Maple V: Jak řešit úlohy na počítači

Tomáš Matoušek

Maple je program, který ti může pomoci při řešení matematicky formulovatelných problémů. Po spuštění čeká, oč ho požádáš. Příkazy, které chceš, aby vyplnil, mu píšeš na „příkazovou řádku“ vždy za symbol „>“. Budou mít červenou barvu a měly by končit středníkem nebo dvojtečkou. Středník znamená, že Maple zpracuje příkaz a hned na něj odpoví, dvojtečka znamená jen „zpracuj a neodpovídej, pokud nedošlo k chybě“. Za jeden znak „>“ můžeš najednou napsat více příkazů. Odpovědi Maplu mohou být rozličné — text, výraz, graf, chybové hlášení, aj. „Textové“ odpovědi Maplu jsou většinou modré nebo černé. Chybová klášení jsou modrá nebo fialová.

Maple pracuje s různými objekty (např. s čísly, proměnnými, výrazy, množinami, funkcemi, ...) a ty mohou mít nějaká jména. Přiřazení jména objektu uděláš většinou takto:

>*jméno objektu*:=*objekt*;

Pozor! Maple je case-senzitivní, tj. rozlišuje mezi velkými a malými písmeny. Maple si pamatuje jména všech objektů, které jsi vytvořil. Změnit objekt přiřazený ke jménu můžeš opětovným přiřazením nebo příkazem `restart`. Tento příkaz mimo jiné způsobí zapomenutí všech dosud uživatelem definovaných jmen.

Základní symboly a objekty, které budeš často potřebovat

Čísla: píší se v desítkové soustavě, desetinná část se odděluje tečkou.

Předdefinované konstanty: `Pi`, `I`, `infinity`, (π, imaginární jednotka, ∞).

Vlastní proměnné: nesmí být pojmenovány jako předdefinované konstanty.

Vlastní řecké proměnné: `alpha`, `beta`, `Delta`, `mu`, `nu`, ... tj. α, β, Δ, μ, ν.

Intervaly: `a..b`, `RealRange(Open(1),infinity)`, tj. $[a, b]$, $(1, \infty)$.

Základní množiny: `natural`, `integer`, `rational`, `real`, `complex`: $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$.

Základní operace: `+`, `-`, `*`, `/`, `^`, `!`.

Základní relace: `=`, `<=`, `>=`, `<>`.

Závorky: kulaté ve výrazech, hranaté jako indexy, např. `a[1]` je a_1.

Množiny: {`1,2,9..15`}.

Uspořádaná k-tice, seznam: `[1,2,2,1]`; i-tý prvek seznamu dostaneme použitím indexu.

Funkce několika proměnných: (*jména proměnných oddělená čárkou*) `->` *vzorec*.

Součet: `sum(`*vzorec*`,`*indexová proměnná* `=` *interval celých čísel*`)`.

Součin: `product(`*vzorec*`,`*indexová proměnná* `=` *interval celých čísel*`)`.

Výčet prvků posloupnosti: `seq(`*člen posloupnosti*`,`*interval celých čísel*`)`.

Výsledek posledního příkazu: `%`.

Balíky

Maple obsahuje velmi mnoho příkazů. Aby se v nich vůbec někdo vyznal, jsou rozděleny do balíků. Seznam balíků je možné najít v nápovědě (Index/packages). Rovněž seznam funkcí a příkazů v balíku a jejich popis zjistíš z nápovědy.

Příkaz `with(`*jméno balíku*`)` definuje funkce obsažené v balíku. Po jeho provedení můžeš s těmito funkcemi pracovat. Pokud chceš použít třeba jen jednu funkci z celého balíku, napíšeš *jméno balíku*`[`*jméno funkce*`]`. Např. `combinat[fibonacci](10)` zavolá funkci fibonacci v kombinatorickém balíku a ta vypíše 10. číslo Fibonacciho posloupnosti. Mnoho příkazů je nadefinováno již při spuštění Maplu, není tedy třeba Maplu říkat, v jakém jsou balíku.

Práce s čísly, proměnnými a výrazy

`evalf(`*výraz*`,`*počet platných číslic*`)` pokusí se numericky vypočítat výraz a výsledek zaokrouhlí.
`isprime(`*číslo*`)` zjistí, zda zadané číslo je prvočíslem.
`ifactor(`*číslo*`)` zobrazí prvočíselný rozklad čísla.
`expand(`*výraz*`)` roznásobí, co roznásobit lze.
`factor(`*výraz*`)` pokusí se rozložit výraz na faktory.
`simplify(`*výraz*`)` zjednoduší výraz.
`subs(`*seznam rovností*`,`*výraz*`)` ve výrazu *výraz* „syntakticky“ substituuje všechny výrazy v seznamu.
`algsubs(`*co* `=` *za co*`,`*výraz*`)` ve výrazu *výraz* substituuje *co* za *za co*.
další příkazy: `eval`, `assume`, `collect`, `combine`, `radsimp`, `igcd`, `ilcm`, `ithprime`, `nextprime`, `prevprime`, `numtheory[tau]`, `numtheory[cfrac]`, `combinat[fibonacci]`, ...

Rovnice, nerovnice a jejich soustavy

`solve(`*množina rovnic nebo nerovnic*`,`*množina proměnných*`)` řeší soustavy rovnic a nerovnic.
`isolve(`*množina diofantických rovnic*`,`*množina proměnných*`)` řeší soustavy diofantických rovnic.
`RootOf(`*rovnice*`)` reprezentuje řešení rovnice.
`allvalues(`*výraz obsahující* `RootOf)` pokud to Maple umí, zobrazí řešení reprezentované symbolem RootOf.
další příkazy: `rsolve, ...`

Posloupnosti, řady, funkce

Vestavěné základní funkce (jejich seznam je v nápovědě Mathematics/Basic Mathematics/Initially known functions): `abs`, `sqrt`, `signum`, `max`, `min`, `binomial`, `ceil` (horní celá část), `floor` (dolní celá část), `Im`, `Re`, `sin`, `cos`, `tan` (nikoliv `tg`!), `arcsin`, `arccos`, `arctan`, `sinh`, `cosh`, `tanh`, `arcsinh`, `arccosh`, `arctanh`, `exp`, `ln`, `log` (stejné jako `ln`), ...

Funkci i posloupnost lze definovat operátorem `->`, funkci pak lze navíc definovat i „po částech“, tj. na různých intervalech různě. Provést to lze příkazem `piecewise(`p_1`,`f_1`,`p_2`,`f_2`,...,`p_n`,`f_n`,`*funkce*`)`, kde $p_1,\ldots,p_n$ jsou podmínky složené z logických spojek `and`, `or` a rovností nebo nerovností, $f_1,\ldots,f_n$ jsou funkce definované na intervalech či v bodech daných podmínkami $p_1,\ldots,p_n$. V bodech, které nesplňují žádnou podmínku můžeš funkci definovat funkcí *funkce*. Např. `piecewise(x>-2 and x<1, x^2, x)`

Pokud je horní hranicí intervalu v součtu konstanta `infinity`, je suma chápána jako řada.
další příkazy: `unapply`, `@`, `@@`, `Heaviside`, `GAMMA`, ...

Grafy reálných funkcí reálné proměnné

`plot(`*seznam funkcí*`,`*proměnná* `=` *interval*`,`*volba1*`,`*volba2*`,...)`

Na intervalu *interval* zobrazí grafy funkcí proměnné *proměnná* uvedených v seznamu.

Vzhled grafů můžeš ovlivnit volbami (jejich plný výčet je v nápovědě Graphics/2D/Options/Overview):
`color=`*seznam barev funkcí* definované barvy jsou např. `red`, `green`, `magenta`, `blue`, ...
`discont=true` pokusí se najít nespojitosti funkcí.

`style=`*seznam stylů* způsob kreslení funkcí (např. `point` — bodově, `line` — spojitě).
`labels=[`*název*`,`*název*`]` pojmenuje souřadnicové osy.
`title=`*název grafu* pojmenuje graf.
další příkazy: `textplot`, `animate`, `display`, `student[leftbox]`, `student[rightbox]`, `student[showtangent]`, ...

Limity, derivace, primitivní funkce, určité integrály

`limit(`*funkce*`,`*proměnná* `=` *hodnota*`,`*směr*`)`
Najde limitu funkce zprava, *směr*=`right`, či zleva, *směr*=`left`, ve vlastním bodě *hodnota*, či v nekonečnu.
`limit(`*člen posloupnosti*`,`*proměnná* `= infinity)` Spočítá limitu posloupnosti.
`diff(`*funkce*`,`*seznam proměnných*`)` Zderivuje funkci podle proměnných v seznamu.
`int(`*funkce*`,`*proměnná*`)` Spočítá primitivní funkci k zadané funkci.
`int(`*funkce*`,`*proměnná* `=` *interval*`)` Spočítá určitý integrál na daném intervalu.

Euklidovská rovinná geometrie (balík geometry)

Obsahuje příkazy pro vytvoření geometrických objektů a pro práci s nimi. Jména objektům se zadávají jako parametry příkazů.
`point(`*název*`,`x`,`y`)` Vytvoří bod o souřadnicích x, y.
`line(`*název*`,`*rovnice* nebo *seznam bodů*`)` Vytvoří přímku danou rovnicí nebo dvěma body.

V nápovědě můžeš najít příkazy vytvořující trojúhelník (`triangle`), čtverec (`square`), kružnici (`circle`), elipsu (`ellipse`), parabolu (`parabola`), hyperbolu (`hyperbola`), obecnou kuželosečku (`conic`), ...

Na vlastnosti geometrického objektu se můžeš zeptat příkazem `detail(`*název objektu*`)`, vykreslit objekty můžeš příkazem `draw(`*seznam objektů*`)`. Maple umí najít i objekty speciálních vlastností např. těžiště trojúhelníka (`centroid`), ortocentrum trojúhelníka (`orthocenter`), kružnici opsanou trojúhelníku (`circumcircle`), kružnici vepsanou trojúhelníku (`incircle`), Appoloniovu kružnici (`Appolonius`). Umí také zjišťovat rozličné vlastnosti objektů, např. rovnoběžnost přímek (`AreParallel`), kolmost přímek (`ArePerpendicular`), kolmost kružnic (`AreOrthogonal`), podobnost trojúhelníků (`AreSimilar`), vzdálenost bodů a bodu a přímky (`distance`), obsah (`area`), a mnohé další. Objekty také můžeš nechat zobrazit ve středové či osové souměrnosti (`reflection`), homotetii (`homothety`), posunout (`translation`), otočit (`rotation`), kruhově zinvertovat (`inversion`) a jinak transformovat.
`convert(`*objekt*`,`*forma*`,`*další parametry*`)`.

Jak již název napovídá, tento příkaz převádí objekt z jedné formy do zadané:

- desetinné číslo → zlomek: `convert(`*číslo*`,fraction)`
- číslo → binární číslo: `convert(`*číslo*`,binary)`
- desítková soustava → jiná soustava: `convert(`*číslo*`,base,`*základ*`)`
- racionální funkce → parciální zlomky: `convert(`*rac. funkce*`,parfrac,`*proměnná*`)`
- funkce → funkce definovaná „po částech“: `convert(`*funkce*`,piecewise)`