

# Программирование на языке Паскаль

- |  |  |
|--|--|
| 1. <a href="#"><u>Введение</u></a>         | 7. <a href="#"><u>Графика</u></a>          |
| 2. <a href="#"><u>Ветвления</u></a>        | 8. <a href="#"><u>Графики функций</u></a>  |
| 3. <a href="#"><u>Сложные условия</u></a>  | 9. <a href="#"><u>Процедуры</u></a>        |
| 4. <a href="#"><u>Циклы</u></a>            | 10. <a href="#"><u>Рекурсия</u></a>        |
| 5. <a href="#"><u>Циклы с условием</u></a> | 11. <a href="#"><u>Анимация</u></a>        |
| 6. <a href="#"><u>Оператор выбора</u></a>  | 12. <a href="#"><u>Случайные числа</u></a> |
|  | 13. <a href="#"><u>Функции</u></a>         |

# Программирование на языке Паскаль

## Тема 1. Введение

# Алгоритм

---

**Алгоритм** – это четко определенный план действий для исполнителя.

## Свойства алгоритма

- **дискретность**: состоит из отдельных шагов (команд)
- **понятность**: должен включать только команды, известные исполнителю (входящие в СКИ)
- **определенность**: при одинаковых исходных данных всегда выдает один и тот же результат
- **конечность**: заканчивается за конечное число шагов
- **массовость**: может применяться многократно при различных исходных данных
- **корректность**: дает верное решение при любых допустимых исходных данных

# Программа

---

**Программа** – это

- алгоритм, записанный на каком-либо языке программирования
- набор команд для компьютера

**Команда** – это описание действий, которые должен выполнить компьютер.

- откуда взять исходные данные?
- что нужно с ними сделать?

# Языки программирования

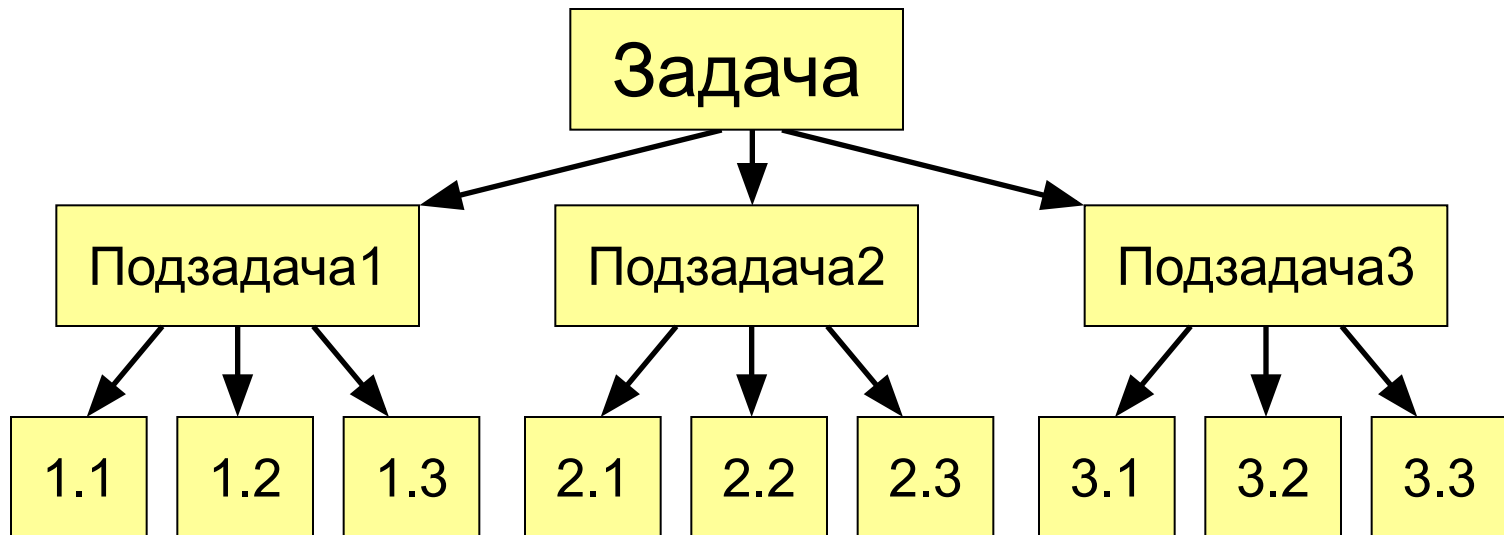
---

- **Машинно-ориентированные (низкого уровня)** - каждая команда соответствует одной команде процессора (ассемблер)
- **Языки высокого уровня** – приближены к естественному (английскому) языку, легче воспринимаются человеком, **не зависят от конкретного компьютера**
  - *для обучения*: Бейсик, ЛОГО, Паскаль
  - *профессиональные*: Си, Фортран, Паскаль
  - *для задач искусственного интеллекта*: Пролог, ЛИСП
  - *для Интернета*: JavaScript, Java, Perl, PHP, ASP

# Язык Паскаль

**1970** – Никлаус Вирт (Швейцария)

- язык для обучения студентов
- разработка программ «сверху вниз»



- разнообразные структуры данных (массивы, структуры, множества)

# Из чего состоит программа?

---

```
program <имя программы>;  
const ...; {константы}  
var ...; {переменные}  
  
{ процедуры и функции }  
begin  
    ... {основная программа}  
end.
```

комментарии в фигурных скобках  
не обрабатываются

# Из чего состоит программа?

---

**Константа** – постоянная величина, имеющая имя.

**Переменная** – изменяющаяся величина, имеющая имя (ячейка памяти).

**Процедура** – вспомогательный алгоритм, описывающий некоторые действия (рисование окружности).

**Функция** – вспомогательный алгоритм для выполнения вычислений (вычисление квадратного корня, **sin**).



# Имена программы, констант, переменных

---

## Имена могут включать

- латинские буквы (A-Z)

заглавные и строчные буквы не различаются

- цифры

имя не может начинаться с цифры

- знак подчеркивания \_

## Имена НЕ могут включать

- русские буквы
- пробелы
- скобки, знаки +, =, !, ? и др.

## Какие имена правильные??

AXby R&B 4Wheel Вася “PesBarbos”  
TU154 [QuQu] \_ABBA A+B

# Константы

**const**

**i2 = 45; { целое число }**

**pi = 3.14; { вещественное число }**

целая и дробная часть отделяются точкой

**qq = 'Вася'; { строка символов }**

можно использовать русские буквы!

**L = True; { логическая величина }**

может принимать два значения:

- True (истина, «да»)
- False (ложь, «нет»)

# Переменные

---

**Переменная** – это величина, имеющая имя, тип и значение. Значение переменной можно изменять во время работы программы.

## Типы переменных:

- integer { целая }
- real { вещественная }
- char { один символ }
- string { символьная строка }
- boolean { логическая }

## Объявление переменных (выделение памяти):

```
var a, b: integer;  
    Q: real;  
    s1, s2: string;
```

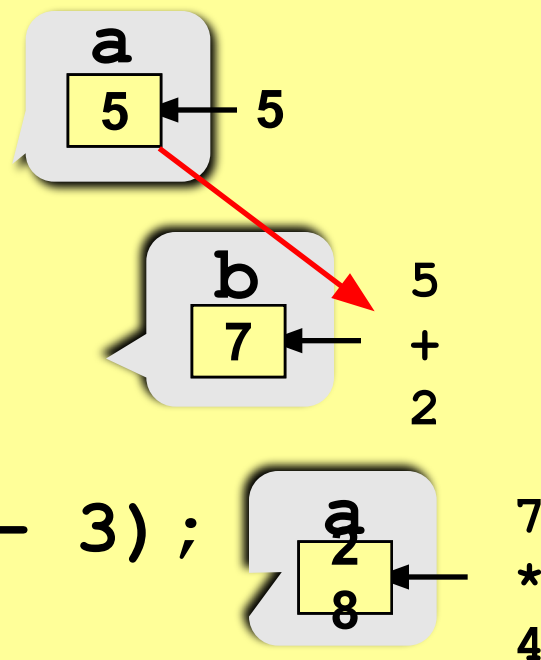
# Как изменить значение переменной?

**Оператор** – это команда языка программирования высокого уровня.

**Оператор присваивания** служит для изменения значения переменной.

**Пример:**

```
program qq;  
var a, b: integer;  
begin  
    a := 5;  
    b := a + 2;  
    a := (a + 2) * (b - 3);  
end.
```



# Оператор присваивания

---

## Общая структура:

*<имя переменной> := <выражение>;*

## Арифметическое выражение может включать

- КОНСТАНТЫ
- имена переменных
- знаки арифметических операций:

+ -

\*

/

div

mod

умножение

деление

деление  
нацело

остаток от  
деления

- ВЫЗОВЫ функций
- круглые скобки ( )

# Какие операторы неправильные?

```
program qq;  
var a, b: integer;  
    x, y: real;  
begin  
    a := 5;  
    10 := x;  
    y := 7,8;  
    b := 2.5;  
    x := 2*(a + y) ;  
    a := b + x;  
end.
```

имя переменной должно  
быть слева от знака :=

целая и дробная часть  
отделяются **точкой**

нельзя записывать  
вещественное значение в  
целую переменную

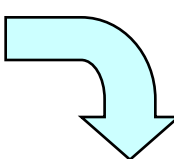
[illegible]

# Порядок выполнения операций

- вычисление выражений в скобках
- умножение, деление, **div**, **mod** слева направо
- сложение и вычитание слева направо

2 3 5 4 1 7 8 6

**z := (5\*a\*c+3\*(c-d)) / a\*(b-c) / b;**

$$x = \frac{a^2 + 5c^2 - d(a+b)}{(c+d)(d-2a)}$$


$$z = \frac{5ac + 3(c-d)}{ab} (b-c)$$

2 6 3 4 7 5 1 12 8 11 10

**x := (a\*a+5\*c\*c-d\*(a+b)) / ((c+d)\*(d-2\*a)) ;**



# Сложение двух чисел

---

**Задача.** Ввести два целых числа и вывести на экран их сумму.

**Простейшее решение:**

```
program qq;  
  var a, b, c:  
integer;  
begin  
  read ( a, b );  
  c := a + b;  
  writeln ( c );  
end.
```

# Оператор ввода

---

`read ( a ) ;`      { ввод значения  
                          переменной `a` }

`read ( a , b ) ;` { ввод значений  
                          переменных `a` и `b` }

## Как вводить два числа?

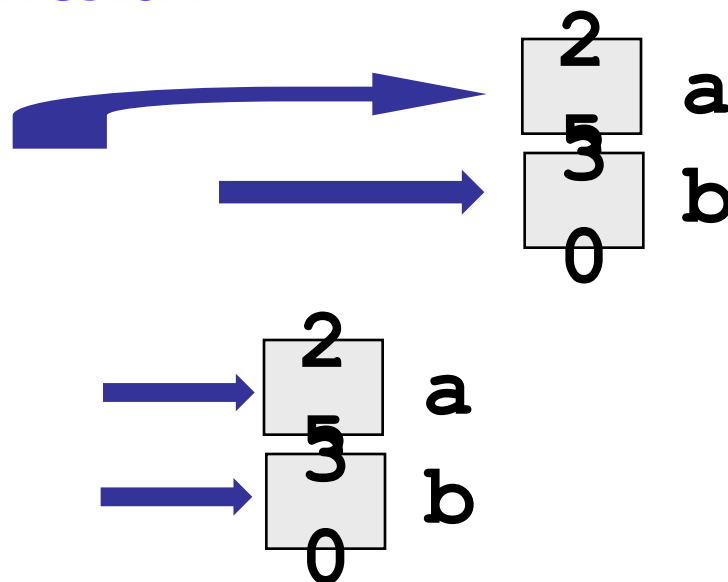
через пробел:

25 30

через *Enter*:

25

30



# Оператор вывода

---

`write ( a );`      { вывод значения  
                            переменной `a` }

`writeln ( a );` { вывод значения  
                            переменной `a` и переход  
                            на новую строку }

`writeln ( 'Привет!' );` { вывод текста }

`writeln ( 'Ответ: ', c );`      { вывод  
                            текста и значения переменной `c` }

`writeln ( a, '+', b, '=', c );`

# Форматы вывода

```

program qq;
var i: integer;
    x: real;
begin
  i := 15;
  writeln ( '>', i, '<' );
  writeln ( '>', i:5, '<' );
  x := 12.345678;
  writeln ( '>', x, '<' );
  writeln ( '>', x:10, '<' );
  writeln ( '>', x:7:2, '<' );
end.

```

ВСЕГО  
СИМВОЛОВ

```

>15<
>   15<

>1.234568E+001<
> 1.23E+001<
>  12.35<

```

ВСЕГО  
СИМВОЛОВ

в дробной  
части

# Полное решение

```
program qq;  
var a, b, c: integer;  
begin  
  writeln('Введите два целых числа');  
  read ( a, b );  
  c := a + b;  
  writeln ( a, '+', b, '=', c );  
end.
```

**Протокол:**

это выводит компьютер

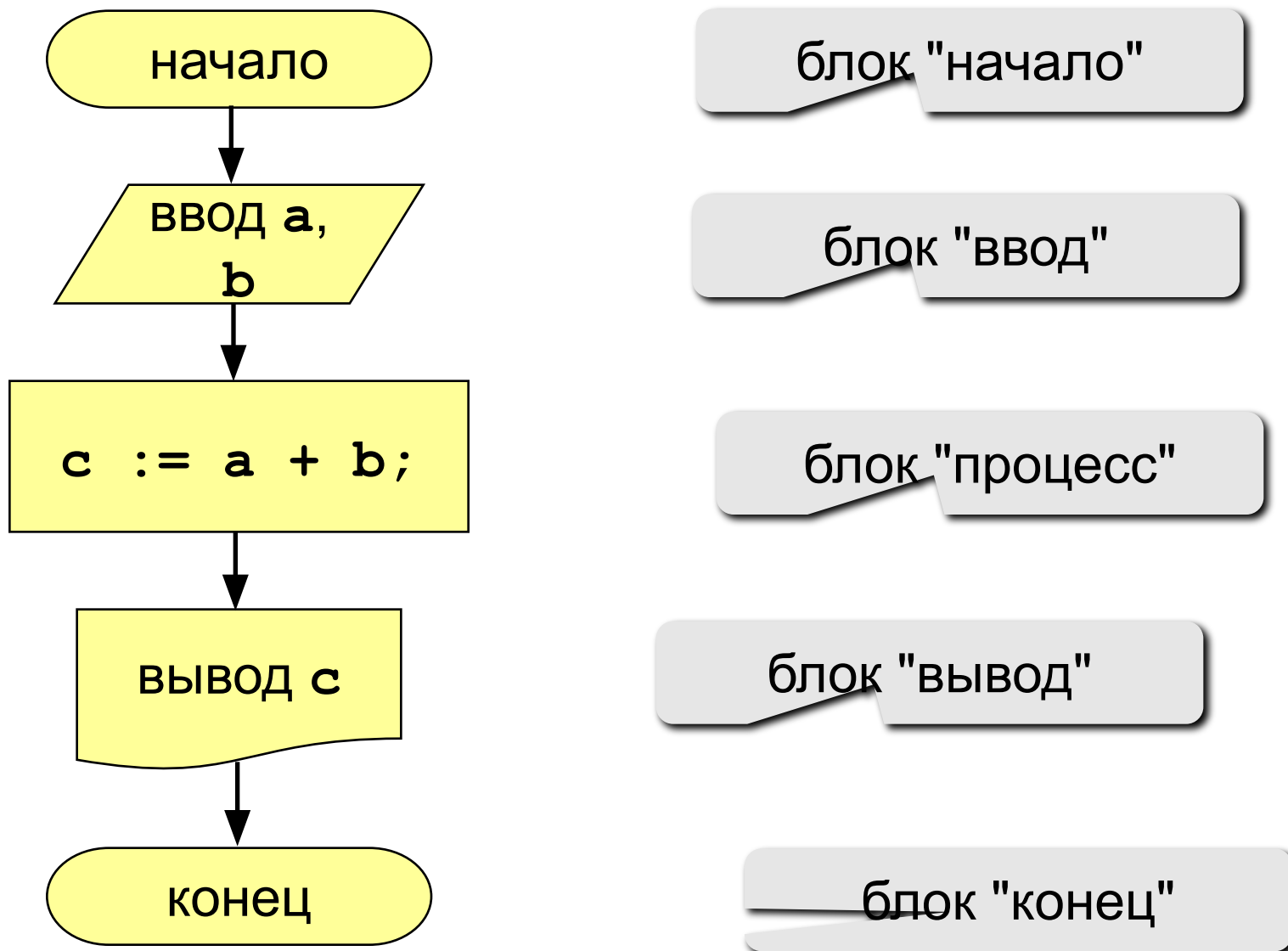
**Введите два целых числа**

**25 30**

это вводит пользователь

**25+30=55**

# Блок-схема линейного алгоритма



# Задания

---

**"4": Ввести три числа, найти их сумму и произведение.**

**Пример:**

Введите три числа:

4 5 7

$$4+5+7=16$$

$$4*5*7=140$$

**"5": Ввести три числа, найти их сумму, произведение и среднее арифметическое.**

**Пример:**

Введите три числа:

4 5 7

$$4+5+7=16$$

$$4*5*7=140$$

$$(4+5+7) / 3 = 5.33$$

# Программирование на языке Паскаль

## Тема 2. Ветвления



# Разветвляющиеся алгоритмы

---

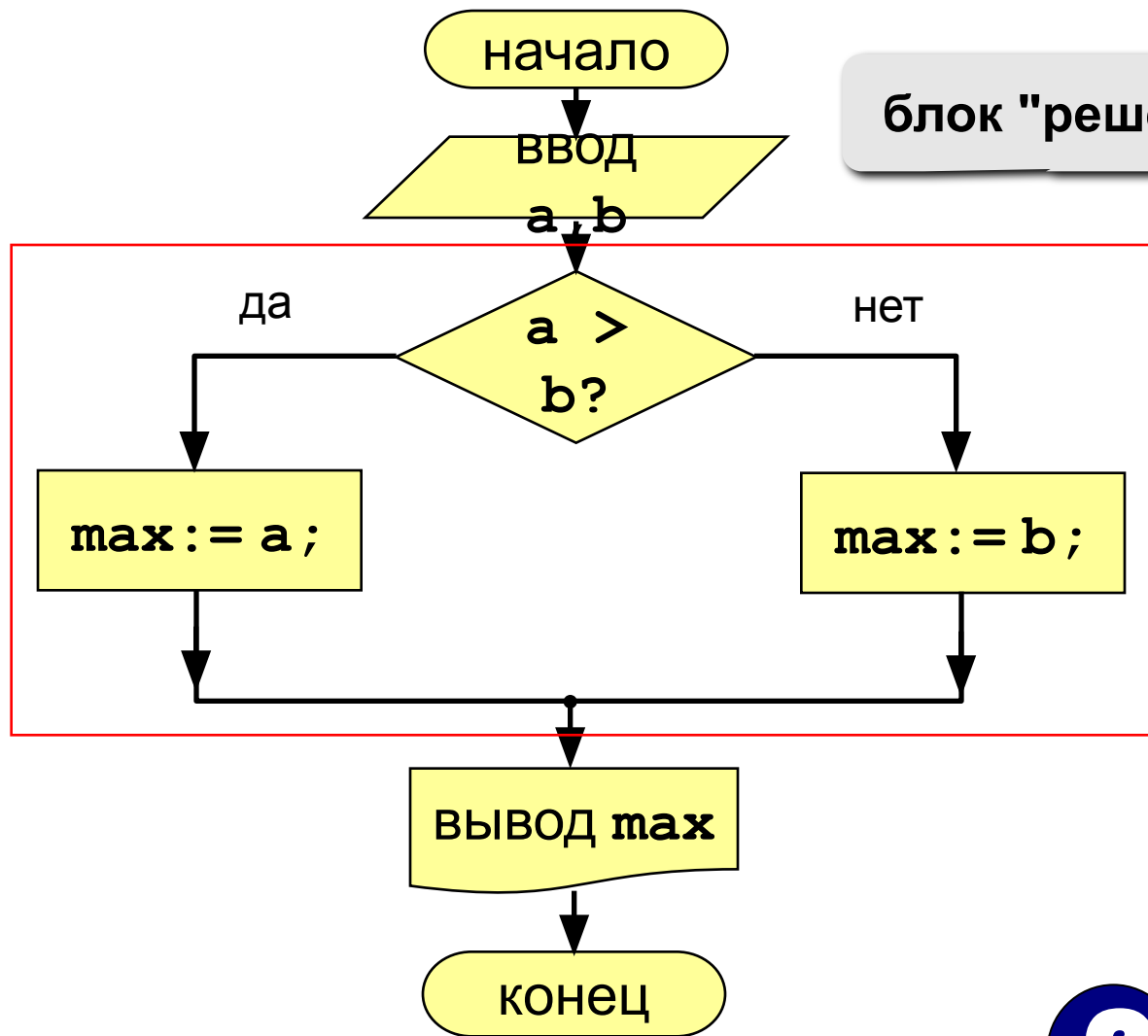
**Задача.** Ввести два целых числа и вывести на экран наибольшее из них.

**Идея решения:** надо вывести на экран первое число, если оно больше второго, или второе, если оно больше первого.

**Особенность:** действия исполнителя зависят от некоторых условий (***если ... иначе ...***).

Алгоритмы, в которых последовательность шагов зависит от выполнения некоторых условий, называются **разветвляющимися**.

# Вариант 1. Блок-схема



блок "решение"

полная форма  
ветвления



Если  $a = b$ ?

# Вариант 1. Программа

```
program qq;  
var a, b, max: integer;  
begin  
  writeln('Введите два целых числа');  
  read ( a, b );  
  if a > b then begin  
    max := a;  
  end  
  else begin  
    max := b;  
  end;  
  writeln ('Наибольшее число ', max);  
end.
```

полная форма  
условного  
оператора

# Условный оператор

---

```
if <условие> then begin
    {что делать, если условие верно}
end
else begin
    {что делать, если условие неверно}
end;
```

## Особенности:

- перед **else** **НЕ** ставится точка с запятой
- вторая часть (**else** ...) может отсутствовать (неполная форма)
- если в блоке один оператор, можно убрать слова **begin** и **end**

# Что неправильно?

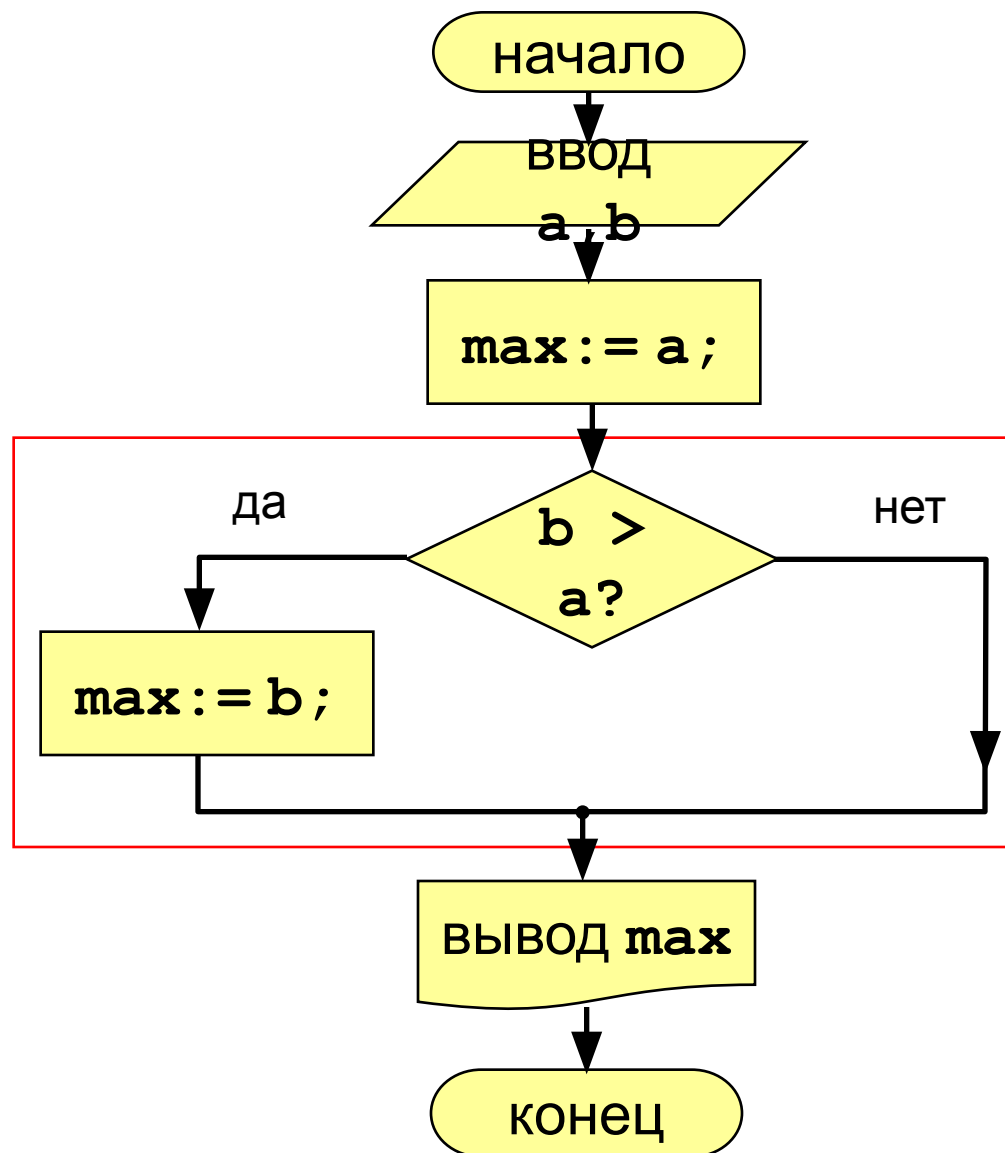
```
if a > b then begin
  a := b;
end
else begin
  b := a;
end;
```

```
if a > b then begin
  a := b; end
else begin
  b := a;
end;
```

```
if a > b then begin
  a := b;
end
else begin
  b := a;
end;
```

```
if a > b then begin
  a := b;
end
else begin
  b := a;
end;
```

## Вариант 2. Блок-схема



неполная форма  
ветвления

## Вариант 2. Программа

```
program qq;  
var a, b, max: integer;  
begin  
  writeln('Введите два целых числа');  
  read ( a, b );  
  max := a;  
  if b > a then  
    max := b;  
  writeln ('Наибольшее число ', max);  
end.
```

неполная форма  
условного  
оператора

## Вариант 2Б. Программа

```
program qq;  
var a, b, max: integer;  
begin  
  writeln('Введите два целых числа');  
  read ( a, b );  
  max := b;  
  if a > b then  
    max := a;  
  writeln ('Наибольшее число ', max);  
end.
```



# Что неправильно?

---

```
if a > b the  
  a := b  
else b := a;
```

```
if a > b then begin  
  a := b;  
end  
else b := a;
```

```
if a > b then  
  a := b  
else b := a;
```

```
if b >= a then  
  b := a;
```

# Задания

---

**"4": Ввести три числа и найти наибольшее из них.**

**Пример:**

Введите три числа:

4      15      9

Наибольшее число 15

**"5": Ввести пять чисел и найти наибольшее из них.**

**Пример:**

Введите пять чисел:

4      15      9      56      4

Наибольшее число 56

# Программирование на языке Паскаль

## Тема 3. Сложные условия

# Сложные условия

---

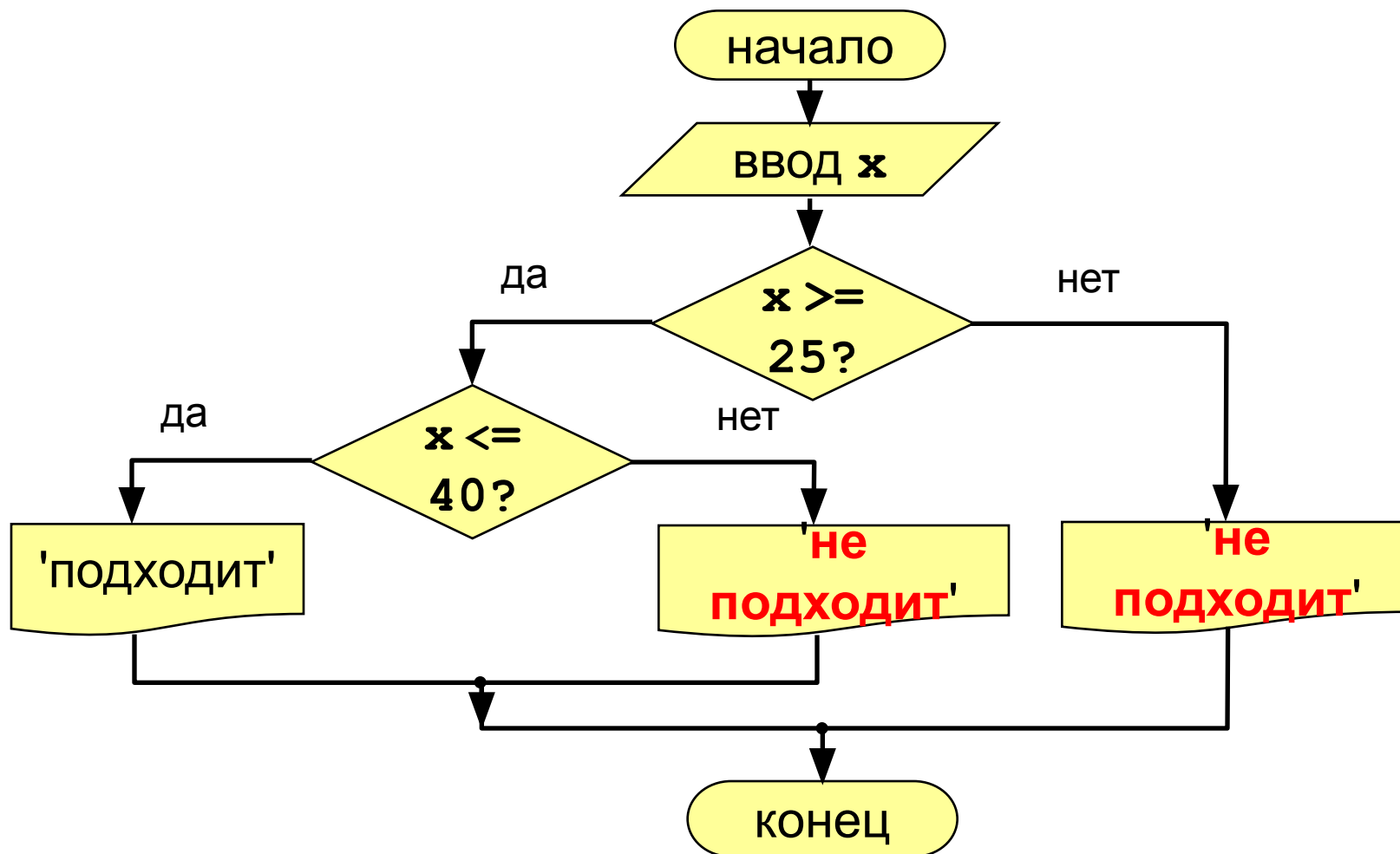
**Задача.** Фирма набирает сотрудников от 25 до 40 лет включительно. Ввести возраст человека и определить, подходит ли он фирме (вывести ответ "подходит" или "не подходит").

**Особенность:** надо проверить, выполняются ли два условия одновременно.



**Можно ли решить известными методами?**

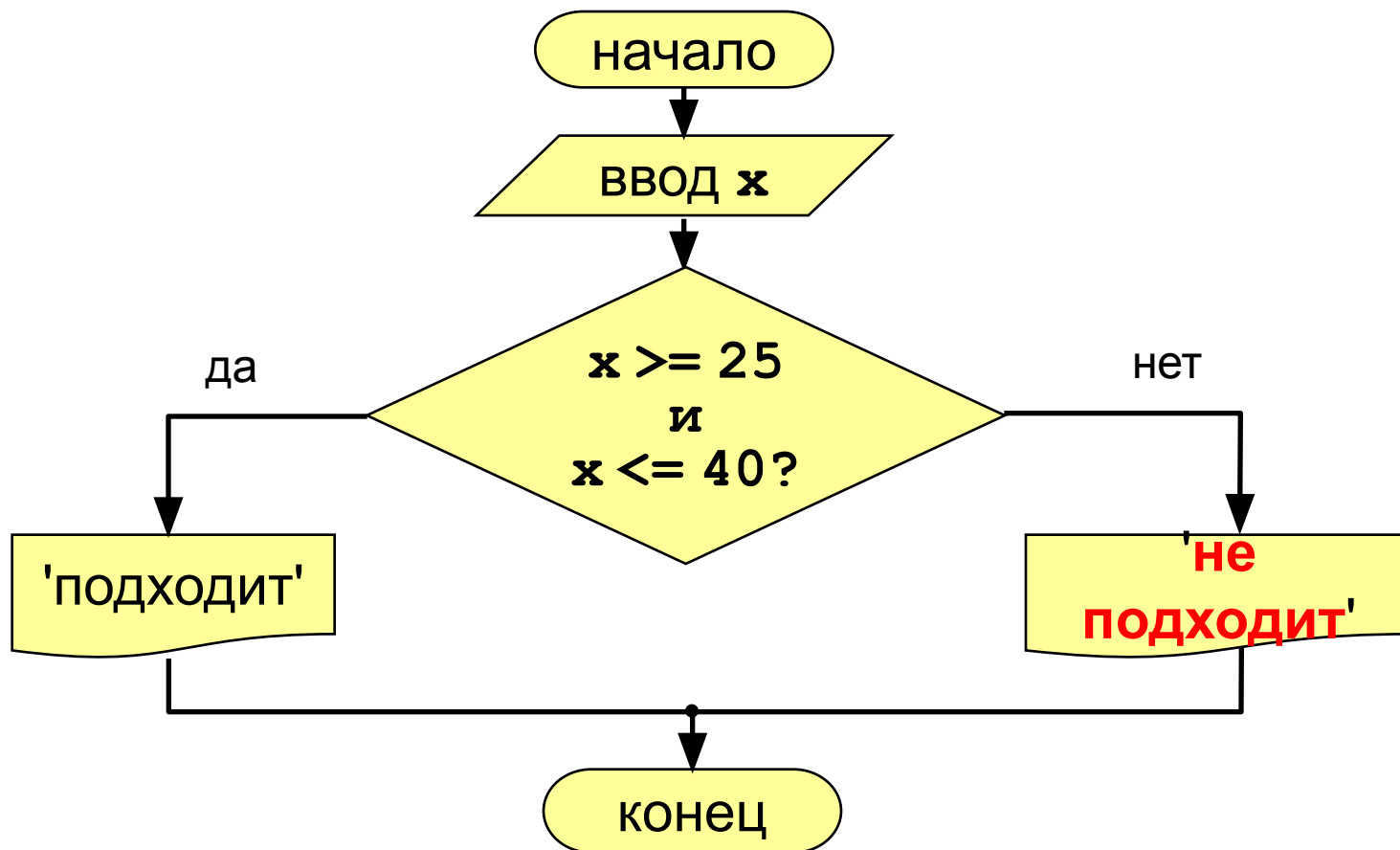
# Вариант 1. Алгоритм



# Вариант 1. Программа

```
program qq;  
var x: integer;  
begin  
    writeln('Введите возраст');  
    read ( x );  
    if x >= 25 then  
        if x <= 40 then  
            writeln ('Подходит')  
        else writeln ('Не подходит')  
    else  
        writeln ('Не подходит');  
end.
```

## Вариант 2. Алгоритм



## Вариант 2. Программа

```
program qq;  
var x: integer;  
begin  
  writeln('Введите возраст');  
  read ( x );  
  if (x >= 25) and (x <= 40) then  
    writeln ('Подходит')  
  else writeln ('Не подходит')  
end.
```

сложное  
условие



# Сложные условия

---

**Сложное условие** – это условие, состоящее из нескольких простых условий (отношений), связанных с помощью **логических операций**:

- **not** – НЕ (отрицание, инверсия)
- **and** – И (логическое умножение, конъюнкция, одновременное выполнение условий)
- **or** – ИЛИ (логическое сложение, дизъюнкция, выполнение хотя бы одного из условий)
- **xor** – исключающее ИЛИ (выполнение только одного из двух условий, но не обоих)

## Простые условия (отношения)

&lt;

&lt;=

&gt;

&gt;=

=

&lt;&gt;

равно

не равно

# Сложные условия

---

## Порядок выполнения

- выражения в скобках
- not
- and
- or, xor
- <, <=, >, >=, =, <>

**Особенность** – каждое из простых условий обязательно заключать в скобки.

## Пример

```
      4      1      6      2      5  
if not (a > b) or (c <> d) and (b <> a)  
then begin  
    ...  
end
```

# Сложные условия

Истинно или ложно при  $a := 2; b := 3; c := 4;$

`not (a > b)`

True

`(a < b) and (b < c)`

True

`not (a >= b) or (c = d)`

True

`(a < c) or (b < c) and (b < a)`

True

`(a < b) xor not (b > c)`

FALSE

Для каких значений **x** истинны условия:

`(x < 6) and (x < 10)`

`(x < 6) and (x > 10)`

`(x > 6) and (x < 10)`

`(x > 6) and (x > 10)`

`(x < 6) or (x < 10)`

`(x < 6) or (x > 10)`

`(x > 6) or (x < 10)`

`(x > 6) or (x > 10)`

$(-\infty, 6)$	$x < 6$
$\emptyset$	
$(6, 10)$	
$(10, \infty)$	$x > 10$
$(-\infty, 10)$	$x < 10$
$(-\infty, 6) \cup (10, \infty)$	
$(-\infty, \infty)$	
$(6, \infty)$	$x > 6$

# Задания

---

**"4": Ввести номер месяца и вывести название времени года.**

**Пример:**

Введите номер месяца:

4

весна

**"5": Ввести возраст человека (от 1 до 150 лет) и вывести его вместе с последующим словом "год", "года" или "лет".**

**Пример:**

Введите возраст:

24

Вам 24 года

Введите возраст:

57

Вам 57 лет

# Программирование на языке Паскаль

## Тема 4. Циклы

# Циклы

---

**Цикл** – это многократное выполнение одинаковой последовательности действий.

- цикл с **известным** числом шагов
- цикл с **неизвестным** числом шагов (цикл с условием)

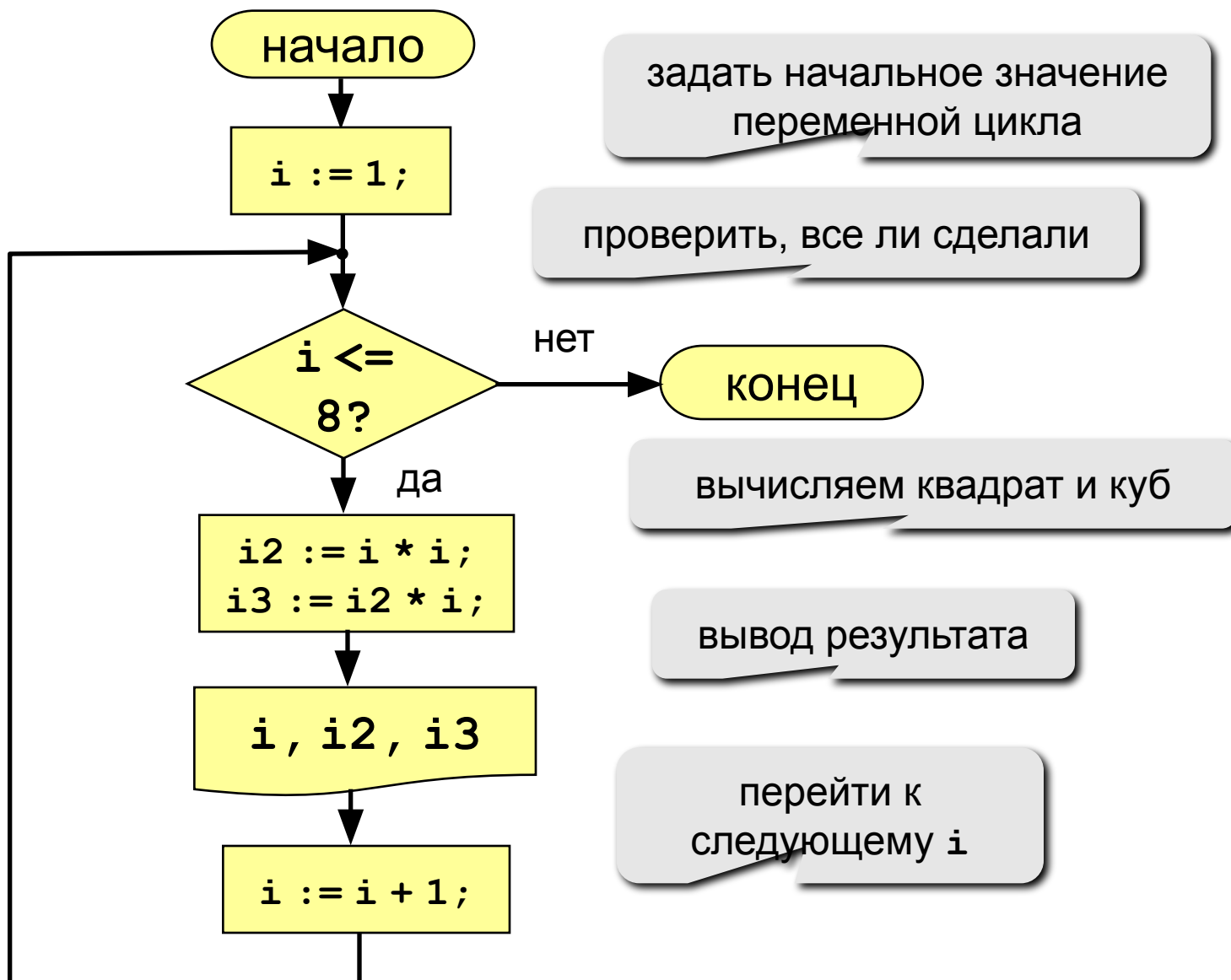
**Задача.** Вывести на экран квадраты и кубы целых чисел от 1 до 8 (от **a** до **b**).

**Особенность:** одинаковые действия выполняются 8 раз.

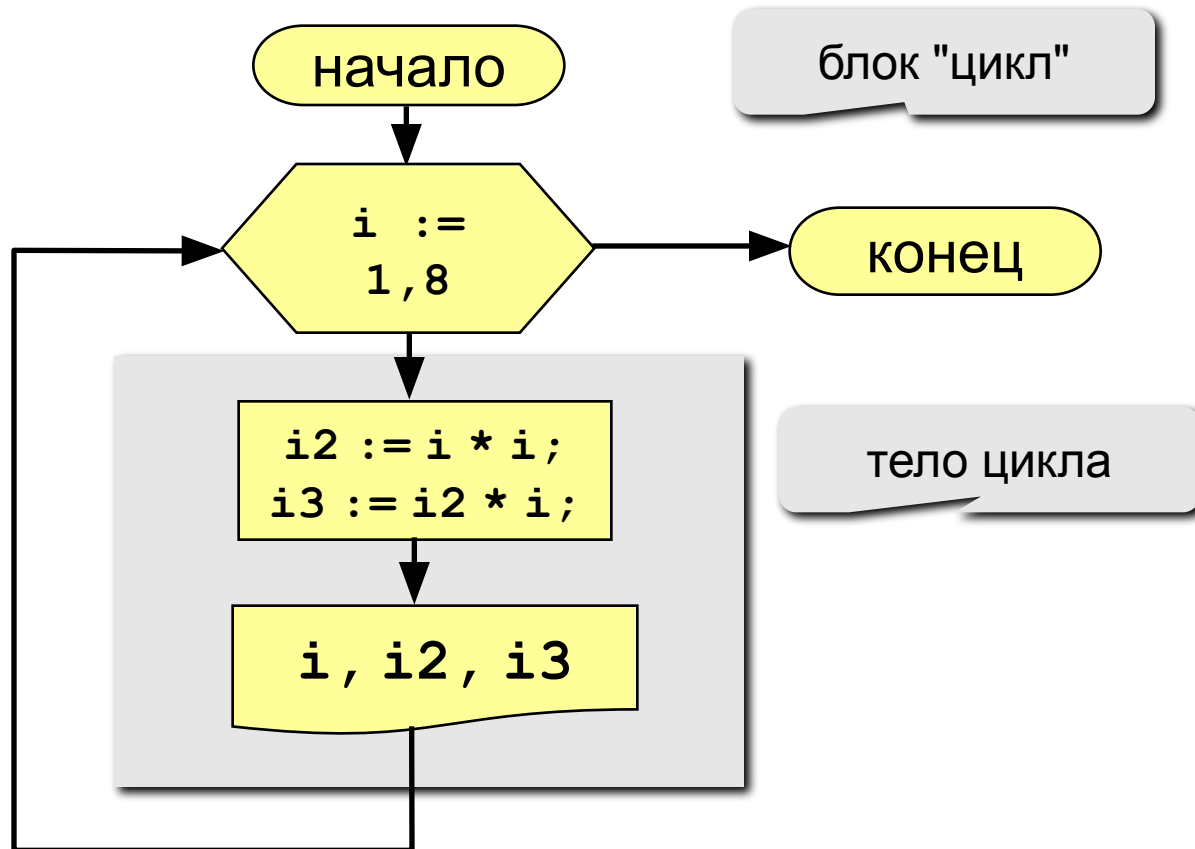


Можно ли решить известными методами?

# Алгоритм



# Алгоритм (с блоком "цикл")





# Программа

```
program qq;  
var i, i2, i3: integer;  
begin
```

переменная цикла

начальное значение

конечное значение

```
  for i:=1 to 8 do begin  
    i2 := i*i;  
    i3 := i2*i;  
    writeln(i:4, i2:4, i3:4);  
  end;
```

```
end.
```

# Цикл с уменьшением переменной

---

**Задача.** Вывести на экран квадраты и кубы целых чисел от 8 до 1 (в обратном порядке).

**Особенность:** переменная цикла должна уменьшаться.

**Решение:**

```
for i:=8 downto 1 do begin
    i2 := i*i;
    i3 := i2*i;
    writeln(i:4, i2:4, i3:4);
end;
```

# Цикл с переменной

---

## Увеличение переменной на 1:

```
for <переменная> := <начальное значение>  
to  
    <конечное значение> do begin  
    {тело цикла}  
end;
```

## Уменьшение переменной на 1:

```
for <переменная> := <начальное значение>  
    downto  
    <конечное значение> do begin  
    {тело цикла}  
end;
```

# Цикл с переменной

---

## Особенности:

- переменная цикла может быть только целой (**integer**)
- шаг изменения переменной цикла всегда равен 1 (**to**) или -1 (**downto**)
- если в теле цикла только один оператор, слова **begin** и **end** можно не писать:

```
for i:=1 to 8 do  
    writeln( 'Привет' ) ;
```

- если конечное значение меньше начального, цикл (**to**) не выполняется ни разу (проверка условия в начале цикла, цикл с предусловием)

# Цикл с переменной

---

## Особенности:

- в теле цикла не разрешается изменять переменную цикла (почему?)
- при изменении начального и конечного значения внутри цикла количество шагов не изменится:

```
n := 8;  
for i:=1 to n do begin  
    writeln('Привет');  
    n := n + 1;  
end;
```

нет  
зацикливания

# Цикл с переменной

## Особенности:

- после выполнения цикла **во многих системах** устанавливается первое значение переменной цикла, при котором нарушено условие:

```
for i:=1 to 8  
  writeln('Привет');  
  writeln('i=', i);
```

i=9

НЕ ДОКУМЕНТИРОВАНО

```
for i:=8 downto 1 do  
  writeln('Привет');  
  writeln('i=', i);
```

i=0

# Сколько раз выполняется цикл?

---

```
a := 1;  
for i := 1 to 3 do a := a + 1;
```

~~a = 4~~

```
a := 1;  
for i := 3 to 1 do a := a + 1;
```

~~a = 1~~

```
a := 1;  
for i := 1 downto 3 do a := a + 1;
```

~~a = 1~~

```
a := 1;  
for i := 3 downto 1 do a := a + 1;
```

~~a = 4~~

# Как изменить шаг?

**Задача.** Вывести на экран квадраты и кубы нечётных целых чисел от 1 до 9.

**Особенность:** переменная цикла должна увеличиваться на 2.

**Проблема:** в Паскале шаг может быть 1 или -1.

**Решение:**

```
for i:=1 to 9 do begin
  if i mod 2 = 1 then begin
    i2 := i*i;
    i3 := i2*i;
    writeln(i:4, i2:4, i3:4);
  end;
end;
```

выполняется  
только для  
нечётных *i*



Что плохо?



# Как изменить шаг? – II

---

**Идея:** Надо вывести всего 5 чисел, переменная **k** изменяется от 1 до 5. Начальное значение **i** равно 1, с каждым шагом цикла **i** увеличивается на 2.

**Решение:**

```
i := 1;  
for k:=1 to 5 do begin  
    i2 := i*i;  
    i3 := i2*i;  
    writeln(i:4, i2:4, i3:4);  
    i := i + 2;  
end;
```

# Как изменить шаг? – III

**Идея:** Надо вывести всего 5 чисел, переменная **k** изменяется от 1 до 5. **Зная k, надо рассчитать i.**

k	1	2	3	4	5
i	1	3	5	7	9

$$i = 2k - 1$$

**Решение:**

```
for k:=1 to 5 do begin
  i := 2*k - 1;
  i2 := i*i;
  i3 := i2*i;
  writeln(i:4, i2:4, i3:4);
end;
```

# Задания

---

**"4":** Ввести *a* и *b* и вывести квадраты и кубы чисел от *a* до *b*.

**Пример:**

Введите границы интервала:

**4 6**

4    16    64

5    25    125

6    36    216

**"5":** Вывести квадраты и кубы 10 чисел следующей последовательности: 1, 2, 4, 7, 11, 16, ...

**Пример:**

1            1            1

2            4            8

4            16           64

...

46    2116   97336

# Программирование на языке Паскаль

## Тема 5. Циклы с условием

# Цикл с неизвестным числом шагов

---

**Пример:** Отпилить полено от бревна. Сколько раз надо сделать движения пилой?

**Задача:** Ввести целое число ( $< 2000000$ ) и определить число цифр в нем.

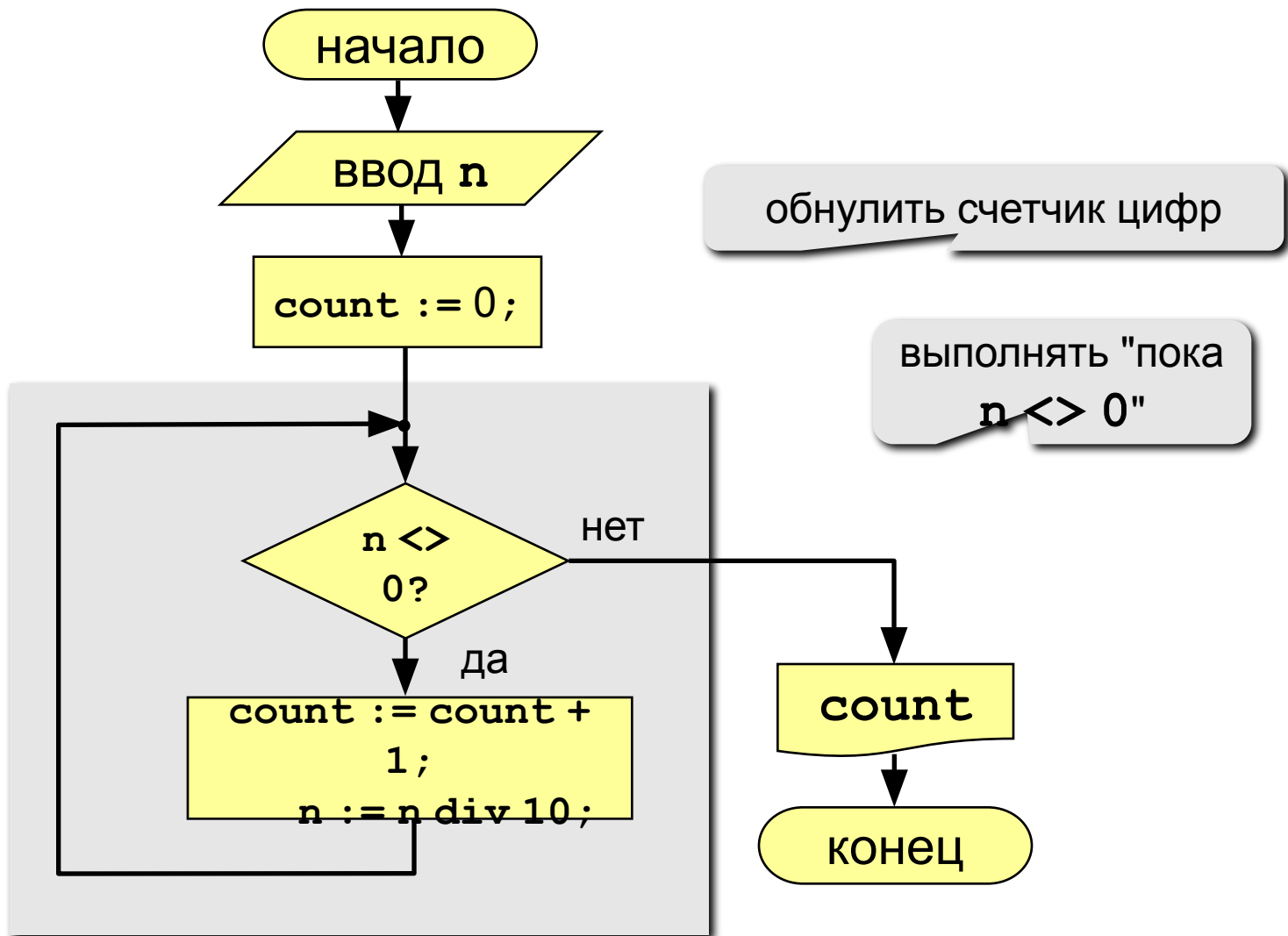
**Идея решения:** Отсекаем последовательно последнюю цифру, увеличиваем счетчик.

n	count
123	0
12	1
1	2
0	3

**Проблема:** Неизвестно, сколько шагов надо сделать.

**Решение:** Надо остановиться, когда  $n = 0$ , т.е. надо делать "пока  $n \neq 0$ ".

# Алгоритм



# Программа

```
program qq;  
var n, count, n1: integer;  
begin  
    writeln('Введите целое число');  
    read(n); n1 := n;  
    count := 0;  
  
    while n <> 0 do begin  
        count := count + 1;  
        n := n div 10;  
    end;  
    writeln('В числе ', n1, ' нашли ',  
           count, ' цифр');  
end.
```

выполнять "пока  
 $n \neq 0$ "



Что плохо?

# Цикл с условием

---

```
while <условие> do begin  
    {тело цикла}  
end;
```

## Особенности:

- МОЖНО ИСПОЛЬЗОВАТЬ СЛОЖНЫЕ УСЛОВИЯ:

```
while (a<b) and (b<c) do begin  
    {тело цикла}  
end;
```

- если в теле цикла только один оператор, слова **begin** и **end** можно не писать:

```
while a < b do  
    a := a + 1;
```



# Цикл с условием

---

## Особенности:

- условие пересчитывается **каждый раз** при входе в цикл
- если условие на входе в цикл ложно, цикл не выполняется ни разу

```
a := 4; b := 6;  
while a > b do  
    a := a - b;
```

- если условие никогда не станет ложным, программа **зацикливается**

```
a := 4; b := 6;  
while a < b do  
    d := a + b;
```

# Сколько раз выполняется цикл?

```
a := 4; b := 6;  
while a < b do a := a + 1;
```

2 раза

~~a~~ = 6

```
a := 4; b := 6;  
while a < b do a := a + b;
```

1 раз

~~a~~ = 10

```
a := 4; b := 6;  
while a > b do a := a + 1;
```

0 раз

~~a~~ = 4

```
a := 4; b := 6;  
while a < b do b := a - b;
```

1 раз

~~b~~ = -2

```
a := 4; b := 6;  
while a < b do a := a - 1;
```

зацикливание

# Замена for на while и наоборот

```
for i:=1 to 10 do begin  
    {тело цикла}  
end;
```

```
i := 1;  
while i <= 10 do begin  
    {тело цикла}  
    i := i + 1;  
end;
```

```
for i:=a downto b do  
begin  
    {тело цикла}  
end;
```

```
i := a;  
while i >= b do begin  
    {тело цикла}  
    i := i - 1;  
end;
```

Замена цикла **for** на **while** возможна **всегда**.

Замена **while** на **for** возможна только тогда, когда можно заранее **рассчитать число шагов цикла**.

# Задания

---

**"4": Ввести целое число и найти сумму его цифр.**

**Пример:**

Введите целое число:

**1234**

Сумма цифр числа 1234 равна 10.

**"5": Ввести целое число и определить, верно ли, что в его записи есть две одинаковые цифры.**

**Пример:**

Введите целое число:

**1234**

Нет.

Введите целое число:

**1224**

Да.

# Последовательности

## Примеры:

• 1, 2, 3, 4, 5, ...

$$a_n = n$$

$$a_1 = 1, \quad a_{n+1} = a_n + 1$$

• 1, 2, 4, 7, 11, 16, ...

$$a_1 = 1, \quad a_{n+1} = a_n + n$$

• 1, 2, 4, 8, 16, 32, ...

$$a_n = 2^{n-1}$$

$$a_1 = 1, \quad a_{n+1} = 2a_n$$

•  $\frac{1}{2}, \frac{1}{2}, \frac{3}{8}, \frac{1}{4}, \frac{5}{32}, \dots$

$\frac{1}{2}, \frac{2}{4}, \frac{3}{8}, \frac{4}{16}, \frac{5}{32}, \dots$

$$a_n = \frac{b_n}{c_n}$$

$$b_1 = 1, \quad b_{n+1} = b_n + 1$$

$$c_1 = 2, \quad c_{n+1} = 2c_n$$

# Последовательности

**Задача:** найти сумму всех элементов последовательности,

$$1, -\frac{1}{2}, \frac{2}{4}, -\frac{3}{8}, \frac{4}{16}, -\frac{5}{32}, \dots$$

которые по модулю больше 0,001:

$$S = 1 - \frac{1}{2} + \frac{2}{4} - \frac{3}{8} + \frac{4}{16} - \frac{5}{32} + \dots$$

**Элемент последовательности (начиная с №2):**

$$a = z \frac{b}{c}$$

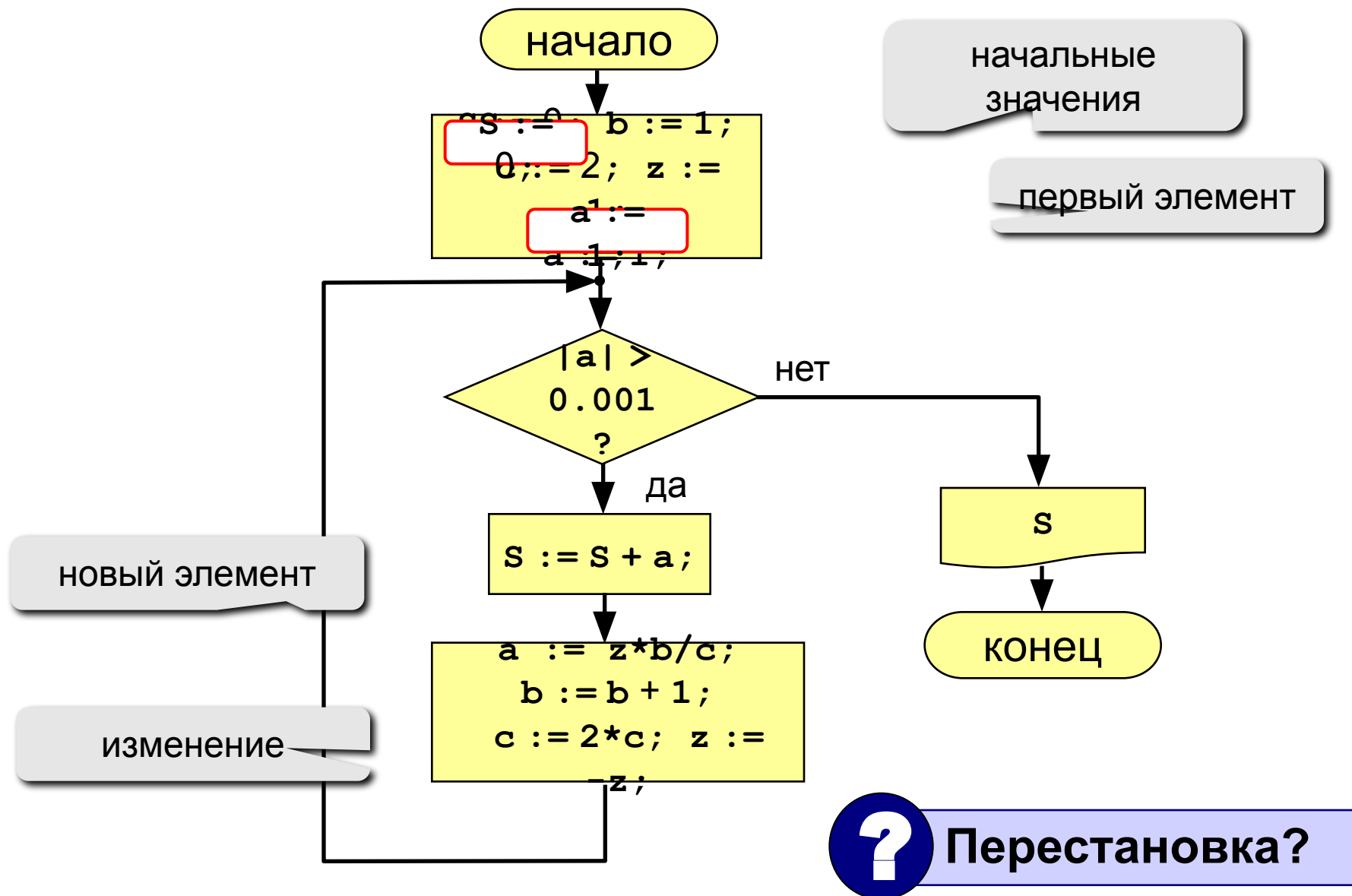
n	1	2	3	4	5	...
b	1	2	3	4	5	...
c	2	4	8	16	32	...
z	-1	1	-1	1	-1	...

`b := b+1;`

`c := 2*c;`

`z := -z;`

# Алгоритм



# Программа

```
program qq;  
var b, c, z: integer;  
    S, a: real;  
begin  
    S := 0; z := -1;  
    b := 1; c := 2; a := 1;  
    while abs(a) > 0.001 do begin  
        S := S + a;  
        a := z * b / c;  
        z := - z;  
        b := b + 1;  
        c := c * 2;  
    end;  
    writeln('S =', S:10:3);  
end.
```

начальные  
значения

увеличение  
суммы

расчет элемента  
последовательности

переход к  
следующему  
слагаемому



# Задания

---

**"4":** Найти сумму элементов последовательности с точностью 0,001:

$$S = 1 + \frac{2}{3 \cdot 3} - \frac{4}{5 \cdot 9} + \frac{6}{7 \cdot 27} - \frac{8}{9 \cdot 81} + \dots$$

**Ответ:**

$$S = 1.157$$

**"5":** Найти сумму элементов последовательности с точностью 0,001:

$$S = 1 + \frac{2}{2 \cdot 3} - \frac{4}{3 \cdot 9} + \frac{6}{5 \cdot 27} - \frac{8}{8 \cdot 81} + \frac{10}{13 \cdot 243} - \dots$$

**Ответ:**

$$S = 1.220$$

# Цикл с постусловием

---

**Задача:** Ввести целое **положительное** число (<2000000) и определить число цифр в нем.

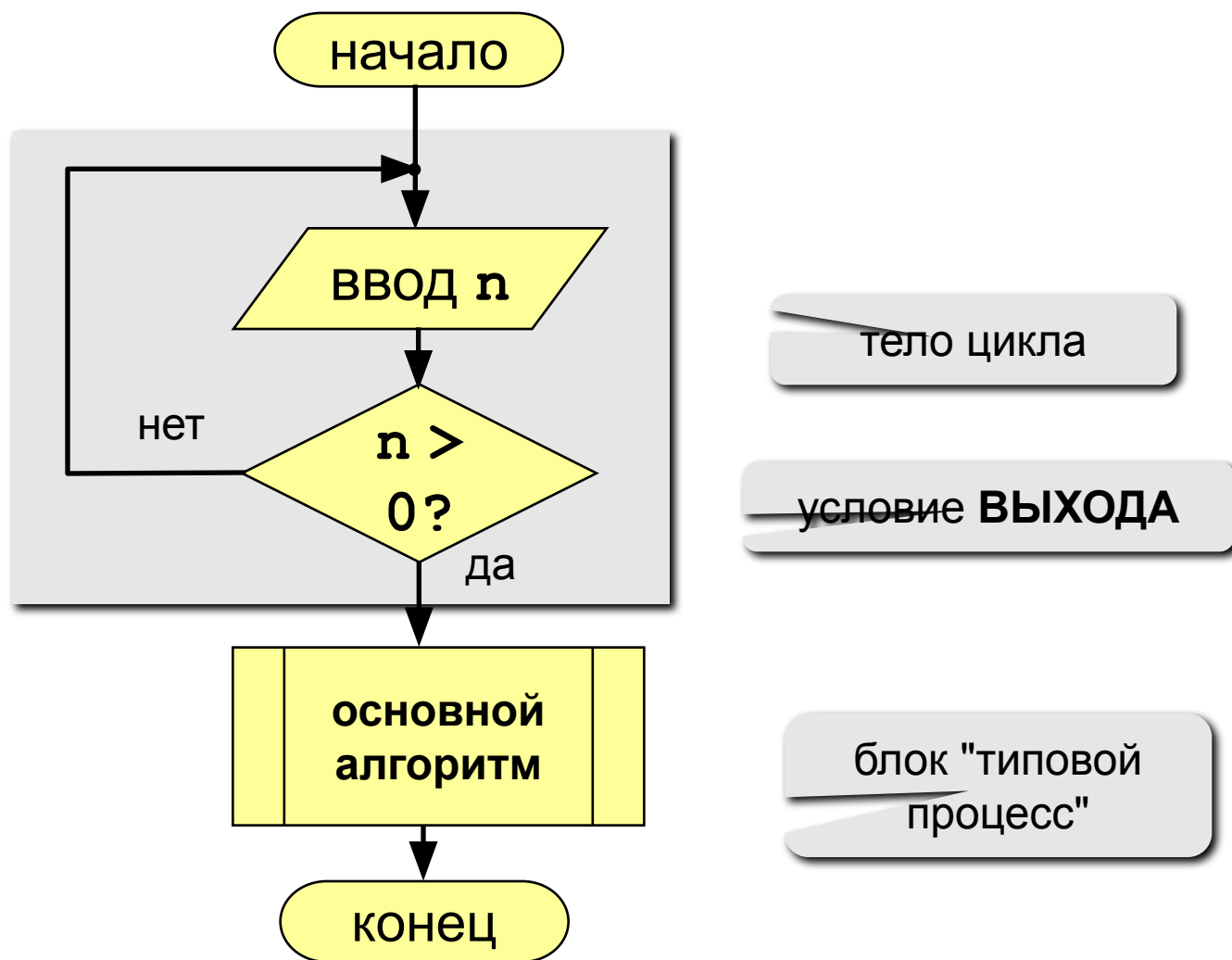
**Проблема:** Как не дать ввести отрицательное число или ноль?

**Решение:** Если вводится неверное число, вернуться назад к вводу данных (цикл!).

**Особенность:** Один раз тело цикла надо сделать в любом случае => проверку условия цикла надо делать в конце цикла (цикл с **постусловием**).

**Цикл с постусловием** – это цикл, в котором проверка условия выполняется в конце цикла.

# Цикл с постусловием: алгоритм



# Программа

```
program qq;  
var n: integer;  
begin  
    repeat  
        writeln('Введите положительное число');  
        read(n);  
        until n > 0;  
    ... { основной алгоритм }  
end.
```

условие **ВЫХОДА**

## Особенности:

- тело цикла всегда выполняется хотя бы один раз
- после слова **until** ("до тех пор, пока не...") ставится условие **ВЫХОДА** из цикла

# Сколько раз выполняется цикл?

```
a := 4; b := 6;  
repeat a := a + 1; until a > b;
```

3 раза

a = 7

```
a := 4; b := 6;  
repeat a := a + b; until a > b;
```

1 раз

a = 10

```
a := 4; b := 6;  
repeat a := a + b; until a < b;
```

**зацикливание**

```
a := 4; b := 6;  
repeat b := a - b; until a < b;
```

2 раза

b = 6

```
a := 4; b := 6;  
repeat a := a + 2; until a < b;
```

**зацикливание**

# Задания (с защитой от неверного ввода)

---

**"4":** Ввести натуральное число и определить, верно ли, что сумма его цифр равна 10.

**Пример:**

Введите число  $\geq 0$ :

**-234**

Нужно положительное число. Нет

Введите число  $\geq 0$ :

**1234**

Да

Введите число  $\geq 0$ :

**1233**

Нет

**"5":** Ввести натуральное число и определить, какие цифры встречаются несколько раз.

**Пример:**

Введите число  $\geq 0$ :

**2323**

Повторяются: 2, 3

Введите число  $\geq 0$ :

**1234**

Нет повторов.

# Программирование на языке Паскаль

## Тема 6. Оператор выбора

# Оператор выбора

---

**Задача:** Ввести номер месяца и вывести количество дней в этом месяце.

**Решение:** Число дней по месяцам:

**28 дней** – 2 (февраль)

**30 дней** – 4 (апрель), 6 (июнь), 9 (сентябрь), 11 (ноябрь)

**31 день** – 1 (январь), 3 (март), 5 (май), 7 (июль),  
8 (август), 10 (октябрь), 12 (декабрь)

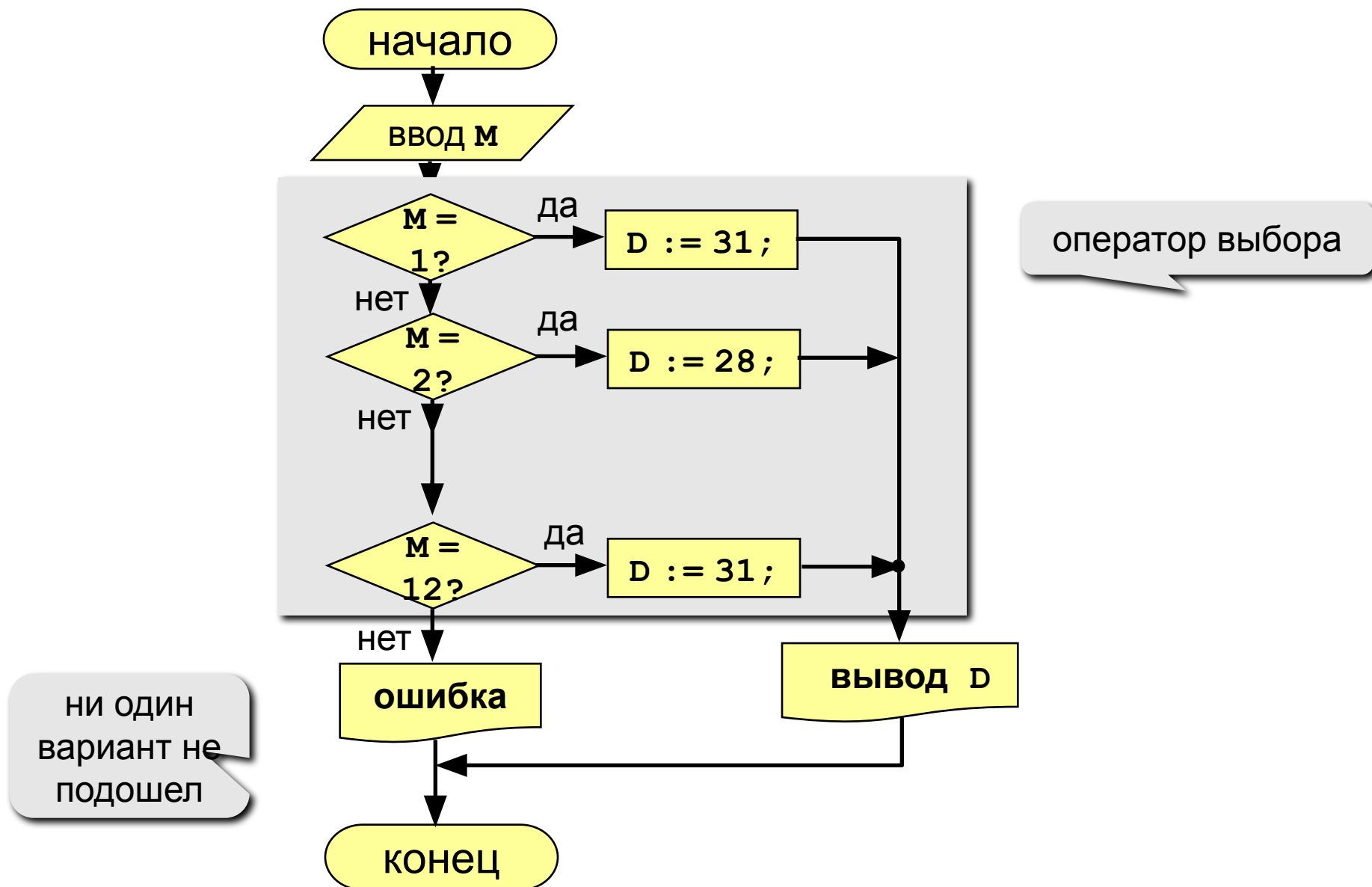
**Особенность:** Выбор не из двух, а из нескольких вариантов в зависимости от номера месяца.



**Можно ли решить известными методами?**



# Алгоритм



# Программа

```
program qq;  
var M, D: integer;  
begin  
    writeln('Введите номер месяца:');  
    read ( M );  
  
    case M of  
        2:          begin D := 28; end;  
        4,6,9,11:   begin D := 30; end;  
        1,3,5,7,8,10,12: D := 31;  
        else          D := -1;  
    end;  
  
    if D > 0 then  
        writeln('В этом месяце ', D, ' дней.')    else  
        writeln('Неверный номер месяца');  
    end.  
end.
```

ни один вариант не подошел

# Оператор выбора

---

## Особенности:

- после **case** может быть имя переменной или арифметическое выражение целого типа (**integer**)

```
case i+3 of
  1: begin a := b; end;
  2: begin a := c; end;
end;
```

или СИМВОЛЬНОГО ТИПА (**char**)

```
var c: char;
...
case c of
  'a': writeln('Антилопа');
  'б': writeln('Барсук');
  else writeln('Не знаю');
end;
```

# Оператор выбора

---

## Особенности:

- если нужно выполнить только один оператор, слова **begin** и **end** можно не писать

```
case i+3 of  
  1: a := b;  
  2: a := c;  
end;
```

- нельзя ставить два одинаковых значения

```
case i+3 of  
  1: a := b;  
  1: a := c;  
end;
```

# Оператор выбора

## Особенности:

- значения, при которых выполняются одинаковые действия, можно группировать

перечисление

диапазон

смесь

```
case i of
  1:                a := b;
  2,4,6:            a := c;
  10..15:           a := d;
  20,21,25..30:    a := e;
  else writeln('Ошибка');
end;
```

# Что неправильно?

```
case a of
  2: begin a := b;
  4: a := c;
end;
```

```
case a of
  2: a := b;
  4: a := c;
end;
```

```
case a of
  2..5: a := b;
  4: a := c;
end;
```

```
case a of
  0..2: a := b;
  3..6: a := c;
end;
```

```
case a+c/2 of
  2: a := b;
  4: a := c;
end;
```

```
begin
case a of
  2: a := b; d := 0; end;
  4: a := c;
end;
```

# Задания (с защитой от неверного ввода)

---

**"4":** Ввести номер месяца и вывести количество дней в нем, а также число ошибок при вводе.

**Пример:**

Введите номер месяца:

-2

Введите номер месяца:

11

В этом месяце 30 дней.

Вы вводили неверно 1 раз.

Введите номер месяца:

2

В этом месяце 28 дней.

Вы вводили неверно 0 раз.

**"5":** Ввести номер месяца и номер дня, вывести число дней, оставшихся до Нового года.

**Пример:**

Введите номер месяца:

12

Введите день:

25

До Нового года осталось 6 дней.

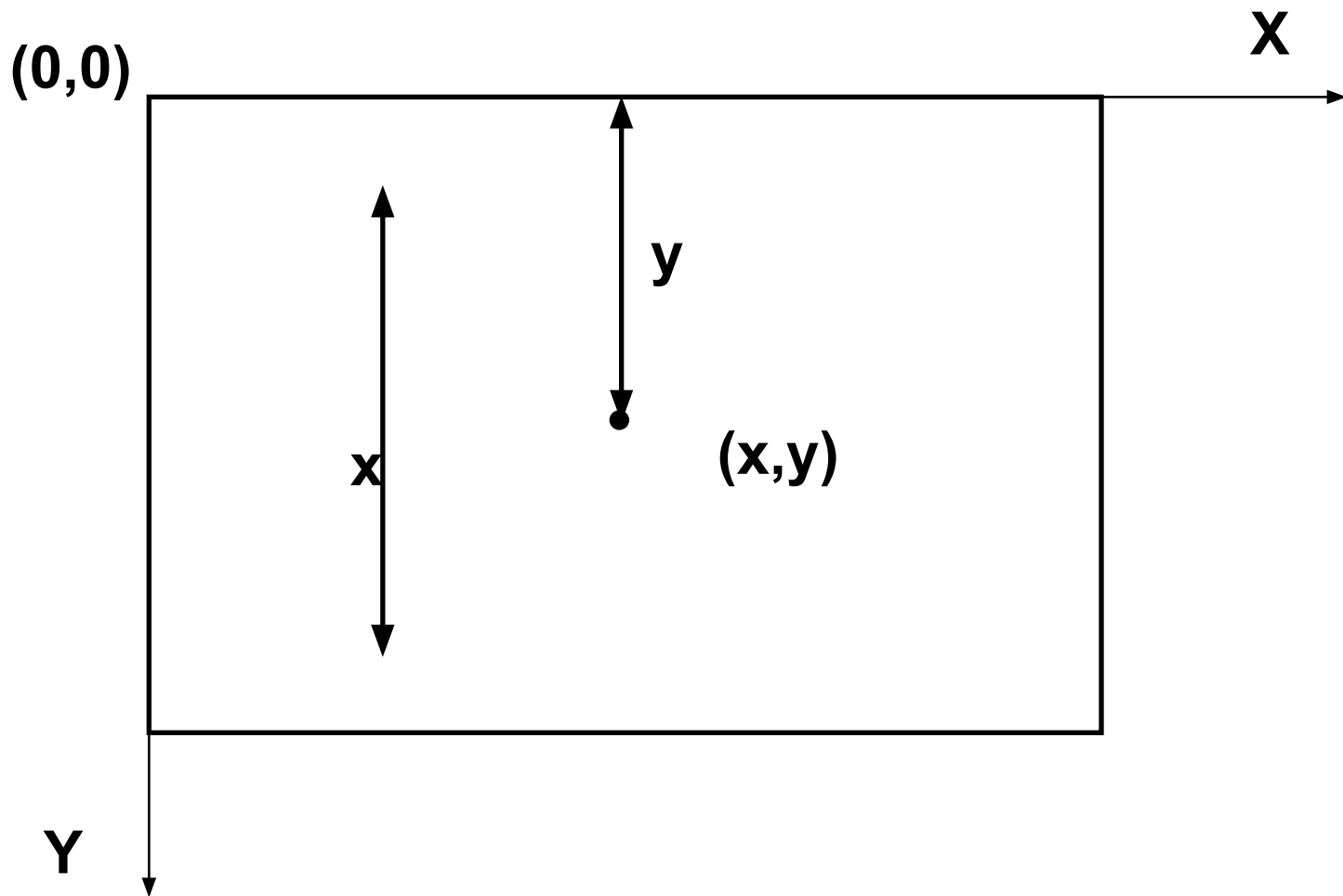
# Программирование на языке Паскаль

## 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);
```

$(x_1, y_1)$



$(x_2, y_2)$



$(x_5, y_5)$



$(x_3, y_3)$

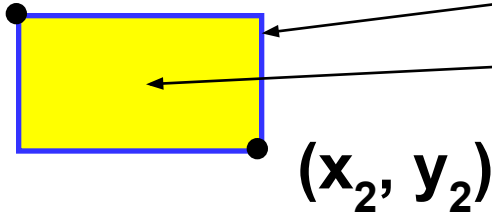


$(x_4, y_4)$

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

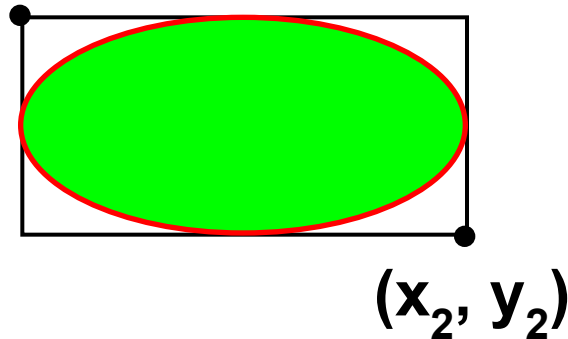
# Фигуры с заливкой

$(x_1, y_1)$

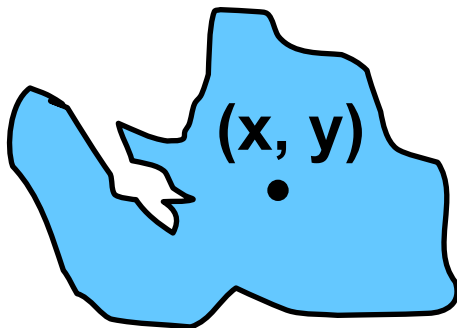


```
Pen (1, 0, 0, 255);  
Brush (1, 255, 255, 0);  
Rectangle (x1, y1, x2, y2);
```

$(x_1, y_1)$



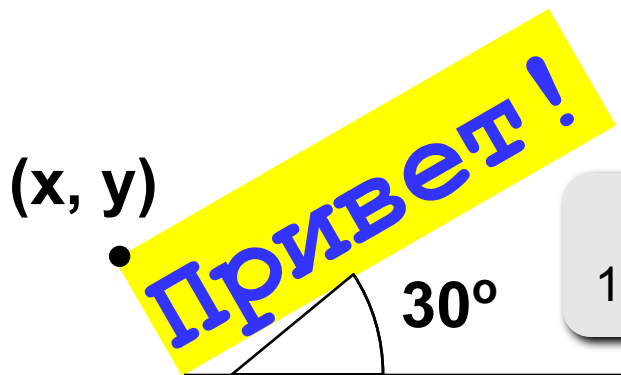
```
Pen (1, 255, 0, 0);  
Brush (1, 0, 255, 0);  
Ellipse (x1, y1, x2, y2);
```



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);
```

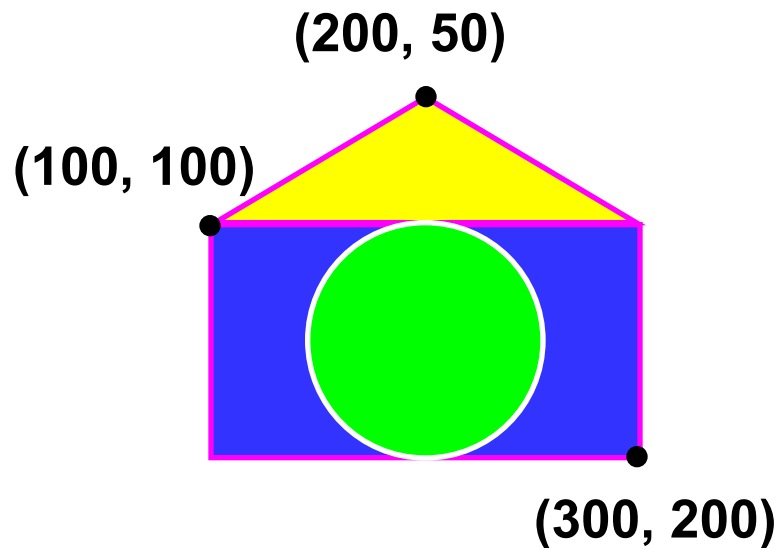
размер  
10 пикселей

угол  
поворота

насыщенность:  
400 – нормальный  
600 – жирный

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

# Пример

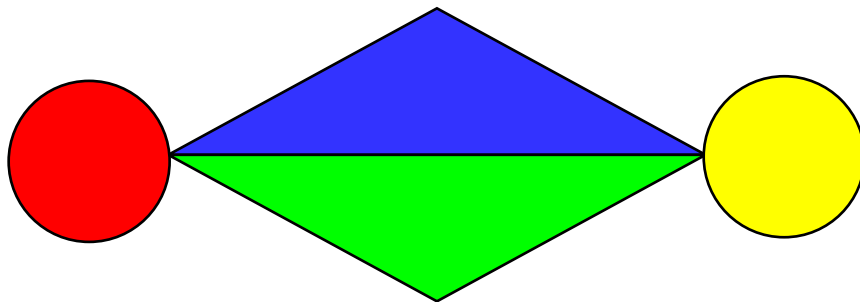


```
program qq;  
begin  
    Pen(2, 255, 0, 255);  
    Brush(1, 0, 0, 255);  
    Rectangle(100, 100, 300, 200);  
    MoveTo(100, 100);  
    LineTo(200, 50);  
    LineTo(300, 100);  
    Brush(1, 255, 255, 0);  
    Fill(200, 75);  
    Pen(2, 255, 255, 255);  
    Brush(1, 0, 255, 0);  
    Ellipse(150,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

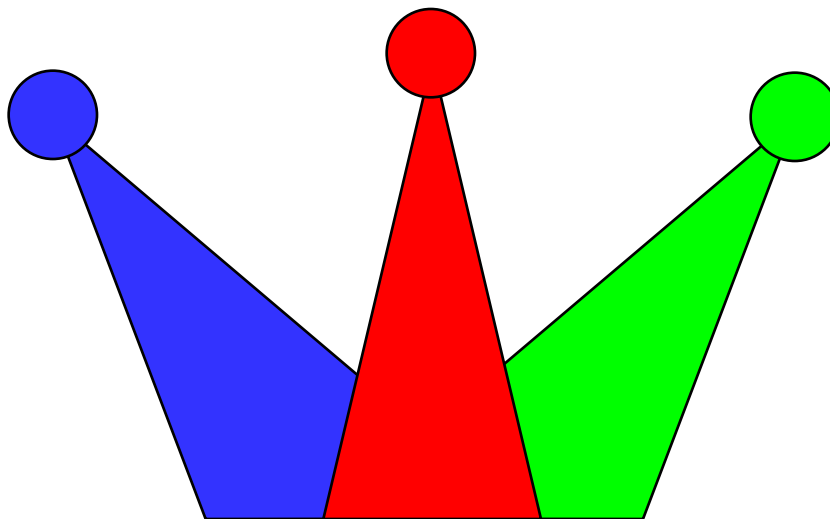
# Задания

---

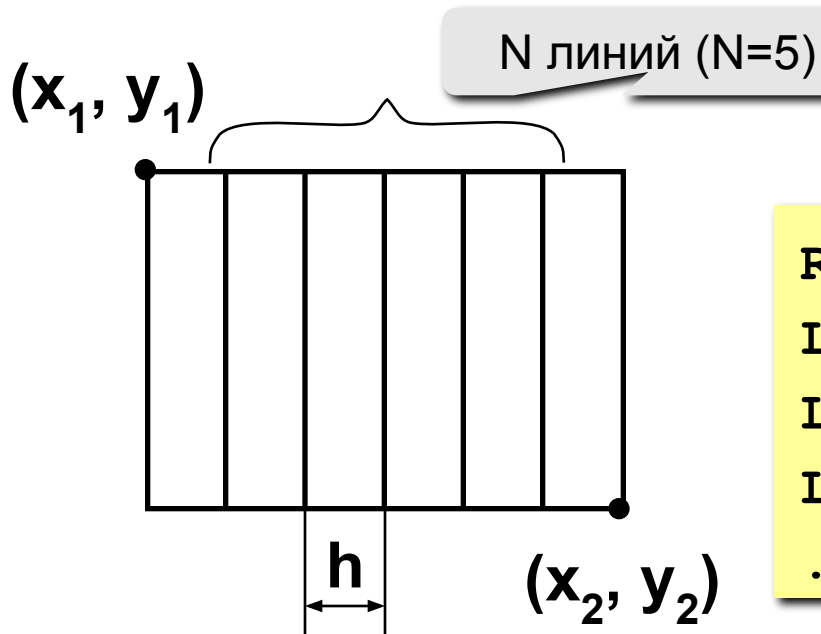
"4": "Лягушка"



"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);
...
```

x

x

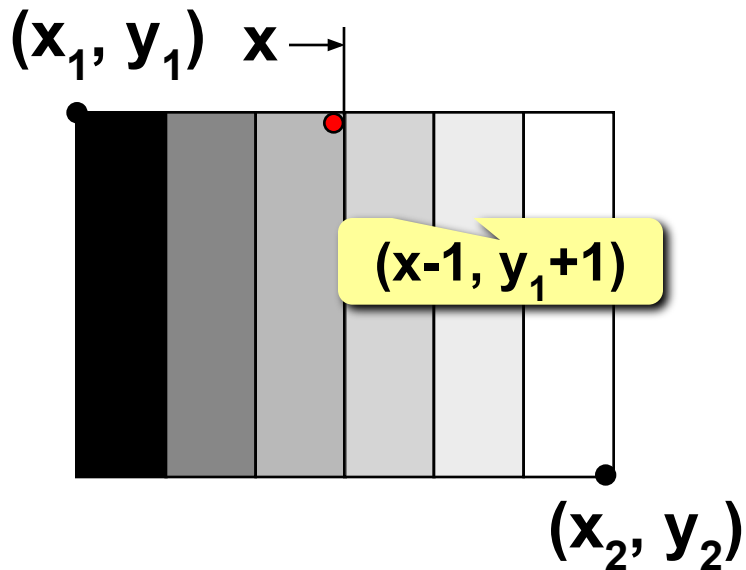
```
h := (x2 - x1) / (N + 1);
Rectangle (x1, y1, x2, y2);
x := x1 + h;
for i:=1 to N do begin
    Line( round(x), y1, round(x), y2);
    x := x + h;
end;
```

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);
```

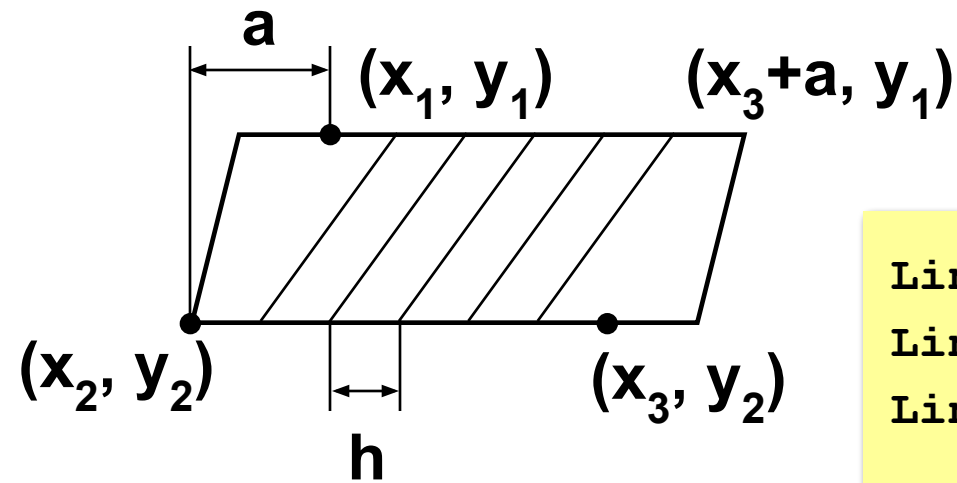
```
  Fill (round(x)-1, y1+1);
```

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

```
end;
```

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

# Штриховка



$$a = x_1 - x_2$$

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

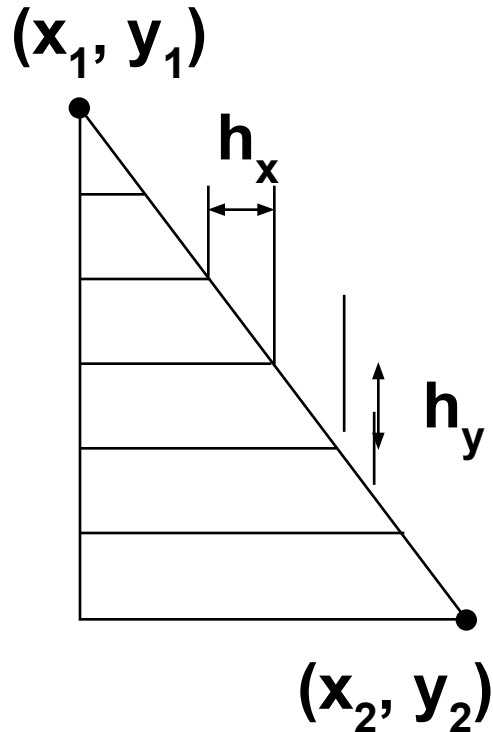
```
Line( x1+h, y1, x1+h-a, y2);
Line( x1+2*h, y1, x1+2*h-a, y2);
Line( x1+3*h, y1, x1+3*h-a, y2);
...
```

**x**

**x-a**

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

# Штриховка



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

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

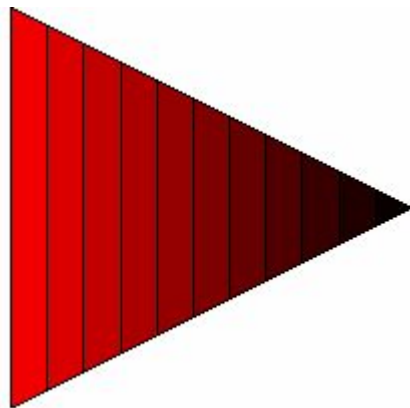
```
Line( x1, y1+hy, x1+hx, y1+hy)
;
Line( x1, y1+2*hy, x1+2*hx,
y1+2*hy);
Line( x1, y1+3*hy, x1+3*hx,
y1+3*hy);
```

```
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;
```

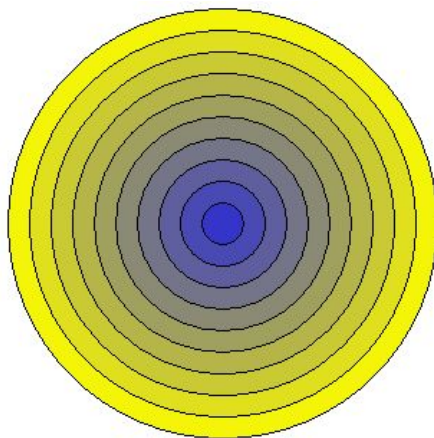
# Задания

---

**"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:**

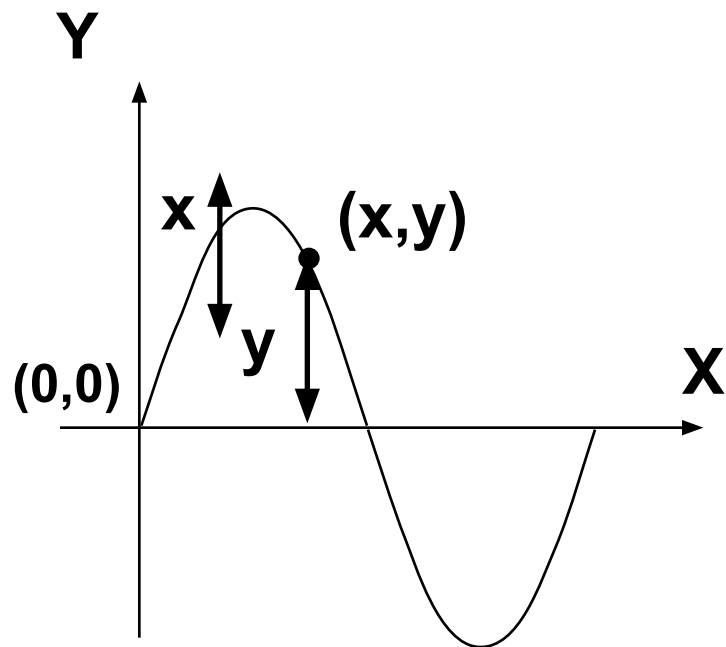
максимальное значение  $y_{\max} = 3$  при  $x = \pi/2$

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

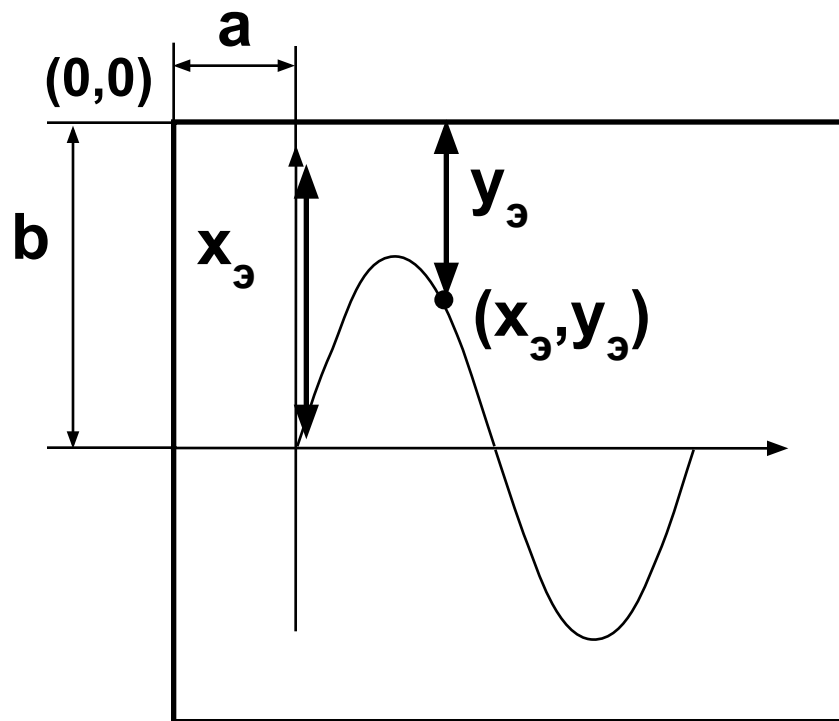
**Проблема:** функция задана в математической системе координат, строить надо на экране, указывая координаты в пикселях.

# Преобразование координат

Математическая  
система координат



Экранная система  
координат (пиксели)



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

$$x_э = a + kx$$

$$y_э = b - ky$$

# 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.
```

2π

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:

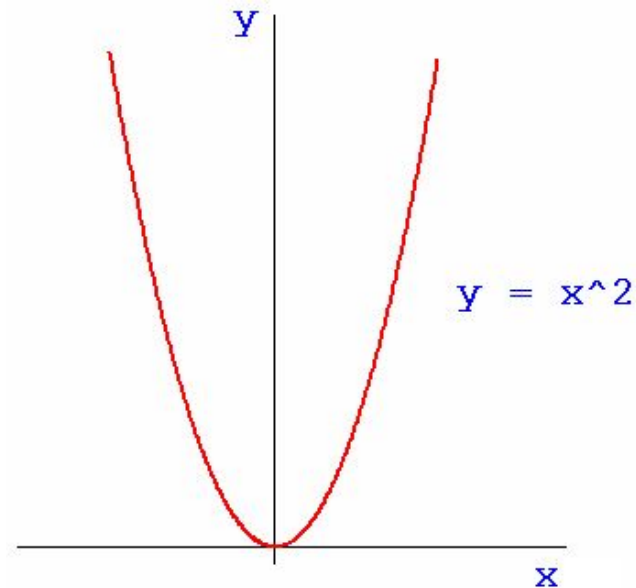
```
var first: boolean;  
...  
begin  
...  
first := True;  
while x <= xmax do begin  
...  
if first then begin  
    MoveTo(xe, ye);  
    first := False;  
end  
else LineTo(xe, ye);  
...  
end;  
end.
```

логическая  
переменная

начальное значение

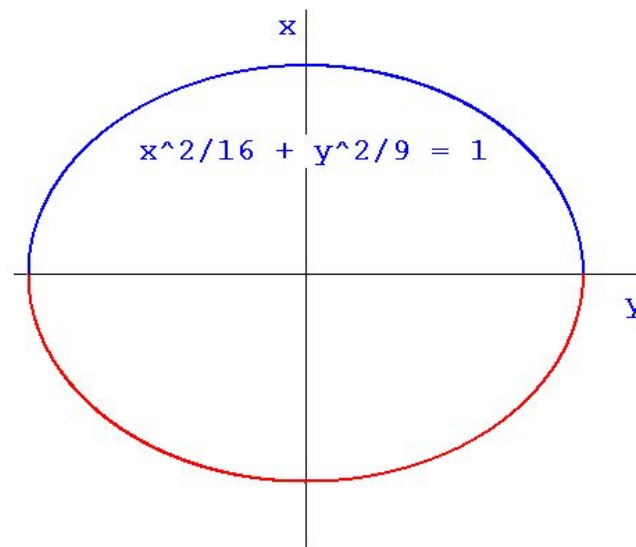
# Задания

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



**"5":** Построить график функции (эллипс)

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



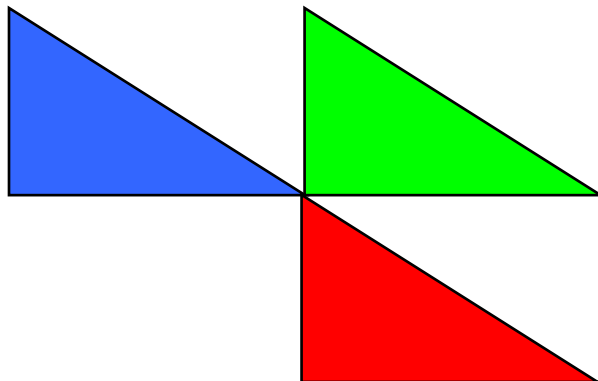
# Программирование на языке Паскаль

## Тема 9. Процедуры

# Процедуры

---

**Задача:** Построить фигуру:



**Можно ли решить известными методами?**

**Общность:** три похожие фигуры.

**общее:** размеры, угол поворота

**отличия:** координаты, цвет



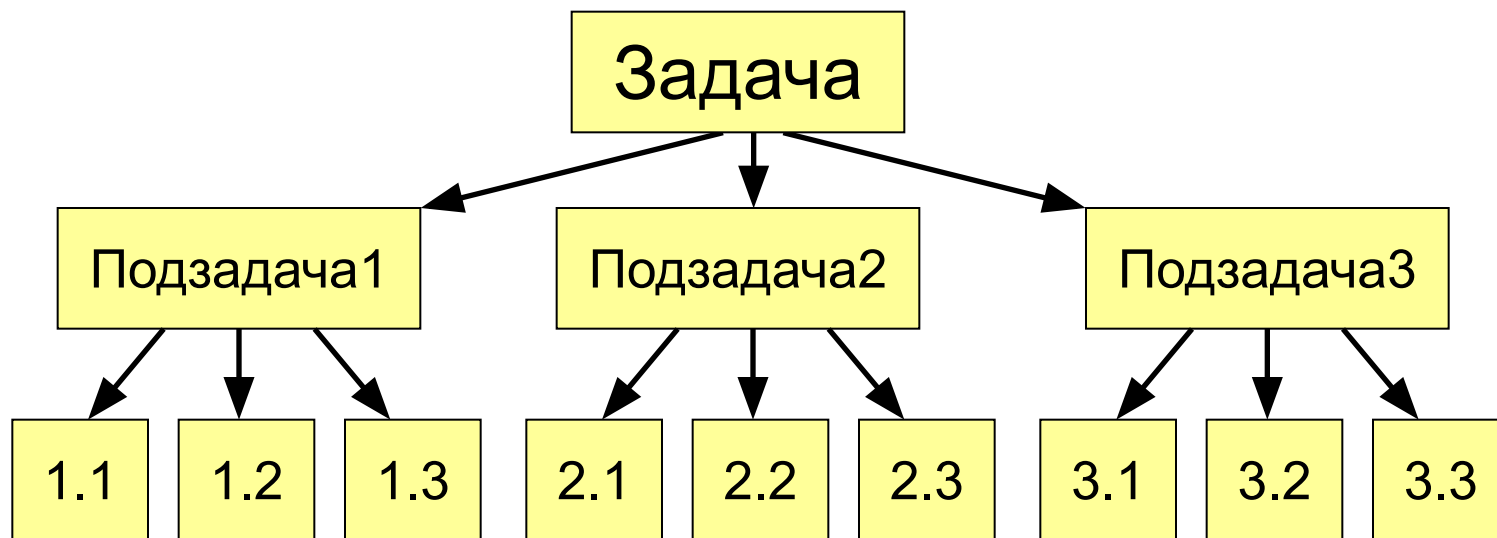
**Сколько координат надо задать?**

# Процедуры

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

## Применение:

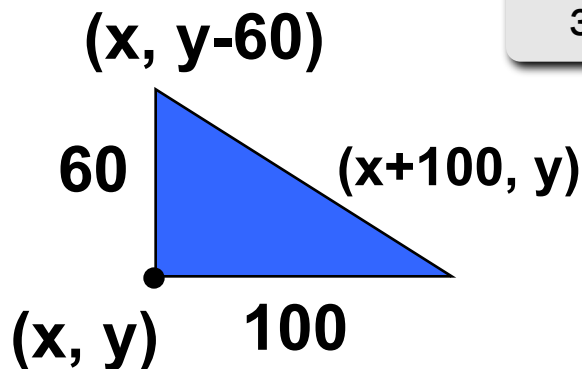
- выполнение одинаковых действий в разных местах программы
- разбивка программы (или другой процедуры) на подзадачи для лучшего восприятия



# 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



заголовок

параметры

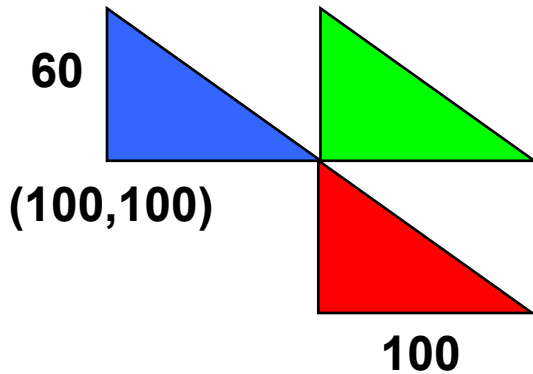
```
procedure Tr(  $x, y, r, g, b$ : integer);  
begin  
    MoveTo( $x, y$ );  
    LineTo( $x, y-60$ );  
    LineTo( $x+100, y$ );  
    LineTo( $x, y$ );  
    Brush(1,  $r, g, b$ );  
    Fill( $x+20, y-20$ );  
end;
```

цвет

координаты

тело 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,

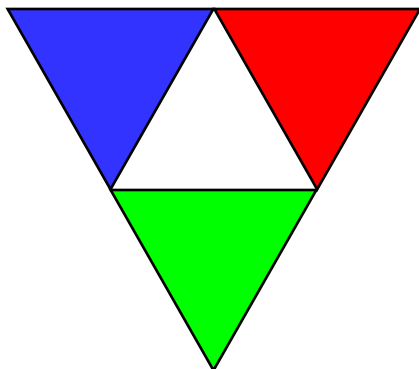
## Запрещенные варианты вызова

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

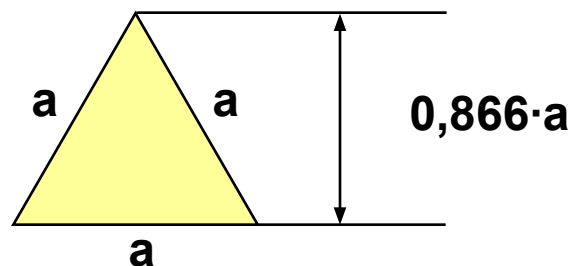
Exchange ( ~~x+2~~, ~~y+2~~ );      { выражения }

# Задания

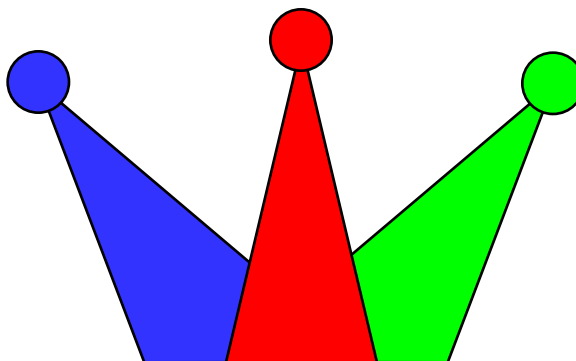
"4": Используя процедуры, построить фигуру.



равносторонний треугольник



"5": Используя процедуры, построить фигуру.



# Программирование на языке Паскаль

## Тема 10. Рекурсия

# Рекурсивные объекты

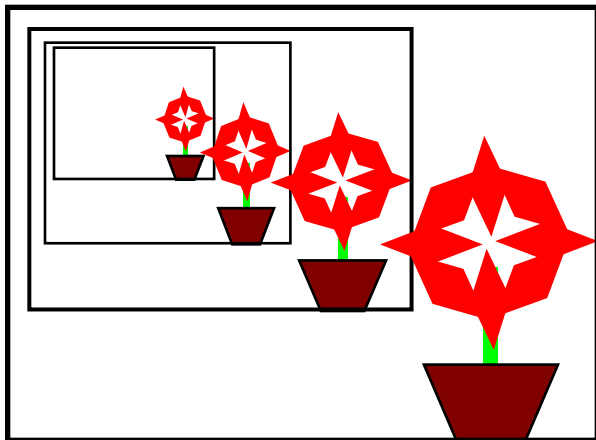
## Примеры:

### 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:

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

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

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

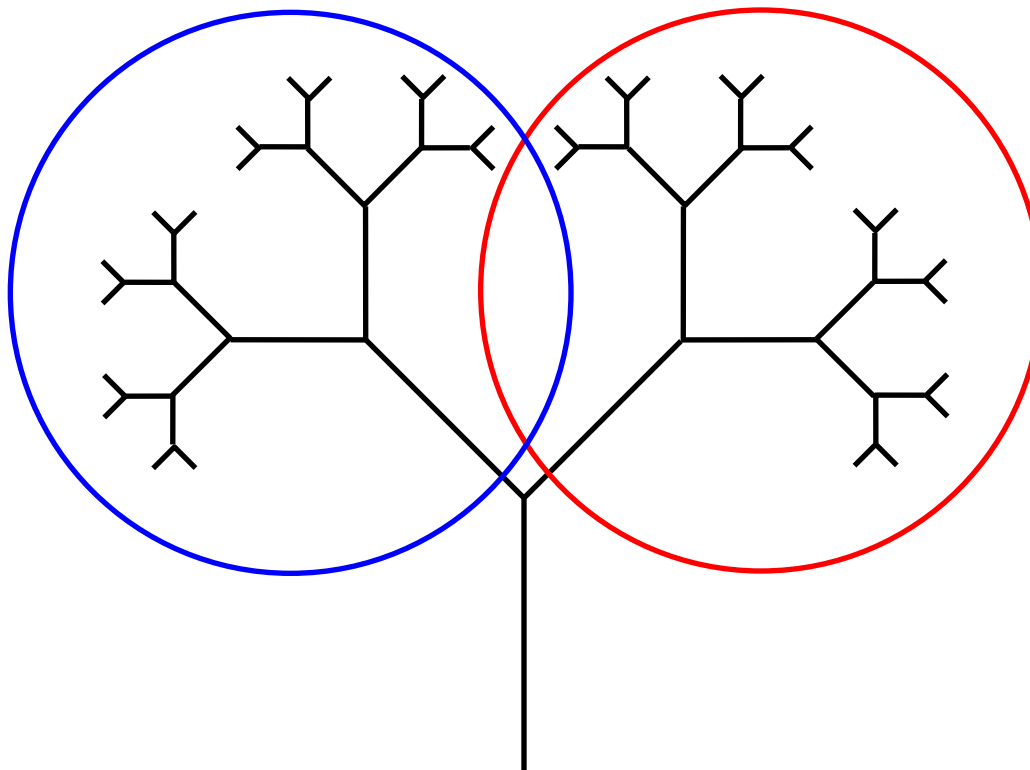
$$N! = N \cdot (N-1) \cdot \square \cdot 2 \cdot 1$$

**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?



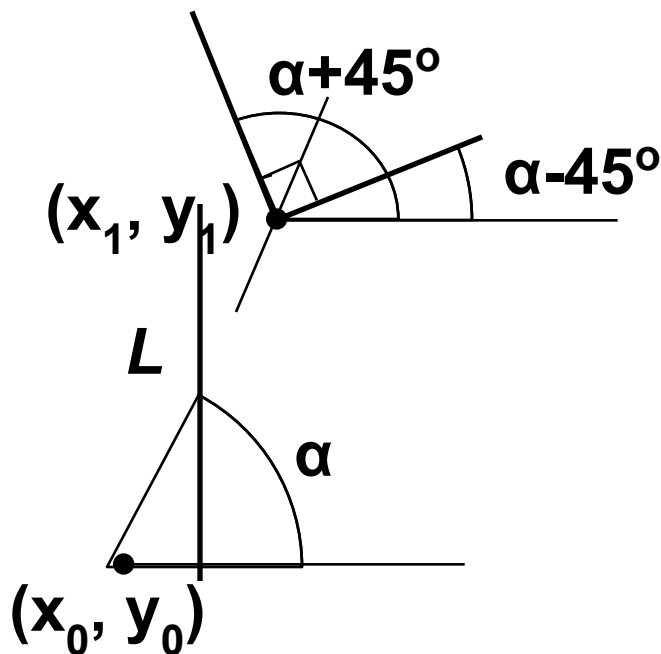
# Дерево Пифагора

## Особенности:

- когда остановиться?

**когда число оставшихся уровней станет равно нулю!**

- деревья имеют различный наклон



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

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

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



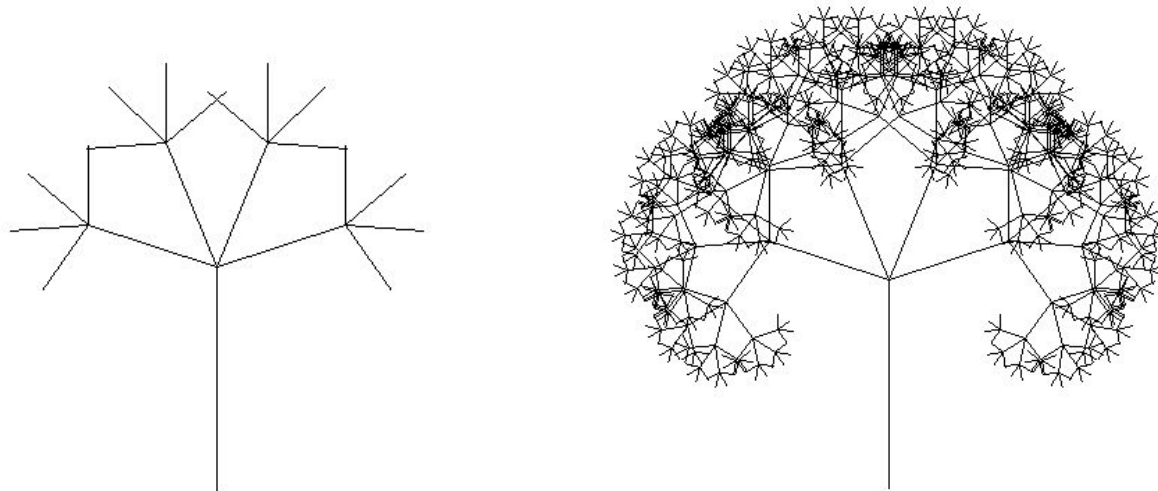
Как наклонить дерево вправо на  $30^\circ$ ?

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

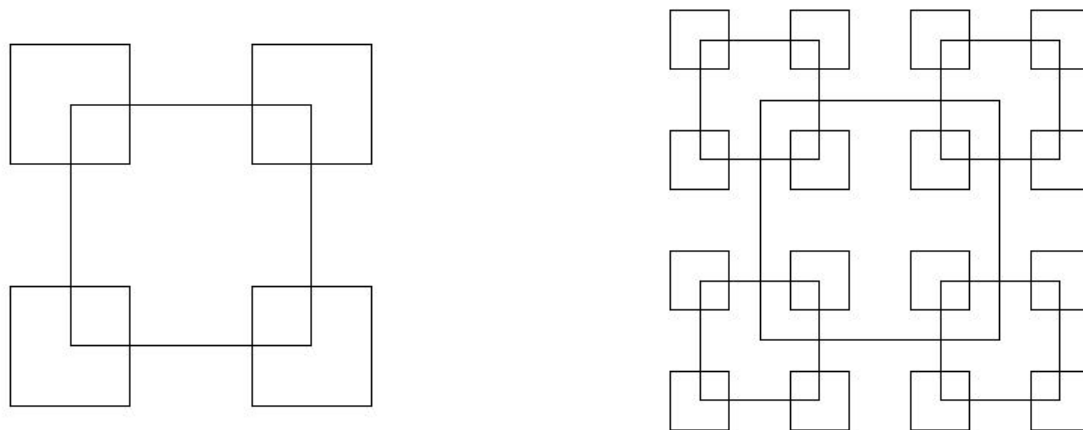
# Задания

---

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



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



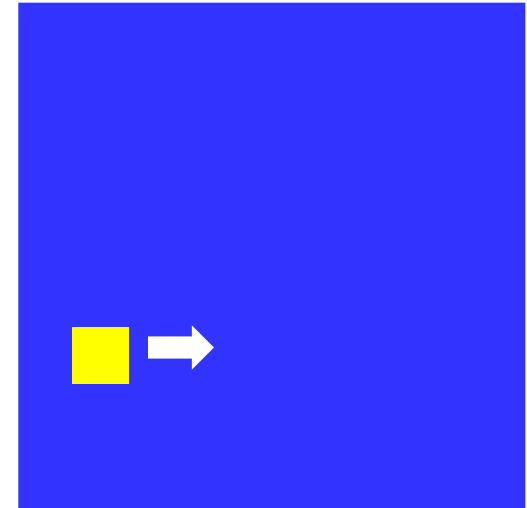
# Программирование на языке Паскаль

## 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

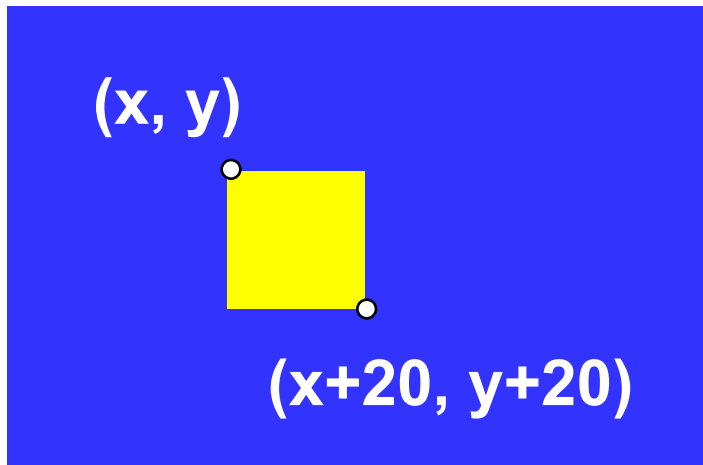
если что-то произошло...

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*

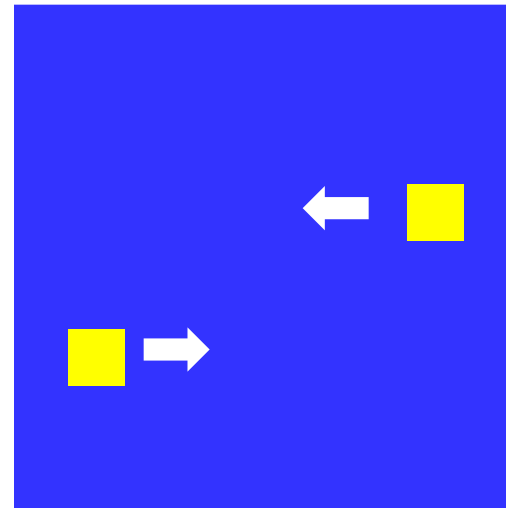
ждем 10 мс

выход при касании  
границы

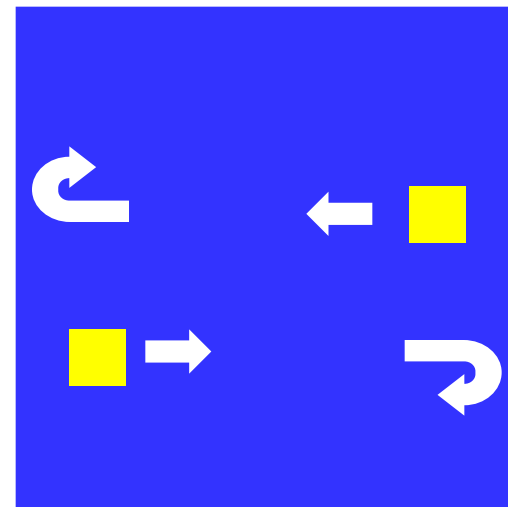
# Задания

---

**"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 перерисовывать 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
```

```
    ...
```

рисует квадрат

```
repeat
```

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

```
    while not IsEvent do;
```

ждем события

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

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

```
    ...
```

```
until stop;
```

только теперь стираем

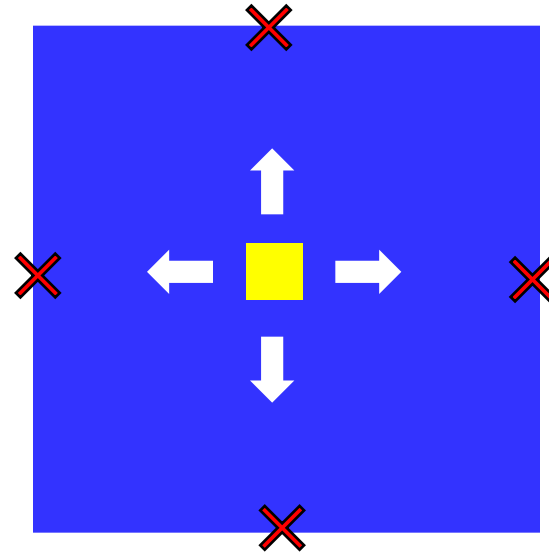
```
end.
```



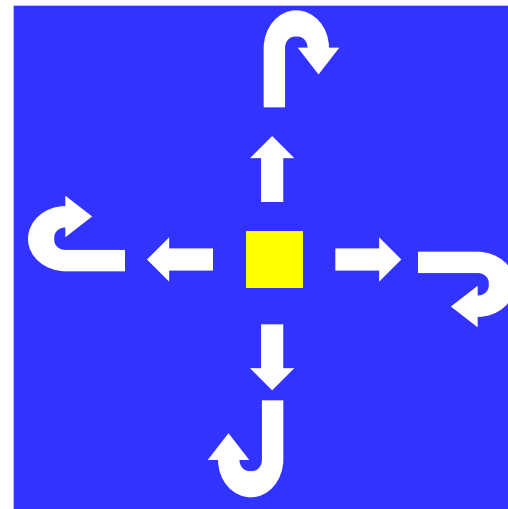
Что можно улучшить?

# Задания

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



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



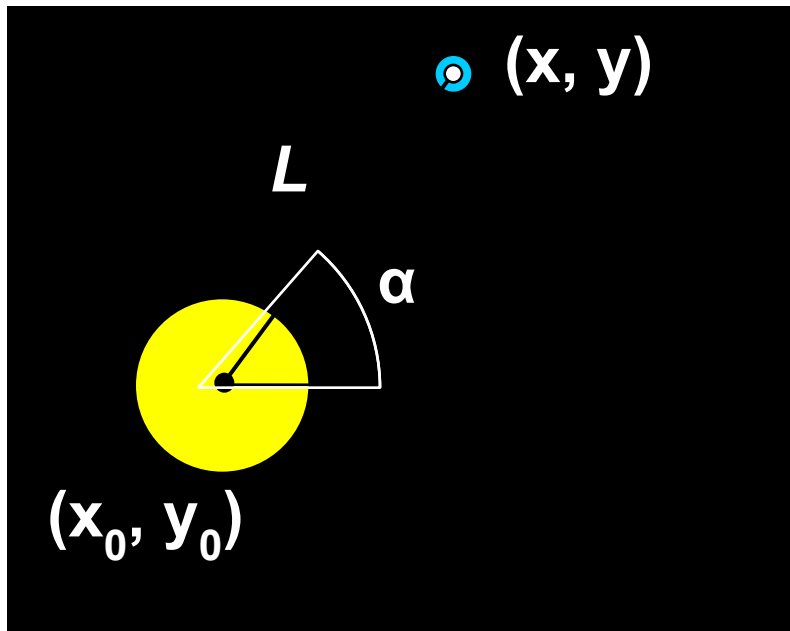


# Вращение

**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**)?

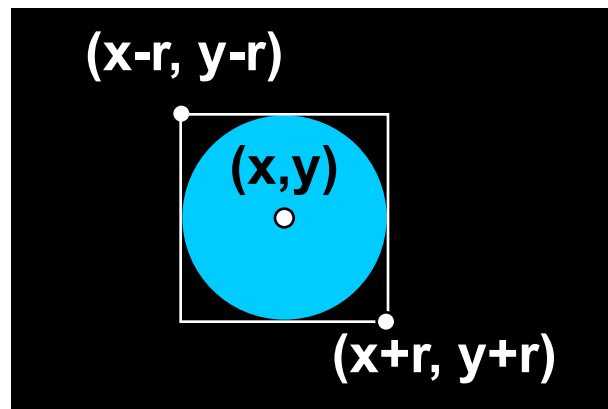
```
procedure Draw(x, y: integer; flag: boolean);  
const r = 10;  
begin  
    if flag then  
        Brush(1, 100, 100, 255)  
    else  
        Brush(1, 0, 0, 0);  
    Ellipse(x-r, y-r, x+r, y+r);  
end;
```

радиус Земли

рисует: 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

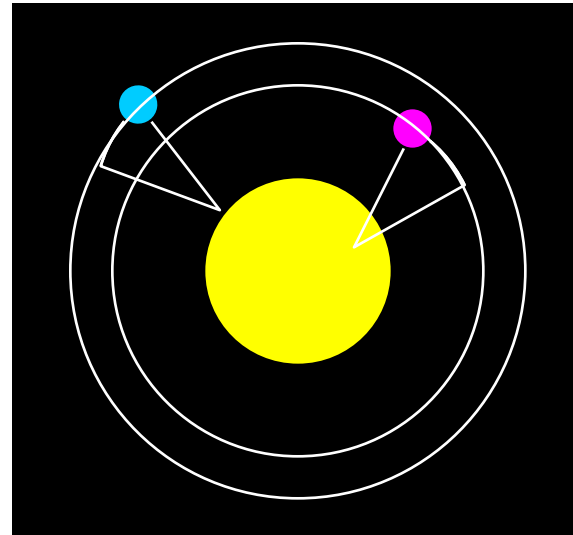
НОВЫЕ КООРДИНАТЫ

ждем 100 мс

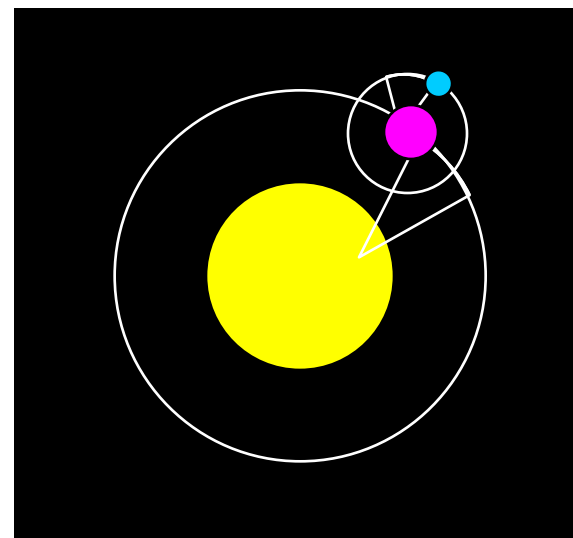
ПОВОРОТ на ha

# Задания

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



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



# Программирование на языке Паскаль

## Тема 12. Случайные числа

# Случайные числа

---

**Случайные явления:** везде...

- бросание монеты ("орел" или "решка")
- падение снега
- броуновское движение
- помехи при телефонной связи
- шум радиозэфира

**Случайные числа** – это такая последовательность чисел, для которой невозможно предсказать следующее даже зная все предыдущие.

**Проблема:** как получить на компьютере?

**Возможные решения:**

- использовать внешний источник шумовых помех
- с помощью математических преобразований

# Псевдослучайные числа

**Псевдослучайные числа** – это такая последовательность чисел, которая обладает свойствами случайных чисел, но каждое следующее число вычисляется по заданной формуле.

## Примеры:

1. Случайные целые числа  $[0, m)$  (линейный конгруэнтный метод)

$$x_n = (a \cdot x_{n-1} + c) \bmod m$$

а, с, m - целые числа

$$x_n = (16807 \cdot x_{n-1} + 12345) \bmod 1073741823$$

простое число

$$2^{30}-1$$

2. Случайные вещественные числа  $[0, 1]$

$$x_n = \left\{ (\pi + x_{n-1})^k \right\}$$

например,  $k = 5$

## Литература:

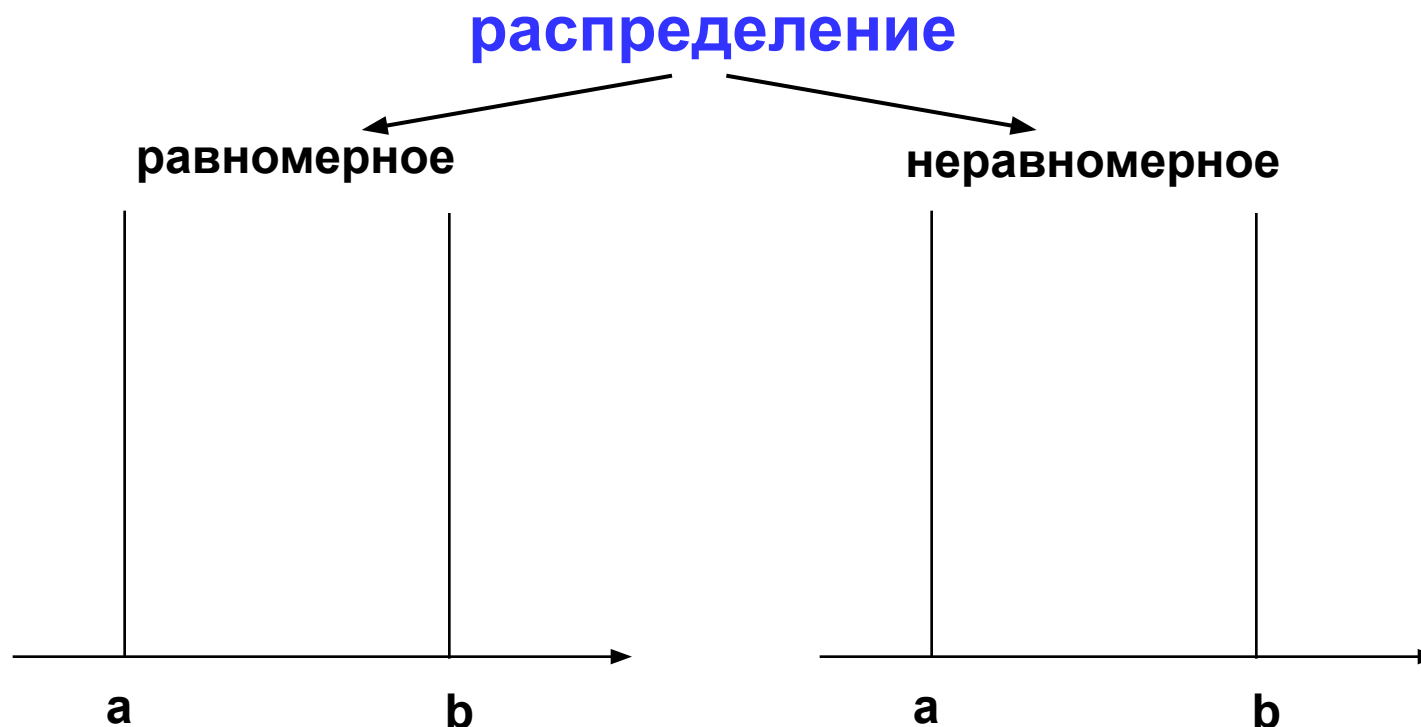
дробная часть числа

Д. Кнут, Искусство программирования для ЭВМ, т.2.



# Распределение случайных чисел

**Модель:** снежинки падают на отрезок  $[a, b]$

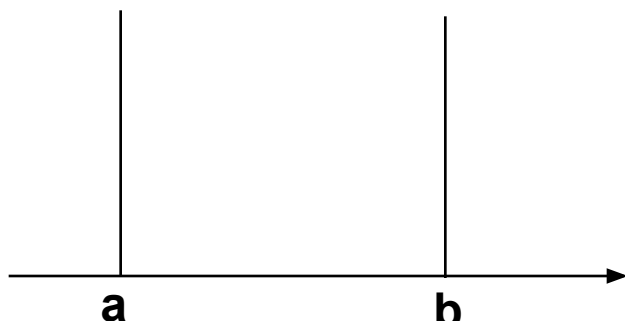


Сколько может быть разных распределений?

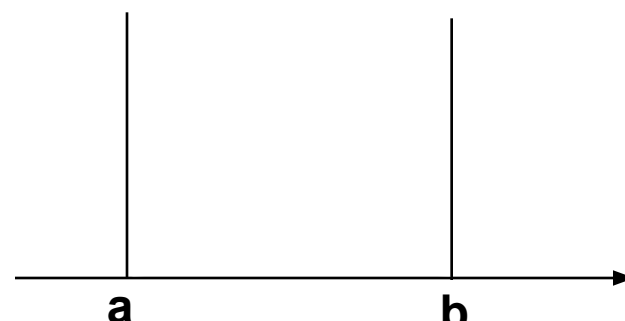
# Распределение случайных чисел

## Особенности:

- распределение – это характеристика **всей последовательности**, а не одного числа
- **равномерное** распределение одно, компьютерные датчики (псевдо)случайных чисел дают равномерное распределение
- неравномерных – много
- любое неравномерное можно получить с помощью равномерного


$$x = \frac{x_1 + x_2}{2}$$

равномерное распределение


$$x = \frac{x_1 + x_2 + \square + x_{12}}{12}$$

равномерное распределение

# Генератор случайных чисел в Паскале

---

## Целые числа в интервале $[0, N]$ :

```
var x: integer;  
...  
x := random ( 100 );    { интервал [0,99] }
```

## Вещественные числа в интервале $[0,1]$

```
var x: real;  
...  
x := random;             { интервал [0,1] }
```

# Случайные числа

---

**Задача:** заполнить прямоугольник 400 на 300 пикселей равномерно точками случайного цвета



**Как получить случайные координаты точки?**

```
x := random ( 400 ) ;  
y := random ( 300 ) ;
```

**Как добиться равномерности?**

обеспечивается автоматически при использовании функции `random`

**Как получить случайный цвет?**

```
Pen (1, random(256), random(256), random(256)) ;  
Point ( x, y ) ;
```

# Программа

```
program qq;  
var x, y, k, code, i: integer;  
    stop: boolean;  
begin  
    stop := False;  
    repeat  
        x := random(400);  
        y := random(300);  
        Pen(1, random(256), random(256), random(256));  
        Point(x, y);  
        if IsEvent then begin  
            Event(k, code, i);  
            if (k = 1) and (code = 27) then stop := True;  
        end;  
    until stop;  
end.
```

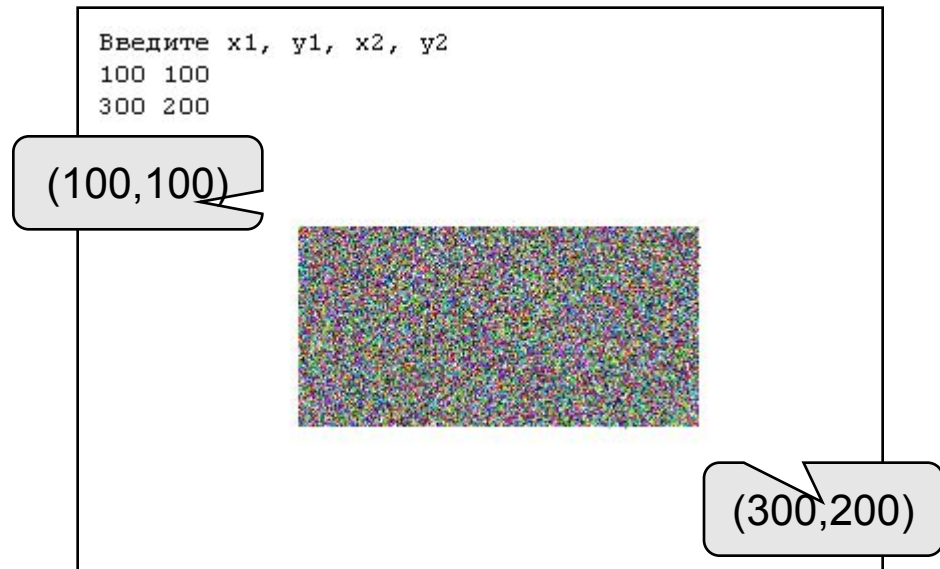
случайные координаты

случайный цвет

ВЫХОД по клавише *Esc*

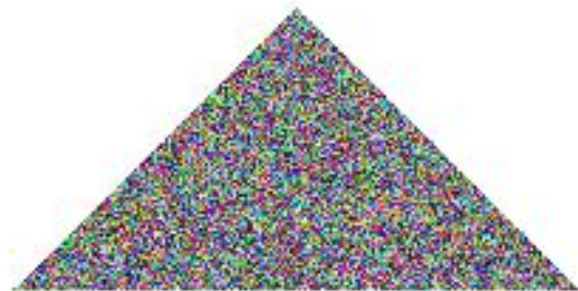
# Задания

**"4":** Ввести с клавиатуры координаты углов прямоугольника и заполнить его точками случайного цвета.



**"5":** Заполнить треугольник точками случайного цвета (равномерно или неравномерно).

**Подсказка:** возьмите равнобедренный треугольник с углом  $45^\circ$ .



# Программирование на языке Паскаль

## Тема 13. Функции

# Функции

---

**Функция** – это вспомогательный алгоритм (подпрограмма), результатом работы которого является некоторое значение.

## Примеры:

- вычисление  $\sin x$ ,  $\cos x$ ,  $\sqrt{x}$
- расчет значений по сложным формулам
- ответ на вопрос (простое число или нет?)

## Зачем?

- для выполнения одинаковых расчетов в различных местах программы
- для создания общедоступных библиотек функций



В чем отличие от процедур?



# Функции

**Задача:** составить функцию, которая вычисляет наибольшее из двух значений, и привести пример ее использования

**Функция:**

формальные параметры

```
function Max (a, b: integer): integer;  
begin  
  if a > b then Max := a  
  else       Max := b;  
end.
```

это результат  
функции

# Функции

## Особенности:

- заголовок начинается словом **function**

```
function Max (a, b: integer): integer;
```

- формальные параметры описываются так же, как и для процедур

```
function qq( a, b: integer; x: real): real;
```

- можно использовать параметры-переменные

```
function Max (var a, b: integer): integer;
```

- В конце заголовка через двоеточие указывается тип результата

- функция `function Max (a, b: integer): integer;` возвращает значение

# Функции

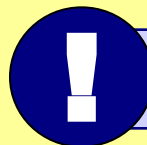
## Особенности:

- МОЖНО объявлять и использовать **локальные переменные**

```
function qq (a, b: integer): float;
  var x, y:
    float;
begin
  ...
end;
```

- знач... зается в  
переменную, имя которой совпадает с названием  
функции; объявлять ее **НЕ НАДО**:

```
function Max (a, b: integer): integer;
begin
  ...
  Max :=
  a;
end;
```



**В Delphi:**

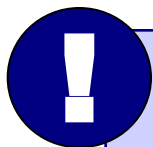
**Result := a;**

# Программа

```
program qq;  
var a, b, c: integer;  
  
function Max (a, b: integer): integer;  
begin  
    ...  
end;  
  
begin  
    writeln('Введите два числа');  
    read(a, b);  
    c := Max ( a, b );  
    writeln('Наибольшее число ', c );  
end.
```

фактические параметры

вызов функции



**Имена переменных, функций и процедур не должны совпадать!**

# Логические функции

**Задача:** составить функцию, которая определяет, верно ли, что заданное число – простое.

**Особенности:**

- ответ – логическое значение (True или False)
- результат функции можно использовать как логическую величину в условиях (if, while)

**Алгоритм:** считаем число делителей в интервале от 2 до N-1, если оно не равно нулю – число составное.

```
count := 0;  
  
for i := 2 to N-1 do  
    if N mod i = 0 then  
        count := count + 1;  
  
if count = 0 then  
    { число N простое }  
else { число N составное }
```



Как улучшить?

# Логические функции

```
program qq;  
var N: integer;
```

результат – логическое значение

```
function Prime (N: integer): boolean;
```

```
var count, i: integer;
```

```
begin
```

```
  i := 2; count := 0;
```

```
  while i*i <= N do
```

```
    if N mod i = 0 then count := count + 1;
```

```
    i := i + 1;
```

```
  end;
```

```
  Prime := (count = 0);
```

```
end;
```

перебор только до  $\sqrt{N}$

условие – это логическое значение

```
begin
```

```
  writeln('Введите целое число');
```

```
  read(N);
```

```
  if Prime(N) then
```

```
    writeln(N, ' - простое число');
```

```
  else writeln(N, ' - составное число');
```

```
end.
```

ВЫЗОВ функции

# Задания

---

**"4":** Составить функцию, которая определяет сумму всех чисел от 1 до N и привести пример ее использования.

**Пример:**

Введите число:

100

сумма = 5050

**"5":** Составить функцию, которая определяет, сколько зерен попросил положить на N-ую клетку изобретатель шахмат (на 1-ую – 1 зерно, на 2-ую – 2 зерна, на 3-ю – 4 зерна, ...)

**Пример:**

Введите номер клетки:

28

На 28-ой клетке 134217728 зерен.

## Задания (вариант 2)

---

**"4":** Составить функцию, которая определяет наибольший общий делитель двух натуральных и привести пример ее использования.

**Пример:**

Введите два числа:

14 21

НОД(14, 21) = 7

**"5":** Составить функцию, которая вычисляет функцию синус как сумму ряда (с точностью 0.001)

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \square$$

**x в радианах!**

**Пример:**

Введите угол в градусах:

45

$\sin(45) = 0.707$



# Конец фильма

---