Osnovno o *Mathematici*

Marko Petkovšek, Bor Plestenjak

1 Sistemi za simbolno računanje

Računalnike so razvili z namenom, da bi olajšali numerično reševanje problemov. Kasneje se je pokazalo, da so računalniki idealni za obdelavo poslovnih in drugih informacij. Že v 60. letih pa so se začele uresničevati sanje številnih matematikov, da bi računalnik ne služil le za "mletje števil", ampak tudi kot pripomoček za ustvarjanje "prave" matematike. Takrat so začeli nastajati prvi sistemi za simbolno računanje, npr. *Macsyma* v ZDA in *Reduce* v Angliji. Kasneje so se jim pridružili še drugi, npr. havajski *Derive*, kanadski *Maple*, Wolframova *Mathematica*, avstralska *Magma* in IBM/NAG-ov *Axiom*. Našteti sistemi so splošni, saj skušajo pokriti čimveč področij matematike. Razen teh pa obstaja še mnogo specializiranih sistemov za posamezna področja, npr. *Pari* za teorijo števil, *Gap* in *Naughty* za teorijo grup, *Macaulay* za komutativno algebro itd. Neprestano se pojavljajo novi specializirani sistemi.

Prednosti *Mathematice* so:

- dober uporabniški vmesnik z bogato grafiko in zvokom,
- izredno pester izbor programskih konstruktov, ki omogočajo različne sloge programiranja,
- preprosta sintaksa, ki nudi univerzalen okvir za delo z objekti najrazličnejših tipov,
- dostopnost na široki paleti strojne opreme, od osebnih računalnikov prek delovnih postaj do superračunalnikov.

Za *Mathematico* bomo odslej uporabljali uveljavljeno kratico "Mma".

2 Zvezki

Zvezki so datoteke, ki jih kreira Mma. Zvezek je sestavljen iz celic, ki so vgnezdene druga v drugi – ima torej drevesno strukturo. Globina gnezdenja je načeloma poljubna. Celice so prikazane z oglatimi zaklepaji na desni strani zaslona. Celico z enim klikom na oklepaj označimo, z dvema pa zapremo (če je odprta) ali odpremo (če je zaprta). Celice so različnih tipov: vhodne, izhodne, tekstovne, naslovne, grafične itd. Zvezek lahko torej oblikujemo kot elektronsko knjigo, ki je razdeljena na celice – poglavja, ta so razdeljena naprej na razdelke in podrazdelke, ki vsebujejo tekst, grafiko, računske primere in definicije funkcij. Uporabnik oz. bralec lahko selektivno odpira tiste sestavne dele, ki ga zanimajo, izvaja primere, dodaja in preskuša svoje lastne primere ali pa izbrane celice izpiše na tiskalnik.

Izračun izraza sprožimo s pritiskom na SHIFT-ENTER, medtem ko ENTER le prelomi vrstico v dve. Nove izraze praviloma vnašamo na koncu zvezka pod črto. Lahko pa jih vnašamo tudi med prejšnjimi celicami, ali pa editiramo kako staro celico. Zaporedje vhodnih vrstic v zvezku se potem ne ujema več s časovnim zaporedjem, ki ga odražajo njihove številke. Pri definiciji prejšnjega rezultata Mma vedno upošteva časovno in ne prostorsko zaporedje.

3 Delo z Mma

Mma je interaktivna: uporabnik vnese izraz, Mma vrne njegovo vrednost. Izraze Mma številči od 1 naprej, tako da se lahko sklicujemo na prejšnje rezultate. Vneseni izraz je označen z `In[n]`, izračunana vrednost pa z `Out[n]`. Npr.:

```
In[1]:= 1+1

Out[1]= 2

In[2]:= Sin[Pi/3]

        Sqrt[3]
Out[2]= -------
           2

In[3]:= Factor[x^3 - 1]

                              2
Out[3]= (-1 + x) (1 + x + x )
```

Mma je torej zelo sposoben kalkulator. Vgrajenih ima namreč blizu 1000 ukazov, uporabnik pa lahko iz njih poljubno sestavlja nove. Besedi "ukaz" in "funkcija" bomo uporabljali sinonimno.

Opazimo:

1. Imena vgrajenih funkcij se začenjajo z veliko začetnico.
2. Argumenti funkcij so obdani z oglatimi oklepaji, npr. `f[x]` in ne `f(x)`, kot smo sicer vajeni.

3.1 Oklepaji

- (): Okrogle oklepaje uporabljamo le za **grupiranje**.
- [ ]: Z oglatimi oklepaji obdamo argumente **funkcije**.
- {}: V zavitih oklepajih naštejemo elemente **seznama**.
- [[ ]]: V dvojnih oglatih oklepajih navedemo **indeks** elementa seznama.
- $(* *)$: Okrogli oklepaji z zvezdico vsebujejo **komentar**.

Zgled:

```
In[1]:= s = {a,b,c,d,e}  (* definiramo seznam s 5 elementi *)

Out[1]= {a, b, c, d, e}

In[2]:= s[[4]]

Out[2]= d
```

3.2 Uporaba prejšnjih rezultatov

Oznaka `%` pomeni prejšnji rezultat, oznaka `%%` predprejšnjega itd. Oznaka `%n` pomeni isto kot `Out[n]`, torej vrednost n-tega izraza po vrsti od zagona programa. Zgled:

```
In[1]:= 2^100

Out[1]= 1267650600228229401496703205376

In[2]:= 3^100

Out[2]= 515377520732011331036461129765621272702107522001

In[3]:= 6^100

Out[3]= 65331862350007090609669026715805782053714371047295487154307196636949\
7
>       141477376

In[4]:= % - %% %%%

Out[4]= 0

In[5]:= Log[2, %1]  (* Log[b,x] je logaritem x pri osnovi b *)

Out[5]= 100
```

Opazimo:

1. Mma zna računati s poljubno (?) velikimi števili.
2. Znak \ za skupino števk na koncu vrstice pomeni, da se število nadaljuje v naslednji vrstici.

3.3 Imena, operacije in relacije

Ime je v Mma zaporedje črk in števk, ki se začne s črko. Dolžina imena ni omejena. Mma loči male in velike črke. Imena vgrajenih funkcij se začno z veliko črko; če so zlepljena iz več besed, se praviloma vsaka od njih začne z veliko črko, npr. `InterpolatingPolynomial`. Izjemoma se imena nekaterih vgrajenih funkcij, ki opisujejo karakteristike sistema, začno z znakom `$`. Funkcija `$Version` npr. pove, katero verzijo Mma uporabljamo.

Mma uporablja simbole `+, -, *, /, ^` za operacije seštevanja, odštevanja, množenja, deljenja in potenciranja. Znak `*` lahko nadomestimo s presledkom ali tudi izpustimo, če to ne vodi do drugačne interpretacije izraza, kot jo želimo.

Mma uporablja simbole `==, !=, <, <=, >, >=` za relacije "enak", "različen", "manjši", "manjši ali enak", "večji" in "večji ali enak". Vrednost izraza oblike `a R b`, kjer je `R` ena od naštetih relacij, je `True`, `False` ali pa kar izraz sam, če Mma ne more ugotoviti, ali relacija velja ali ne. Mma pozna še relaciji "sintaktično enak", `===`, in "sintaktično različen", `=!=`, ki pa vedno vrneta vrednost `True` oziroma `False`: če Mma ne vé, ali sta izraza enaka ali ne, sta gotovo sintaktično različna. Zgled:

```
In[1]:= a = 1

Out[1]= 1

In[2]:= b = 3

Out[2]= 3

In[3]:= a < b

Out[3]= True

In[4]:= a < c

Out[4]= 1 < c
```

```
In[5]:= c == 2

Out[5]= c == 2

In[6]:= c === 2

Out[6]= False

In[7]:= c =!= 2

Out[7]= True
```

3.4 Prirejanje in brisanje vrednosti

Spremenljivkam priredimo vrednost z operatorjem `=`, npr. `a = x`. Tu je `a` ime spremenljivke, `x` pa poljuben izraz (z naslednjo izjemo: Če spremenljivka `a` ob času prirejanja še nima definirane vrednosti, izraz `x` ne sme vsebovati `a`, sicer lahko pride do neskončne zanke). Odslej bo Mma spremenljivko `a` v vsakem izrazu, v katerem nastopa, nadomestila z izrazom `x` – dokler pač spremenljivki `a` ne priredimo nove vrednosti.

Če želimo zbrisati vrednost spremenljivke `a`, napišemo `a = .` in spremenljivka `a` je spet svež simbol kot na začetku. V nekaterih funkcijah nastopajo namreč *vezane spremenljivke*. Takšna je npr. integracijska spremenljivka integrala ali neznanka pri reševanju enačb. Pogosto se nam dogaja, da začnemo nov račun in uporabimo spremenljivko, ki smo ji prej priredili kako vrednost, kot vezano. To ima lahko hude posledice, ki pa jih odpravi brisanje vrednosti vezanih spremenljivk.

3.5 Substitucija, transformacijska pravila in reševanje enačb

Izraz oblike `x -> a`, kjer sta `x` in `a` poljubna izraza, je *transformacijsko pravilo*. Leva stran `x` je pogosto kar spremenljivka. Transformacijsko pravilo uporabimo na izrazu `b` tako, da zapišemo `b /. x -> a`. Rezultat je izraz `b`, v katerem so podizrazi oblike `x` nadomeščeni z izrazom `a`. Namesto posameznega pravila lahko na danem izrazu uporabimo tudi seznam več transformacijskih pravil. Zgled:

```
In[1]:= 2 x^2 y + 3 x + 1 /. {x -> 1, y -> -2}

Out[1]= 0
```

Vgrajene funkcije, ki rešujejo enačbe, praviloma vračajo rezultate v obliki seznamov transformacijskih pravil za neznanke. Taka je npr. funkcija `Solve` za reševanje sistemov algebraičnih (pa tudi nekaterih drugih) enačb. `Solve` ima dva argumenta, prvi je seznam enačb, drugi seznam spremenljivk. če je enačba ali spremenljivka ena sama, je ni treba postaviti v seznam. Ker ima sistem enačb v splošnem več rešitev, `Solve` vrne seznam rešitev, posamezna rešitev pa je seznam transformacijskih pravil za neznanke. Zgledi:

```
In[1]:= Solve[x^2 + x - 2 == 0, x]

Out[1]= {{x -> -2}, {x -> 1}}

In[2]:= x^2 + x - 2 /. %

Out[2]= {0, 0}

In[3]:= Solve[{x^2 + 5 y^2 == 5, 2 x^2 + y^2 == 2}, {x, y}]

                -Sqrt[5]       -2 Sqrt[2]          -Sqrt[5]       2 Sqrt[2]
Out[3]= {{x -> --------, y -> ----------}, {x -> --------, y -> ---------},
                   3               3                  3              3

          Sqrt[5]       -2 Sqrt[2]          Sqrt[5]       2 Sqrt[2]
```

```
>   {x -> -------, y -> ----------}, {x -> -------, y -> ---------}}
              3              3                3              3

In[4]:= {x^2 + 5 y^2 == 5, 2 x^2 + y^2 == 2} /. %

Out[4]= {{True, True}, {True, True}, {True, True}, {True, True}}

In[5]:= Solve[2 x + 1 == 2 x, x]

Out[5]= {}

In[6]:= Solve[(x + 1)(x - 1) == x^2 - 1, x]

Out[6]= {{}}
```

Kot vidimo, je rešitev zapisana v taki obliki, da je vrednosti neznank preprosto vstaviti v kak izraz - npr. v enačbo ali v sistem enačb in preveriti, ali je zares izpolnjen(a). Vrstici `5` in `6` kažeta, kaj vrne `Solve`, če dobi protislovno enačbo ali pa identiteto.

Primer: Izračunajmo z Mma ploščino lika, ki ga oklepata parabola $f(x) = 1+3x-x^2$ in premica $g(x) = 2x - 1$.

```
In[7]:= f[x_] := 1 + 3 x - x^2

In[8]:= g[x_] := 2 x - 1

In[9]:= Solve[{f[x] == g[x]}, x]

Out[9]= {{x -> -1}, {x -> 2}}

In[10]:= Integrate[f[x] - g[x], {x, x /. %[[1]], x /. %[[2]]}]

         9
Out[10]= -
         2
```

Opazimo:

1. Funkcijo definiramo z operatorjem `:=`, argumentom funkcije pa moramo na levi strani definicije dodati podčrtaj.
2. Določeni integral funkcije $f(x)$ v mejah od a do b izračunamo z ukazom `Integrate[f[x], {x, a, b}]`.

4 Aritmetika

Mma razlikuje cela, racionalna, realna in kompleksna števila. Racionalna števila imajo števec in imenovalec, ki sta celi števili. Z realnimi števili so mišljena, kot običajno pri programskih jezikih, števila v pomični piki. Komponenta kompleksnega števila je lahko cela, racionalna ali realna.

Funkcije za doseganje komponent:

- `Numerator[r]` vrne števec racionalnega števila `r`,
- `Denominator[r]` vrne imenovalec racionalnega števila `r`,
- `Re[c]` vrne realno komponento kompleksnega števila `c`,
- `Im[c]` vrne imaginarno komponento kompleksnega števila `c`.

Pomembna funkcija je `N[x]`, ki vsa števila v izrazu `x` pretvori v realna števila.

4.1 Naključna števila

Ukaz	**Opis**
`Random[]`	Naključno izbrano realno število med 0 in 1.
`Random[Integer]`	Naključno izbrano število iz množice $\{0, 1\}$.
`Random[Integer, {`a`, `b`}]`	Naključno izbrano celo število med a in b.
`Random[Real, {`a`, `b`}]`	Naključno izbrano realno število med a in b.
`SeedRandom[`n`]`	Inicializacija generatorja naključnih števil.

5 Seznami

Elemente seznamov pišemo med zavitimi oklepaji, npr. `{a, b, c, a, 1, 2}`.

Ukaz	**Opis**
`List[`*a1, a2, ...* `]`	isto kot `{`*a1, a2, ...* `}`.
`Range[`*min, max, korak* `]`	Sestavi seznam števil (aritmetično zaporedje).
`Table[`*izr, iterator* `]`	Tabelira *izr* pri vrednostih, ki jih določa *iterator*.
`Array[`*f, dim* `]`	Sestavi tabelo vrednosti funkcije f razsežnosti *dim*.
`Length[`*sez* `]`	Vrne dolžino seznama *sez*.
`First[`*sez* `]`	Vrne prvi element seznama *sez*.
`Last[`*sez* `]`	Vrne zadnji element seznama *sez*.
`Rest[`*sez* `]`	Vrne seznam *sez* brez prvega elementa.
`Take[`*sez, n* `]`	Vrne seznam prvih n elementov seznama *sez*.
`Take[`*sez, -n* `]`	Vrne seznam zadnjih n elementov seznama *sez*.
`Drop[`*sez, n* `]`	Vrne seznam *sez* brez prvih n elementov.
`Drop[`*sez, -n* `]`	Vrne seznam *sez* brez zadnjih n elementov.
`Part[`*sez, i* `]`ali *sez*`[[i]]`	Vrne i-ti element seznama *sez*.
`Select[`*sez, f* `]`	Vrne seznam elementov *sez*, pri katerih je predikat f izpolnjen.
`Insert[`*sez, el, i* `]`	V seznam *sez* vstavi element *el* na i-to mesto.
`Delete[`*sez, i* `]`	Iz seznama *sez* zbriše i-ti element.
`ReplacePart[`*sez, el, i* `]`	V seznamu *sez* nadomesti element na i-tem mestu z elementom *el*.
`Reverse[`*sez* `]`	Preobrne seznam *sez*.
`RotateLeft[`*sez, n* `]`	Zasuče *sez* za n mest v levo.
`RotateRight[`*sez, n* `]`	Zasuče *sez* za n mest v desno.
`Sort[`*sez* `]`	Uredi elemente seznama *sez*.
`Permutations[`*sez* `]`	Sestavi seznam vseh permutacij seznama *sez*.
`Append[`*sez, el* `]`	Vrne seznam *sez*, ki mu je kot zadnji dodan element *el*.
`Prepend[`*sez, el* `]`	Vrne seznam *sez*, ki mu je kot prvi dodan element *el*.
`AppendTo[`*sez, el* `]`	Seznamu *sez* na koncu doda element *el*.
`PrependTo[`*sez, el* `]`	Seznamu *sez* na začetku doda element *el*.
`Join[`*s1, s2, ...* `]`	Stakne sezname *s1, s2, ...*
`Union[`*s1, s2, ...* `]`	Vrne unijo množic *s1, s2, ...*
`Intersection[`*s1, s2, ...* `]`	Vrne presek množic *s1, s2, ...*
`Complement[`*s1, s2, ...* `]`	Vrne komplement unije preostalih množic v množici *s1*.
`Partition[`*sez, n* `]`	Razdeli seznam *sez* na podsezname dolžine n.
`Partition[`*sez, n, k* `]`	Razdeli seznam *sez* na podsezname dolžine n z zamikom k.
`Flatten[`*sez* `]`	Odpravi vgnezdene sezname v seznamu *sez*.
`Map[`*f, sez* `]`ali `f /@` *sez*	Uporabi funkcijo f na vseh elementih seznama *sez*.
`Apply[`*f, sez* `]`ali `f @@` *sez*	Nadomesti glavo seznama *sez* z f.
`Nest[`*f, x, n* `]`	Vrne $f(f(\dots f(x)\dots))$, kjer f nastopa n-krat.
`NestList[`*f, x, n* `]`	Vrne seznam $\{x, f(x), f(f(x)), \dots, f(f(\dots f(x)\dots))\}$.
`FixedPoint[`*f, x* `]`	Vrne "limito" zaporedja $x, f(x), f(f(x)), \dots$
`Fold[`*f, x,* $\{a_1, a_2, \dots, a_n\}$`]`	Vrne $f(\dots f(f(x, a_1), a_2), \dots, a_n)$.
`FoldList[`*f, x,* $\{a_1, a_2, \dots, a_n\}$`]`	Vrne seznam $\{x, f(x, a_1), \dots, f(\dots f(f(x, a_1), a_2), \dots, a_n)\}$.

6 Teorija števil

- `Mod[n, k]`: ostanek pri deljenju n s k
- `Quotient[n, k]`: celoštevilski količnik pri deljenju n s k
- `GCD[n1, n2, ...]`: največja skupna mera števil n_1, $n_2, \ldots$
- `LCM[n1, n2, ...]`: največji skupni delitelj števil n_1, $n_2, \ldots$
- `ExtendedGCD[n, k]`: razširjeni Evklidov algoritem
- `PrimeQ[n]`: preveri, ali je n praštevilo
- `Prime[n]`: n-to praštevilo
- `PrimePi[n]`: število praštevil, ki so manjša od n
- `FactorInteger[n]`: razcep števila n na prafaktorje v obliki seznama parov {praštevilo, eksponent}
- `Divisors[n]`: seznam deliteljev števila n
- `DivisorSigma[k, n]`: vsota k-tih potenc deliteljev števila n
- `EulerPhi[n]`: Eulerjeva funkcija $\varphi(n)$
- `MoebiusMu[n]`: Möbiusova funkcija $\mu(n)$
- `JacobiSymbol[n, m]`: Jacobijev simbol
- `PowerMod[a, b, n]`: $a^b \bmod n$

7 Kombinatorika

- `Factorial[n]` ali `n!`: faktoriela (fakulteta)
- `Factorial2[n]` ali `n!!`: dvojna faktoriela
- `Binomial[x,k]`: binomski koeficient $\binom{x}{k}$
- `Multinomial[n1, n2, ..., nk]`: polinomski koeficient $\binom{n_1+n_2+\cdots+n_k}{n_1,n_2,\ldots,n_k}$
- `Pochhammer[x, k]`: rastoča potenca $x(x+1)\cdots(x+k-1)$
- `StirlingS1[n, k]`: Stirlingovo število 1. vrste $s_n^{(k)}$
- `StirlingS2[n, k]`: Stirlingovo število 2. vrste $S_n^{(k)}$
- `BernoulliB[n]`: Bernoullijevo število B_n
- `BernoulliB[n, x]`: Bernoullijev polinom $B_n(x)$
- `EulerE[n]`: Eulerjevo število E_n

- `EulerE[n, x]`: Eulerjev polinom $E_n(x)$
- `PartitionsP[n]`: število razčlenitev (particij) števila n
- `PartitionsQ[n]`: število razčlenitev števila n na različne člene
- `Permutations[s]`: seznam vseh permutacij seznama s

8 Izrazi

8.1 Sintaksa izrazov

Osnovni in pravzaprav tudi edini sintaktični element Mma je *izraz*. Kategorija izrazov v Mma obsega vse – ne le aritmetičnih izrazov v ožjem smislu, ampak tudi tisto, čemur v drugih programskih jezikih rečemo stavki ali ukazi. Izrazi so:

- *nesestavljeni*, ali
- *sestavljeni*.

Nesestavljene izraze imenujemo tudi *atomi*. Mma pozna šest vrst atomov:

- `Integer` (celo število),
- `Rational` (racionalno število),
- `Real` (približno realno število),
- `Complex` (približno ali točno kompleksno število),
- `Symbol` (simbol oziroma ime),
- `String` (niz).

Sintaksa števil je običajna. Eksponentni zapis realnega števila dobimo tako kot v matematiki, npr. `6.28*10^(-23)` (zvezdico lahko seveda zamenjamo s presledkom). Kompleksno število i pišemo `I`; komponenti kompleksnega števila sta lahko celi, racionalni ali realni (lahko različnih tipov). Sintakso simbolov oziroma imen smo že spoznali. Nize pišemo med dvojnimi narekovaji, npr. `"to je niz"`.

Funkcije `AtomQ`, `IntegerQ`, `NumberQ` povedo, ali je njihov argument atom, celo število oziroma število (atom enega od prvih štirih tipov). Tip atoma nam pove funkcija `Head`.

Sestavljen izraz je oblike

```
g[a1, a2, ..., an],
```

kjer je `g` *glava*, `x1`, `x2`, ..., `xn` pa so *argumenti*. Pri tem so lahko glava in argumenti spet poljubni izrazi. Število argumentov n sme biti enako nič. Primer je izraz `Random[]`, ki vrne naključno realno število med 0 in 1.

Če je `x` sestavljen izraz, nam `Head[x]` vrne glavo, `Length[x]` število argumentov, `x[[i]]` pa i-ti argument izraza `x`.

Opisali smo *notranjo sintakso* izrazov v Mma. Pri komuniciranju z uporabnikom pa za najpogosteje uporabljane operacije in relacije Mma razume in uporablja *zunanjo sintakso*, ki je bližja običajni matematični rabi. Nekaj primerov:

zunanja sintaksa	notranja sintaksa
`a + b + c`	`Plus[a, b, c]`
`a b c`	`Times[a, b, c]`
`a^b`	`Power[a, b]`
`{a, b, c}`	`List[a, b, c]`
`a == b`	`Equal[a, b]`
`a = b`	`Set[a, b]`
`a /. x -> y`	`ReplaceAll[a, Rule[x, y]]`

Notranjo sintakso izraza `x` si lahko ogledamo z ukazom `FullForm[x]`.

8.2 Semantika izrazov

Semantiko izrazov opišemo operacijsko, torej tako, da podamo postopek, po katerem Mma izračuna vrednost izraza. Glavni značilnosti postopka določanja vrednosti sta:

- *računanje od znotraj navzven* (najprej se izračunajo vsi argumenti, nato se izračuna vrednost funkcije),
- *vztrajno računanje* (vrednost izraza se računa spet in spet, dokler se še kaj spreminja).

Postopek računanja vrednosti $v(x)$ izraza x je v zelo grobem približku takle:

1. Če je x atom, je $v(x) = x$.
2. Če je $x = g[a_1, a_2, \ldots, a_n]$, Mma izračuna $v(x)$ takole:
   (a) Rekurzivno izračuna $h = v(g)$ in $b_i = v(a_i)$ za $i = 1, 2, \ldots, n$. Naj bo $y = h[b_1, b_2, \ldots, b_n]$.
   (b) Če v podatkovni bazi prepisovalnih pravil ni nobenega pravila, katerega leva stran bi se lahko prilagodila izrazu y, je $v(x) = y$. Sicer Mma poišče prvo takšno pravilo in vrednosti, ki jih dobijo parametri pri prilagajanju, po imenu prenese v desno stran pravila. Tako nastane nov izraz z, ki ga Mma izračuna rekurzivno, in vrne $v(z)$ kot vrednost izraza x.

Kasneje bomo opisali nekatere izjeme pri postopku izračunavanja izrazov.

Zgled: Definirajmo funkciji `f[x]` in `g[x,y]` ter izračunajmo `g[f[a], g[f[b], f[c]]]`.

```
In[1]:= f[x_] := 3x

In[2]:= g[x_, y_] := x + y

In[3]:= g[f[a], g[f[b], f[c]]]

Out[3]= 3 a + 3 b + 3 c
```

Z ukazom `Trace[x]` si lahko ogledamo vrednosti vmesnih izrazov pri računanju vrednosti izraza `x`.

```
In[4]:= Trace[g[f[a], g[f[b], f[c]]]]

Out[4]= {{f[a], 3 a}, {{f[b], 3 b}, {f[c], 3 c}, g[3 b, 3 c], 3 b + 3 c},

>    g[3 a, 3 b + 3 c], 3 a + (3 b + 3 c), 3 a + 3 b + 3 c}
```

9 Grafika

9.1 Dvorazsežna grafika

- `ListPlot[{v1, v2, ...}]` nariše točke s koordinatami $(1, v_1)$, $(2, v_2)$, ...
- `ListPlot[{{x1,y1}, {x2,y2}, ...}]` nariše točke s koordinatami (x_1, y_1), (x_2, y_2), ...
- `Plot[y, {x, xmin, xmax}]` nariše eksplicitno podano krivuljo $y = y(x)$ na intervalu $x_{\min} \leq x \leq x_{\max}$.
- `Plot[{y1, y2, ...}, {x, xmin, xmax}]` nariše več krivulj hkrati.
- `ParametricPlot[{x, y}, {t, tmin, tmax}]` nariše parametrično podano krivuljo $x = x(t)$, $y = y(t)$ na intervalu $t_{\min} \leq t \leq t_{\max}$.
- `ParametricPlot[{{x1, y1}, {x2, y2}, ...}, {t, tmin, tmax}]` nariše več krivulj hkrati.
- `DensityPlot[z, {x, xmin, xmax}, {y, ymin, ymax}]` prikaže ploskev $z = z(x, y)$ s senčenjem kvadratne mreže na pravokotniku $x_{\min} \leq x \leq x_{\max}$, $y_{\min} \leq y \leq y_{\max}$ z različnimi odtenki sivine. Bela barva ustreza največji vrednosti funkcije z, črna pa najmanjši.
- `ContourPlot[z, {x, xmin, xmax}, {y, ymin, ymax}]` prikaže ploskev $z = z(x, y)$ nad pravokotnikom $x_{\min} \leq x \leq x_{\max}$, $y_{\min} \leq y \leq y_{\max}$ s plastnicami in s senčenjem območij med sosednjima plastnicama z različnimi odtenki sivine. Bela barva ustreza največji vrednosti funkcije z, črna pa najmanjši.

Grafične funkcije sprejemajo še poljubno število dodatnih argumentov, ki jih imenujemo *opcije*. Opcija ima obliko `ime opcije -> vrednost opcije`. Tako npr. ukaz

```
Plot[Tan[x], {x,-6,6}, AspectRatio -> Automatic, PlotRange -> {-7,7}]
```

nariše graf funkcije tg x na intervalu $-6 \leq x \leq 6$, pri čemer prva opcija zahteva, naj bosta enoti na abscisni in ordinatni osi enako dolgi, druga pa, naj se prikaže le tisti del grafa, za katerega je $-7 \leq y \leq 7$.

Opcije, ki jih pozna posamezna funkcija, npr. `f`, in njihove privzete vrednosti, dobimo z ukazom `Options[f]`.

Ko je grafični objekt že narisan, ga lahko ponovno prikažemo z ukazom `Show`, pri čemer lahko spreminjamo vrednosti nekaterih opcij. Funkcijo tg x narišemo lahko najprej brez opcij

```
Plot[Tan[x], {x,-6,6}],
```

ker pa z rezultatom nismo zadovoljni, prikažemo sliko še enkrat z drugačnimi vrednostmi nekaterih opcij:

```
Show[%, AspectRatio -> Automatic, PlotRange -> {-7,7}]
```

9.2 Trirazsežna grafika

- `ListPlot3D[m]` nariše ploskev, ki ima v tockah kvadratne mreže višine, ki jih podaja matrika m.
- `Plot3D[z, {x, xmin, xmax}, {y, ymin, ymax}]` nariše eksplicitno podano ploskev $z = z(x, y)$ nad pravokotnikom $x_{\min} \leq x \leq x_{\max}$, $y_{\min} \leq y \leq y_{\max}$.
- `ParametricPlot3D[{x,y,z}, {u, umin, umax}, {v, vmin, vmax}]` nariše parametrično podano ploskev $x = x(u, v)$, $y = y(u, v)$, $z = z(u, v)$ za $u_{\min} \leq u \leq u_{\max}$, $v_{\min} \leq v \leq v_{\max}$.
- `ParametricPlot3D[{x,y,z}, {t, tmin, tmax}]` nariše parametrično podano prostorsko krivuljo $x = x(t)$, $y = y(t)$, $z = z(t)$ na intervalu $t_{\min} \leq t \leq t_{\max}$.
- `ParametricPlot3D[{x1,y1,z1}, {x2,y2,z2}, ...]` nariše več objektov hkrati.

9.3 Sestavljanje grafičnih objektov

Pri naštetih grafičnih funkcijah je slika le "stranski učinek". Vrednost, ki jo vrnejo, pa je ustrezni *grafični objekt*. Izraz v Mma je grafični objekt, če ima eno izmed naslednjih glav: `Graphics, Graphics3D, SurfaceGraphics, DensityGraphics, ContourGraphics`. Funkcije `Plot, ParametricPlot` in `ListPlot` vrnejo objekt tipa `Graphics`, `Plot3D` in `ListPlot3D` vrneta tip `SurfaceGraphics`, `ParametricPlot3D` vrne `Graphics3D`, `DensityPlot` vrne `DensityGraphics` in `ContourPlot` vrne `ContourGraphics`.

Grafične objekte pa lahko sestavljamo tudi sami in jih prikažemo s `Show`. Argument grafičnega objekta je seznam grafičnih elementov in grafičnih ukazov. Elementi se rišejo po vrsti, grafični ukazi pa veljajo za vse tiste elemente, ki jim v seznamu sledijo, oziroma dokler jih kak drug grafični ukaz ne prekliče. Ukaz

```
Show[Graphics[ {Point[{-1,0}], RGBColor[1,0,0], Point[{1,0}]} ]]
```

bo narisal točko $(-1, 0)$ črno, točko $(1, 0)$ pa rdeče.

Grafični elementi v ravnini (pod glavo `Graphics`) so:

- `Point[{x,y}]`: točka (x, y)
- `Line[{{x1,y1}, {x2,y2}, ...}]`: lomljena črta skozi točke (x_1, y_1), (x_2, y_2), ...
- `Circle[{x,y}, r]`: krožnica s središčem v (x, y) in s polmerom r
- `Circle[{x,y}, r, {fi1,fi2}]`: krožni lok
- `Circle[{x,y}, {a,b}]`: elipsa s središčem v (x, y) in s polosema a, b
- `Circle[{x,y}, {a,b}, {fi1,fi2}]`: eliptični lok
- `Disk[{x,y}, r]`: (zapolnjen) krog s središčem v (x, y) in s polmerom r, itd.
- `Rectangle[{xmin,ymin}, {xmax,ymax}]`: zapolnjen pravokotnik
- `Polygon[[{{x1,y1}, {x2,y2}, ...}]`: zapolnjen mnogokotnik z oglišči (x_1, y_1), (x_2, y_2), ...

Vse te elemente lahko uporabljamo tudi v prostoru (pod glavo `Graphics3D`), le da imajo tam točke pač po tri koordinate. Razen tega imamo na voljo še grafični element `Cuboid[{xmin,ymin,zmin},{xmax,ymax,zmax}]`, ki predstavlja kvader.

10 Algebra

10.1 Preoblikovanje izrazov

V nadaljevanju "ulomek" pomeni racionalno funkcijo.

- `Together[x]`: vsoto ulomkov x postavi na skupni imenovalec
- `Apart[x]`: zapiše ulomek x kot vsoto delnih ulomkov
- `Factor[x]`: števec in imenovalec izraza x razcepi nad obsegom racionalnih števil na nerazcepne faktorje
- `Expand[x]`: v izrazu x distribuira množenje čez seštevanje
- `ExpandNumerator[x]`: v števcih členov izraza x distribuira množenje čez seštevanje
- `ExpandDenominator[x]`: v imenovalcih členov izraza x distribuira množenje čez seštevanje
- `ExpandAll[x]`: v števcih in imenovalcih členov izraza x distribuira množenje čez seštevanje
- `Cancel[x]`: okrajša ulomek x
- `Simplify[x]`: izraz x preoblikuje na več načinov in med dobljenimi izrazi izbere najkrajšega (tj. tistega, ki minimizira vrednost funkcije `LeafCount`)
- `PowerExpand[x]`: podizraze oblike $(ab)^x$ nadomesti z $a^x b^x$
- `Collect[x, t]`: zapiši izraz x kot polinom spremenljivke t

10.2 Polinomi

- `PolynomialQ[p, x]`: ali je p polinom spremenljivke x?
- `Exponent[p, x]`: največji eksponent spremenljivke x v polinomu p
- `Coefficient[p, x, n]`: koeficient polinoma p pri potenci x^n
- `CoefficientList[p, x]`: seznam koeficientov polinoma p spremenljivke x, začenši s prostim členom
- `PolynomialQuotient[p, q, x]`: količnik pri deljenju polinoma $p(x)$ s polinomom $q(x)$
- `PolynomialRemainder[p, q, x]`: ostanek pri deljenju polinoma $p(x)$ s polinomom $q(x)$
- `PolynomialDivision[p, q, x]`: seznam $\{k, r\}$, kjer je k količnik, r pa ostanek pri deljenju polinoma $p(x)$ s polinomom $q(x)$
- `PolynomialGCD[p1, p2, ...]`: največji skupni delitelj polinomov $p_1, p_2, \ldots$.
- `PolynomialLCM[p1, p2, ...]`: najmanjši skupni večkratnik polinomov $p_1, p_2, \ldots$.
- `FactorList[p]`: seznam parov $\{q_i, n_i\}$, kjer je q_i faktor z eksponentom n_i v razcepu polinoma p na nerazcepne faktorje nad obsegom racionalnih števil
- `Resultant[p, q, x]`: rezultanta polinomov $p(x)$ in $q(x)$

- `Decompose[p, x]`: seznam polinomov $\{p_1, p_2, \ldots, p_n\}$ spremenljivke x, tako da je $p(x) = p_1(p_2(\ldots(p_n)\ldots))$
- `GroebnerBasis[{p1, p2, ...}, {x1, x2, ...}]`: poišče reducirano Gröbnerjevo bazo za ideal, ki ga generirajo polinomi $p_1(x_1, x_2, \ldots)$, $p_2(x_1, x_2, \ldots)$, glede na leksikografsko urejenost $x_1 > x_2 > \ldots$.

10.3 Linearna algebra

Vektorje predstavimo kot sezname, matrike kot sezname vrstic. Tenzor ranga r predstavimo kot vgnezden seznam globine r. Če sta u in v vektorja dolžine n, a pa število, je

- `a u` produkt vektorja u s skalarjem a,
- `u + v` vsota vektorjev u in v,
- `u v` vektor s komponentami $(u_1v_1, u_2v_2, \ldots, u_nv_n)$,
- `u.v` skalarni produkt vektorjev u in v.

Če ima matrika a toliko stolpcev kot matrika b vrstic, je `a.b` produkt matrik a in b.

- `Array[f, {n1, n2, .., nk}]`: tenzor ranga k z elementi $f(i_1, i_2, \ldots, i_k)$
- `DiagonalMatrix[d]`: diagonalna matrika s seznamom diagonalnih elementov d
- `IdentityMatrix[n]`: n-vrstna enotska matrika
- `Transpose[a]`: transponiranka matrike a
- `a[[i,j]]`: i,j-ti element matrike a
- `a[[i]]`: i-ta vrstica matrike a
- `Transpose[a][[j]]`: j-ti stolpec matrike a
- `a[[{i1,..,ik}, {j1,..,jm}]]`: podmatrika matrike a, sestavljena iz elementov na križiščih vrstic $i_1, \ldots, i_k$ in $j_1, \ldots, j_m$
- `VectorQ[x]`: ali je x vektor?
- `MatrixQ[x]`: ali je x matrika?
- `Dimensions[x]`: seznam števila vrstic, stolpcev, ... tenzorja x
- `Det[a]`: determinanta kvadratne matrike a
- `Inverse[a]`: inverzna matrika kvadratne matrike a
- `Minors[a, k]`: matrika determinant vseh kvadratnih k-vrstnih podmatrik matrike a
- `MatrixPower[a, n]`: n-ta potenca matrike a
- `MatrixExp[a]`: matrična eksponentna funkcija matrike a
- `NullSpace[a]`: baza jedra matrike a

- `LinearSolve[a, b]`: rešitev x sistema linearnih enačb $ax = b$
- `Length[a[[1]]] - Length[NullSpace[a]]`: rang matrike a
- `RowReduce[a]`: matrika a, reducirana z elementarnimi vrstičnimi transformacijami
- `Eigenvalues[a]`: seznam lastnih vrednosti matrike a
- `Eigenvectors[a]`: seznam linearno neodvisnih lastnih vektorjev matrike a
- `Eigensystem[a]`: seznam oblike { *lastne vrednosti, lastni vektorji* } matrike a; istoležne lastne vrednosti in lastni vektorji si ustrezajo
- `SingularValues[a]`: singularni razcep številske matrike a
- `PseudoInverse[a]`: psevdoinverzna matrika številske matrike a
- `QRDecomposition[a]`: QR razcep številske matrike a
- `SchurDecomposition[a]`: Schurov razcep številske matrike a

11 Analiza

11.1 Konstante

- `Pi`: π
- `E`: $e = \lim_{n\to\infty}(1 + 1/n)^n$
- `EulerGamma`: $\gamma = \lim_{n\to\infty}(\sum_{k=1}^{n} 1/k - \ln n)$
- `Catalan`: $\sum_{n=0}^{\infty}(-1)^n/(2n+1)^2$
- `GoldenRatio`: $\varphi = (1 + \sqrt{5})/2$
- `Degree`: $\pi/180$
- `I`: kompleksno število i
- `Infinity`: ∞
- `-Infinity`: $-\infty$

11.2 Elementarne funkcije

11.2.1 Algebraične:

- `x+y`, `x y`, `x-y`, `x/y`, `x^y`: vsota, produkt, razlika, kvocient, potenca
- `Sqrt[x]`: $\sqrt{x}$

11.2.2 Trigonometrične:

- `Sin[x]`, `Cos[x]`, `Tan[x]`, `Cot[x]`: $\sin x$, $\cos x$, $\mathrm{tg}\, x$, $\mathrm{ctg}\, x$
- `Sec[x]`, `Csc[x]`: $1/\cos x$, $1/\sin x$
- `ArcSin[x]`, `ArcCos[x]`, `ArcTan[x]`, `ArcCot[x]`: $\arcsin x$, $\arccos x$, $\mathrm{arctg}\, x$, $\mathrm{arcctg}\, x$

11.2.3 Eksponentne, logaritemske in hiperbolične:

- `Exp[x]` ali `E^x`: e^x
- `Log[x]`: $\ln x$
- `Log[b,x]`: $\log_b x$
- `Sinh[x]`, `Cosh[x]`, `Tanh[x]`, `Coth[x]`: $\mathrm{sh}\, x$, $\mathrm{ch}\, x$, $\mathrm{th}\, x$, $\mathrm{cth}\, x$
- `Sech[x]`, `Csch[x]`: $1/\mathrm{ch}\, x$, $1/\mathrm{sh}\, x$
- `ArcSinh[x]`, `ArcCosh[x]`, `ArcTanh[x]`, `ArcCoth[x]`: $\mathrm{arsh}\, x$, $\mathrm{arch}\, x$, $\mathrm{arth}\, x$, $\mathrm{arcth}\, x$

11.3 Infinitezimalni račun

- `Limit[f, x->a]`: limita izraza f, ko gre x proti a
- `Limit[f, x->a, Direction->1]`: leva limita izraza f, ko gre x proti a
- `Limit[f, x->a, Direction->-1]`: desna limita izraza f, ko gre x proti a
- `D[f, x]`: odvod izraza f po x
- `D[f, {x,n}]`: n-ti odvod izraza f po x
- `D[f, x1, x2, .., xn]`: parcialni odvod izraza f po $x_n, \ldots, x_2, x_1$
- `D[f, x, NonConstants->s]`: odvod izraza f po x, pri čemer so spremenljivke iz seznama s implicitno odvisne od x
- `Dt[f]`: totalni diferencial izraza f
- `Dt[f, x]`: odvod izraza f po x, pri čemer so vse nastopajoče spremenljivke implicitno odvisne od x
- `Dt[f, x1, x2, .., xn]`: parcialni odvod izraza f po $x_n, \ldots, x_2, x_1$, pri čemer so vse nastopajoče spremenljivke implicitno odvisne od $x_1, x_2, \ldots, x_n$
- `Dt[f, x, Constants->s]`: odvod izraza f po x, pri čemer so spremenljivke iz seznama s neodvisne od x
- `SetAttributes[c, Constant]`: c je konstanta v vseh okoliščinah
- `f'[x]`, `f''[x]`, ...: $f'(x), f''(x), \ldots$

12 Vzorci

VZOREC	PREDSTAVLJA
`a` (atom)	`a`
`_`	poljuben izraz
`x_`	poljuben izraz (z imenom `x`)
`__`	poljubno neprazno zaporedje izrazov
`x__`	poljubno neprazno zaporedje izrazov (z imenom `x`)
`___`	poljubno zaporedje izrazov
`x___`	poljubno zaporedje izrazov (z imenom `x`)
`x_h`	izraz z glavo `h`
vz`?f`	izraz, ki ustreza vzorcu *vz* in pri katerem je predikat `f` izpolnjen
vz `/; pog`	izraz, ki ustreza vzorcu *vz* in ki zadošča pogoju `pog`
`x:`*vz*	izraz, ki ustreza vzorcu *vz* (z imenom `x`)
$vz_1 \vert vz_2 \vert \ldots \vert vz_n$	izraz, ki ustreza vsaj enemu od vzorcev $vz_1, vz_2, \ldots, vz_n$
vz..	neprazno zaporedje izrazov, od katerih vsak ustreza vzorcu *vz*
vz...	zaporedje izrazov, od katerih vsak ustreza vzorcu *vz*
`x_:`*v*	privzeta vrednost za manjkajoči argument `x`
`x_.`	vgrajena privzeta vrednost za manjkajoči argument `x`
$vz_0[vz_1 \ldots, vz_n]$	izraz oblike `f[x1, ..., xn]`, kjer `f` ustreza vzorcu vz_0, `xi` pa vzorcu vz_i

13 Pogoste napake

Zelo čudne stvari se lahko dogodijo, kadar:

1. pozabimo znak `&` na koncu definicije čiste funkcije,
2. v enačbi uporabimo znak `=` namesto `==`,
3. v vzorcu oblike `x_?f`, kjer je `f` čista funkcija, le-te ne obdamo z okroglimi oklepaji (`_` veže namreč močneje kot `&`),
4. argumentu funkcije v telesu funkcije spreminjamo vrednost (če želimo to početi, uvedemo lokalnega "dvojnika", npr. namesto `f[x_] := Module[{}, x = x + 1]` napišemo `f[x_] := Module[{xx = x}, xx = xx + 1]`),
5. simbol z vrednostjo uporabimo kot vezano spremenljivko ali kot nedoločenko (argument čiste funkcije v polnem zapisu, integracijska spremenljivka, neznanka v sistemu enačb),
6. ne postavimo presledka med faktorja, prvi od katerih je simbol.