

Программирование на языке Паскаль

(7 класс)

1. [Введение](#)
2. [Ветвления](#)
3. [Сложные условия](#)
4. [Циклы](#)
5. [Циклы с переменной](#)
6. [Графика](#)
7. [Процедуры](#)
8. [Анимация](#)
9. [Функции](#)
10. [Случайные числа](#)

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?

Оператор – это команда языка программирования высокого уровня.

1970 – язык Паскаль (Н. Вирт)

Простейшая программа

название программы

```
program qq;  
begin { начало программы }  
end.  { конец программы }
```

комментарии в фигурных скобках
не обрабатываются



Что делает эта программа?

Вывод текста на экран

```
program qq;  
begin  
▶ write('2+');  
▶ writeln('2=?'); { на новую строку}  
▶ writeln('Ответ: 4');  
end.
```

Протокол:

2+

Ответ: 4

Задания

«4»: Вывести на экран текст "лесенкой"

Вася

пошел

гулять

«5»: Вывести на экран рисунок из букв

Ж

ЖЖЖ

ЖЖЖЖЖ

ЖЖЖЖЖЖЖ

НН НН

ZZZZZ

Переменные

Задача. Ввести с клавиатуры два числа и найти их сумму.

Протокол:

Введите два целых числа

25 30

25+30=55

компьютер

пользователь

компьютер считает сам!

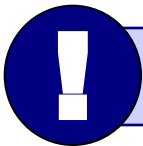


1. Как ввести числа в память?
2. Где хранить введенные числа?
3. Как вычислить?
4. Как вывести результат?

Программа

```
program qq;  
begin  
  { ввести два числа }  
  { вычислить их сумму }  
  { вывести сумму на экран }  
end.
```

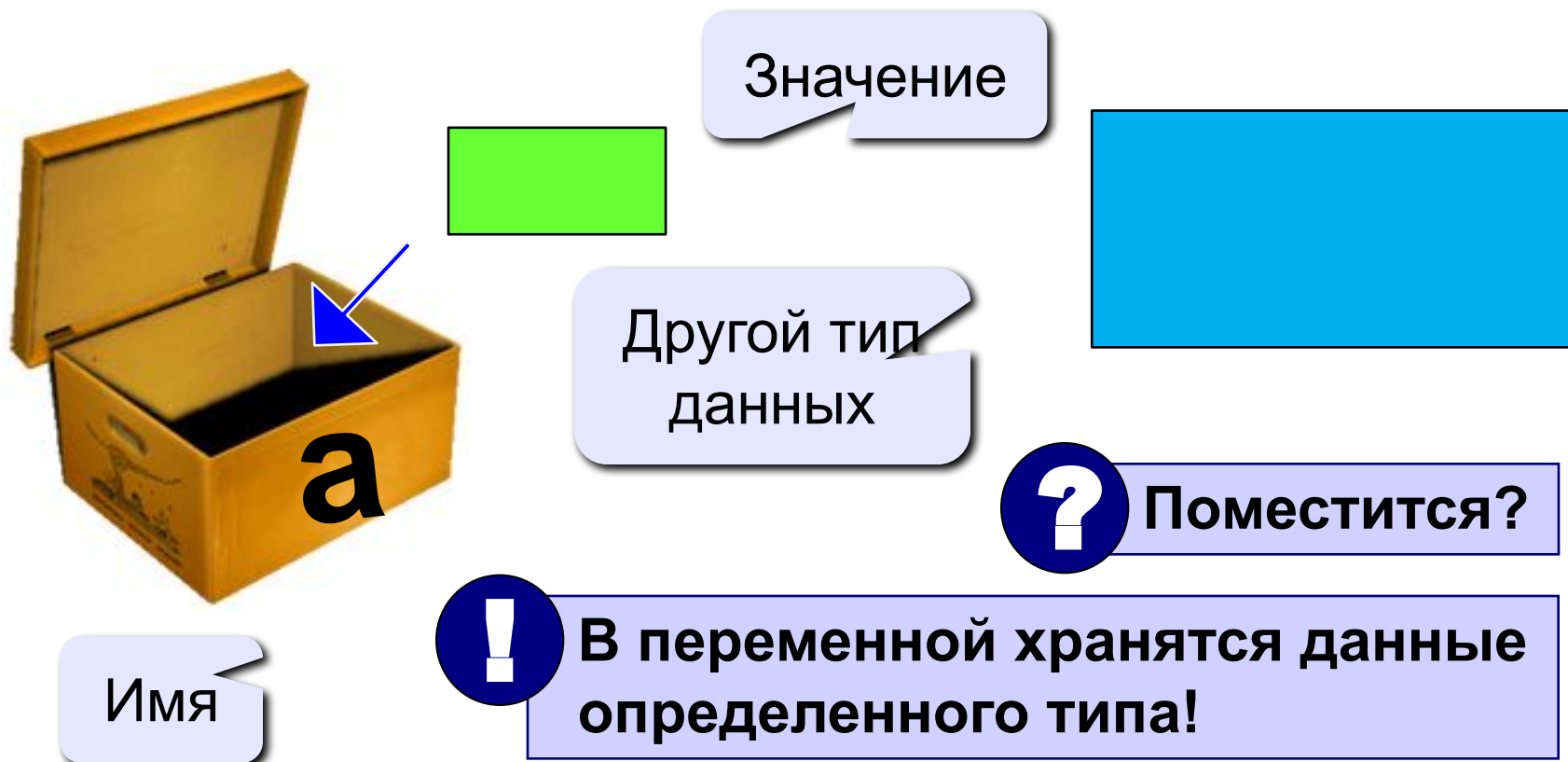
Псевдокод: алгоритм на русском языке с элементами Паскаля.



Компьютер не может исполнить псевдокод!

Переменные

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



Имена переменных

В именах **МОЖНО** использовать

- латинские буквы (A-Z)

заглавные и строчные буквы не различаются

- цифры

имя не может начинаться с цифры

- знак подчеркивания _

В именах **НЕЛЬЗЯ** использовать

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

Какие имена правильные??

AXby R&B 4Wheel Вася "PesBarbos"

TU154 [QuQu] _ABBA A+B

Переменные

Типы переменных:

- integer { целая }
- real { вещественная }
- и другие...

Выделение
места в памяти

Объявление переменных:

variable – переменная

тип – целые

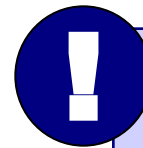
```
var a, b, c: integer;
```

СПИСОК ИМЕН
переменных

Как записать значение в переменную?

Оператор
присваивания

```
a := 5;
```

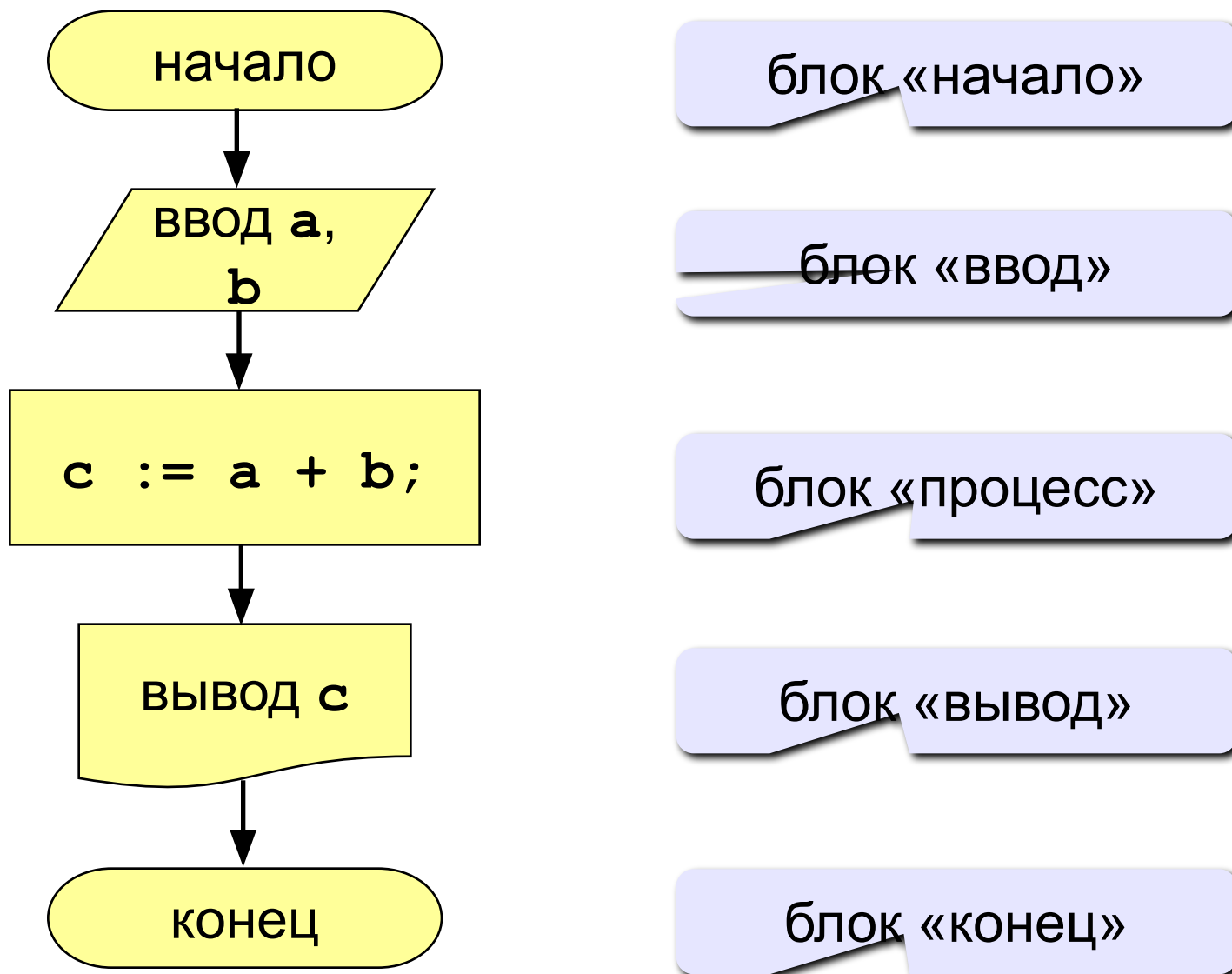


При записи нового
значения старое
стирается!

Оператор – это команда языка программирования (инструкция).

Оператор присваивания – это команда для записи нового значения в переменную.

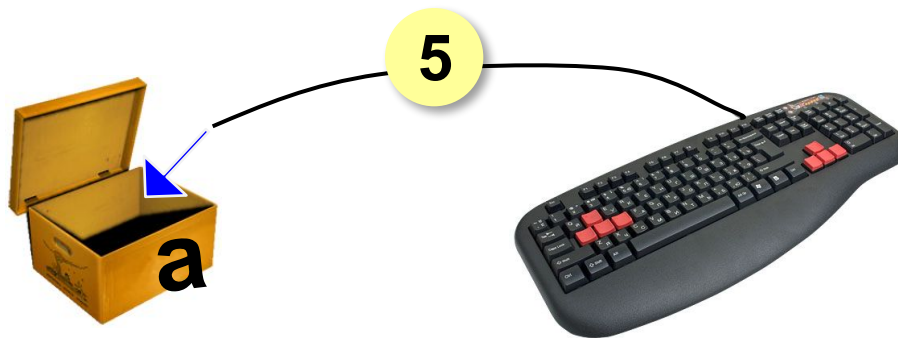
Блок-схема линейного алгоритма



Как ввести значение с клавиатуры

Оператор
ввода

```
read ( a );
```



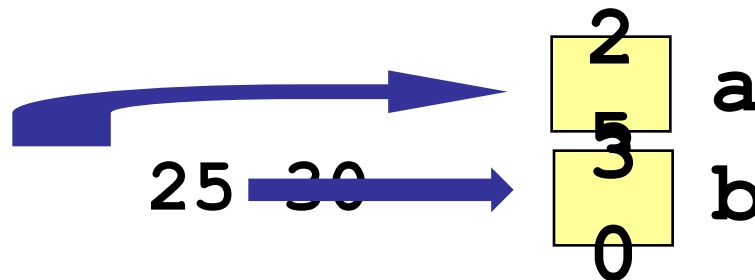
1. Программа ждет, пока пользователь введет значение и нажмет *Enter*.
2. Введенное значение записывается в переменную *a*.

Ввод значений двух переменных

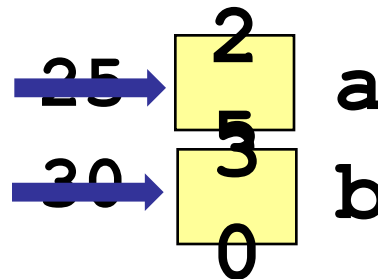
```
read ( a, b );
```

Ввод значений двух переменных (через пробел или *Enter*).

через пробел:



через *Enter*:



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);`

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.
```



Что плохо?

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:

КОМПЬЮТЕР

Введите два целых числа

25 30

ПОЛЬЗОВАТЕЛЬ

25+30=55

Задания

«3»: Ввести три числа, найти их сумму.

Пример:

Введите три числа:

4 5 7

$$4+5+7=16$$

«4»: Ввести три числа, найти их сумму и произведение.

Пример:

Введите три числа:

4 5 7

$$4+5+7=16$$

$$4*5*7=140$$

Задания

«5»: Ввести три числа, найти их сумму, произведение и среднее арифметическое.

Пример:

Введите три числа:

4 5 7

$$4+5+7=16$$

$$4*5*7=140$$

$$(4+5+7) / 3 = 5.333333$$

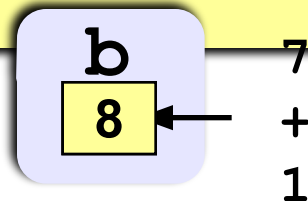
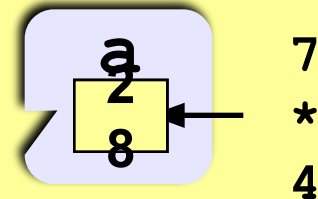
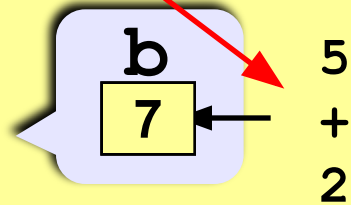
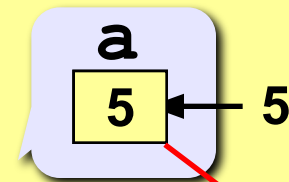
Как изменить значение переменной?

Пример:

```

program qq;
var a, b: integer;
begin
  a := 5;
  b := a + 2;
  a := (a + 2) * (b - 3);
  b := b + 1;
end.

```



Арифметические операции

+ сложение **-** вычитание

***** умножение **/** деление

div деление нацело (остаток отбрасывается)

mod остаток от деления

```
var a, b: integer;  
begin  
    a := 7*3 - 4;  
    a := a * 5;  
    b := a div 10;  
    a := a mod 10;  
end.
```

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

Порядок выполнения операций

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

1 2 4 5 3 6

```
z := (5*a+c) / a * (b-c) / b;
```

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

$$z = \frac{5a+c}{ab} (b-c)$$

2 3 5 4 1 10 6 9 8 7

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


Вывод целых чисел

```
program qq;  
var a, b: integer;  
begin  
  a := 15;  
  b := 45;  
  writeln ( a, b );  
  writeln ( a:4, b:4 );  
end.
```

1545

15 45

СИМВОЛОВ
на ЧИСЛО

Вывод вещественных чисел

```

program qq;
var x: real;
begin
  x := 12.345678;
  writeln ( x );
  writeln ( x:10 );
  writeln ( x:7:2 );
end.

```

ВСЕГО
СИМВОЛОВ

1,234568·10¹

1.234568E+001

1.23E+001

12.35

ВСЕГО
СИМВОЛОВ

В дробной
части

Программирование на языке Паскаль

Тема 2. Ветвления

Разветвляющиеся алгоритмы

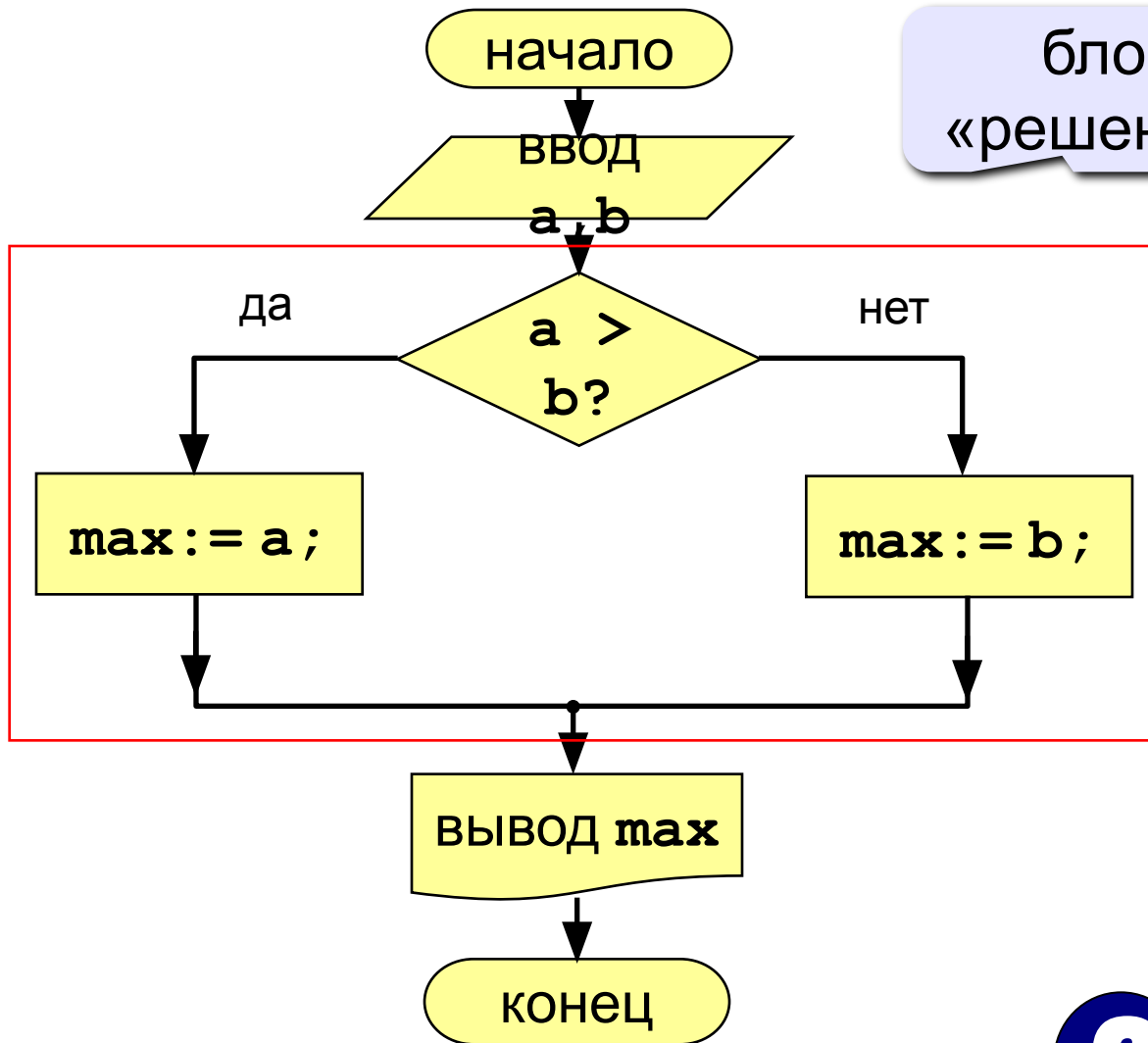
Задача. Ввести два целых числа и вывести на экран наибольшее из них.

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?

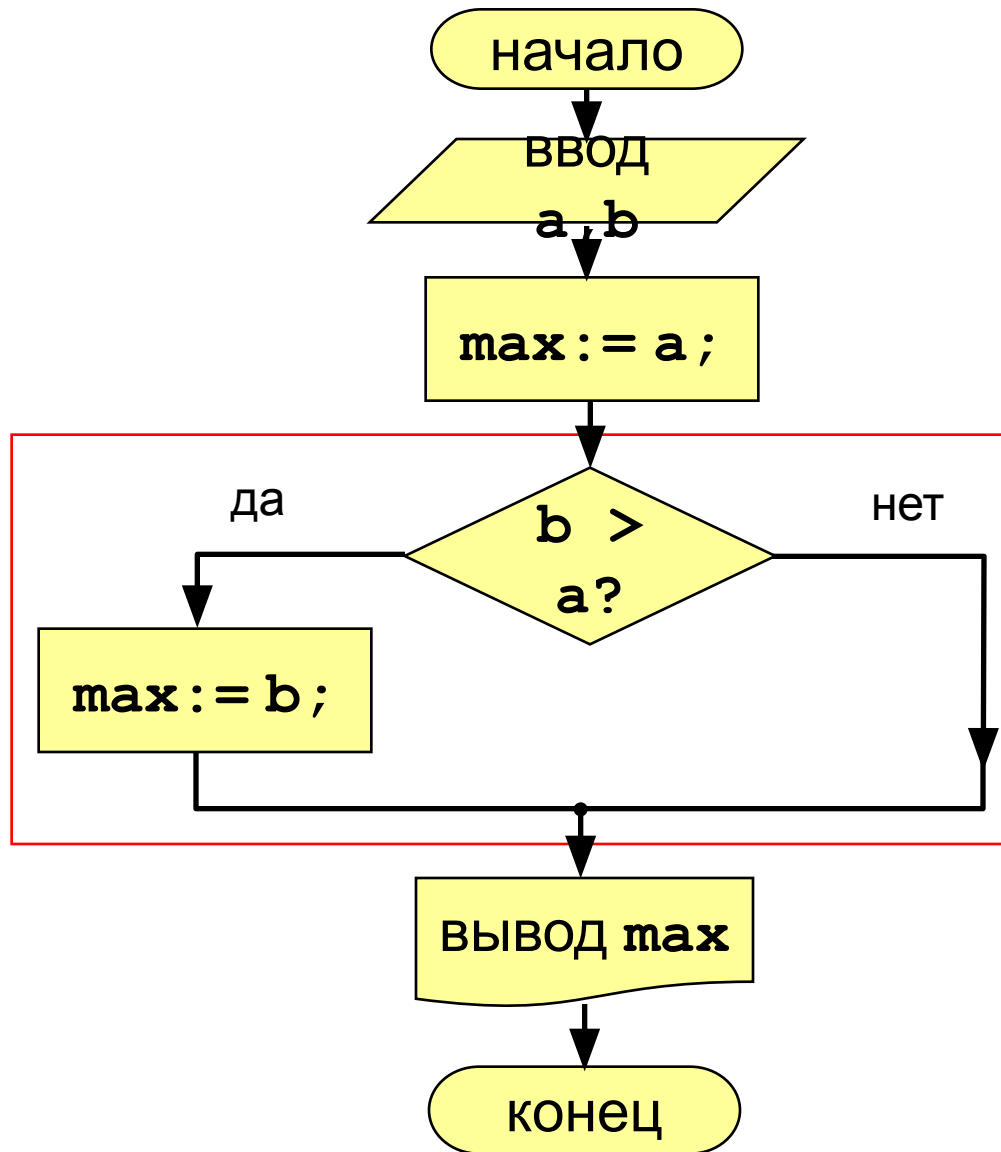
```
if a > b then begin
  a := b;
end
else begin
  b := a;
end;
```

```
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 then  
    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

«3»: Ввести два числа и вывести их в порядке возрастания.

Пример:

Введите два числа:

15 9

Ответ: 9 15

«4»: Ввести три числа и найти наибольшее из них.

Пример:

Введите три числа:

4 15 9

Наибольшее число 15

Задания

«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

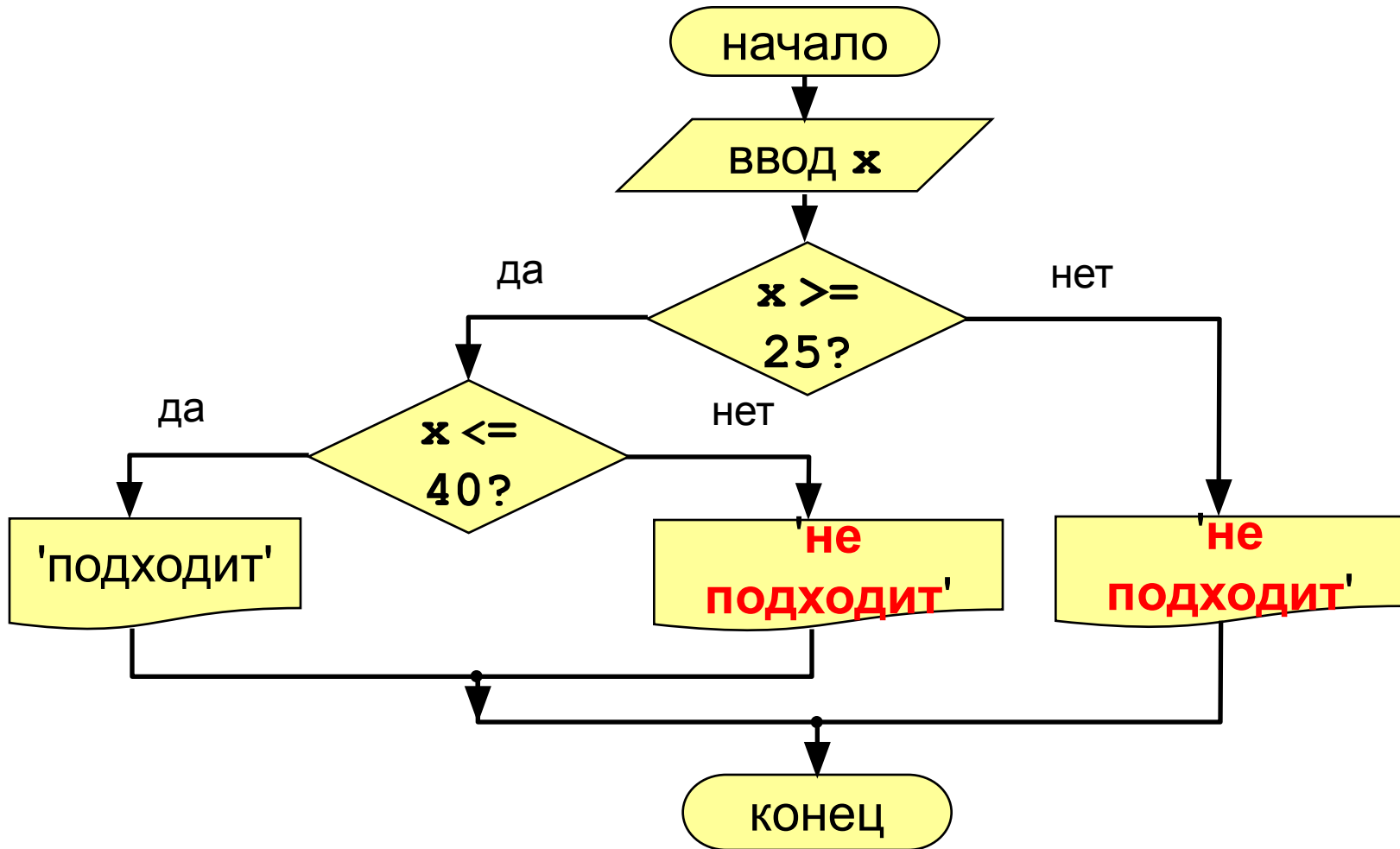
Задача. Фирма набирает сотрудников от 25 до 40 лет включительно. Ввести возраст человека и определить, подходит ли он фирме (вывести ответ «подходит» или «не подходит»).

Особенность: надо проверить, выполняются ли два условия одновременно.



Можно ли решить известными методами?

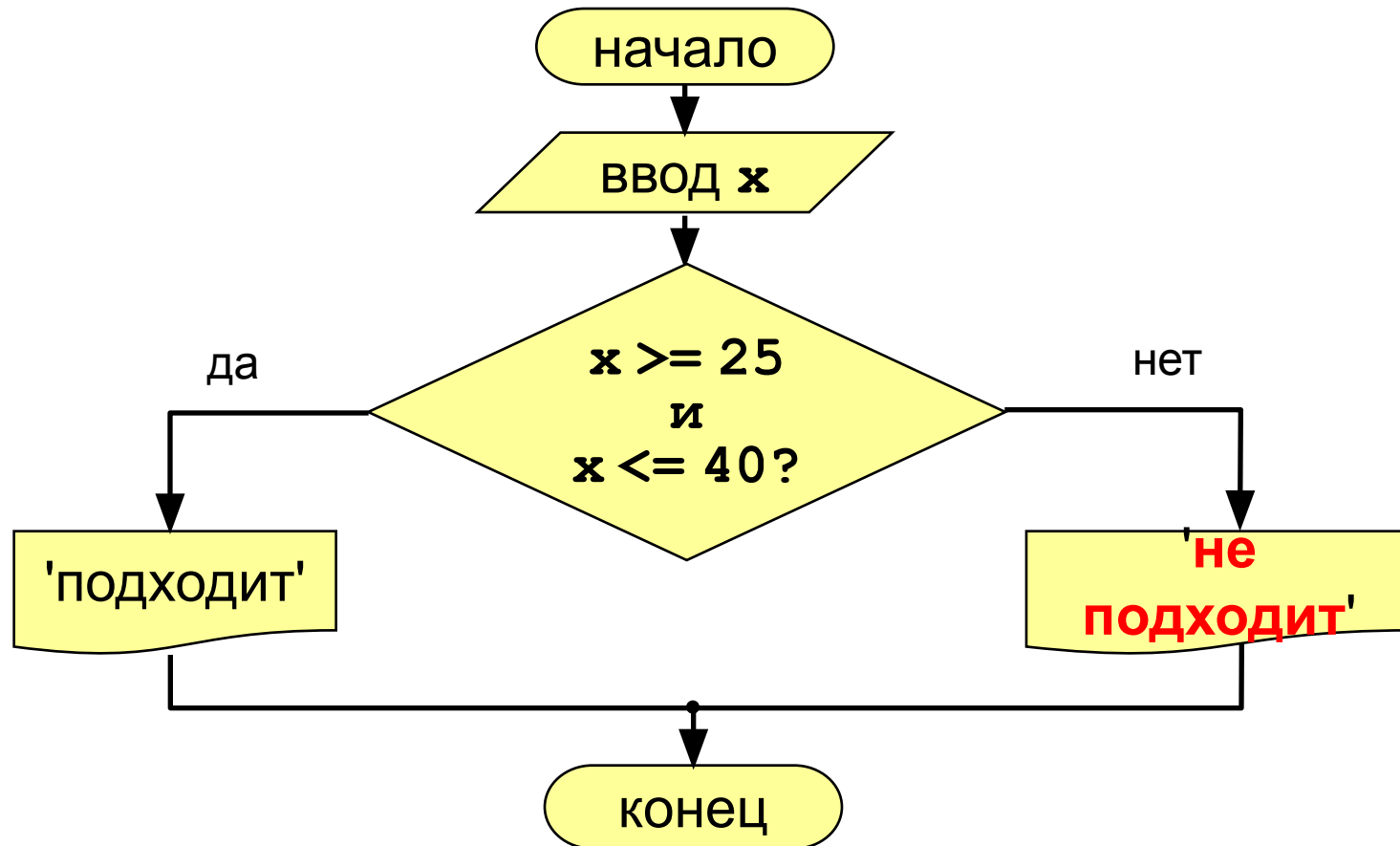
Вариант 1. Алгоритм



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

Простые условия (отношения)

равно

< <= > >= = <>

не равно

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

- **not** – НЕ (отрицание, инверсия)
- **and** – И (одновременное выполнение условий)
- **or** – ИЛИ (выполнение хотя бы одного из условий)

Сложные условия

Порядок выполнения (приоритет = старшинство)

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

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) and (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

«3»: Ввести три числа и определить, верно ли, что они вводились в порядке возрастания.

Пример:

Введите три числа:

4 5 17

да

«4»: Ввести номер месяца и вывести название времени года.

Пример:

Введите номер месяца:

4

весна

Задания

«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

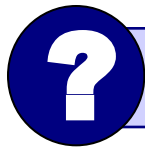
Циклы

Цикл – это многократное выполнение одинаковой последовательности действий.

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

Задача. Вывести на экран 5 раз слово «Привет».

Особенность: одинаковые действия выполняются 5 раз.



Можно ли решить известными методами?

Циклы

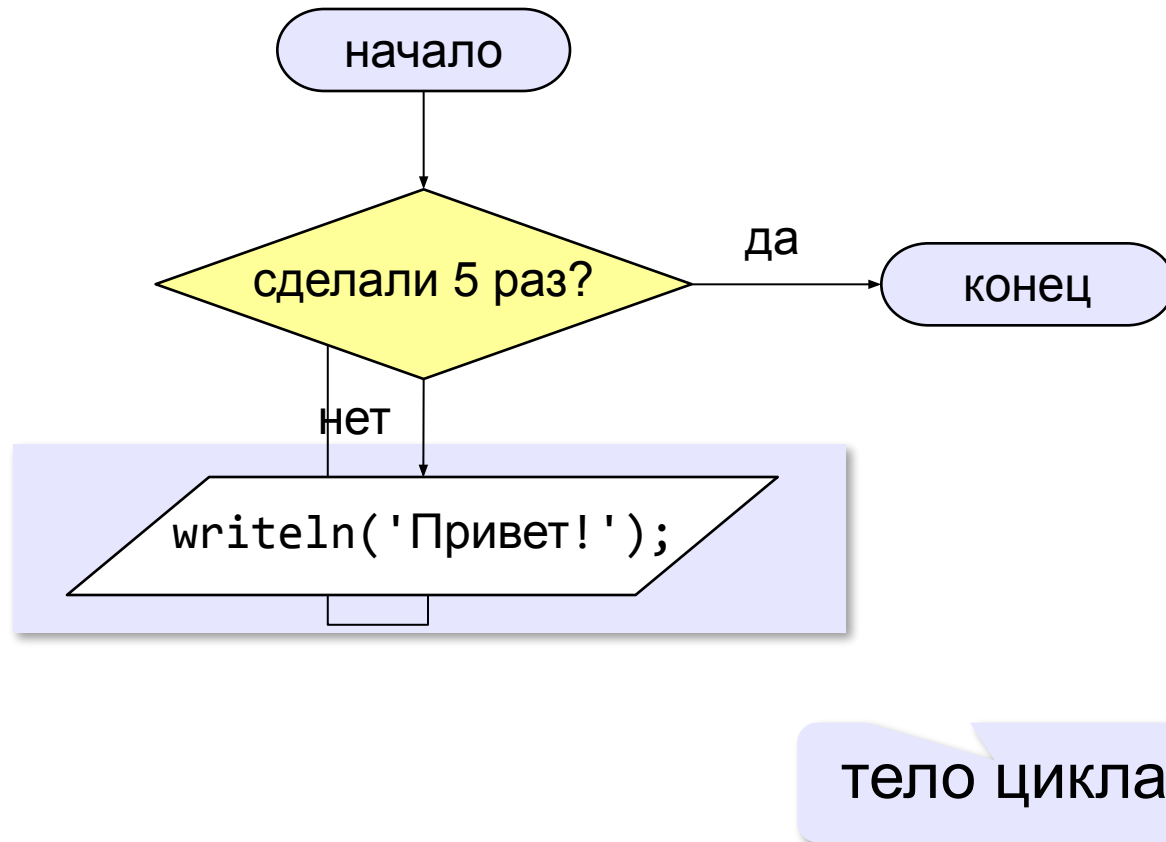
```
program qq;  
begin  
  writeln ( ' Привет ' ) ;  
  writeln ( ' Привет ' ) ;  
  writeln ( ' Привет ' ) ;  
  writeln ( ' Привет ' ) ;  
  writeln ( ' Привет ' ) ;  
end.
```



Что плохо?

Циклы

Блок-схема:



Циклы

```
program Privet;
```



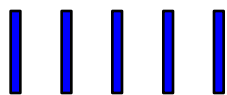
Как отсчитать ровно 5 раз?

```
begin
```

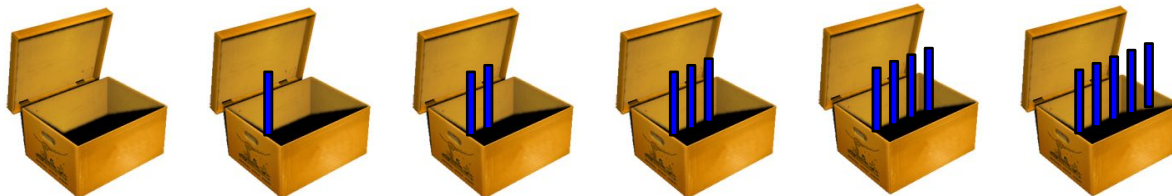
```
{ сделай 5 раз }
```

```
  writeln( 'Привет' );
```

```
end.
```

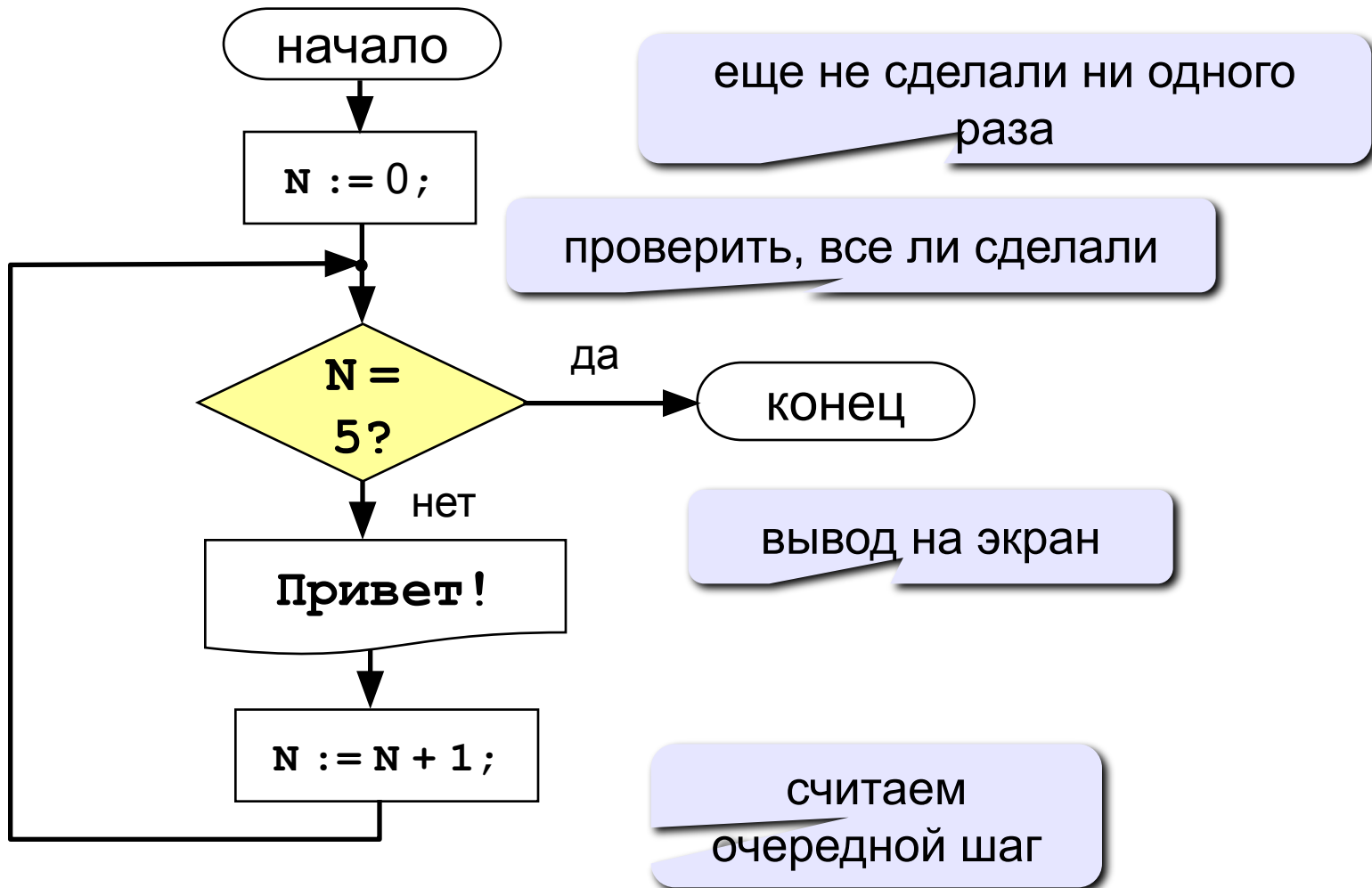


Как запоминать, сколько раз уже сделали?



$N := N + 1;$

Алгоритм

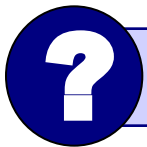


Цикл с условием

```
program Privet2;  
var N: integer;  
begin  
  N := 0;  
  while ( N <> 5 ) do begin  
    writeln( 'Привет!' );  
    N := N + 1;  
  end  
end.
```



Как изменить число шагов?



Как ввести число шагов с клавиатуры?

Ввод количества

```
program Privet2;  
var N , K : integer;  
begin  
    N := 0;  
    writeln( 'Введите число  
шагов: ' );  
    read( K );  
    while ( N <> K ) do begin  
        writeln( 'Привет! ' );  
        N := N + 1;  
    end  
end.
```

Цикл с условием

Вместо знаков вопроса добавьте числа и операторы так, чтобы цикл выполнялся ровно 5 раз:

```
program Privet3;  
var N: integer;  
begin  
    N := 5;  
    while ( N <> 0 ) do begin  
        writeln( 'Привет!' );  
        N := N - 1;  
    end  
end.
```

Что получим?

```
program Primer1;  
var N: integer;  
begin  
  N := 1;  
  while ( N <= 5 ) do begin  
    writeln(N);  
    N := N + 1;  
  end  
end.
```



1
2
3
4
5

Что получим?

```
program Primer2;  
var N: integer;  
begin  
  N := 1;  
  while ( N <= 5 ) do begin  
    writeln(N);  
    N := N + 2;  
  end  
end.
```



1
3
5

Что получим?

```
program Primer3;  
var N: integer;  
begin  
  N := 2;  
  while ( N <> 5 ) do begin  
    writeln(N);  
    N := N + 2;  
  end  
end.
```

2
4
6
8
10
12
14
16
...



Условие цикла никогда не станет ложным – это закливание!

Что получим?

```
program Primer4;  
var N: integer;  
begin  
  N := 1;  
  while ( N <= 5 ) do begin  
    writeln(N*N*N) ;  
    N := N + 1;  
  end  
end.
```



1
8
27
64
125

Что получим?

```
program Primer5;  
var N: integer;  
begin  
  N := 5;  
  while ( N >= 1 ) do begin  
    writeln(N*N*N) ;  
    N := N - 1;  
  end  
end.
```

125
64
27
8
1



Задания

«3»: Ввести натуральное число вывести квадраты и кубы всех чисел от 1 до этого числа.

Пример:

Введите натуральное число:

3

1: 1 1

2: 4 8

3: 9 27

«4»: Ввести два целых числа a и b ($a \leq b$) и вывести квадраты все чисел от a до b .

Пример:

Введите два числа:

4 5

4*4=16

5*5=25

Задания

«5»: Ввести два целых числа a и b ($a \leq b$) и вывести сумму квадратов всех чисел от a до b .

Пример:

Введите два числа:

4 10

Сумма квадратов 371

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.

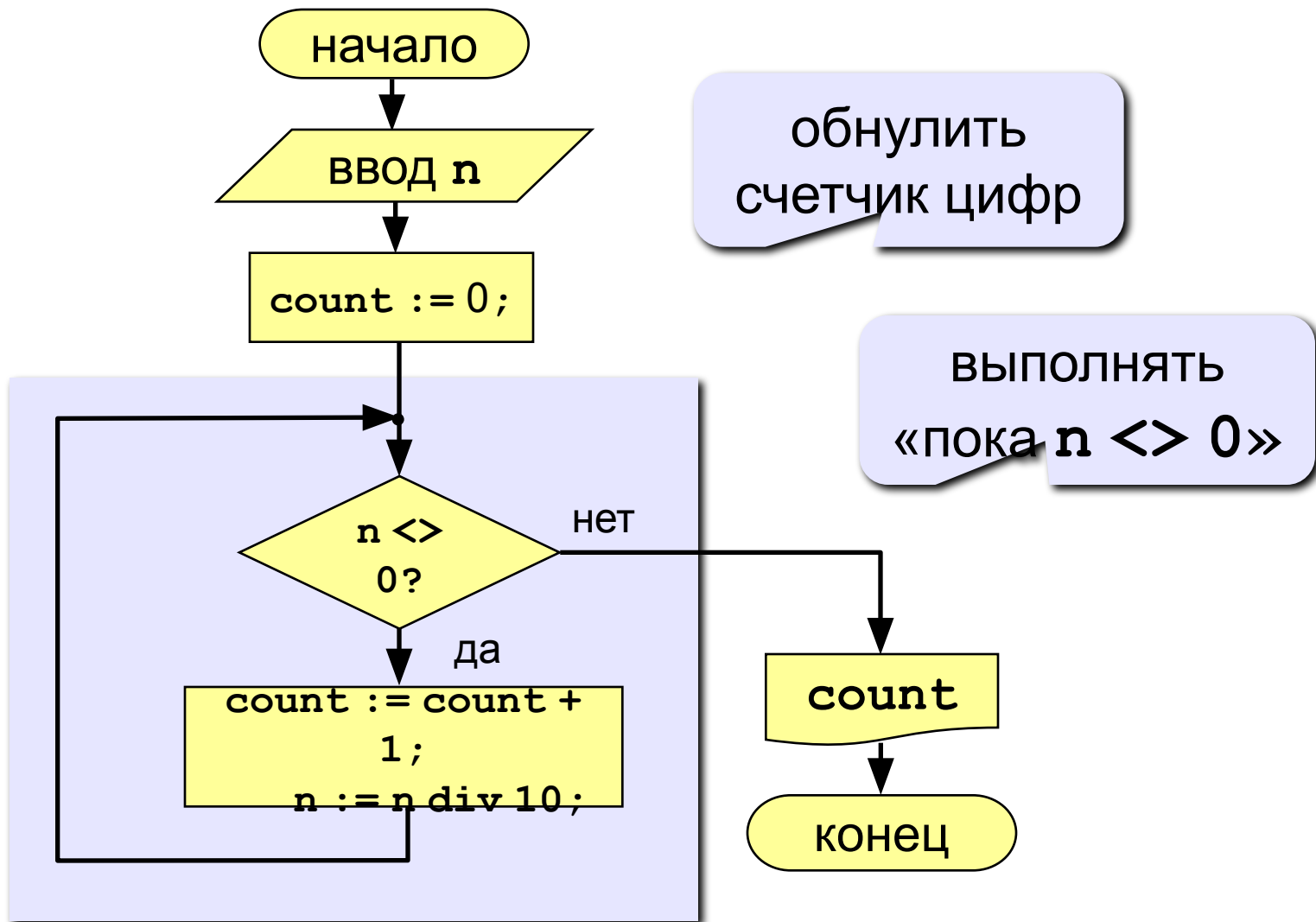
Идея решения: Отсекаем последовательно последнюю цифру, увеличиваем счетчик.

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 <> 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

Задания

«3»: Ввести целое число и определить, верно ли, что в нём ровно 3 цифры.

Пример:

Введите число: Введите число:

123

1234

Да.

Нет.

«4»: Ввести целое число и найти сумму его цифр.

Пример:

Введите целое число:

1234

Сумма цифр числа 1234 равна 10.

Задания

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

Пример:

Введите целое число:

1232

Нет.

Введите целое число:

1224

Да.

«6»: Ввести целое число и определить, верно ли, что в его записи есть две одинаковые цифры, **НЕ** обязательно стоящие рядом.

Пример:

Введите целое число:

1234

Нет.

Введите целое число:

1242

Да.

Задания-2

«3»: Ввести целое число и определить, верно ли, что в нём ровно 1 цифра «9».

Пример:

Введите число:

193

Да.

Введите число:

1994

Нет.

«4»: Ввести целое число и определить, верно ли, что все его цифры четные.

Пример:

Введите число:

2684

Да.

Введите число:

2994

Нет.

Задания-2

«5»: Ввести целое число и определить, верно ли, что все его цифры расположены в порядке возрастания.

Пример:

Введите целое число:

1238

Да.

Введите целое число:

1274

Нет.

«6»: Ввести целое число и «перевернуть» его, так чтобы первая цифра стала последней и т.д.

Пример:

Введите целое число:

1234

4321

Введите целое число:

782

287

Цикл с постусловием

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

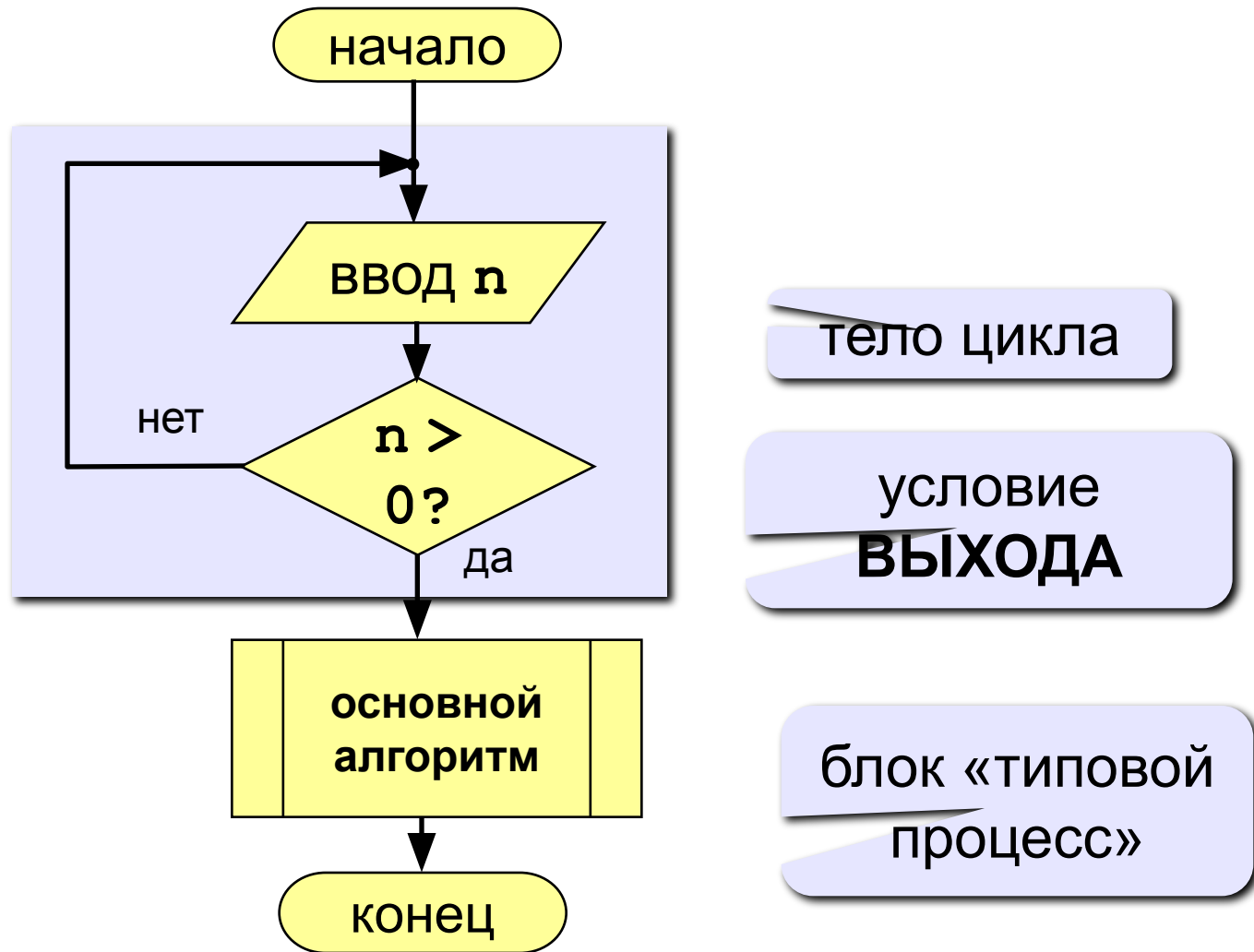
Проблема: Как не дать ввести отрицательное число или ноль?

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

until n > 0;

... { основной алгоритм }

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 ≥ 0 :

-234

Нужно положительное число.

Введите число ≥ 0 :

1234

Да

Введите число ≥ 0 :

1233

Нет

«5»: Ввести натуральное число и определить, какие цифры встречаются несколько раз.

Пример:

Введите число ≥ 0 :

2323

Повторяются: 2, 3

Введите число ≥ 0 :

1234

Нет повторов.

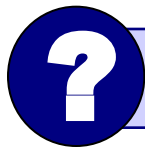
Программирование на языке Паскаль

Тема 5. Циклы с переменной

Цикл с переменной

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

Особенность: одинаковые действия выполняются 8 раз.



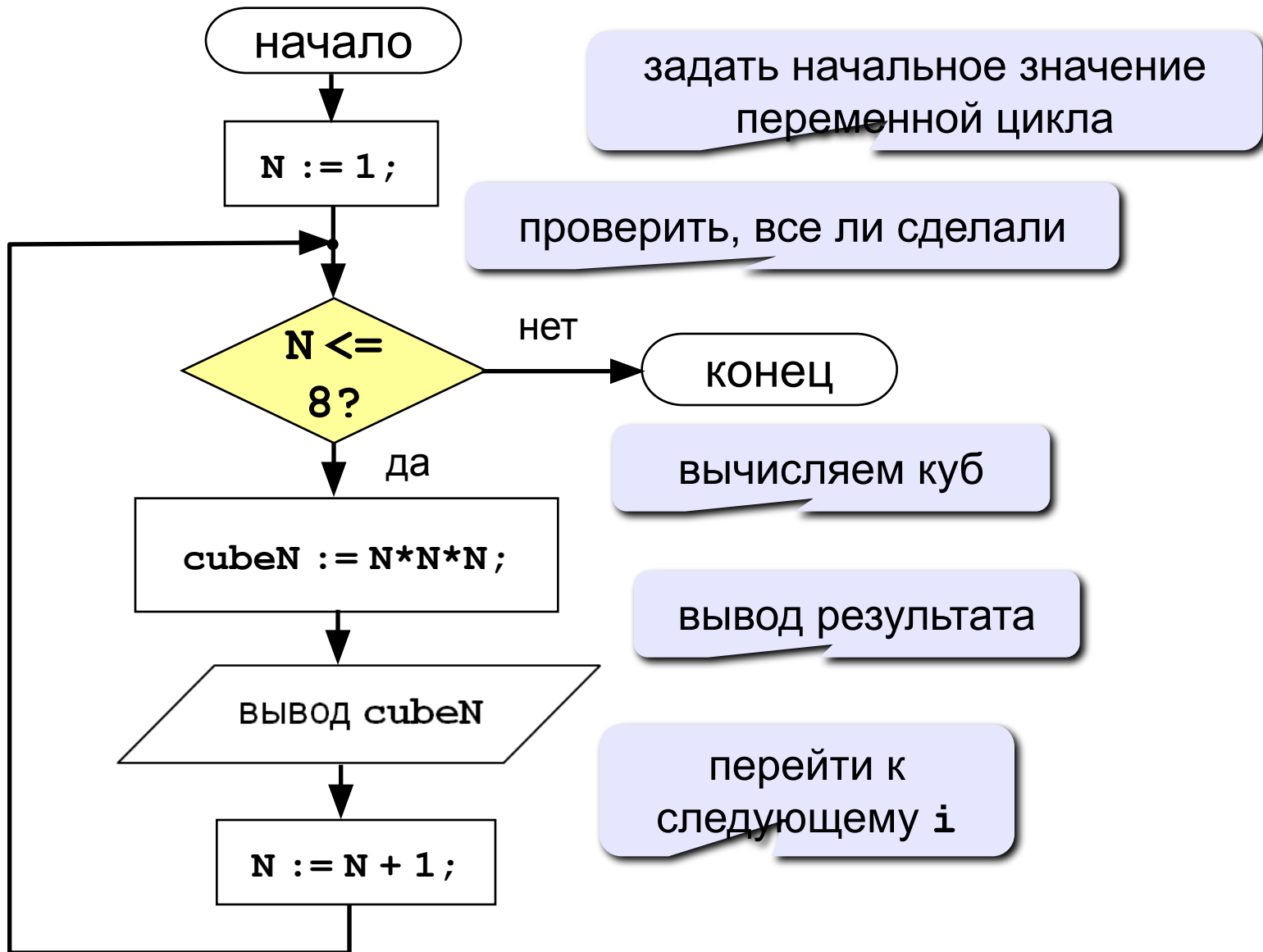
Можно ли решить известными методами?

```
N := 1;           { очередное число }  
cubeN := N*N*N;  { куб этого числа }  
writeln(N:4, cubeN:4);  
N := 2;  
...
```



А если начальное и конечное значения вводятся с клавиатуры?

Алгоритм



Цикл с переменной

Задача: вывести кубы натуральных чисел от 1 до 8.

```
program Cubes;  
var N, cubeN: integer;  
begin  
  N :=  
  1;  
  while ( N <= 8 ) do begin  
    cubeN := N*N*N;  
    writeln (cubeN) ;  
    N := N + 1;  
  end  
end.
```

3 действия с N

Цикл с переменной

Задача: вывести кубы натуральных чисел от 1 до 8.

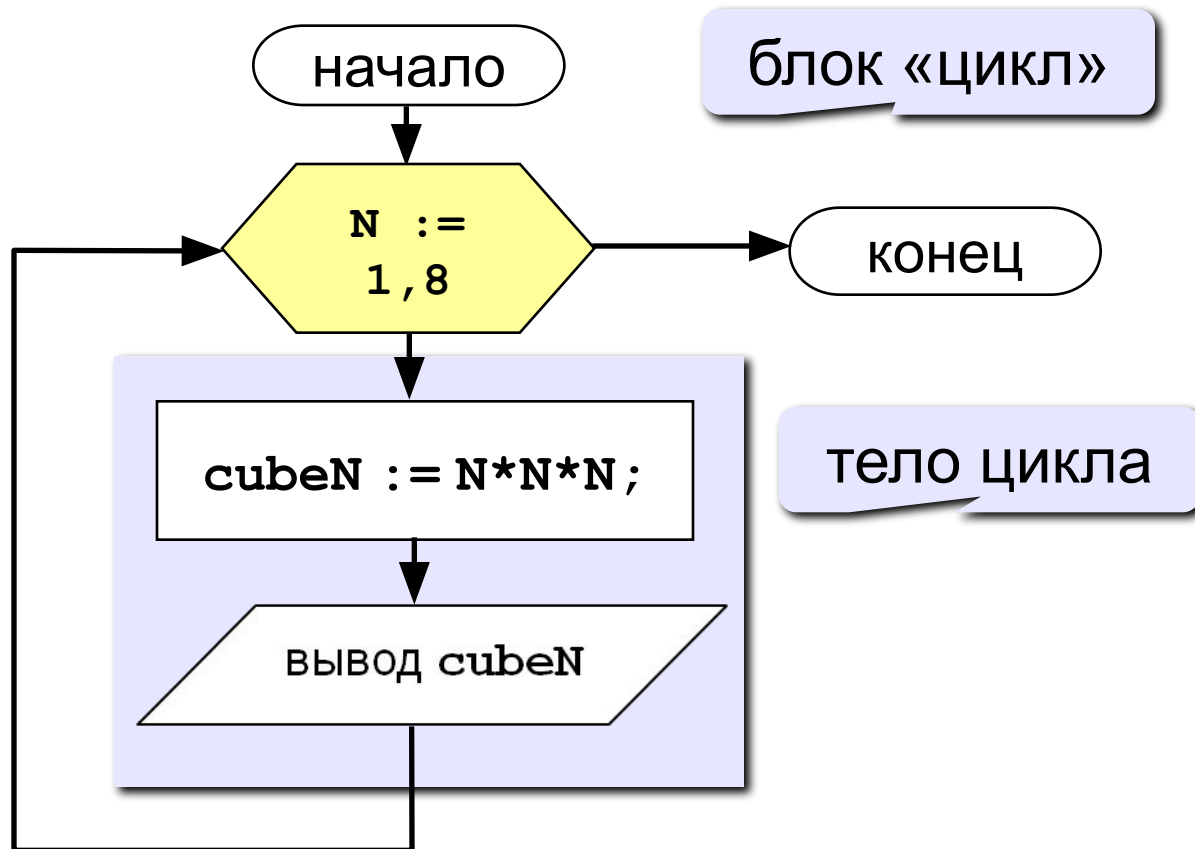
```
program Cubes2;  
var N, cubeN: integer;  
begin  
  for N:=1 to 8 do begin  
    cubeN:=N*N*N;  
    writeln(cubeN)  
  end  
end.
```

для 1, 2, 3, ..., 8



Как обойтись без переменной `cubeN`?

Алгоритм (с блоком «цикл»)



Программа

```
program qq;
```

```
var N, cubeN: integer;
```

```
begin
```

начальное значение

переменная
цикла

конечное значение

```
  for N:=1 to 8 do begin
```

```
    cubeN := N*N*N;
```

```
    writeln(N:4, cubeN:4)
```

```
  end
```

```
end.
```

Цикл с уменьшением переменной

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

Особенность: переменная цикла должна уменьшаться.

Решение:

```
for N:=8 downto 1 do begin
  cubeN := N*N*N;
  writeln(N:4, cubeN: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
защипливания

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 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 N:=1 to 9 do begin
  if N mod 2 = 1 then begin
    cubeN := N*N*N;
    writeln(N:4, cubeN: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 **N** равно 1, с каждым шагом цикла **N** увеличивается на 2.

Решение:

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

Как изменить шаг? – III

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

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

$$N = 2k - 1$$

Решение:

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

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

«3»: Ввести два целых числа a и b ($a \leq b$) и вывести кубы всех чисел от a до b .

Пример:

Введите два числа:

4 6

$$4 * 4 * 4 = 64$$

$$5 * 5 * 5 = 125$$

$$6 * 6 * 6 = 216$$

«4»: Ввести натуральное число A и вывести числа от A до 1 (через одно) в порядке убывания.

Пример:

Введите натуральное число:

8

Ответ: 8 6 4 2

Задания

«5»: Ввести целое число a и вывести сумму квадратов всех чисел от 1 до a с шагом 0.1.

Пример:

Введите последнее число:

3

Сумма 91.7

$$1^2 + 1.1^2 + 1.2^2 + \dots + a^2$$

Задания-2

«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

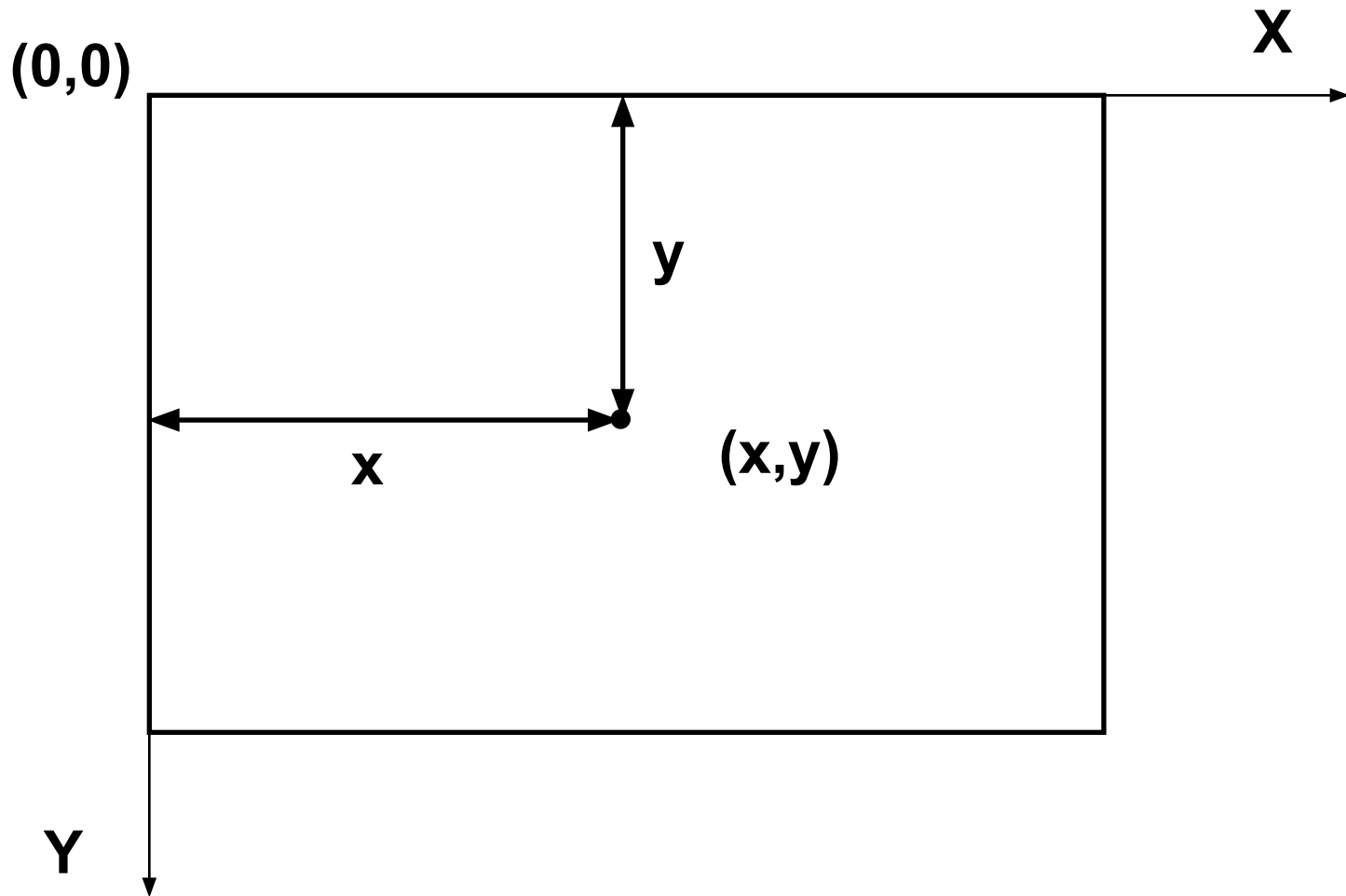
...

46: 2116 97336

Программирование на языке Паскаль

Тема 6. 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

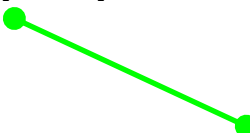
Точки, отрезки и ломаные

(x, y)



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

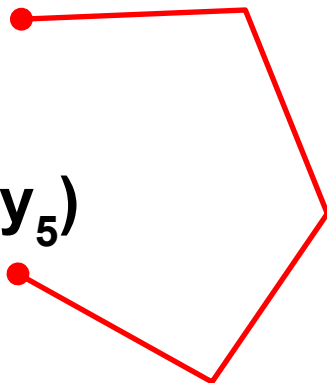
(x_1, y_1)



(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)

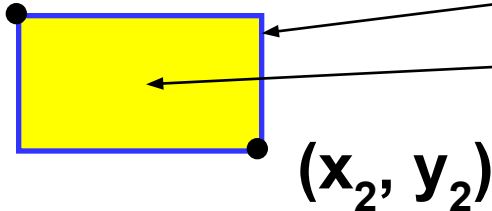
(x_4, y_4)

(x_5, y_5)

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

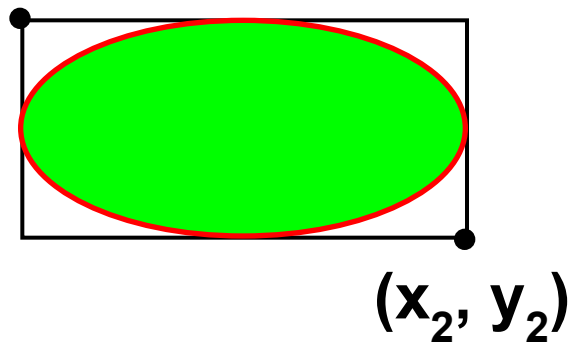
Фигуры с заливкой

(x_1, y_1)

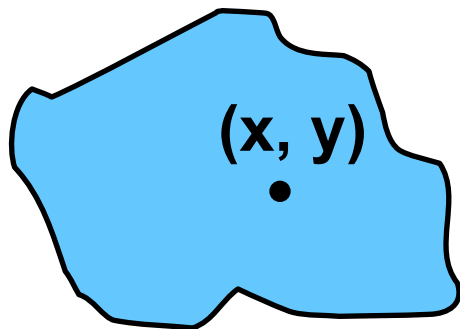


```
Pen (1, 0, 0, 255);  
Brush (1, 255, 255, 0);  
Rectangle (x1, y1, x2, y2);
```

(x_1, y_1)



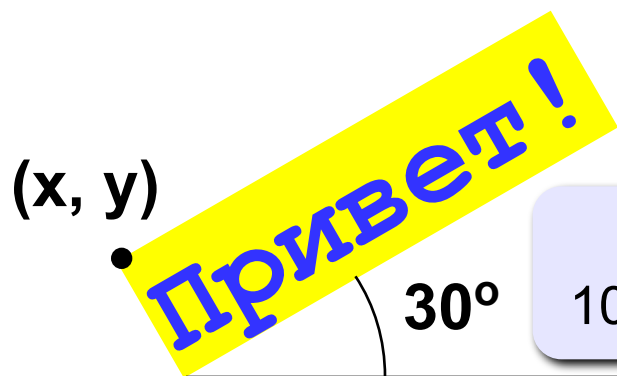
```
Pen (1, 255, 0, 0);  
Brush (1, 0, 255, 0);  
Ellipse (x1, y1, x2, y2);
```



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);
```

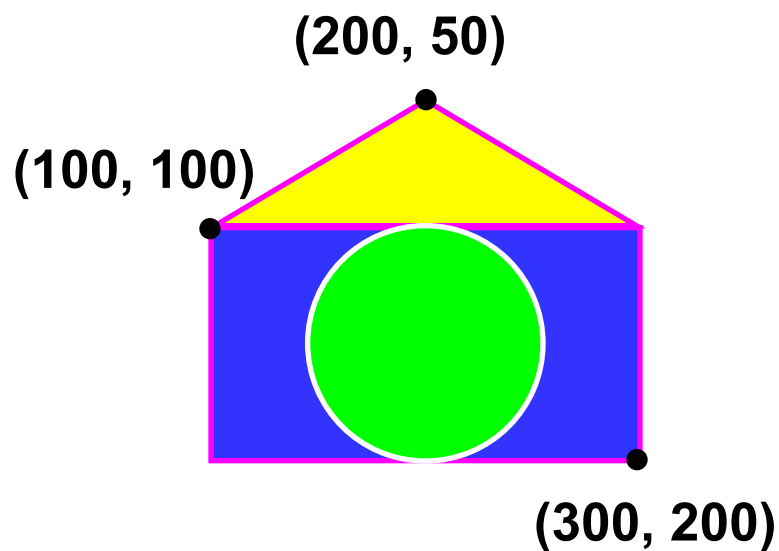
размер
10 пикселей

угол
поворота

насыщенность:
400 – нормальный
600 – жирный

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

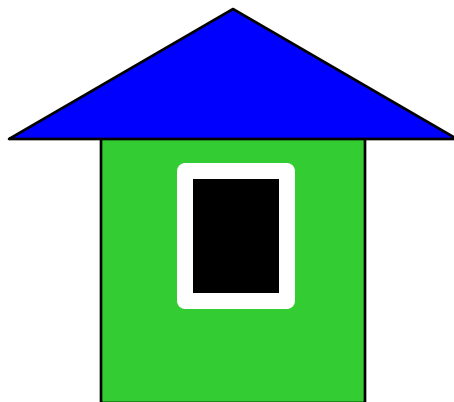
Пример



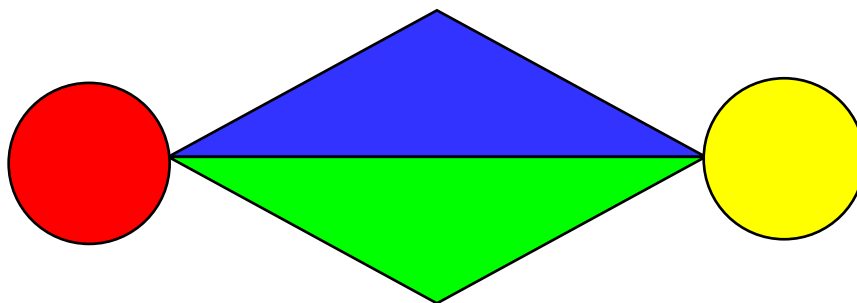
```
program qq;  
begin  
    Pen(2, 255, 0, 255);  
    Brush(1, 0, 0, 255);  
    Rectangle(100, 100, 300, 200);  
    MoveTo(100, 100);  
    LineTo(200, 50);  
    LineTo(300, 100);  
    Brush(1, 255, 255, 0);  
    Fill(200, 75);  
    Pen(2, 255, 255, 255);  
    Brush(1, 0, 255, 0);  
    Ellipse(150,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

Задания

«3»: «Домик»

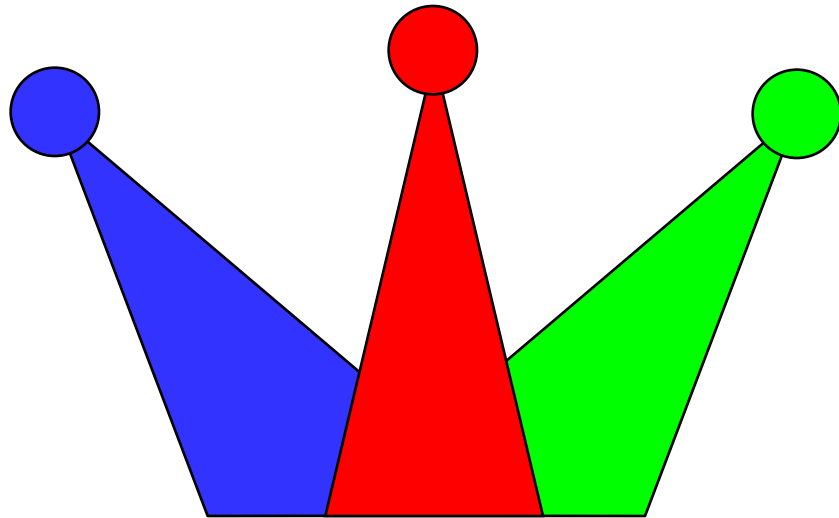


«4»: «Лягушка»



Задания

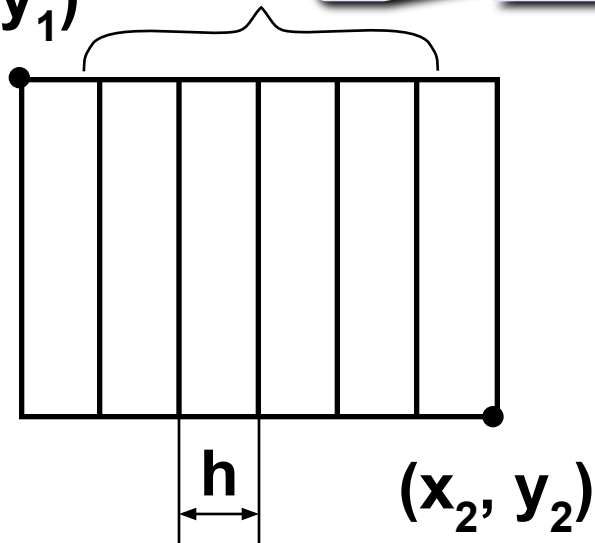
«5»: «Корона»



Штриховка

(x_1, y_1)

N линий (N=5)



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

```
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

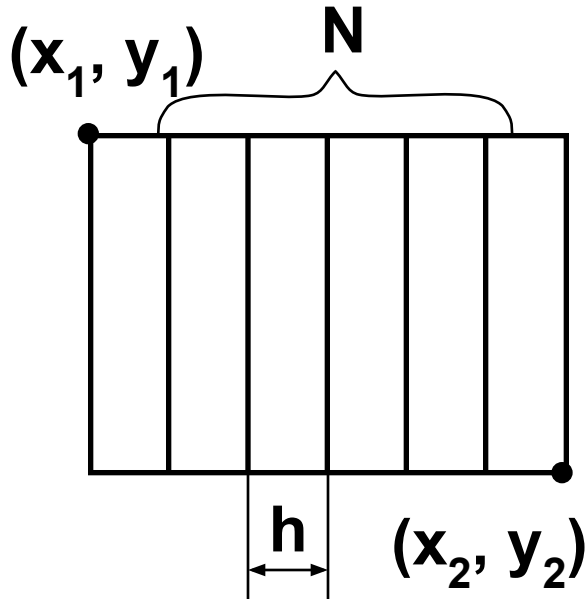
x

```
Rectangle (x1, y1, x2, y2);
h := (x2 - x1) / (N + 1);
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

Штриховка (программа)

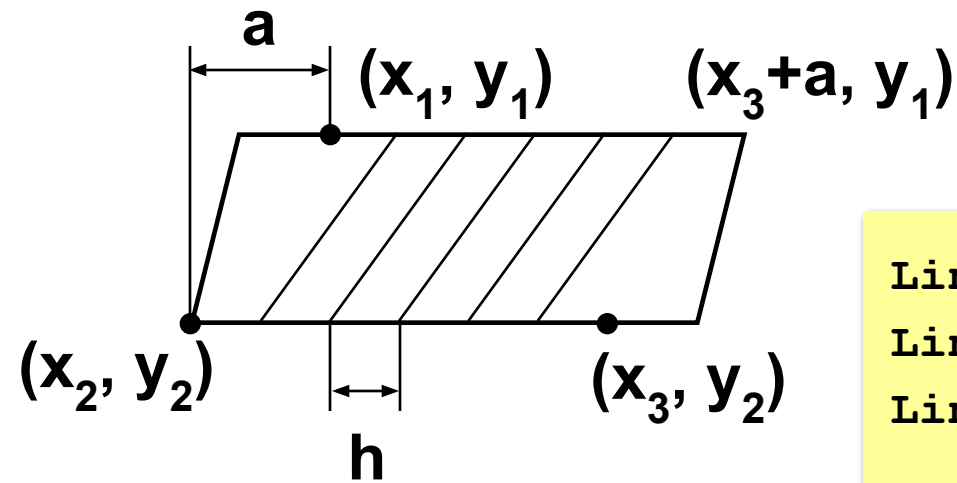


```

program qq;
var i, x1, x2, y1, y2, N: integer;
    h, x: real;
begin
    x1 := 100; y1 := 100;
    x2 := 300; y2 := 200;
    N := 10;
    Rectangle (x1, y1, x2, y2);
    h := (x2 - x1) / (N + 1);
    x := x1 + h;
    for i:=1 to N do begin
        Line(round(x), y1, round(x), y2);
        x := x + h;
    end;
end.

```

Штриховка



$$a = x_1 - x_2$$

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

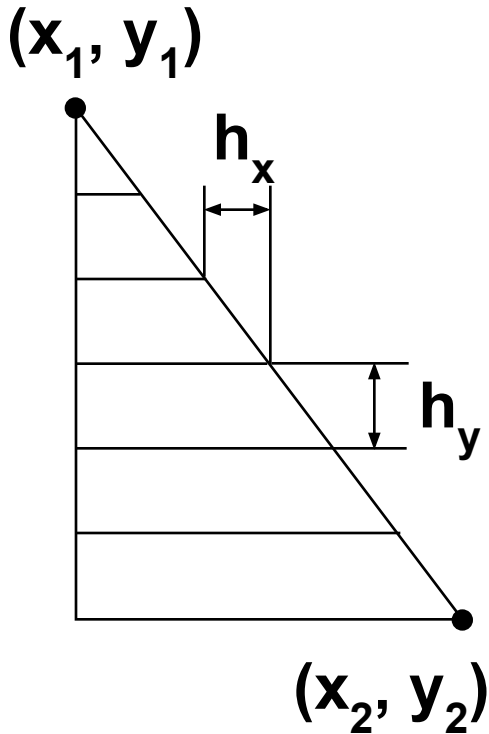
```
Line( x1+h, y1, x1+h-a, y2 );
Line( x1+2*h, y1, x1+2*h-a, y2 );
Line( x1+3*h, y1, x1+3*h-a, y2 );
...
```

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



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

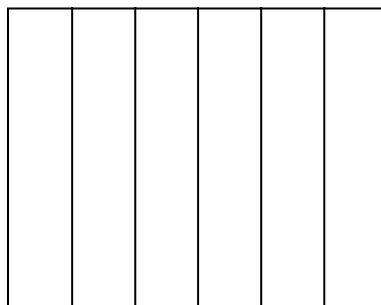
$$h_y = \frac{y_2 - y_1}{N + 1}$$

```
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);
```

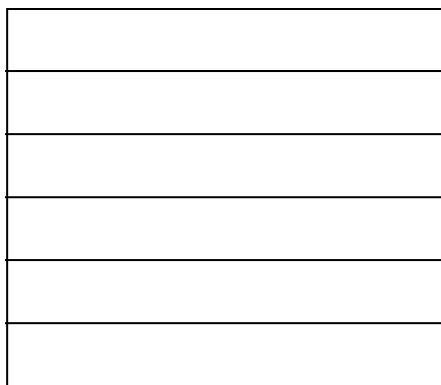
```
hx := (x2 - x1) / (N + 1);
hy := (y2 - y1) / (N + 1);
x := x1 + hx; y := y1 + hy;
for i:=1 to N do begin
  Line(x1, round(y), round(x), round(y));
  x := x + hx; y := y + hy;
end;
```

Задания

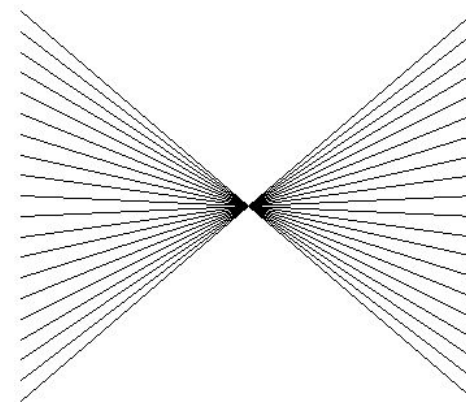
«3»: Ввести с клавиатуры количество линий,
построить фигуру и выполнить штриховку:



«4»: Ввести с клавиатуры количество линий,
построить фигуру и выполнить штриховку:

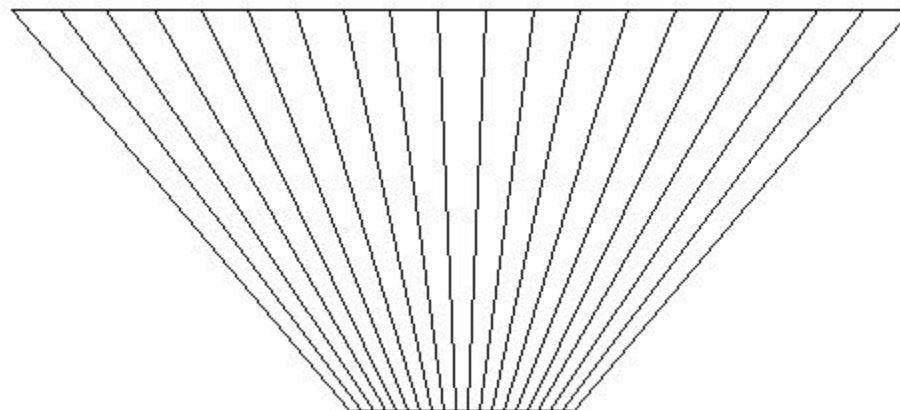


или



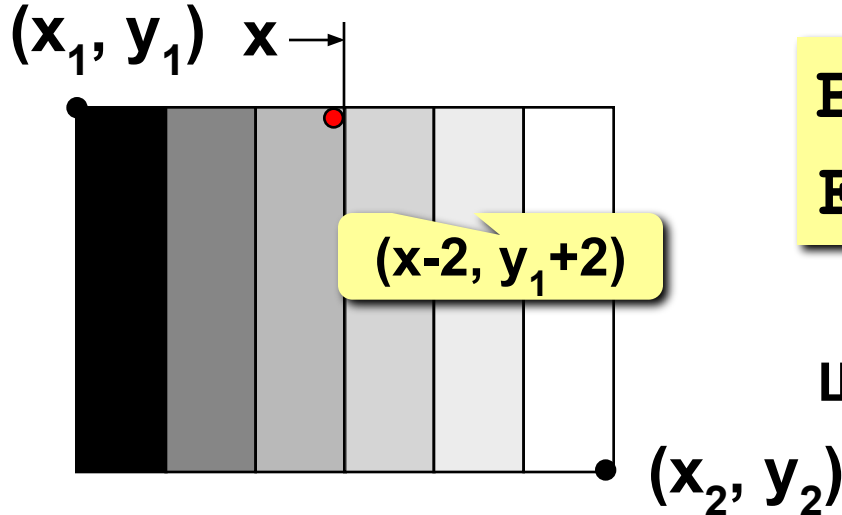
Задания

«5»: Ввести с клавиатуры количество линий и построить фигуру:



Как менять цвет?

серый: $R = G = B$



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

Шаг изменения c :

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

```
hc := 255 div N;
c := 0;
```

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

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

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

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

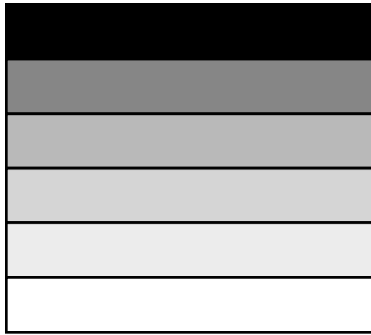
```
  Fill(round(x)-2, y1+2);
```

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

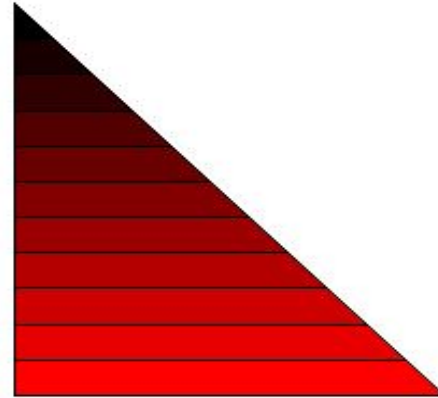
```
end;
```

Задания

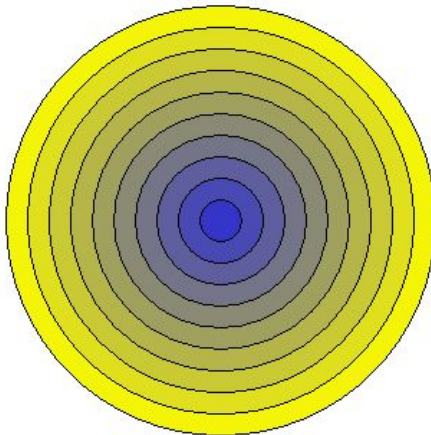
«4»: Ввести с клавиатуры число линий штриховки и построить фигуру, залив все области разным цветом.



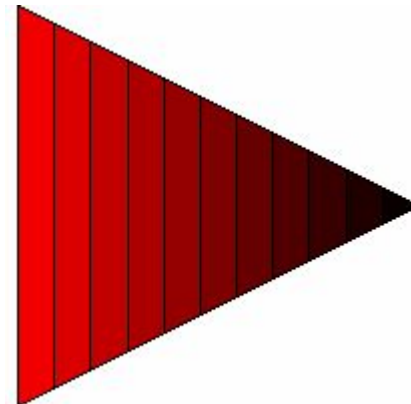
или



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



или

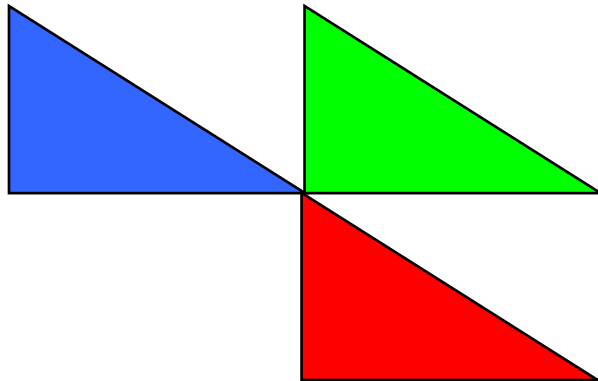


Программирование на языке Паскаль

Тема 7. Процедуры

Процедуры

Задача: Построить фигуру:



Можно ли решить известными методами?

Общность: три похожие фигуры.

общее: размеры, угол поворота

отличия: координаты, цвет



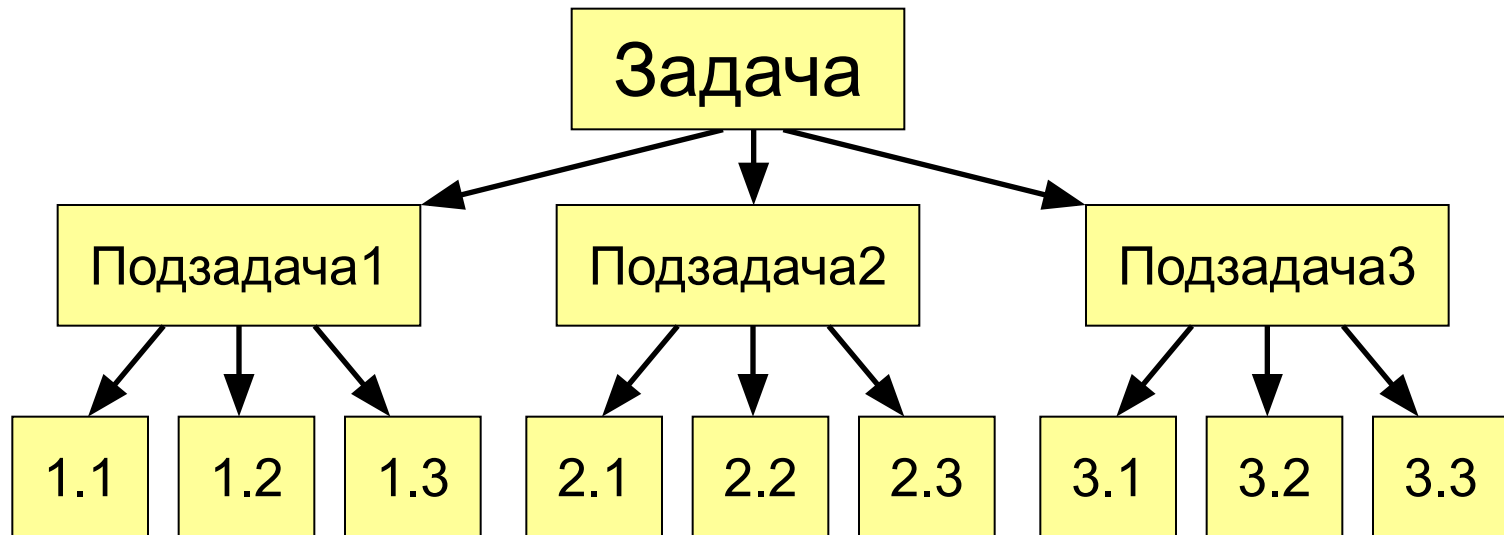
Сколько координат надо задать?

Процедуры

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

Применение:

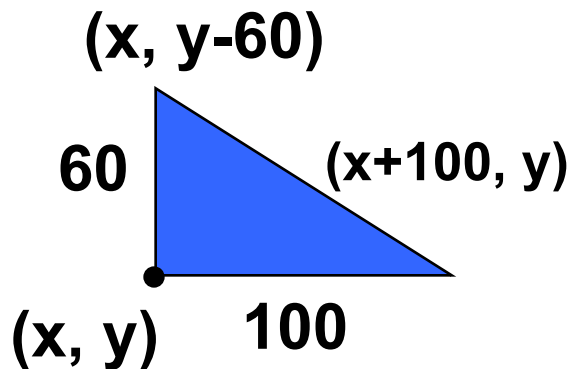
- выполнение одинаковых действий в разных местах программы
- разбивка программы (или другой процедуры) на подзадачи для лучшего восприятия



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;
  
```

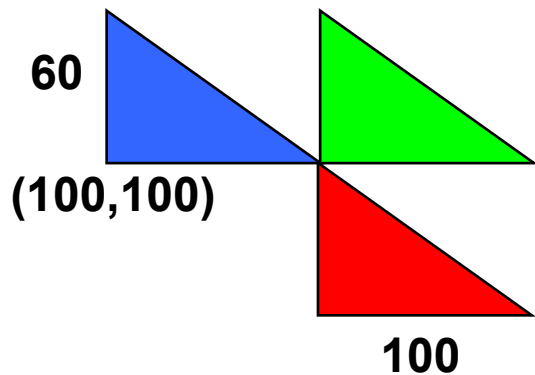
цвет

координаты

тело
процедуры

Программа

формальные параметры



ВЫЗОВЫ
процедуры

```
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
```

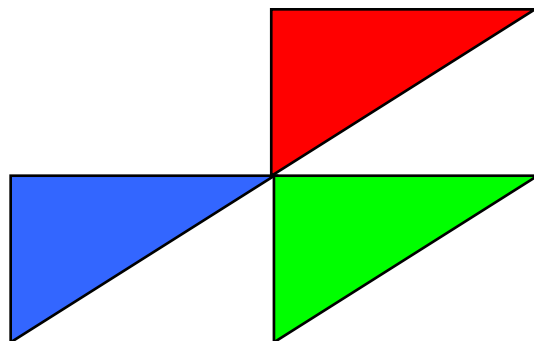
```
  ...
```

```
end.
```

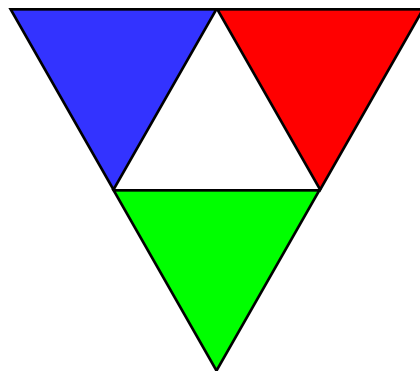
локальные
переменные

Задания

«3»: Используя одну процедуру, построить фигуру.

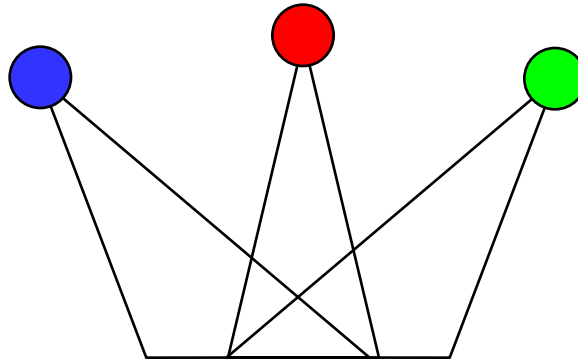


«4»: Используя одну процедуру, построить фигуру.



Задания

«5»: Используя одну процедуру, построить фигуру.



Построение графиков функций

Задача: построить график функции $y = x^2$ на интервале от -2 до 2.

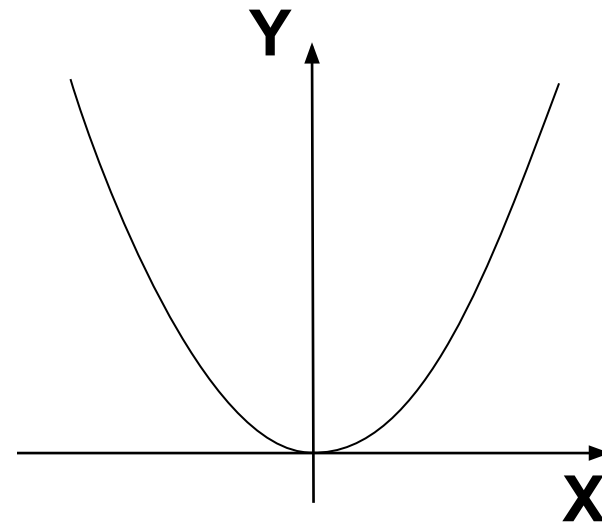
Анализ:

максимальное значение

$$y_{\max} = 4 \quad \text{при} \quad x = \pm 2$$

минимальное значение

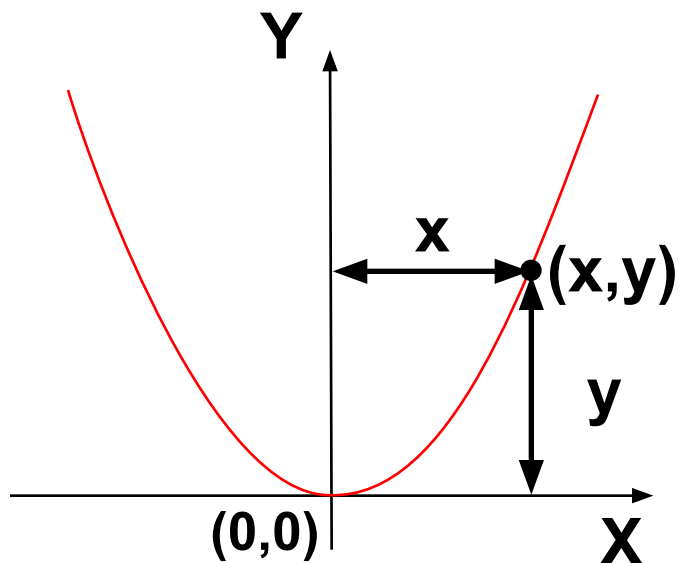
$$y_{\min} = 0 \quad \text{при} \quad x = 0$$



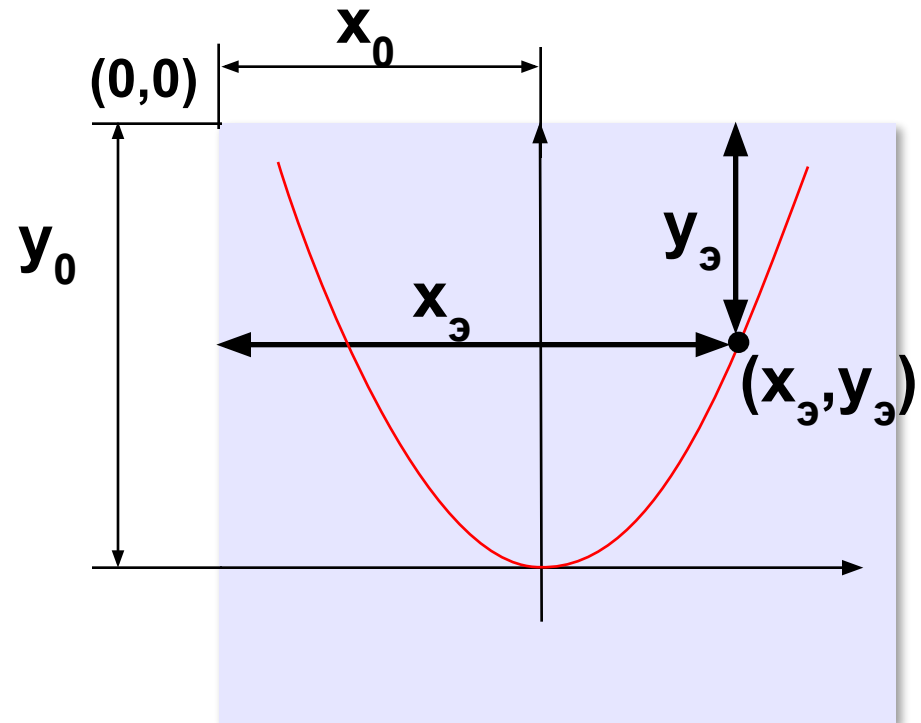
Проблема: функция задана в математической системе координат, строить надо на экране, указывая координаты в пикселях.

Преобразование координат

Математическая
система координат



Экранная система
координат (пиксели)



k – масштаб (длина
изображения единичного
отрезка на экране)

$$x_э =$$

$$y_э =$$

Программа

```
program qq;
const x0 = 150; y0 = 250; k = 50;
      xmin = -2; xmax = 2;
var x, y, h: real;
     xe, ye: integer;
begin
  Line(0, y0, x0+150, y0);
  Line(x0, 0, x0, y0+20);
  x := xmin; h := 0.02;
  while x <= xmax do begin
    y := x*x;
    xe := x0 + round(k*x);
    ye := y0 - round(k*y);
    Point (xe, ye);
    x := x + h;
  end;
end.
```

на экране

h – шаг изменения x

цикл
построения
графика

оси координат



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

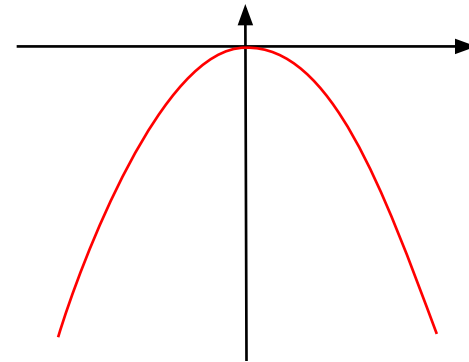
начальное значение

Задания

«3»: Построить график функции

$$y = -x^2$$

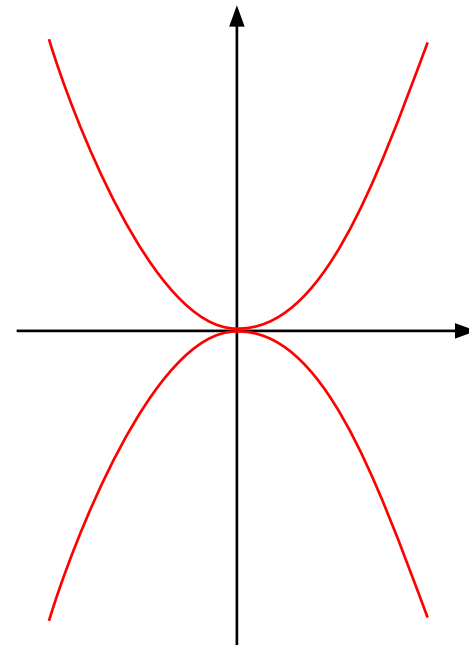
на интервале $[-2,2]$.



«4»: Построить графики функций

$$y = x^2 \text{ и } y = -x^2$$

на интервале $[-2,2]$.

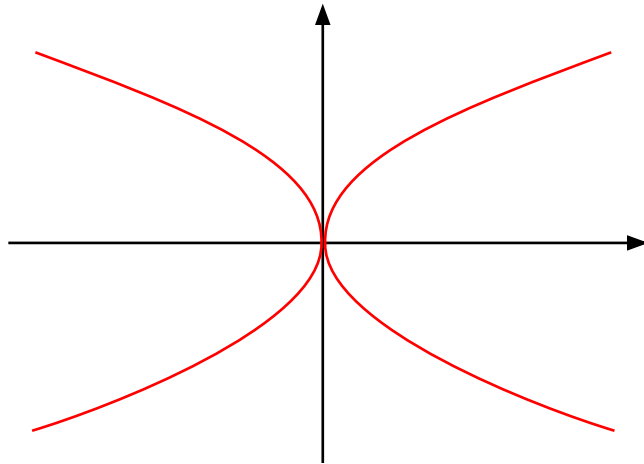


Задания

«5»: Построить графики функций

$$x = y^2 \quad \text{и} \quad x = -y^2$$

на интервале $[-2,2]$.



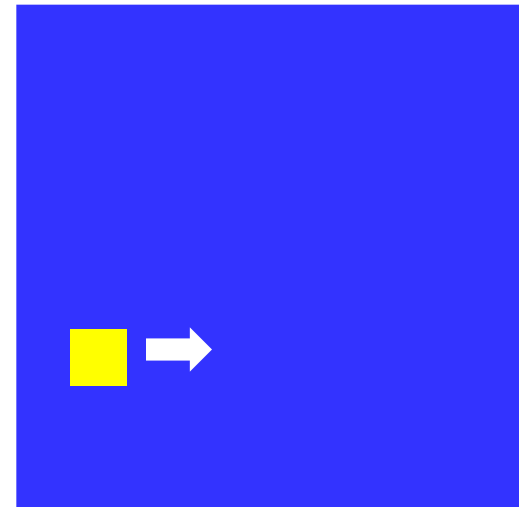
Программирование на языке Паскаль

Тема 8. 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;
```

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

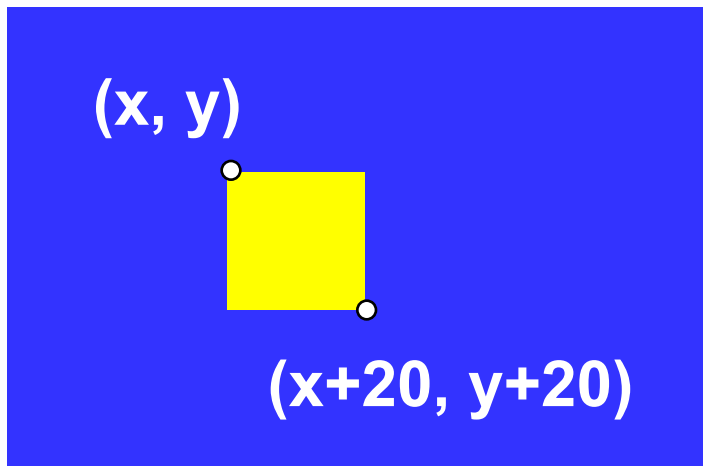
запуск цикла

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

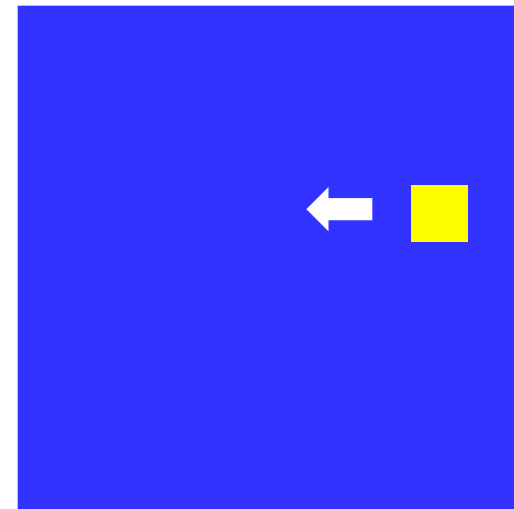
выход по
клавише Esc

ждем 10 мс

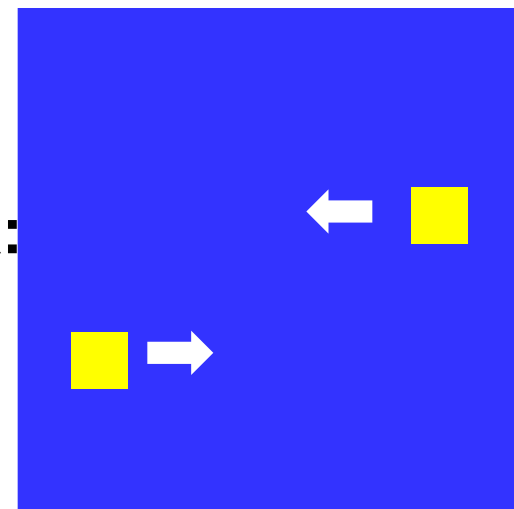
выход при
касании границы

Задания

«3»: Квадрат двигается справа налево:

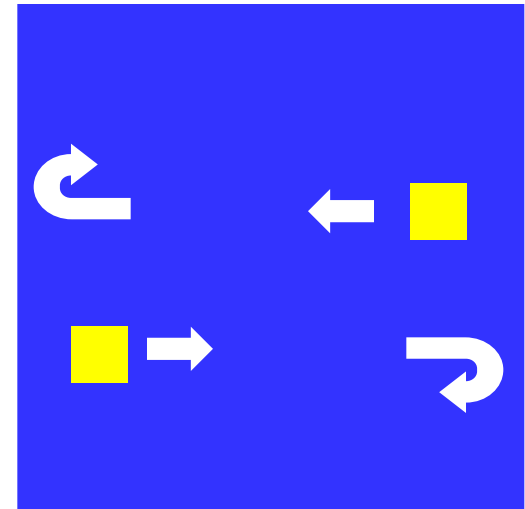


«4»: Два квадрата двигаются в противоположных направлениях:



Задания

«5»: Два квадрата двигаются в противоположных направлениях 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

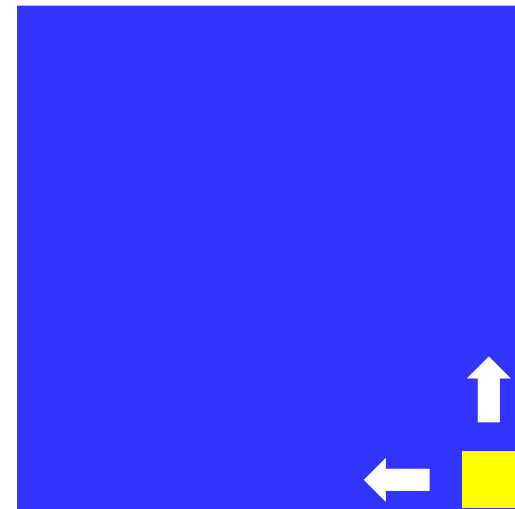
```
end.
```



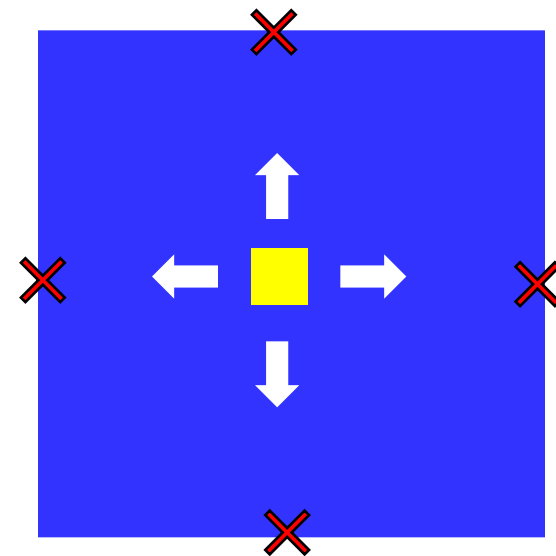
Что можно улучшить?

Задания

«3»: Квадрат в самом начале стоит в правом нижнем углу, и двигается при нажатии стрелок только вверх или влево:

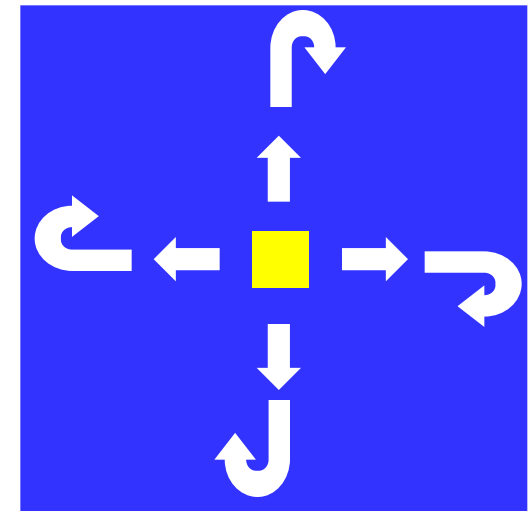


«4»: Квадрат двигается при нажатии стрелок, однако не может выйти за границы синего квадрата:



Задания

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



Программирование на языке Паскаль

Тема 9. Функции

Функции

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

Примеры:

- Вычисление модуля и других функций
- 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

формальные параметры

Функция:

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

локальная
переменная

результат -
целое число

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;
```

- в конце заголовка через двоеточие указывается **тип результата**

- функция

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

 ММЫ

Функции

Особенности:

- МОЖНО ОБЪЯВЛЯТЬ И ИСПОЛЬЗОВАТЬ **локальные переменные**

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

- знач... зается в
пере... М
функции; объявлять ее **НЕ НАДО**:

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

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!

Задания

«3»: Составить функцию, которая определяет наименьшее из трех чисел и привести пример ее использования.

Пример:

Введите два числа:

28 15

Наименьшее число – 15.

«4»: Составить функцию, которая определяет наибольшее из трех чисел и привести пример ее использования.

Пример:

Введите три числа:

28 15 10

Наибольшее число – 28.

Задания

«5»: Составить функцию, которая определяет сумму всех чисел от 1 до N и привести пример ее использования.

Пример:

Введите число :

100

сумма = 5050

Логические функции

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

Особенности:

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

Алгоритм: если число делится на 2, оно четное.

```
if N mod 2 = 0 then
    { число N четное }
else { число N составное }
```

Логические функции

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

результат – логическое значение

```
function Chet(N: integer): boolean;  
begin  
    if N mod 2 = 0 then  
        Chet := True  
    else Chet := False;  
end;
```

ИЛИ

$\text{Chet} := (\text{N mod } 2) = 0;$

```
begin  
    writeln('Введите целое число');  
    read(N);  
    if Chet(N) then  
        writeln(N, ' – четное число')  
    else writeln(N, ' – нечетное число');  
end.
```

ВЫЗОВ функции

Задания

«3»: Составить функцию, которая определяет, верно ли, что число оканчивается на 0.

Пример:

Введите число: Введите число:

170 237

Верно. Неверно.

«4»: Составить функцию, которая определяет, верно ли, что в числе вторая цифра с конца больше 6.

Пример:

Введите число: Введите число:

178 237

Верно. Неверно.

Задания

«5»: Составить функцию, которая определяет, верно ли, что переданное ей число – простое (делится только на само себя и на единицу).

Пример:

Введите число: Введите число:

29 28

Простое число. Составное число.

Программирование на языке Паскаль

Тема 10. Случайные числа

Случайные процессы

Случайно...

- 1) встретить друга на улице
- 2) разбить тарелку
- 3) найти 10 рублей
- 4) выиграть в лотерею

Случайный выбор:

- 1) жеребьевка на соревнованиях
- 2) выигравшие номера в лотерее

Как получить случайность?



Случайные числа на компьютере

Электронный генератор



- нужно специальное устройство
- нельзя воспроизвести результаты

Псевдослучайные числа – обладают свойствами случайных чисел, но каждое следующее число вычисляется по заданной формуле.

Метод середины квадрата (Дж. фон Нейман)

564321

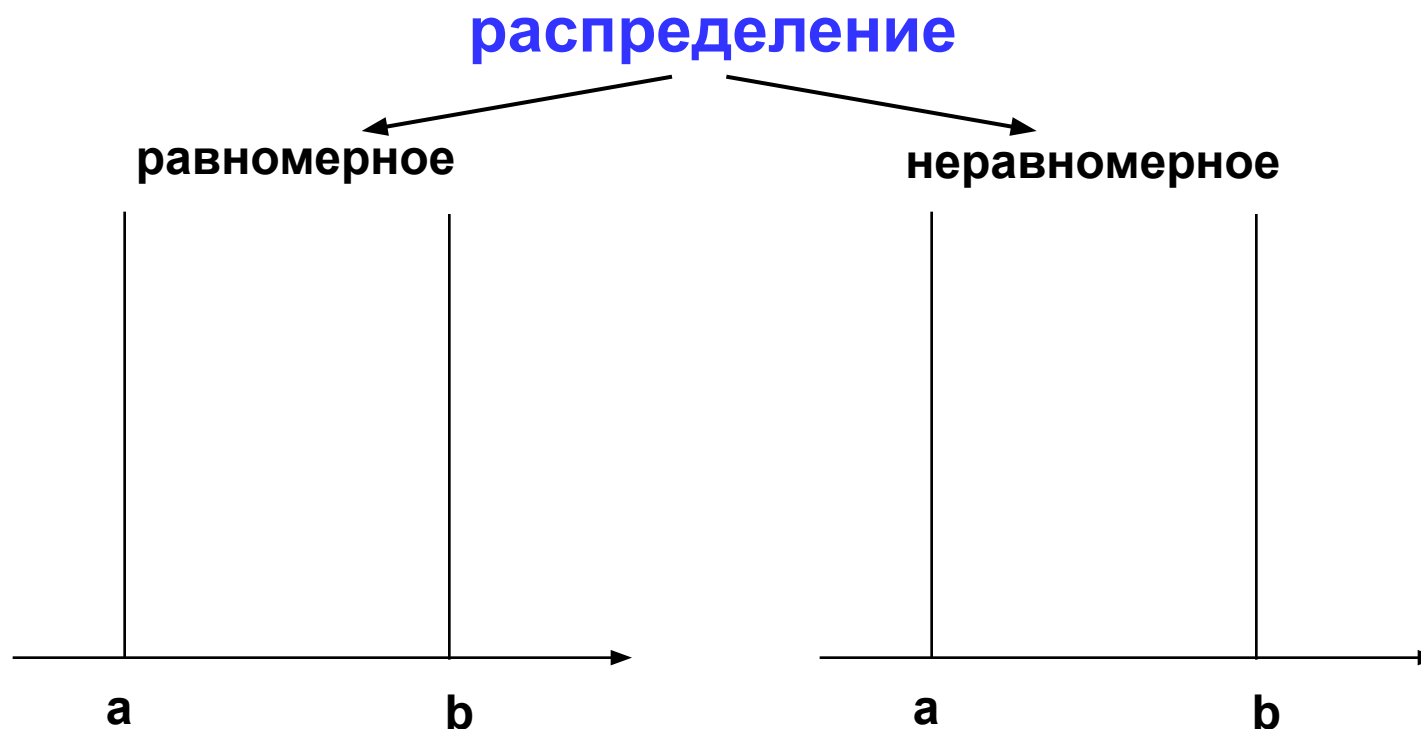
318458191041

209938992481

в квадрате малый период
(последовательность
повторяется через 10^6 чисел)

Распределение случайных чисел

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

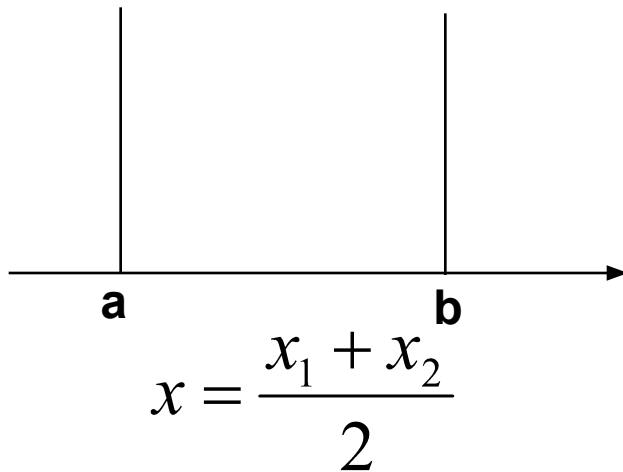


Сколько может быть разных распределений?

Распределение случайных чисел

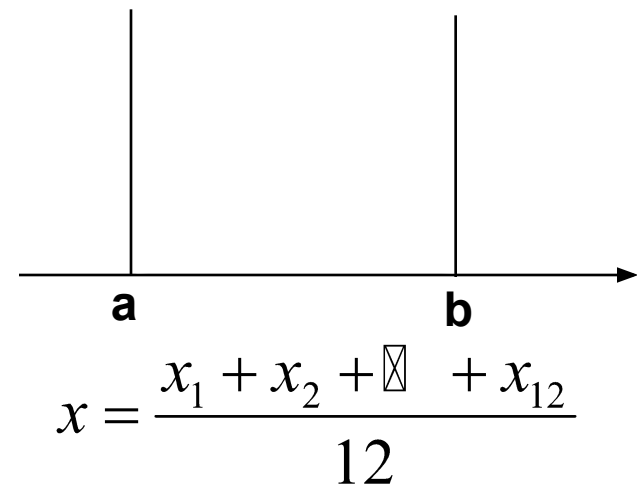
Особенности:

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



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

равномерное распределение



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

неравномерное 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)

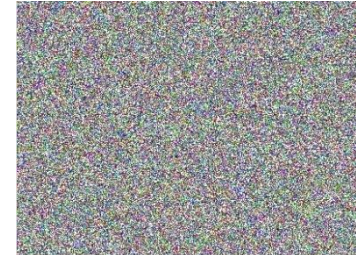
```
var x: real;
```

```
...
```

```
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; случайные координаты  
    repeat  
        x := random(400);  
        y := random(300); случайный цвет  
        Pen(1, random(256), random(256), random(256));  
        Point(x, y);  
        if IsEvent then begin ВЫХОД ПО КЛАВИШЕ Esc  
            Event(k, code, i);  
            if (k = 1) and (code = 27) then stop := True;  
        end;  
    until stop;  
end.
```

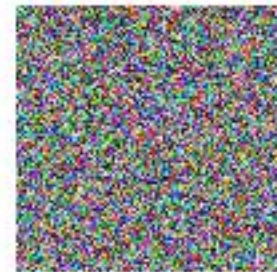
Задания

«3»: Заполнить квадрат точками случайного цвета.
размер квадрата ввести с клавиатуры:

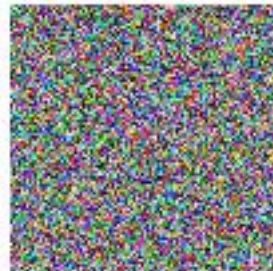
Пример:

Введите размер квадрата:

150

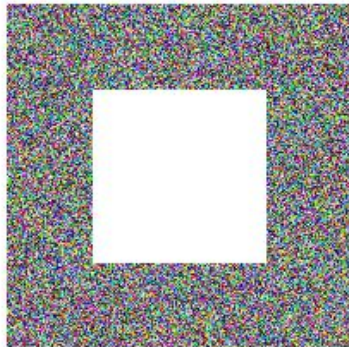


«4»: Заполнить область точками случайного цвета:

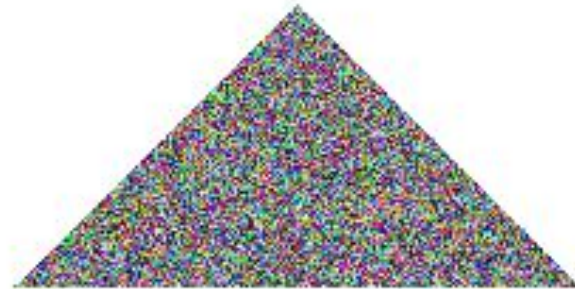


Задания

«5»: Заполнить область точками случайного цвета:

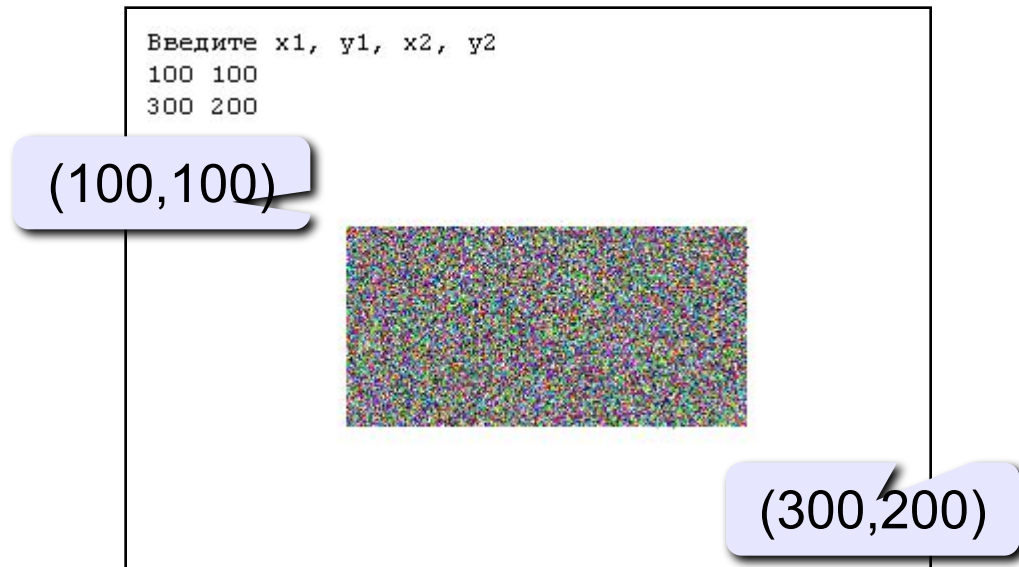


или



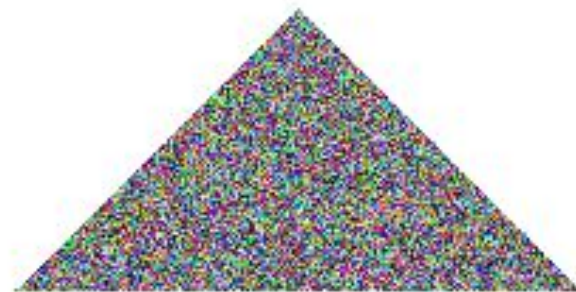
Задания

«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° .



Конец фильма
