

Úvod do programu MAPLE

MAPLE V Release 9 (11)

M. Jahoda
Ústav chemického inženýrství
VŠCHT Praha

<http://www.vscht.cz/uchi/maple>

<http://www.maplesoft.com>



The image displays two overlapping Maple software windows. The top window, titled "Maple 9 - [Untitled (1) - [Server 1]]", shows a menu with options like Text, Standard Math, Maple Input, and Standard Math Input. A plot of the equation $\alpha := 5$ is visible, with axes ranging from -1 to 1. The bottom window, titled "Maple 8 - [Untitled (1) - [Server 1]]", shows a similar interface with a plot of the equation $\alpha := 5$ and a status bar at the bottom indicating Time: 0.7s, Bytes: 3.06M, and Available: 2.62G.

Maple 9 - [Untitled (1) - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

Text Ctrl+T
Standard Math Ctrl+R
Maple Input Ctrl+M
Standard Math Input Ctrl+G

Execution Group ▶
Plot ▶
Spreadsheet
Paragraph ▶
Section
Subsection
HyperLink...
Object...
Page Break Ctrl+Enter

$\alpha := 5$

Live

1
0.5
0
-0.5
-1

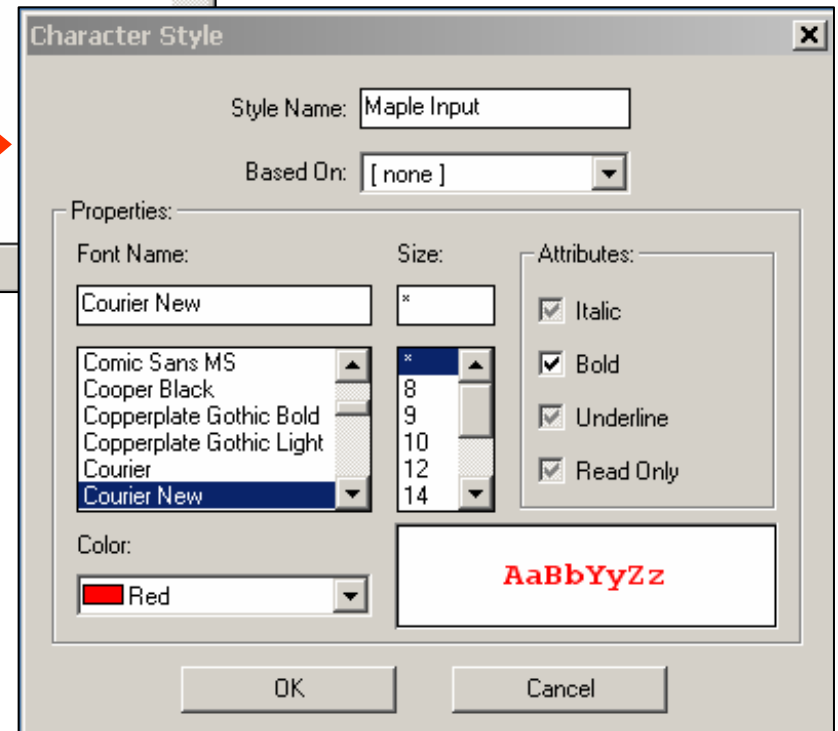
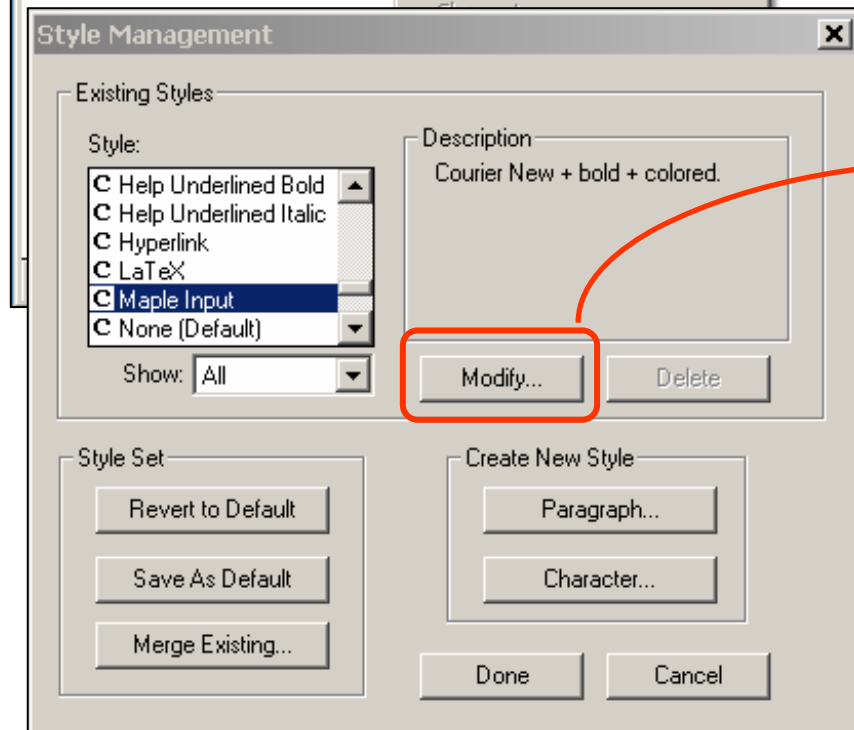
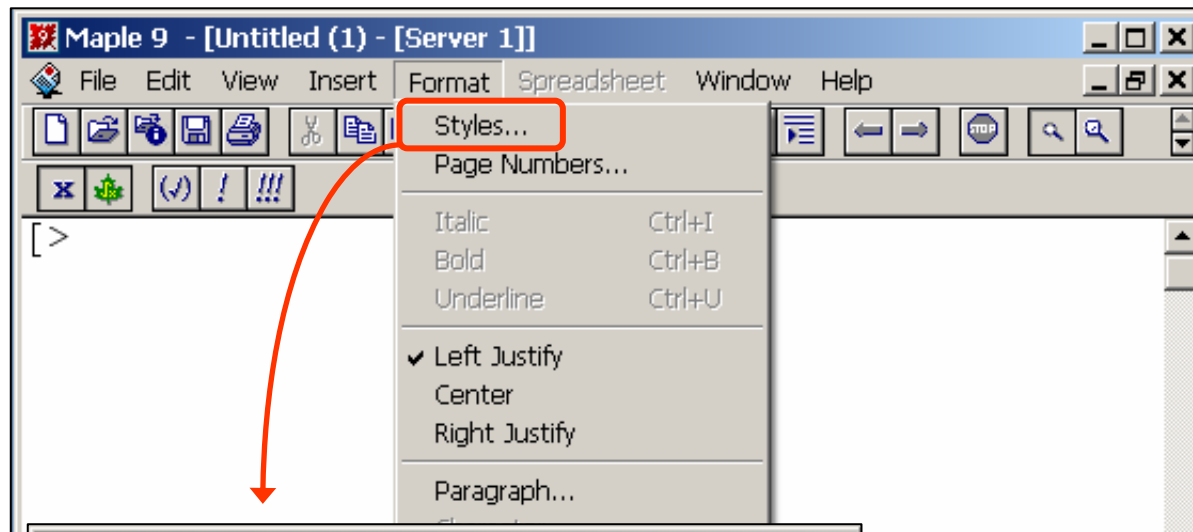
0.5 1

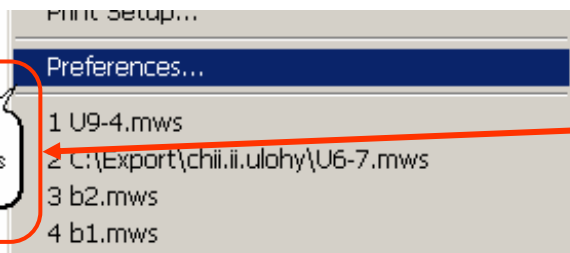
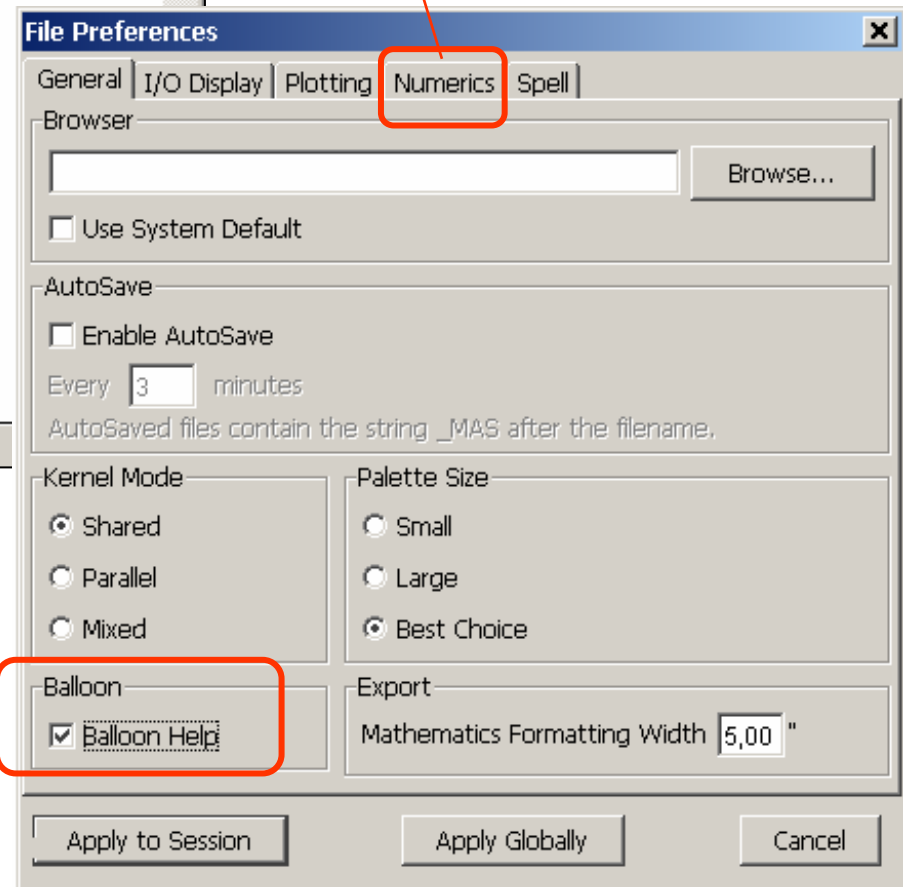
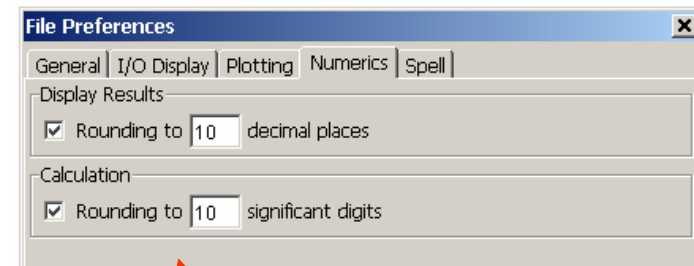
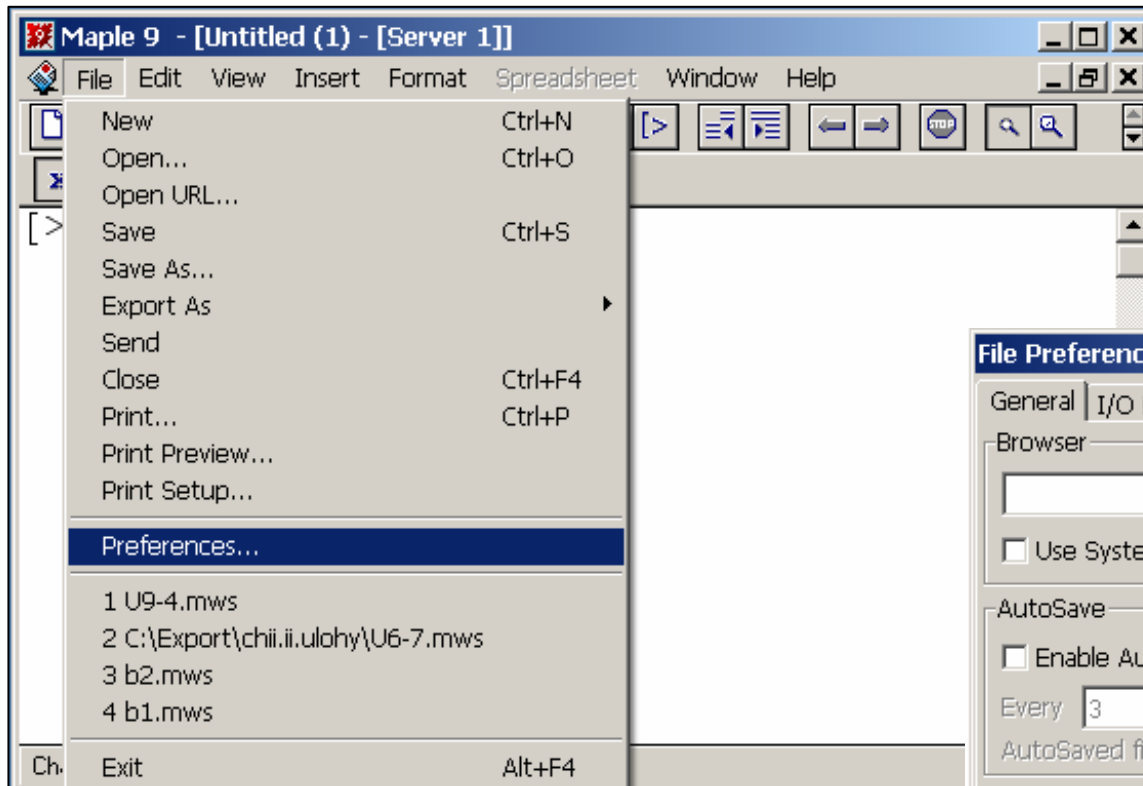
Maple 8 - [Untitled (1) - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

$\alpha := 5$

Time: 0.7s Bytes: 3.06M Available: 2.62G





Change
Maple options



Maple 9 - [Untitled (1) - [Server 1]]

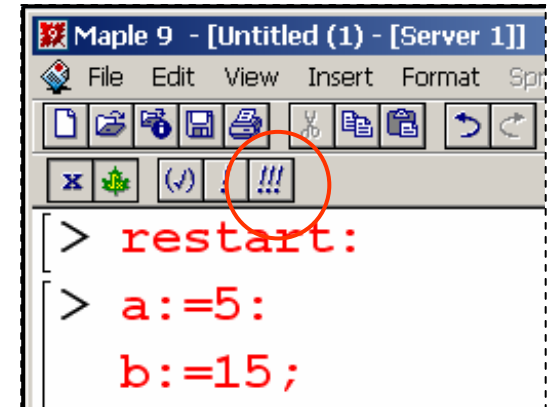
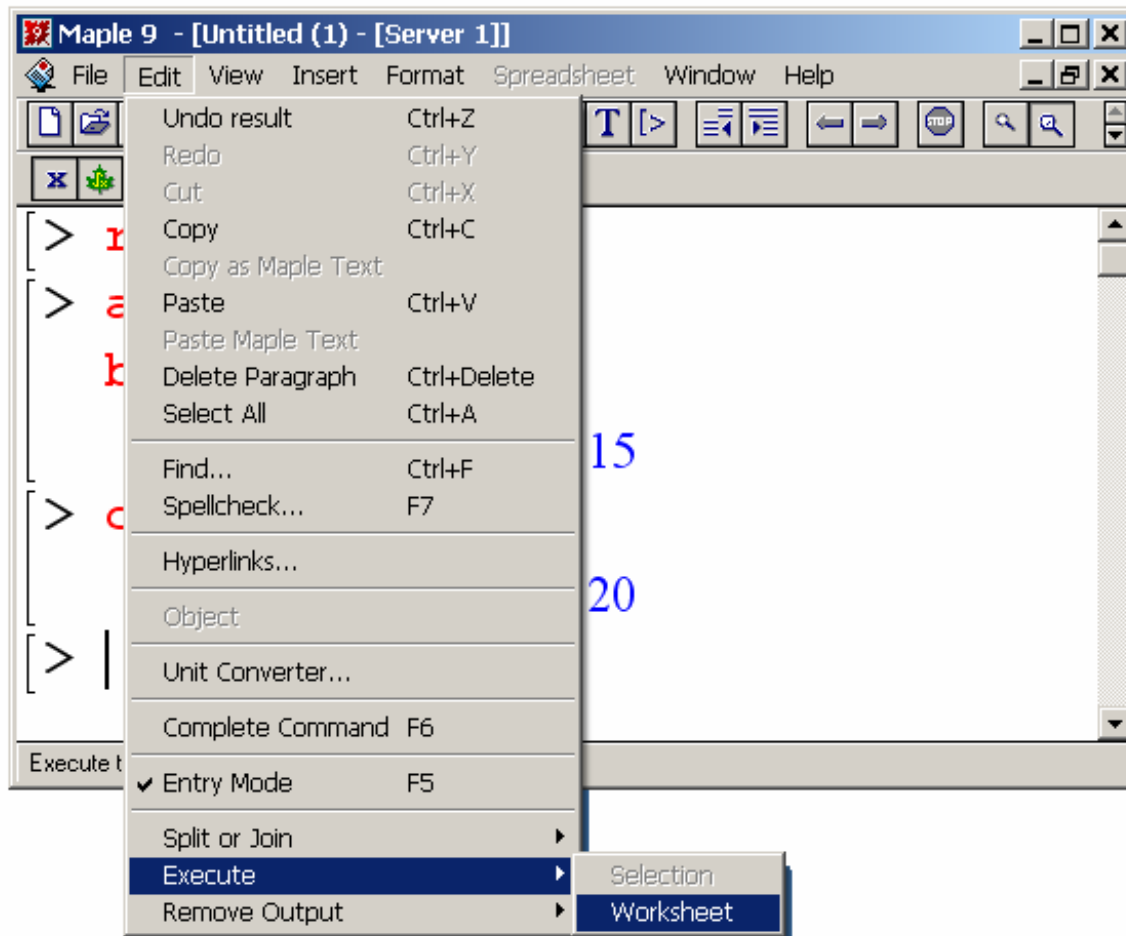
File Edit View Insert Format Spreadsheet Window Help

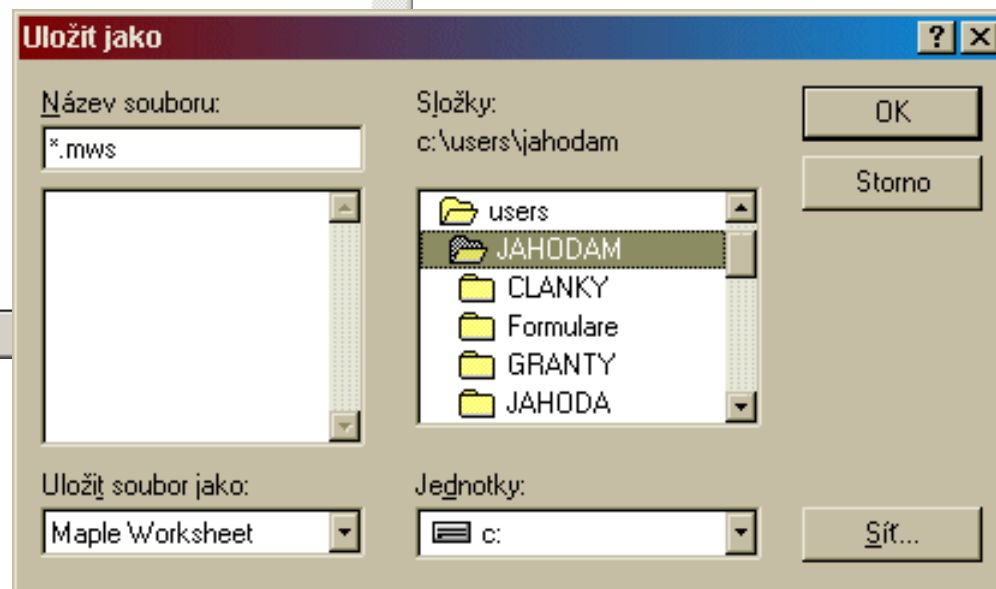
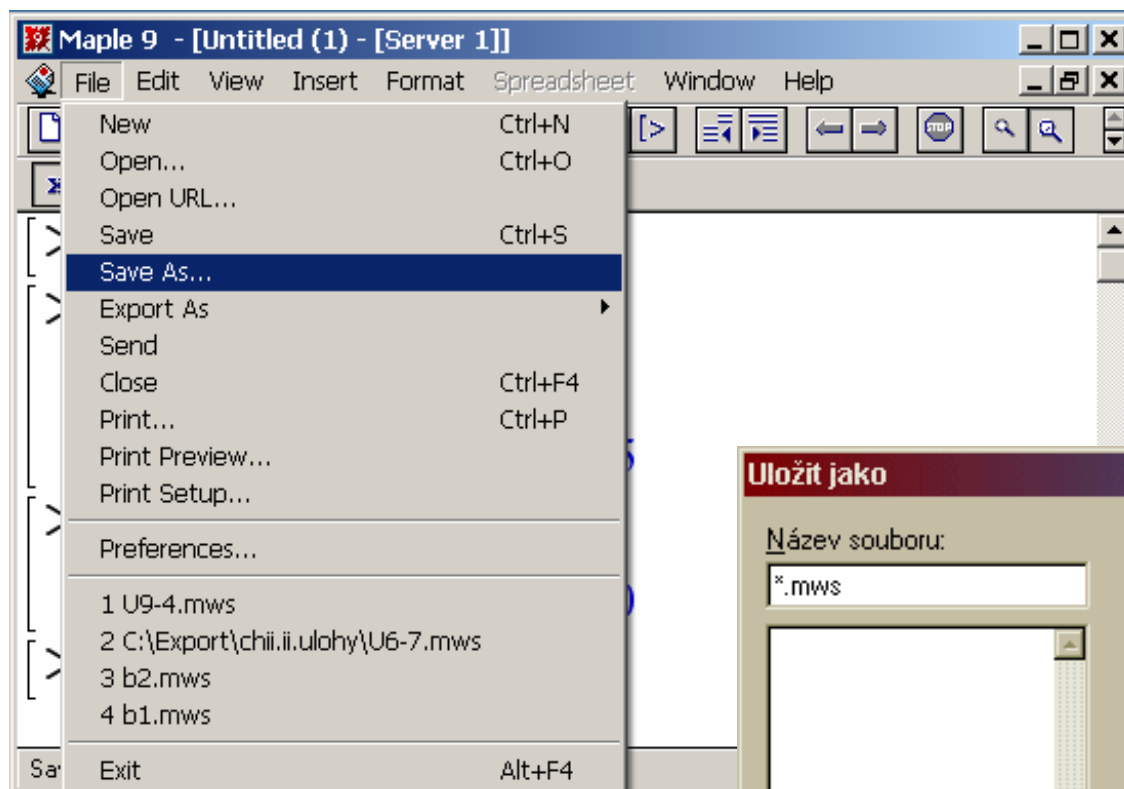
restart:
a:=5:
b:=15;
c:=a+b;
>

Skupiny na pracovní ploše

$b := 15$
 $c := 20$

Time: 0.6s Bytes: 1.94M Available: 1.90G





Uživatelské soubory PATŘÍ do uživatelských adresářů!



Výpočty se vkládají na příkazový řádek, který končí buď : nebo ;

➤ : výsledek se neznázorní

➤ ; výsledek se znázorní na konci skupiny do výsledkové oblasti

```
> 2 + 3:
```

```
> 2 + 3;
```

5

Výraz může být rozložen na několik řádků, bude vykonán až po nalezení ukončovacího znaku

```
> 2 +
```

```
> 3;
```

5

Jména proměnných **NELZE** rozdělovat!



> Proměnná := výraz;

Jména proměnných:

- rozlišení VELKÝCH a malých písmen
- jméno nesmí obsahovat: mezeru a speciální znaky $\#^*.;></$
- některá slova jsou vyhrazena: **Re**, **Pi**, **D**, ...

Při vkládání příkazů nezáleží na pořadí

```
> a:=5:  
> b:=3:  
> c:= a + b;
```

c:=8

```
> c:= a + b;  
> b:=3:  
> a:=5;
```

c:=8

MAPLE užívá symbolovou matematiku, tzn. pamatuje si zadání rovnic



Co dodržovat:

- jednotnou formu tvorby jmen proměnných
- jasný a přehledný zápis příkazů
 - užití textového popisu
 - členění výpočtu do logických oblastí
- na počátku zadat známé hodnoty
- výpočet koncipovat co nejvíce obecně

Doporučeno:

- na první příkazovém řádku příkaz

> restart:
- před výpočtem uložit pracovní oblast
- přepočítat celou oblast od začátku



$x+y$	součet
$x-y$	rozdíl
$x*y$	součin
x/y	podíl
x^y nebo $x**y$	mocnina
$\text{abs}(x)$	absolutní hodnota
$x!$	faktoriál
$\text{sqrt}(x)$ nebo $x^{(1/2)}$	druhá odmocnina
$\text{exp}(x)$	exponenciální funkce
$\ln(x)$ nebo $\log(x)$	přirozený logaritmus
$\log_{10}(x)$	dekadický logaritmus
$\sin(x)$	funkce sinus
$\cos(x)$	funkce cosinus
$\tan(x)$	funkce tangens
Pi	Ludolfova konstanta (3,141592654)



Znázornění nečíselných výsledků

evalf()

```
> Pi;
```

$$\pi$$

```
> evalf(Pi);
```

$$3.141592654$$

```
> a:=4/3;
```

$$a := \frac{4}{3}$$

```
> evalf(a);
```

$$1.3333333$$

```
> Plocha:=(Pi*d^2)/4;
```

$$Plocha := \frac{1}{4} \pi d^2$$

```
> d:=0.2:
```

```
> Plocha;
```

$$.010000000 \pi$$

```
> evalf(Plocha);
```

$$.03141592654$$



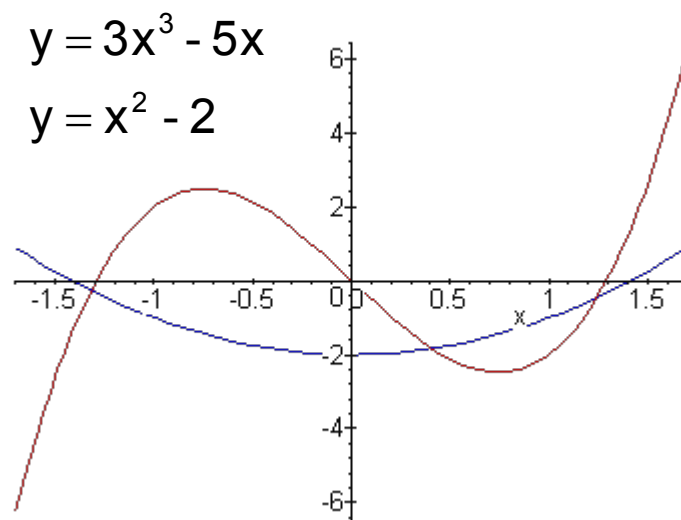
`fsolve(<rovnice>, <hledané proměnné>, <parametry>)`

```
> f1 := y = 3*x^3 - 5*x;  
> f2 := y = x^2 - 2;  
> fsolve({f1,f2},{x,y});  
      {x = 1.242775801, y = -.4555083087}
```

```
> fsolve({f1,f2},{x,y}, x=0..1);  
      {x = .4073721348, y = -1.834047944}
```

```
> fsolve({f1,f2},{x,y}, x=-1.5..0);  
      {x = -1.316814602, y = -.2659993030}
```

```
> fsolve({f1,f2},{x,y}, {x=0..1.5,y=-.5..0});  
      {x = 1.242775801, y = -.4555083087}
```





```
> f1 := y = 3*x^3 - 5*x:  
> f2 := y = x^2 - 2:  
> fsolve({f1,f2},{x,y});  
      {x = 1.242775801, y = -.4555083087}
```

MAPLE hodnoty x a y znázorní, ale pro další výpočty je nezná:

```
> x;  
      x
```

Přiřazení vypočtených hodnot `assign()`

```
> f1 := y = 3*x^3 - 5*x:  
> f2 := y = x^2 - 2:  
> s:=fsolve({f1,f2},{x,y});  
      s:= {x = 1.242775801, y = -.4555083087}
```

```
> assign(s):  
> x;  
      1.242775801
```

Po užití příkazu **assign** nelze opět použít **fsolve** dokud hodnotu proměnné neodeberme `unassign()`

```
> unassign('x','y');
```

Jména proměnných musí být v apostroftech.



`dsolve(<rovnice, poč.podmínky>, <jména proměnných>, <parametry>)`

$$\frac{dy}{dx} + y \cos(x) = \sin(2x)$$

```
> dsolve(diff(y(x),x) + y(x)*cos(x) = sin(2*x), y(x));
```

$$y(x) = 2 \sin(x) - 2 + e^{(-\sin(x))} _C1$$

Pokud je počáteční hodnota y rovna 0: $y(0)=0$

```
> d1:=diff(y(x),x) + y(x)*cos(x) = sin(2*x):
```

```
> s1:=dsolve({d1, y(0)=0}, y(x));
```

$$s1 := y(x) = 2 \sin(x) - 2 + e^{(-\sin(x))}$$



Počáteční hodnota y je 0: $y(0)=0$
a zároveň nás zajímá výsledná hodnota funkce pro $x = 1$

```
> d1:=diff(y(x),x) + y(x)*cos(x) = sin(2*x):  
> s1:=dsolve({d1, y(0)=0}, y(x), type=numeric);  
s1 := proc(rkf45_x) ... end
```

```
> s1(1);  
[x = 1, y(x) = .5450938736403281
```

nebo

```
> d1:=diff(y(x),x) + y(x)*cos(x) = sin(2*x):  
> s1:=dsolve({d1, y(0)=0}, y(x));  
s1 := y(x) = 2 sin(x) - 2 + e(-sin(x))
```

```
> assign(s1):  
> x:=1:  
> evalf(s1);  
y(1) = 0.540938712
```



`int(<funkce>,<jméno proměnné>)`

Obecné řešení

$$\int (x^3 - 3x^2 + 1) dx$$

```
> int(x^3-3*x^2+1, x);
```

$$\frac{1}{4}x^4 - x^3 + x$$

nebo

```
> fx := x^3-3*x^2+1:
```

```
> int(fx, x);
```

$$\frac{1}{4}x^4 - x^3 + x$$

Určitý integrál

$$\int_0^4 (x^3 - 3x^2 + 1) dx$$

```
> fx := x^3-3*x^2+1:
```

```
> int(fx, x=0..4);
```

4

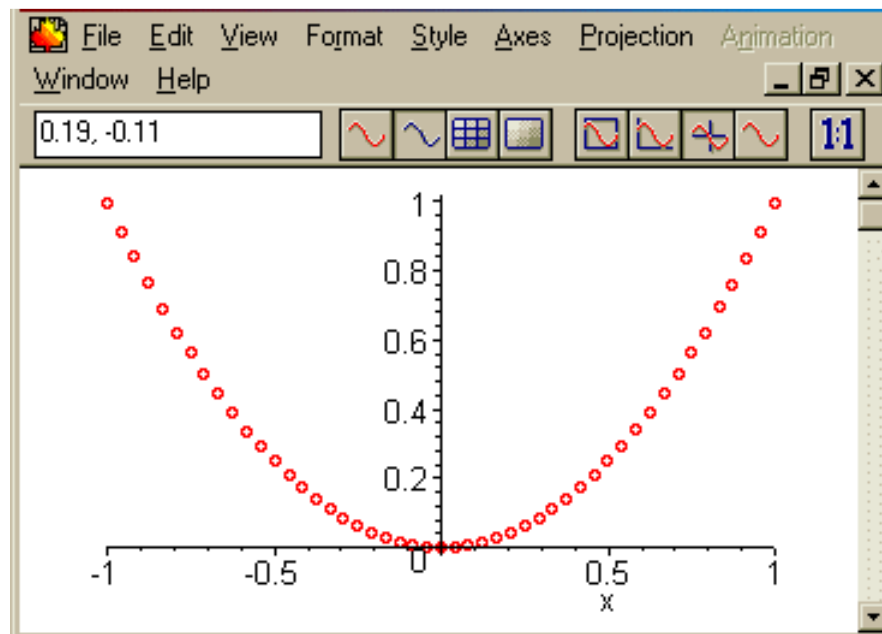


```
plot(<funkce>, <osa x>, <osa y>, <parameter>)
```

Příklad: `> plot(x^2, x = -1..1, color = red, style = point, symbol = circle);`

Nepovinné parametry

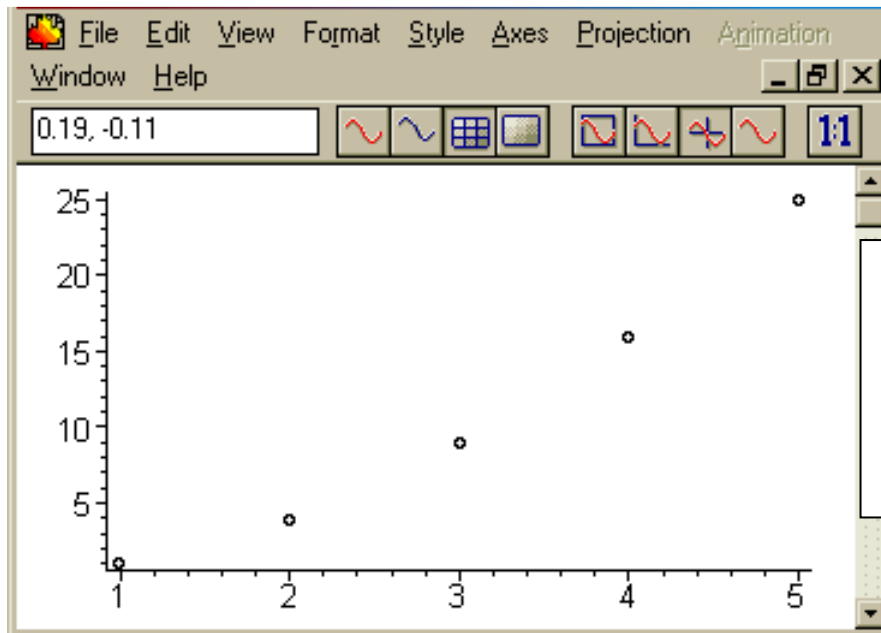
style	point		line	
symbol	box circle diamond		cross point	
thickness	0	1	2	3
color	black red green		blue cyan yellow	
linestyle	0, 1, 6 a více 2, 3, 6		spojitá čárkovaná	





```
> restart:  
> with(plots):
```

```
> dataXY:= [[1,1], [2,4], [3,9], [4,16], [5,25]]:  
> pointplot (dataXY, symbol = circle);
```

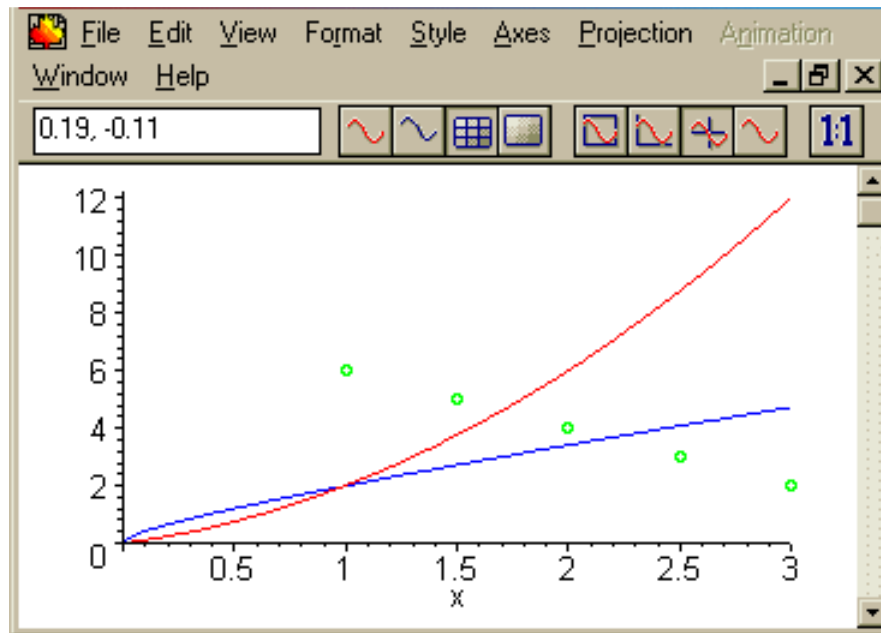


nebo

```
> X := [1, 2, 3, 4, 5]:  
> Y := [1, 4, 9, 16, 25]:  
> dataXY := array([X, Y]):  
> pointplot(dataXY, symbol = circle);
```



```
> with(plots)      # volani graficke knihovny  
> fce1 := plot(x + x^2, x = 0..3, color = red):  
> fce2 := plot(x + x^0.5, x = 0..3, color = blue):  
> XYdata := [[1, 6], [1.5, 5], [2, 4], [2.5, 3], [3, 2]]:  
> dataXY := plot(XYdata, style = point, symbol = circle, color = green):  
> display({fce1, fce2, dataXY});
```





```
if <podmínka> then <příkaz>  
| elif <podmínka> then <příkaz> |  
| else <příkaz> |  
fi
```

← tyto výrazy jsou nepovinné |

- příkazy za then se vykonají, pokud je podmínka pravdivá
- konstrukcí elif můžete mít v rozhodovacím členu kolik chcete

podmínka může obsahovat

relační operátory

= rovno
> větší než
< menší než
<> nerovno
>= větší nebo rovno
<= menší nebo rovno

logické operátory

and logický součin
or logický součet
not negace

logická jména

true pravda
false nepravda

- rozhodovací členy lze vnořovat
- není-li ani podmínka **if** ani žádná podmínka **elif** pravdivá, vykonají se příkazy následující klauzuli **else**, pokud je v příkazu uvedena



Máme zadanou hodnotu A a rozhodujeme o hodnotách B , C

- Hodnota $B = A$, jestliže A je menší než nula.
- Hodnota $C = A$, jestliže A je větší než nula.
- Jestliže $A = 0$, výpočet se zastaví.

>restart:

>A:=0: # vstupni hodnota

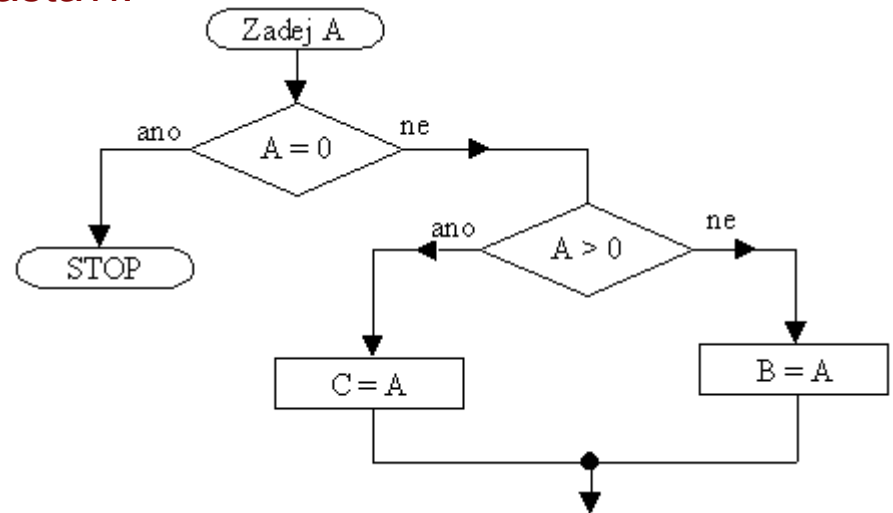
>if A=0 then

>ERROR(`Nulová hodnota A`)

>elif A > 0 then C:=A:

>elif A < 0 then B:=A:

>fi;



>restart:

>A:=0: # vstupni hodnota

>if A > 0 then

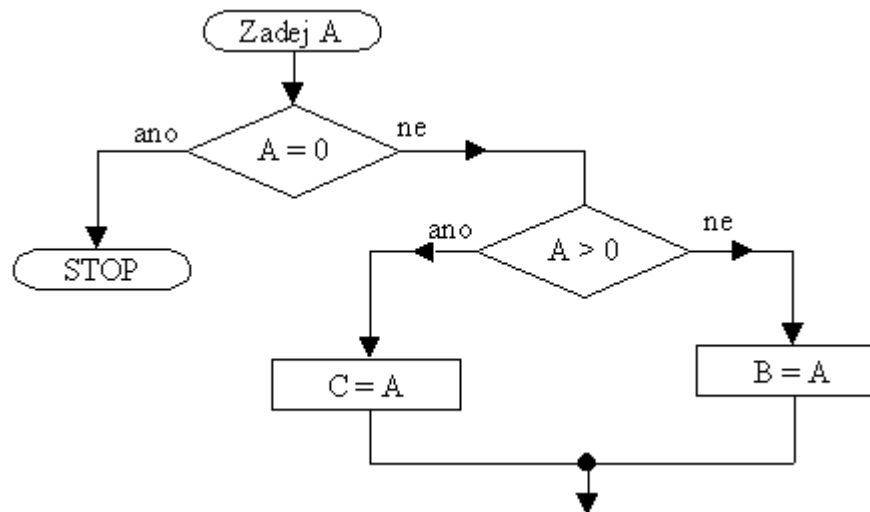
>C:=A:

>elif A < 0 then B:=A:

>else

>ERROR(`Nulová hodnota A`)

>fi;



>restart:

>A:=0: # vstupni hodnota

>if A <> 0 then

>if A > 0 then

>C:=A:

>else

>B:=A:

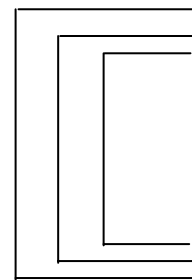
>fi;

>else

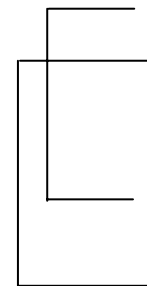
>ERROR(`Nulová hodnota A`)

>fi;

Vnoření rozhodovacích členů



správně



špatně



```
|for <jméno_proměnné>| |from <výraz>| |by <výraz>| |to <výraz>| |while <výraz>| do  
<posloupnost příkazů>  
od;
```

Příklad for ... do

```
> for i from 0 by 2 to 6 do  
> print(i):  
> od;
```

0
2
4
6

Proměnná *i* nabývá hodnoty:

od 0	from 0
do 6	to 6
s krokem 2	by 2

```
> for i from 3 by 1.5 to 6 do  
> print(i):  
> od;
```

3
4.5
6

násobek nemusí být celé číslo



Příklad **for ... do** vnořené cykly

```
> for i from 1 to 3 do  
> for j from 1 to 2 do  
> print(i,j):  
> od;  
> od;
```

```
1, 1  
1, 2  
2, 1  
2, 2  
3, 1  
3, 2
```



Příklad while ... do

```
> i := 0;  
> while i < 6 do  
> i := i + 2;  
> od;  
  
i := 0  
i := 2  
i := 4  
i := 6
```

Přerušení cyklu **break**

```
> k := 0;  
> while k < 100 do  
> k := k + 1;  
> if k = 5 then break fi;  
> od;  
  
k := 1  
k := 2  
k := 3  
k := 4  
k := 5
```



```
<jméno procedury>:= proc (<parametry>)  
local <místní proměnné>;  
options <parametry>;  
<soubor příkazů>;  
end;
```

nepovinné

Parametry **options** :

remember

builtin

operator

trace

arrow

angle

system

help:

```
> ?remember  
> ?builtin
```



```
> f := proc(x, y)
```

```
> x + y;
```

```
> end;
```

```
f := proc(x,y) x+y end
```

• definice procedury

• tělo procedury

• konec procedury

```
> f(4,5);
```

9

```
> g := f;
```

g := f

```
> g(4,5);
```

9

znázornění procedury:

```
> op(f);
```

```
proc(x,y) x+y end
```

```
> print(f);
```

```
proc(x,y) x+y end
```

```
> f := proc(x, y)
```

```
> x + y;
```

```
> end:
```

```
> f(w, z);
```

w + z



hledání maximální hodnoty ze tří čísel

```
> max3 := proc(a, b, c)
>   print('Hledani maxima z cisel', a, b, c);
>   if a < b then
>     if b < c then c else b fi
>   elif a < c then c
>   else a
>   fi;
> end;
```

definice procedury

tělo procedury

konec procedury

```
max3(8,1,5);
      Hledani maxima z cisel, 8, 1, 5
              8
```

```
> a:= max3(5,4,6);
      Hledani maxima z cisel, 5, 4, 6
              a := 6
```



```
> g := proc(a)
> a + 5;
> end:
> f := proc(x)
>   local y;
>   y := x + g(x);
>   if y>0 then
>     print(`vypocet`)
>     RETURN(y);
>   else
>     print(`nulova hodnota`)
>     RETURN(0);
>   fi:
> end:
```

```
> f(2);
```

vypocet
9

```
> f(-5);
```

nulova hodnota
0



```
> m:=proc()  
> local vystup:  
> if nargs=0 then ERROR(`zadne vstupni hodnoty `) fi:  
> if not(type([args], list(numeric)))  
>   then RETURN(`procname(args)`);  
> fi:  
> vystup := args[1];  
> end:
```

```
> m();  
Error, (in m) zadne vstupni hodnoty
```

```
> m(4);  
4
```

```
>m(7,8);  
7
```



```
> m:=proc()
> local vystup:
> if nargs=0 then ERROR(`zadne vstupni hodnoty `) fi:
> if not(type([args], list(numeric)))
>   then RETURN(`procname(args)`);
> fi:
> vystup := args[2];
> end:
```

```
> m();
      Error, (in m) zadne vstupni hodnoty
```

```
> m(4);
      Error, (in m) improper op or subscript selector
```

```
>m(7,8);
```

**list****table****array**

list

tvorba vektoru - ruční zadání

```
> data1 := [1, 2, 3];  
data1 := [1, 2, 3]  
> data2 := [4, 5, 6];  
data2 := [4, 5, 6]  
> data3 := [op(data1), op(data2)];  
data3 := [1, 2, 3, 4, 5, 6]
```

prvek vektoru

```
> data3[3];  
3
```

počet prvků vektoru

```
> n := nops(data3);  
n := 6
```



tvorba vektoru - zadání cyklem

```
> k := [0, 2, 4, 6]:
```

```
> for i from 1 to 4 do
```

```
> a[i] := k[i] *10:
```

```
> lprint(i, a[i]);
```

```
> od:
```

```
1      0
```

```
2      20
```

```
3      40
```

```
4      60
```

```
> a[2];
```

```
20
```

```
> a;
```

```
a
```

```
> a := convert(a, list);
```

```
a := [0, 20, 40, 60]
```



další možnost přepočtu vektoru (bez cyklu)

```
> Kdata := [0, 2, 4, 6]:  
> a := evalf(map(x -> x*10, Kdata));
```

```
> a;  
a := [0, 20, 40, 60]
```

```
> a[2];  
20
```



```
> VlnovaDelka := table([cervena=610, modra=480]);
```

```
      VlnovaDelka := table([  
        cervena=610  
        modra=480  
      ])
```

```
> VlnovaDelka[cervena];
```

```
      610
```

```
> VlnovaDelka[zelena]:=520:
```

```
> VlnovaDelka;
```

```
      VlnovaDelka
```

```
print(VlnovaDelka);
```

```
      VlnovaDelka := table([  
        zelena=520  
        cervena=610  
        modra=480  
      ])
```



```
> A := array([ [1,2,3],[x,x,5],[1/2, 4,-1]]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ x & x & 5 \\ 1/2 & 4 & -1 \end{bmatrix}$$

maticové operace

evalm

maticový a vektorový součet	+	evalm(A + B)
součin matice-matice	&*	evalm(A &* B)
součin matice-vektor		
inverzní matice	⁻¹	evalm(A ⁻¹)
exponent n	⁽ⁿ⁾	evalm(A ³)
číselný součin	*	evalm(2*A)



```
readdata("soubor", <typ>, <počet sloupců>)
```

datový soubor *data.txt*
existuje v adresáři *c:\temp*

data.txt

0	10
1	20
2	30

```
> a := readdata("c:/temp/data.txt", integer,2);
```

```
      a := [[0, 10], [1, 20], [2, 30]]
```

```
> a := readdata("c:/temp/data.txt", float,2);
```

```
      a := [[0, 10.], [1., 20.], [2., 30.]]
```

```
> readlib(readdata):
```



```
> a := readdata("c:/temp/data.txt", integer, 2);  
      a := [[0, 10], [1, 20], [2, 30]]  
  
> a[1];  
      [0, 10]  
  
> b:=covnert(a, table)  
      b:= table(  
        (1,1) = 0  
        (1, 2) = 10  
        (2, 1) = 1  
        (2, 2) = 20  
        (1, 3) = 2  
        (2, 3) = 30  
      )  
  
> b[1, 2];  
      10
```



```
> print(b);
```

```
      b := table([  
      (1,1) = 0  
      (1, 2) = 10  
      (2, 1) = 1  
      (2, 2) = 20  
      (1, 3) = 2  
      (2, 3) = 30  
      ])
```

Přepočet všech prvků

```
> c := map(x -> x^2, b);
```

```
      a := table([  
      (1,1) = 0  
      (1, 2) = 100  
      (2, 1) = 1  
      (2, 2) = 400  
      (1, 3) = 4  
      (2, 3) = 900  
      ])
```



```
save(<proměnná1>, <proměnná2>, ... <proměnnán>, `soubor`)
```

```
> save(c, "c:/temp/vystup.txt");
```