunk (a szabad integrációs konstansok nélkül), ha differenciálegyenleteinket kezdeti értékek előírásával egészítjük ki. Elsőrendű egyenleteknél egy, másodrendűeknél két, az `y(x)`-re vonatkozó kezdeti értéket írhatunk elő. Így kezdetiérték-problémát (angolul Initial Value Problem, IVP) kapunk. Az egyenletet és a kezdeti értékeket vagy listában, vagy halmazban adhatjuk át a `dsolve`-nak.

```
kep1 := [de, y(0) = 0]; dsolve(kep1, y(x));
kep2 := {de, y(0) = 0}; dsolve(kep2, y(x));
```

$$kep1 := \left[\frac{d}{dx}y(x) = 1 - 2\,y(x), y(0) = 0\right]$$
$$y(x) = 1/2 - 1/2\,e^{-2x}$$
$$kep2 := \left\{\frac{d}{dx}y(x) = 1 - 2\,y(x), y(0) = 0\right\}$$
$$y(x) = 1/2 - 1/2\,e^{-2x}$$

Az ellenőrzés most is az előzőekhez hasonlóan történhet, legegyszerűbben a `dsolve`-val. Ez most ellenőrzi a kezdeti feltételek teljesülését is. Figyeljük meg, hogy a visszaadott érték listás IVP esetén `[0, 0]`, a halmazos megadásnál viszont csak `{0}`.

```
Mo := dsolve(kep1, y(x)); odetest(Mo,kep1);
Mo := dsolve(kep2, y(x)); odetest(Mo,kep2);
```

$$Mo := y(x) = 1/2 - 1/2\,e^{-2x}$$
$$[0, 0]$$
$$Mo := y(x) = 1/2 - 1/2\,e^{-2x}$$
$$\{0\}$$

Differenciálegyenletek VI

Néha a dsolve csak valamilyen implicit vagy `RootOf`-os alakban tudja megadni a megoldást. Az is előfordulhat, hogy egyáltalán nem boldogul. Ekkor semmit, pontosabban a `NULL` konstanst adja vissza.

```
de1 := diff(y(x),x) = 1 + x/y(x);
Mo := dsolve(de1);
odetest(Mo,de1);
```

$$de1 := \frac{d}{dx}y(x) = 1 + \frac{x}{y(x)}$$

$$Mo := -1/2\,\ln\left(-\frac{x^2 + y(x)\,x - (y(x))^2}{x^2}\right) - 1/5\,\sqrt{5}\operatorname{arctanh}\left(1/5\,\frac{(-2\,y(x) + x)\,\sqrt{5}}{x}\right) - \ln(x) - \_C1 = 0$$

$$0$$

```
de2 := diff(y(x),x) = 1 - x*sqrt(y(x));
Mo := dsolve(de2,y(x));
```

$$de2 := \frac{d}{dx}y(x) = 1 - x\sqrt{y(x)}$$

$$Mo :=$$

Differenciálegyenletek VII

Ha részletesen szeretnénk tudni, hogy mit csinál a `dsolve`, növeljük meg az `infolevel[dsolve]` értékét:

```
infolevel[dsolve]:= 3;
de := diff(y(x),x) = lambda*y(x) + (y(x)^2);
dsolve(de,y(x));
odetest(%,de);
infolevel[dsolve]:= 1;
```

$$infolevel_{dsolve} := 3$$

$$de := \frac{d}{dx}y(x) = \lambda\, y(x) + (y(x))^2$$

```
Methods for first order ODEs:
--- Trying classification methods ---
trying a quadrature
trying 1st order linear
trying Bernoulli
<- Bernoulli successful
```

$$y(x) = \frac{\lambda}{-1 + \mathrm{e}^{-\lambda x}\_C1\,\lambda}$$

$$0$$

$$infolevel_{dsolve} := 1$$

Differenciálegyenletek VIII

A `dsolve` képes különböző jellegű közelítő megoldások meghatározására is. Erre akkor lehet szükség, ha a pontos megoldás nem ismert, vagy túl bonyolult függvény. Így például hatványsorral is tudja közelíteni a kezdetiérték-probléma megoldását, ha a `type=series` opciót adjuk meg. Ezt hívják Taylor-sor módszernek. Az eredmény végén `O(...)` jelöli a közelítő megoldás hibatagját. A `convert` eljárással az eredményt közönséges polinommá alakíthatjuk.

```
de := diff(y(x),x) = 1 - x*sqrt(y(x));
KEP := {de, y(1) =1};
Mo := dsolve(KEP);
```

$$de := \frac{d}{dx}y(x) = 1 - x\sqrt{y(x)}$$

$$KEP := \left\{\frac{d}{dx}y(x) = 1 - x\sqrt{y(x)}, y(1) = 1\right\}$$

$$Mo :=$$

```
dsolve(KEP,y(x),type=series);
convert(%,polynom);
```

$$y(x) = 1 - \frac{1}{2}(-1+x)^2 + \frac{1}{12}(-1+x)^3 + \frac{5}{96}(-1+x)^4 - \frac{7}{960}(-1+x)^5 + O\left((-1+x)^6\right)$$

$$y(x) = 1 - \frac{1}{2}(-1+x)^2 + \frac{1}{12}(-1+x)^3 + \frac{5}{96}(-1+x)^4 - \frac{7}{960}(-1+x)^5$$

Differenciálegyenletek IX

A Maple meg tud oldani magasabb rendű (akár parciális) differenciálegyenleteket és differenciálegyenlet-rendszereket is. Másodrendű egyenletek esetében szokás ún. peremérték feladatokat (Boundary Value Problem, BVP) vizsgálni. Ezeknél az ismeretlen `y(x)` függvényre két peremfeltételt adunk meg, például előírjuk értékét két különböző helyen. Az alábbi példa a harmonikus rezgőmozgások legegyszerűbb esetének felel meg.

```
de3 := diff(y(x),[x,x]) = -k*y(x);
dsolve (de3);
```

$$de3 := \frac{d^2}{dx^2}y(x) = -ky(x)$$
$$y(x) = \_C1 \sin\left(\sqrt{k}x\right) + \_C2 \cos\left(\sqrt{k}x\right)$$

```
PEP := [de3, y(0) = 0, y(1) = 1];
dsolve(PEP);
odetest(%, PEP);
```

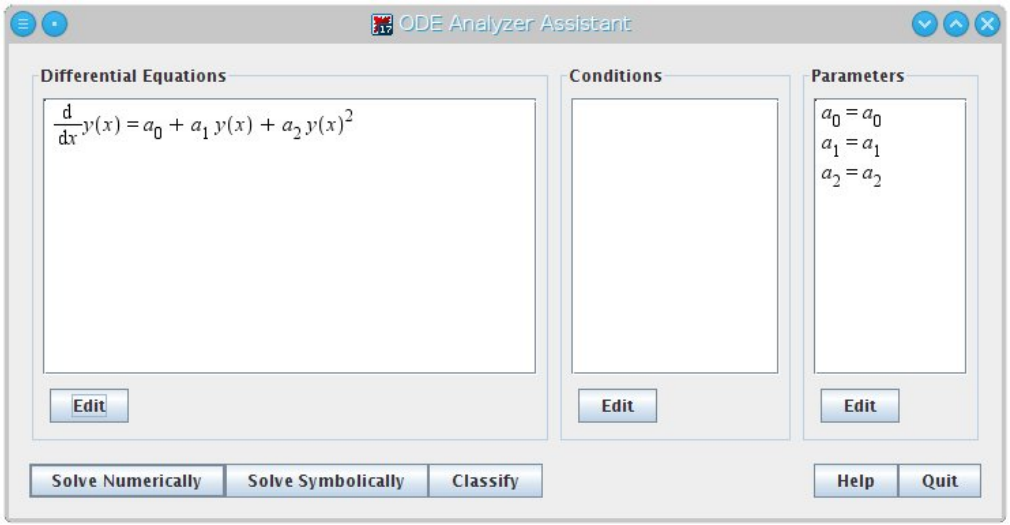
$$PEP := \left[\frac{d^2}{dx^2}y(x) = -ky(x), y(0) = 0, y(1) = 1\right]$$
$$y(x) = \frac{\sin(\sqrt{k}x)}{\sin(\sqrt{k})}$$
$$[0, 0, 0]$$

Differenciálegyenletek X

A `dsolve`-hoz is tartozik egy interaktív (maplet-es) grafikus felület. Itt a megoldandó differenciálegyenlet mellett megadhatunk különböző mellékfeltételeket, a paraméterekre vonatkozó megkötéseket, stb. Lásd `?dsolve,interactive`.
Ezen kívül a `DEtools` csomag még számos hasznos segédeszközt tartalmaz.

```
de := diff(y(x),x) = 1 - x*y(x);
dsolve[interactive]({de});
```

A `dsolve[interactive]` hívása után felbukkanó ablak képe az alábbi ábrán látható.



14. előadás

14. előadás

Ebben az előadásban először a Maple hatékony programozásához szükséges további eszközöket mutatunk be, melyekkel akár saját Maple csomagokat is készíthetünk.
Ezután röviden áttekintjük a legfontosabb, az oktatásban és a gyakorlati feladatokban is jól használható Maple csomagokat.

8. Programozási alapok

Táblák I.

A táblákban tetszőleges objektumokat tárolhatunk, ezekhez indexek (kulcsok) segítségével férhetünk hozzá. Lásd `?table`. Először egy üres táblát hozunk létre, majd különböző elemeket helyezünk el benne.
Táblanevek esetében a Maple alapból nem végez teljes kiértékelést, ezt a `print` vagy az `eval` használatával érhetjük el.

```
t := table();
```

$$t := table([\,])$$

```
t[4] := "négy"; t["öt"] := 5; t[4,5] := "negyvenöt";
t[x*y] := 9;t["kisprimek"] := {2,3,5,7,11};
```

$$t_{4} := \text{"négy"}$$
$$t_{\text{"öt"}} := 5$$
$$t_{4,5} := \text{"negyvenöt"}$$
$$t_{xy} := 9$$
$$t_{\text{"kisprimek"}} := \{2, 3, 5, 7, 11\}$$

```
t; print(t); eval(t);
```

$$t$$
$$table([4 = \text{"négy"}, \text{"kisprimek"} = \{2, 3, 5, 7, 11\}, (4, 5) = \text{"negyvenöt"}, \text{"öt"} = 5, xy = 9])$$
$$table([4 = \text{"négy"}, \text{"kisprimek"} = \{2, 3, 5, 7, 11\}, (4, 5) = \text{"negyvenöt"}, \text{"öt"} = 5, xy = 9])$$

Táblák II.

A tábla összes indexét az `indices` függvény adja vissza. A táblában tárolt összes értéket az `entries` függvénnyel kaphatjuk meg. Tábla tartalmának formázott kiíratására használhatjuk például a `printf` függvényt.

```
indices(t);
```

$$[4], [\text{"kisprimek"}], [4, 5], [\text{"öt"}], [xy]$$

```
entries(t);
```

$$[\text{"négy"}], [\{2, 3, 5, 7, 11\}], [\text{"negyvenöt"}], [5], [9]$$

```
for ix in indices(t) do
  printf("A(z) %a indexhez tartozó bejegyzés: %a\n", ix, t[op(ix)])
> end do;

A(z) [4] indexhez tartozó bejegyzés: "négy"
A(z) ["kisprimek"] indexhez tartozó bejegyzés: {2, 3, 5, 7, 11}
A(z) [4, 5] indexhez tartozó bejegyzés: "negyvenöt"
A(z) ["öt"] indexhez tartozó bejegyzés: 5
A(z) [x*y] indexhez tartozó bejegyzés: 9
```

Táblák III.

Ha értékadás jobb oldalán táblanév szerepel, akkor a baloldali változó értéke erre a táblára való hivatkozás lesz.

```
tnew := t;
verify(t, tnew, 'table');
is(tnew=t);
```

$$tnew := t$$
$$true$$
$$true$$

```
t := sin(x);
eval(tnew);
```

$$t := sin(x)$$
$$sin(x)$$

Új indexes változónevek első előfordulásakor a Maple a változónévnek megfelelő új táblát hoz létre:

```
x[1] := valami;
type(x,'table');
```

$$x_1 := valami$$
$$true$$

Rekordok I.

A *rekord* adattípus más programozási nyelvekből is ismerős lehet. A Maple belső implementáció modulokat használ.Lásd `?record`.
Új rekordokat a `Record` eljárással hozhatunk létre. Az egyes mezők elérése a mezőnevekkel, a **rekordnév** : −**mezőnév** jelöléssel történik. A `max_record_depth` Maple kernelopció befolyásolja, hogy milyen mélyen értékeli ki a rendszer a rekordokat.

```
r := Record('a' = "alma", 'b' = 33,
        'belsorekord' = Record(m1 = 13, m2 = true));
r:-a;
r:-belsorekord;
r:-belsorekord:-m1;
```

$$r := Record(a = \text{"alma"}, b = 33, belsorekord = Record(...))$$
$$\text{"alma"}$$
$$belsorekord$$
$$13$$

```
kernelopts(max_record_depth = 0): eval(r);
kernelopts(max_record_depth = 1): eval(r);
kernelopts(max_record_depth = 2): eval(r);
```

$$Record(...)$$
$$Record(a = \text{"alma"}, b = 33, belsorekord = Record(...))$$
$$Record(a = \text{"alma"}, b = 33, belsorekord = Record(m1 = 13, m2 = true))$$

Rekordok II.

Mit jelentek az alábbi értékadások? Figyeljük meg az eredményeket.

```
s := Record('a' = Pi*sqrt(2), 'b' = "masodik");
t := s;
t:-b := "új második";
s:-a, s:-b;
t:-a, t:-b;
```

$$s := Record(a = \pi\sqrt{2}, b = \text{"masodik"})$$
$$t := s$$
$$b := \text{"új második"}$$
$$\pi\sqrt{2}, \text{"új második"}$$
$$\pi\sqrt{2}, \text{"új második"}$$

Rekordok egyenlőségét a verify eljárással tesztelhetjük.

```
verify(r, r, 'record');
verify(r, s, 'record');
verify(r, t, 'record');
verify(s, t, 'record');
```

$$true$$
$$false$$
$$false$$
$$true$$

Rekordok III.

A *Record* adattípus segítségével természetes módon definiálhatunk különféle objektumokat, például köröket. Sőt új Maple típust is bevezethetünk, a neve legyen *kor*.

```
k1 := Record('kozeppont' = [0, 1], 'sugar' = 3);
```

$$k1 := Record(kozeppont = [0, 1], sugar = 3)$$

```
`type/kor` := 'record(kozeppont, sugar)';
```

$$type/kor := record(kozeppont, sugar)$$

```
terulet := (c::kor) ->(c:-sugar)^2 * Pi;
terulet(k1);
kerulet := (c::kor) -> (c:-sugar) * 2*Pi;
kerulet(k1);
```

$$terulet := (c :: kor) -> c : -sugar^2 Pi$$
$$9\pi$$
$$kerulet := (c :: kor) -> 2c : -sugar\pi$$
$$6\pi$$

Az `if` utasítás I.

Ha utasítások egy sorozatát csak bizonyos feltétel(ek) teljesülése esetén akarjuk végrehajtani, akkor feltételes utasítást használunk. Ennek lehetséges alakjai:

`if` **feltétel1** `then` **utasítás1** `fi;`
`if` **feltétel1** `then` **utasítás1** `else` **utasítás2** `fi;`
`if` **feltétel1** `then` **utasítás1** `elif` **feltétel2** `then` **utasítás2** `...fi;`
`if` **feltétel1** `then` **utasítás1** `elif` **feltétel2** `then` **utasítás2** `...else` **utasítás** `fi;`

A Maple sorra vizsgálja az utasításban lévő feltételeket, és amint talál egy `true` logikai értékűt, végrehajtja a hozzá tartozó utasításokat. Ha egyik feltétel sem teljesül, azaz mindegyik értéke `false` vagy `FAIL`, akkor csak az `else` utáni utasítások hajtódnak végre, ha létezik az `else` ág.
A feltételekben bármilyen logikai kifejezés állhat, használhatóak a relációjelek (`<`, `<=` , `>`, `>=`, `=`, `<>`), a logikai műveleti jelek (`and`, `or`, `not`) és a logikai értékek (`true`, `false`, `FAIL`).
Ha a `then` vagy az `else` után csak egyetlen utasítás áll, nem szükséges utána a `;` vagy a `:` kitétele. A `fi` kulcsszó helyett `end if` is írható, használjuk inkább ezt az olvashatóbb változatot.

Az if utasítás II.

A feltételes utasítások alkalmazására két egyszerű példa: számok maximuma és abszolut értéke

```
restart;
a:= -10; b:=-2;
```

$$-10$$
$$-2$$

```
if  a > b then a
  else b
end if;
```

$$-2$$

```
if a < 0 then -a
  else a
end if;
```

$$10$$

Az `if` utasítás III.

Bizonyos szimbolikus kifejezésekre a Maple nem tudta eldönteni a >, < vagy a = relációk teljesülését. Ilyenkor segíthet az `is(...)` eljárás beiktatása. Néhány példa:

```
if sqrt(2) <= 0 then print("Igen") else print("Nem") end if;
if 3 < Pi then print("Igen") else print("Nem") end if;
```

```
Error, cannot determine if this expression is true or false: 2^(1/2) <= 0
Error, cannot determine if this expression is true or false: 3 < Pi
```

```
if is(sqrt(2) <= 0) then print("Igen") else print("Nem") end if;
if is(3 < Pi) and is(Pi < 4) then print("Igen") else print("Nem") end if;
```

"Nem"
"Igen"

```
kif := Pi^3 - 3^3 + (33083/600)*exp(1)^3/(3-5^(7/2));
evalf(kif);
evalb(0 < kif);
is(0 < kif);
```

$$\pi^3 - 27 + \frac{33083}{600}\,\frac{\left(e^1\right)^3}{3 - 125\,\sqrt{5}}$$

$$0.001036197$$

$$0 < \pi^3 - 27 + \frac{33083}{600}\,\frac{\left(e^1\right)^3}{3 - 125\,\sqrt{5}}$$

$$true$$

Az `if` utasítás IV.

Vigyázzunk a `FAIL` logikai érték kezelésére!

```
Bval := FAIL;
if Bval then print("Teljesült a feltétel!") end if;
if not Bval then print("Teljesült a feltétel!") end if;
```

FAIL

```
Bval := false;
if Bval then print("Teljesült a feltétel!") end if;
if not Bval then print("Teljesült a feltétel!") end if;
```

false
"Teljesült a feltétel!"

A `for` utasítás I.

A `for` ciklus utasítás egy utasítássorozat ismételt végrehajtását teszi lehetővé. A Maple két típusát használja. Az első a „from-to” („tól-ig”) ciklus:

`for` **ciklusváltozó** `from` **kezdőérték** `by` **növekmény** `to` **végérték** `while` **logikaifeltétel** `do` **ciklusmag** `od;`

Hatására a **ciklusváltozó** minden lehetséges értéke mellett végrehajtódik a **ciklusmag** utasítás(sorozat), amíg a **logikaifeltétel** értéke `true`. A **ciklusváltozó** értéke **kezdőérték**-től kezdve **növekmény** lépésközzel halad, amíg értéke túl nem lépi a **végérték**-et.
Az utasításból szinte minden, így a `for` **ciklusváltozó**, a `from` **kezdőérték**, a `by` **növekmény**, a `to` **végérték** és a `while` **logikaifeltétel** rész is elhagyható. A **ciklusmag** is lehet üres. Ilyenkor a következő alapértelmezések szerint jár el a Maple:

ciklusváltozó	üres („dummy”) változó
kezdőérték	1
növekmény	1
végérték	`infinity`
logikaifeltétel	`true`

A ciklusmagban használhatók a `next` és a `break` utasítások a ciklus adott futásából, illetve a ciklusból való végleges kilépésre. Az `od` kulcsszó helyett `end do` is írható, használjuk inkább ezt az olvashatóbb változatot!

A for utasítás II.

A Maple szimbolikus lehetőségeit kihasználva így csinálhatunk egy „általános” (szintaktikus kategóriákkal fölírt) for ciklusból egy „konkrét”, végrehajtható parancsot:

```
restart;
for `ciklusváltozó` from `kezdőérték`  by `növekmény` to `végérték` do
   `ciklusmag`;
end do;
```

```
Error, increment of for loop must be numeric
```

```
`ciklusváltozó` := i: `kezdőérték` := 1:
`növekmény`     := 1: `végérték`   := 3:
`ciklusmag`     := 'printf("Helló világ! ")':
for `ciklusváltozó` from `kezdőérték`  by `növekmény` to `végérték` do
   `ciklusmag`;
end do;
```

Helló világ! Helló világ! Helló világ!

Az előző értékadásokkal tulajdonképpen az alábbi ciklus utasítást adtuk meg:

```
for i from 1 to 3 by 1 do
  printf("Helló világ! "):
end do;
```

Helló világ! Helló világ! Helló világ!

A `for` utasítás III.

Írjunk olyan ciklust, amely kiírja az első k darab prímszámot. Itt az n ciklusváltozót „automatikusan” kezeli a Maple, nincs szükség kezdőérték, növekmény és végérték megadására.

```
k := 10: numprimes := 0:
for n while numprimes < k do
  if isprime(n) then  print(n);  numprimes := numprimes +1;
  end if;
end do;
```

$$2$$
$$3$$
$$5$$
$$7$$
$$11$$
$$13$$
$$17$$
$$19$$
$$23$$
$$29$$

A `for` utasítás IV.

Írjunk olyan ciklust, amely sorban kipróbálja az egész n-ekre 0-tól 39-ig, hogy n^2+n+41 prímszám-e („hamis” prímszám formula).

```
for n from 0 to 39 do
    printf( "%d:%a ", n^2+n+41, isprime(n^2+n+41) ):
end do;
```

```
41:true 43:true 47:true 53:true 61:true 71:true 83:true 97:true 113:true
131:true 151:true 173:true 197:true 223:true 251:true 281:true 313:true
347:true 383:true 421:true 461:true 503:true 547:true 593:true 641:true
691:true 743:true 797:true 853:true 911:true 971:true 1033:true 1097:true
1163:true 1231:true 1301:true 1373:true 1447:true 1523:true 1601:true
```

No de meddig lesz ez igaz? Egy megfelelő `while` feltétel beiktatásával megtudjuk abból, hol állt le a ciklus. Ehhez még ciklusmag sem kell.
A ciklusváltozó végértékéből látszik, hogy $n = 40$ esetén már nem ad prímet a fenti formula.

```
for n from 0 while isprime( n^2+n+41 ) do end do;
n; n^2+n+41; ifactor(%);
```

$$40$$
$$1681$$
$$41^2$$

A for utasítás V.

A ciklusutasítás másik alakja az ún. „in" ciklus, amelyben a ciklusváltozó egy Maple objektum (tipikusan lista vagy sorozat) elemein fut végig. Alakja:

```
for ciklusváltozó in objektum while logikaifeltétel do ciklusmag od;
```

```
for i in [1,sqrt(2),"barack"] do printf("i értéke: %a\n",i) end do;

i értéke: 1
i értéke: 2^(1/2)
i értéke: "barack"

for i in x^3-3*x+sin(x) do printf("i értéke: %a\n",i) end do;

i értéke: x^3
i értéke: -3*x
i értéke: sin(x)

for i in "alma" do printf("i értéke: %a\n",i) end do;

i értéke: "a"
i értéke: "l"
i értéke: "m"
i értéke: "a"
```

Az op függvénycsaláddal is „szimulálhatunk" ilyen ciklusokat:

```
L := [1,sqrt(2),"barack"]: for i to nops(L) do printf("i értéke: %a\n", op(i,L)) end do;

i értéke: 1
i értéke: 2^(1/2)
i értéke: "barack"
```

Futásidő mérése

A futásidő (CPU idő) mérésére a `time` függvényt használhatjuk. Ha csak egyetlen Maple utasítás végrehajtási idejére vagyunk kíváncsiak, adjuk meg a `time` függvény argumentumaként. Ilyenkor nem írja ki a Maple az utasítás outputját. Az időmérést célszerű mindig egy `restart` után kezdeni, hogy kiküszöböljük a Maple cache mechanizmusának torzító hatását. Lásd `?time`.

```
ifactor(2^222+17);
```

$$(3)^4\,(150561874262832393364 81)\,(217163381)\,(10290476271139733129094424397)$$

```
restart;
time(ifactor(2^222+17));
```

$$6.151$$

Hosszabb utasítássorozat esetében az elején és a végén hívjuk a `time` függvényt, és a két időpont különbsége adja a (másodpercekben mért) végrehajtási időt. Az utasításokat kettősponttal zárjuk (nem kérünk outputot), mivel ez is sok időt vihet el.

```
restart;
kezdesi_ido := time();
ifactor(2^222+17):
harmonikus_sor := sum(1/'k', k = 1..50000):
# esetleges további utasítások ...
befejezesi_ido := time();
vegrehajtasi_ido := befejezesi_ido - kezdesi_ido;
```

$$kezdesi\_ido := 12.531$$
$$befejezesi\_ido := 18.731$$
$$vegrehajtasi\_ido := 6.200$$

Eljárások I.

Az ismétlődő, többször végrehajtandó számítási feladatokat más nyelvekhez hasonlóan Maple *eljárásokként* adhatjuk meg.
Az eljárások definiálásának szintaxisa alább látható (a `local`, `global`, `options`, `description`, `uses` sorok bármelyike hiányozhat, a paramétersorozat is lehet üres paraméter nélküli eljárásoknál). A sorok végén állhat `;` is.

eljárásnév `:= proc(`**paramétersorozat**`)`
`local` **névsorozat**`:`
`global` **névsorozat**`:`
`options` **névsorozat**`:`
`description` **sztring**`:`
`uses` **névsorozat**`:`
utasítássorozat`:`
`end:`

A paramétersorozat adja meg az eljárás formális paramétereit. A lokális, illetve globális változók megadása nem kötelező, de ajánlott (a Maple egy beépített algoritmus szerint maga dönti el, mely változókat tekinti lokálisnak, illetve globálisnak).
Az eljárásoknak mindig van visszatérési értékük: ez alapesetben az utolsó végrehajtott művelet eredménye. Ha más visszatérési értéket szeretnénk, akkor a
`return` **kifejezés**
utasítást kell használnunk. Ez az eljárás futását megszakítja, s a megadott kifejezés értékével tér vissza. Az eljárást záró `end` helyett `end proc` is írható, használjuk inkább ezt az olvashatóbb változatot!

Eljárások II.

Más programozási nyelvekhez hasonlóan az eljárásokban definiálhatunk lokális változókat. Ha a lokális változóknak értéket adunk az eljárás hívásakor, ettől még az azonos nevű globális változók értéke nem változik, sőt ezeket el sem érhetjük az eljárás kódjának futása közben, „belülről”.
Nézzük meg, hogyan kezeli a Maple a `g` globális változót és a `localdemo` eljárás `g` lokális változóját.
A lokális változók deklarálásakor rögtön kezdőértékeket is megadhatunk nekik, ezt tettük a `h` változóval.

```
g := "vadalma";
localdemo := proc( )
   local g, h := Pi:
   g := sqrt(2);
   print(g, h):
   return g:
end proc:
s := localdemo();
g;
```

$$g := \text{"vadalma"}$$
$$\sqrt{2}, \pi$$
$$s := \sqrt{2}$$
$$\text{"vadalma"}$$

Eljárások III.

Írjunk a faktoriális függvényt definiáló két Maple eljárást, egy ciklikusat és egy rekurzívat.

```
fact1 := proc(n)
  local prod, i;
  description "fact1: az n szám faktoriálisát meghatározó eljárás":
  prod := 1;
  for i from 2 to n do
    prod := prod * i
  end do;
end proc:
fact2 := proc(n)
  description "fact2: az n szám faktoriálisát meghatározó eljárás":
  if n = 0 then 1 else n * fact2(n-1) end if
end proc;

fact1(4);
fact2(5);
```

```
                              24
                             120
```

Eljárások IV.

Az eljárások fejében a `description` után megadott leírást és az eljárás paramétereit adja vissza a `Describe` eljárás.

```
Describe(fact1);
```

```
fact1: az n szám faktoriálisát meghatározó eljárás
fact1( n )
```

A Maple legnagyobb közös osztót számoló beépített `gcd` eljárását is lekérdezhetjük.

```
Describe(gcd);
```

```
gcd( aa, bb, cofa::name, cofb::name )
```

Eljárások V.

Mi történik, ha a faktoriálist számító valamelyik előző eljárásunkat „hibás" vagy „értelmetlen" értékkel hívjuk meg?

```
fact1(sin(x));
Error, (in fact1) final value in for loop must be numeric
or character
```

Írjunk olyan `fact3` eljárást, amely bemenő paraméterének típusát vizsgálva igyekszik ezt a hibát kikerülni.
Eljárások futása közben hibaüzeneteket az `error` utasítás végrehajtásával írathatunk ki. Hatására megszakad az aktuális utasítássorozat végrehajtása, és a vezérlés „egy szinttel följebb” kerül. Legegyszerűbb változatában csak a hibaüzenetnek megfelelő sztringet kell megadni.
Ha meg akarjuk jeleníteni az eljáráshívás paramétereit is, a `%1`, `%2`, ... jelöléssel hivatkozhatunk rájuk. Lásd `?error`. Néha a Maple „magától" is végrehajt ilyen utasításokat, ha valamilyen futási hiba lép föl.
Figyelmeztetéseket a `WARNING` eljárással adhatunk meg, ekkor nem szakad meg a végrehajtás.

Eljárások VI.

```
fact3 := proc(n)
   local prod, i;
   description "fact3: az n szám faktoriálisát meghatározó eljárás":
   if not type(n, posint) then
     error "n = %1, de a függvény csak pozitív egész számok faktoriálisát számolja.",n:
   elif n > 50 then
     WARNING("(in fact3) Az n = %1 szám elég nagy, jól meggondolta?!",n):
   end if:
   prod := 1:
   for i from 2 to n do prod := prod * i end do:
end proc:

fact3(4);
```

$$24$$

```
fact3(sin(x));
Error, (in fact3) n = sin(x), de a függvény csak pozitív egész számok faktoriálisát számolja.

fact3(-11);
Error, (in fact3) n = -11, de a függvény csak pozitív egész számok faktoriálisát számolja.

fact3(51);

Warning, (in fact3) Az n = 51 szám elég nagy, jól meggondolta?!
```

$$155111875328738228022424301646930321106325972001698611200000000000$$

Eljárások VII.

Az előző problémákat még elegánsabban, a formális paraméterek és a visszadott érték típusának megadásával is kezelhetjük. A **paraméternév::típusnév** specifikációval megadhatjuk, hogy az eljárás hívásakor az adott paraméter értéke milyen típusú lehet. A lehetséges típusmegadásokat a Helpből kereshetjük ki: `?type`

```
> fact4 := proc(n::posint)::posint:
>     if n = 1 then 1 else n * fact3(n-1) end if
> end proc:

fact4(4);
fact4(sin(x));
fact4(-11);
```

24

```
Error, invalid input: fact4 expects its 1st argument, n, to be of type
posint, but received sin(x)
Error, invalid input: fact4 expects its 1st argument, n, to be of type
posint, but received -11
```

Eljárások VIII.

Készítsünk a Pascal-háromszög sorait kiíró eljárást. Használjuk föl, hogy a Maple tartalmazza a `binomial` eljárást: `binomial(i,j)` az $\binom{i}{j}$ binomiális együtthatót adja vissza.

```
pascal := proc(m::nonnegint , n::nonnegint)::NULL:
   local i, j;
   option Copyright = " Virágh János, 2014":
   description "A Pascal háromszöget kiíró eljárás":
   for i from m to n do
     seq( binomial(i,j), j=0..i );
     print(%);
   end do;
end proc:

pascal(0,2);
```

$$1$$
$$1,\ 1$$
$$1,\ 2,\ 1$$

```
pascal(4,6);
```

$$1,\ 4,\ 6,\ 4,\ 1$$
$$1,\ 5,\ 10,\ 10,\ 5,\ 1$$
$$1,\ 6,\ 15,\ 20,\ 15,\ 6,\ 1$$

Eljárások IX.

Vajon milyen operandusai vannak a `pascal` eljárásnak? Mit tudhatunk meg az `op` függvénycsalád segítségével? Az alábbi hívásnál az egyes operandusokat a `[]` listazárójelbe tettük, hogy jobban látsszanak azok is, amelyek most `NULL` értékűek. Az eljárások `op` paranccsal visszakapható egyes részkifejezései:

0. a *procedure* típusnév
1. paraméter lista
2. lokális változók listája
3. opciók
4. remember tábla (lásd később)
5. description
6. globális változók listája
7. belső lexikális tábla
8. a visszaadott érték típusa

További részletek: `?proc`

```
nops(eval(pascal));
seq([op(i,eval(pascal))], i=0..nops(eval(pascal)));
```

$$[procedure], [m :: nonnegint, n :: nonnegint], [i, j], [Copyright = \text{" Virágh János, 2014"}], [\,],$$
$$[\text{"A Pascal háromszöget kiíró eljárás"}], [\,], [\,], [NULL]$$

Eljárások X.

Következő példánkban a Heron-képlet szerinti közelítő gyökvonó algoritmust implementálunk. Figyeljük meg az eljárás különböző hívásainak eredményét. A visszaadott érték típusa az input típusától függ.

```
mysqrt := proc(x::numeric)
     local r, i:
     r := x:
     for i to 5 do r := (r*r + x)/(2*r) end do;
     return r:
end proc:

mysqrt(2);
evalf(%);
```

$$\frac{886731088897}{627013566048}$$

$$1.414213562$$

```
mysqrt(2.0);
```

$$1.414213562$$

```
mysqrt(Pi)
Error, invalid input: mysqrt expects its 1st argument, x,
to be of type numeric, but received Pi
```

Eljárások XI.

A `printlevel` egész értékű rendszerváltozó értékének megnövelésével részletesebb információkat kapunk a Maple utasítások végrehajtása közben.

```
printlevel := 5:
mysqrt(2.0);
{-> enter mysqrt, args = 2.0}
```

$$r := 2.0$$

```
<- exit mysqrt (now at top level) = 1.414213562}
```

$$1.414213562$$

Ejárásaink kódjában a `trace` opció megadásával kérhetünk részletesebb nyomkövetést. (Hasonló eredményre vezet a `printlevel := 6` értékadás.)
A Maple további, interaktív nyomkövetési eljárásokat is tartalmaz, lásd `?debugger`, de ezzel nem foglalkozunk.

```
printlevel := 1:
mysqrttr := proc(x::numeric)
    local r, i;
    option trace:
    r := x;
    for i to 5 do r := (r*r + x)/(2*r) end do;
end proc:
mysqrt_tr(2.0);
{-> enter mysqrttr, args = 2.0}
```

$$r := 2.0$$
$$r := 1.500000000$$
$$r := 1.416666666$$
$$r := 1.414215686$$
$$r := 1.414213562$$
$$r := 1.414213562$$

```
<- exit mysqrttr (now at top level) = 1.414213562}
```

$$1.414213562$$

Eljárások XII.

A Fibonacci sorozat szokásos rekurzív kiszámolásakor az ismétlődő rekurzív hívások miatt egy újabb tag kiszámítása majdnem dupla annyi ideig tart, mint az előzőé. Ezen segíthetünk, ha a `remember` opcióval azt kérjük, hogy az eljárás tárolja minden korábban kiszámolt tag értékét az eljárás „remember táblájában”. Ezt használjuk ki az eljárás `fibr` nevű változatában. A táblák tartalmát szükség esetén törölhetjük, lásd `?forget`.

```
restart;
fib := proc(k::integer)::integer;
    if k = 0 or k = 1 then 1
    else fib(k-1) + fib(k-2)
    end if
end proc:
seq(time(fib(k)),k=20..30);
```

$$0.021, 0.035, 0.062, 0.096, 0.157, 0.256, 0.408, 0.664, 1.074, 1.764, 2.785$$

```
fibr := proc(k::integer)::integer;
    option remember:
    if k = 0 or k = 1 then 1
    else fibr(k-1) + fibr(k-2)
    end if:
end proc:
> seq(time(fibr(k)),k=20..30);
```

$$0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.$$

Eljárások XIII.

A Maple több olyan speciális rendszerváltozót használ, melyek az éppen futó eljárásokról tartalmaznak információt. A `procname` változó az éppen futó eljárás nevét tárolja. Az `args` nevű rendszerváltozó segítségével érhetjük el az eljárás hívásakor megadott összes aktuális paraméterből álló kifejezéssorozatot. A híváskor megadott paraméterek számát az `nargs` változó tárolja. Így olyan eljárásokat is írhatunk, amelyek változó számú paraméterrel hívhatók.

```
sumproc := proc( )
  if nargs = 0 then error "Legalább egy paraméter kell!"
  elif  nargs = 1 and type(args[1], `+`) then [op(args[1])]
  elif  nargs = 2 then args[1] + args[2]
  else WARNING(" (in sumproc) Nem értem..."); print(args);
  end if:
end proc:
```

```
sumproc(x^2-2*x-sin(x));
```

$$[x^2, -2x, -\sin(x)]$$

```
sumproc(x-2, cos(z));
```

$$x - 2 + \cos(z)$$

```
sumproc(max(a,b));
```

```
Warning, (in sumproc) Nem értem...
```

$$max(a, b)$$

```
sumproc();
```

```
Error, (in sumproc) Legalább egy paraméter kell!
```

Eljárások XIV.

Az `ifactor` eljárás egész számok prímtényezőkre bontását végzi. Vajon hogyan működik? Próbáljunk meg a Helpen túl további információkat kérni a rendszertől.

```
restart;
ifactor(2^23+7);
```

$$(3)(5)(79)(7079)$$

```
Describe(ifactor);
```

```
ifactor( n )
```

```
print(ifactor);
```

proc(n)...**end proc**

```
eval(ifactor);
```

proc(n)...**end proc**

Eljárások XV.

Ettől nem lettünk sokkal okosabbak... Az `interface` eljárással a `verboseproc` rendszerváltozó értékét megnövelve már „mindent” elárul a Maple. (A hosszú programlistának csak az elejét mutatjuk.)

```
interface( verboseproc=2 );
```

$$1$$

```
print(ifactor):

proc(n)
local sol, r, t1;
global `ifactor/bottom`;
option remember, system,
`Copyright (c) 1991 by the University of Waterloo. All rights reserved.`;
    if nargs < 1 then error "argument required"
    elif 1 < nargs and not type(args[2], 'name') then
        error "second argument must be a name"
    end if;
    if type(n, 'integer') then
        if 0 < n then sol := 1; r := n
        elif n < 0 then sol := -1; r := -n
        else return 0
        end if
    elif type(n, 'fraction') then return
        procname(op(1, n), args[2 .. -1])/procname(op(2, n), args[2 .. -1])
    elif type(n, {'`*`', 'list', 'set', 'relation'}) then
    ...
```

Eljárások XVI.

A `showstat` eljárás a `verboseproc` értékétől függetlenül az `lprint` eljárás outputjának megfelelő 1D formátumban írja ki az eljárás kódját. Ha
`showstat(`**eljárásnév, kezdősor..végsor**`)`
paraméterekkel hívjuk, akkor csak az adott kezdősortól a befejező sorszámig írja ki a részletes eljáráskódot.

```
showstat(ifactor, 1..4);

ifactor := proc(n)
local sol, r, t1;
global `ifactor/bottom`;
   1   if nargs < 1 then
   2     error "argument required"
       elif 1 < nargs and not type(args[2],'name') then
   3     error "second argument must be a name"
       end if;
   4   if type(n,'integer') then
         ...
       elif type(n,'fraction') then
         ...
       elif type(n,{'`*`', 'list', 'set', 'relation'}) then
         ...
       elif type(n,`^`) and type(op(2,n),'integer') then
         ...
       elif type(n,('``')('integer')) then
         ...
       else
         ...
       end if;
       ...
end proc
```

Eljárások XVII.

A -> nyíl operátorral felírt függvények valójában a Maple eljárások
`option operator, arrow`
opciókkal definiált speciális esetei. Az alábbi `f1` és `f2` függvények – a különböző megadás ellenére – ekvivalensek. Ez látszik a `showstat` eljárás listáiból is.

```
f1 := (x) -> sin(x) + 1;
f2 := proc(x)
option operator, arrow:
sin(x) + 1:
end proc;
```

$$f1 := x- > \sin(x) + 1$$
$$f2 := x- > \sin(x) + 1$$

```
type(f1,`operator`);
type(f2,`operator`);
```

$$true$$
$$true$$

```
showstat(f1);
showstat(f2);
```

```
f1 := proc(x)
   1   sin(x)+1
end proc

f2 := proc(x)
   1   sin(x)+1
end proc
```

Modulok I.

A `module` a Pascal nyelvből ismerthez hasonló fogalom, összetartozó, elnevezett elemek kollekciója. Lásd `?module`. Az alábbi példa – az objektumorientált programozás szellemében – a „téglalap” fogalmának megfelelő egyszerű modult implementál. A tömörebb kód miatt az exportált változóknak rögtön kezdőértéket is adunk, és beillesztjük a modul rövid leírását.

```
teglalap1 := module()
 export hosszusag:=6, szelesseg:=5;
 description "téglalapot implementáló modul";
end module:
\begin{minp}
Describe(teglalap1);

#téglalapot implementáló modul
module teglalap1:

    hosszusag::integer = 6

    szelesseg::integer = 5
```

A modul exportált elemei a `:-` jelöléssel vagy a `[ ]` szelekciós operátorral érhetők el.

```
teglalap1 :- szelesseg, teglalap1[szelesseg];
teglalap1 :- hosszusag, teglalap1[hosszusag];
```

$$5,5$$
$$6,6$$

```
teglalap1 :- szelesseg := 4;
teglalap1 :-hosszusag := teglalap1 :-hosszusag + 4;
```

$$szelesseg := 4$$
$$hosszusag := 10$$

Modulok II.

A modulok lokális változóikban tárolhatják bizonyos adataikat, ezeket kívülről nem lehet közvetlenül elérni, módosítani.

```
teglalap2 := module()
   export hosszusag:=4, szelesseg:=5, teruletfv;
   local  elozo_terulet;
   elozo_terulet := hosszusag * szelesseg;
   teruletfv := proc()
        local t;
        t := hosszusag * szelesseg;
        if   t < elozo_terulet then printf("Összementem, az új területem: %d\n",t)
        elif t > elozo_terulet then printf("Megnőttem,  az új területem: %d\n",t)
        else printf("Semmi változás, a területem: %d\n",t)
        end if;
        elozo_terulet := t:
        NULL: # ez a trükk csak azért kell, hogy ne írja ki kétszer a területet...
   end proc:
end module:

teglalap2:-teruletfv();

Semmi változás, a területem: 20

> teglalap2:-hosszusag := 100:
> teglalap2:-teruletfv();

Megnőttem,  az új területem: 500
```

Modulok III.

Létrehozhatunk modulokat generáló eljárásokat is. Az `ujTeglalap` eljárás minden hívásakor egy új téglalap modult ad vissza. Megírhattuk volna így is:
`... m := module() ... end module: return m: end proc:`
de az alábbi kód rövidebb.

```
ujTeglalap := proc(h, s)
   module()
    export hosszusag:=h, szelesseg:=s, teruletfv:
    teruletfv   := proc() hosszusag * szelesseg end proc:
   end module:
end proc:

teglalap3 := ujTeglalap(8,4);

   teglalap3 := module() export hosszusag, szelesseg, teruletfv;  end module

teglalap3 :- teruletfv();
```

$$32$$

```
teglalap3 :- hosszusag := 10;
```

$$hosszusag := 10$$

```
teglalap3 :- teruletfv();
```

$$40$$

Csomagkészítés I.

Mint a 10. Előadásban is említettük, a Maple kernel alapfunkcióin túl a rendszer összes további „tudása” a Maple saját nyelvén megírt, és a hatékonyabb elérés miatt Maple könyvtárakba (`library`) összegyűjtött Maple csomagokon (`package`) alapul.
A csomagok elkészítésének technikai háttere az idők során többször változott, a Maple újabb változataiban a csomagok speciális modulok, lásd `module,package`. Erre rövidesen visszatérünk.

Csomagkészítés II.

A „régi” módszer szerint saját csomag létrehozásához csak egy üres táblára van szükségünk; ebben definiáljuk új eljárásainkat/függvényeinket.

```
restart;
ujcsomag1 := table():
ujcsomag1[duplasin]:= x->sin(x) + sin(2*x):
ujcsomag1[absmin]  := (x,y)->min(abs(x),abs(y)):
ujcsomag1[fib]     := proc(n) if (n = 0 or n = 1) then 1
    else ujcsomag1[fib](n-1) + ujcsomag1[fib](n-2)
    end if
end proc:
> print(ujcsomag1);
```

$$table([absmin = ((x,y) \to \min(|x|,|y|)), duplasin = (x \to \sin(x) + \sin(2x)), fib = proc(n)\ if\ n = 0\ or\ n = 1\ then\ 1\ else\ ujcsomag1[fib](n-1) + ujcsomag1[fib](n-2)\ end\ if\ end\ proc])$$

```
ujcsomag1[duplasin](Pi/5), ujcsomag1[absmin](-4, 3), ujcsomag1[fib](5);
```

$$\sin\left(\frac{\pi}{5}\right) + \sin\left(\frac{2\pi}{5}\right),\ 3,\ 8$$

Az `ujcsomag1` nevű csomag még csak a memóriában van definiálva. De már ugyanúgy használhatjuk, pl. betölthetjük a `with` paranccsal, mint a Maple saját csomagjait.

```
with(ujcsomag1);
duplasin(Pi/5), absmin(-4, 3), fib(5);
```

$$[absmin, duplasin, fib]$$
$$\sin\left(\frac{\pi}{5}\right) + \sin\left(\frac{2\pi}{5}\right),\ 3,\ 8$$

Csomagkészítés III.

Az elkészült csomagot a `save` paranccsal elmenthetjük; a fájl neve egyezzen meg a csomag nevével, azaz a tábla neve legyen.
A `libname` környezeti változó tartalmazza az elérhető könyvtárak útvonalát. Ehhez adjuk hozzá a saját csomagot tartalmazó könyvtárunk útvonalát.
Ezután már ebben a könyvtárban is keres a Maple csomagok betöltésekor, vagyis csomagunk pontosan úgy használható, mint a többi „gyári” Maple csomag.

```
save(ujcsomag1,"/home/viragh/Maple/ujcsomag1.m");

restart;
libname;
libname:=libname, "/home/viragh/Maple";
```

"/opt/maple17/lib", "."

$$libname := \text{"/opt/maple17/lib", ".", "/home/viragh/Maple"}$$

```
with(ujcsomag1);
duplasin(Pi/5), absmin(-4, 3), fib(5);
```

$$[absmin, duplasin, fib]$$

$$\sin\left(\frac{\pi}{5}\right) + \sin\left(\frac{2\pi}{5}\right),\ 3,\ 8$$

Csomagkészítés IV.

A Maple újabb változataiban a csomagok készítésének ajánlott módja *modulok* használata a `package` opcióval (lásd `?package,module`).
Írjuk meg így az előző `ujcsomag1`-gyel lényegében azonos tartalmú `ujcsomag2` csomagot. A csomag exportált elemeit a moduloknál szokásos `[ ]` vagy `:-` jelöléssel érhetjük el.

```
ujcsomag2 := module()
    export duplasin2, absmin2, fib2;
    option package;
    duplasin2 := x -> sin(x) + sin(2*x):
    absmin2   := (x,y) -> min(abs(x), abs(y)):
    fib2      := proc(n)
        if (n = 0 or n = 1) then 1
        else ujcsomag2[fib2](n-1) + ujcsomag2[fib2](n-2)
        end if
    end proc:
end module:

ujcsomag2[duplasin2](Pi/5), ujcsomag2[absmin2](-4, 3), ujcsomag2[fib2](5);
ujcsomag2:-duplasin2(Pi/5), ujcsomag2:-absmin2(-4, 3), ujcsomag2:-fib2(5);
```

$$\sin\left(\frac{\pi}{5}\right)+\sin\left(\frac{2\pi}{5}\right),\ 3,\ 8$$
$$\sin\left(\frac{\pi}{5}\right)+\sin\left(\frac{2\pi}{5}\right),\ 3,\ 8$$

Csomagkészítés V.

A `savelib` eljárással az elkészült modulból rögtön `.mla` kiterjesztésű Maple könyvtárat (`library`) készíthetünk, s később bármikor betölthetjük ebből a csomag eljárásait. Első paramétere a csomag neve, a második az elkészítendő könyvtárfájl neve.

```
savelib('ujcsomag2',"/home/viragh/Maple/ujcsomag2.mla");

restart;
libname;
libname:=libname, "/home/viragh/Maple";
```

"/opt/maple17/lib", "."

$$libname := \text{"/opt/maple17/lib", ".", "/home/viragh/Maple"}$$

```
with(ujcsomag2);
duplasin(Pi/5), absmin(-4, 3), fib(5);
```

$$[absmin, duplasin, fib]$$

$$\sin\left(\frac{\pi}{5}\right) + \sin\left(\frac{2\pi}{5}\right),\ 3,\ 8$$

Csomagkészítés VI.

A Maple tartalmaz egy `package` nevű adattípust, lásd `?type,package`. Ezt fölhasználva vizsgálhatjuk, hogy egy név valóban csomag neve-e.
Az éppen betöltött csomagok listáját a `packages` eljárás adja. Az `unwith` eljárás a `with` „inverze”, törli a memóriából a megadott nevű, a `with`-del korábban betöltött csomagot.
A `with` csak a legfelső szinten, a Maple interaktív használatakor alkalmazható. Eljárások, modulok, stb. kódján belül a `use` kulcsszó után adjuk meg, hogy az illető eljárás milyen csomagokat kíván használni – ennek hatása lényegében megegyezik a `with` parancséval.

```
type(ujcsomag2,'package');
type(plots,'package');
type(Plots,'package');
packages();
```

true
true
false
[*ujcsomag2*]

```
unwith(ujcsomag2);
packages();
```

[ ]