

# **ОСНОВЫ ЯЗЫКА Pascal**

## **Неуправляемое движение объектов**

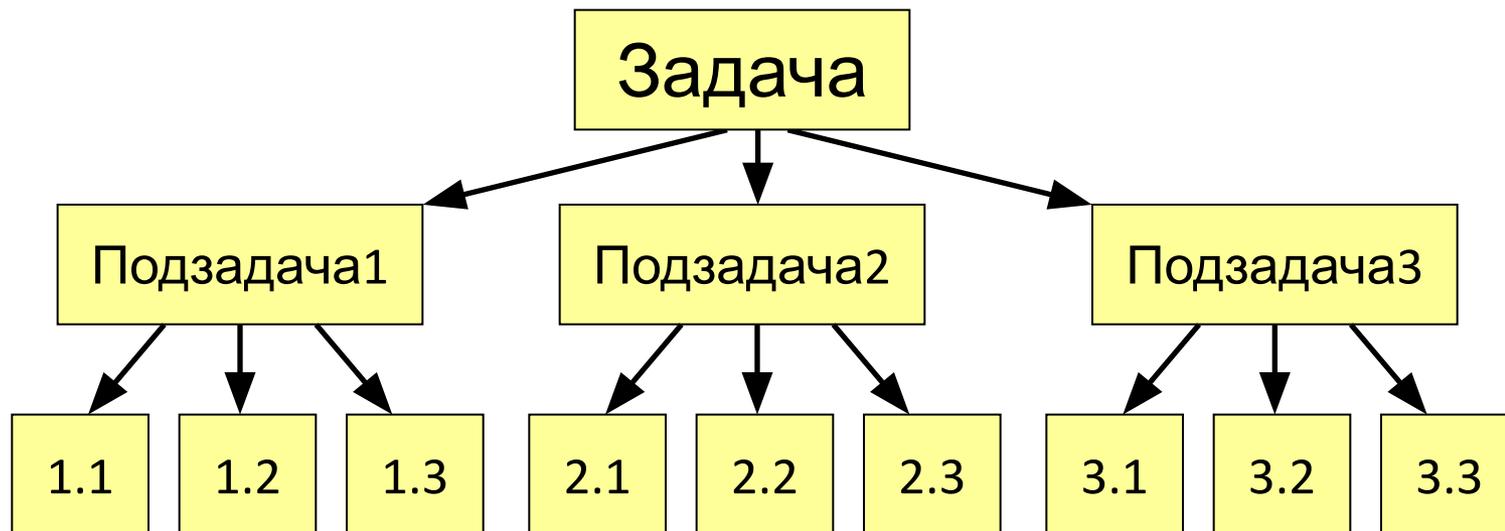
# Процедуры

---

**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

---

## Особенности:

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

```
procedure Krug(x, y, r: integer; col:longint);
```

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

```
Krug (200, 100, 50, Green);
```

x

y

r

col

# 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,

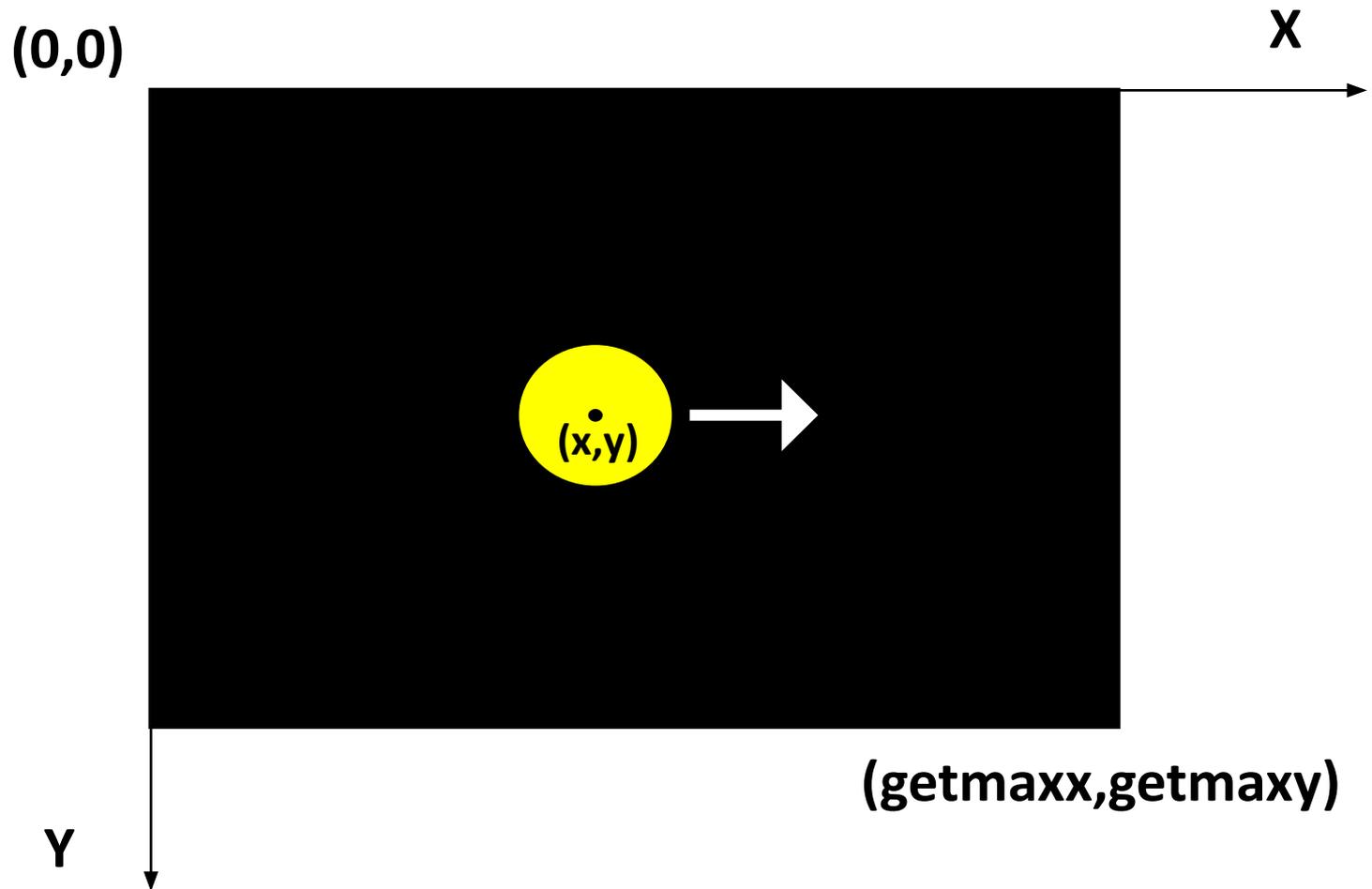
## Запрещенные варианты вызова

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

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

# Неуправляемое движение

---



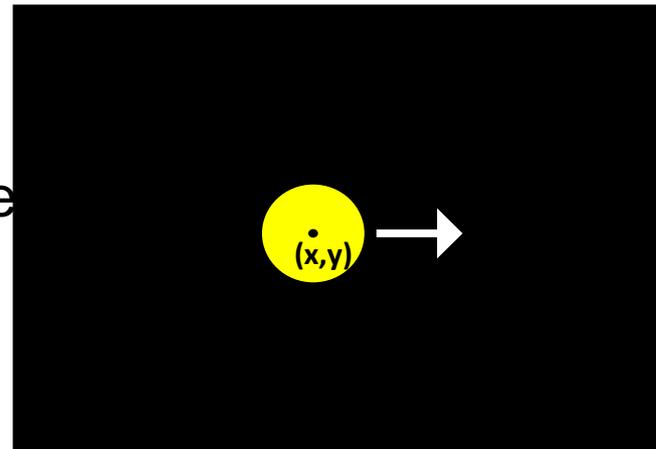
# Анимация

---

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

**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

# Процедура (отрисовки круга)

---

По заданным **координатам**, **радиусу** и **цвету** отрисовываем круг в нужном месте:

```
procedure Krug(x, y, r: integer; col: longint);  
begin  
    setcolor(col);  
    circle(x, y, r);  
    setfillstyle(1, col);  
    floodfill(x, y, col);  
end;
```

# Процедура (рисование и стирание)

## Идеи

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

**Delay (ms)** – задержка в миллисекундах

Изменяемые параметры

```
procedure Neupr(var x, y: integer; r, h: integer;
col: longint);
begin
  Krug(x, y, r, black);
  x := x + h;
  Krug(x, y, r, col);
  Delay(50);
end;
```

Стираем: рисуем круг цветом фона

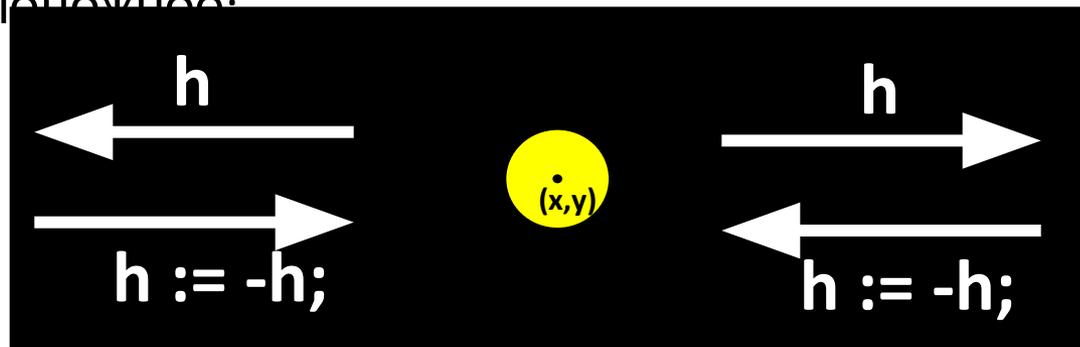
Сдвиг по оси X на шаг h

Ждем 50мс

Рисуем в новом месте цветом col

# Отталкивание от границ экрана

Приращение  $h$  при достижении границы меняет своё значение на противоположное:



```
procedure Neupr(var x, y, h: integer;  
r: integer; col: longint);  
begin  
  Krug(x, y, r, black);  
  x := x + h;  
  if (x < 0) or (x > getmaxx) then h := -h;  
  Krug(x, y, r, col);  
  Delay(50);  
end;
```

# Полная программа

```
Uses wingraph, wincrt;  
var x, y, r, h, gd, gm: integer;
```

```
procedure Krug(x,y,r: integer; col: longint);  
begin  
  ...  
end;
```

```
procedure Neupr(var x, y, h: integer;  
r: integer; col: longint);  
begin  
  ...  
end;
```

```
Begin
```

```
  Initgraph(gd, gm, '');
```

```
  x := 30;  
  y := getmaxy div 2;  
  r := 50;  
  h := 5;  
  Krug(x, y, r, yellow);  
  repeat  
    Neupr(x, y, h, r, yellow);  
  until Keypressed;  
  Closegraph;
```

```
end.
```

процедура Krug обязательно до  
Neupr (до своего вызова)

процедуры

начальные  
условия

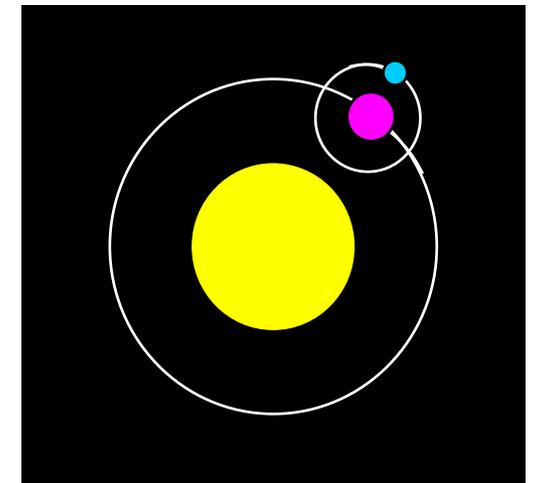
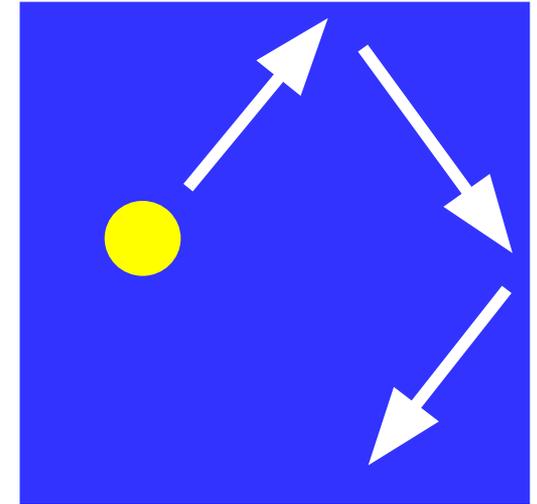
первая отрисовка  
круга

выход при нажатии  
клавиши

# Задание

---

1. Нарисовать мишень, случайно появляющуюся и исчезающую, использовать процедуру отрисовки мишени.
2. Нарисовать круг, движущийся по оси X и отталкивающийся от границ экрана.
2. Нарисовать круг, движущийся по оси Y и отталкивающийся от границ экрана.
3. Нарисовать круг, движущийся по диагоналям и отталкивающийся от границ экрана.
4. Нарисовать 2 разноцветных круга, движущихся по диагоналям и отталкивающих от границ экрана.
5. Нарисовать круг, движущийся по диагоналям и отталкивающийся от границ цветной области экрана (меньше самого экрана).
6. Нарисовать движение Земли вокруг Солнца.
7. Нарисовать модель Солнце-Земля-Луна

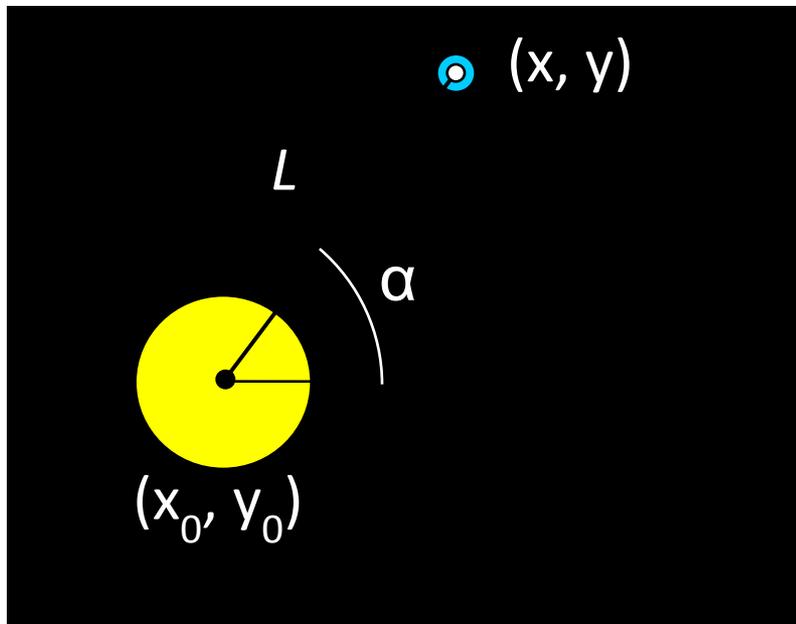


# Вращение

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

**Решение:** использовать в качестве `a := a + ha;`

`ha := 1*pi/180; {шаг 1° за 100 мс}` поворота  $\alpha$   
, где



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

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

```
x := round(x0 + L*cos(a));  
y := round(y0 - L*sin(a));
```