Cvičení 9: Delphi - křivka daná explicitně, parametricky, polárně, speciální křivky

1 Opakování - unita Graph2D.pas

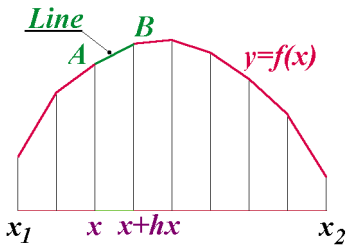
- Nezbytné soubory: *Graph2d.pas* a *Graph2d.dfm*
- Přidání se provádí pomocí menu: *Project - Add To Project...*
- Pro přístup k novým procedurám se využívá objektu: `Draw2D`
- Před použitím nových procedur objektu `Draw2D` je vždy potřeba zobrazit nový formulář a inicializovat obraz příkazy:

```
Draw2D.Visible := True;
Draw2D.InitImage(500,500);
```

- Důležité procedury a funkce:
  - `procedure InitImage(Width,Height:Integer);`
  - `procedure Scale(x1,x2,y1,y2:double);`
  - `procedure XAxis(x1,x2,y:double;Red,Green,Blue:byte);`
    `procedure YAxis(y1,y2,x:double;Red,Green,Blue:byte);`
  - `procedure XScale(x1,x2,y:double;Red,Green,Blue:byte);`
    `procedure YScale(y1,y2,x:double;Red,Green,Blue:byte);`
  - `procedure PutPoint(X:TPoint;Red,Green,Blue:byte);`
  - `procedure GetPoint(X:TPoint;var Red,Green,Blue:byte);`
  - `procedure Line(X,Y:TPoint;Red,Green,Blue:byte);`
  - `procedure Triangle(A,B,C:TPoint;Red,Green,Blue:byte);`
  - `procedure FillTriangle(A,B,C:TPoint;Red,Green,Blue:byte);`
  - `procedure Rectangle(A,B,C,D:TPoint;Red,Green,Blue:byte);`

2 Křivky typu $y = f(x)$

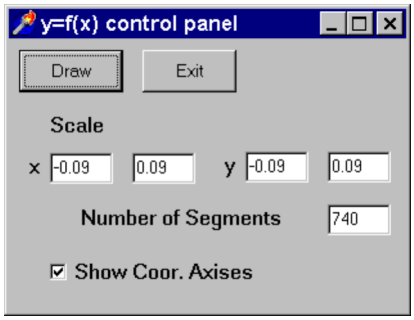
Z hlediska konstrukce se jedná o nejjednodušší křivky. Lze je konstruovat jako lomené čáry složené z úseček dle připojeného obrázku:



3 Příklad:

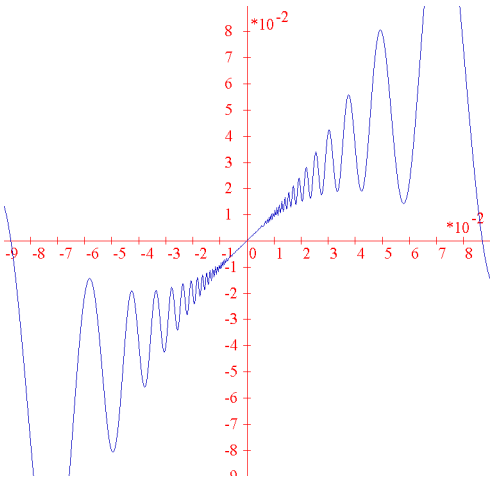
Sestrojme reakci na stisk tlačítka `procedure TForm1.btnDrawClick`, která sestrojí graf funkce $y = x+10x^2 \sin\frac{1}{x}$; $y(0) = 0$. Nejdříve vytvoříme ovládací panel ze známých komponent tak, jak je uvedeno na obrázku.

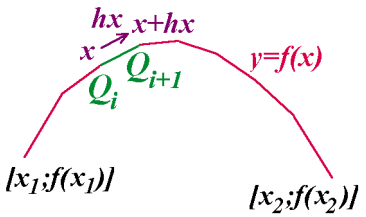
V panelu bude uživatel zadávat počet segmentů `NumberOfSegments`, z kterých má být křivka sestrojena. Nejdříve sestrojíme uživatelskou kreslicí plochu, souřadné osy a definujeme barvu sestrojované křivky. Vidíme, že funkce f je definována uvnitř procedury. Vlastní konstrukce spočívá v nastavení kroku hx a v postupné konstrukci úseček AB. Procedura reagující na stisk tlačítka *Draw* bude vypadat následovně:




```
procedure TForm1.btnDrawClick(Sender: TObject);
var x,hx              :Double;
    A,B               :TPoint;
    x1,x2,y1,y2       :Double;
    numberOfSegments :Integer;
    code              :Integer;

function f(x:Double):Double;
begin
  if x=0 then f:=0
         else f:=x+10*x*x*sin(1/x);
end;

begin
  val(edtX1.Text,x1,code);
  val(edtX2.Text,x2,code);
  val(edtY1.Text,y1,code);
  val(edtY2.Text,y2,code);
  val(edtNumberOfSegments.Text,numberOfSegments,code);

  Draw2D.Visible := True;
  Draw2D.InitImage(500,500);
  Draw2D.ClearImage(255,255,255);
  Draw2D.Scale(x1,x2,y1,y2);

  if chBoxAxis.Checked then
                       begin
                         Draw2D.XScale(x1,x2,0,255,0,0);
                         Draw2D.YScale(y1,y2,0,255,0,0);
                       end;
  hx:=(x2-x1)/numberOfSegments;
  x:=x1;
  A[1]:=x;
  A[2]:=f(x);
  repeat
    B[1]:=x+hx;
    B[2]:=f(x+hx);
    Draw2D.Line(A,B,0,0,255);
    A:=B;
    x:=x+hx;
  until x>x2
end;
```

V případě křivek bývá výhodné posloupnost bodů, kterými křivka prochází, nejdříve celou spočítat a teprve potom proložit křivku. Nám již známá unita *Graph2D.pas* obsahuje proceduru, která toto umožňuje.





4 Použití procedury `Draw2D.PolyLine`

Pozorný uživatel unity *Graph2D.pas* si jistě povšiml nově definováného typu `TArrayOfPoints = array [0..800] of TPoint;` což je posloupnost maximálně 800 bodů `TPoint`. Tento nový typ využívá procedura:

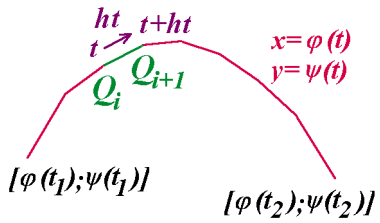
- `procedure TDraw2D.PolyLine(X:TArrayOfPoints;n:Word;Red,Green,Blue:Byte);`
  - propojení `n` bodů v množině bodů `X` barvou `Red,Green,Blue`.

Do proměnné typu `TArrayOfPoints` se budou ukládat body, kterými má procházet naše křivka. S nástroji, které již máme k dispozici, je její konstrukce jednoduchá. Upravme předchozí příklad. Zdrojový text bude velmi podobný předchozímu:

```
procedure TForm1.btnDrawClick(Sender: TObject);
var x,hx              :Double;
    x1,x2,y1,y2       :Double;
    numberOfSegments :Integer;
    code,i            :Integer;
    Q                 :TArrayOfPoints;

function f(x:Double):Double;
begin
  if x=0 then f:=0
         else f:=x+10*x*x*sin(1/x);
end;

begin
  val(edtX1.Text,x1,code);
  val(edtX2.Text,x2,code);
  val(edtY1.Text,y1,code);
  val(edtY2.Text,y2,code);
  val(edtNumberOfSegments.Text,numberOfSegments,code);

  Draw2D.Visible := True;
  Draw2D.InitImage(500,500);
  Draw2D.ClearImage(255,255,255);
```

```
  Draw2D.Scale(x1,x2,y1,y2);

  if chBoxAxis.Checked then
                       begin
                         Draw2D.XScale(x1,x2,0,255,0,0);
                         Draw2D.YScale(y1,y2,0,255,0,0);
                       end;
  hx:=(x2-x1)/numberOfSegments;
  i:=0;
  x:=x1;
  repeat
    Q[i,1]:=x;
    Q[i,2]:=f(x);
    x:=x+hx;
    i:=succ(i);
  until x>x2+hx;
  Draw2D.PolyLine(Q,i,0,0,255);
end;
```

5 Křivky zadané parametricky a polárně



Parametrické rovnice křivky v rovině jsou obecně tvaru $x = \phi(t), y = \psi(t)$, polární rovnice pak $\rho = f(\phi)$. Konstrukce těchto křivek je velmi podobná, ukážeme si ji na konkrétním příkladu:

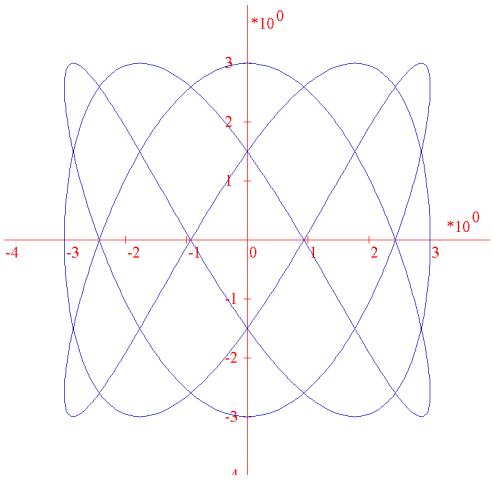
Příklad: Sestrojte křivku určenou obecně parametrickými rovnicemi $x = a \sin \omega_1 t;\ y = b \cos \omega_2 t$ (jedná se o tzv. Lissajousovy křivky, které opisuje kyvadlo rozkmitané ve dvou navzájem kolmých rovinách a amplitudami a, b a úhlovými frekvencemi ω_1, ω_2). Náš úkol řeší procedura `TForm1.btnDrawClick`, která reaguje na stisk tlačítka *Draw*. Nejdříve však upravíme náš hlavní formulář tak, abychom mohli zadávat parametry t_1, t_2. Zdrojový kód pak vypadá následovně:

```
procedure TForm1.btnDrawClick(Sender: TObject);
var t,ht              :Double;
    x1,x2,y1,y2,t1,t2:Double;
    numberOfSegments :Integer;
    code,i           :Integer;
    Q                :TArrayOfPoints;

procedure Lissajous(t:double;var x,y:double);
begin
  x:=3*sin(3*t);
  y:=3*cos(5*t);
end;

begin
  val(edtX1.Text,x1,code);
  val(edtX2.Text,x2,code);
  val(edtY1.Text,y1,code);
  val(edtY2.Text,y2,code);
  val(edtT1.Text,t1,code);
```

```
  val(edtT2.Text,t2,code);
  val(edtNumberOfSegments.Text,numberOfSegments,code);

  Draw2D.Visible := True;
  Draw2D.InitImage(500,500);
  Draw2D.ClearImage(255,255,255);
  Draw2D.Scale(x1,x2,y1,y2);

  if chBoxAxis.Checked then
                       begin
                         Draw2D.XScale(x1,x2,0,255,0,0);
                         Draw2D.YScale(y1,y2,0,255,0,0);
                       end;
  ht:=(t2-t1)/NumberOfSegments;
  i:=0;
  t:=t1;
  repeat
    Lissajous(t,Q[i,1],Q[i,2]);
    t:=t+ht;
    i:=succ(i);
  until t>t2+ht;
  Draw2D.PolyLine(Q,i,0,0,255);
end;
```

Nejprve jsou deklarovány konstanty `t1, t2`, které definují rozsah parametru. Místo proměnných `x, hx` je deklarován parametr `t` a jeho krok `ht`. Na deklaraci parametrických rovnic ve zdrojovém kódu musíme místo funkce použít proceduru `Lissajous(t:double;var x,y:double);`, neboť jako výstup potřebujeme dvě proměnné `x, y`.

Výstupní parametry od vstupních je třeba v hlavičce procedury oddělit klíčovým slovem `var`. Následují parametrické rovnice (s konkrétními volbami za a, b, ω_1, ω_2). Ve vlastní proceduře jsme se omezil opět výhradně na konstrukci křivky. Nejdříve je potřeba nastavit krok parametru `ht` a jeho počáteční hodnotu. V cyklu pak naplňujeme pole `Q` parametrickými rovnicemi, dokud parametr `t` krokem `ht` neproběhne celý interval $< t_1; t_2 >$ (indexem `i` počítáme body v posloupnosti `Q`). Nakonec křivku vykreslíme opět pomocí `PolyLine`. Výsledné zobrazení pro $t \in < 0; 2\pi >$ vypadá následovně:

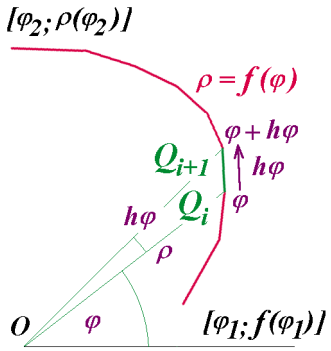


6 Samostatný úkol

Vykreslete spirálu pomocí polárních souřadnic.

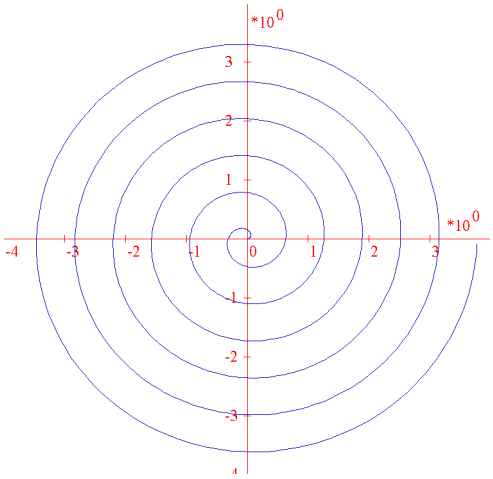
Nápověda: Postupujte podobně jako v parametickém vyjádření, pouze upravíme proceduru pro výpočet bodů následovně:

```
procedure Spiral(phi:real;var x,y:real);
var rho:Real;
begin
```




```
  rho:=0.1*phi;
  x:=rho*cos(phi);
  y:=rho*sin(phii);
end;
```

Výsledná spirála by měla vypadat takto:



7 Křivky zadané rovnicí $f(x, y) = 0$

Nejjednodušší algoritmus pro vykreslování takovýchto křivek je takový, že procházíme jednotlivé pixely na kreslicím plátně a pokud tento bod splňuje rovnici $f(x, y) = 0$, tak tento pixel obarvíme barvou. Zdrojový kód pro vykreslení kružnice $x^2 + y^2 - 16 = 0$ vypadá následovně:

```
procedure TForm1.btnDrawClick(Sender: TObject);
var hx,hy,x,y         :Double;
    x1,x2,y1,y2       :Double;
    code              :Integer;
    A                 :TPoint;

function f(x,y:double) :Double;
begin
  f := x*x+y*y-16;
end;

begin
  val(edtX1.Text,x1,code);
  val(edtX2.Text,x2,code);
  val(edtY1.Text,y1,code);
  val(edtY2.Text,y2,code);

  Draw2D.Visible := True;
```

```
  Draw2D.InitImage(500,500);
  Draw2D.ClearImage(255,255,255);
  Draw2D.Scale(x1,x2,y1,y2);

  if chBoxAxis.Checked then
                       begin
                         Draw2D.XScale(x1,x2,0,255,0,0);
                         Draw2D.YScale(y1,y2,0,255,0,0);
                       end;
  hx:=(x2-x1)/Draw2D.Image1.Width;
  hy:=(y2-y1)/Draw2D.Image1.Height;
  x:=x1;
  repeat
    y:=y1;
    repeat
      if (f(x,y)*f(x,y+hy)<0) or (f(x,y+hy)*f(x+hx,y+hy)<0)
        then begin
               A[1]:=x;
               A[2]:=y;
               Draw2D.PutPoint(A,0,0,255);
             end;
      y:=y+hy;
    until y>y2;
    x:=x+hx;
  until x>x2;
end;
```

Elementy `hx, hy` nám určují velikost pixelu a pomocí dvou cyklů `repeat` procházíme celý obraz. Při splnění podmínky `(f(x,y)*f(x,y+hy)<0) or (f(x,y+hy)*f(x+hx,y+hy)<0)`, tedy že někde v tomto pixelu platí $f(x, y) = 0$, vykreslíme příslušný bod.