2. přednáška (grafika a maplovský programovací jazyk)

Grafika v Maplu

Mnoho možností nám poskytují balíky plots a plottools.

```
> restart;
> with(plots);
```

[*animate, animate3d, animatecurve, arrow, changecoords, complexplot, complexplot3d, conformal, conformal3d, contourplot, contourplot3d, coordplot, coordplot3d, densityplot, display, dualaxisplot, fieldplot, fieldplot3d, gradplot, gradplot3d, implicitplot, implicitplot3d, inequal, interactive, interactiveparams, intersectplot, listcontplot, listcontplot3d, listdensityplot, listplot, listplot3d, loglogplot, logplot, matrixplot, multiple, odeplot, pareto, plotcompare, pointplot, pointplot3d, polarplot, polygonplot, polygonplot3d, polyhedra_supported, polyhedraplot, rootlocus, semilogplot, setcolors, setoptions, setoptions3d, spacecurve, sparsematrixplot, surfdata, textplot, textplot3d, tubeplot*]

```
> with(plottools);
```

[*arc, arrow, circle, cone, cuboid, curve, cutin, cutout, cylinder, disk, dodecahedron, ellipse, ellipticArc, getdata, hemisphere, hexahedron, homothety, hyperbola, icosahedron, line, octahedron, parallelepiped, pieslice, point, polygon, project, rectangle, reflect, rotate, scale, semitorus, sphere, stellate, tetrahedron, torus, transform, translate*]

```
>
```

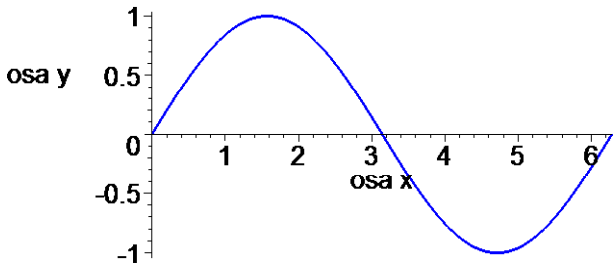
Explicitně zadané funkce jedné reálné proměnné

Vykreslení grafu funkce jedné proměnné:

```
> f:=sin(x);
```

$$f := \sin(x)$$

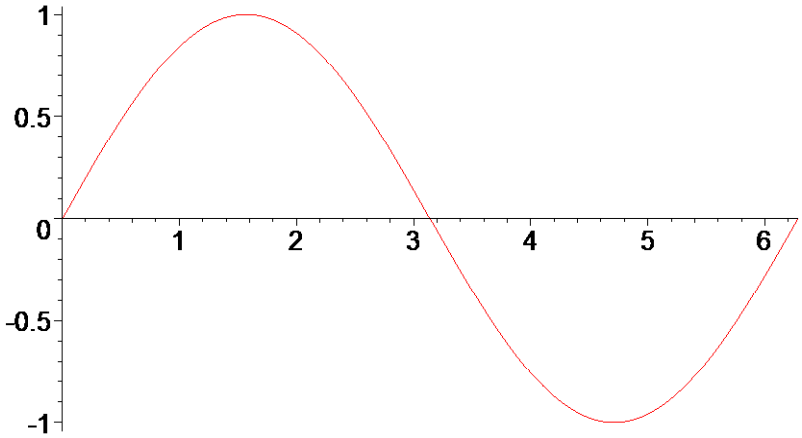
```
> plot(f,x=0..2*Pi, color=blue, thickness=3, labels=[`osa x`,`osa y`], legend=`sin(x)`);
```



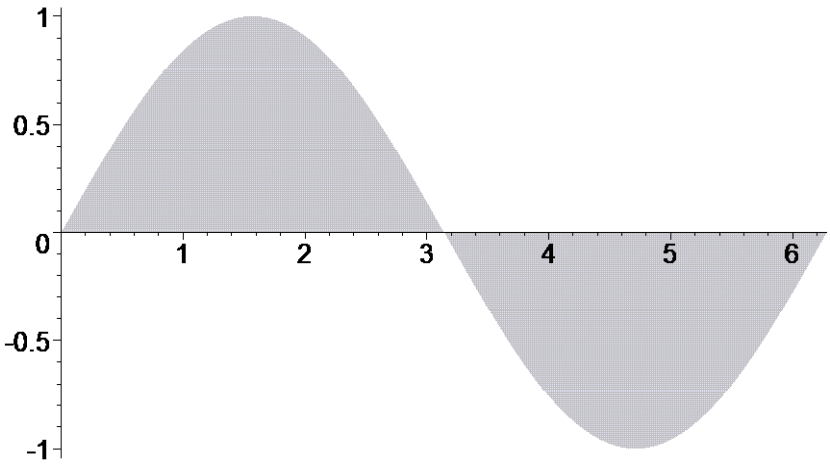

```
> f:=x->sin(x);
```

$$f := x \to \sin(x)$$

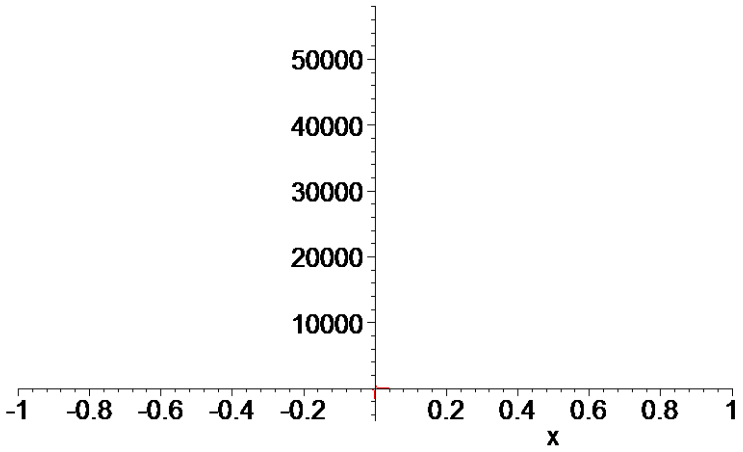
```
> plot(f,0..2*Pi);
```



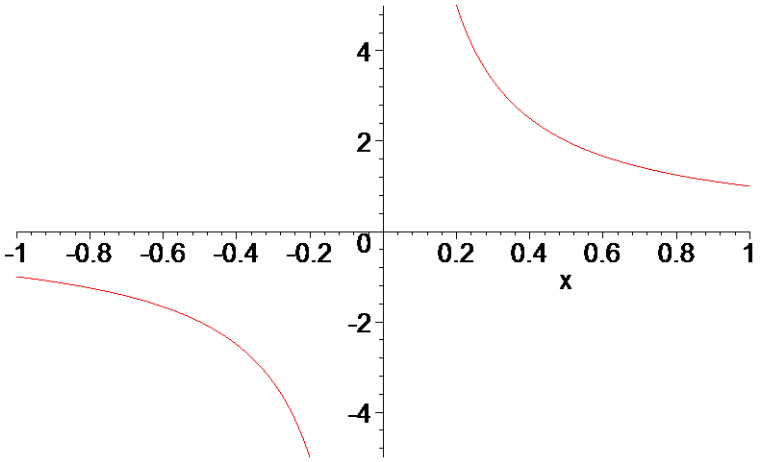

```
> plot(f,0..2*Pi,filled=true,color=grey);
```



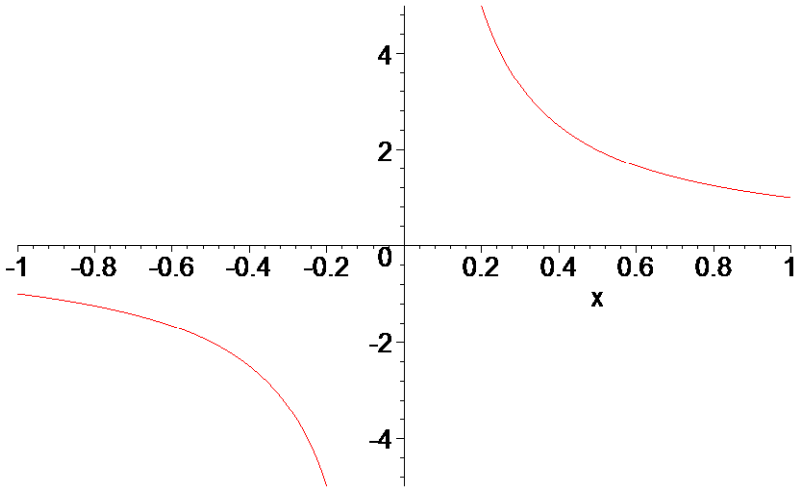

```
> plot(1/x,x=-1..1);
```



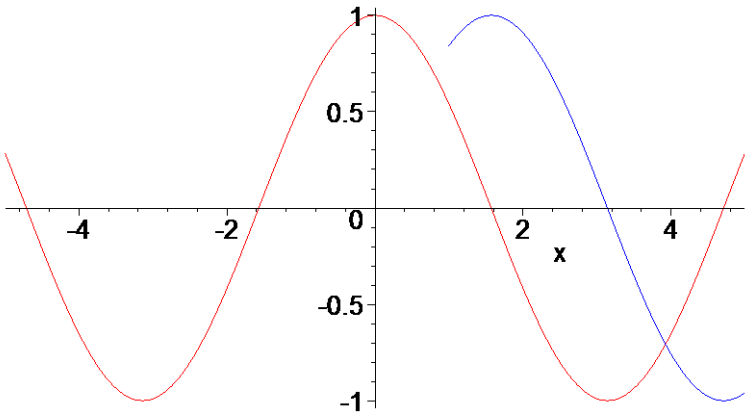

```
> plot(1/x,x=-1..1,-5..5);
```




```
> plot(1/x,x=-1..1,-5..5,discont=true);
```




```
> a:=plot(sin(x),x=1..5,color=blue):
> b:=plot(cos(x),x=-5..5,color=red):
> display([a,b]);
```



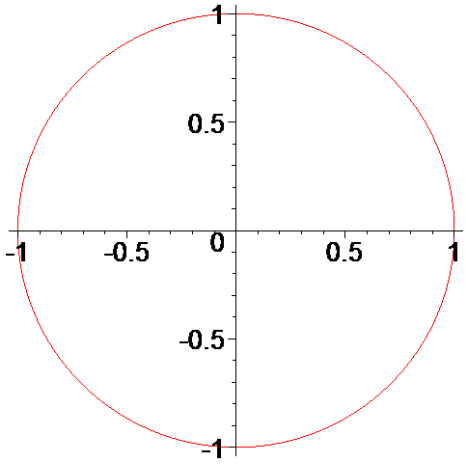
Příkaz **display** je součástí balíku **plots**.

```
>
```

Křivky v rovině dané parametricky

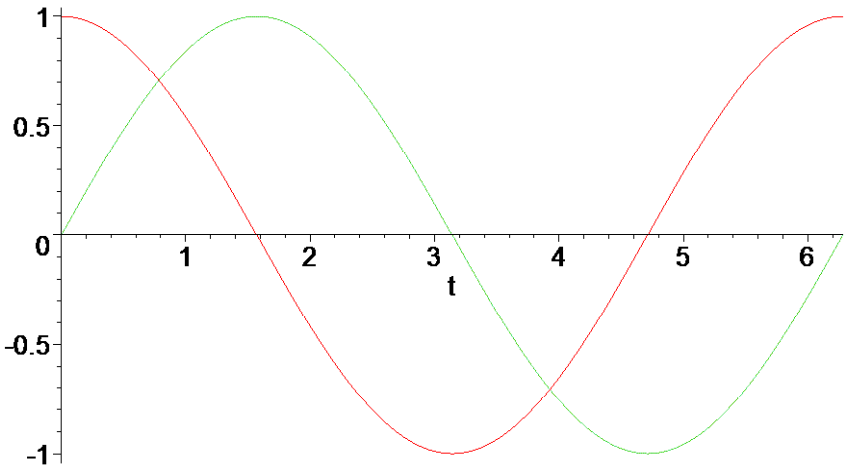
Křivka v rovině daná parametricky (x=f(t), y=g(t)):

```
> plot([cos(t),sin(t),t=0..2*Pi]);
```



Pozor !!!

```
> plot([cos(t),sin(t)],t=0..2*Pi);
```

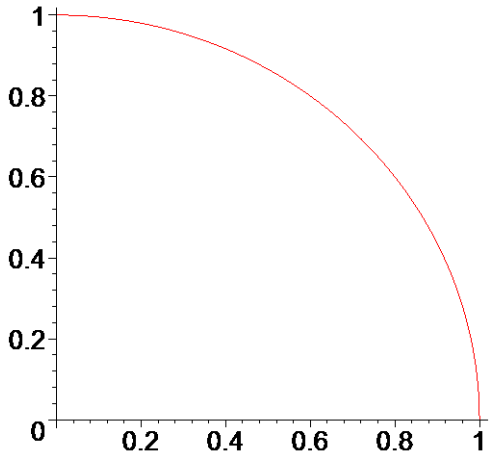


V tomto případě Maple nakreslil dvě funkce (sinus a kosinus) na intervalu <0, 2 π>.

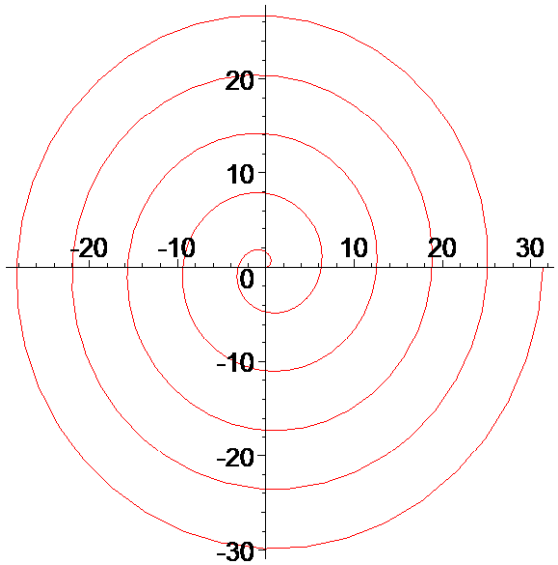
>

Křivky v rovině v polárních souřadnicích

```
> plots[polarplot](1,t=0..Pi/2);
```




```
> polarplot(t,t=0..10*Pi);
```

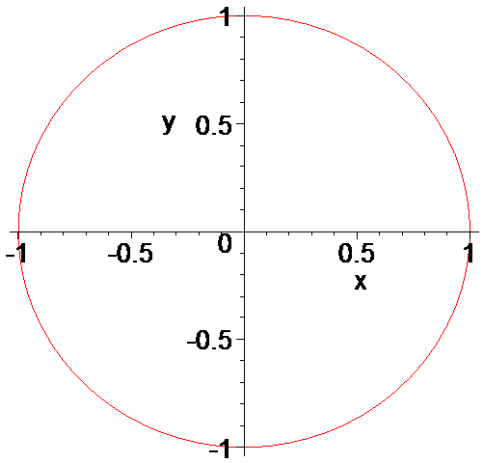


Vyzkoušejte to i v moderním rozhraní.

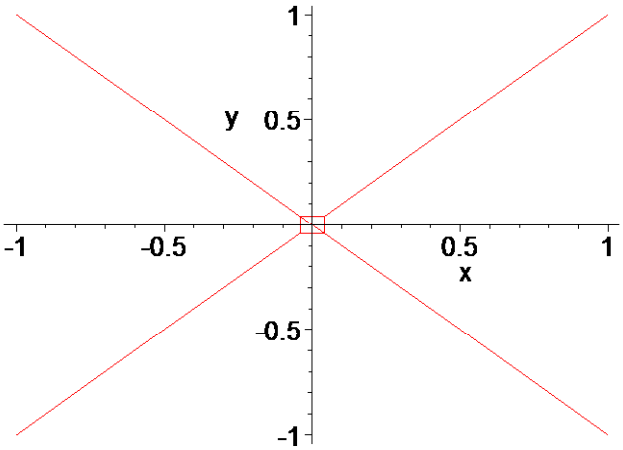
>

Grafy funkcí implicitně zadaných

```
> implicitplot(x^2+y^2=1,x=-1..1,y=-1..1);
```

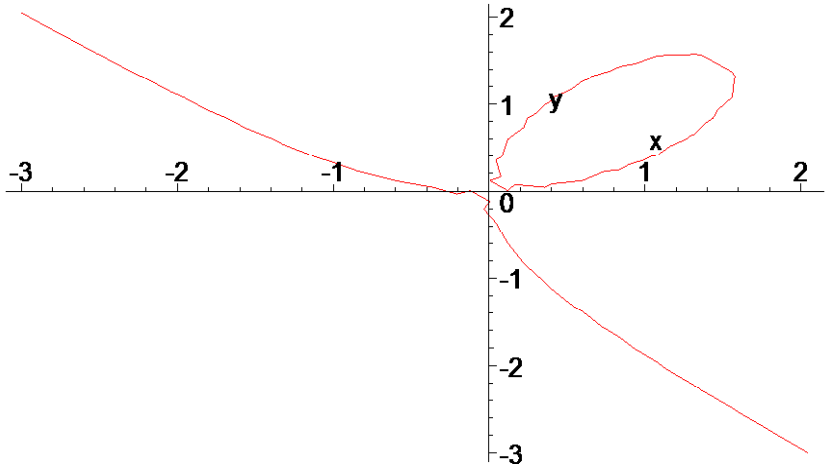



```
> implicitplot(x^2-y^2=0,x=-1..1,y=-1..1);
```



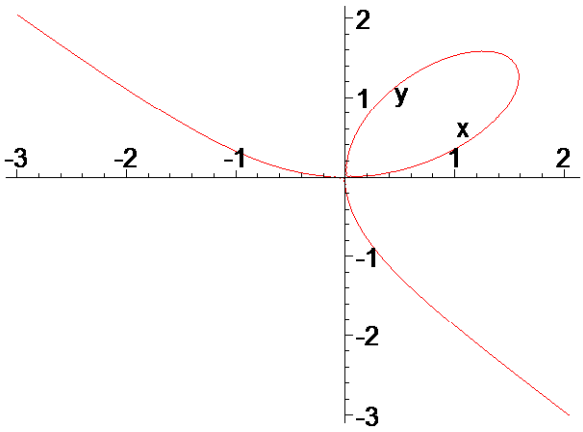
Descartesův list:

```
> implicitplot(x^3+y^3-3*x*y,x=-3..3,y=-3..3);
```



Vylepšení:

```
> implicitplot(x^3+y^3-3*x*y,x=-3..3,y=-3..3,grid=[100,100])
  ;
```

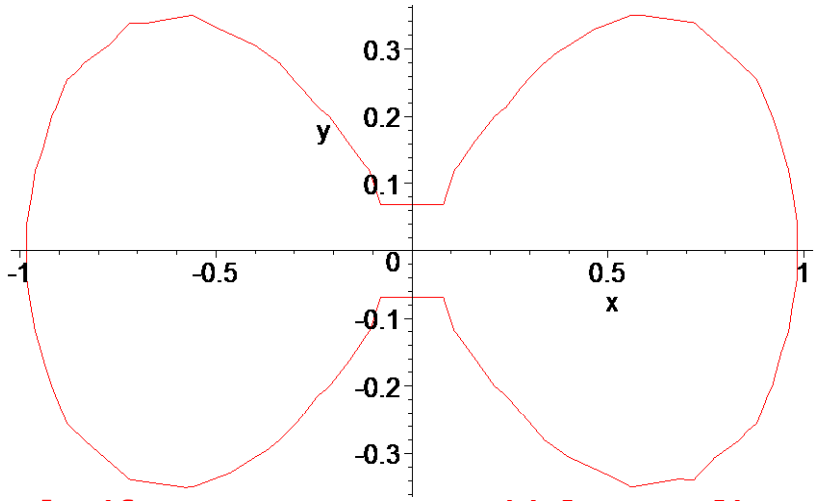


Lemniskáta:

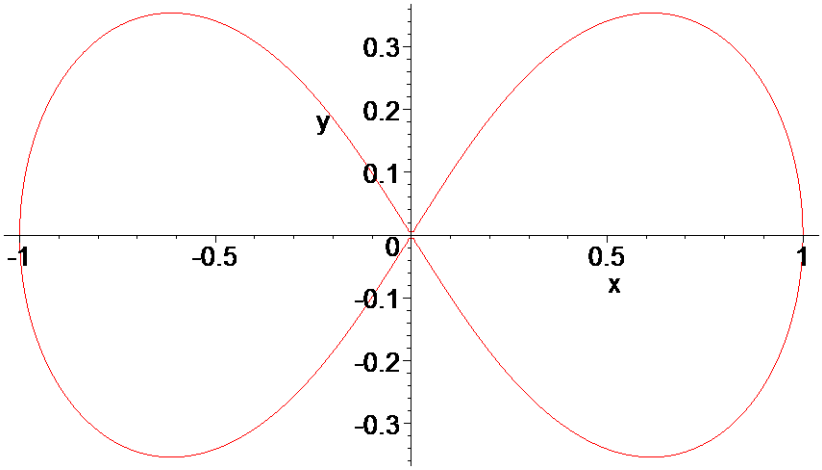
```
> f:=(x^2+y^2)^2+y^2-x^2=0;
```

$$f := (x^2 + y^2)^2 + y^2 - x^2 = 0$$

```
> implicitplot(f,x=-2..2,y=-1..1);
```



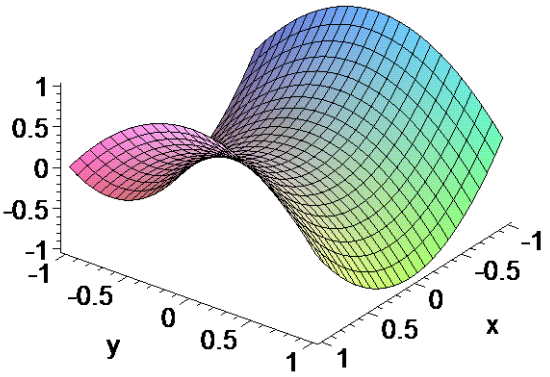

```
> implicitplot(f,x=-2..2,y=-1..1,grid=[300,300]);
```



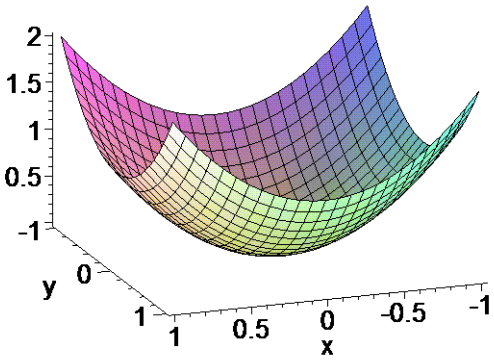

```
>
```

Grafy funkcí dvou proměnných

```
> plot3d(x^2-y^2,x=-1..1,y=-1..1,axes=framed);
```

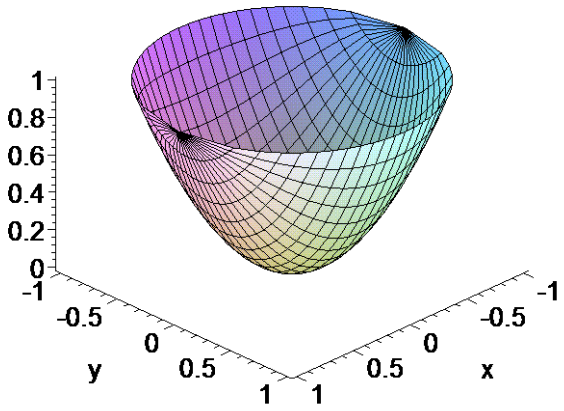



```
> plot3d(x^2+y^2,x=-1..1,y=-1..1,axes=framed);
```



Meze nemusí být konstantní:

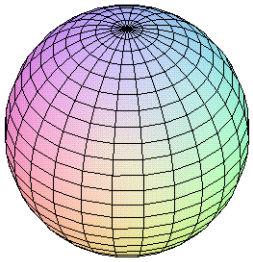
```
> plot3d(x^2+y^2,x=-1..1,y=-sqrt(1-x^2)..sqrt(1-x^2),axes=fr
  amed);
```




```
>
```

Plochy v prostoru dané parametricky

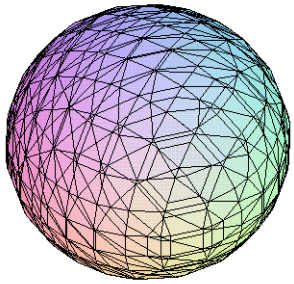
```
> plot3d([cos(t)*cos(s),sin(t)*cos(s),sin(s)],t=0..2*Pi,s=-P
  i/2..Pi/2);
```



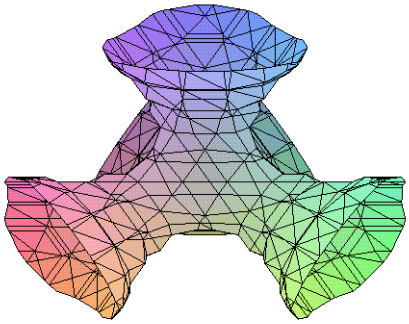
```
>
```

Funkce dvou proměnných daná implicitně

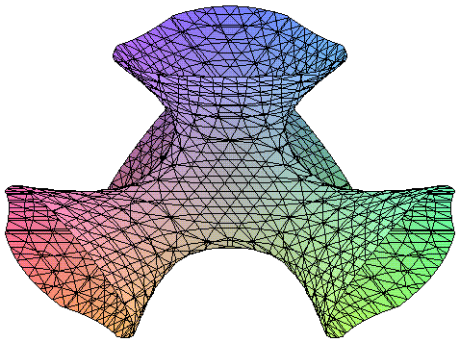
```
> implicitplot3d(x^2+y^2+z^2=1,x=-1..1,y=-1..1,z=-1..1);
```



```
> implicitplot3d(x^3+y^3+z^3+1=(x+y+z+1)^3, x=-2..2,
  y=-2..2, z=-2..2);
```



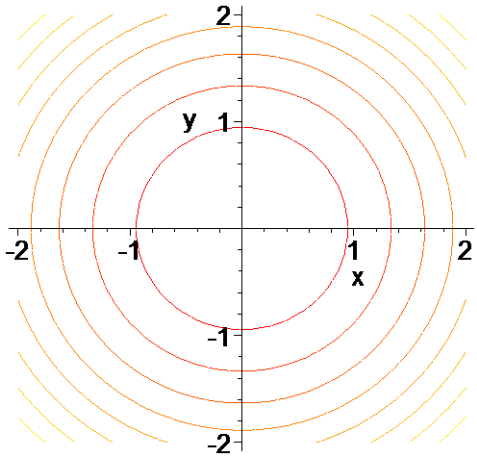
```
> implicitplot3d(x^3+y^3+z^3+1=(x+y+z+1)^3, x=-2..2,
  y=-2..2, z=-2..2, grid=[20,20,20]);
```



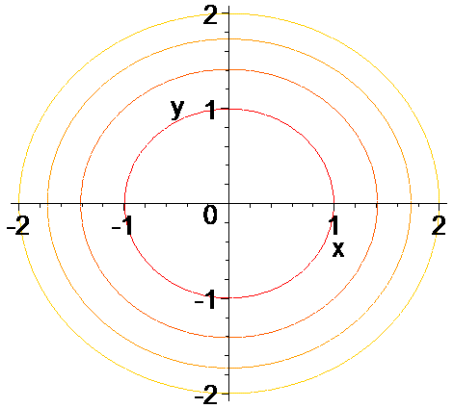
```
>
```

Vrstevnice funkce dvou proměnných

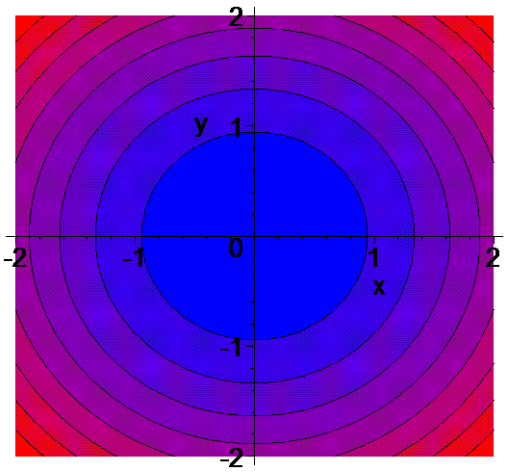
```
> contourplot(x^2+y^2,x=-2..2,y=-2..2);
```



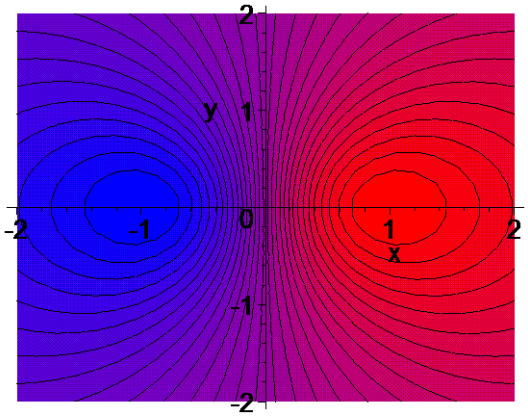

```
> contourplot(x^2+y^2,x=-2..2,y=-2..2,contours=[1,2,3,4]);
```



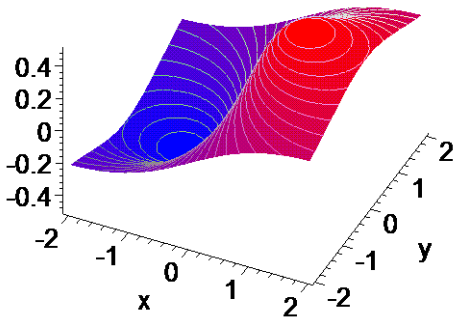

```
> contourplot(x^2+y^2,x=-2..2,y=-2..2,coloring=[blue,red],fi
  lled=true);
```



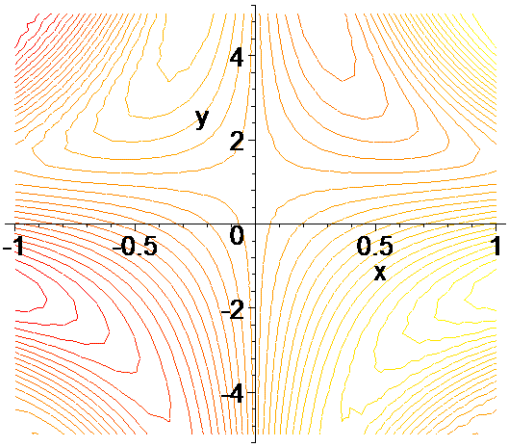

```
> contourplot(x/(x^2+y^2+1),x=-2..2,y=-2..2,coloring=[blue,r
  ed],filled=true,contours=30);
```



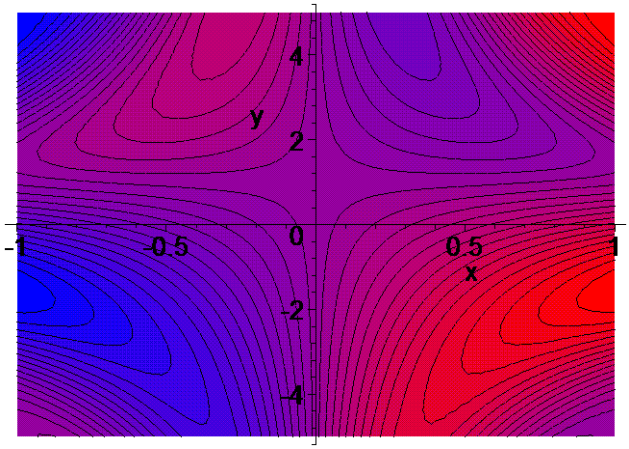

```
> contourplot3d(x/(x^2+y^2+1),x=-2..2,y=-2..2,coloring=[blue
  ,red],filled=true,contours=30,axes=framed);
```




```
> contourplot(x-sin(x*y),x=-1..1,y=-5..5,contours=30);
```



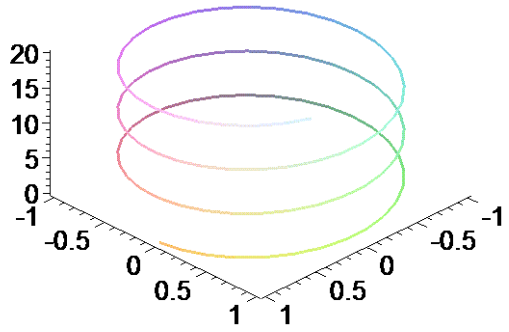

```
> contourplot(x-sin(x*y),x=-1..1,y=-5..5,contours=30,grid=[5
  0,50],coloring=[blue,red],filled=true);
```




```
>
```

Křivky v prostoru

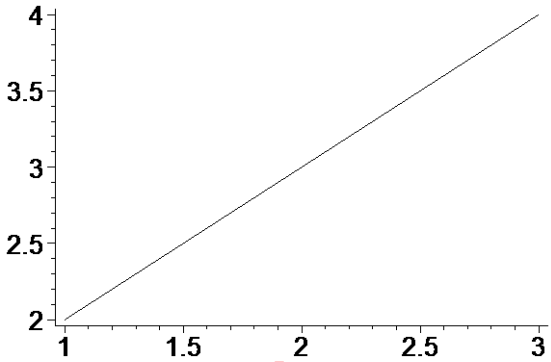
```
> spacecurve([cos(t),sin(t),t],t=0..20,axes=framed,numpoints
  =100,thickness=3);
```



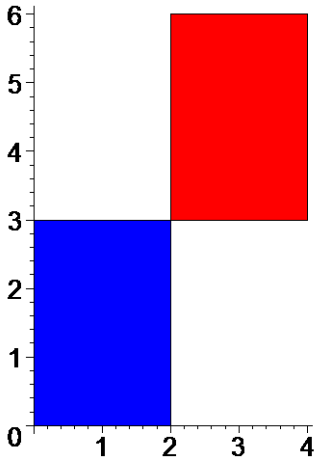

```
>
```

Další ukázky možností balíků plots a plottools

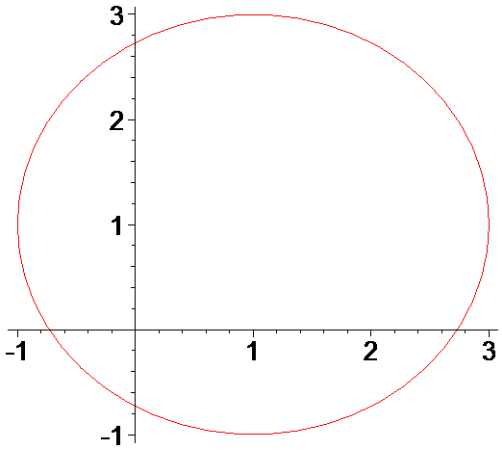
```
> l:=line([1,2],[3,4]):
> display(l);
```



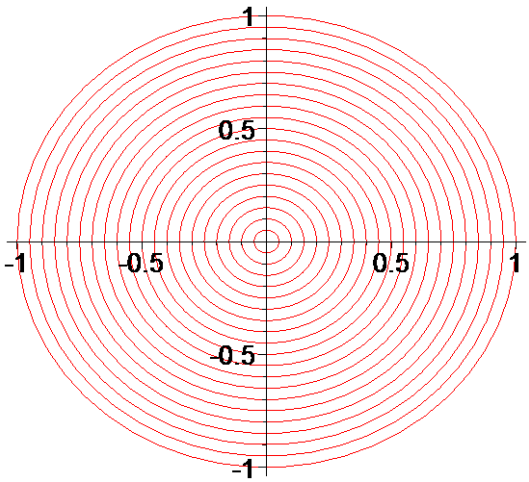

```
> r1:=rectangle([2,3],[4,6],color=red):
> r2:=rectangle([0,0],[2,3],color=blue):
> display(r1,r2,scaling=constrained);
```



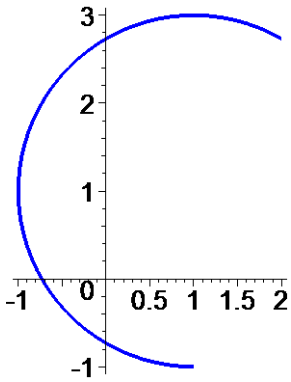

```
> c:=circle([1,1],2,color=red):
> display(c);
```




```
> cc:=seq(circle([0,0],i/20,color=red),i=1..20):
> display([cc]);
```



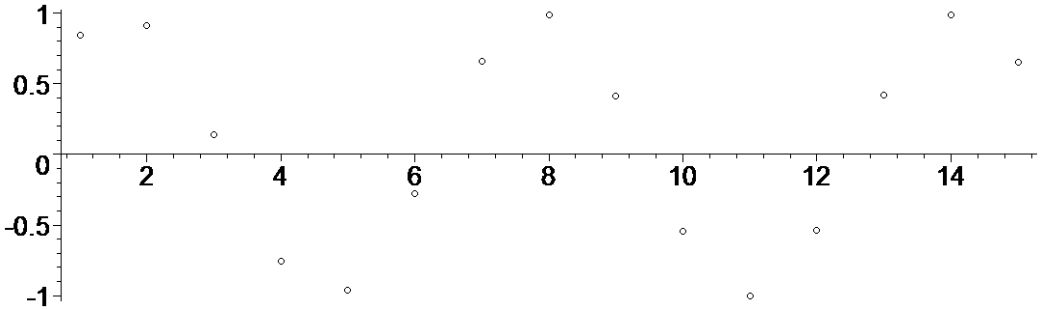

```
> a:=arc([1,1],2,Pi/3..3/2*Pi,color=blue,thickness=4):
> display(a,scaling=constrained);
```



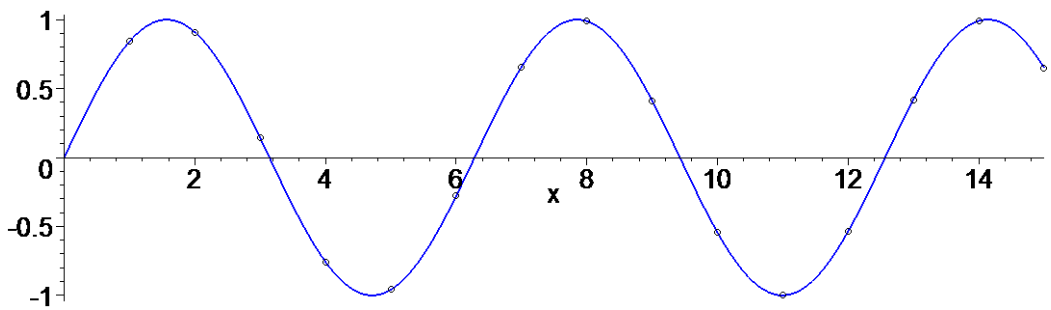

```
>
> s:=seq([n,sin(n)],n=1..15);
```

$$s := [1, \sin(1)], [2, \sin(2)], [3, \sin(3)], [4, \sin(4)], [5, \sin(5)], [6, \sin(6)], [7, \sin(7)], [8, \sin(8)], [9, \sin(9)], [10, \sin(10)], [11, \sin(11)], [12, \sin(12)], [13, \sin(13)], [14, \sin(14)], [15, \sin(15)]$$

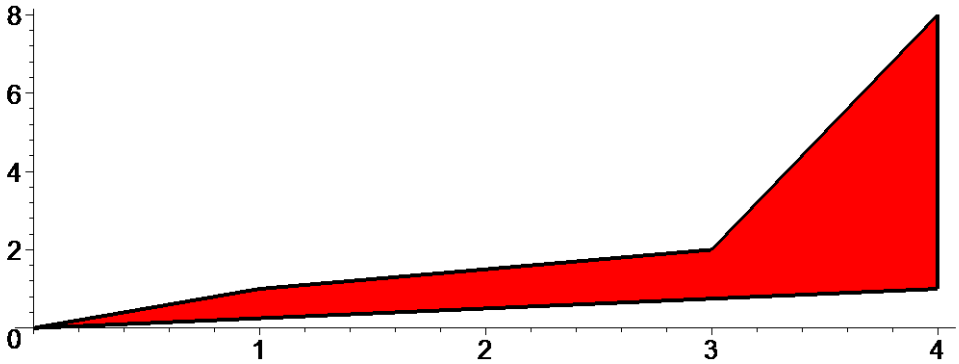
```
> pl1:=pointplot([s],symbol=circle,symbolsize=15):
> pl2:=plot(sin(x),x=0..15,color=blue,thickness=2):
> display([pl1]);
```



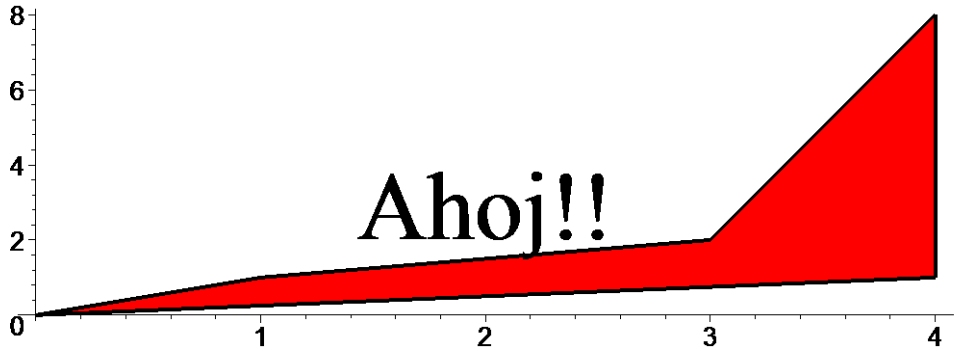

```
> display([pl1,pl2]);
```




```
> p:=polygonplot([[4,1],[0,0],[1,1],[3,2],[4,8]],thickness=4
  ,color=red):
> display(p);
```




```
> t:=textplot([2,3,"Ahoj!!"],font=[TIMES,ROMAN,36]):
> display([p,t]);
```




```
>
```

Vlastní procedury a maplovský programovací jazyk

Vlastní procedury tvoříme příkazem **proc**. Struktura vypadá následovně:

název funkce := proc(parametr1,parametr2,....)
.... tělo funkce .....
end;

```
> soucet:=proc(a,b)
> a+b;
> end;
```

soucet := **proc**(*a*, *b*) *a* + *b* **end proc**

```
> soucet(13,45);
```

```
> s:=soucet(23,12);
```

$$s := 35$$

V Maplu můžeme napsat proceduru, která nemá pevný počet parametrů.

```
> soucet:=proc() local n,s,i;
> n:=nargs;
> sum(args[i],i=1..n);
> end;
```

$$soucet := \mathbf{proc}() \ \mathbf{local}\ n, s, i;\ n := \mathrm{nargs};\ \mathrm{sum}(\mathrm{args}[i], i = 1 \mathinner{..} n)\ \mathbf{end\ proc}$$

```
> soucet(1,23,4,2,3);
```

$$33$$

```
>
```

Pro tvorbu složitějších procedur budeme potřebovat následující:

Větvení (if ... then ... elif ... else ... fi)

Napište proceduru, která vrací $\frac{n}{2}$ pro sudé n a $3n+1$ pro liché n (n je přirozené).

```
> p:=proc(n)
> if irem(n,2)=0 then n/2 else 3*n+1 fi;
> end;
```

$$p := \mathbf{proc}(n)\ \mathbf{if}\ \mathrm{irem}(n, 2) = 0\ \mathbf{then}\ 1 / 2{*}n\ \mathbf{else}\ 3{*}n + 1\ \mathbf{end\ if}\ \mathbf{end\ proc}$$

```
> p(10);
```

$$5$$

```
> p(11);
```

$$34$$

Lze i jinak:

```
> pp:=n->if irem(n,2)=0 then n/2 else 3*n+1 fi;
```

$$pp := n \rightarrow \mathbf{if}\ \mathrm{irem}(n, 2) = 0\ \mathbf{then}\ \frac{1}{2} n\ \mathbf{else}\ 3\, n + 1\ \mathbf{end\ if}$$

```
> pp(10);
```

$$5$$

```
> pp(11);
```

$$34$$

```
>
```

3n+1 problém:

Definujme posloupnost a1 (libovolné přirozené), a2=p(a1), a3=p(a2), a4=p(a3), ...
Problém: Je v takové posloupnosti vždy číslo 1 ???

Funguje to pro všechna a1 až do 19. 2^{58} (bylo experimentálně ověřeno - r. 2008). Obecně se neví.

Napište proceduru, která pro daná dvě čísla vypíše jejich aritmetický a geometrický průměr.

```
> prumer:=proc(x,y) local a,g;
> a:=(x+y)/2;
> print(`Aritmetický průměr:`,a);
> if x>=0 and y>=0 then
>   g:=sqrt(x*y);
>   print(`Geometrický průměr:`,g);
> else
>   print(`Geometrický průměr nemá smysl`);
> fi;
> end;
```

prumer := **proc**(*x*, *y*)

local *a*, *g*;

a := 1 / 2∗*x* + 1 / 2∗*y*;

print(`*Aritmetický průměr:*`, *a*);

if 0 ≤ *x* **and** 0 ≤ *y* **then** *g* := sqrt(*x*∗*y*); print(`*Geometrický průměr:*`, *g*)

else print(`*Geometrický průměr nemá smysl*`)

end if

end proc

```
> prumer(4,9);
```

Aritmetický průměr:, $\frac{13}{2}$

Geometrický průměr:, 6

```
> prumer(1,-2);
```

Aritmetický průměr:, $\frac{-1}{2}$

Geometrický průměr nemá smysl

Napište proceduru, která pro záporná čísla vrací -1, pro čísla z intervalu <0,1> vrací původní číslo a pro čísla z (1,nekonečno) vrací (dolní) celou část původního čísla.

```
> f:=proc(x)
> if x<0 then -1
> elif x>1 then floor(x)
> else x;
```

```
> fi;
> end;
```

$f := \mathbf{proc}(x)\ \mathbf{if}\ x < 0\ \mathbf{then}\ -1\ \mathbf{elif}\ 1 < x\ \mathbf{then}\ \mathrm{floor}(x)\ \mathbf{else}\ x\ \mathbf{end\ if\ end\ proc}$

```
> f(-10);
```

$$-1$$

```
> f(0);
```

$$0$$

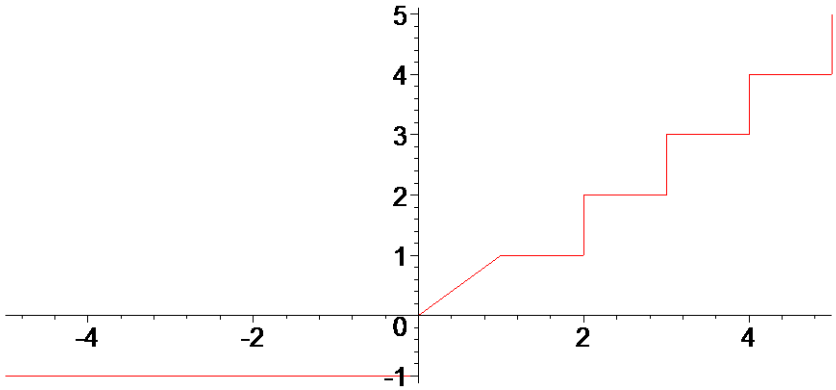
```
> f(3/4);
```

$$\frac{3}{4}$$

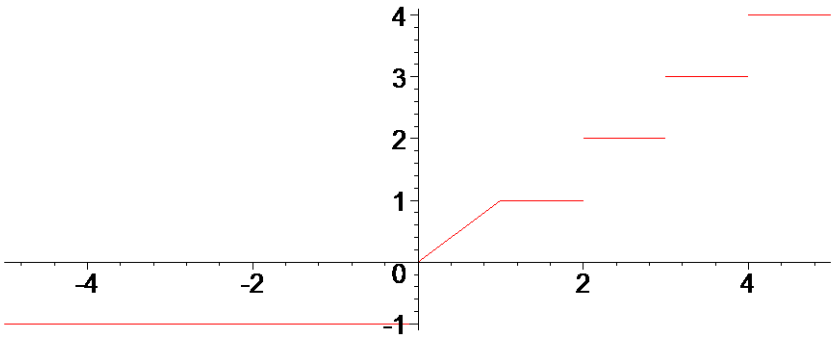
```
> f(12/5);
```

$$2$$

```
> plot(f,-5..5);
```




```
> plot(f,-5..5,discont=true);
```




```
> f(Pi);
Error, (in f) cannot determine if this expression is true or false: Pi <
0
```

Maple neumí rozhodnout, zda $\pi>0$ nebo $\pi<0$!!!!!!

Jak to spravíme???

```
> f:=proc(x)
> if evalf(x)<0 then -1
> elif evalf(x)>1 then floor(x)
> else x;
> fi;
```

```
> end;
```

$f :=$ **proc**(x) **if** evalf(x) < 0 **then** -1 **elif** 1 < evalf(x) **then** floor(x) **else** x **end if end proc**

```
> f(-10);
```

$$-1$$

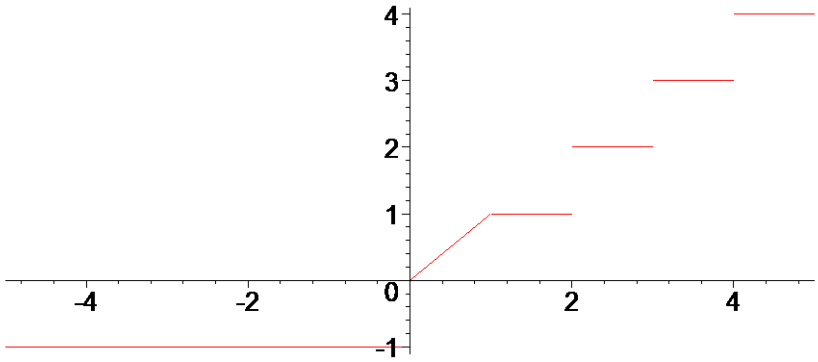
```
> f(1/2);
```

$$\frac{1}{2}$$

```
> f(Pi);
```

$$3$$

```
> plot(f,-5..5,discont=true);
```



Jiný způsob, jak to opravit:

```
> evalb(Pi=0);
```

false

```
> evalb(Pi>0);
```

$$0 < \pi$$

```
> evalb(22/7>0);
```

true

```
> max(Pi,0);
```

$$\pi$$

Uvědomme si, že $x \le y$, právě když $\max(x, y) = y$ a to nastane, právě když $\min(x, y) = x$.

Pro ostré nerovnosti je to trošku složitější:
$x < y$, právě když neplatí $\max(x, y) = x$ a to nastane, právě když neplatí $\min(x, y) = y$.

```
> f:=proc(x)
> if not min(x,0)=0 then -1
> elif not max(x,1)=1 then floor(x)
> else x;
> fi;
> end;
```

$f :=$ **proc**(x)

```
    if not (min(x, 0) = 0) then -1
    elif not (max(x, 1) = 1) then floor(x)
    else x
    end if
end proc
```

```
> f(-10);
```

$$-1$$

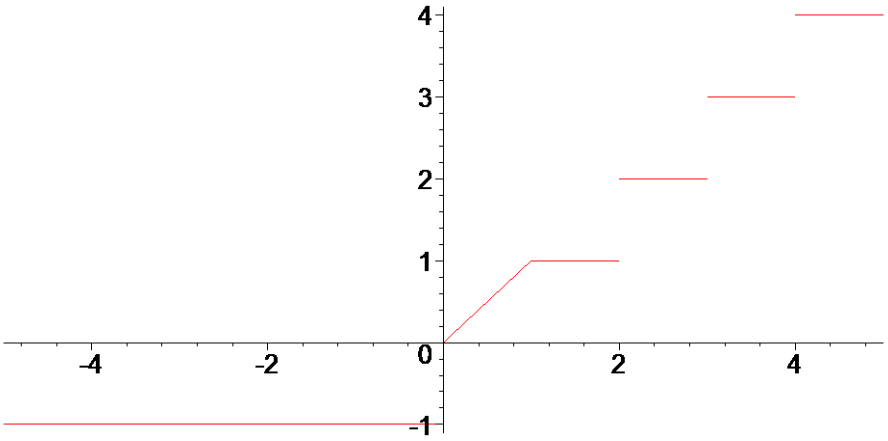
```
> f(3/4);
```

$$\frac{3}{4}$$

```
> f(Pi);
```

$$3$$

```
> plot(f,-5..5,discont=true);
```



Další možností, jak funkci definovat, je použít užitečný příkaz **piecewise**.

```
> f:=x->piecewise(x<0,-1,x>1,floor(x),x);
```

$$f := x \rightarrow \text{piecewise}(x < 0, -1, 1 < x, \text{floor}(x), x)$$

```
> f(-10);
```

$$-1$$

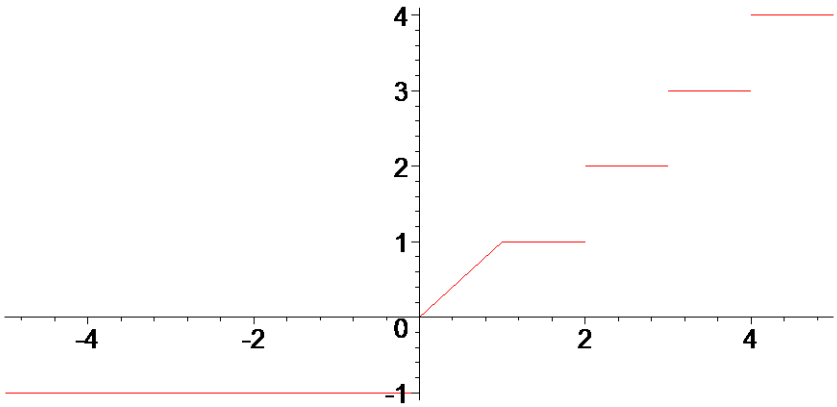
```
> f(3/4);
```

$$\frac{3}{4}$$

```
> f(Pi);
```

$$3$$

```
> plot(f,-5..5,discont=true);
```




```
>
```

Cyklus for

Cyklus **for** používáme zejména v případech, kdy víme, kolikrát se bude cyklus opakovat.

Napište proceduru, která vypíše prvních n prvočísel.

```
> prvoc:=proc(n) local i;
> for i from 1 to n do
>  print(ithprime(i));
> od;
> end;
```

prvoc := **proc**(*n*) **local** *i*; **for** *i* **to** *n* **do** print(ithprime(*i*)) **end do end proc**

```
> prvoc(10);
```

2
3
5
7
11
13
17
19
23
29

Lze i jinak (bez cyklu, pomocí příkazu **seq**):

```
> prvoc:=proc(n) local i;
> seq(ithprime(i),i=1..n);
> end;
```

prvoc := **proc**(*n*) **local** *i*; seq(ithprime(*i*), *i* = 1 .. *n*) **end proc**

```
> prvoc(10);
```

2, 3, 5, 7, 11, 13, 17, 19, 23, 29

```
> prvoc(10)[5];
```

11

Napište proceduru, která vypíše všechna přirozených čísla tvaru $3k+2$, která jsou nejvýše rovna danému číslu.

```
> vypis:=proc(n) local i;
> for i from 2 by 3 to n do
>  print(i);
> od;
> end;
```

vypis := **proc**(*n*) **local** *i*; **for** *i* **from** 2 **by** 3 **to** *n* **do** print(*i*) **end do end proc**

```
> vypis(20);
```

2

5

8

11

14

17

20

```
>
```

Cyklus while

Tento cyklus obvykle použijeme, pokud dopředu nevíme počet opakování.

Napište proceduru, která vypíše všechna prvočísla nejvýše rovna danému přirozenému číslu.

```
> vypis:=proc(n) local i;
> i:=1;
> while ithprime(i)<=n do
>   print(ithprime(i));
>   i:=i+1;
> od;
> end;
```

vypis := **proc**(*n*)
local *i*;
i := 1; **while** ithprime(*i*) ≤ *n* **do** print(ithprime(*i*)); *i* := *i* + 1 **end do**
end proc

```
> vypis(20);
```

2
3
5
7
11
13
17
19
9

Procedura vypsala ještě navíc číslo 9. Proč ???

```
> v:=vypis(20);
```

2
3
5
7
11
13
17
19
$v := 9$

```
> v;
```

9

```
>
```

```
> vypis:=proc(n) local i;
> i:=1;
> while ithprime(i)<=n do
>   print(ithprime(i));
>   i:=i+1;
> od;
> return;
> end;
```

vypis := **proc**(*n*)
local *i*;
i := 1; **while** ithprime(*i*) ≤ *n* **do** print(ithprime(*i*)); *i* := *i* + 1 **end do**; **return**
end proc

```
> vypis(20);
```

2
3

```
                                5
                                7
                               11
                               13
                               17
                               19
```

Kombinace for a while:

Napište proceduru, která vypíše nejvýše prvních n prvočísel, která jsou nejvýše rovna danému m.

```
> vypis:=proc(n,m) local i;
> for i from 1 to n while ithprime(i)<=m do
>   print(ithprime(i));
> od;
> end;
```

vypis := **proc**(*n*, *m*)
local *i*;
    for *i* **to** *n* **while** ithprime(*i*) ≤ *m* **do** print(ithprime(*i*)) **end do**
end proc

```
> vypis(10,100);
                                2
                                3
                                5
                                7
                               11
                               13
                               17
                               19
                               23
                               29
> vypis(10,20);
                                2
                                3
                                5
                                7
                               11
                               13
                               17
```

```
>
```

Animace

Pokud chceme vytvořit nějakou složitější animaci, je dobré napsat si proceduru, která umí vykreslit obecný snímek požadované animace, a poté použít příkaz animate.

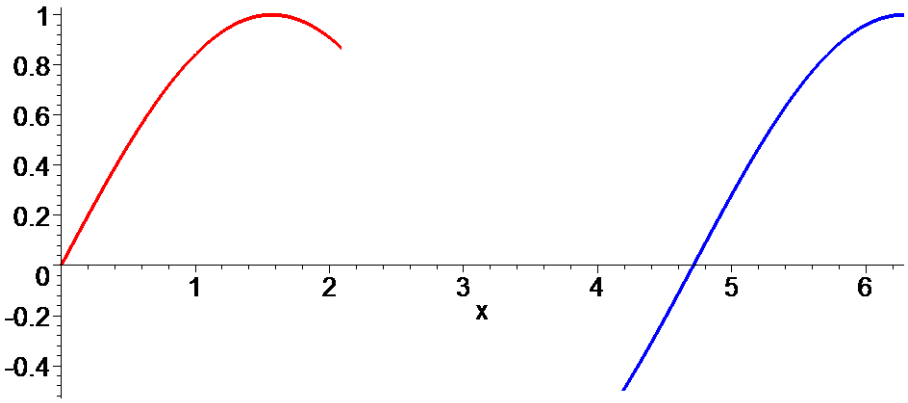
```
> restart;
> with(plots):
```

Následující animace zobrazuje vykreslování funkcí sinus a kosinus na intervalu <0, 2 π>. Přitom se funkce sinus vykresluje zleva doprava a funkce kosinus naopak.

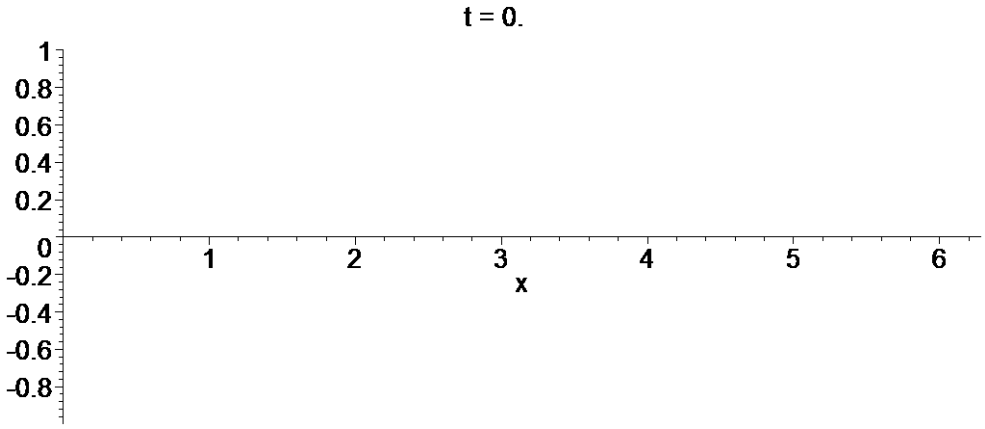
```
> s:=proc(t) local x,p1,p2;
> p1:=plot(sin(x),x=0..t,thickness=4,color=red):
> p2:=plot(cos(x),x=2*Pi-t..2*Pi,thickness=4,color=blue):
> display([p1,p2]);
> end;
```

s := **proc**(*t*)
local *x*, *p1*, *p2*;
p1 := plot(sin(*x*), *x* = 0 .. *t*, *thickness* = 4, *color* = *red*);
p2 := plot(cos(*x*), *x* = 2∗π – *t* .. 2∗π, *thickness* = 4, *color* = *blue*);
plots:-*display*([*p1*, *p2*])
end proc

```
> s(2/3*Pi);
```




```
> animate(s,[t],t=0..2*Pi,frames=20);
```



>

Cvičení

1) Nakreslete křivku danou parametricky rovnicemi $x = \cos(t)^3$, $y = \sin(t)^3$, kde t probíhá interval <0, 2 π>. Jaká je délka této křivky?

2) Vykreslete vrstevnice (s kótami -10, -9, ..., 0, ..., 9, 10) funkce $f(x, y) = x^4 + y^4 - (x + y)^2$. Poznáte na základě těchto vrstevnic, kde má funkce f lokální extrémy?

3) Vytvořte proceduru, která pro dané přirozené číslo vypíše řešení výše zmiňovaného 3n+1 problému (výpis posloupnosti skončí, jakmile se v ní objeví jednička). Pomocí této procedury vyřešte $3n + 1$ problém pro výchozí číslo 8423.

4) Najděte nejmenší přirozené číslo n splňující podmínku $3125^{2741} \leq n!$.

5) Napište proceduru, která nakreslí anuloid daných rozměrů.

6) Napište proceduru, která do jednoho obrázku vykreslí danou funkci společně s první derivací na daném intervalu <*a*, *b*>. Funkci nakreslete modře a derivaci červeně.

7) Napište proceduru, která vypočte n-tý člen Fibonacciho posloupnosti (a1=1, a2=1, a3=a2+a1=2, a4=a3+a2=3, a5=a4+a3=5, ...). Jaký je 3849. člen?

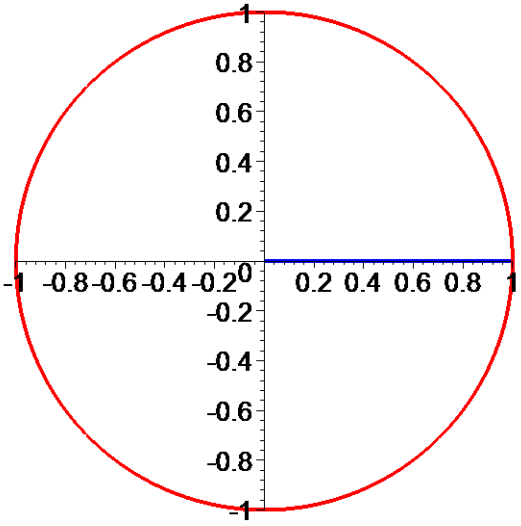
8) Nakreslete elipsu se středem v bodě (1,3) a s poloosami 7 a 4.

9) Napište proceduru, která vypočte největší společný dělitel dvou přirozených čísel pomocí Eukleidova algoritmu. Můžete přitom využívat funkcí **irem** a **iquo**.
Správnost procedury zkontrolujte na konkrétním příkladu pomocí maplovského příkazu **igcd**.

10) Napište proceduru, která najde dokonalé číslo větší než dané přirozené číslo.
Přirozené číslo se nazývá dokonalé, jestliže je rovno součtu všech svých kladných dělitelů (s výjimkou čísla samotného).
Např. číslo 6 je dokonalé, neboť dělitelé čísla 6 jsou 1, 2, 3, 6 a 6=1+2+3. Další dokonalé číslo je 28 (dělitelé jsou 1, 2, 4, 7, 14, 28 a platí 1+2+4+7+14=28).
Pokud tedy na vstupu bude např. 3, procedura by měla najít nejbližší dokonalé číslo větší než 3, tj. 6. Pokud bude na vstupu 6, mělo by se vrátit 28, atd.
Pomocí této procedury najděte ještě alespoň dvě další dokonalá čísla.

11) Pokuste se nakreslit šachovnici *n* x *n*, kde *n* je parametr.

12) Vytvořte následující animaci:



Pokud jste došli až sem, spusťte následující příkaz:

```
> plot([6*cos(t),2/3*((6*cos(t))^2+abs(6*cos(t))-6)/((6*cos(t))
  ^2+abs(6*cos(t))+2)+6*sin(t),t=0..2*Pi],filled=true,color=red
  ,axes=none);

>
```