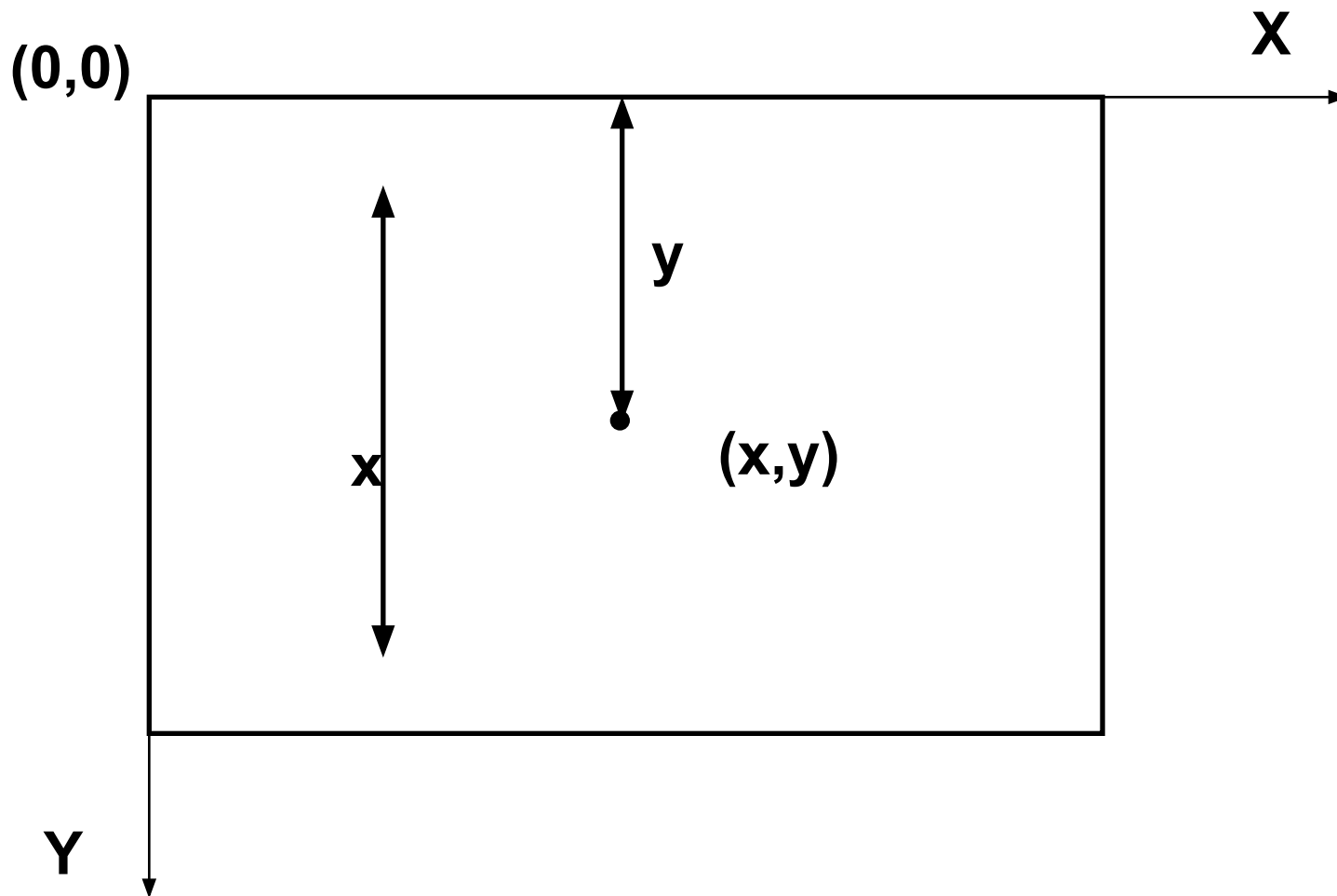


# Программирование на языке Паскаль

## Тема 7. Графика

# Система координат

---



# Управление цветом

---

## Цвет и толщина линий, цвет точек:

```
Pen ( 1, 255, 0, 0 );
```

толщина линии	R(red) 0..255	G(green) 0..255	B(blue) 0..255
------------------	------------------	--------------------	-------------------

## Цвет и стиль заливки:

```
Brush ( 1, 0, 255, 0 );
```

0 – ВЫКЛЮЧИТЬ 1 - ВКЛЮЧИТЬ	R	G	B
-------------------------------	---	---	---

## Цвет текста:


```
TextColor ( 0, 0, 255 );
```

R	G	B
---	---	---

# Точки, отрезки и ломаные

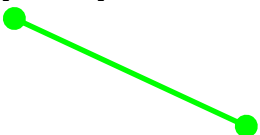
---

$(x, y)$



```
Pen (1, 0, 0, 255);  
Point (x, y);
```

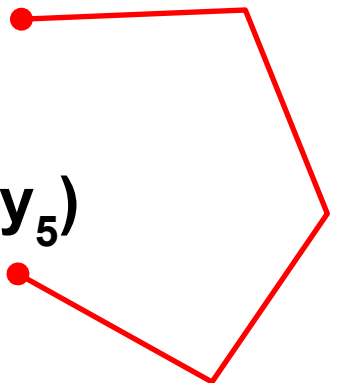
$(x_1, y_1)$



$(x_2, y_2)$

```
Pen (1, 0, 255, 0);  
Line (x1, y1, x2, y2);
```

$(x_1, y_1)$



$(x_2, y_2)$

$(x_3, y_3)$

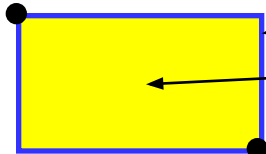
$(x_4, y_4)$

$(x_5, y_5)$

```
Pen (1, 255, 0, 0);  
MoveTo (x1, y1);  
LineTo (x2, y2);  
LineTo (x3, y3);  
LineTo (x4, y4);  
LineTo (x5, y5);
```

# Фигуры с заливкой

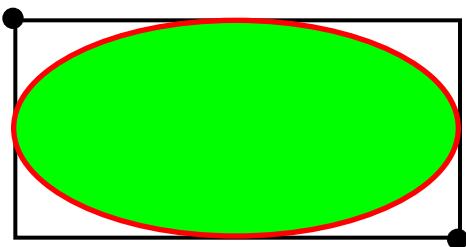
$(x_1, y_1)$



$(x_2, y_2)$

```
Pen (1, 0, 0, 255);  
Brush (1, 255, 255, 0);  
Rectangle (x1, y1, x2, y2);
```

$(x_1, y_1)$

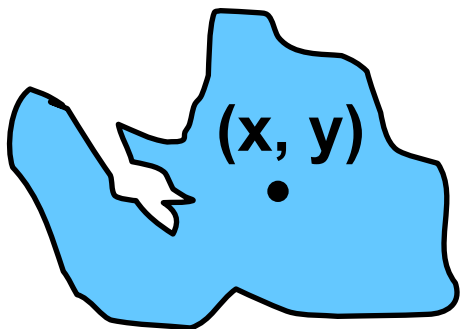


$(x_2, y_2)$

```
Pen (1, 255, 0, 0);  
Brush (1, 0, 255, 0);  
Ellipse (x1, y1, x2, y2);
```



Как отменить заливку?



```
Brush (1, 100, 200, 255);  
Fill (x, y);
```

# Текст

---



```
TextColor (0, 0, 255);  
Brush (1, 255, 255, 0);  
Font (20, 30, 600);
```

размер  
10 пикселей

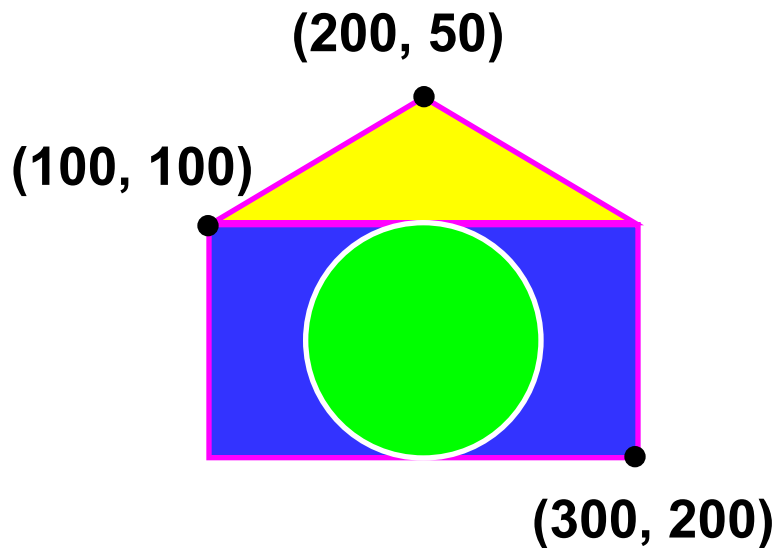
угол  
поворота

насыщенность:  
400 – нормальный  
600 – жирный

```
MoveTo (x, y);  
writeln ('Привет!');
```

# Пример

---

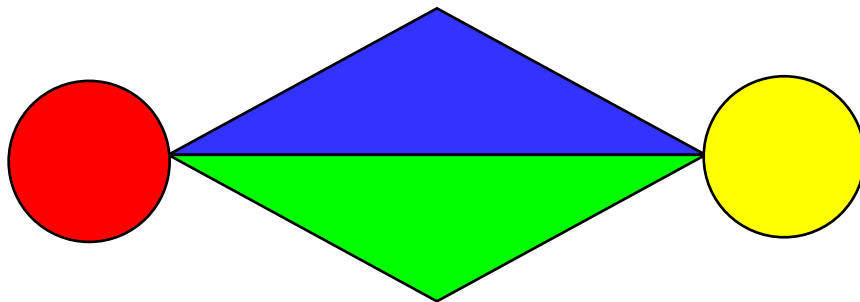


```
program qq;  
begin  
    Pen(2, 255, 0, 255);  
    Brush(1, 0, 0, 255);  
    Rectangle(100, 100, 300, 200);  
    MoveTo(100, 100);  
    LineTo(200, 50);  
    LineTo(300, 100);  
    Brush(1, 255, 255, 0);  
    Fill(200, 75);  
    Pen(2, 255, 255, 255);  
    Brush(1, 0, 255, 0);  
    Ellipse(150, 100, 250, 200);  
end.
```

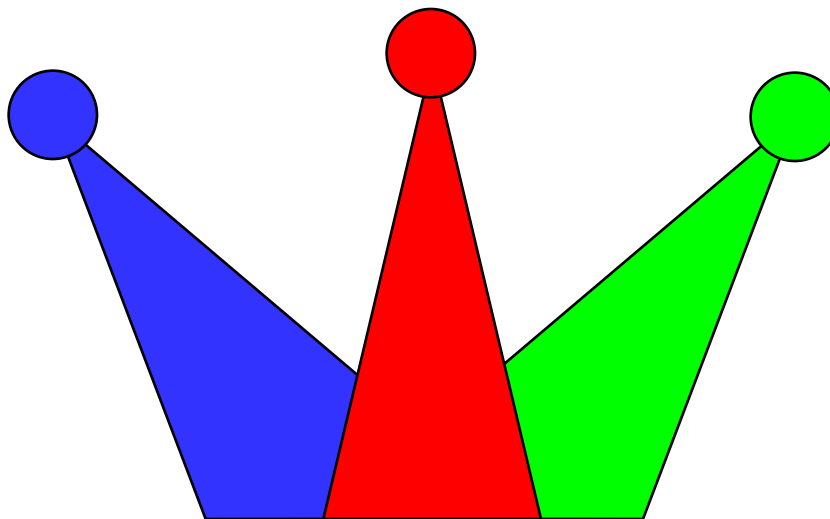
# Задания

---

"4": "Лягушка"

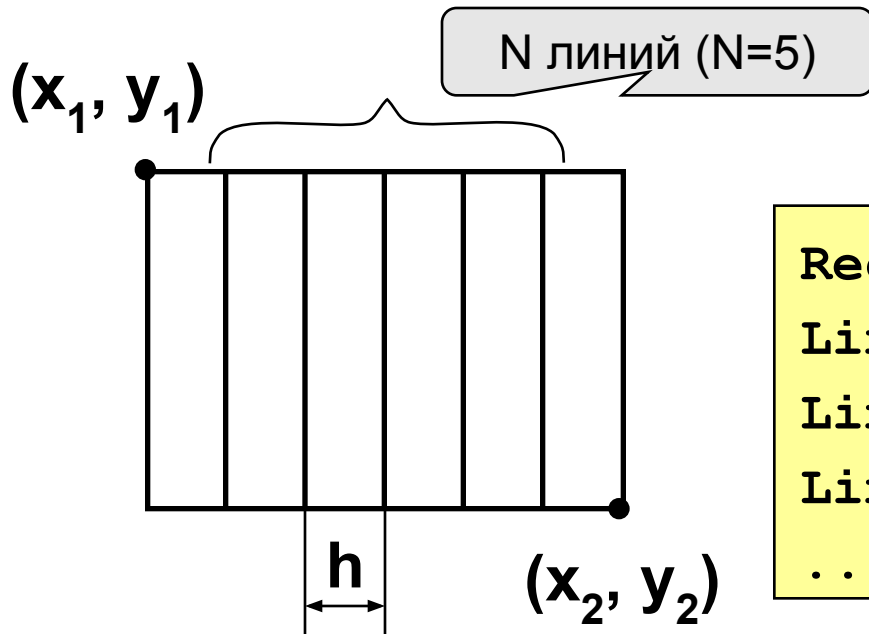


"5": "Корона"





# Штриховка



$$h = \frac{x_2 - x_1}{N + 1}$$

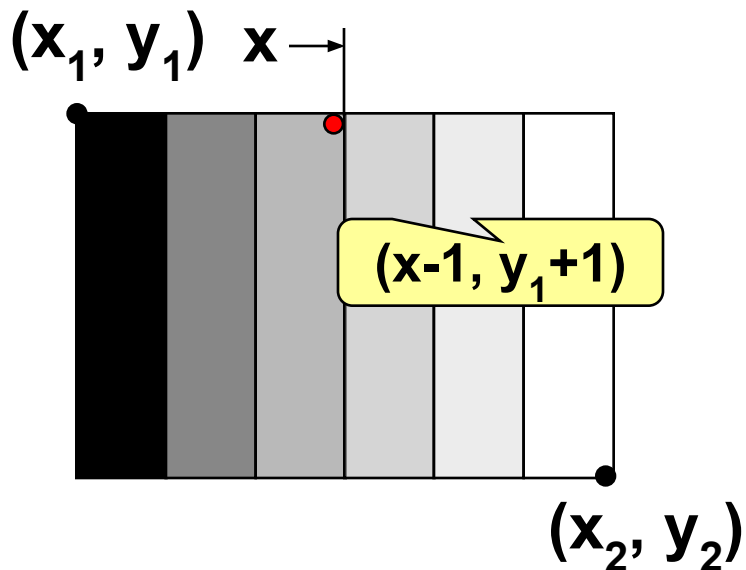
```
Rectangle (x1, y1, x2, y2);  
Line (x1+h, y1, x1+h, y2);  
Line (x1+2*h, y1, x1+2*h, y2);  
Line (x1+3*h, y1, x1+3*h, y2);  
...
```

```
h := (x2 - x1) / (N + 1);  
Rectangle (x1, y1, x2, y2);  
x := x1 + h;  
for i:=1 to N do begin  
  Line( round(x), y1, round(x), y2);  
  x := x + h;  
end;
```

~~var x, h: real;~~

округление до  
ближайшего целого

# Как менять цвет?



серый:  $R = G = B$

```
Brush ( 1, c, c, c );  
Fill ( ???, ??? );
```

Шаг изменения  $c$ :

$$h_c = \frac{255}{N+1}$$

```
hc := 255 div (N + 1);
```

```
c := 0;
```

```
for i:=1 to N+1 do begin
```

```
  Line (round(x), y1, round(x), y2);
```

```
  Brush (1, c, c, c);
```

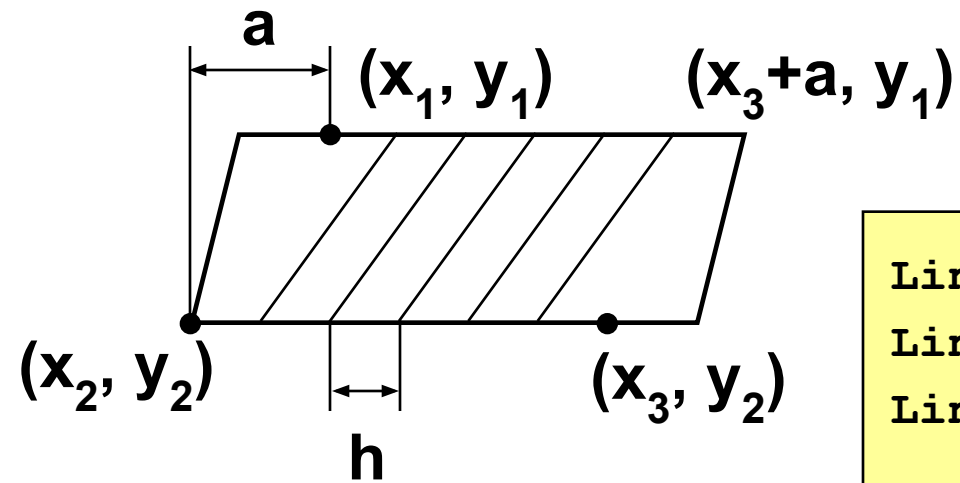
```
  Fill (round(x)-1, y1+1);
```

```
  x := x + h; c := c + hc;
```

```
end;
```

```
var c, hc: integer;
```

# Штриховка



$$a = x_2 - x_1$$

$$h = \frac{x_3 - x_2}{N + 1}$$

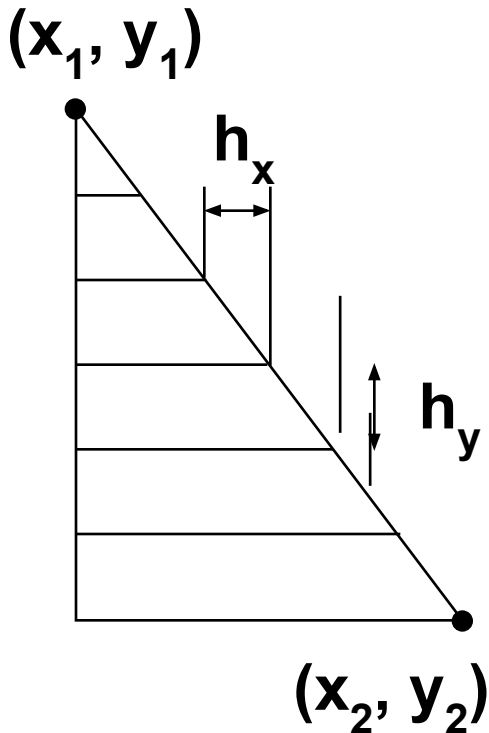
```
Line( x1+h, y1, x1+h-a, y2 );  
Line( x1+2*h, y1, x1+2*h-a, y2 );  
Line( x1+3*h, y1, x1+3*h-a, y2 );  
...
```

x

x-a

```
h := (x3 - x2) / (N + 1);  
a := x2 - x1;  
x := x1 + h;  
for i:=1 to N do begin  
  Line( round(x), y1, round(x-a), y2 );  
  x := x + h;  
end;
```

# Штриховка



$$h_x = \frac{x_2 - x_1}{N + 1}$$

$$h_y = \frac{y_2 - y_1}{N + 1}$$

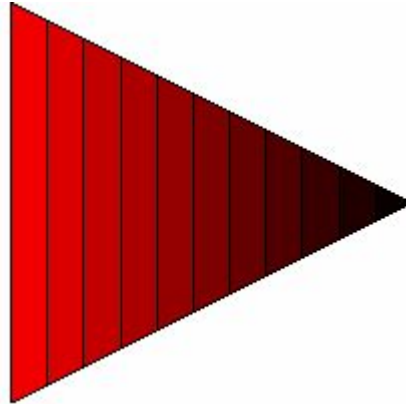
```
Line( x1, y1+hy, x1+hx, y1+hy)
;
Line( x1, y1+2*hy, x1+2*hx,
y1+2*hy);
Line( x1, y1+3*hy, x1+3*hx,
y1+3*hy);
```

```
hx := (x2 - x1) / (N + 1);
hy := (y2 - y1) / (N + 1);
x := x1 + hx; y := y1 + hy;
for i:=1 to N do begin
  Line( x1, round(y), round(x), round(y));
  x := x + hx; y := y + hy;
end;
```

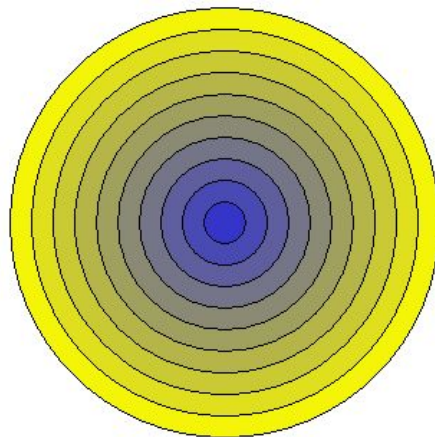
# Задания

---

**"4"**: Ввести с клавиатуры число линий штриховки и построить фигуру, залив все области разным цветом.



**"5"**: Ввести с клавиатуры число окружностей и построить фигуру, залив все области разным цветом.



# Программирование на языке Паскаль

## Тема 8. Графики функций

# Построение графиков функций

---

**Задача:** построить график функции  $y = 3 \sin(x)$  на интервале от 0 до  $2\pi$ .

**Анализ:**

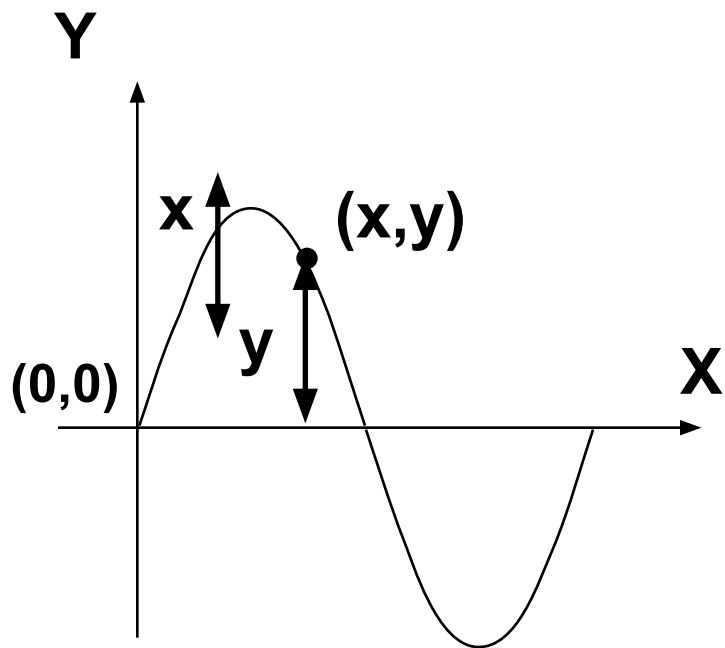
максимальное значение  $y_{\max} = 3$  при  $x = \pi/2$

минимальное значение  $y_{\min} = -3$  при  $x = 3\pi/2$

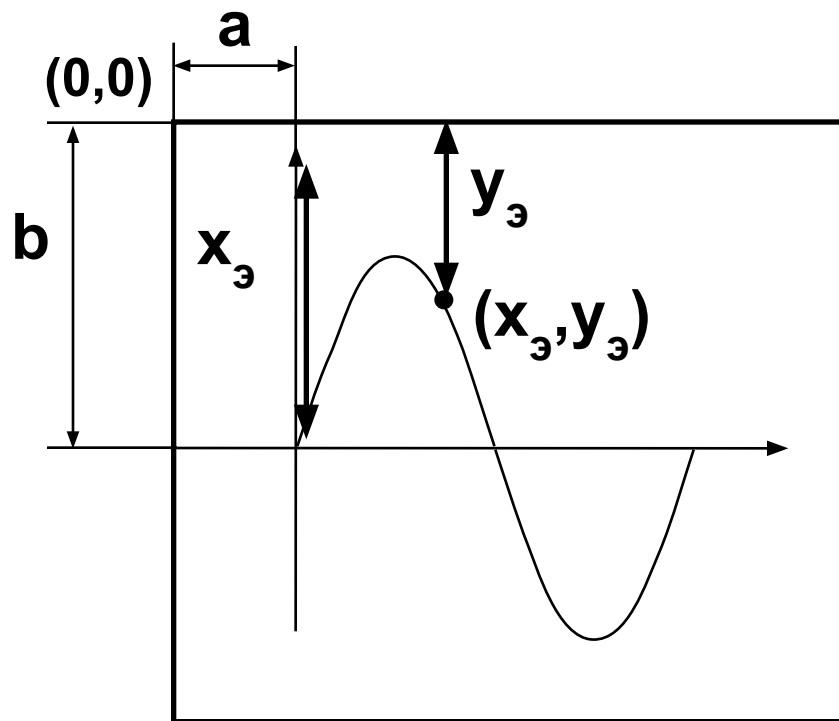
**Проблема:** функция задана в математической системе координат, строить надо на экране, указывая координаты в пикселях.

# Преобразование координат

Математическая  
система координат



Экранная система  
координат (пиксели)



**k** — масштаб (длина  
изображения единичного  
отрезка на экране)

$$x_{\text{э}} = a + kx$$

$$y_{\text{э}} = b - ky$$



# Программа

```
program qq;  
const a = 50; b = 200; k = 50;  
      xmin = 0; xmax = 6.2832;  
var x, y, h: real;  
     xe, ye, w: integer;  
begin  
  w := round((xmax - xmin)*k);  
  Line(a-10, b, a+w, b);  
  Line(a, 0, a, 2*b);  
  x := xmin; h := 0.05;  
  while x <= xmax do begin  
    y := 3*sin(x);  
    xe := a + round(k*x);  
    ye := b - round(k*y);  
    Point (xe, ye);  
    x := x + h;  
  end;  
end.
```

2п

h — шаг изменения x

w — длина оси OX в пикселях

оси координат

на экране

цикл  
построения  
графика



Что плохо?

# Как соединить точки?

## Алгоритм:

Если первая точка  
перейти в точку  $(x_э, y_э)$   
иначе  
отрезок в точку  $(x_э, y_э)$

выбор  
варианта  
действий

## Программа:

```
var first: boolean;  
...  
begin  
...  
first := True;  
while x <= xmax do begin  
...  
if first then begin  
    MoveTo(xe, ye);  
    first := False;  
end  
else LineTo(xe, ye);  
...  
end;  
end.
```

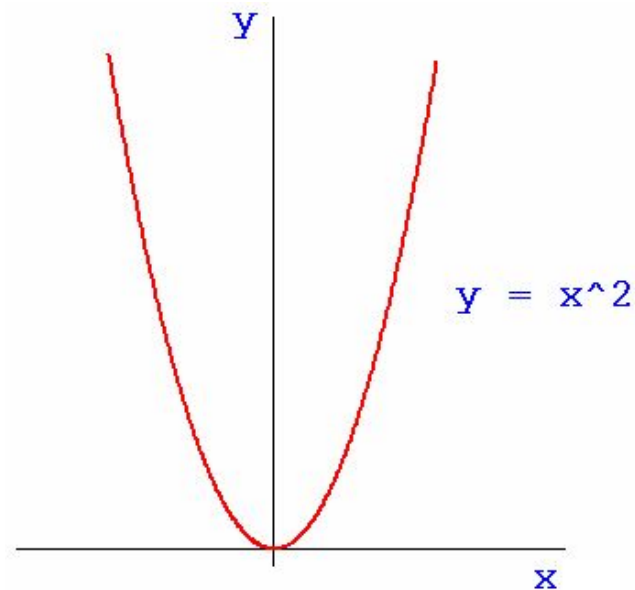
логическая  
переменная

начальное значение

# Задания

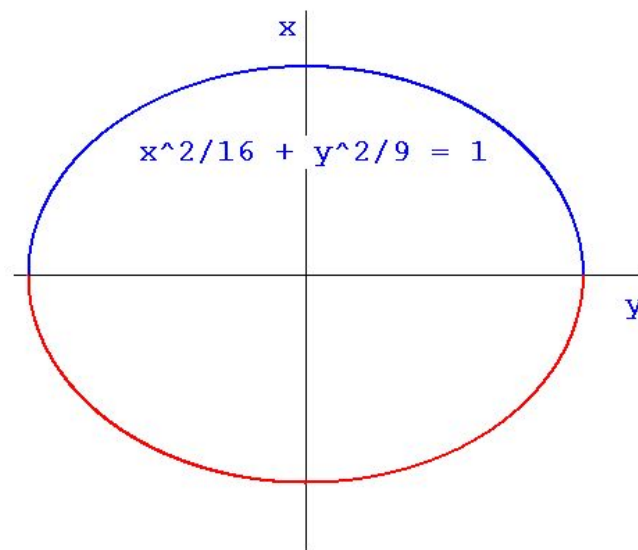
---

"4": Построить график функции  $y = x^2$  на интервале  $[-3,3]$ .



"5": Построить график функции (эллипс)

$$\frac{x^2}{16} + \frac{y^2}{9} = 1$$



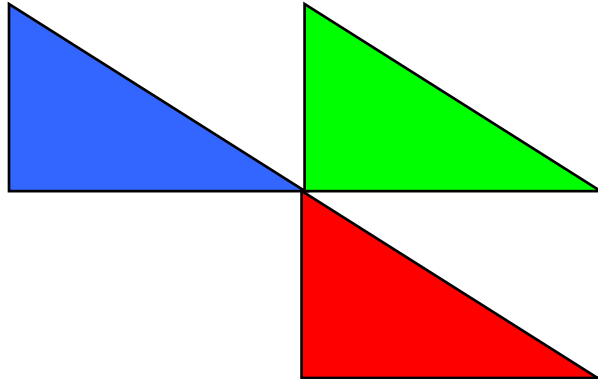
# Программирование на языке Паскаль

## Тема 9. Процедуры

# Процедуры

---

**Задача:** Построить фигуру:



**Можно ли решить известными методами?**

**Общность:** три похожие фигуры.

**общее:** размеры, угол поворота

**отличия:** координаты, цвет



**Сколько координат надо задать?**

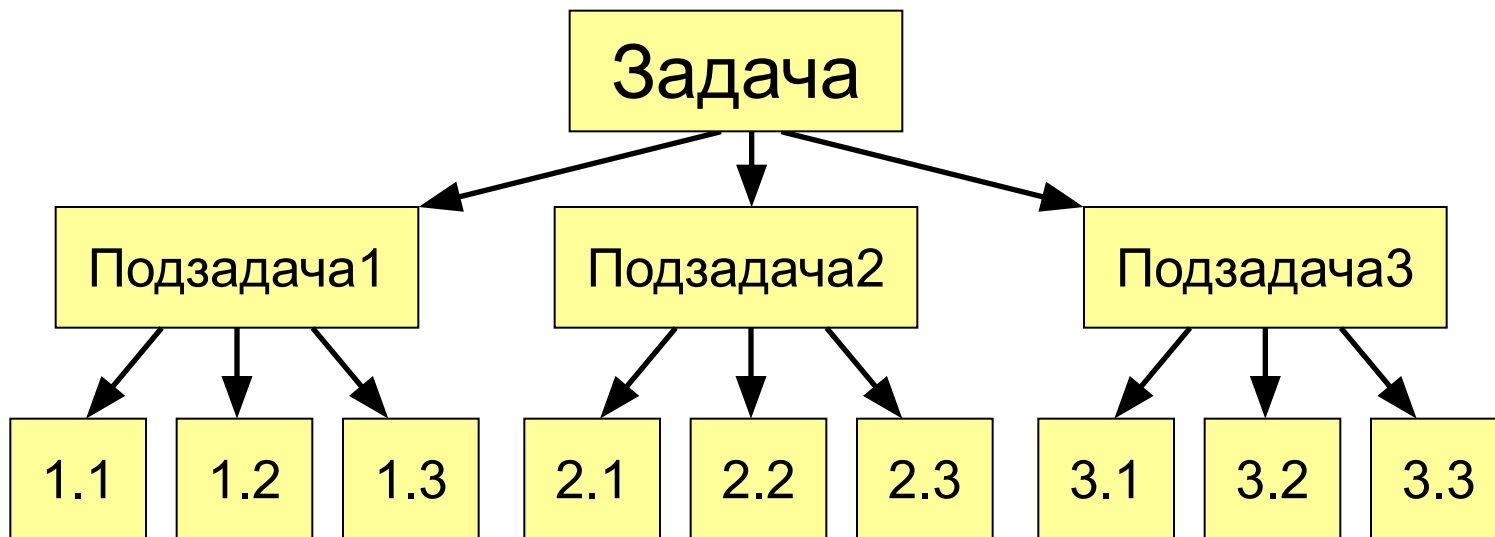
# Процедуры

---

**Процедура** – это вспомогательный алгоритм, который предназначен для выполнения некоторых действий.

## Применение:

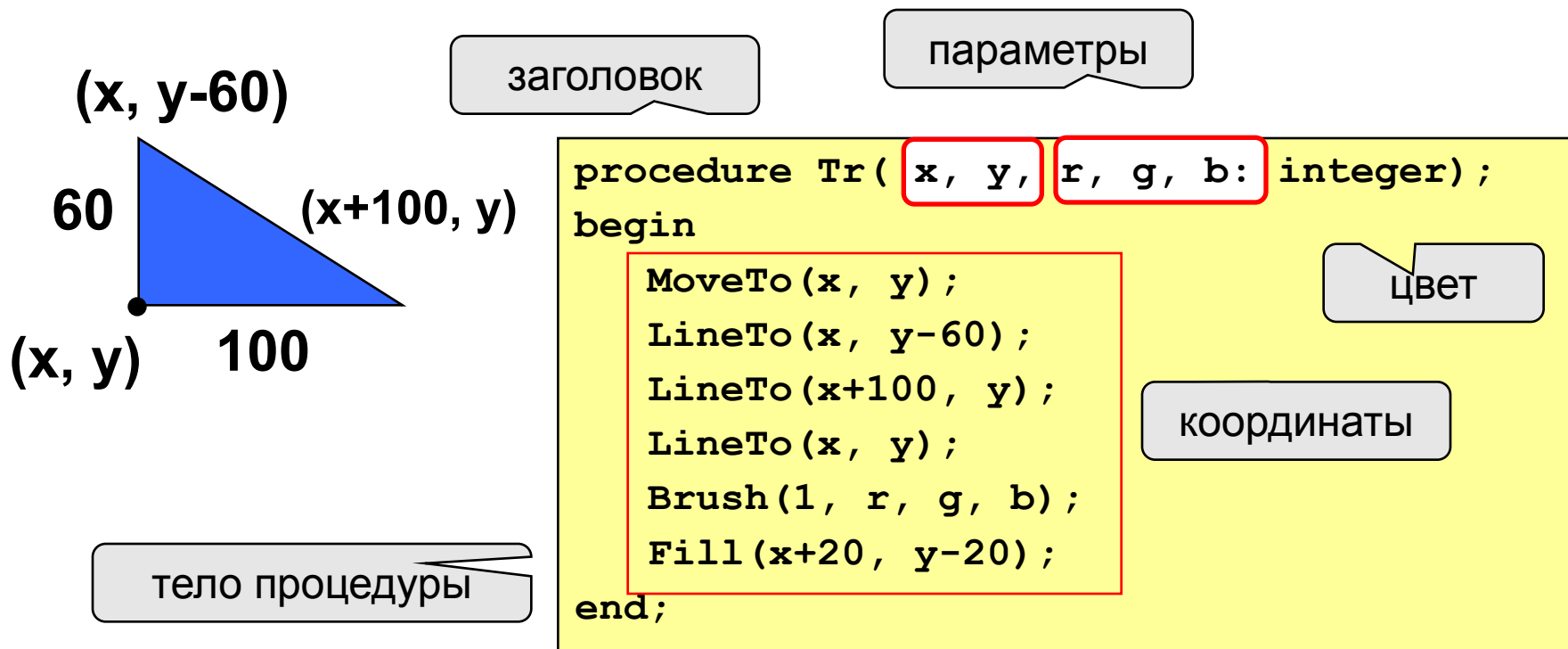
- выполнение одинаковых действий в разных местах программы
- разбивка программы (или другой процедуры) на подзадачи для лучшего восприятия



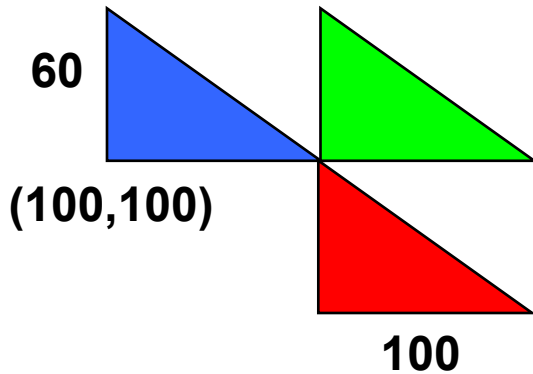
# Процедуры

## Порядок разработки:

- выделить одинаковые или похожие действия (три фигуры)
- найти в них **общее** (размеры, форма, угол поворота) и **отличия** (координаты, цвет)
- отличия записать в виде неизвестных переменных, они будут **параметрами** процедуры



# Программа



ВЫЗОВЫ  
процедуры

формальные параметры

```
program qq;  
  procedure Tr( x, y, r, g, b: integer);  
  begin  
    ...  
  end;  
begin  
  Pen(1, 255, 0, 255);  
  Tr(100, 100, 0, 0, 255);  
  Tr(200, 100, 0, 255, 0);  
  Tr(200, 160, 255, 0, 0);  
end.
```

процедура

фактические параметры



# Процедуры

---

## Особенности:

- все процедуры расположены **выше** основной программы
- в заголовке процедуры перечисляются **формальные** параметры, они обозначаются именами, поскольку могут меняться

```
procedure Tr( x, y, r, g, b: integer);
```

- при вызове процедуры в скобках указывают **фактические** параметры (числа или арифметические выражения) **в том же порядке**

```
Tr (200, 100, 0, 255, 0);
```

x

y

r

g

b

# Процедуры

---

## Особенности:

- для каждого формального параметра после двоеточия указывают его тип

```
procedure A (x: real; y: integer; z: real);
```

- если однотипные параметры стоят рядом, их перечисляют через запятую

```
procedure A (x, z: real; y, k, l: integer);
```

- внутри процедуры параметры используются так же, как и переменные

# Процедуры

---

## Особенности:

- в процедуре можно объявлять дополнительные **локальные** переменные, остальные процедуры не имеют к ним доступа

```
program qq;  
  procedure A(x, y: integer);  
    var a, b: real;  
  begin  
    a := (x + y) / 6;  
    ...  
  end;  
begin  
  ...  
end.
```

локальные  
переменные

# Параметры-переменные

**Задача:** составить процедуру, которая меняет местами значения двух переменных.

**Особенности:**

надо, чтобы изменения, сделанные в процедуре, стали известны вызывающей программе

```
program qq;
var x, y: integer;

procedure Exchange ( a, b: integer );
var c: integer;
begin
  c := a; a := b; b := c;
end;

begin
  x := 1; y := 2;
  Exchange ( x, y );
  writeln ( 'x = ', x, ' y = ', y );
end;
```

эта процедура  
работает с  
**КОПИЯМИ**  
параметров

**x = 1 y = 2**

# Параметры-переменные

параметры могут изменяться

```
procedure Exchange ( var a, b: integer );  
var c: integer;  
begin  
  c := a; a := b; b := c;  
end;
```

## Применение:

таким образом процедура (и функция) может возвращать несколько значений,

## Запрещенные варианты вызова

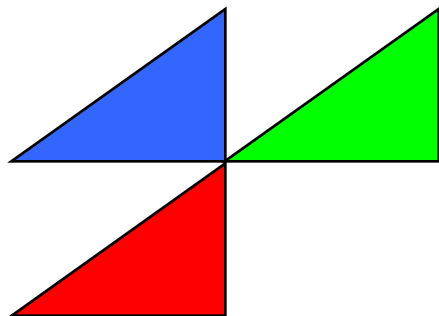
Exchange ( ~~2~~, ~~3~~ );      { числа }

Exchange ( ~~x+z~~, ~~y+2~~ );      { выражения }

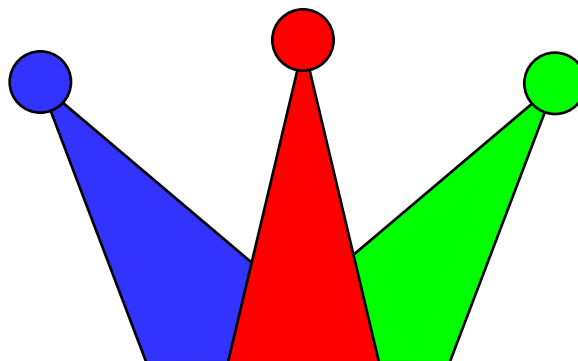
# Задания

---

"4": Используя процедуры, построить фигуру.



"5": Используя процедуры, построить фигуру.



# Программирование на языке Паскаль

## Тема 10. Рекурсия

# Рекурсивные объекты

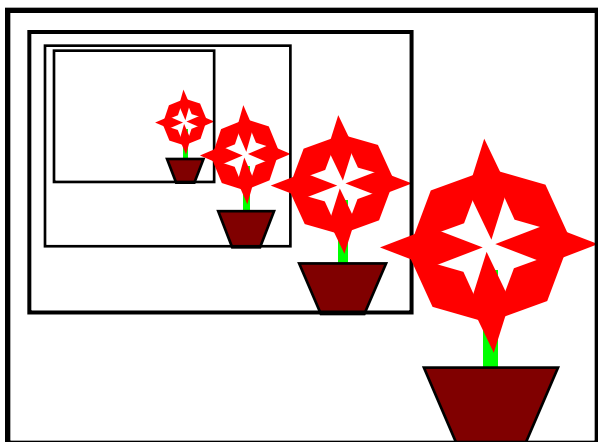
## Примеры:

### Сказка о попе и собаке:

У попа была собака, он ее любил.  
Она съела кусок мяса, он ее убил.  
В ямку закопал, надпись написал:

**Сказка о попе и собаке**

### Рисунок с рекурсией:



### Факториал:

$$N! = \begin{cases} 1, & \text{если } N = 1, \\ N \cdot (N-1)!, & \text{если } N > 1. \end{cases}$$

$$1! = 1, \quad 2! = 2 \cdot 1! = 2 \cdot 1, \quad 3! = 3 \cdot 2! = 3 \cdot 2 \cdot 1$$

$$4! = 4 \cdot 3! = 4 \cdot 3 \cdot 2 \cdot 1$$

$$N! = N \cdot (N-1) \cdot \square \cdot 2 \cdot 1$$

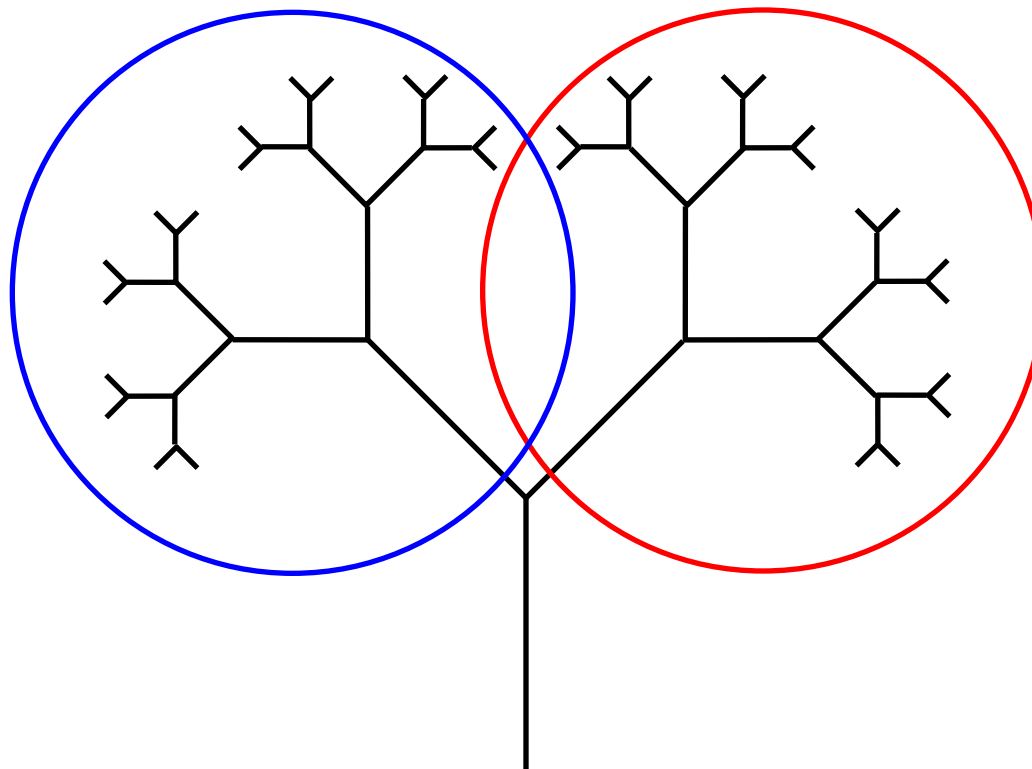
**Рекурсивный объект** – это объект, определяемый через один или несколько таких же объектов.



# Дерево Пифагора

**Дерево Пифагора из N уровней** – это ствол и отходящие от него симметрично **два дерева Пифагора из N-1 уровней**, такие что длина их стволов в 2 раза меньше и угол между ними равен  $90^\circ$ .

6 уровней:



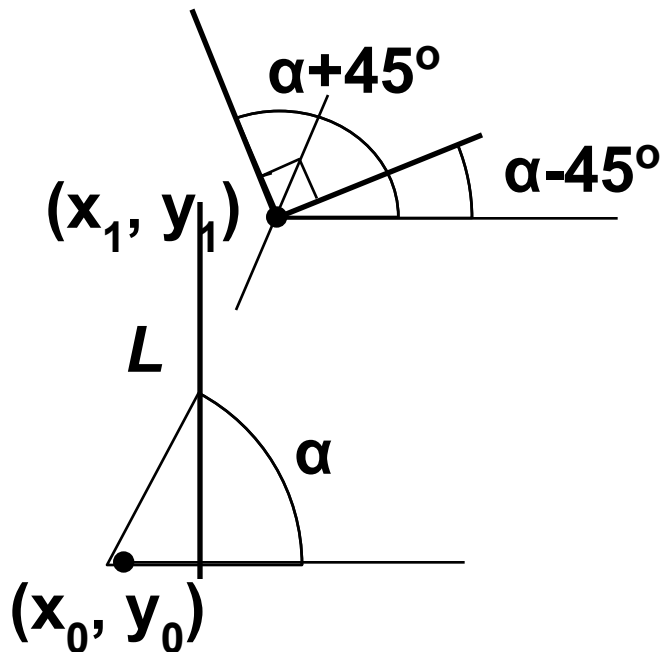
Как доказать, что это рекурсивная фигура?

# Дерево Пифагора

## Особенности:

- когда остановиться?
- деревья имеют различный наклон

когда число оставшихся уровней станет равно нулю!



$$x_1 = x_0 + L \cdot \cos(\alpha)$$

$$y_1 = y_0 - L \cdot \sin(\alpha)$$

наклон "дочерних" деревьев

$$\alpha + \pi/4$$

$$\alpha - \pi/4$$

# Процедура

угол  $\alpha$

длина ствола

```
procedure Pifagor(x0, y0, a, L: real;
                 N: integer);
const k = 0.6;      { изменение длины }
var x1, y1: real;  { локальные переменные }
begin
    if N > 0 then begin
        x1 := x0 + L*cos(a);
        y1 := y0 - L*sin(a);
        Line (round(x0), round(y0),
              round(x1), round(y1));
        Pifagor (x1, y1, a+pi/4, L*k, N-1);
        Pifagor (x1, y1, a-pi/4, L*k, N-1);
    end;
end;
```

закончить, если N=0

рекурсивные  
вызовы

**Рекурсивной** называется процедура,  
вызывающая сама себя.

# Программа

```
program qq;
```

```
  procedure Pifagor(x0, y0, a, L: real;  
                  N: integer);
```

```
  ...  
end;
```

угол  $\alpha$

длина ствола

```
begin
```

```
  Pifagor (250, 400, pi/2, 150, 8);
```

```
end;
```

$x_0$

$y_0$

число уровней



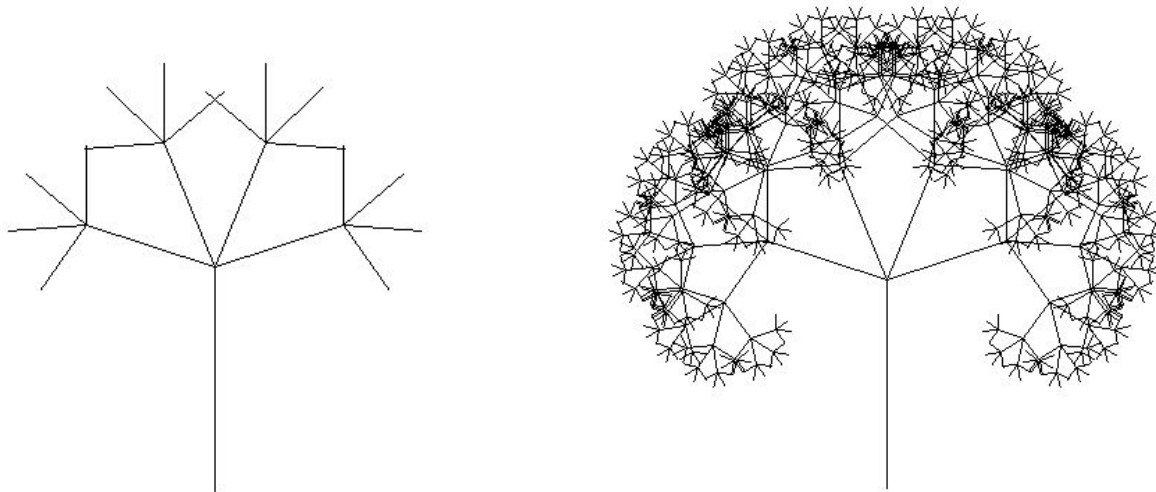
Как наклонить дерево вправо на  $30^\circ$ ?

```
Pifagor (250, 400, 2*pi/3, 150, 8);
```

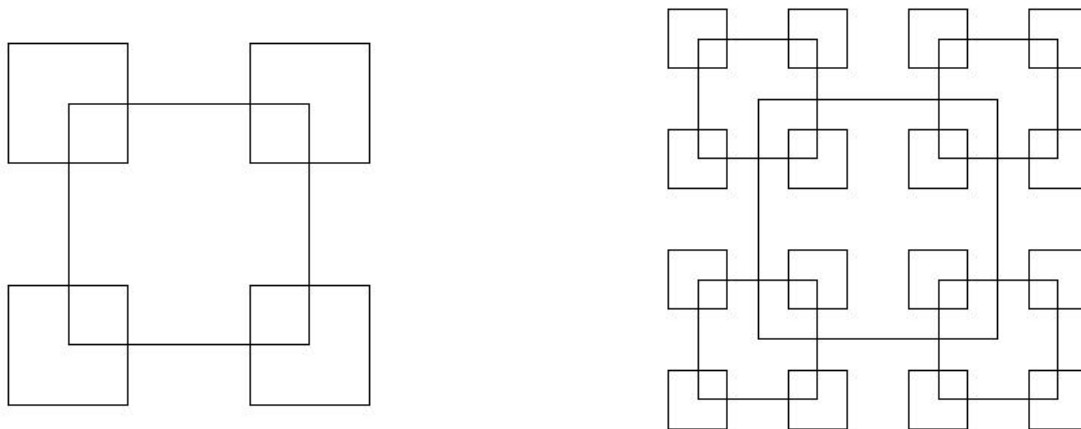
# Задания

---

**"4"**: Используя рекурсивную процедуру, построить фигуру:



**"5"**: Используя рекурсивную процедуру, построить фигуру:



# Программирование на языке Паскаль

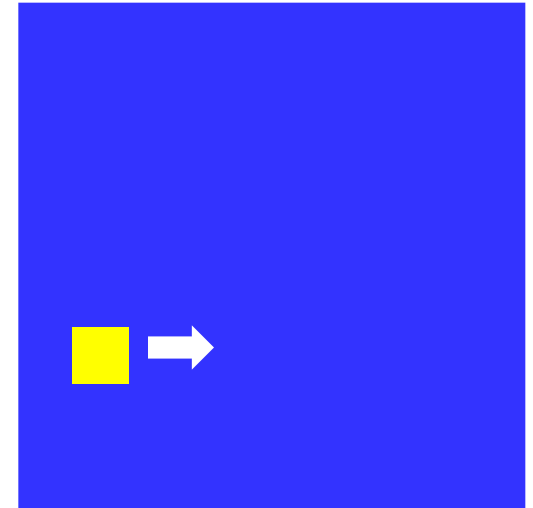
## Тема 11. Анимация

# Анимация

---

**Анимация** (англ. *animation*) – оживление изображения на экране.

**Задача:** внутри синего квадрата 400 на 400 пикселей слева направо движется желтый квадрат 20 на 20 пикселей. Программа останавливается, если нажата клавиша **Esc** или квадрат дошел до границы синей области.



**Проблема:** как изобразить перемещение объекта на экране?

**Привязка:** состояние объекта задается координатами **(x,y)**

**Принцип анимации:**

1. рисуем объект в точке **(x,y)**
2. задержка на несколько миллисекунд
3. стираем объект
4. изменяем координаты **(x,y)**
5. переходим к шагу 1

# Как "поймать" нажатие клавиши?

---

**Событие** – это изменение в состоянии какого-либо объекта или действие пользователя (нажатие на клавишу, щелчок мышкой).

**IsEvent** – логическая функция, которая определяет, было ли какое-то действие пользователя.

**Event** – процедура, которая определяет, какое именно событие случилось.

```
if IsEvent then begin
  Event(k, x, y);
  if k = 1 then
    writeln('Клавиша с кодом ', x)
  else { k = 2 }
    writeln('Мышь: x=', x, ' y=', y);
end;
```

```
var k, x, y: integer;
```



# Как выйти из цикла при нажатии *Esc*?

```
program qq;  
var stop: boolean;  
    k, code, i: integer;  
begin  
    stop := False;  
    repeat  
        if IsEvent then begin  
            Event(k, code, i);  
            if (k = 1) and (code = 27) then  
                stop := True;  
        end;  
        ...  
    until stop;  
end;
```

True, если надо остановиться

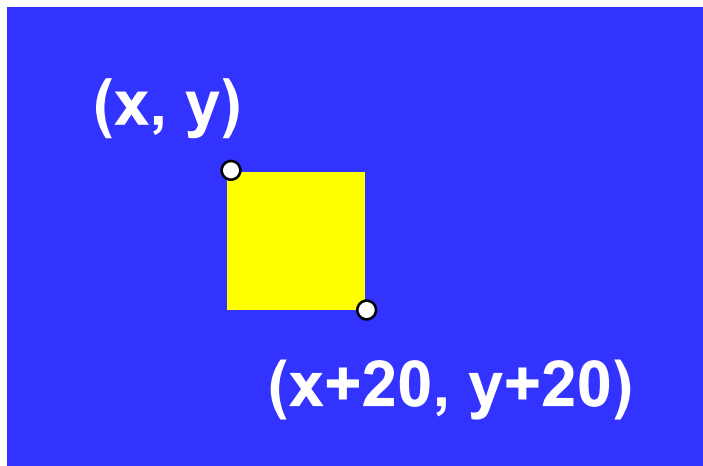
запуск цикла

если что-то произошло...

что произошло?

если нажата клавиша с  
кодом 27 (*Esc*), то стоп

# Процедура (рисование и стирание)



## Идеи

- одна процедура рисует и стирает
- стереть = нарисовать цветом фона
- границу квадрата отключить (в основной программе)

рисовать (**True**) или нет (**False**)?

```
procedure Draw(x, y: integer; flag: boolean);  
begin  
  if flag then  
    Brush(1, 255, 255, 0)  
  else  
    Brush(1, 0, 0, 255);  
  Rectangle(x, y, x+20, y+20);  
end;
```

рисуем: цвет кисти – желтый

стираем: цвет кисти – синий

только заливка!

# Полная программа

```
program qq;  
var x, y, k, code, i: integer;  
    stop: boolean;  
procedure Draw(x,y: integer; flag: Boolean);  
begin  
    ...  
end;  
begin  
    Brush(1, 0, 0, 255);  
    Rectangle(10, 10, 400, 400);  
    Pen(0, 0, 0, 255);  
    x := 10; y := 200; stop := false;  
    repeat  
        if IsEvent then begin  
            ...  
        end;  
        Draw(x, y, True);  
        Delay(10);  
        Draw(x, y, False);  
        x := x + 1;  
        if x >= 400-20 then stop := true;  
    until stop;  
end.
```

процедура

синий фон

отключить границу

начальные условия

выход по  
клавише *Esc*

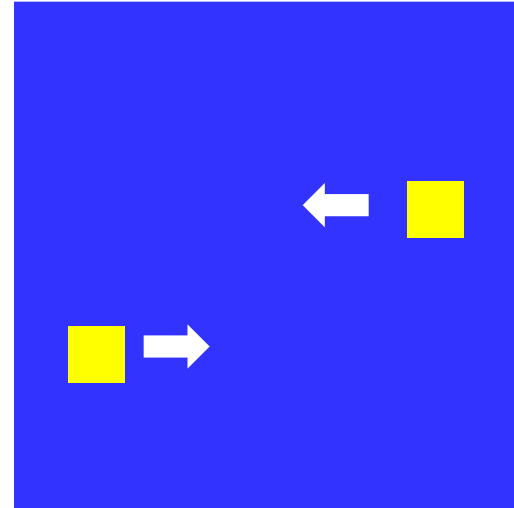
ждем 10 мс

выход при касании  
границы

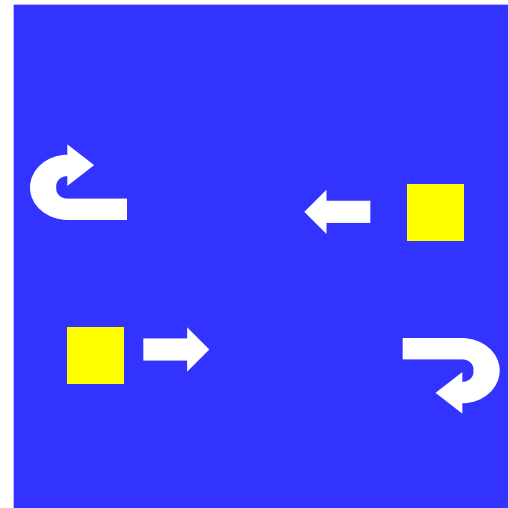
# Задания

---

**"4"**: Два квадрата двигаются в противоположном направлении:



**"5"**: Два квадрата двигаются в противоположном направлении и отталкиваются от стенок синего квадрата:



# Управление клавишами

**Задача:** жёлтый квадрат внутри синего квадрата управляется клавишами-стрелками. Коды клавиш:

влево – **37**                      вверх – **38**            Esc – **27**  
вправо – **39**                      вниз – **40**

**Проблема:** как изменять направление движения?

**Решение:**

```
if IsEvent then begin
  Event ( k, code, i);
  if k = 1 then begin
    case code of
      37: x := x - 1; 38: y := y - 1;
      39: x := x + 1; 40: y := y + 1;
      27: stop := True;
    end;
  end;
end;
```

если было нажатие на клавишу, ...

# Программа

```
program qq;  
var x, y, k, code, i: integer;  
    stop: boolean;
```

процедура

```
procedure Draw(x,y: integer; flag: Boolean);  
begin  
    ...  
end;
```

```
begin
```

```
    ...
```

ОСНОВНОЙ ЦИКЛ

```
repeat
```

```
    Draw(x, y, True);
```

```
    Delay(20);
```

```
    Draw(x, y, False);
```

```
    if IsEvent then begin
```

```
        ...
```

```
    end;
```

```
until stop;
```

обработка  
СОБЫТИЙ

```
end.
```



Что плохо?

# Как убрать мигание?

---

**Проблема:** даже если не нажата никакая клавиша, квадрат перерисовывается через каждые 20 мс (мигание!)

**Что хочется:** не перерисовать квадрат, если не было никакого события

**Решение:** нарисовать квадрат и **ждать** события

**Новая проблема:** как **ждать** события?

**Решение новой проблемы:** пустой цикл "пока не случилось событие, ничего не делай":

```
while not IsEvent do;
```

# Программа

```
program qq;  
var x, y, k, code, i: integer;  
    stop: boolean;
```

процедура

```
procedure Draw(x,y: integer; flag: Boolean);  
begin  
    ...  
end;
```

```
begin
```

```
    ...
```

рисуем квадрат

```
repeat
```

```
    Draw(x, y, True);
```

```
    while not IsEvent do;
```

```
        Draw(x, y, False);
```

```
        Event(k, code, i);
```

```
        ...
```

```
until stop;
```

ждем события

только теперь стираем

```
end.
```



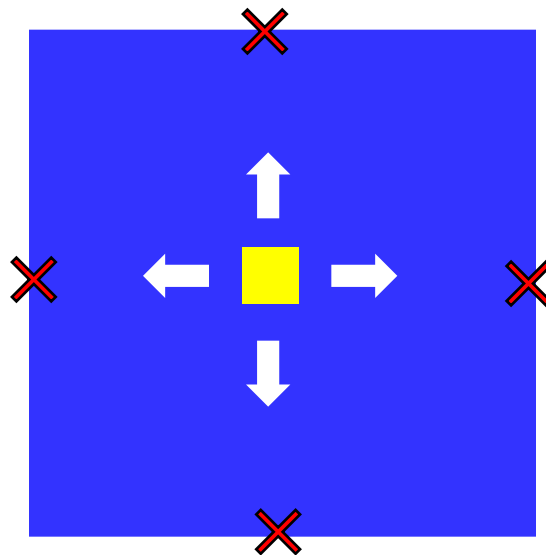
Что можно улучшить?



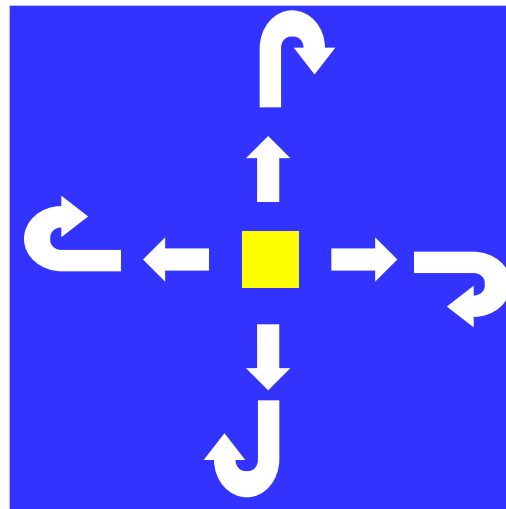
# Задания

---

**"4"**: Квадрат двигается при нажатии стрелок, однако не может выйти за границы синего квадрата:



**"5"**: Квадрат непрерывно двигается, при нажатии стрелок меняет направление и отталкивается от стенок синего квадрата:



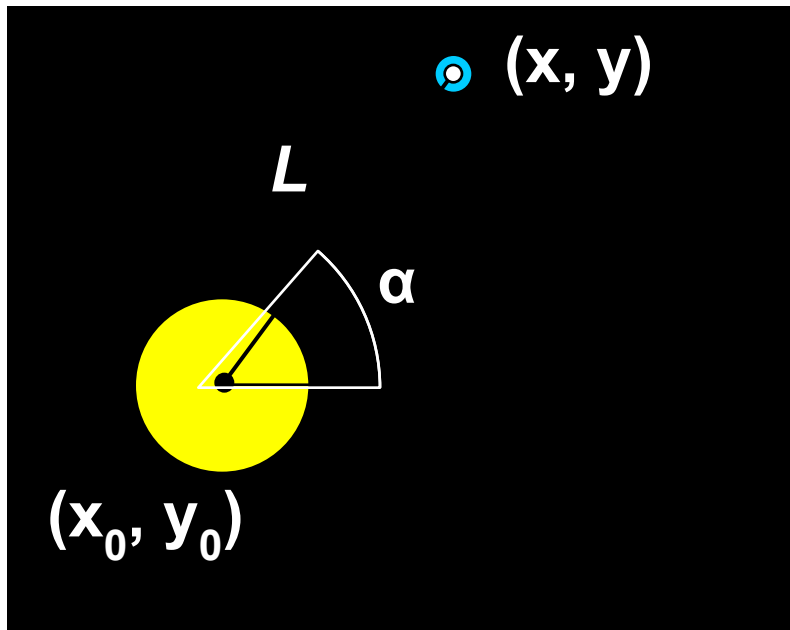
# Вращение

---

**Задача:** изобразить модель вращения Земли вокруг Солнца.

**Проблема:** движение по окружности, как изменять координаты?

**Решение:** использовать в качестве независимой переменной (менять в цикле) угол поворота  $\alpha$



$$x = x_0 + L \cdot \cos(\alpha)$$

$$y = y_0 - L \cdot \sin(\alpha)$$

# Процедура

рисовать (True) или нет (False)?

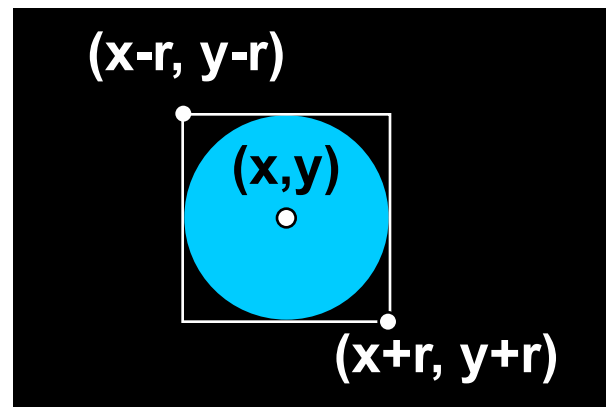
```
procedure Draw(x, y: integer; flag: boolean);  
const r = 10;  
begin  
  if flag then  
    Brush(1, 100, 100, 255)  
  else  
    Brush(1, 0, 0, 0);  
  Ellipse(x-r, y-r, x+r, y+r);  
end;
```

радиус Земли

рисуем: цвет кисти – голубой

стираем: цвет кисти – черный

только заливка!



# Константы и переменные

```
program qq;
const rSun = 60;      { радиус Солнца}
      L  = 150;      { радиус орбиты Земли }
      x0 = 200;      { координаты центра Солнца}
      y0 = 200;

var x, y,              { координаты Земли }
    k, code, i: integer; { для Event }
    a, ha: real;       { угол поворота, шаг }
    stop: boolean;     { признак остановки программы }
procedure Draw(x, y: integer; flag:
Boolean);
begin
    ...
end;
begin
    ...
end.
```

# Основная программа

```
program qq;
...
begin
  Brush(1, 0, 0, 0);  Fill(1,1);
  Brush(1, 255, 255, 0);
  Ellipse(x0-rSun, y0-rSun, x0+rSun, y0+rSun);
  a := 0; ha := 1*pi/180; { начальный угол, шаг 1° за 100 мс }
  stop := false;
  Pen(0,0,0,0);      { отключаем контуры }
  repeat
    x := round(x0 + L*cos(a));
    y := round(y0 - L*sin(a));
    Draw(x, y, True);
    Delay(100);
    Draw(x, y, False);
    if IsEvent then begin
      Event(k, code, i);
      if (k = 1) and (code = 27) then stop := true;
    end;
    a := a + ha;
  until stop;
end.
```

залить фон черным

рисует Солнце

НОВЫЕ КООРДИНАТЫ

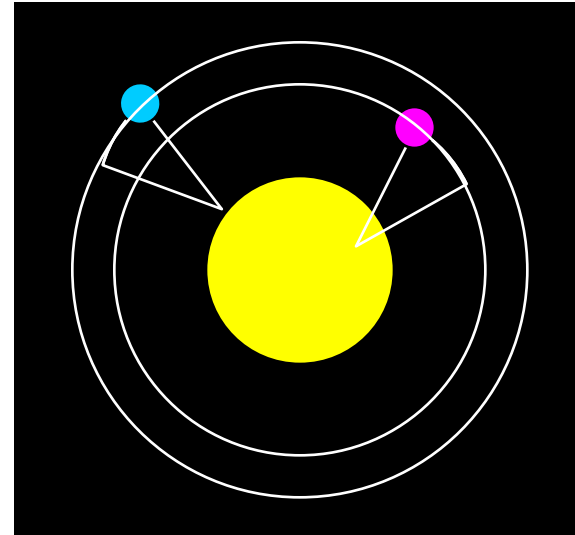
ждем 100 мс

ПОВОРОТ на ha

# Задания

---

**"4"**: Изобразить модель Солнца с двумя планетами, которые вращаются в противоположные стороны:



**"5"**: Изобразить модель системы Солнце-Земля-Луна:

