ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Федеральное государственное образовательное учреждение
высшего профессионального образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Н.И. АМЕЛИНА, А.А. ЧЕКУЛАЕВА, М.И. ЧЕРДЫНЦЕВА

ГРАФИКА В СИСТЕМЕ PascalABC

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по курсу

«ИНФОРМАТИКА»

для студентов 1 курса дневного и вечернего отделений

факультета математики, механики и компьютерных наук

Ростов-на-Дону

2008

Методические указания разработаны сотрудниками кафедры прикладной математики и программирования старшими преподавателями Н.И. Амелиной и А.А. Чекулаевой и доцентом Чердынцевой М.И.

Печатается в соответствии с решением кафедры прикладной математики и программирования факультета математики, механики и компьютерных наук ЮФУ, протокол № 1 от 4 сентября 2008г.

СОДЕРЖАНИЕ

1 РАБОТА С ГРАФИКОЙ

Графическое изображение представляет собой совокупность отдельных точек – *пикселей*, которые можно закрасить в различные цвета. Каждый пиксель имеет две координаты: X и Y. Ось OX направлена слева направо, начиная с 0, а ось OY – сверху вниз, также начиная с 0. То есть левый верхний пиксель имеет координаты (0,0).

В состав стандартных средств языка Паскаль не входят операторы для работы с графикой. В каждой реализации транслятора для этого предназначена специальная библиотека, содержащая процедуры и функции для работы в графическом режиме, как в системе программирования **ТурбоПаскаль,** или с графическим окном системы **PascalABC**. Для использования средств соответствующей библиотеки необходимо в начале программы выполнить ее подключение следующим образом:

`Uses` *имя библиотеки*;

Графические возможности модуля **Graph** системы программирования **ТурбоПаскаль** приводятся, например, в [6].

В системе **PascalABC** библиотека графики **GraphABC** подключается так:

```
Uses GraphABC;
```

Описание некоторых, часто используемых, процедур и функций, входящих в библиотеку **GraphABC**, приведено в Приложении. Более подробную информацию можно получить в справке по **PascalABC**(клавиша F1) или в [2].

В системе **PascalABC** графическое окно выводится как дочернее окно. При этом остаются доступными средства консольного ввода–вывода информации. Для переключения между окнами следует использовать клавишу F6 или соответствующие команды меню.

2 ПРОЦЕДУРНЫЙ ТИП

Процедурный тип описывает класс процедур или функций, имеющих однотипные заголовки, т.е. однотипные списки параметров. Имена формальных параметров не являются существенными в этих описаниях. Важно только их количество, порядок следования, типы и способ передачи, а также тип результата для функции.

Определение процедурного типа аналогично заголовку подпрограммы, но при этом имя подпрограммы не задается. Например,

```
type PROC1 = procedure (a, b, c: real; var d: real);
     PROC2 = procedure;
     FUNC  = function (x: real):real;
     FUNC1 = function : real;
```

Введенный процедурный тип может быть использован при описании формальных параметров подпрограмм. Например,

```
procedure MENU (x, y: integer; SHOW: PROC2);
function  SUMMA(EPS: real; F: FUNC);
```

Если подпрограмма имеет параметр процедурного типа, то при ее вызове в качестве фактического параметра должна использоваться процедура (функция), заголовок которой соответствует заголовку, описанному в процедурном типе. Это должна быть обязательно пользовательская процедура (функция), использовать в качестве фактического параметра стандартные функции нельзя.

Например, функции с заголовками, которые соответствуют типу `FUNC`:

```
function F1(x: real):real;
function G2(x: real):real;
```

могут быть использованы как фактические параметры при вызове функции `SUMMA`:

```
Sf := SUMMA (0.001, F1);
Sg := SUMMA (0.00001, G2);
```

3 ПРИМЕРЫ ПРОГРАММ

Пример 1 Построить столбчатую диаграмму (гистограмму), отображающую динамику среднесуточных температур воздуха в течение недели.

Для ввода значений температуры, можно воспользоваться датчиком случайных чисел Random.

```
program Graf_1;
uses GraphABC;
const dx=50; {ширина столбика}
      z=20;  {расстояние между столбиками}
      k=7;   {число дней в неделе}
      delta = 10;{диапазон изменения температур от -10° до 10°}
var x, y, color, i, t, xm, ym: integer;
begin
  TextOut(250,0,'ТЕМПЕРАТУРА ВОЗДУХА ЗА НЕДЕЛЮ');
  xm:=WindowWidth;
 ym:=WindowHeight;
  y:=ym div 2;
  x:=dx+z;
  Line(x-z,y,xm-z,y);          {ось OX}
  Line(x-z,z,x-z,ym-z);        {ось OY}
  TextOut(x-2*z,y,'0');
  for i:=1 to k do
  begin
    t:= Random(2*delta)-delta;
    if t>0 then color:=clRed
           else color:=clBlue;
    SetPenColor(color);
    Rectangle(x,(y-10*t),x+dx,y);
```

```
    x:=x+dx+z
  end
end.
```

Результат работы программы `Graf_1` приведен на рисунке 1.

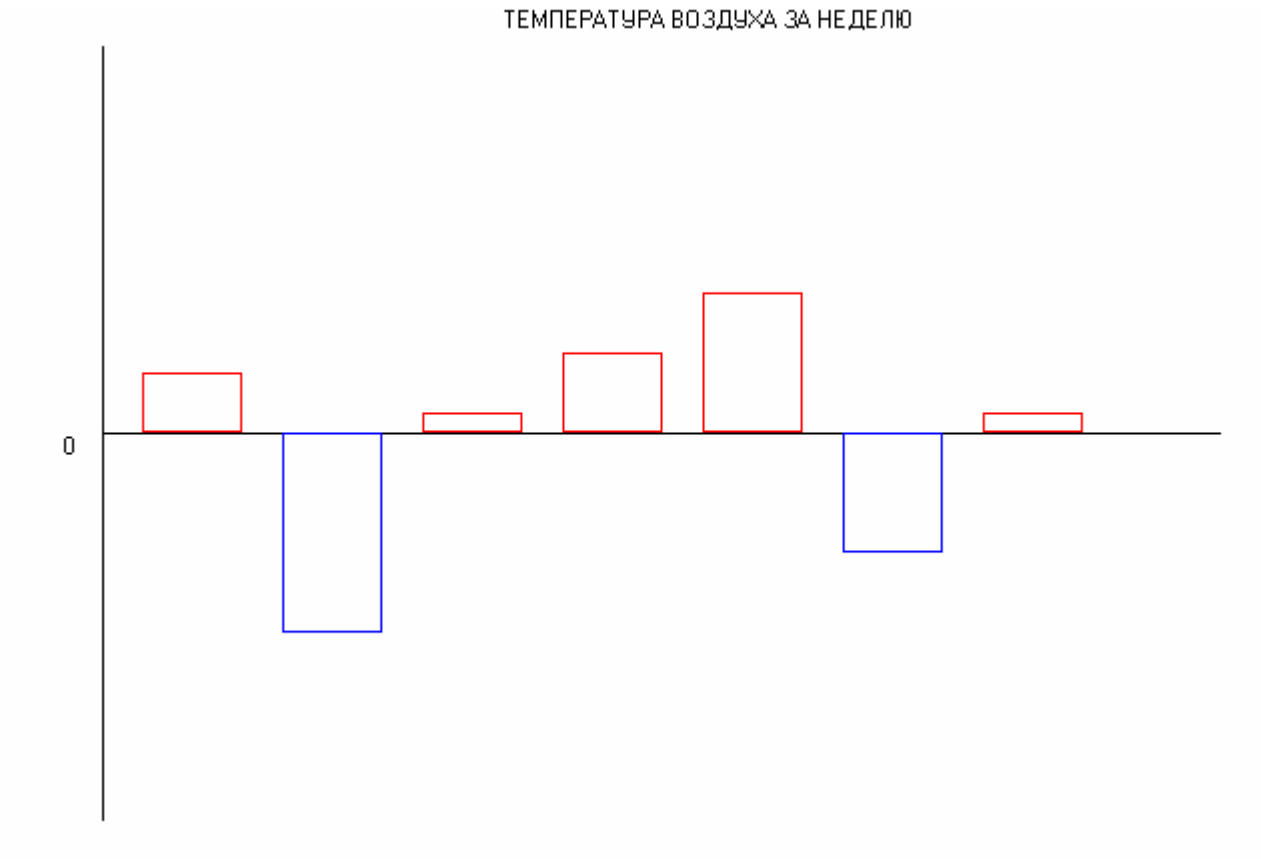

Рисунок 1 Гистограмма

Упражнения

1) Ввести в гистограмме подписи значений температуры под соответствующими столбцами.
2) Увеличить период наблюдений за температурой воздуха до декады, месяца.
3) Заменить диапазон изменения температур на [-20^{o} , $+40^{o}$].
4) Выполнить заливку столбиков гистограммы цветом границы прямоугольника (красным или синим), используя процедуру заливки `FloodFill` библиотеки **GraphABC** (см. Приложение).

5) Объяснить, почему при заливке столбиков (упражнение 4)) в случае нулевой температуры ось OX, а иногда и ось OY, окрашивается цветом границы.
6) Выполнить заливку столбиков гистограммы цветами, отличными от цветов границ, например, темно–красным или темно–синим (см. Приложение).
7) Организовать в программе ввод значений температур.
8) Построить гистограмму, отображающую значения дневных и ночных температур за неделю.

Пример 2 Построить график функции *f(x)* на отрезке [*a*, *b*]. Считать, что функция на отрезке определена и непрерывна. При построении графика использовать такой масштаб, чтобы отрезок [*a*, *b*] занимал все графическое окно по ширине. Точки, соответствующие наибольшему (y_{max}) и наименьшему (y_{min}) значениям функции, должны располагаться соответственно в верхней и нижней части графического окна.

Если значение *x=0* принадлежит отрезку [*a, b*], то изобразить на графике ось *OY*. Если отрезок [y_{min} ,y_{max}] содержит значение 0, то изобразить на графике ось *OX*.

Наибольшую трудность при построении графика вызывает его масштабирование в границах графического окна. Координатные оси графического окна направлены слева – направо (ось *OX*) и сверху – вниз (ось *OY*). Графические координаты принимают только целые неотрицательные значения. Максимальные значения графических координат определяются размерами графического окна (шириной и высотой). Точка в левом верхнем углу графического окна имеет графические координаты (0, 0). Точка в правом нижнем углу – графические координаты (*W, H*), где *W* – ширина графического окна, *H* – высота графического окна.

Для определения графических координат x_g и y_g можно воспользоваться следующими формулами преобразования:

$x_g = round\ (x_0 + (x - a) * Mx)$,

где $x \in [a, b]$, $Mx = (x_W - x_0)/(b - a)$,

x_0, x_W – графические координаты границы рисунка графика;

$y_g = round(y_0 + (y_{max} - y) * My)$,

где $y \in [y_{min}, y_{max}]$, $My = (y_H - y_0)/(y_{max} - y_{min})$,

y_0, y_H – графические координаты границы рисунка графика;

round (X) – функция округления вещественного X до ближайшего целого.

Координаты x_0, x_W, y_0, y_H могут быть установлены равными соответственно 0, W, 0, H – в этом случае график займет все графическое окно. Можно установить значения x_0, x_W, y_0, y_H такими, чтобы вокруг графика были отступы определенного размера.

```
program Graf_2;
uses GraphABC;
var a, b : real;
    n : integer;
function F(x:real):real;
begin
  F:= x*x-4*abs(x)+3
end;
procedure MaxMin(a, b, h : real; var ymin, ymax : real);
var x,y:real;
begin
  ymin:=F(a); ymax:=ymin;
  x:=a;
```

```
  while x < b+h/2 do
  begin
    y:=F(x);
    if y < ymin then
      ymin:=y;
    if y > ymax then
      ymax:=y;
    x:=x+h
  end
end;
procedure GrFunc(a, b : real; n : integer);
var ymin, ymax x, y, h, Mx, My : real;
    xg, yg, xgp, ygp, ox, oy,
    x0, y0, xW, yH, i : integer;
  function prx(x : real):integer;
  begin
    prx:=round(x0+(x-a)*Mx)
  end;
  function pry(y : real):integer;
  begin
    pry:=round(y0+(ymax-y)*My)
  end;
begin
  h:=(b-a)/n; MaxMin(a,b,h,ymin,ymax);
  x0:=0; xW:=WindowWidth;
  y0:=0; yH:=WindowHeight;
  Mx:=(xW-x0)/(b-a);
  My:=(yH-y0)/(ymax-ymin);
```

```
  { Выводятся оси }
  if ymin*ymax < 0 then
  begin
    oy:=pry(0);
    Line(x0,oy,xW,oy);
    TextOut( xW-15,oy+10,'Ox')
  end;
  if a*b < 0 then
  begin
    ox:=prx(0);
    Line(ox,y0,ox,yH);
    TextOut( ox+10,y0+15,'Oy')
  end;
  x:=a;y:=f(a);
  xgp:=prx(x); ygp:=pry(y);
  for i:=1 to n do
  begin
    x:=x+h; y:=f(x);
    xg:=prx(x); yg:=pry(y);
    Line(xgp,ygp,xg,yg);
    xgp:=xg; ygp:=yg
  end;
end;
begin
  n:=50;
  write ('Введите концы отрезка a..b ');
  readln(a,b);
  GrFunc(a,b,n)
end.
```

Замечание В процедуре `GrFunc` используются две вспомогательные функции `prx` и `pry` – выполняющие преобразование координат в графические. Они описаны как локальные, так как их использование вне процедуры `GrFunc` не имеет смысла. При этом, чтобы не усложнять список параметров этих функций, в этих функциях используются нелокальные переменные `x0, a, Mx, y0, ymax, My`.

Результат работы программы – график функции $f(x) = x^2 - 4\,abs(x) + 3$ на отрезке $[-3, 3]$ приведен на рисунке 2.

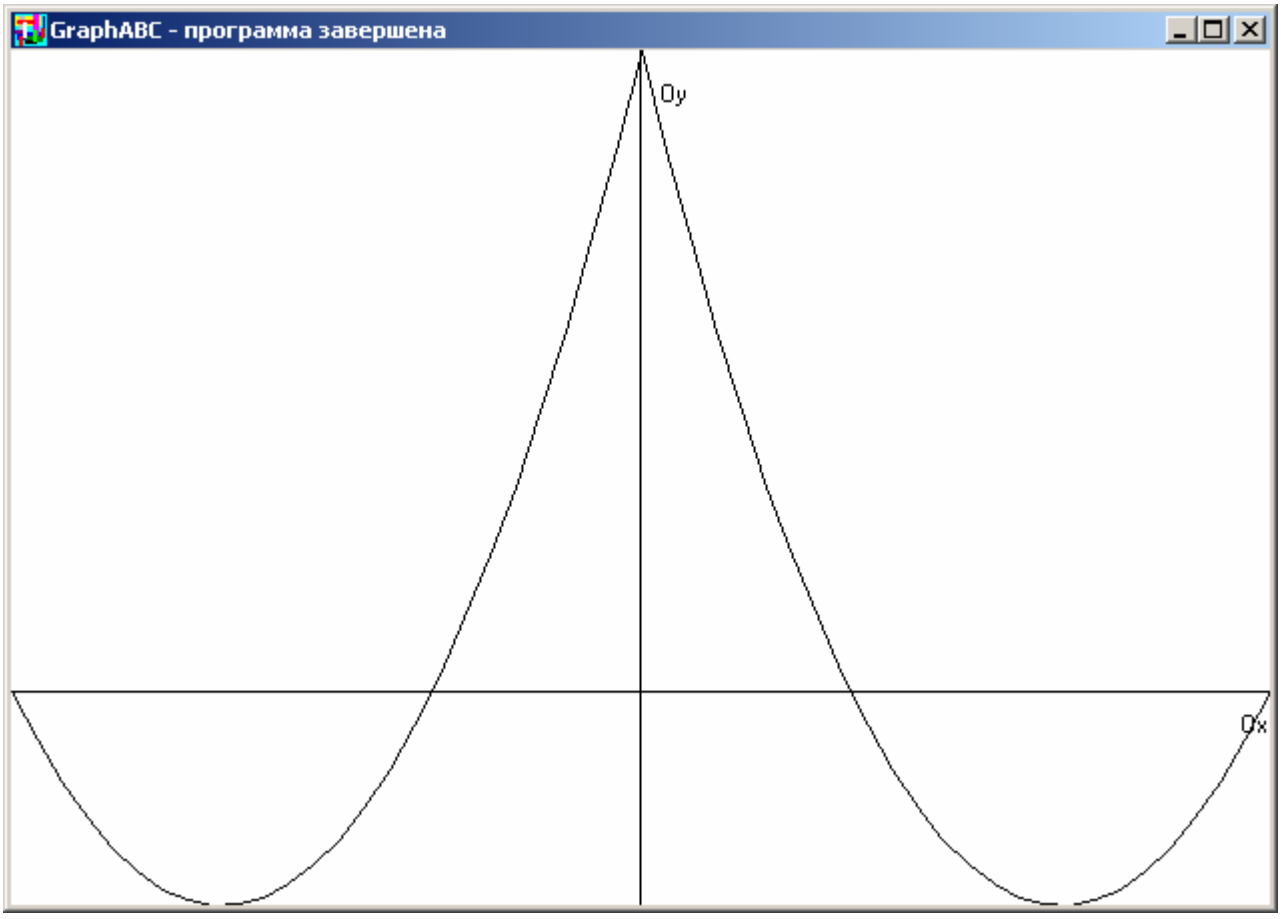

Рисунок 2 График функции

Упражнения

1) Проанализировать влияние изменения количества точек разбиения на вид графика (например, четное и нечетное количество точек).
2) Внести изменения так, чтобы график изображался с отступом от границы графического окна.
3) Изменить цвет пера при рисовании в графическом окне: график рисовать одним цветом, а оси – другим.
4) Для облегчения анализа графика удобно, кроме осей координат, которые часто могут отсутствовать на графике, наносить координатную сетку. Добавить в процедуру построения графика возможность изображения равномерной сетки, выводом которой можно было бы управлять с помощью дополнительного параметра логического типа.
5) При использовании координатной сетки (см. предыдущее упражнение) предусмотреть вывод значений, соответствующих линиям сетки.
6) Уменьшить количество точек разбиения при построении графика до 10–20. Построить график с выделением опорных точек (точек, в которых проводилось вычисление значения функции). Например, опорные точки можно отмечать контрастным цветом или маленькими графическими фигурами – окружностями, квадратами, отрезками вертикальных линий и т.д.
7) Добавить в заголовок процедуры построения графика `GrFunc(a,b,n)` параметр процедурного типа (см. раздел 2), чтобы при ее вызове в качестве фактического параметра можно было использовать любую функцию типа `F(x)`. В программе описать несколько функций этого типа и для выбора одной из них для построения графика использовать оператор выбора CASE.
8) Изменить процедуру построения графика `GrFunc(a,b,n)` так, чтобы можно было выводить одновременно графики двух функций.

4 ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

Даны две функции *f(x)* и *g(x)* (таблица 1).

Уравнение *f(x) = 0* имеет на указанном в таблице 1 отрезке единственный корень. Вычислить приближенное значение корня уравнения на отрезке с заданной точностью одним из приближенных методов [5]: методом половинного деления, методом касательных, методом хорд и т. п.

Уравнение *g(x) = 0* имеет на указанном в таблице 1 отрезке несколько корней. Исследовать функцию, используя процедуру построения ее графика, и определить отрезки, на которых находятся корни уравнения. Найти каждый из корней уравнения с заданной точностью одним из приближенных методов.

Создать программу решения уравнений *f(x) = 0* и *g(x) = 0* , используя процедуру построения графика функции для отделения корней и один или несколько методов приближенного решения уравнения для уточнения корней. Программа должна обладать дружественным интерфейсом и в диалоге предоставлять следующие возможности: выбор одного из уравнений, вывод графика функции на заданном отрезке, выбор метода решения и поиск корня уравнения на отрезке. Программа должна иметь модульную структуру и содержать меню (или набор меню) с перечнем возможностей и оператор (операторы) выбора соответствующего пункта меню.

Таблица 1. Функции и отрезки

Функция *f(x)*	Отрезок	Функция *g(x)*	Отрезок
1) $x^2 cos\ 2x+1$	[0, π/2]	$x^4 – x^3 – 2x^2 + 3x – 3$	[–3, 3]
2) $tg\ x – (x+1)/2$	[0, π/4]	$x^4 – 4\ x – 1$	[–2, 3]
3) $x^5 – 0.3 \mid x – 1 \mid$	[0, 1]	$x^4 – 2x^3 + x – 1.5$	[–2, 3]
4) $2x – cos\ x$	[0, π/2]	$x^4 + 4x^3 + 4.8\ x^2 + 16x – 1$	[–5, 2]
5) $2x^2 + 4x – 1$	[0, 1]	$5^{\ x} – 6x – 3$	[–2, 3]

Продолжение таблицы 1

6) $0.9x - \sin(x^{1/2}) - 0.1$	[0, 1.5]	$x^3 + 3x^2 - 60x + 100$	[–12, 10]
7) $x^3 - \sin x$	[0.5, 1]	$x^3 + 5x^2 - 4x - 20$	[– 6, 2]
8) $5(x-3)^2 - 9$	[0, 10]	$e^x - 10x$	[0, 7]
9) $x + \ln(x + 0.5) - 0.5$	[0, 2]	$x^4 + 2x^3 + x - 1$	[–3, 2]
10) $5x - 8 \ln x - 8$	[1, 5]	$x^3 - 2x^2 - 4x + 7$	[–3, 3]
11) $x^5 - x - 0.2$	[1, 1.1]	$2\cos(x + \pi/6) + x^2 - 3x + 2$	[0, 5]
12) $x^3 - 0.2x^2 - 0.2x - 1.2$	[1, 1.5]	$x^3 + 3.1x^2 + 0.28x - 0.06$	[–3.5, 0.2]
13) $x + 0.06x^2 + 0.15x - 0.8$	[0, 2.5]	$x^3 + 4x^2 - 7x - 10$	[–6, 3]
14) $x \cdot 2^x - 1$	[0, 1]	$x^3 + 7.1x^2 + 0.64x - 0.42$	[–7.5, 0.3]
15) $\frac{2\sin^2 x}{3} - \frac{3\cos^2 x}{4}$	[0, π/2]	$x^3 + 2x^2 - 0.09x - 0.18$	[–3, 1]
16) $x^2 - \sin 5x$	[0.5, 0.6]	$x^3 - 3.9x^2 - 0.6x + 0.8$	[–1, 5]
17) $10x - 10 - \sin x$	[1, 2]	$x^3 + 6.8x^2 - 9.25x + 2.8$	[–8.2, 0.9]
18) $(4 + x^2)(e^x - e^{-x}) - 18$	[1.2, 1.3]	$x^3 + 6.3x^2 + 1.7x - 0.6$	[–7, 0.5]
19) $x^3 - 2x^2 + x - 3$	[2.1, 2.2]	$3x^3 + 13x^2 - 160x + 100$	[–10, 7]
20) $5x^3 - 10x^2 + 5x - 1$	[0, 1]	$tg(0.55x + 0.1) - x^2$	[–1, 1]
21) $(x-1) \cdot \sin x - x - 1$	[–π/2, 0]	$2x^3 + 3x^2 - 12x - 8$	[–3, 3]
22) $e^x - 2(1 - x^2) - 1$	[0, 1]	$x^3 + 18x^2 - 40x + 20$	[–21, 2]
23) $1 + x^2 + 4\sin x$	[–1,0]	$x^3 - 6x + 2$	[–3, 3]
24) $2x + \lg(2x+3) - 1$	[0, 1]	$x^3 - 5x + 3$	[–3, 3]
25) $x^3 + x^2 + x + 1$	[–2, 1]	$\frac{(2x^2 - 5x + 1)^2}{x^2} - x - 3 \cdot \lg x$	[1, 5]

Приложение

Модуль GraphABC системы PascalABC

Стандартный модуль **GraphABC** системы **PascalABC** содержит типы, константы, переменные, процедуры и функции, позволяющие создавать изображения в специальном в *графическом окне*.

В приложении приведены основные данные о возможностях модуля **GraphABC**: управление цветом, рисование простейших графических объектов, вывод текста, управление графическим окном. Более подробную информацию можно получить в справке по **PascalABC** или в [2].

Описания подпрограмм модуля даны в алфавитном порядке: сначала процедуры, потом функции.

УПРАВЛЕНИЕ ЦВЕТОМ

Стандартные цвета задаются константами, приведенными в алфавитном порядке в таблице 1.

Таблица 1 Стандартные цвета модуля **GraphABC**

Имя	Цвет	Интенсивность
`clAqua`	бирюзовый	0, 255, 255
`clBlack`	черный	0, 0, 0
`clBlue`	синий	0, 0, 255
`clBrown`	коричневый	128, 64, 0
`clCream`	кремовый	255, 251, 240
`clDarkGray`	темно-серый	64, 64, 64
`clFuchsia`	сиреневый	255, 0, 255
`clGray`	серый	128, 128, 128

Продолжение таблицы 1

clGreen	зеленый	0, 128, 0
clLtGray	светло-серый	192, 192, 192
clLime	ярко-зеленый	0, 255, 0
clMaroon	темно-красный	128, 0, 0
clMoneyGreen	цвет зеленых денег	192, 220, 192
clNavy	темно-синий	0, 0, 128
clOlive	оливковый	128, 128, 0
clPurple	фиолетовый	128, 0, 128
clRed	красный	255, 0, 0
clSkyBlue	голубой	166, 202, 240
clTeal	сине-зеленый	0, 128, 128
clWhite	белый	255, 255, 255
clYellow	желтый	255, 255, 0

Функции для работы с цветом

Имя	Назначение
RGB	код цвета
GetRed	выделение красной составляющей цвета
GetGreen	выделение зеленой составляющей цвета
GetBlue	выделение синей составляющей цвета
GetPixel	текущий цвет пикселя
clRandom	случайный цвет

Функция clRandom

function clRandom: integer;

Возвращает случайный цвет.

Функция GetBlue

function GetBlue(color: integer): integer;

Выделяет синюю составляющую из цвета color (целое в диапазоне от 0 до 255).

Функция GetGreen

function GetGreen(color: integer): integer;

Выделяет зеленую составляющую из цвета color (целое в диапазоне от 0 до 255);

Функция GetRed

function GetRed(color: integer): integer;

Выделяет красную составляющую из цвета color (целое в диапазоне от 0 до 255);

Функция GetPixel

function GetPixel(x,y: integer): integer;

Возвращает текущее значение цвета пикселя с координатами (x,y).

Функция RGB

function RGB(r,g,b: integer): integer;

Возвращает код цвета, содержащий красную (Red), зеленую (Green) и синюю (Blue) 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).

Количество возможных цветов равно 256^3 =16777216.

ПРОСТЕЙШИЕ ГРАФИЧЕСКИЕ ОБЪЕКТЫ

Простейшие графические объекты называются *графическими примитивами*.

Процедуры рисования графических примитивов

Имя	Назначение
`SetPixel`	закраска пикселя цветом
`Line`	рисование линии
`Rectangle`	рисование прямоугольника
`Circle`	рисование окружности
`Ellipse`	рисование эллипса

Процедура `Circle`

```
procedure Circle(x,y,r: integer);
```

Рисует окружность с центром в точке `(x,y)` и радиусом `r`.

Процедура `Ellipse`

```
procedure Ellipse(x1,y1,x2,y2: integer);
```

Рисует эллипс, заданный своим описанным прямоугольником с координатами противоположных вершин `(x1,y1)` и `(x2,y2)`.

Процедура `Line`

```
procedure Line(x1,y1,x2,y2: integer);
```

Рисует отрезок с началом в точке `(x1,y1)` и концом в точке `(x2,y2)`.

Процедура `Rectangle`

```
procedure Rectangle(x1,y1,x2,y2: integer);
```

Рисует прямоугольник, заданный координатами противоположных вершин `(x1,y1)` и `(x2,y2)`.

Процедура SetPixel

```
procedure SetPixel(x,y,color: integer);
```

Закрашивает один пиксель с координатами `(x,y)` цветом `color`.

РИСОВАНИЕ ГРАФИЧЕСКИХ ОБЪЕКТОВ

Рисование графических объектов осуществляется *пером* и *кистью*. Линии, ограничивающие объекты, рисуются пером.

Действия с пером

Имя	Назначение
`PenX`	текущая координата X пера
`PenY`	текущая координата Y пера
`SetPenColor`	установка цвета пера
`PenColor`	текущий цвет пера
`MoveTo`	перемещение пера
`LineTo`	рисование отрезка от текущего положения пера
`SetPenWidth`	установка ширины пера
`PenWidth`	текущая ширина пера
`SetPenStyle`	установка стиля пера
`PenStyle`	текущий стиль пера
`SetPenMode`	установка режима пера
`PenMode`	текущий режим пера

Процедура LineTo

```
procedure LineTo(x,y: integer);
```

Рисует отрезок от текущего положения пера до точки `(x,y)`.

Координаты пера при этом также становятся равными `(x,y)`.

Процедура MoveTo

```
procedure MoveTo(x,y: integer);
```

Передвигает невидимое перо к точке с координатами (x,y).

Эта функция работает в паре с функцией LineTo(x,y).

Процедура SetPenColor

```
procedure SetPenColor(color: integer);
```

Устанавливает цвет пера, задаваемый параметром color.

Процедура SetPenMode

```
procedure SetPenMode(m: integer);
```

Устанавливает режим пера, задаваемый параметром m.

Процедура SetPenStyle

```
procedure SetPenStyle(ps: integer);
```

Устанавливает стиль пера, задаваемый параметром ps.

Процедура SetPenWidth

```
procedure SetPenWidth(w: integer);
```

Устанавливает ширину пера, равную w пикселям.

Функция PenColor

```
function PenColor: integer;
```

Возвращает текущий цвет пера.

Функция PenMode

```
function PenMode: integer;
```

Возвращает текущий режим пера.

Функция PenStyle

```
function PenStyle: integer;
```

Возвращает текущий стиль пера.

Функция PenWidth

```
function PenWidth: integer;
```

Возвращает текущую ширину пера.

Функция PenX

```
function PenX: integer;
```

Возвращает текущую координату X пера.

Функция PenY

```
function PenY: integer;
```

Возвращает текущую координату Y пера.

Стили пера задаются константами, приведенными в таблице 2.

Таблица 2 Стили пера

Код	Имя	Линия
0	psSolid	————————————
1	psDash	– – – – – – – – – – – – – – – – – – – –
2	psDot	----------
3	psDashDot	–·–·–·–·–·–·–·–·–·–·–
4	psDashDotDot	–··–··–··–··–··–··–··–··–
5	psClear	невидимая линия

По умолчанию используется стиль psSolid . Штриховые стили устанавливаются только для пера шириной 1.

Режим пера определяет, как цвет пера взаимодействует с цветом поверхности. В **GraphABC** два режима пера:

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.

Внутренность объекта может закрашиваться кистью или заданным цветом процедурой FloodFill :

```
procedure FloodFill (x,y: integer; color: integer);
```

Точка (x,y) – любая точка внутри ограниченной области.

Закрашивает одноцветную область цветом color, начиная с точки (x,y).

Если область незамкнута, то заливка “разольется” по графическому окну.

Действия с кистью

Имя	Назначение
SetBrushColor	установка цвета кисти
BrushColor	текущий цвет кисти
FillRect	заливка прямоугольника

По умолчанию кисть имеет белый цвет.

Процедура SetBrushColor

```
procedure SetBrushColor(color: integer);
```

Устанавливает цвет кисти, задаваемый параметром color.

Функция BrushColor

```
function BrushColor: integer;
```

Возвращает текущий цвет кисти.

Процедура FillRect

```
procedure FillRect (x1,y1,x2,y2: integer);
```

Закрашивает прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2), текущим цветом кисти.

РАБОТА С ТЕКСТОМ

Вывод текста в графическое окно осуществляется процедурой `TextOut`:

```
procedure TextOut(x,y: integer; S: string);
```

Точка `(x,y)` задает верхний левый угол прямоугольника, который будет содержать текст из строки `S`.

Выводит текст (строку `S`) с позиции `(x,y)`.

Процедура `TextOut` меняет текущие координаты `(x,y)`.

Действия со шрифтом

Имя	Назначение
`SetFontColor`	установка цвета шрифта
`FontColor`	текущий цвет шрифта
`SetFontSize`	установка размера шрифта в пунктах
`FontSize`	текущий размер шрифта в пунктах
`SetFontName`	установка наименования шрифта
`FontName`	текущее наименование шрифта
`SetFontStyle`	установка стиля шрифта
`SetFontSize`	текущий стиль шрифта
`TextWidth`	текущая ширина строки
`TextHeight`	текущая высота строки

Процедура `SetFontColor`

```
procedure SetFontColor(color: integer);
```

Устанавливает цвет шрифта.

Процедура SetFontName

```
procedure SetFontName(name: string);
```

Устанавливает наименование шрифта.

По умолчанию установлен шрифт MS Sans Serif.

Наиболее распространенные шрифты: Times, Arial и Courier.

Наименование шрифта можно набирать без учета регистра.

Процедура SetFontSize

```
procedure SetFontSize(sz: integer);
```

Устанавливает размер шрифта в пунктах (по умолчанию 8 пунктов; один пункт приблизительно равен 0,3 мм).

Процедура SetFontStyle

```
procedure SetFontStyle(fs: integer);
```

Устанавливает стиль шрифта.

Функция FontColor

```
function FontColor: integer;
```

Возвращает текущий цвет шрифта (по умолчанию черный).

Функция FontName

```
function FontName: string;
```

Возвращает текущее наименование шрифта.

Функция FontSize

```
function FontSize: integer;
```

Возвращает текущий размер шрифта в пунктах.

Функция FontStyle

```
function FontStyle: integer;
```

Возвращает текущий стиль шрифта.

Функция TextHeight

```
function TextHeight(S: string): integer;
```

Возвращает высоту строки `S` в пикселях при текущих настройках шрифта.

Стили шрифта задаются константами, приведенными в таблице 3.

Функция TextWidth

```
function TextWidth(S: string): integer;
```

Возвращает ширину строки `S` в пикселях при текущих настройках шрифта.

Таблица 3 Стили шрифта

Код	Имя	Название
0	`fsNormal`	обычный
1	`fsBold`	жирный
2	`fsItalic`	наклонный
3	`fsBoldItalic`	жирный наклонный
4	`fsUnderline`	подчеркнутый
5	`fsBoldUnderline`	жирный подчеркнутый
6	`fsItalicUnderline`	наклонный подчеркнутый
7	`fsBoldItalicUnderline`	жирный наклонный подчеркнутый

УПРАВЛЕНИЕ ГРАФИЧЕСКИМ ОКНОМ

Часть процедур и функций, позволяющих управлять графическим окном, приведена ниже. За более подробной информацией следует обратиться к справке по **PascalABC**.

Все размеры устанавливаются и возвращаются в пикселях.

Действия с графическим окном

`SetWindowCaption`	установка заголовка графического окна
`WindowCaption`	заголовок графического окна
`ClearWindow`	очистка графического окна
`SetWindowWidth`	установка ширины графического окна
`WindowWidth`	ширина графического окна
`SetWindowHeight`	установка высоты графического окна
`WindowHeight`	высота графического окна
`SetWindowSize`	установка ширины и высоты графического окна
`SetWindowLeft`	установка отступа графического окна от левого края экрана
`WindowLeft`	отступ графического окна от левого края экрана
`SetWindowTop`	установка отступа графического окна от верхнего края экрана
`WindowTop`	отступ графического окна от верхнего края экрана
`SetWindowPos`	установка отступа графического окна от от левого и верхнего края экрана
`SaveWindow`	сохранение содержимого графического окна

Процедура `ClearWindow`

`procedure` `ClearWindow;`

Очищает графическое окно белым цветом.

`procedure` `ClearWindow(color: integer);`

Закрашивает графическое окно цветом `color`.

Процедура SetWindowCaption

```
procedure SetWindowCaption(s: string);
```

Устанавливает заголовок графического окна.

Процедура SetWindowHeight

```
procedure SetWindowHeight(h: integer);
```

Устанавливает высоту графического окна.

Процедура SetWindowLeft

```
procedure SetWindowLeft(l: integer);
```

Устанавливает отступ графического окна от левого края экрана.

Процедура SetWindowPos

```
procedure SetWindowPos(l,t: integer);
```

Устанавливает отступ графического окна от левого и верхнего края экрана.

Процедура SetWindowSize

```
procedure SetWindowSize(w,h: integer);
```

Устанавливает ширину и высоту графического окна.

Процедура SetWindowTop

```
procedure SetWindowTop(t: integer);
```

Устанавливает отступ графического окна от верхнего края экрана.

Процедура SetWindowWidth

```
procedure SetWindowWidth(w: integer);
```

Устанавливает ширину графического окна.

Процедура SaveWindow

```
procedure SaveWindow (fname: string);
```

Сохраняет содержимое графического окна в файле формата bmp с именем `fname`.

Функция `WindowCaption`

```
function WindowCaption: string;
```

Возвращает заголовок графического окна.

Функция `WindowHeight`

```
function WindowHeight: integer;
```

Возвращает высоту графического окна.

Функция `WindowLeft`

```
function WindowLeft: integer;
```

Возвращает отступ графического окна от левого края экрана.

Функция `WindowTop`

```
function WindowTop: integer;
```

Возвращает отступ графического окна от верхнего края экрана.

Функция `WindowWidth`

```
function WindowWidth: integer;
```

Возвращает ширину графического окна.

ЛИТЕРАТУРА

1 Абрамов С.А. Начала информатики / С.А. Абрамов, Е.В. Зима – М.: Наука, 1989. –256 с.

2 Абрамян М.Э. Основы программирования на языке Паскаль: Скалярные типы данных, управляющие операторы, процедуры и функции / М.Э. Абрамян, С.С. Михалкович – Ростов–на–Дону: ООО "ЦИИР", 2004. –198 с.

3 Задачи по программированию / С.А. Абрамов и др. – М.: Наука, 1988. – 224 с.

4 Задачи по программированию / Н.И. Амелина и др. – М.: Вузовская книга, 2000. – 104 с.

5 Методы программирования. Учебное пособие / Н.И. Минакова, Е.С. Невская, Г.А. Угольницкий, А.А. Чекулаева, М.И. Чердынцева. – М.: Вузовская книга, 1999. – 280 с.

6 Фаронов В.В. Турбо Паскаль 7.0. Начальный курс: учебное пособие. – М.: КНОРУС, 2006. – 576 с.