Programování v prostředí MATHEMATICA

Příkaz Print

Print[] ... slouží pro výpis textu a hodnot proměnných

```
Print["TEXT"]
```

```
TEXT
```

```
x = 5
```

```
5
```

```
Print["x = ", 5]
```

```
x = 5
```

Druhy přiřazení

x = 5 ... přiřazení hodnoty
i++ ... přičtení jedničky, výsledek se projeví až při dalším volání i
i-- ... odečtení jedničky, výsledek se projeví až při dalším volání i
++i ... přičtení jedničky, výsledek se projeví ihned
--i ... odečtení jedničky, výsledek se projeví ihned
i += d ... přičtení d k hodnotě i, i = i + d
i -= d ... odečtení d od hodnoty i, i = i - d
x *= c ... vynásobení x konstantou c, x = x*c
x /= c ... vydělení x konstantou c, x = x/c

```
x = {2, 3, 4}; x *= 4
```

```
{8, 12, 16}
```

Relační operátory

Rovnost

Equal[x, y] nebo x == y ... číselné srovnání, neporovnává formát čísla
SameQ[x, y] nebo x === y ... totožnost, porovnává hodnotu i formát čísla

```
Equal[3, 3.]
```

```
True
```

```
SameQ[3, 3.]
```

```
False
```

Nerovnost

Unequal[x, y] nebo x != y ... číselné srovnání, neporovnává formát čísla
UnsameQ[x, y] nebo x =!= y ... totožnost, porovnává hodnotu i formát čísla

Další porovnání

Less[x, y] nebo x <y ... pravdivé pro x menší než y
Greater[x, y] nebo x > y ... pravdivé pro x větší než y

LessEqual[x, y] nebo x <=y nebo x≤y... pravdivé pro x menší nebo rovno y
GreaterEqual[x, y] nebo x >= y nebo x≥y ... pravdivé pro x větší nebo rovno y

---

Logické operátory

And[výraz1, výraz2, ...] nebo výraz1 && výraz2 && ... Logický součin
Or[výraz1, výraz2, ...] nebo výraz1 || výraz2 || ... Logický součet
Nand[výraz1, výraz2, ...] ... Negace logického součinu
Nor[výraz1, výraz2, ...] ... Negace logického součtu
Xor[výraz1, výraz2, ...] ... nonekvivalence (Exklusive OR)
Not[výraz] nebo !výraz ... negace

```
7 > 4 && 2 ≠ 3
```

```
True
```

LogicalExpand[výraz] ... zjednodušuje (rozkládá) složené logické výrazy

```
LogicalExpand[p == q && r == s || p == q || r == s]
```

```
q == p || s == r
```

Funkce pro testování

IntegerQ[] ... rozhodne, zda se jedná o celé číslo
NumberQ[] ... rozhodne, zda se jedná o číslo
EvenQ[]/OddQ[] ... rozhodne, zda se jedná o sudé/liché číslo
PrimeQ[] ... rozhodne, zda se jedná o prvočíslo
NumericQ[] ... rozhodne, zda lze výraz vyjádřit numericky
Positive[], Negative[] ... rozhodne, zda se jedná o kladné / záporné číslo
PolynomialQ[] ... rozhodne, zda se jedná o polynom
MatrixQ[] ... rozhodne, zda se jedná o matici
VectorQ[] ... rozhodne, zda se jedná o vektor
ValueQ[] ... rozhodne, zda je pro proměnnou definována nějaká hodnota
MemberQ[vektor, prvek] ... rozhodne, zda vektor obsahuje daný prvek
FreeQ[vektor, prvek] ... Opak funkce MemberQ[]

```
EvenQ[12]
```

```
True
```

```
OddQ[12]
```

```
False
```

```
MemberQ[{a, b, c, d}, a]
```

```
True
```

Iterátory

{imax} ... iteruje imax krát
{i, imax} ... i se mění od 1 do imax s krokem 1
{i, imin, imax} ... i se mění od imin do imax s krokem 1
{i, imin, imax, di} ... i se mění od imin do imax s krokem di
{i, imin, imax}, {j, jmin, jmax} ... i se mění od imin do imax s krokem 1, pro každou hodnotu i se mění j od jmin do jmax s krokem 1

```
Table[{i, j, i * j}, {i, 0, 3}, {j, 0, 3}]
```

```
{{{0, 0, 0}, {0, 1, 0}, {0, 2, 0}, {0, 3, 0}}, {{1, 0, 0}, {1, 1, 1}, {1, 2, 2}, {1, 3, 3}},
 {{2, 0, 0}, {2, 1, 2}, {2, 2, 4}, {2, 3, 6}}, {{3, 0, 0}, {3, 1, 3}, {3, 2, 6}, {3, 3, 9}}}
```

Definice globální a lokální proměnné

Definice lokální proměnné

```
a = 5
```

```
5
```

Definice globální proměnné

Module[{proměnné}, výchozí hodnoty, příkazy]

```
Module[{a}, a = 20; a = a!; a]
```

```
2 432 902 008 176 640 000
```

```
a
```

```
5
```

Ryzí funkce

Ryzí funkce umožňuje definovat vyhodnocení funkce, bez definice jejího názvu

Function[proměnná, funkce][hodnota]

```
Function[x, x^2][5]
```

```
25
```

```
Function[{x, y}, x^2 + y^2][2, 5]
```

```
29
```

- **Symboly # a &; # ... argument funkce, & ... ryzí funkce**

```
#^2 &[5]
```

```
25
```

```
#1^2 + #2^2 &[2, 5]
```

```
29
```

Opakované vyhodnocení funkce

- **Nest[funkce, x, n] ... vyhodnocení funkce n - krát s počáteční hodnotou x**

```
Nest[fce, x, 3]
```

```
fce[fce[fce[x]]]
```

```
divide2[n_] := n / 2
```

```
Nest[divide2, 125, 5]
```

$$\frac{125}{32}$$

- **NestList[funkce, x, n] ... vyhodnocení funkce n - krát s počáteční hodnotou x, vytvoří seznam průběžných výsledků**

```
NestList[fce, x, 3]
```

```
{x, fce[x], fce[fce[x]], fce[fce[fce[x]]]}
```

```
NestList[divide2, 125, 5]
```

$$\left\{125, \frac{125}{2}, \frac{125}{4}, \frac{125}{8}, \frac{125}{16}, \frac{125}{32}\right\}$$

- **NestWhile[funkce, x, test] ... funkce se vyhodnocuje, dokud platí zadaná podmínka, vypisuje se první hodnota, která podmínku nesplňuje**

```
NestWhile[divide2, 120, EvenQ]
```

```
15
```

- **NestWhileList[funkce, x, test] ... funkce se vyhodnocuje, dokud platí zadaná podmínka, vypisuje se vektor průběžných výsledků**

```
NestWhileList[divide2, 120, EvenQ]
```

```
{120, 60, 30, 15}
```

Podmínkové funkce

- **Select[list, text] ... vybere všechny prvky, které splňují zadanou podmínku**

Pozn. : Podmínka musí být psána ve tvaru ryzí funkce nebo testovací funkce

```
Select[{30, -150, 19, 16, 80, 14, -28, 86}, # > 20 &]
```

```
{30, 80, 86}
```

```
Select[{30, -150, 19, 16, 80, 14, -28, 86}, # > 20 &, 2]
```

```
{30, 80}
```

```
Select[{30, -150, 19, 16, 80, 14, -28, 86}, OddQ]
```

```
{19}
```

- **If[test, pravda, nepravda] ... vybere všechny prvky, které splňují zadanou podmínku**

```
If[7 > 8, Print["Pravda"], Print["Nepravda"]]
```

```
Nepravda
```

- **If[test, pravda, nepravda,nelzeurčit] ... vybere všechny prvky, které splňují zadanou podmínku**

```
If[prom1 > prom2, a, b]
```

```
If[prom1 > prom2, a, b]
```

```
If[prom1 > prom2, a, b, c]
```

```
c
```

- **Which[test1, hodnota1, test2, hodnota2, ...] ... provede vyhodnocení test1 až testN, vrací hodnotu hodnota1 až hodnotaN, která odpovídá prvnímu pravdivému testu**

```
a = 3
```

```
3
```

```
Which[a == 1, Print["Jedna"], a == 2, Print["Dvě"], a == 3, Print["Tři"]]
```

```
Tři
```

- **Switch[testovanývýraz, hodnotavýrazu1, výsledek1, hodnotavýrazu2, výsledek2, ...] ... vyhodnotí porovnání testovaného výrazu s hodnotami 1 až N a následně vrací odpovídající výsledek**

```
Switch[3, 1, a, 2, b, 3, c]
```

```
c
```

Programování cyklů

- **Do[výraz, {i, imin, imax, krok}] ... vyhodnotí výraz pro i od imin po imax s daným krokem**

```
Do[Print[i^2], {i, 5}]
```

```
1
4
9
16
25
```

- **While[test, výraz] ... Opakovaně provádí výraz, pokud je splněna podmínka test, při nesplnění podmínky tento cyklus končí**

```
a = 10; While[a > 5, a = a - 1; Print[a]]
```

```
9
8
7
6
5
```

- **For[start, test, zmena, vyraz] ... Opakovaně provádí výraz, pokud je splněna podmínka test, při nesplnění podmínky tento cyklus končí. Zároveň mění hodnotu danou počáteční podmínkou start dle výrazu zmena**

```
For[i = 1, i < 5, i++, Print[i]]
```

```
1
2
3
4
```

Kontrola a řízení průběhu cyklů

- **Break[] ... opustí nejbližší uzavřenou smyčku**

```
t = 1; Do[t *= k; Print[t]; If[t > 19, Break[]], {k, 10}]
```

```
1
2
6
24
```

Continue[] ... přistoupí k dalšímu kroku aktuální smyčky

```
t = 1; Do[t *= k; If[k < 3, Continue[]]; Print[t], {k, 10}]
```

```
6
24
120
720
5040
40 320
362 880
3 628 800
```

- **Return[hodnota] ... vrátí hodnotu a opustí všechny procedury a cykly funkce**

```
f[x_] :=
 {t = 1; Do[t *= k; Print[t]; If[t > 100, Return["Výsledek je větší než 100"]], {k, x}]}
f[10]
```

```
1
2
6
24
120
{Výsledek je větší než 100}
```

- **Goto[navesti] ... přesune činnost programu na návěstí dané funkcí Label[navesti]**

```
q = 2; Label[skok]; Print[q]; q += 2; If[q < 11, Goto[skok]]; Print["KONEC"]
```

```
2
4
6
8
10
KONEC
```