**Построение графических рисунков с использование процедур и анимации**

***Рисунок «Деревня»***

Есть программа, рисующая домик и дерево в абсолютных координатах (т.е. построение ведется в конкретных координатах точек, изображение нельзя подвинуть). Вот она:

**Program** domik;

**uses** GraphABC;

**begin**

SetWindowSize(500, 500);

{Коричневая стена}

SetPenWidth(2);

SetBrushColor(clBrown);

FillRect(100,50,150,100);

{Крыша желтого цвета}

SetPenColor(clBrown);

Line(125,25,80,75);

Line(125,25,170,75);

FloodFill(125, 30,clYellow);

{Крона дерева}

SetPenColor(clGreen);

SetBrushColor(clGreen);

Ellipse(100,150,150,200);

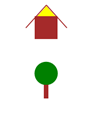
{ствол дерева}

SetBrushColor(clBrown);

FillRect(120,200,130,230);

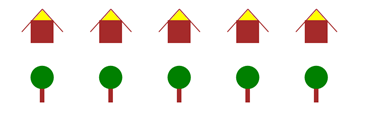
**end**.

На экране это выглядит так (проверьте сами):



**Задание для желающих (см. подсказки после рисунка):**

*Нарисовать “поселок”, который выстроился вдоль горизонтальной оси экрана – оси X (см. рисунок).*



*Подсказки:*

1. Рисунок одного домика и дерева надо оформить как процедуру, пересчитав построение дерева и домика теперь в относительных координатах). Например, вот так:  
   **procedure** domik(x:integer);

**begin**

SetPenWidth(2);

SetBrushColor(clBrown);

FillRect(x,50,x+50,100);

SetPenColor(clBrown);

Line(x+25,25,x-20,75);

Line(x+25,25,x+70,75);

FloodFill(x+25, 30,clYellow);

SetPenColor(clGreen);

SetBrushColor(clGreen);

Ellipse(x,150,x+50,200);

SetBrushColor(clBrown);

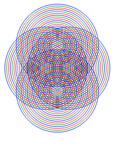
FillRect(x+20,200,x+30,230);

**end**;

1. построение рисунка начинается с левого верхнего угла стены первого дома. Координата Y не изменяется. Чтобы начать рисовать второй и следующий домик, нужно координату X увеличить на 150 (50 точек – ширина первого дома и 100 точек – расстояние между домиками).

***Рисунок «Красивый узор»***

Напишем программу, которая рисует красивые графические узоры из концентрических окружностей. Вот такой узор:



Концентрические окружности имеют общий центр и разные радиусы.  
Будем использовать **процедуру uzor** для рисования концентрических окружностей с центром, в качестве параметров в процедуру будем передавать центр концентрических окружностей **x y**, количество окружностей **n** и шаг изменения радиуса **v**. В цикле for  будем рисовать заданное число окружностей, при этом радиус окружностей будет постоянно увеличиваться (**r:=r+v;**)  
Чтобы цвет окружностей чередовался, будем использовать оператор выбора **case** и в зависимости от остатка от деления на 3 (число цветов можно увеличить) будет устанавливаться соответствующий цвет. Для определения остатка от деления используется  команда **mod**. Отличие процедуры **DrawCircle** от **Circle** в том, что первая рисует только линию окружности, не заполняя внутренней части. То есть, попросту говоря – окружность пустая внутри.

**procedure** uzor(x, y, n, v : integer);

**var** i,r:integer;

**begin**

r:=10;

**for** i:=1 **to** n **do begin**

r:=r+v;

SetPenColor(clRed);

**case** i **mod** 3 **of**

0 : SetPenColor(clRed);

1 : SetPenColor(clGreen);

2 : SetPenColor(clBlue);

**end**;

drawcircle (x,y,r);

**end**;

**end**;

Теперь в теле основной программы мы можем вызывать процедуру **uzor**  с различными значениями параметров. Если изменять координаты **x** и **y**, то можно красиво наложить узоры из концентрических окружностей. Приведенный выше рисунок получается вот такой программой:

**program** uzor\_1;

**uses** GraphABC;

**procedure** uzor(x, y, n, v : integer);

**var** i,r:integer;

**begin**

r:=10;

**for** i:=1 **to** n **do begin**

r:=r+v;

SetPenColor(clRed);

**case** i **mod** 3 **of**

0 : SetPenColor(clRed);

1 : SetPenColor(clGreen);

2 : SetPenColor(clBlue);

**end**;

drawcircle (x,y,r);

**end**;

**end**;

**begin**

setWindowSize(500, 500);

uzor(100,150,20,4);

uzor(150,150,20,4);

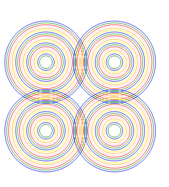
uzor(125,100,20,4);

uzor(125,200,20,4);

**end**.

**Задание для желающих:**

*Нарисовать узор (см. образец) с числом цветов 6:*



***Рисунок с анимацией «Столкновение шаров»***

Вспомним, как создается простейшая анимация (команда ClearWindow стирает предыдущее положение шара, команда Sleep(1) организует задержку перед стиранием):

**program** krug;

**uses** GraphABC;

**var** I,X,Y,D: integer;

**begin**

X:=20; Y:=30; D:=100;

**for** i:=1 **to** 500 **do**

**begin**

ClearWindow;

SetBrushColor(clGreen);

Ellipse(X+I,Y,X+I+D,Y+D);

Sleep(1);

**end**;

**end**.

Усовершенствуем программу, удалив мерцание при перерисовке. Напомню, что команда LockDrawing блокирует вывод в графическое окно, осуществляя рисование только во внеэкранном буфере, а Redraw перерисовывает готовое изображение из буфера в графическое окно (проверьте сами).

**program** krug;

**uses** GraphABC;

**var** I,X,Y,D: integer;

**begin**

X:=20; Y:=30; D:=100;

LockDrawing;

**for** i:=1 **to** 500 **do**

**begin**

ClearWindow;

SetBrushColor(clGreen);

Ellipse(X+I,Y,X+I+D,Y+D);

Redraw;

Sleep(1);

**end**;

**end**.

И вот программа «Столкновение шаров» (один шар движется, а другой – в покое). Проверьте ее работу, если интересно.

**program** dva\_shara;

**uses** GraphABC;

**var** I,X,Y,D,X1,X2: integer;

**begin**

X:=20;

Y:=30;

D:=70;

X1:=350;

X2:=500;

LockDrawing;

**for** I:=1 **to** X1-D **do**

// зеленый шар движется, красный стоит на месте

**begin**

ClearWindow;

SetBrushColor(clGreen);

Ellipse(X+I,Y,X+I+D,Y+D);

SetBrushColor(clRed);

Ellipse(X+X1,Y,X+X1+D,Y+D);

Redraw;

Sleep(5);

**end**;

**for** I:=X1 **to** X2 **do**

// зеленый шар стоит на месте, красный шар движется

**begin**

ClearWindow;

SetBrushColor(clGreen);

Ellipse(X+X1-D,Y,X+X1,Y+D);

SetBrushColor(clRed);

Ellipse(X+I,Y,X+I+D,Y+D);

Redraw;

Sleep(5);

**end**;

**end**.

**Задание для желающих:**

*На основе разобранной программы, создайте программу, которая выводит на экран два шара: красный шар стоит на месте, а зеленый движется по горизонтали к нему навстречу. После столкновения зеленый шар двигается в обратном направлении, а красный шар начинает падать вниз (движение по вертикали).*