# Základní práce s čísly a výrazy

## Konstanty

Konstanty se píší velkým písmenem, např.

- Pi značí Ludolfovo číslo 3.14...
- I značí imaginární jednotku i
- E značí Eulerovo číslo 2.71...
- Infinity značí mekonečno

## Čísla

Napíše-li se např. zlomek jako podíl celých čísel, dá program výsledek podílu jen tehdy, je-li opět celé číslo. Chceme-li dostat výsledek jako desetinné číslo, je nutné programu sdělit, že buď pracuje s reálnými čísly (to se provede přidáním tečky za jedno celé číslo, např. 3./4 nebo 3/4.) nebo, že chcete výsledek numericky (což se vyznačí pomocí N, např. 3/4//N nebo N[3/4]). Program vypíše numericky spočítaný výsledek obvykle na 5 desetinných míst. Jiný počet míst se určí následovně, např. Ludolfovo číslo na 20 desetinných míst, N[Pi,20]. Pokud se napíše pouze Pi, program odpoví také Pi, protože přesnou hodnotu nemůže uvést.

## Aritmetické operace

Aritmetické operace se značí standardním způsobem:
+, -, *, ^, ! (sčítání, odčítání, násobení, dělení, umocňování, faktoriál).

Odmocňování nezáporných čísel se vypočte pomocí mocniny, např. čtvrtá odmocnina ze 7 se zjistí pomocí příkazu 7^(1/4) -- bez reálného zápisu (např. 7.^(1/4)) nebo numerického výpočtu (např. 7^(1/4)//N) program jen opíše příkaz, protože byl zadán v oboru přirozených čísel a v něm přesná odpověď neexistuje.

Odmocňování záporných čísel výše uvedeným způsobem dá komplexní číslo i pro liché odmocniny:

```
(-8)^(1/3) // N
```

```
1. + 1.73205 i
```

Je více možností, jak dosáhnout reálného výsledku, všechny jsou ale komplikovanější. Je možné nahrát starý balíček RealOnly příkazem <<Miscellaneous`RealOnly`. Ale nemůžeme pak počítat s komplexními čísly a navíc se nahrané balíčky hůře odstraňují (např. resetováním jádra - pak se ale zruší i definice).
Další možností je si lichou odmocninu záporných čísel nadefinovat, např. následujícím způsobem.

```
odm1[x_, n_] := If[Element[(n + 1) / 2, Integers],
  Piecewise[{{x^(1 / n), x ≥ 0}, {- (-x)^(1 / n), x < 0}}],
  If[x ≥ 0, x^(1 / n), "Nejde"]]
```

```
odm1[-8, 3]
```

$-2$

```
odm1[-8, 2]
```

Nejde

```
odm1[-8, 5] // N
```

$-1.51572$

```
odm1[-8, 3 / 2]
```

Nejde

```
odm1[-8 + I, 3]
```

— *GreaterEqual::nord : Invalid comparison with $-8+i$ attempted.* ≫

— *Less::nord : Invalid comparison with $-8+i$ attempted.* ≫

$$\begin{cases} \sqrt[3]{-8+i} & -8+i \geq 0 \\ -\sqrt[3]{8-i} & -8+i < 0 \end{cases}$$

Jednoduše lze liché odmocniny počítat pomocí kořenů polynomu. Číslo 1 na konci příkazu znamená, že se má brát první kořen - to u liché odmocniny dá správný výsledek, u sudé odmocniny kladného čísla dostaneme záporný kořen.. To lze snadno ošetřit (zkuste to i s podmínkou, že n je celé číslo).

```
odm2[x_, n_] := Root[y^n - x, 1]
```

```
odm2[-8, 3]
```

$-2$

```
odm2[8, 10] // N
```

$-1.23114$

```
odm2[-1, 4] // N
```

$-0.707107 - 0.707107\, i$

Další možností je použít příkaz Solve[y^n==x,Reals], kde lze za n dosazovat i necelé hodnoty. Zkuste pomocí tohoto příkazu napsat vhodnou definici, která zahrnuje všechny mocniny reálných čísel (tj. i umocňování záporných čísel na některá racionální čísla). Následující obdoba definice odm1 takovou vlastnost má.

```
mocnina[x_, a_] := If[Element[x, Reals] && x >= 0, x^a,
  If[Element[x, Reals] && x < 0 && Element[a, Rationals] &&
    (EvenQ[Numerator[a]] ||
      OddQ[Numerator[a] * Denominator[a]]) ,
   (2 * Boole[EvenQ[Numerator[a]]] - 1) * (-x) ^a, "Nejde"]]
```

```
mocnina[-8, 8 / 6] // N
```

16.

```
mocnina[-8, 10 / 6] // N
```

−32.

```
mocnina[8, 10 / 6] // N
```

32.

```
mocnina[-8, Pi] // N
```

Nejde

```
mocnina[-8, 1 / 2] // N
```

Nejde

## *Užitečné příkazy*

Následující přehled uvádí některé příkazy, které usnadní různé operace. Snadno je vidět, co příkazy s matematickým výrazem provedou.

```
Expand[(x + 2) (x - 1)]
```

$x^2 + x - 2$

```
Factor[%]
```

$(x - 1)(x + 2)$

```
Together[2 / y + 1 / (y - 1)]
```

$$\frac{3\,y - 2}{(y - 1)\,y}$$

```
Apart[%]
```

$$\frac{2}{y} + \frac{1}{y - 1}$$

```
Simplify[a^2 - 4 a + 4]
```

$(a - 2)^2$

```
Numerator[(a + 7) / (2 b - 3)]
```

$a + 7$

```
Denominator[(a + 7) / (2 b - 3)]
```

$2\,b - 3$