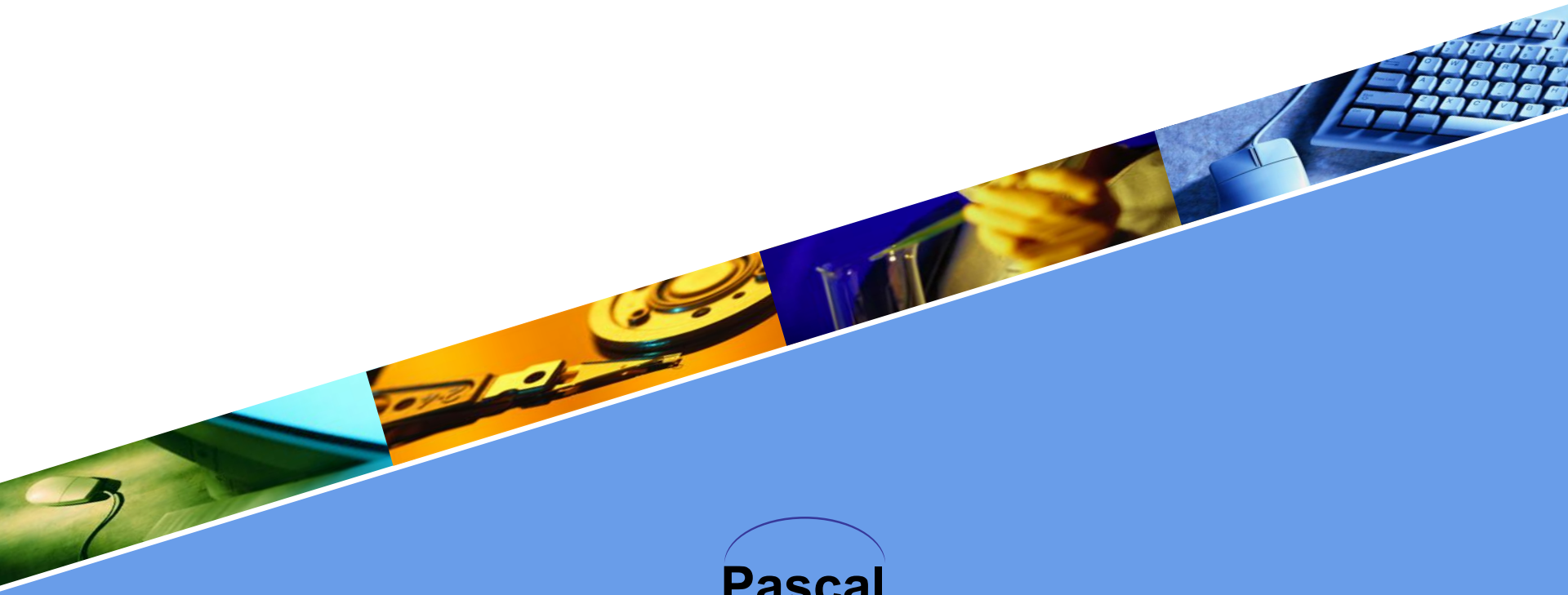
Алгоритмы разветвляющейся структуры, программирование на языке Pascal

10 «А» класс

Pascal

Теория



Pascal

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.**

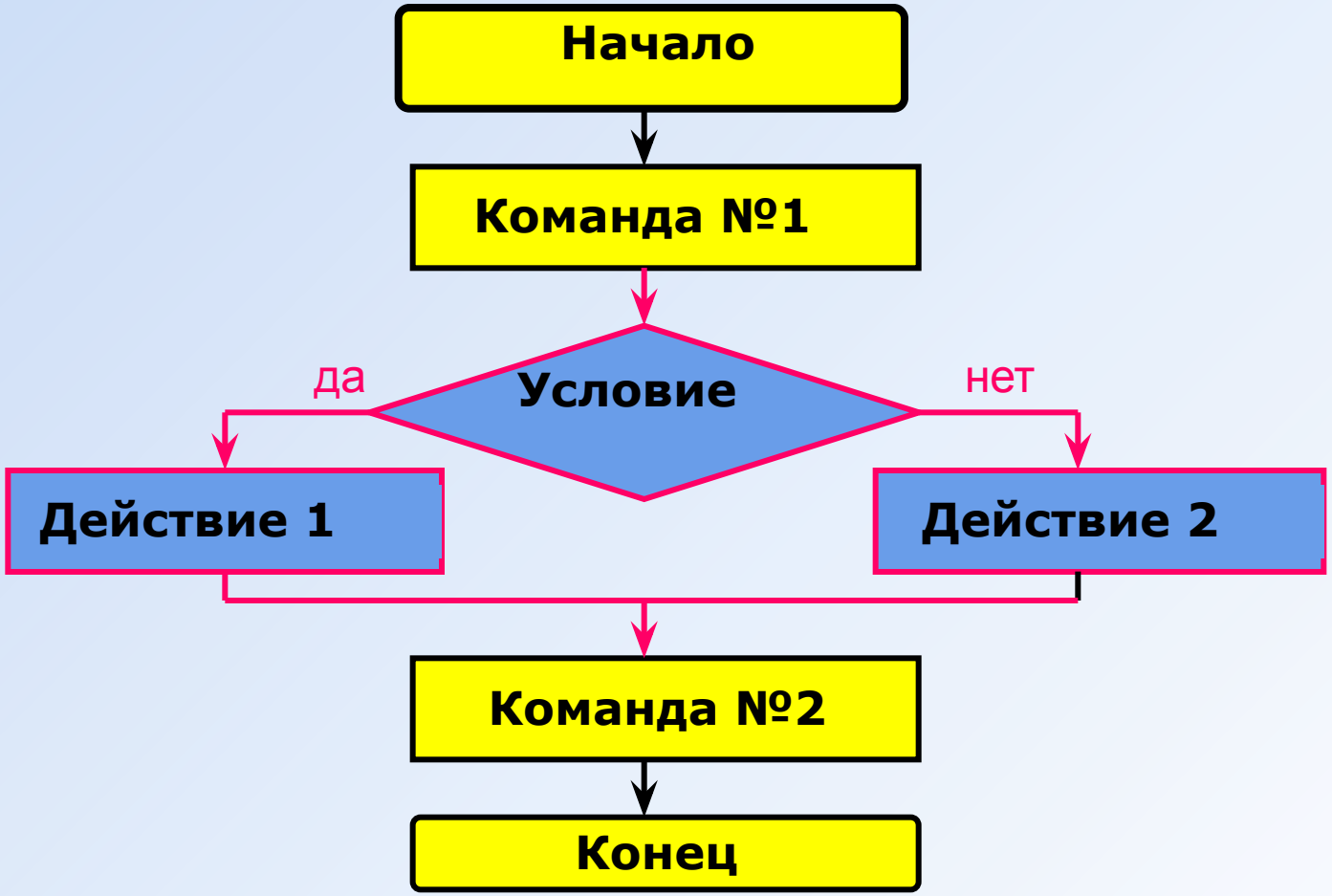
Алгоритм, представленный *словесным* способом описания

Начало

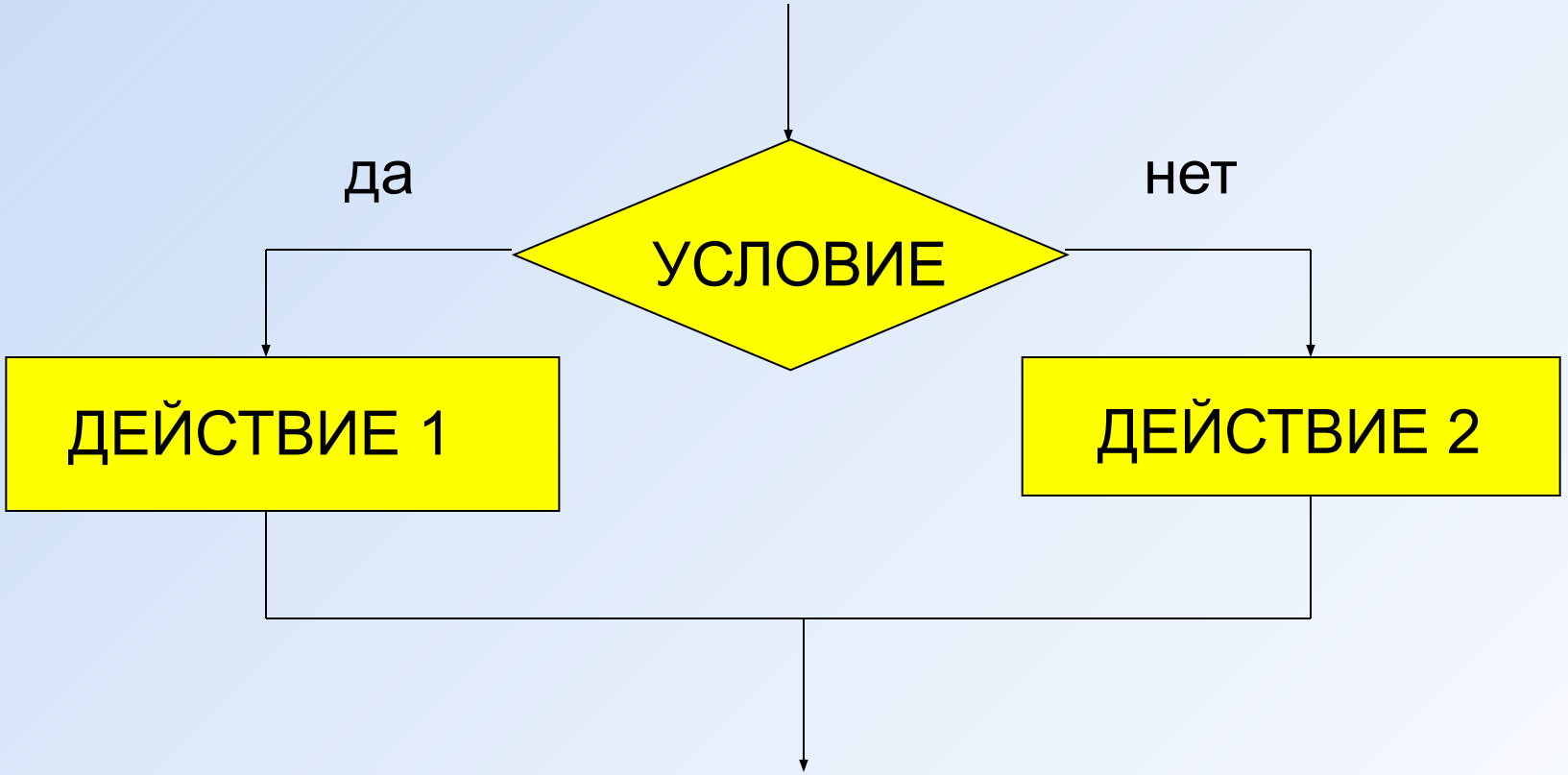
1. Команда №1
2. **ЕСЛИ** условие **ТО** действие1
   ИНАЧЕ действие2
3. Команда №2

Конец

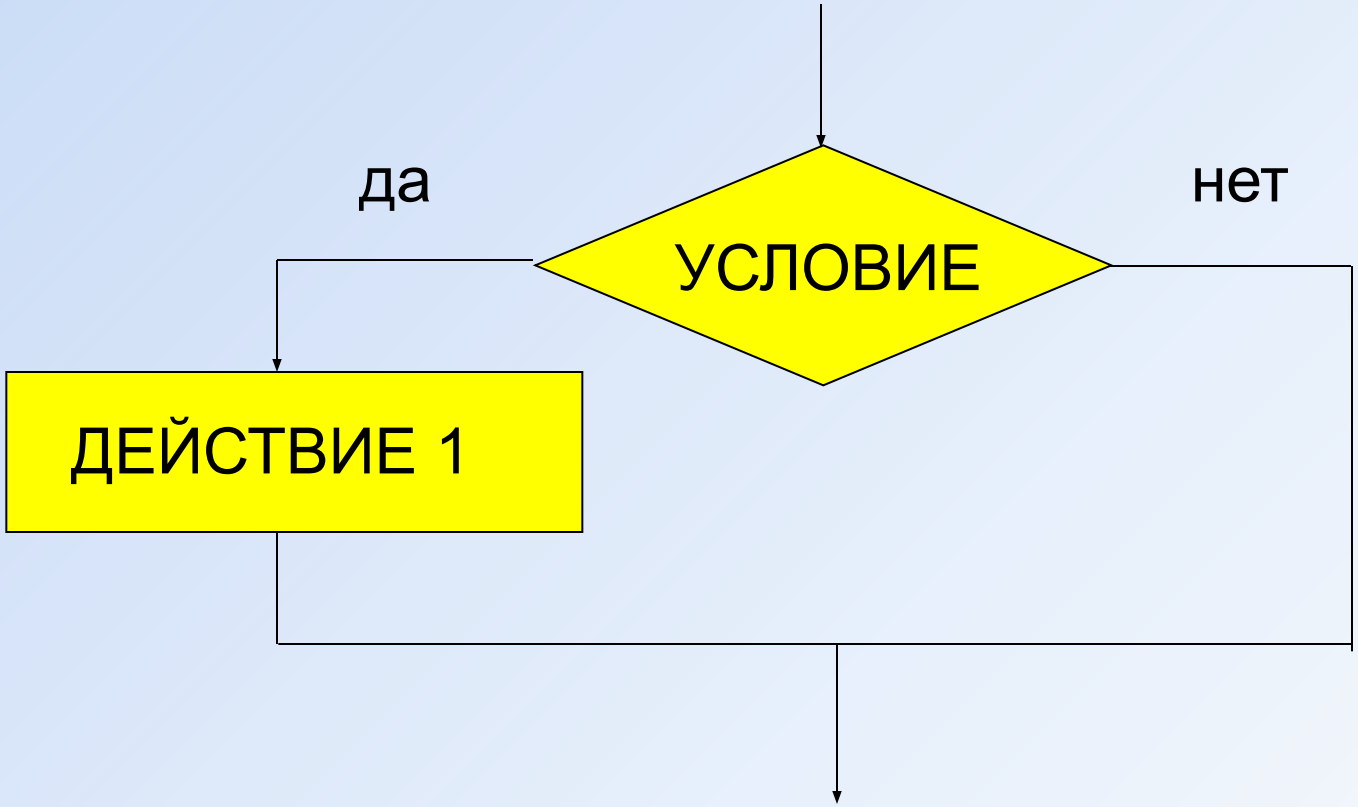
Алгоритм, представленный *графическим* способом описания



Полная форма ветвления



Неполная форма ветвления



Условный оператор

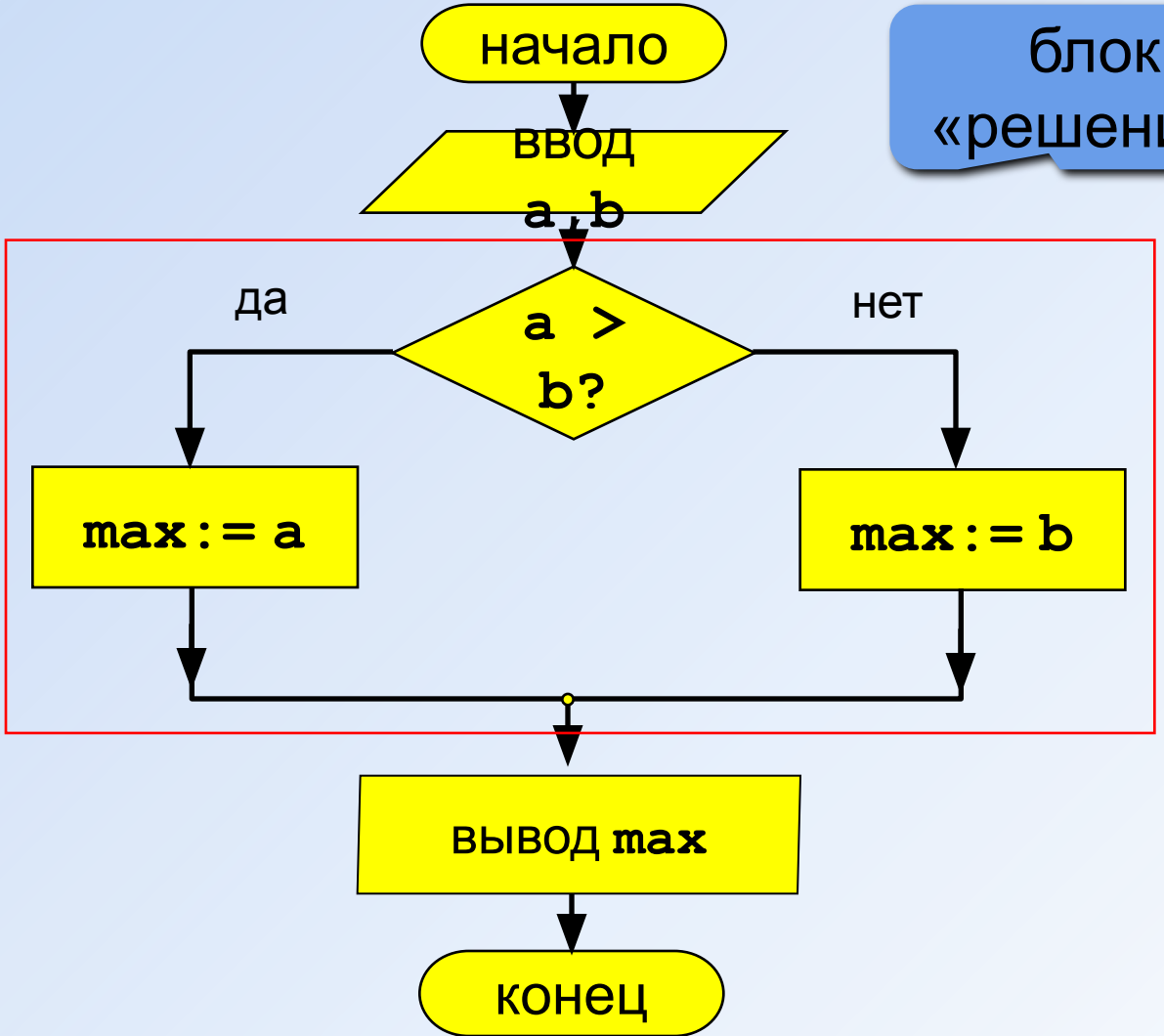
```
if <условие> then begin
   {что делать, если условие верно}
end
else begin
   {что делать, если условие неверно}
end;
```

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***

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

1 способ решения. Блок-схема



блок «решение»

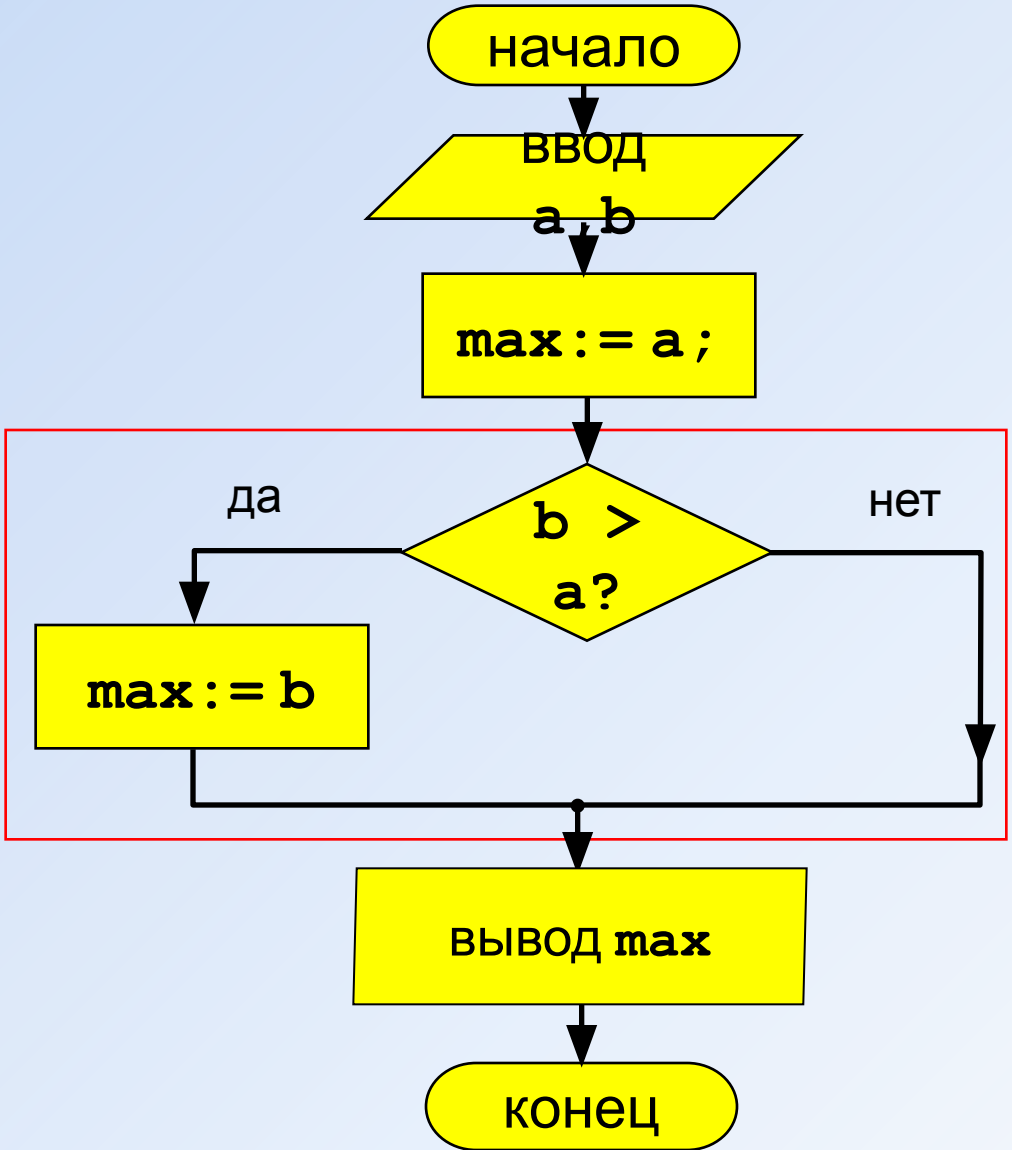
полная форма ветвления

1 способ решения. 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

2 способ решения. Блок-схема



неполная форма ветвления

2 способ решения. Программа

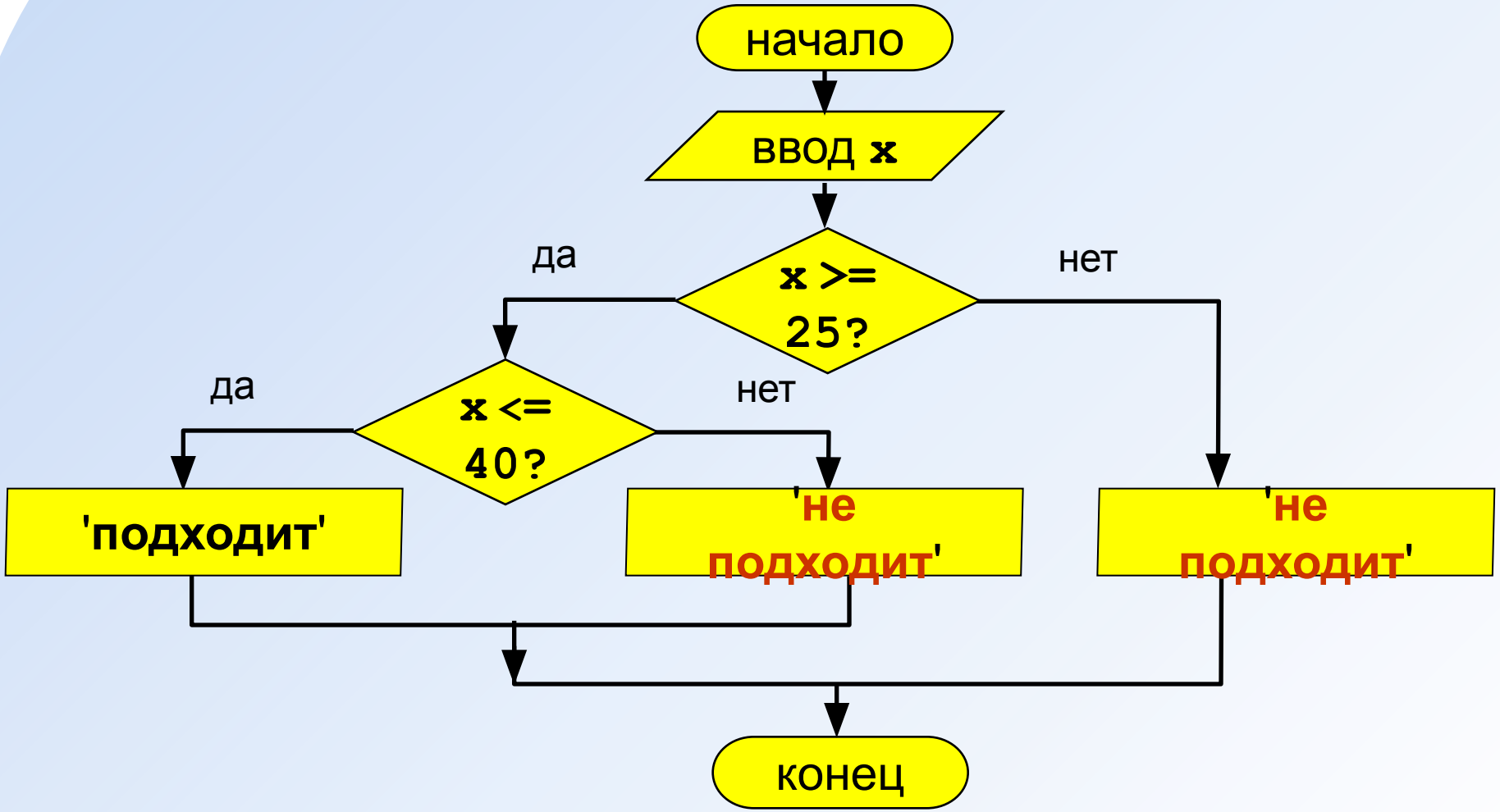
```
program qq;
var a, b, max: integer;
begin
   writeln('Введите два целых числа');
   read ( a, b );
   max := a;
   if b > a then
      max := b;
   writeln ('Наибольшее число ', max);
end.
```

неполная форма условного оператора

2 способ решения. Программа другая

```
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.
```

1 способ решения. Блок-схема



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»).

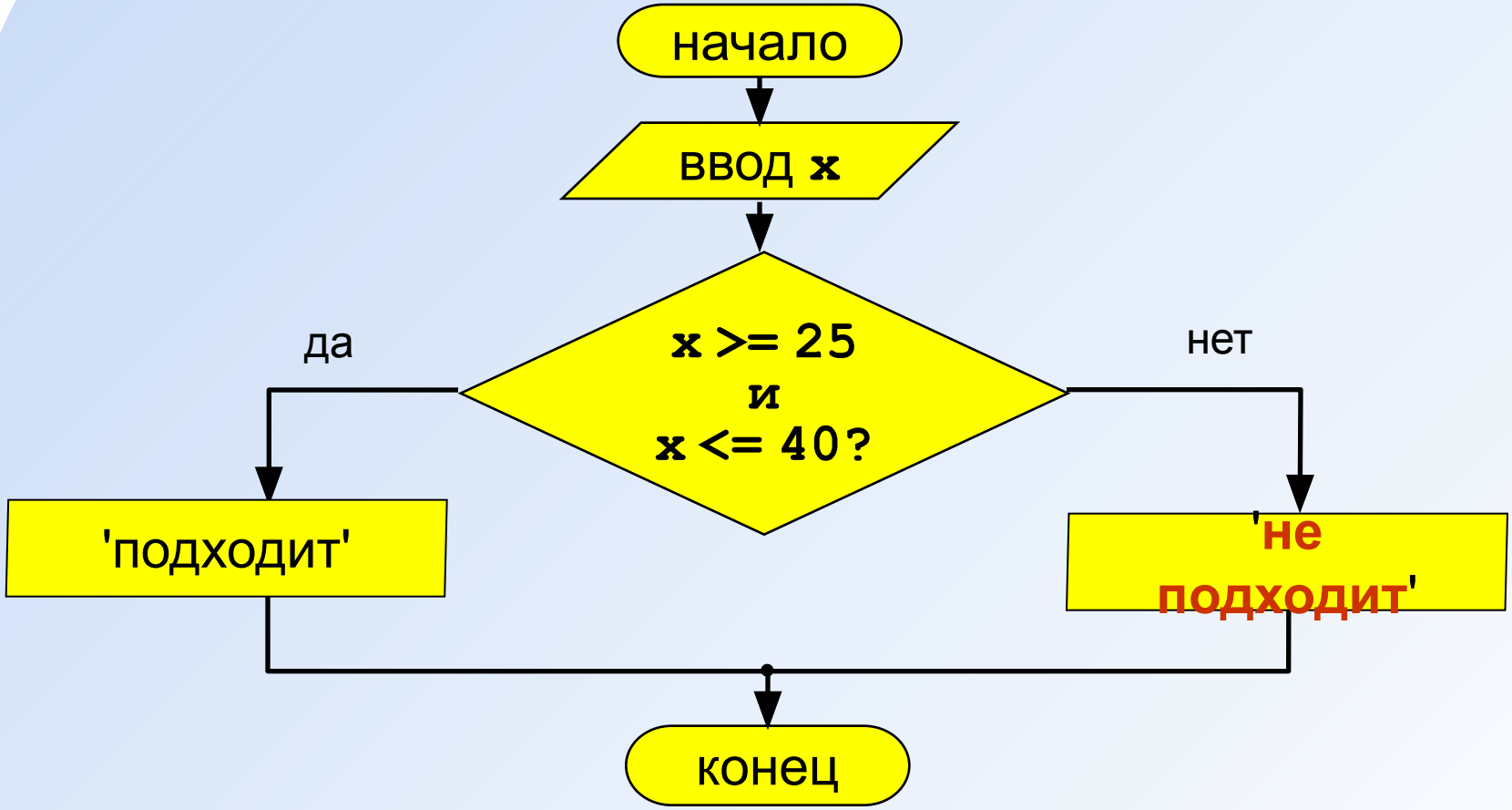
Особенность: надо проверить, выполняются ли два условия одновременно.

Можно ли решить известными методами?

1 способ решения. 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.
```

2 способ решения. Блок-схема



2 способ решения. 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 (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)

Сложные условия

Порядок выполнения (приоритет = старшинство)

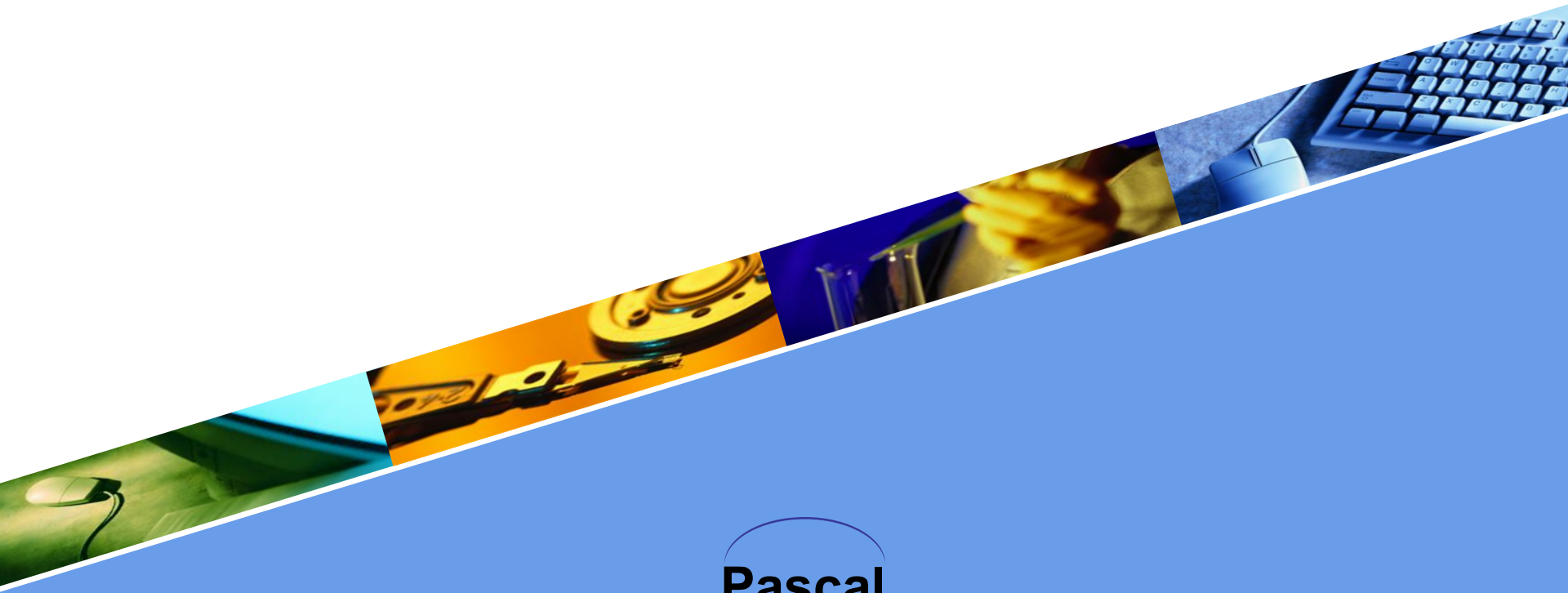
- выражения в скобках
- **not**
- **and**
- **or, xor**
- **<, <=, >, >=, =, <>**

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
```

Практика



Pascal

1. Написать алгоритм вычисления значения y, если

$$y = \begin{cases} 12x^2, если & x \le 16, \\ 3x - x^3, если & x > 16. \end{cases}$$

Алгоритм

Функция задана двумя различными аналитическими выражениями на двух участках координатной оси.

Если x<=16, то y=12*x*x.

Если же x>16, то y=3*x-x*x*x.

Второе неравенство является противоположным первому, поэтому достаточно поставить одно первое условие.

Начало

Ввод x ;

Если x<=16 **то**

y=12*x*x

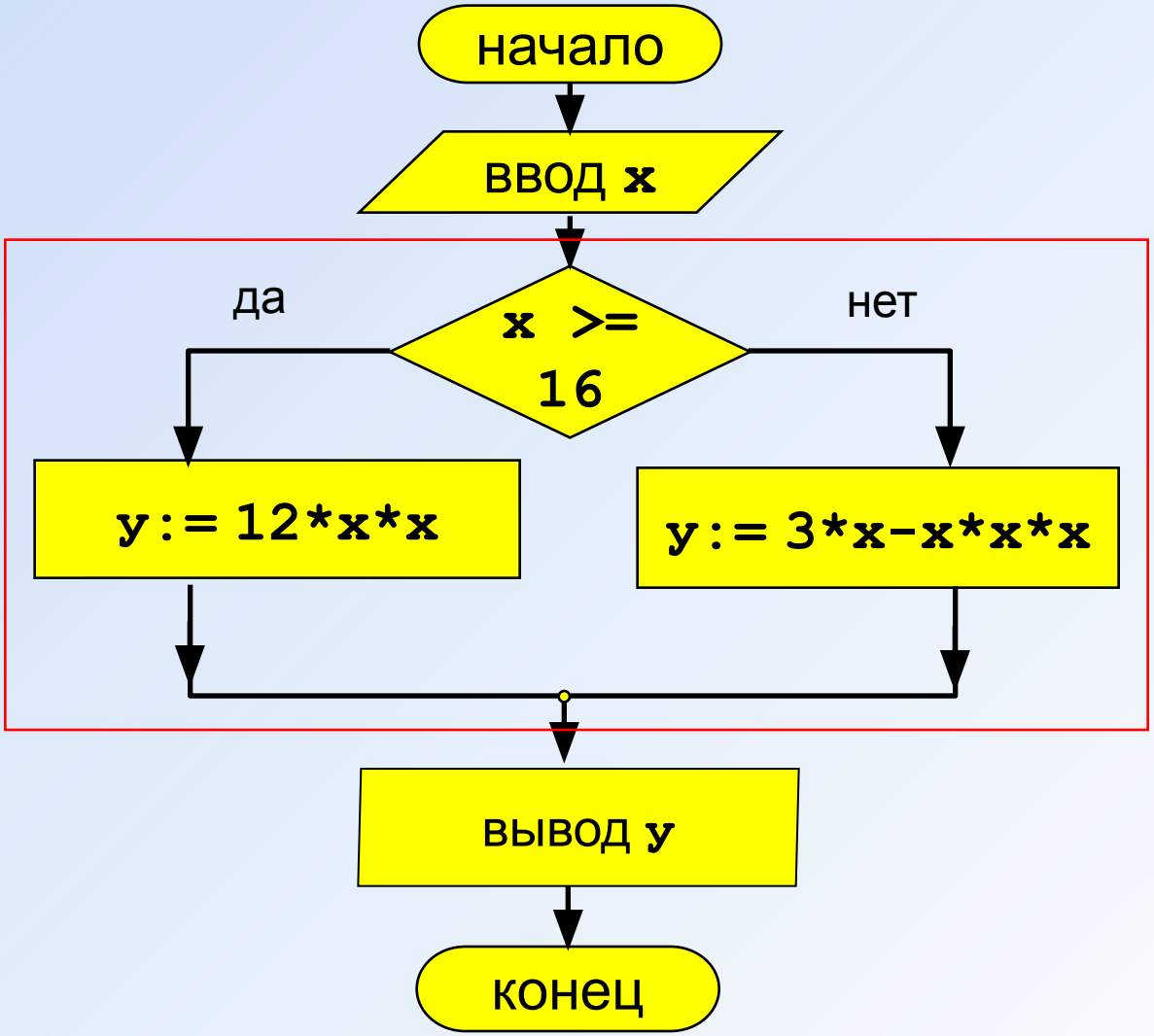
иначе

y=3*x-x*x*x;

Вывод y;

Конец.

Блок-схема



Программа

```
program qq;
   var x, y: real;
   begin
    writeln('Введите значение аргумента x');
    read ( x );
    if x >= 16 then
        y:=12*x*x
    else
        y:=3*x-x*x*x;
    writeln ('y=‘, y);
   end.
```

2. Определить является ли треугольник со сторонами a, b, c равносторонним треугольником.

Алгоритм

Треугольник является равносторонним, если все стороны равны между собой.

Начало

Ввод a,b,c **;**

Если a=b **и** b=c **то**

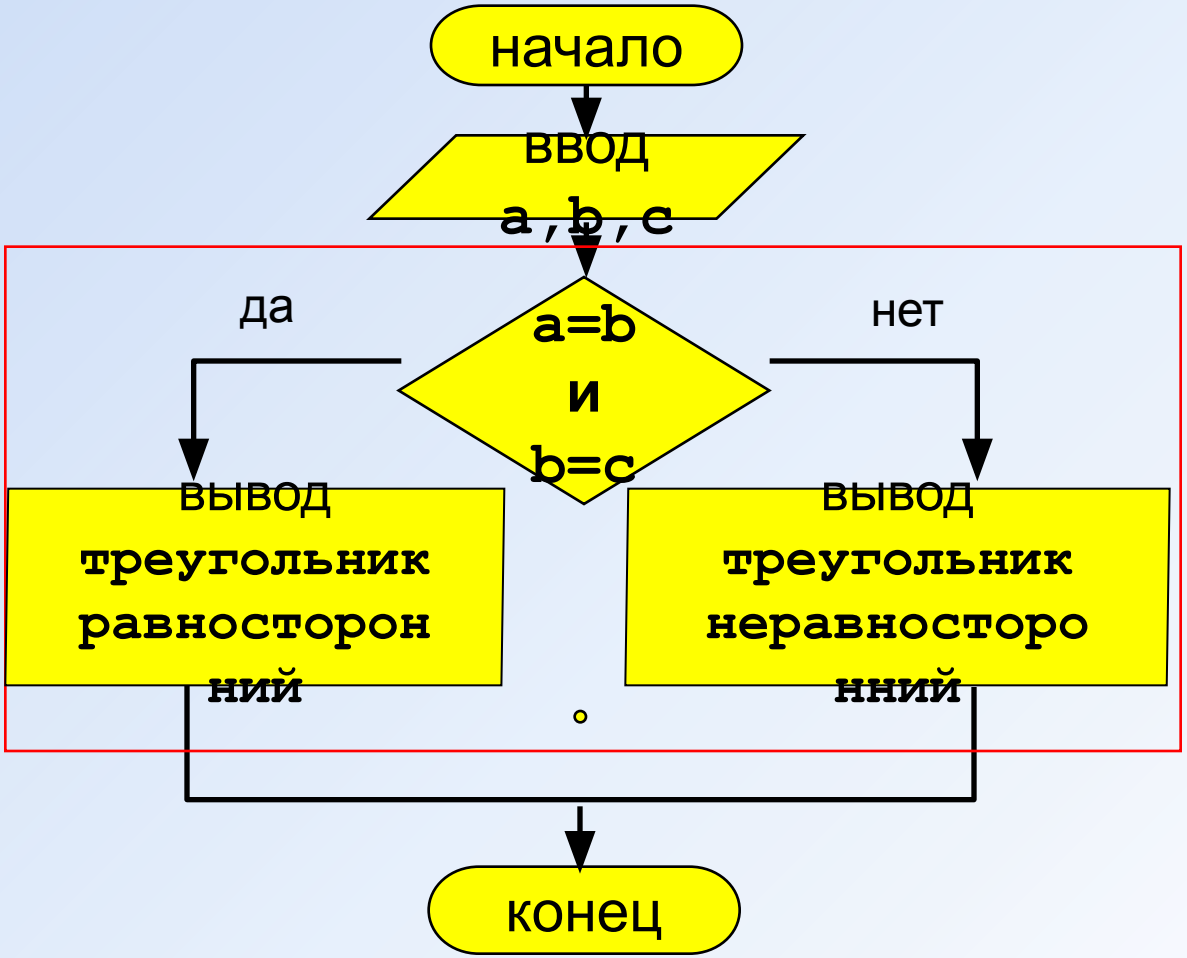
вывод (треугольник равносторонний)

иначе

вывод (треугольник неравносторонний)**;**

Конец.

Блок-схема

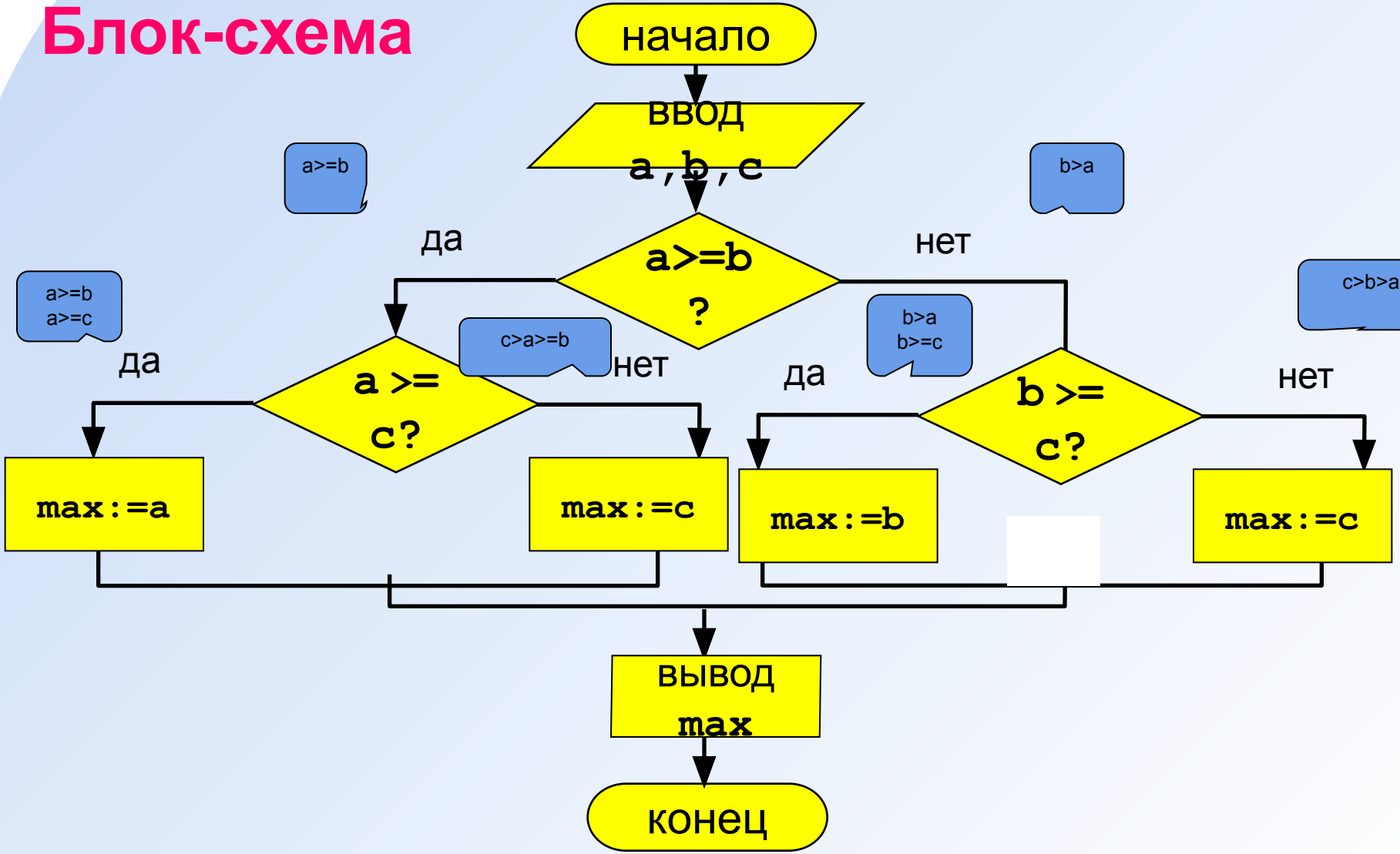

начало
ввод
a,b,c
a=b
и
b=c
да
нет
вывод
треугольник
равносторон
ний
вывод
треугольник
неравносторо
нний
конец

Программа

```
program qq;
  var x, y: real;
  begin
   writeln('Введите длины сторон a, b, c');
   read ( a,b,c );
   if (a=b) and (b=c) then
        writeln('треугольник равносторонний')
   else
        writeln('треугольник неравносторонний');
  end.
```

3. Найти наибольшее (максимум) среди трёх чисел.

Блок-схема



Программа

```
program qq;
var a, b, c, max: real;
begin
 writeln('Введите числа a, b, c');
 read ( a,b,c );
 if a>=b then
       if a>=c then
             max:=a
       else
             max:=c
 else
       if b>=c then
             max:=b
       else
             max:=c;
 writeln (max);
end.
```