CPutils unit

Utilities for C++Builder

Overview

CPutils unit includes more than 60 procedures and functions that can be used from the C++Builder code. The actual platform is Windows 10 and RAD Studio 10.2, but most of the procedures can be used also with other versions of C++ and Windows. *CPutils* contains tools that may help you to handle screen-related code, creation of custom menus, confirmation and information boxes, 2d and 3d function plots, system of linear equations, matrix operations, other useful mathematic items, binary sorting, spline functions and more.

You obtain the unit in the form of .pas file, more than 1200 lines. You can use or edit the code as you wish, including the whole unit or a part of it to your source code.

To use *CPutils*, put the "dutils" item inside your uses clause, and make the unit visible from your code (paste it into the same directory or update the corresponding search path in Project->Options->Directories.

It is supposed that you use CPutils within a C++ form application project. Suppose also that the main form of your project is Form1.

CPutils requires the following events of Form1 to be defined (in Form->Events):

```
void __fastcall TForm1::FormClick(TObject *Sender)
{
Form1->Tag=1;  but=1;
}

void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button,
TShiftState Shift,
      int X, int Y)
{
if(Button==mbLeft)but=1;
if(Button==mbRight)but=2;
}

void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button,
TShiftState Shift,
      int X, int Y)
{
but=0;
}

void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X,
int Y)
{
Application->ProcessMessages();
mszx=X; mszy=Y;
}

void __fastcall TForm1::FormKeyPress(TObject *Sender, System::WideChar &Key)
{
xf0=Key;
}
//---------------------------------------------------------------------------
```

To initialize CPutils, insert the call void initprg(); , or add the following code into your main function (when your program starts).

```
frm=form1; frm->Tag=0;
cnv=frm->Canvas; frm->Font->Name="System"; apms();
Maxx=frm->ClientWidth;
Maxy=frm->ClientHeight;
bity=new Graphics::TBitmap(); //Working image space for zapscry
bity->Height=Maxy+1; bity->Width=Maxx+1;
bitma=new Graphics::TBitmap(); //Working image space
bitma->Height=Maxy+1; bitma->Width=Maxx+1;  cnvb=bitma->Canvas;
bitmazap=new Graphics::TBitmap(); //Working image space for zapscr
bitmazap->Height=Maxy+1; bitmazap->Width=Maxx+1;
brandomize();
loutyy=1; gdziey=1; xf0=0;
```

In the foolowing sections you can find specifications of the functions provided in the unit *CPutils.cpp*

Screen-related tools

Form dimensions - getfdim

void getfdim(TForm Form);
Defines the dimensions of the Form, stores the form client width and height in global predefined variables Maxx and Maxy, respectively.

Save screen image - zapscr

void zapscr();
Stores the screen image in bitmap *bitmazap*.

Retrieve screen image - czytscr

void czytscr();
Reads the screen image stored by zapscr;

Save screen image - zapscry

void zapscry();
Stores the screen image in bitmap *bity*.

Retrieve screen image - czytscry

void czytscry();
Reads the screen image stored by zapscry;

Clear a rectangle with color - czysrec

void czysrec(TCanvas *x,int x0, int y0, int x1, int y1,
byte a, byte b, byte c)
Clears the rectangle (x0,y0,x1,y1) with color rgb a,b,c. a,b, anc must be between 0 and 255.

Store rectangular image - zaprect

void zaprect(TForm *frm, int a, int b, int c, int d);
Stores a rectangular region image in the file obr.dat.
Top left vertice at (a,b), right bottom in (c,d);

Retrieve rectangular image - czytrect

void czytrect(TForm *frm, int a, int b, int c, int d);
Reads rectangle image stored by zaprect. The upper left corner of the image is located at (a,b).

Move cursor - movcur

void movcur(int x, int y);
Moves the cursor to (x,y);

Clear screen with color - czys

void czys(TCanvas *cnv,int r,int g,int b);
Fills the canvas x with color rgb:a,b,c.

Set pen color - sc

sc(TCanvas *cnv,byte c1,int c2,int c3);
Set the pen color (r,g,b) for canvas x.

Set brush color - filc
void filc(TCanvas *cnv,byte c1,int c2,int c3);
Set the brush color (r,g,b) for canvas x.

Display a text - out1
void out1(TCanvas *cnv,int a, int b,TColor col,AnsiString x);
Displays the string t on the canvas x, at (a,b) with text color cl.

Display consecutive text line - outlne
void outlne(TCanvas *cnv,TColor col, AnsiString y);
Each call to outlne displays consecutive text line on canvas.
The line number grows automatically, store in the global variable loutyy
If the first char "|" reight allign
if the first char "%" does not wait at the end of the page

Wait for mouse - zaczek
void zaczek();
Waits for a mouse click.

Display a message - pisak
void pisak(TCanvas *cnv,int a, int b,TColor cl, TColor clt,
AnsiString t1, AnsiString t2,AnsiString t3,AnsiString t4,AnsiString t5);
Displays up to five lines of text t1,t2,t3,t4,t5 on the canvas x, The background color is given by rgb c1,c2,c3.

Draw a line - line
void line(TCanvas *cnv,int a, int b, int c, int d);
Draws a line on canvas x using actual pen color.

Draws a transparent rectangle - czworob
void czworob(TCanvas *cnv,int a,int b,int c,int d,TColor col);
Draws a rectangle on the canvas xc, using actual pen color, does not fill the rectangle.

3D poin-to-screen coordinates - ekranf
void ekranf(float x, float y, float z, float dlx, float al, float be,
float *xs, float *ys, float *odl);
Returns in xs,ys the 2d screen coordinates (in pixels) of the 3d point (x,y,z).
dlx - distance to eye
al,bet - view angles
It is supposed that the point is with a cube, where x,y,z belong to [-1,1]
Uses the auxillianction *xyzz.*

Clearing a rectangle - czarny, bialy
void czarny(TCanvas *cnv, int a, int b, int c, int d);
void bialy(TCanvas *cnv, int a, int b, int c, int d);
czarny fills the rectangle (a,b,c,d) with black. *bialy* fills it with white. cnv is the actual canvas.

Display text - out10
void out10(TCanvas *cnv,int a, int b,TColor col,AnsiString x);

Displays text x on canvas cnv using color col. If the color is bright, the text is printed on black background, otherwis the text if printed with the white background.

Drawing figures with mouse - rysreg

void rysreg(TCanvas *cnv, int shap, int *r,int* x1, int*y1, int* x2, int*y2);
cnv - actual canvas
if shap=0 creates a circle, returns the readius in in r, center in (x1,x2) in pixels
if shap=1 creates a rectangle, returns in (x1,y1),(x2,y2) the vertices in pixels
Drag the shape with the mouse (left button down). When the button is released, you are asked if the figure is OK. If not, the procedure is repeated.

Binary sorting

CPutils includes tools for fast sorting, using a binary tree. Sorting 10000 real numbers takes only 10 milliseconds, and sorting one million elements needs about two seconds.

Creating binary tree - initsort

void initsort();

This creates a binary tree that can be used to sort records. The key to sort is a real number. The sorted records are defined by the following.

```
struct xdt00 // data to be sorted
// idat - index
// t - key
{float t; int idat;};

struct tx00levt //data element with pointers for sorting
// stored in a binary tree structure
// defined by the pointers pre (predecessor) and suca,sucb (sucessors)
// dt is the pointer to event data
{xdt00 *dt; tx00levt *suca,*sucb,*pre;};
```

variable *t* is the key. The records are sorted in ascending order. See procedure *sortprob* below for an example of application.

Insert an element to the tree - addtosorttree

void addtosortree(float t, int idat);
Each call to this function inserts a new element to the tree. For example, use the following code to insert an element with the key *t* being a random number

```
for(k=0;k<n;k++){
  x=Frandom();
  addtosortree(x,k);}
```

Retrieving the consecutive minimal elements - getelem

xdt00 *getelem();

A call to getelem returns a record with consecutive minimal value of *t*. See procedure *sortprob* below for an example of application.

Sorting example - sortprob

```
void sortprob(){
int k,n; float x; xdt00 *y;
```

```
n=500; czys(cnv,0,0,0);
initsort();
for(k=0;k<n;k++){
  x=Frandom(); // numbers to be sorted
  addtosortree(x,k);
}
for(k=0;k<n;k++){ // retrieving and displaying the sorted elements
  y=getelem(); Application->ProcessMessages();
  if(xf0==27)exit(0);
  outlne(cnv,clWhite,FloatToStr(y->t)+" "+IntToStr(k)+" "+IntToStr(y->idat));
}
zaczek();
}
```

This function inserts to the tree n=500 elements with random values of the key, and then displays the elements, sorted with ascending order.

Clear the sort tree - cleartree

void cleartree();
This clears the existing sort tree.

Menus and dialogs

The use of CPutils menus is different from the C++Builder main menu of the form. The CPutils menus are created dynamically by calls from the user code. The menu returns the number of the selected option. Then the user can code anything he wants directly in the same code segment using the *case* or *if* instructions.

Simple menu - menu01

void menu01(TForm *frm, bool vert, int a, int b, AnsiString *nag, TColor clo,
  bool disab[],AnsiString t[], int *opt);
This creates a simple menu.
frm - actual form
vert - vaertical menu if true, otherwise horizontal menu at the top of the form
if vert=true then a,b defines the position of the menu on the screen. If vart=false then a and b are not used.
nag - the caption for the vertical menu
disab -if disab[n] is true, the n-th menu option becomes disabled
t - array of strings with names of consecutive options
opt - the result=the number of the selected option
Note: t is an array of 50 strings. However, the number of strings (options) is limited by the form dimensions. If you define *n* strings 0...n-1**, then the (n+1)th string[n] must be empty.**
See the section "Some examples of application codes" for an example of application.

Popup menu - menu02

void menu02(TForm *frm, TColor clo, TColor cl2, bool disab[],AnsiString t[],
Tt2 t2[], int *opt, int *opt2);

Displays the main horizontal menu at the top of the form. The captions of the main menu are in the string array t, first one is t[1]. The options for popup menu numebr n are in the array t2[n], for example T2[1][1]="First one"; T2[1][2]="Second"; T2[1][3]="Last"; T2[1][4]="";
The caption of the first option in number one. For a submenu n of m options, the caption t2[n][m+1] must be empty (of zero length).
If the option n of the main menu should not have a popup, then set t2[n][1]=""; See the section **Some examples of application codes** for and example of a call to *menu02*.
The result is returned in opt (main menu option) and opt2 (selected submenu option).
See the section "Some examples of application codes" for an example of application.

Confirmation dialog - confirm

bool confirm(TForm *frm, int a, int b, AnsiString nag);
Displays confirmation box with options Yes and No. Waits untol an option is selected.
Returns true of false for Yes and No, respectively.
frm - parent form
a,b - pozycja
nag - caption
Note: uses zapscry() and czytscry() with bitmap "bity"

Information box - barinfo, barinfo2

void barinfo(AnsiString t); void barinfo2(AnsiString t);
Displays the string *t* at the top of the actual form.
Use barinfo for black text on white background, barinfo2 for white text on black background.

Read several items - czyt50

void czyt50(TForm *frm,int ile,int x, int y, AnsiString t[], AnsiString v[], AnsiString *nag);
reads up to 50 strings from a dialog box
frm - actual form
ile - number of items to read
x,y - position of the box
nag - caption of the dialog
t - array of strings, captions of consecutive data items
v - the result: array of item read

This can be used to enter up to 50 strings. *frm* is the actual form, ile is the number of items to enter, (x,y) is the position of the upper left corner of the main box, t is the array of item captions (what to enter), and v is the array of strings with the results.
See the section "Some examples of application codes" for an example of application.

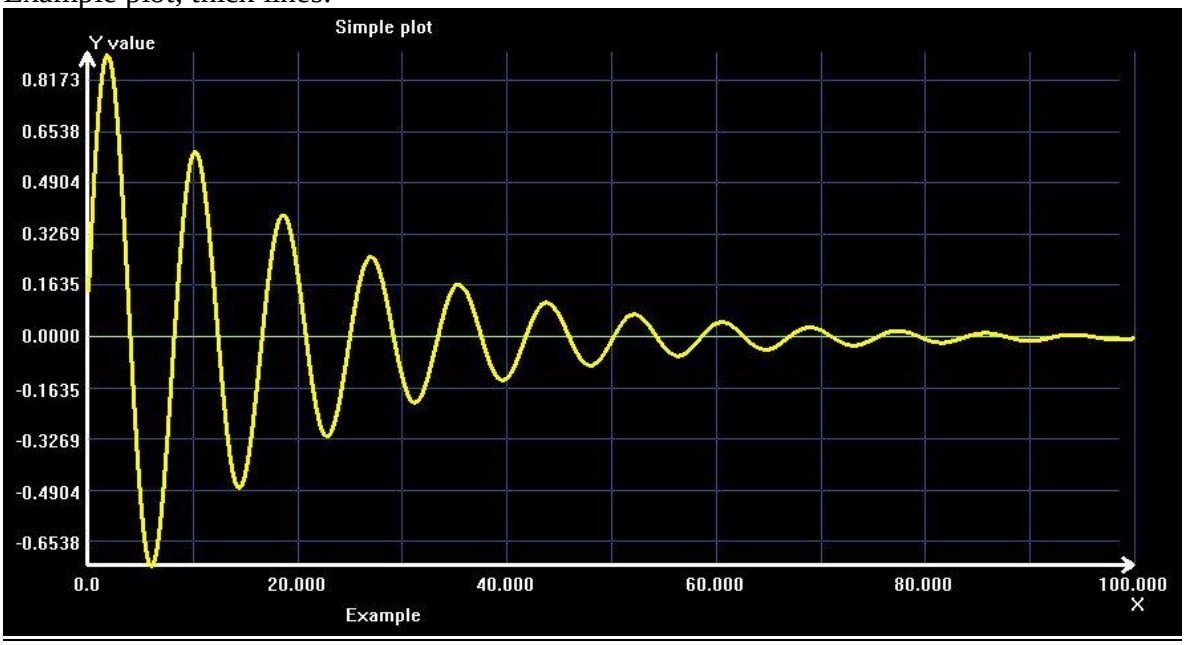
Plots and other graphics

Function plot - traz

void traz(TCanvas *cnv, ttrzc x,int a, int b, int c, int d, int ile,
  float max, float min, float lowx, float highx,

AnsiString title, AnsiString xaxis, AnsiString yaxis,
bool jak, bool clear, bool krata, bool thick, bool czar, int znak)
Simple plot on canvas cnv.
x - plot data, (0 - (ile-1)
(a,b) - left upper corner, (c,d) - right lower
ile - how many points in x (first point in x[0])
max,min - max and min value of x, used if jak=2
lowx, lowy - min and max value for indepenent variable (X-axis)
title - title
xaxis - X axis caption
yaxis - Y axis caption
jak=true normalizes automatically
clear=true clears the area
clear=false does not clear the area, may be used for multiple plots
if clear=0 you must clear the area before
krata=true shows x-x grid
thick=true the curve with thick lines
czar=true whit lines on black background
znak - number displayed on the curve, if negative, no number(s)

Example plot, thick lines:



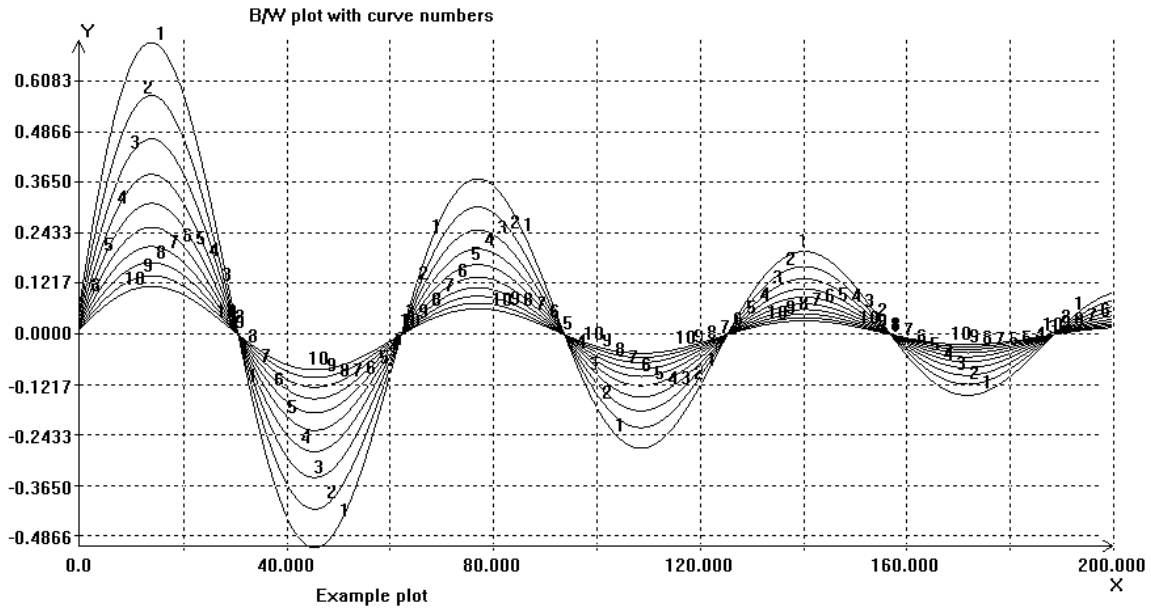
Multiple plot - multiplo

void multiplo(TCanvas *cnv,ttrzmult x,int a, int b, int c, int d, int ile, int ilep,
float lowx, float highx, AnsiString title, AnsiString xaxis, Tt2 te,
bool krata, bool thick, bool czar)

Displays the plot of multiple functions stored in array x of type ttrzmult

typedef float ttrzc[1000];
typedef ttrzc ttrzmult[50];
For example, x[3][200] is a point 200 of function number 3 (index starts at zero).
ile - how many functions (up to 50), ilep - number of data point for each function (up to 1000). Other parameters as in function *traz*.

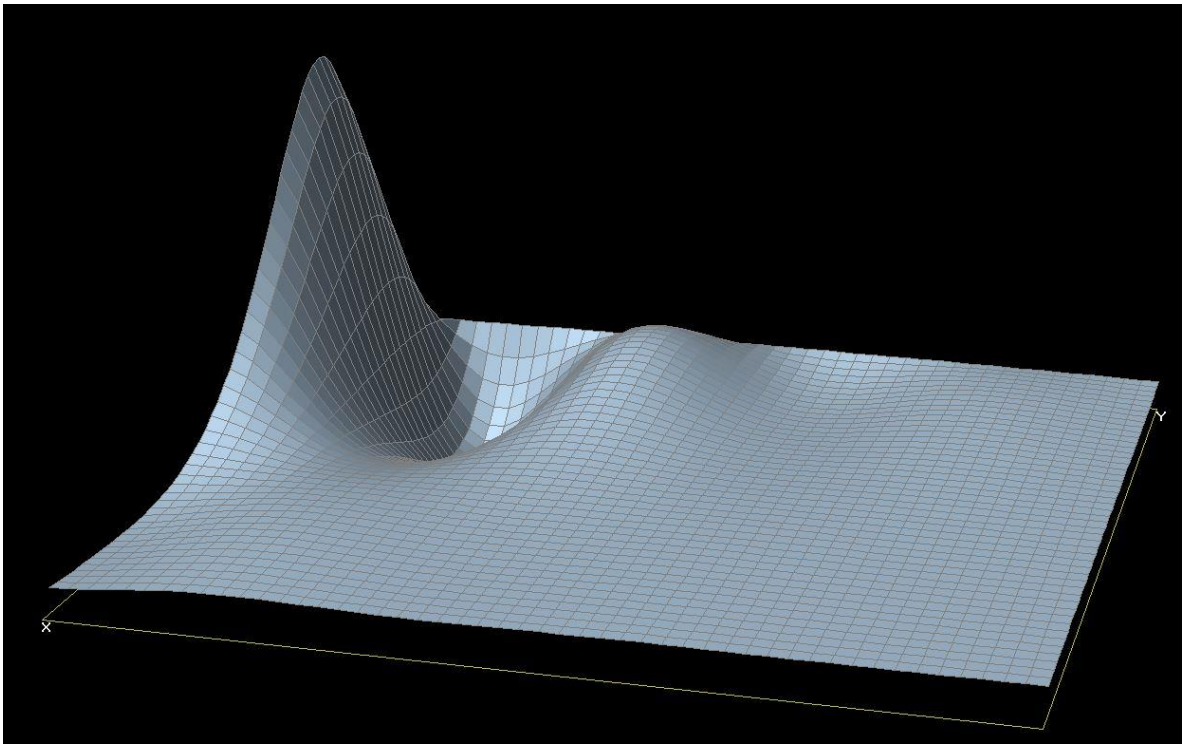
Example:



procedure powplb(xc:tcanvas;n,m:integer;alf,bet:real;var x:mttp;jak,
tak:integer;osx,osy,osz:string; bc1,bc2,bc3:byte);

3D plot of a function of two variables - plot3d

void plot3d(TCanvas *cnv,int n,int m,float dlx, float al, float be,
float wsp, float wspx, int xshift, int yshift, float h,float x[], bool colo,
AnsiString axis1, AnsiString axis2);
3D plot on canvas cnv, changes view angle with mouse. Uses *plot3dim*.
n,m - dimensions of plotted area, (number of data points)
Values to plot in x[k*m+j], j,k - indexes of a point on the area
j - from left to right, k - toward viewer
cnv - canvas to plot, n,m - matrix dimension (k,j)
dlx - distance from the viewer, wsp - horizontal size (norm. 1),
al, be - view angles in radians, for example 0.1, 0.4
wspx - vertical size (normal 1)
xshift, yshift - shift horizontal and vertical on the screen
h - x offet of original data, applied after normalization, normally zero
colo=true
To change the view angles, cklick on the plot and then drag the displayed area to the desired position.
See the section "Some examples of application codes" for an example of application.

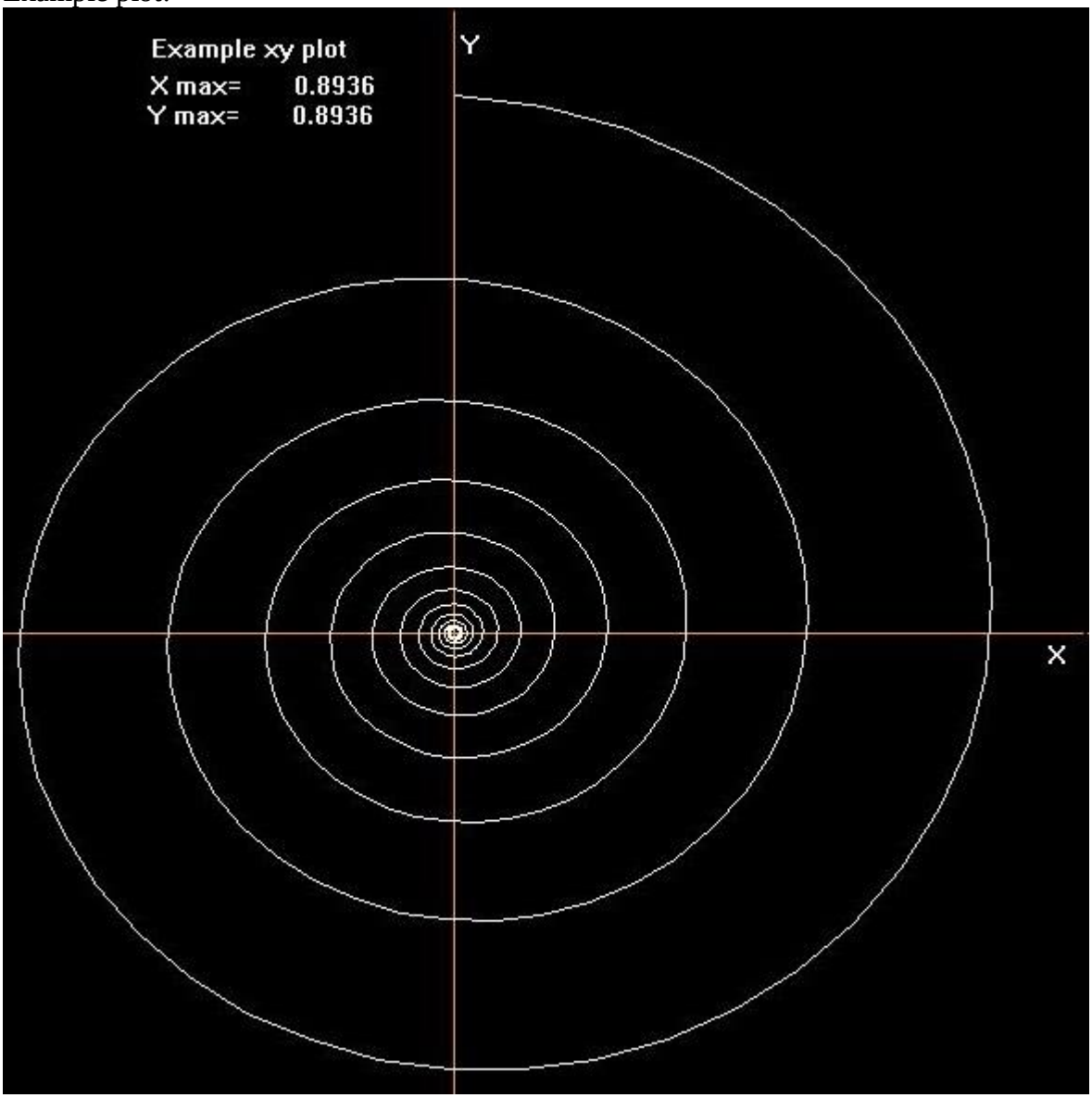


Example: *plot3d*

XY plot - xyplot

void xyplot(TCanvas *cnv, int k, ttrzc x, ttrzc y, float wsp, float xm,
float ym, AnsiString hor, AnsiString ver, AnsiString tit, int ofx, int ofy,
bool col, bool norm);
cnv - canvas to plot
k - number of points, takes abs(k)
x,y - data to plot (values of two functions)
xm,ym - scale (max) if norm=false
hor - horizontal axis caption, ver - vertical, tit - title of the plot
ofx,ofy - screen offset x,y in pixels
norm=true to normalize
col=true for black background
norm=true to normalize automatically, otherwise uses xm and ym

Example plot:



Some useful math functions

Linear equations - GausRow

void GausRow(int n, float *x, float *pra, float *ax, bool *err);
Resolves a set of n linear equations
x - unknown
n = number of equations
ax - matrix ax(k,j) -> ax[(k*N+j], k - wiersz, j - kolumna
pra - array of right-hand sides
err = true if error

Note that the coefficients of the equations must be stored in one-dimensional array ax, row by row. For example, if your left-hand side matrix ix y[k][j] then you shoud store it as follows (k - row, j- column):
ax[k*n+j]=x[k][j];
Note: GausRow changes the variable (array) pra.
See the section "Some examples of application codes" for an example of application.

Disserential equations - RHF5

void RKF5(int n, float h, float t, float x[], float *err);
This is the Rnge-Kutta-Fehlberg 5-th order integration procedure for ordinary differential equations. The equation system must have the following form:
$dx_1/dt=f_1(x,t)$
$dx_2/dt=f_2(x,t)$
....
....
$dx_n/dt=f_n(x,t)$
where $x=(x_1,x_2,...,x_n)$ is the model state vector, $f_1,f_2,...,f_n$ are the righ-hand side functions and t is an independent variable.
Parameters:
n - number of equations
h - integration step
t - independent variable
err - estimated error
The user must provide the procedure deriv that calculates the functions $f_1,f_2,...,f_n$, as follows (example):

```
void deriv(int n, float de[], float x[], float t){ //example
float a,b,c,d;
a=0.2; b=0.008; c=0.4; d=0.002;
de[0]=a*x[0]-b*x[0]*x[1];
de[1]=-c*x[1]+d*x[0]*x[1];
//outlne(cnv,clGreen,FloatToStr(de[0])+"  "+FloatToStr(de[1]));
}
```

The function *deriv* is declared in *CPutil.h*, but not defined in *CPutil.cpp*. The user must provide the code for *deriv*, due to his/her model equations. Normally, *RKF5* is called repeatedly to calculate the consecutive model states, hundreds or thousands times over a given time interval.
Note: inside *RKF5* there is the declaration float **d,*y,*k1,*k2,*k3,*k4,*k5,*k6*; and the corresponding calls to *malloc* function to get the necessary memory for these variables. At the end of *RKF5* the memory reserved for these variables is freed. To avoid these memory operations at each integration step and to get faster execution, the user may remove the declaration and memory reservations from *RKF5* and declare the variables as global. In this case the memory allocation should be done only once, before starting the executions of *RKF5*, and the memory must be freed after calculating the whole trajectory (only once).
See the section "Some examples of application codes" for an example of application.

Point-to-line distance - dpunktpros

void dpunktpros(float a, float b, float c,
float x1, float y1, float z1,float x2, float y2, float z2,float *d, float p[]);

Calculates the distance between 3d point (a,b,c) and the line defined by two points (x1,y1,z1) and (x2,y2,z2).
The distance is returned in* d, the corresponding point on the line in (p[0],p[1],p[2])

Point-to-section distance - crosslin

void crosslin(float x, float xk, float y, float yk,
float a, float ak, float b, float bk, float *xp, float *yp, bool *p){
Calculates the point of intersection of two lines
First line given by (x,xk), second by (y,yk)
Result: (xp,yp) - point of lines intersection
returns p true if sections intersect

Point-to-plane distance - dpunktpow

float dpunktpow(float a,float b, float c, float d, float x, float y, float z)
Returns distance d from the plane with equation ax+by+cz+d=0
to the point (x,y,z)

Determinant 3x3 - deter

float deter(tmatrx a)
Returns determinant of 3x3 matrix

Matrix product - matrprod

void matrprod(tmatrx a, tmatrx b, tmatrx y);
Calculates 3x3 matrix:tmatrix that is the result of multiplying matrix a by matrix b.
Result iin matrix y.

Matrix-vector product - matrvec

void matrvec(tmatrx a, b[], y[]);
Calculates the product of matrix a and vector b[3]. Result in y[] (0,1,2).

Equation of the plane - rowpow

void rowpow(float x1,float y1,float z1,
float x2,float y2,float z2,
float x3, float y3, float z3,
float *aa, float *bb, float *cc, float *dd);
In aa,bb,cc and dd returns the coefficient of the plane equation
aa*s+bb*y*cc*z+dd=0
The plane is defined by three ponts (x1,y,1z1), (x2,y2,z2) and (x3,y3,z3).
Note: the three points cannot be located at the same line.

Weibull distribution - Weibull

float Weibull(float lambda,float k);

Each call to Weibull returns a consecutive value of the random variable with Weibull distribution. *lambda* is the scale parameter and *k* is the shape parameter.

Lognormal distribution - Lognormal

float Lognormal(float mu, float sigm);
Returns the value of the random variable with lognormal distribution. *mu* and *sigm* are the mean and standard deviation of the variable's natural logarithm,

Uniform distribution - Frandom, Rnd

float Frandom();float Rnd(float s, float x);
Frandom eturns the value of the random variable with uniform distribution within the interval (0,1). Rnd generates the value within (s-d,s+d)-

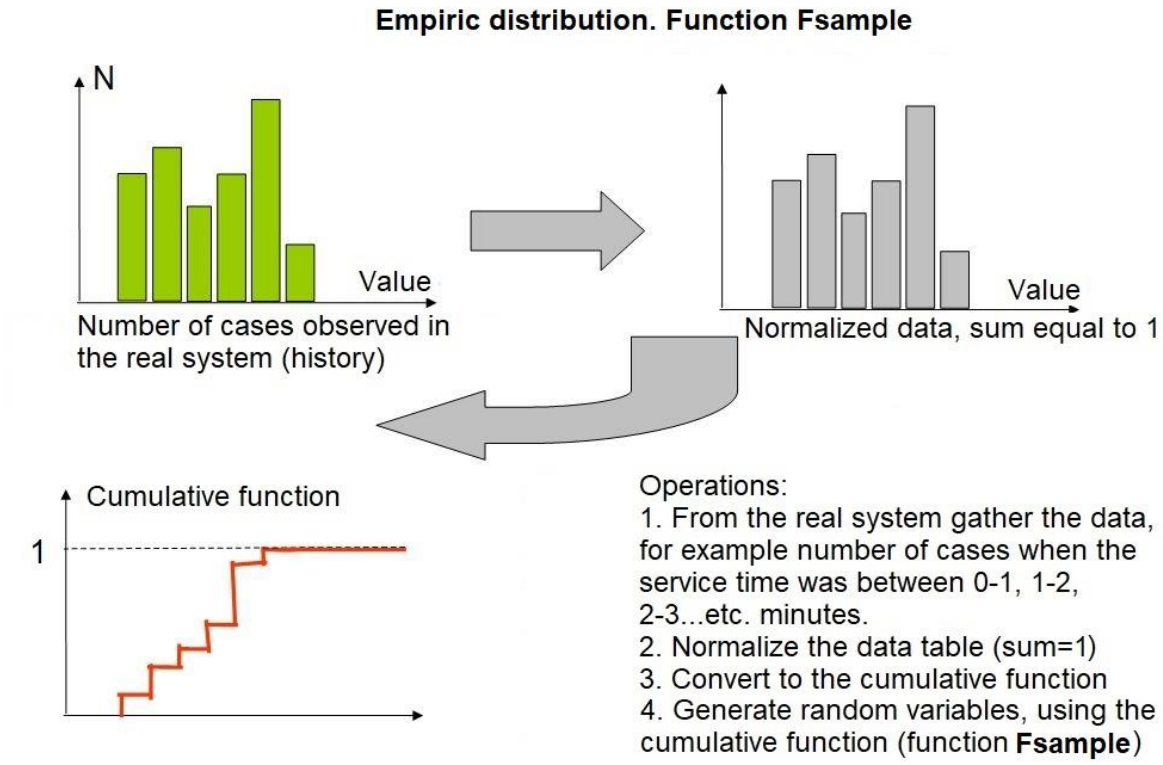
Normal distribution - Normal

float Normal(float s, float d);
Returns the value of the random variable with normal distribution. *s* is the expected value and *d* is the standard deviation.

Sample (empirical) distribution - Fsample

float Fsample(int n, float low, float hig, float his[]);
function sample(m:integer; h:hist):real;
n - number of elements in array his
low, hig - the range: min and max values to generate
his - cumulative distribution function

The F*sample* function makes it possible to introduce a user distribution in the form of a histogram and then to generate random numbers according to this distribution
Fsample returns a real random value generated due to the distribution defined by the programmer and stored in an array *float his[].* Array *his* should represent an approximation of the cumulative distribution function, being non-decreasing and defined betwen 0 and 1. For exmple, if you gather from the real system a table that contains the number of clients served between 0 and 1 minute stored in *x[0],* during 1-2 minutes stored in *x[1],* 2-3 minutes etc., then first normalize the array x so that the sum of all its values be equal to one. Then, construct his so, that *his[n+1]=x[n+1]-x[n].* The obtained arra y his can be use as an argument of Fsample.



Triangular distribution - Triandis

float Triandis(float a, float b, float c);
Generates a random value with triangular distribution within a<b<c.

Erlang distribution - Erlang

float Erlang(int n, float s);
Generates a random value with Erlangr distribution.
n - function order
s - expected value

Exponential distribution - Negexp

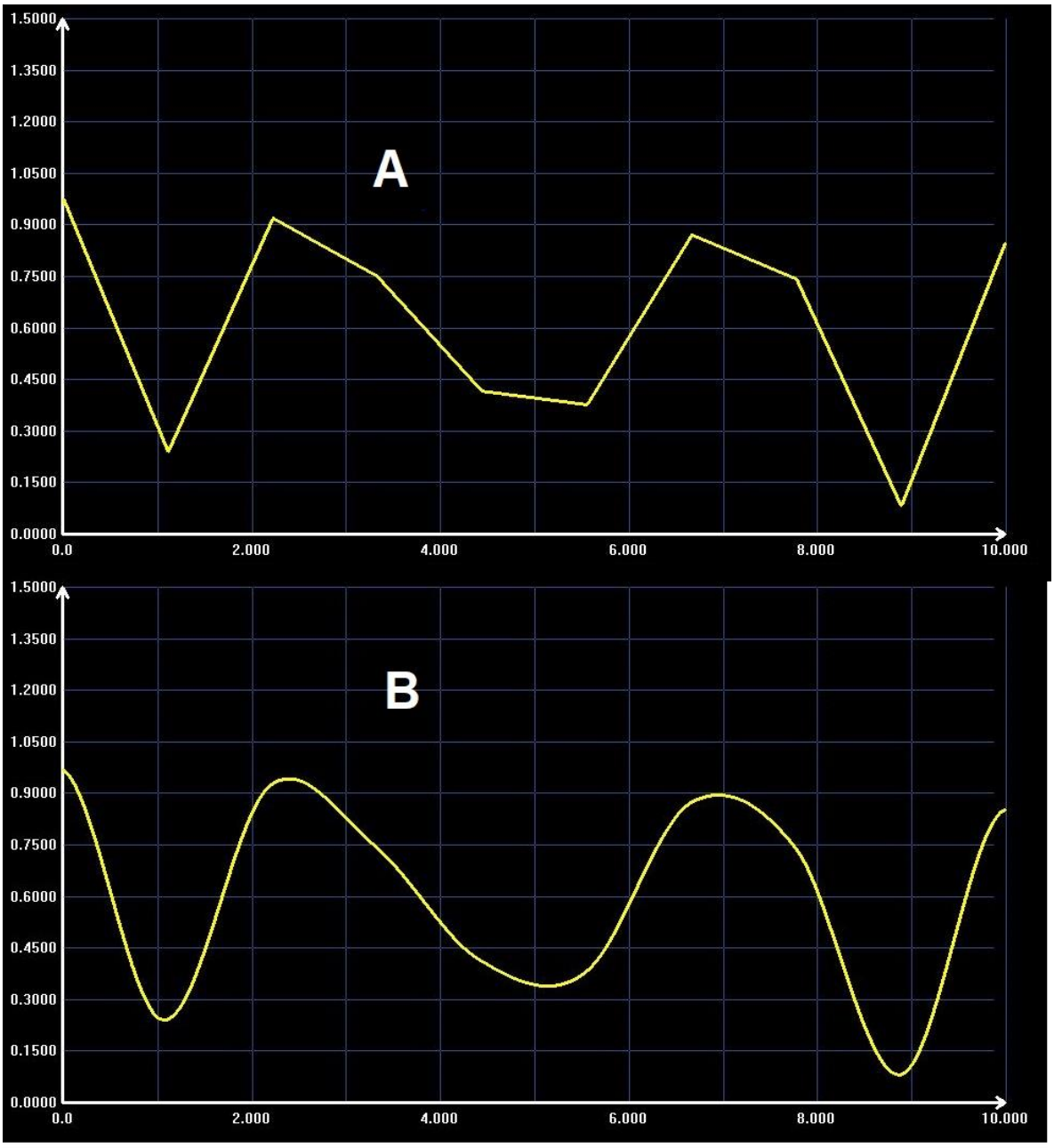
float Negexp(float s);
Generates a random value with negative-exponential distribution.
s - expected value
Useful for between-arrival intervals for the Poisson arrival process.

Poisson distribution - Poisson

int Poisson(float s,float x);
Returns the integer value with the Poisson distribution. n is the parameter of the distribution.
x is the expected number of occurrences.
Note: Do not use this distribution for inter-arrival time while simulating the Poisson arrivals process. This is a serious mistake. For the inter-arrival time use the exponential distribution.

Spline functions - splinef

void splinef(int k, int j, ttrzc x, tabspline y);
void resplin(int *k, ttrzc z);
splinef generates spline functions for a given data represented by an array x.
Creates spline function s(x)=axxx+bxx+cx+d
k - number of original points, j - number of new interpolated points in each interval. Must be j*k<=5000
Uses procedure *resplin*. The result is returned in the array y (smoothed data). See the section "Some example application codes" for an example of application.
For example, if you have original data that cover the interval 0 to 10 with step equal to 1.0, you have 11 original points 0,1,2...10. In this case k=11. If the number of interpolated points within each interval is equal to 20, then the *splinef* procedure will create (k-1)*20=200 points of the smoothed curve.
Limitation: 6<k<50.
See the section "Some examples of application codes" for an example of application.



A - original data B - smoothed with spline functions

Miscellaneous

Initiating the program for CPutils - initprg

void initprg(TForm *form);

Use this to initiate some variables needed by CPutils. form is the actual form where your program starts. Put it at the beginning of your main program function, or before you start using CPutils codes. The following instruction are executed by initprg:

```
frm=form; frm->Tag=0;
cnv=frm->Canvas; frm->Font->Name="System"; apms();
```

```
Maxx=frm->ClientWidth;
Maxy=frm->ClientHeight;
bity=new Graphics::TBitmap(); //Working image space for zapscry
bity->Height=Maxy+1; bity->Width=Maxx+1;
bitma=new Graphics::TBitmap(); //Working image space
bitma->Height=Maxy+1; bitma->Width=Maxx+1;  cnvb=bitma->Canvas;
bitmazap=new Graphics::TBitmap(); //Working image space for zapscr
bitmazap->Height=Maxy+1; bitmazap->Width=Maxx+1;
brandomize();
loutyy=1; gdziey=1; xf0=0;
```

Computer user name - cpname

AnsiString cpname();
Returns the name.

Computer id - cpnumb

AnsiString cpnumb();
Returns the computer id, uses system("wmic bios get serialnumber > sn.txt");

Get short file name - shortfil

AnsiString shortfil(AnsiString x, int jak);
void initprg();
Returns short name file, without path and without extension.
If jak=0 then removes both path and extension,
jak=1 removes path, jak=2 removes extension
For example,
outlne(cnv,clBlack,shortfil("c:\\algo\\otro\\filek.dat",0));
prints *filek*.

Comparing strings - podobne

bool podobne(AnsiString a, AnsiString b);
Removes spaces from the two strings Returns true if the resulting strings are equal to each other, false otherwise. Is not case sensitive. For example "John Connor" is treated as equal to "johnconnor" and to "JOHN CONNOR"

Processing application messages - apm, apms

void apm();
void apms();
apm executes Application->ProseccMessages(). *aoms* does the same and waits 50 msec.

Randomize - brandomize

void brandomize();
Similar to c++ randomize(). Initiates the rendom number generator *random* and calls it 50 times, to omit eventual initial "warm up".

Some examples of application codes

Menus, dialogs

The code below displays the following:
* simple vertical menu

* simple horizontal menu
* pop-up menu
* confirmation box "Exit program?"

```
void codeexample1(){
// Menus and confirmation box
AnsiString t[50];  Tt2 t2[50]; int k,j;
TPanel *kuku; AnsiString *g=new AnsiString; int opt,opt2;
AnsiString f;  AnsiString v[50]; AnsiString *nag=new AnsiString;
String z;  bool disab[50];
czys(cnv,0,0,200);
barinfo("This is an info bar displayed by 'barinfo'. Click to continue"); zaczek();
barinfo2("This is an info bar displayed by 'barinfo'. Click to continue"); zaczek();
czys(cnv,200,100,0);
*g="Menu example: menu01";
t[0]="Menu title";
t[1]="Option one"; t[2]="Second"; t[3]="Cats and dogs"; t[4]="Birds";
t[5]="Other option"; t[6]="Last one";t[7]="";
for(k=0;k<50;k++)disab[k]=false;
disab[2]=true; // option 2 disabled
menu01(frm,true,100,50,g,clBlue,disab,t,&opt); // simple menu, vertical
out1(cnv,100,50,clWhite,"Selected option: "+IntToStr(opt)+" Click to continue");
zaczek(); czys(cnv,0,100,200);
out1(cnv,100,50,clWhite,"Simple horizontal menu");
menu01(frm,false,100,50,g,clSilver,disab,t,&opt);  // simple menu, horizontal
out1(cnv,100,20,clWhite,"Selected option: "+IntToStr(opt)+" Click to continue");
zaczek(); czys(cnv,0,0,200);
// Filling all sub-menu options with 'xxx '+option numbers
for(k=0;k<50;k++)for(j=0;j<50;j++)
  t2[k][j]=AnsiString("xxx ")+IntToStr(k)+","+IntToStr(j);
for(k=0;k<50;k++)t2[k][10]="";
t2[4][2]="Cancel"; // Option 4 , 2 cancels the sub-menu
t[3]="No pop-up";t2[3][1]="";  // main menu option 3 without popmenu
out1(cnv,100,50,clWhite,"Menu with pop-ups");
menu02(Form1,clAqua,clWhite,disab,t,t2,&opt,&opt2);
czys(cnv,0,0,200);
out1(cnv,100,20,clWhite,"Selected options: "+IntToStr(opt)+"  "+IntToStr(opt2));
zaczek();
if( confirm(frm,Maxx/2,100,"Exit program?")) exit(0);;
}
```

The *codeexample2* function displays an input dialog with 45 items. The user can enter new values to the strings. Press "Done" button to terminate. Then the new values are displayed. Remember that it can read up to 50 strings.

```
void codeexample2(){
// reads 45 user-defined string values
AnsiString t[50];  Tt2 t2[50]; int k,j;
AnsiString v[50];AnsiString nag,z;
nag="Reading strings";
t[0]="First one";t[1]="Somthing"; t[2]="Other";
t[3]="Nice string"; t[4]="A value";
```

```
t[5]="Hello"; t[6]="His name"; t[7]="My cat"; ;
for(k=8;k<45;k++)t[k]="Apple "+IntToStr(k);
for(k=0;k<45;k++){v[k]=IntToStr(k)+" ??"; }
czyt50(Form1,45,100,100,t,v,nag);
czys(cnv,0,0,0);
for(k=0;k<45;k++)outlne(cnv,clWhite,"String "+IntToStr(k)+" is "+v[k]);
zaczek(); apms();
if( confirm(frm,500,100,"Terminate program?")) exit(0);;
zaczek();
}
```

Random number generators

```
void codeexample3(){
// Calls random numbergenerators, displays the resulting desnsity function
// The random number generator is called 1,000,000 times. The generated
// values accumulate in array u, covering 200 value intervals
// between zero and three. Then, the shape stored in u is displayed
// using traz function
int k,j,shape,opt; float u[200]; float x,e;  AnsiString t[50];
AnsiString v,nag; bool disab[50];
nag="Select distribution";
t[1]="Weibull"; t[2]="Normal"; t[3]="Uniform (1-2)"; t[4]="Erlang";
t[5]="Exponential"; t[6]="Triangular";t[7]="Lognormal";
t[8]="Terminate"; t[9]=""; czys(cnv,0,0,150);
for(k=0;k<50;k++)disab[k]=false;
do{
menu01(frm,true,100,50,nag,clBlue,disab,t,&opt); // simple menu, vertical
if(opt==8)return;
for(k=0;k<200;k++)u[k]=0.0;
shape=1.0; e=0.0;
for(k=0;k<1000000;k++){
switch(opt){
  case 1:x=Weibull(shape,5); v="Weibull"; break;
  case 2:x=Normal(1,0.2); v="Normal";  break;
  case 3:x=Frandom()+1; v="Uniform"; break;
  case 4:x=Erlang(3,1); v="Erlang"; break;
  case 5:x=Negexp(1.0); v="Exponential"; break;
  case 6:x=Triandis(1,1.5,3); v="Triangular"; break;
  case 7:x=Lognormal(0,1); v="Lognormal"; break;
  case 8:break;
 }
e+=x;
j=floor(x/3.0*200.0);
if(j<200 && j>=0)
u[j]+=1.0/200.0;
  }
e/=1000000;
traz(cnv,u,100,100,Maxx-100,Maxy-100,200,1.0,0.0, 0.0,3.0,
v+" exp = "+FloatToStr(e),"X","Y",true,true,true,true,true,-1);
zaczek(); }
while(opt!=8);
}
```

Plots

Shows several example plots.

```
void codeexample4(){
// shows examples of various plotting functions
float x[10000]; int k,j; AnsiString u,v; ttrzc g,h; Tt2 te;
ttrzmult y; bool krata,thick;
krata=true; thick=true;
for(k=0;k<800;k++)for(j=0;j<15;j++)y[j][k]=sin(k*0.04)*exp(-
(k*0.2+10*j)*0.03);
for(k=0;k<800;k++)g[k]=sin(k*0.1)*exp(-k*0.03);
traz(cnv,g,100,100,Maxx-100,Maxy-100,800,1.0,0.0, 0.0,8.0,
"Simple plot example","X","Y",true,true,krata,thick,true,1);
zaczek();
traz(cnv,g,100,100,Maxx-100,Maxy-100,800,1.0,0.0, 0.0,8.0,
"Plot example","X","Y",true,true,krata,thick,false,-1);
zaczek();
traz(cnv,g,100,100,Maxx-100,Maxy-100,800,1.0,0.0, 0.0,8.0,
"Plot example","X","Y",true,true,krata,thick,false,1);
zaczek();
for(k=0;k<15;k++)te[k]="next one";
multiplo(cnv,y,100,100,Maxx-100,Maxy-100,800,15,
  0,800,"Multiple plot example","X",te, krata,thick,false);
for(k=0;k<800;k++)
{g[k]=sin(k*0.08)*exp(-k*0.005);h[k]=cos(k*0.08)*exp(-k*0.005);}
xyplot(cnv,800,g,h, 0.7, 1.0, 1.0, "X","Y","XY plot", -300,0,true,true);
zaczek();
u="X"; v="Y";
for(k=0;k<50;k++)for(j=0;j<50;j++)x[k*50+j]=sin(0.1*k)*cos(0.3*j)*exp(-
(k+j)*0.1);
plot3d(cnv,50,50, 3.0, 0.1, 0.4,  1.0,0.5, 0,300, -0.2,x,true,false,u,v);
plot3d(cnv,50,50, 3.0, 0.3, 0.2, 1.0,0.5, 0,300, -0.2,x,true,true,u,v);
plot3d(cnv,50,50, 3.0, 0.2,0.3, 1.0,0.5, 0,300, -0.2,x,false,false,u,v);
if( confirm(frm,500,100,"Terminate program?")) exit(0);
}
```

Spline functions

```
void codeexample5(){
// Spline functions
ttrzc z; tabspline y; int k;  int skol;
skol=20;
for(k=0;k<skol;k++)z[k]=Frandom(); z[0]=0;
czys(cnv,0,0,0);
traz(cnv,z,100,100,Maxx-100,Maxy-100,skol,1.0,0.0, 0.0,10.0,
"Splin example: original data","X","Y",false,true,true,true,true,-1);
zaczek(); czys(cnv,0,0,0);
splinef(skol,20,z,y);
for(k=0;k<(skol-1)*20;k++)z[k]=y[k];
traz(cnv,z,100,100,Maxx-100,Maxy-100,(skol-1)*20,1.0,0.0, 0.0,10.0,
"Splin example: smoothed data","X","Y",false,true,true,true,true,-1);
```

```
zaczek();
if( confirm(frm,500,100,"Terminate program?")) exit(0);
}
```

Differential equations

The code below is an example of application of RKF5. The model is described by the equations of the classic Lotka-Volterra prey-predator model. See function deriv for the model parameters and equations. d[0] and d[1] are the two derivatives, x[0] and x[1] are state variables: x[0] is the amount of prey and x[1] is the amount of predator.
Funtion codeexample7 defines the initial conditions and then calculates the model trajectory over the interval 0-100, with time-step equal to 0.2.
After calculating the trajectory, th results are displayed in the form of time-plots and XY plot.

```
void deriv(int n, float de[], float x[], float t){
// for RKF5
float a,b,c,d;
a=0.2; b=0.008; c=0.4; d=0.002;
de[0]=a*x[0]-b*x[0]*x[1];
de[1]=-c*x[1]+d*x[0]*x[1];
}

void codeexample7(){
// Ordinary differential equations
ttrzmult tm; Tt2 te;  ttrzc xp; ttrzc yp;
int k,j; float h,t,err; float x[2];
x[0]=500.0; x[1]=50.0; h=0.2;  t=0.0;
for(j=0;j<2;j++)tm[j][0]=x[j];
for(k=0;k<499;k++){apm(); if (xf0==27) exit(0);
   RKF5(2,h,t,x,&err);
   for(j=0;j<2;j++)tm[j][k+1]=x[j];
   }
czys(cnv,255,255,255);
te[0]="Prey"; te[1]="Predator";
multiplo(cnv,tm,100,100,Maxx-100,Maxy-100,500,2,0,100.0,"Prey-predator model",
"Time",te,true,true,false);
for(k=0;k<500;k++){xp[k]=tm[0][k]; yp[k]=tm[1][k];}
xyplot(cnv,500,xp,yp,0.8,0.0,0.0, "Prey","Predator","XY (phase) plot",0,0,false,true);
zaczek();
exit(0);
}
```