Лекция 1

Важна ли графика для программистов? Графика важна т.к. сразу можно увидеть результаты своего труда. Она нужна, по крайней мере, как вспомогательное средство для **визуализации** результатов, динамики выполнения алгоритмов. Важность визуализация – это наглядность, высокая скорость и большие объемы восприятия информации.

Некоторые универсальные языки программирования могут иметь стандартную поддержку графики. Например, язык Pascal имеет возможность создавать несложные графические приложения. Для программирования графики на языке Pascal существует модуль Graph, при помощи процедур которого можно рисовать изображения.

```
uses Graph;
var GraphDriver,GraphMode: integer;
begin
  InitGraph(GraphDriver,GraphMode, ‘’);
  <процедуры и функции для рисования>
CloseGraph;
end.
```

Это было удобно при MS DOS, где основным режимом был текстовый. С появлением Windows основным режимом стал графический.

При программировании графических приложения под Windows, используя обработчик события WM_PAINT или OnPaint (последнее - если используется какой-нибудь объектный каркас для Windows-приложения), нельзя рисовать долго, поскольку программа на время его работы блокируется. А если рисовать в других обработчиках, то нарисованное стирается при следующей перерисовке окна.

С появлением языка Pascal ABC, в котором графическая библиотека подключается крайне просто, многие проблемы были решены. В этом языке графическое приложение похоже на обычную программу, в которой подключается графический модуль GraphABC и в блоке операторов после begin end можно писать графические команды.

```
program Demo;
uses GraphABC;
…
Begin
      <процедуры и функции для рисования>
end.
```

По умолчанию графический экран PascalABC содержит 640 точек по горизонтали и 400 точек по вертикали. Начало отсчета – левый верхний угол. При запуске такой программы возникает специальное графическое окно, и все рисование происходит именно на нем. Изображение на нем не пропадает при перерисовке, программа не окажется заблокированной на время рисования и можно рисовать сколь угодно долго и не сложно осуществлять анимацию.

Например:

```
program DemoPixel;
uses GraphABC;
 var  x,y:integer;
begin
  for x:=0 to WindowWidth-1 do
  for y:=0 to WindowHeight-1 do
    SetPixel(x,y,RGB(2*x-y,x-3*y,x+y));
end.
```

```
program DemoTime;
uses GraphABC,Utils;
   Var x0,y0:integer;
       t: DateTime;
       s1,s2,s3:string;
begin
SetWindowCaption('DemoTime');
SetWindowSize(500,100);
SetFontName('Arial');
```

```
SetFontStyle(fsBoldItalic);
SetFontSize(70);
SetFontColor(RGB(Random(255),Random(255),Random(255)));
  x0 := (WindowWidth - TextWidth('00 : 00 : 00 ')) div 2;
  y0 := (WindowHeight - TextHeight('00 : 00 : 00 ')) div 2;
  while True do
  begin
    t := CurrentDateTime;
    //s := string.Format('{0:d2}:{1:d2}:{2:d2}',t.Hour,t.Minute,t.Second);
    str(t.Hour:2,s1);
    str(t.Minute:2,s2);
    str(t.Second:2,s3);
    TextOut(x0,y0,s1+' : '+s2+' : '+s3+' ');
    Sleep(1000);
  end;
end.
```

Возможности модуля растровой графики GraphABC практически совпадают с графическими возможностями Borland Delphi. Процедуры и функции рисования и установки параметров рисования аналогичны методам и свойствам класса TCanvas в Delphi. Например, вместо свойства Canvas.Brush.Color используется пара: процедура SetBrushColor(color) и функция BrushColor.

Модуль GraphABC представляет собой простую графическую библиотеку и предназначен для создания несобытийных графических и анимационных программ в процедурном и частично в объектном стиле. Рисование осуществляется в специальном графическом окне, возможность рисования в нескольких окнах отсутствует. Кроме этого, в модуле GraphABC определены простейшие события мыши и клавиатуры, позволяющие создавать элементарные событийные приложения. Основная сфера использования модуля GraphABC - обучение.

Модуль GraphABC основан на графической библиотеке GDI+, но запоминает текущие перо, кисть и шрифт, что позволяет не передавать их в качестве параметров при вызове графических примитивов. К свойствам пера, кисти и шрифта можно получать доступ как в процедурном, так и в объектном стиле. Например, для доступа к цвету текущего пера используется процедура SetPenColor(c) и функция PenColor, а также свойство Pen.Color.

В модуле GraphABC можно управлять самим графическим окном и компонентом GraphABCControl, на котором осуществляется рисование. По умолчанию компонент GraphABCControl занимает всю клиентскую часть графического окна, однако, на графическое окно можно добавить элементы управления, уменьшив область, занимаемую графическим компонентом (например, так сделано в модулях Robot и Drawman).

Для работы с рисунками используется класс Picture, позволяющий рисовать на себе те же графические примитивы, что и на экране.

Режим блокировки рисования на экране (LockDrawing) позволяет осуществлять прорисовку лишь во внеэкранном буфере, после чего с помощью метода Redraw восстанавливать все графическое окно. Данный метод используется для ускорения анимации и создания анимации без мерцания.

Модуль GraphABC содержит константы, типы, процедуры, функции и классы для рисования в графическом окне. Они подразделяются на следующие группы:

1. Графические примитивы.
2. Действия с цветом.
3. Действия с пером.
4. Действия с кистью.
5. Действия со шрифтом.
6. Действия с рисунками.
7. Действия с графическим окном.
8. Задание режимов вывода

Наиболее используемые графические процедуры и функции.

1. Управление экраном.

SetWindowWidth(w). Устанавливает ширину графическогоокна;

SetWindowHeight(h) -Устанавливает высоту графическогоокна;

2. Графические примитивы.

Точка SetPixel(x,y,color). Закрашиваетодин пиксел с координатами (x,y) цветом color. Например, SetPixel(300,200,clred).

Линии Line(x1,y1,x2,y2). Рисует отрезок с началом в точке (x1,y1) и концом в точке (x2,y2). Например, line(100,50,500,250).

Процедура MoveTo(x,y). Передвигает невидимое перо к точке с координатами (x,y). Эта процедура обычно работает в паре с процедурой LineTo(x,y).

Процедура LineTo(x,y). Рисует отрезок от текущего положения пера до точки (x,y). Координаты пера при этом также становятся равными (x,y).

Прямоугольник Rectangle(x1,y1,x2,y2). Рисует прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2). Например, Rectangle(50,50,200,200).

Процедура Polygon(A, n) строит ломаную линию по n точкам, координаты которых заданы в массиве A элементов типа Point.

Процедура Polyline(A, n) строит замкнутую ломаную линию по n точкам, координаты которых заданы в массиве A элементов типа Point.

Окружность Circle(x,y,r). Рисует окружность с центром в точке (x,y) и радиусом r. Например, Circle(500,200,100).

Дуга окружности Arc(x,y,r,a1,a2). Рисует дугу окружности с центром в точке (x,y) и радиусом r, заключенной между двумя лучами, образующими углы a1 и a2 с осью OX (a1 и a2 – вещественные, задаются в градусах и отсчитываются против часовой стрелки). Например, Arc(300,250,150,45,135);

3. Некоторые используемые цвета

clBlack – черный	clWhite – белый	clMaroon – темно-красный
clTeal – сине-зеленый	clGray – темно-серый	clRed – красный
clLime – ярко-зеленый	clNavy – темно-синий	clGreen – зеленый
clBrown – коричневый	clLtGray – светло-серый	clBlue – синий
clSkyBlue – голубой	clYellow – желтый	clMedGray – серый

Random(16777215) – случайный цвет из всей палитры цветов Паскаля.

Функция RGB(r,g,b) возвращает целое значение, являющееся кодом цвета, который содержит r красную, g зеленую и b синюю составляющие (r, g и b – целые в диапазоне от 0 до 255, причем, 0 соответствует минимальной интенсивности, 255 – максимальной). Например, RGB(Random(255),Random(255),Random(255)), где функция Random(x) возвращает случайное целое в диапазоне от 0 до x-1.

Процедура цвет линии SetPenColor(color) устанавливает цвет пера, задаваемый параметром color. Например, setpencolor(clred).

Процедура заливка цветом FloodFill(x,y,color) заливает область одного цвета цветом color, начиная с точки (x,y). Например, Rectangle(50,50,200,200); FloodFill(100,100,clBlue)

4. Шрифты.

Процедура procedure SetFontName(name: string) устанавливает наименование шрифта.

Процедура procedure SetFontStyle(style: integer); - устанавливает стиль шрифта.

Процедура procedure SetFontSize(sizesz: integer); - устанавливает размер шрифта в пунктах.

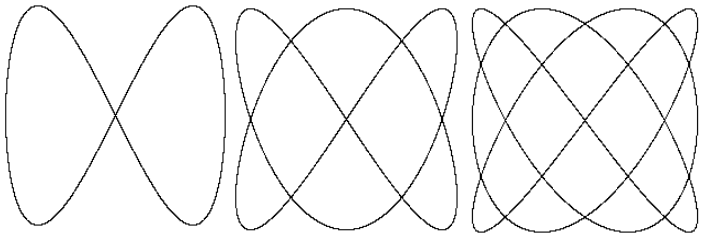
Процедура procedure SetFontColor(color: integer); - устанавливает цвет шрифта.

Процедура procedure TextOut(x, y, s) выводит строку s в позицию (x,y). Точка (x,y) задает верхний левый угол прямоугольника, который будет содержать текст из строки s).

Например,

```
SetFontName('Arial');
SetFontStyle(fsBoldItalic);
SetFontSize(40);
SetFontColor(RGB(Random(255),Random(255),Random(255)));
```

Некоторые задания для выполнения лабораторных работ.



$w_1=1$; $w_2=2$ $w_1=2$; $w_2=3$ $w_1=3$; $w_2=4$

Рис. 1. Фигуры Лиссажу

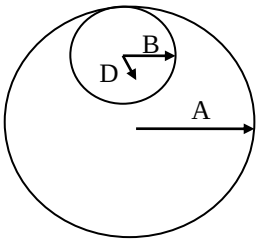
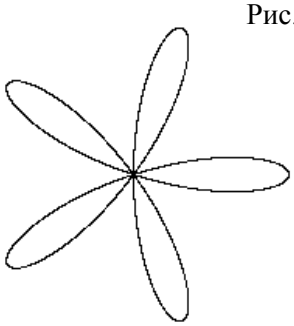
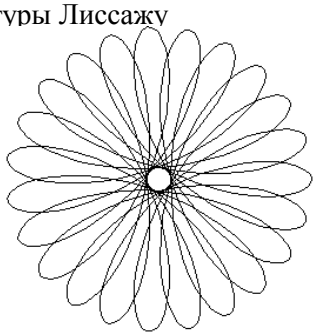


Рис. 2. Модель для построения спирографа





A=15; B=10; D=5; M=3 A=75; B=40; D=30; M=8

Рис.3. Спирограф

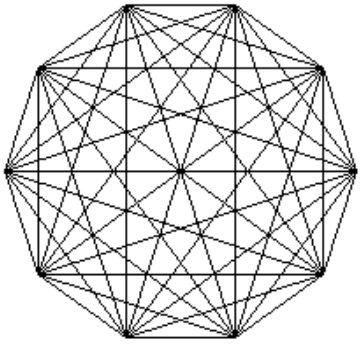


Рис. 4. Кружева

Фигуры Лиссажу:

X=sin w_1t

Y= sin w_2t ,

где $2\pi \geq t \geq 0$, w_1 и w_2 – константы

Спирограф:

X=(A - B) cos t + D cos f

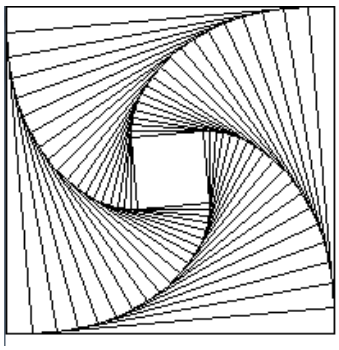
Y=(A - B) sin t - D sin f ,

где f = (A/B) t,

D < B < A, 2π M ≥ t ≥ 0,

M = B/H

(H - наибольший общий делитель A и B)

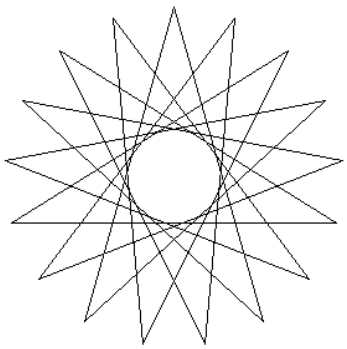


Кружева:

x_i=R *cos (2π/N*i)

y_i=R* cos (2π/N*i),

где R – радиус, N – количество вершин, причем каждая i-ая вершина соединяется с каждой вершиной, начиная с i+1-ой до N вершины.



Квадраты, закрученные по спирали:

$x_{i+1}= x_i + (x_j - x_i)*(1-t)$

$y_{i+1}= y_i + (y_j - y_i)*(1-t)$

где x_i, y_i и x_j y_j – координаты i-ых и соседних j-ых вершин прямоугольников, i =1.. N, N – количество прямоугольников, а t – коэффициент, в соответствии с которым делятся стороны прямоугольников, t =0.. 1. На рисунке t =9.92, а N=20.

Звездочка:

x_i=R *cos (2π/N*i)

y_i=R* cos (2π/N*i),

где R – радиус, N – количество вершин, причем каждая i-ая вершина соединяется только с i+M -ой вершиной. На рисунке M =7, а N=17.

Расширенный перечень графических процедур и функций.

Графические примитивы.

procedure SetPixel(x,y,color: integer); - закрашивает один пиксел с координатами (x,y) цветом color.

function GetPixel(x,y): integer; -возвращает текущее значение цвета для пиксела с координатами (x,y).

procedure MoveTo(x,y: integer); - передвигает невидимое перо к точке с координатами (x,y); эта функция работает в паре с функцией LineTo(x,y).

procedure LineTo(x,y: integer); - рисует отрезок от текущего положения пера до точки (x,y); координаты пера при этом также становятся равными (x,y).

procedure Line(x1,y1,x2,y2: integer); - рисует отрезок с началом в точке (x1,y1) и концом в точке (x2,y2).

procedure Circle(x,y,r: integer); - рисует окружность с центром в точке (x,y) и радиусом r.

procedure Ellipse(x1,y1,x2,y2: integer); - рисует эллипс, заданный своим описанным прямоугольником с координатами противоположных вершин (x1,y1) и (x2,y2).

procedure Rectangle(x1,y1,x2,y2: integer); - рисует прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2).

procedure RoundRect(x1,y1,x2,y2,w,h: integer); - рисует прямоугольник со скругленными краями; (x1,y1) и (x2,y2) задают пару противоположных вершин, a w и h – ширину и высоту эллипса, используемого для скругления краев.

procedure Arc(x,y,r,a1,a2: integer); - рисует дугу окружности с центром в точке (x,y) и радиусом r, заключенной между двумя лучами, образующими углы a1 и a2 с осью OX (a1 и a2 – вещественные, задаются в градусах и отсчитываются против часовой стрелки).

procedure Pie(x,y,r,a1,a2: integer); - рисует сектор окружности, ограниченный дугой (параметры процедуры имеют тот же смысл, что и в процедуре Arc).

procedure Chord(x,y,r,a1,a2: integer); - рисует фигуру, ограниченную дугой окружности и отрезком, соединяющим ее концы (параметры процедуры имеют тот же смысл, что и в процедуре Arc).

procedure TextOut(x,y: integer; s: string); - выводит строку s в позицию (x,y) (точка (x,y) задает верхний левый угол прямоугольника, который будет содержать текст из строки s).

procedure FloodFill(x,y,color: integer); - заливает область одного цвета цветом color, начиная с точки (x,y).

procedure FillRect(x1,y1,x2,y2: integer); - заливает прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2), цветом текущей кисти.

procedure Polygon(var a; n: integer); строит ломаную по n точкам, координаты которых заданы в массиве a элементов типа Point.

procedure Polyline(var a; n: integer); -строит замкнутую ломаную по n точкам, координаты которых заданы в массиве a элементов типа Point.

Цветовые константы и функции для работы с цветом

Модуль GraphABC содержит константы и функции для работы с цветами. Тип ColorType, описывающий цвет, определен следующим образом:

type ColorType=integer; - стандартные цвета задаются символическими константами:

clBlack – черный	clPurple – фиолетовый	clWhite – белый
clMaroon – темно-красный	clRed – красный	clNavy – темно-синий
clGreen – зеленый	clBrown – коричневый	clBlue – синий
clSkyBlue – голубой	clYellow – желтый	clCream – кремовый
clAqua – бирюзовый	clOlive – оливковый	clFuchsia – сиреневый
clTeal – сине-зеленый	clGray – темно-серый	clLime – ярко-зеленый
clMoneyGreen – цвет зеленых денег	clLtGray – светло-серый	clDkGray – темно-серый
clMedGray – серый	clSilver – серебряный	

Для работы с цветами используются следующие функции.

function RGB(r,g,b: integer): ColorType; - возвращает целое значение, являющееся кодом цвета, который содержит красную, зеленую и синюю составляющие с интенсивностями r, g и b соответственно (r, g и b – целые в диапазоне от 0 до 255, причем, 0 соответствует минимальной интенсивности, 255 – максимальной).

function GetRed(color: ColorType): integer; - выделяет красную составляющую из цвета color (целое в диапазоне от 0 до 255);

function GetGreen(color: ColorType): integer; - выделяет зеленую составляющую из цвета color (целое в диапазоне от 0 до 255);

function GetBlue(color: ColorType): integer; - выделяет синюю составляющую из цвета color (целое в диапазоне от 0 до 255).

Действия с пером

function PenX: integer;

function PenY: integer; - возвращают текущие координаты пера.

procedure SetPenColor(color: integer); - устанавливает цвет пера, задаваемый параметром color.

function PenColor: integer; - возвращает текущий цвет пера.

procedure SetPenWidth(w: integer); - устанавливает ширину пера, равную w пикселам.

function PenWidth: integer; - возвращает текущую ширину пера.

procedure SetPenStyle(ps: integer); - устанавливает стиль пера, задаваемый параметром ps.

function PenStyle: integer; - возвращает текущий стиль пера.

Стили пера задаются следующими именованными константами:

psSolid Сплошная линия (установлено по умолчанию)

psDash Штриховая линия

psDot Пунктирная линия

psDashDot Штрихпунктирная линия

psDashDotDot Линия, чередующая штрих и два пунктира

psClear Отсутствие линии

procedure SetPenMode(m: integer); - устанавливает режим пера, задаваемый параметром m.

function PenMode: integer; - возвращает текущий режим пера. Режим пера определяет, как цвет пера взаимодействует с цветом поверхности.

Режимы пера задаются следующими именованными константами:

pmCopy – обычный режим; при рисовании цвет поверхности заменяется цветом пера;

pmNot – режим инвертирования; при рисовании цвет поверхности инвертируется (становится негативным), а цвет пера при этом игнорируется.

Действия с кистью

procedure SetBrushColor(color: integer); - устанавливает цвет кисти, задаваемый параметром color.

function BrushColor: integer; - возвращает текущий цвет кисти.

procedure SetBrushPicture(fname: string); - устанавливает в качестве образца для закраски кистью образец, хранящийся в файле fname, при этом текущий цвет кисти при закраске игнорируется.

procedure ClearBrushPicture; - очищает рисунок-образец, выбранный для кисти.

procedure SetBrushStyle(bs: integer); - устанавливает стиль кисти, задаваемый параметром bs.

function BrushStyle: integer; - возвращает текущий стиль кисти.

Стили кисти задаются следующими именованными константами:

bsSolid	bsCross	bsClear	bsDiagCross
bsHorizontal	bsBDiagonal	bsVertical	bsFDiagonal

Действия со шрифтом

procedure SetFontColor(color: integer); - устанавливает цвет шрифта.

function FontColor: integer; - возвращает текущий цвет шрифта.

procedure SetFontSize(sz: integer); - устанавливает размер шрифта в пунктах.

function FontSize: integer; - возвращает текущий размер шрифта в пунктах.

procedure SetFontName(name: string); - устанавливает наименование шрифта.

function FontName: string; - возвращает текущее наименование шрифта.

По умолчанию установлен шрифт, имеющий наименование MS Sans Serif. Наиболее распространенные шрифты – это Times, Arial и Courier New. Наименование шрифта можно набирать без учета регистра.

procedure SetFontStyle(fs: integer); - устанавливает стиль шрифта.

function FontStyle: integer; - возвращает текущий стиль шрифта.

Стили шрифта задаются следующими именованными константами:

fsNormal – обычный;

fsBold – жирный;

fsItalic – наклонный;

fsBoldItalic – жирный наклонный;

fsUnderline – подчеркнутый;

fsBoldUnderline – жирный подчеркнутый;

fsItalicUnderline – наклонный подчеркнутый;

fsBoldItalicUnderline – жирный наклонный подчеркнутый.

function TextWidth(s: string): integer; - возвращает ширину строки s в пикселях при текущих настройках шрифта.

function TextHeight(s: string): integer; - возвращает высоту строки s в пикселях при текущих настройках шрифта.

Действия с графическим окном

procedure ClearWindow; - очищает графическое окно белым цветом.

procedure ClearWindow(c: ColorType); - очищает графическое окно цветом c.

function WindowWidth: integer; - возвращает ширину графического окна.

function WindowHeight: integer; - возвращает высоту графического окна.

function WindowLeft: integer; - возвращает отступ графического окна от левого края экрана.

function WindowTop: integer; - возвращает отступ графического окна от верхнего края экрана.

function WindowCaption: string; - возвращает заголовок графического окна.

procedure SetWindowWidth(w: integer); - устанавливает ширину графического окна.

procedure SetWindowHeight(h: integer); - устанавливает высоту графического окна.

procedure SetWindowLeft(l: integer); - устанавливает отступ графического окна от левого края экрана.

procedure SetWindowTop(t: integer); - устанавливает отступ графического окна от верхнего края экрана.

procedure SetWindowSize(w,h: integer); - устанавливает ширину и высоту графического окна.

procedure SetWindowPos(l,t: integer); - устанавливает отступ графического окна от левого и верхнего края экрана.

procedure SetWindowCaption(s: string); - устанавливает заголовок графического окна.

procedure SetWindowTitle(s: string); - устанавливает заголовок графического окна. Синоним SetWindowCaption.

procedure SaveWindow(fname: string); - сохраняет содержимое графического окна в файл с именем fname.

procedure LoadWindow(fname: string); - выводит в графическое окно рисунок из файла с именем fname. Файл ищется вначале в текущем каталоге, а затем в каталоге PascalABC\Media\Images.

procedure FillWindow(fname: string); - заполняет графическое окно мозаикой из рисунка, содержащегося в файле с именем fname.

procedure FillWindow(n: integer); - заполняет графическое окно мозаикой из рисунка с описателем n.

procedure CloseWindow; - закрывает графическое окно.

function ScreenWidth: integer; - возвращает ширину экрана.

function ScreenHeight: integer; - возвращает высоту экрана.

procedure CenterWindow; - центрирует графическое окно по центру экрана.

procedure MaximizeWindow; - максимизирует графическое окно на экране.

procedure NormalizeWindow; - восстанавливает положение графического окна на экране.

Все размеры устанавливаются и возвращаются в пикселах.

Задание режимов вывода

procedure SetDrawingSurface(n: integer); - устанавливает в качестве канвы для рисования рисунок с описателем n. В результате весь графический вывод осуществляется не на экран, а на рисунок; настройки кисти, пера и шрифта также осуществляются для рисунка.

procedure SetDrawingSurface(p: Picture); - устанавливает в качестве канвы для рисования рисунок с описателем n. В результате весь графический вывод осуществляется не на экран, а на рисунок; настройки кисти, пера и шрифта также осуществляются для рисунка.

procedure RestoreDrawingSurface; - устанавливает в качестве канвы для рисования графическое окно.

procedure Redraw; - осуществляет перерисовку окна вывода при заблокированном выводе в графическое окно.

procedure LockDrawing; - блокирует вывод в графическое окно, осуществляя рисование только во внеэкранном буфере. Для перерисовки графического окна требуется вызвать процедуру Redraw. Если графический вывод перенаправлен в рисунок вызовом процедуры SetDrawingSurface, то не оказывает никакого воздействия на вывод.

procedure UnlockDrawing; - снимает блокировку вывода в графическое окно.

procedure LockScreenBuffer; - блокирует вывод во внеэкранный буфер графического окна. После вызова этой процедуры рисование незначительно ускоряется, однако, изображение графического окна перестает восстанавливаться.

procedure UnlockScreenBuffer; - снимает блокировку вывода во внеэкранный буфер графического окна.

function DrawingIsLocked: boolean; - возвращает True, если вывод в графическое окно заблокирован, и False в противном случае.

procedure SetRedrawProc(procedure RedrawProc); - устанавливает пользовательскую процедуру для перерисовки содержимого графического окна, вызываемую автоматически в тот момент, когда требуется его перерисовка. В настоящее время используется в модуле ABCObjects для автоматической перерисовки всех графических объектов и фона.

Блокировка вывода в графическое окно с последующим вызовом Redraw используется для простейшего создания анимации без мерцания.