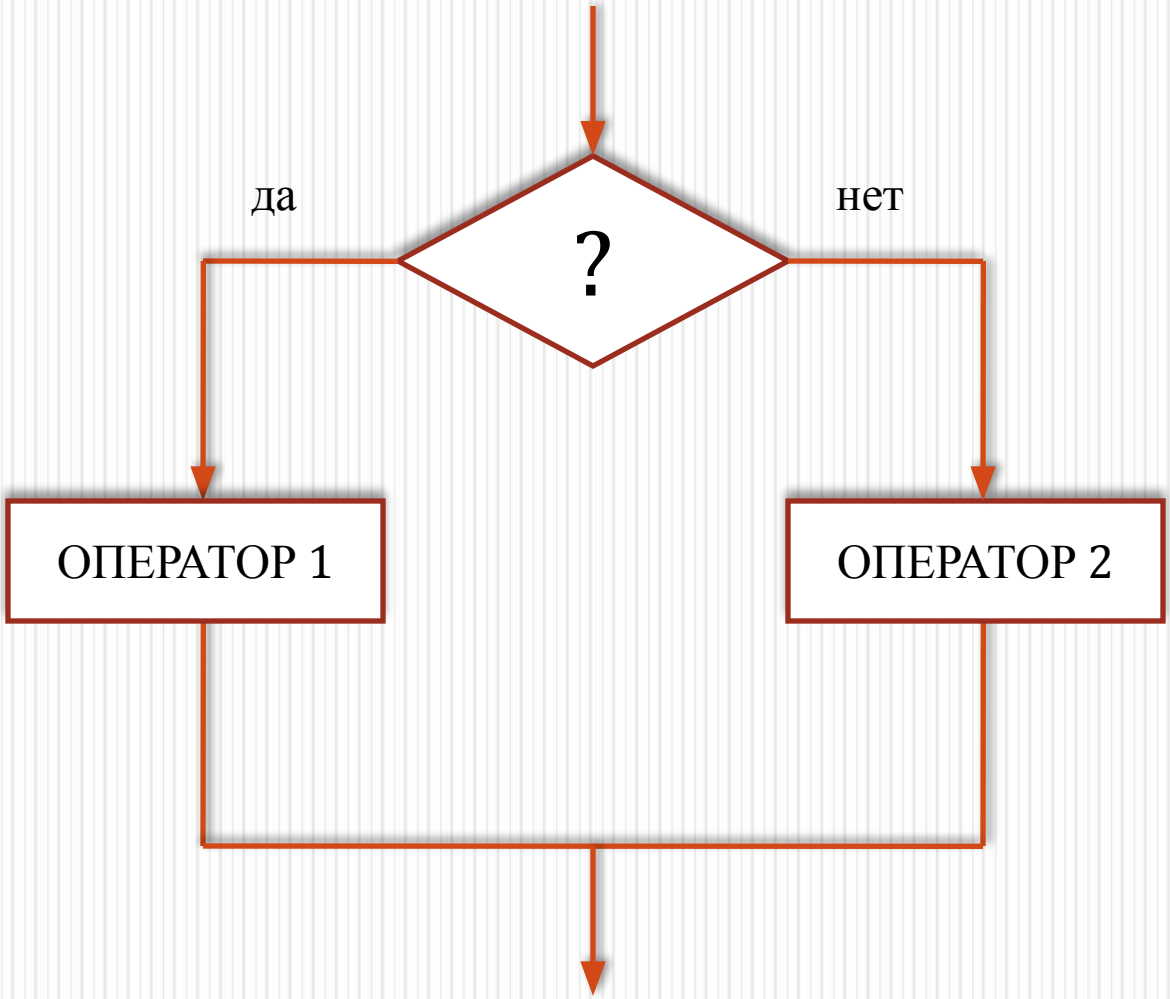
Pascal: Условный оператор

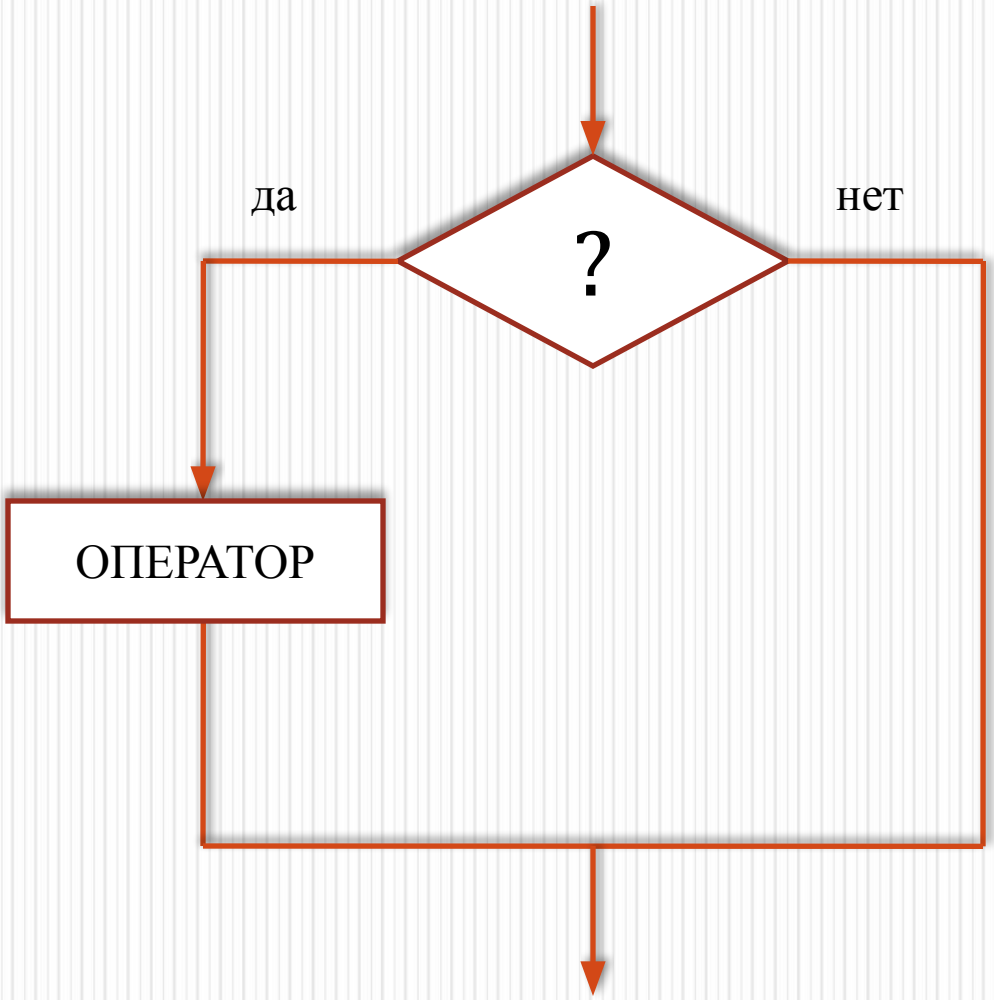
Во многих жизненных ситуациях принятие того или иного решения зависит от выполнения одного или нескольких условий.

Виктор Михайлович Васнецов.
Витязь на распутье (1878).

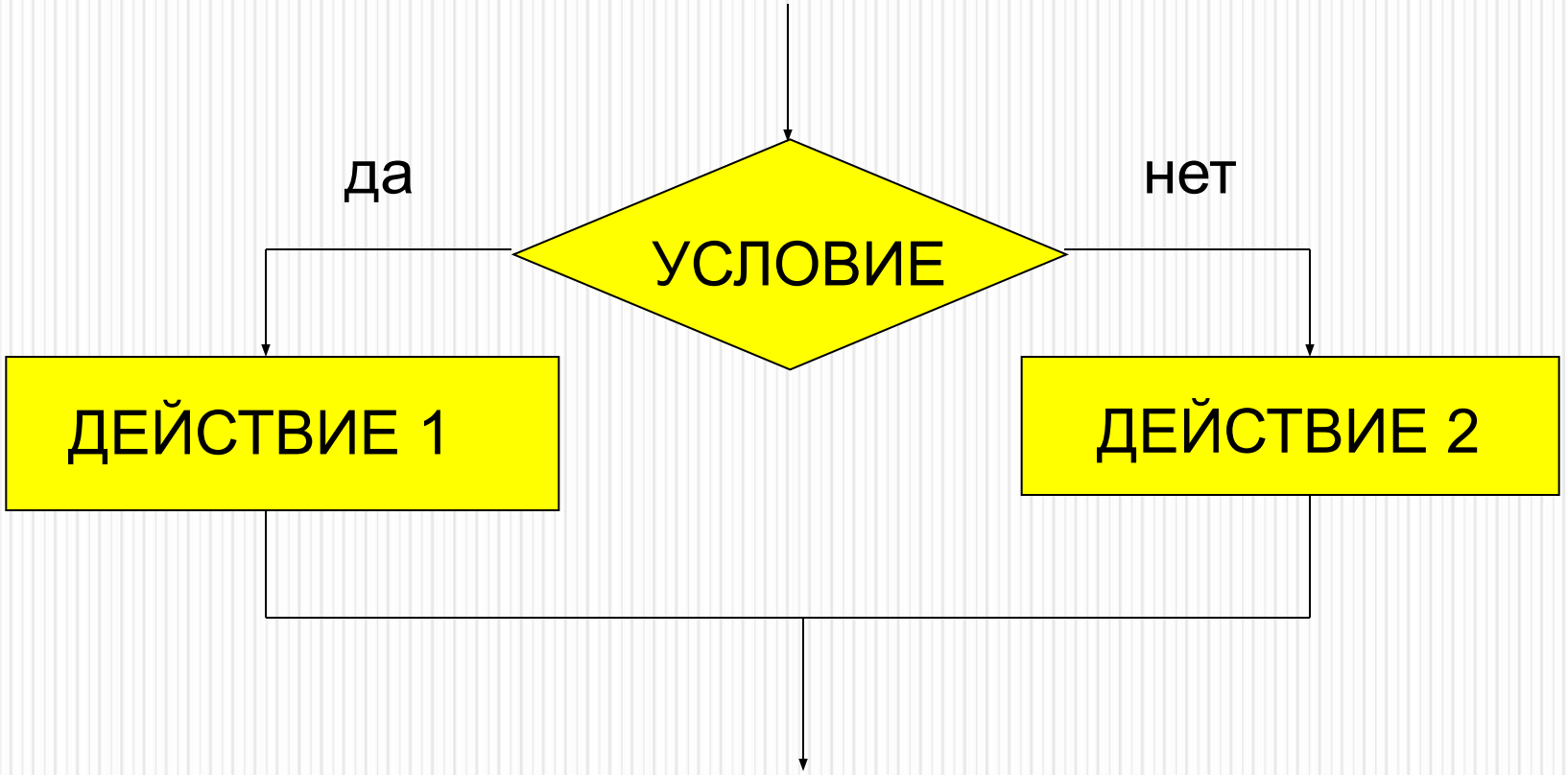
ПОЛНАЯ ФОРМА



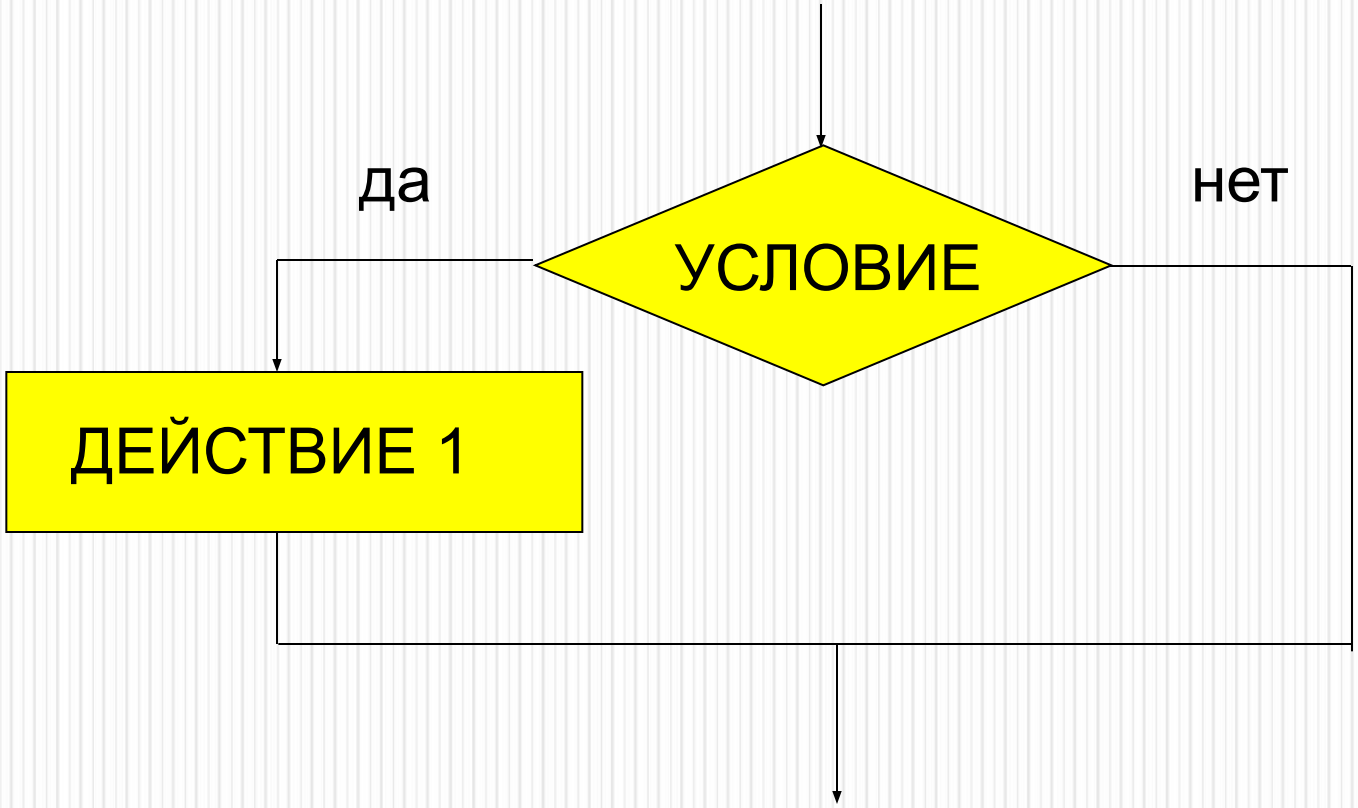
НЕПОЛНАЯ ФОРМА



Полная форма ветвления



Неполная форма ветвления



Условный оператор

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

Особенности:

- перед ***else*** **НЕ** ставится точка с запятой

- Формат полного оператора ветвления:

if <*логическое выражение*>
then <оператор 1>
else <оператор 2>;

- Формат неполного оператора ветвления:

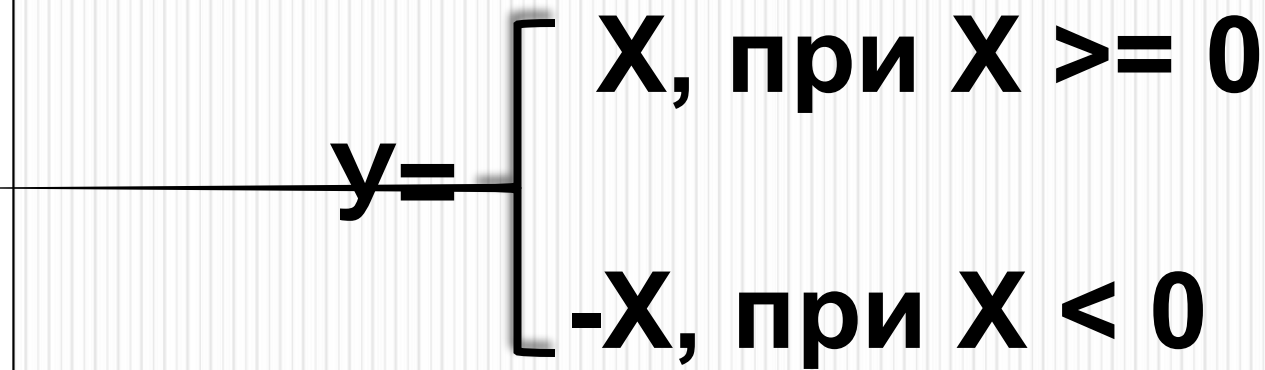
if <*логическое выражение*>
then <оператор>;

Рассмотрим простой пример задачи из курса алгебры.

Требуется построить алгоритм вычисления значения функции y=|x|. Она задается соотношением:

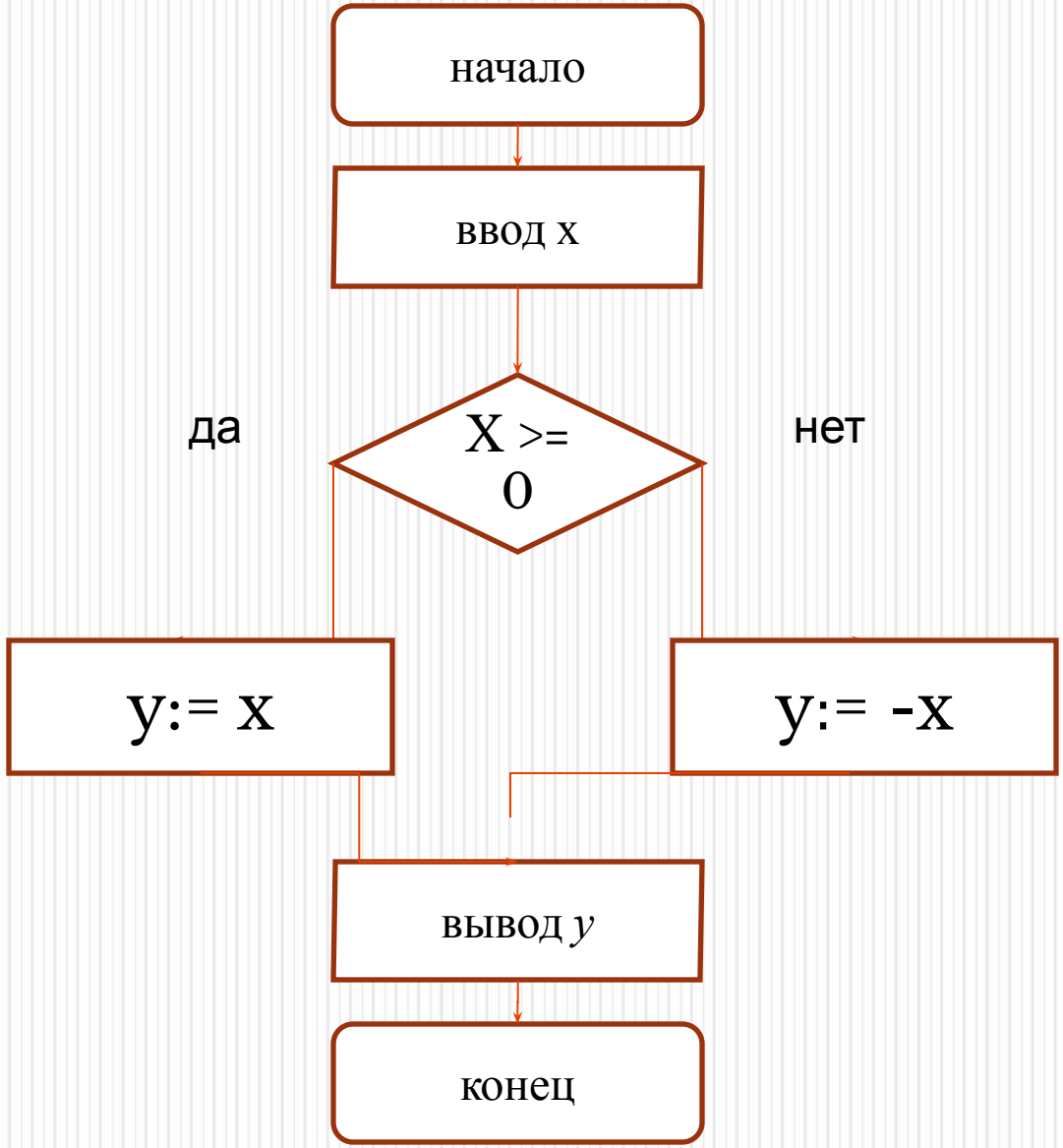
$$y = \begin{cases} X, \text{ при } X >= 0 \\ -X, \text{ при } X < 0 \end{cases}$$

При решении этой задачи требуется выполнить следующие условия:
проверить больше или равен 0 значение *x*.
Если *x* больше или равен 0, то в *y* записать значение *x*, если меньше 0, то присвоить *y* значение –*x*



Алгоритм задачи может быть записан:

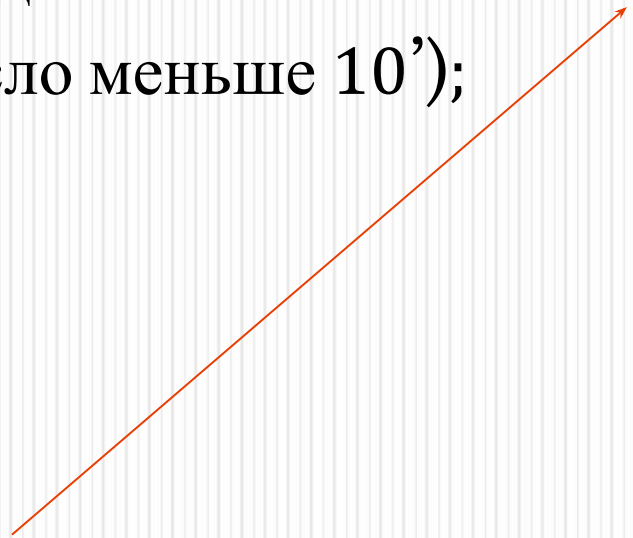
ЕСЛИ x>=0
ТО y:=x
ИНАЧЕ y:=-x;



Пример программы:

```
Program modul;
   Var x,y: integer;
Begin
Writeln('Введите число');
Readln(x);
If x>0
    then y :=x
    else y:=-x;
Writeln('y = ', y);
End.
```

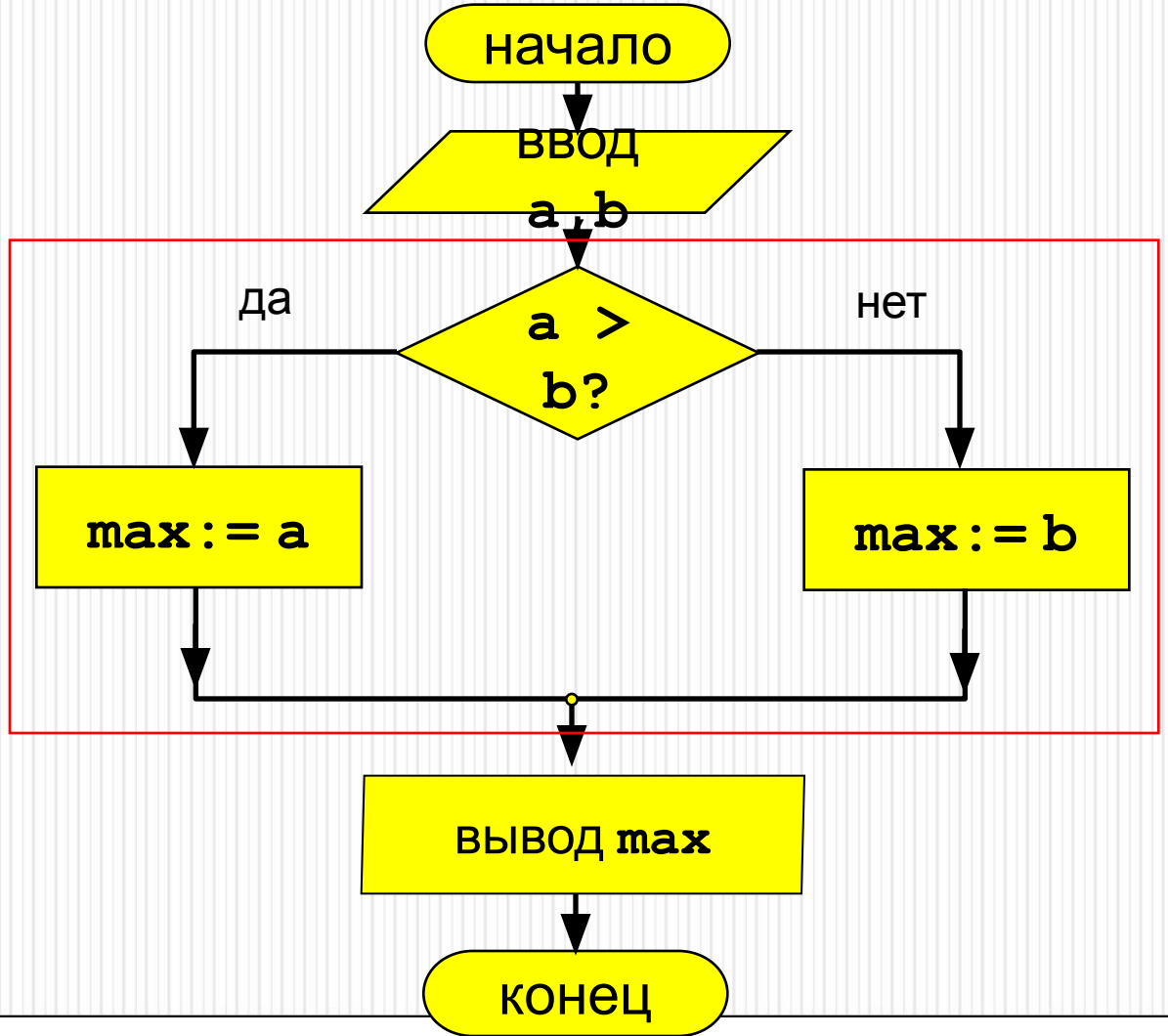
```
Program B1;
   Var a: Integer;
Begin
  readln(a);
  if a>10 then writeln('введеное число больше 10')
  else writeln('введеное число меньше 10');
end.
```



Обратите внимания, «;» не ставится

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

Блок-схема



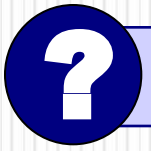
Из двух заданных целых чисел выбрать наибольшее.

```
program one;
  var x, y, max: integer;
 begin
     write ('Введите два целых числа: ');
      readln (x, y);
       if x >= y
         then max:=x
         else  max:=y;
      writeln ('наибольшее = ',max)
end.
```

Сложные условия

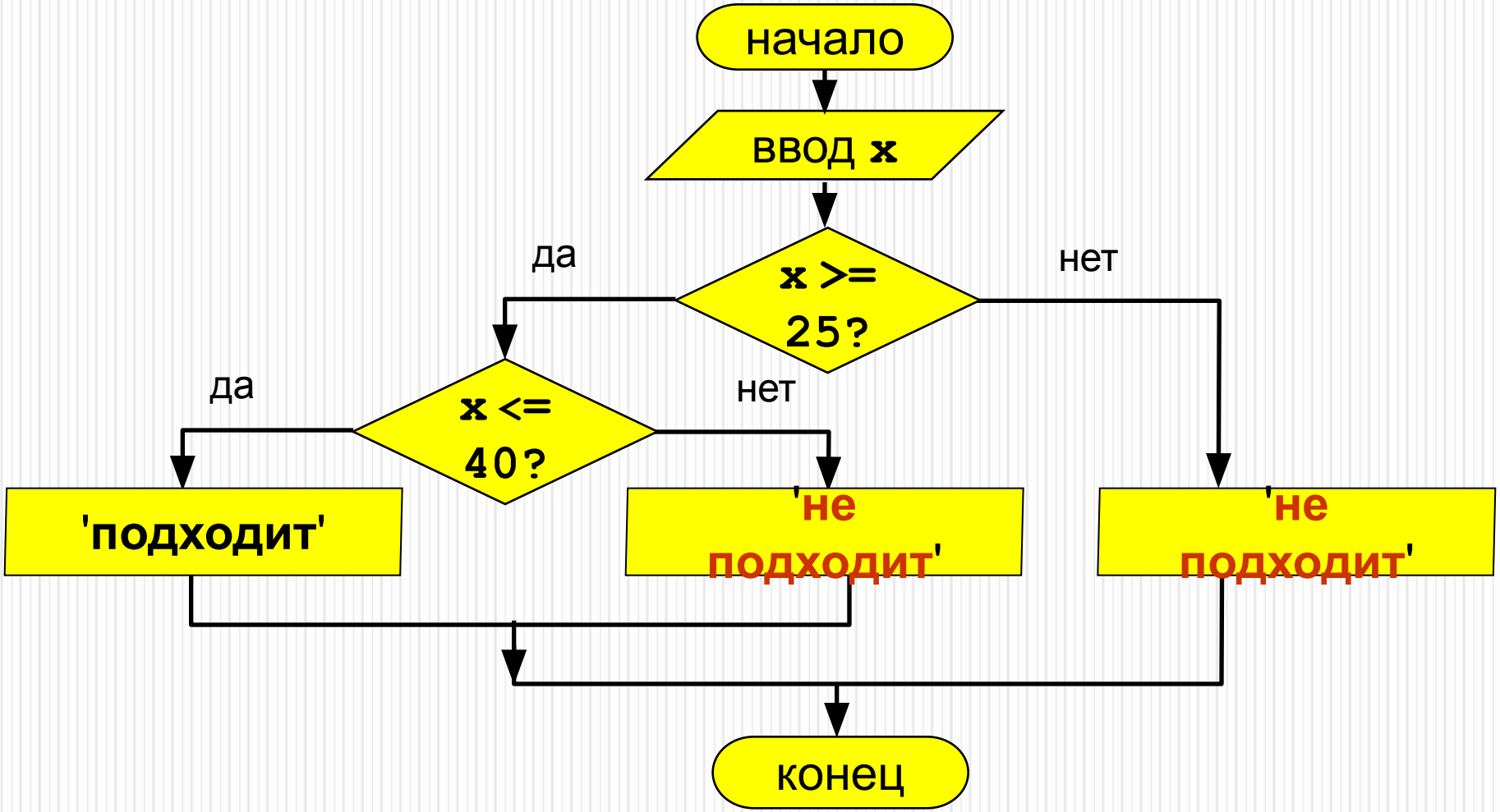
Задача. Фирма набирает сотрудников от 25 до 40 лет включительно. Ввести возраст человека и определить, подходит ли он фирме (вывести ответ «подходит» или «не подходит»).

Особенность: надо проверить, выполняются ли два условия одновременно.



Можно ли решить известными методами?

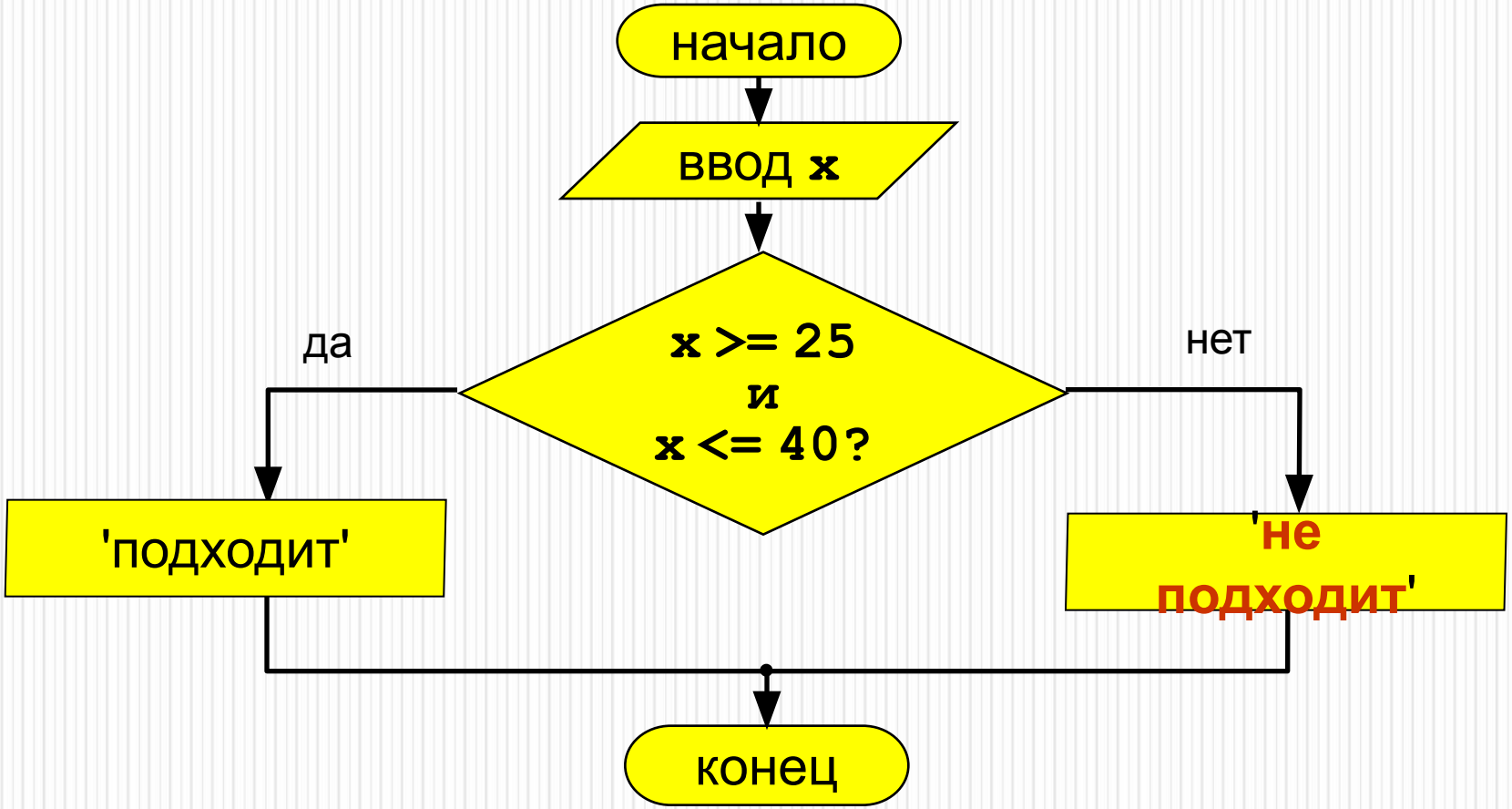
1 способ решения. Блок-схема



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
```

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.

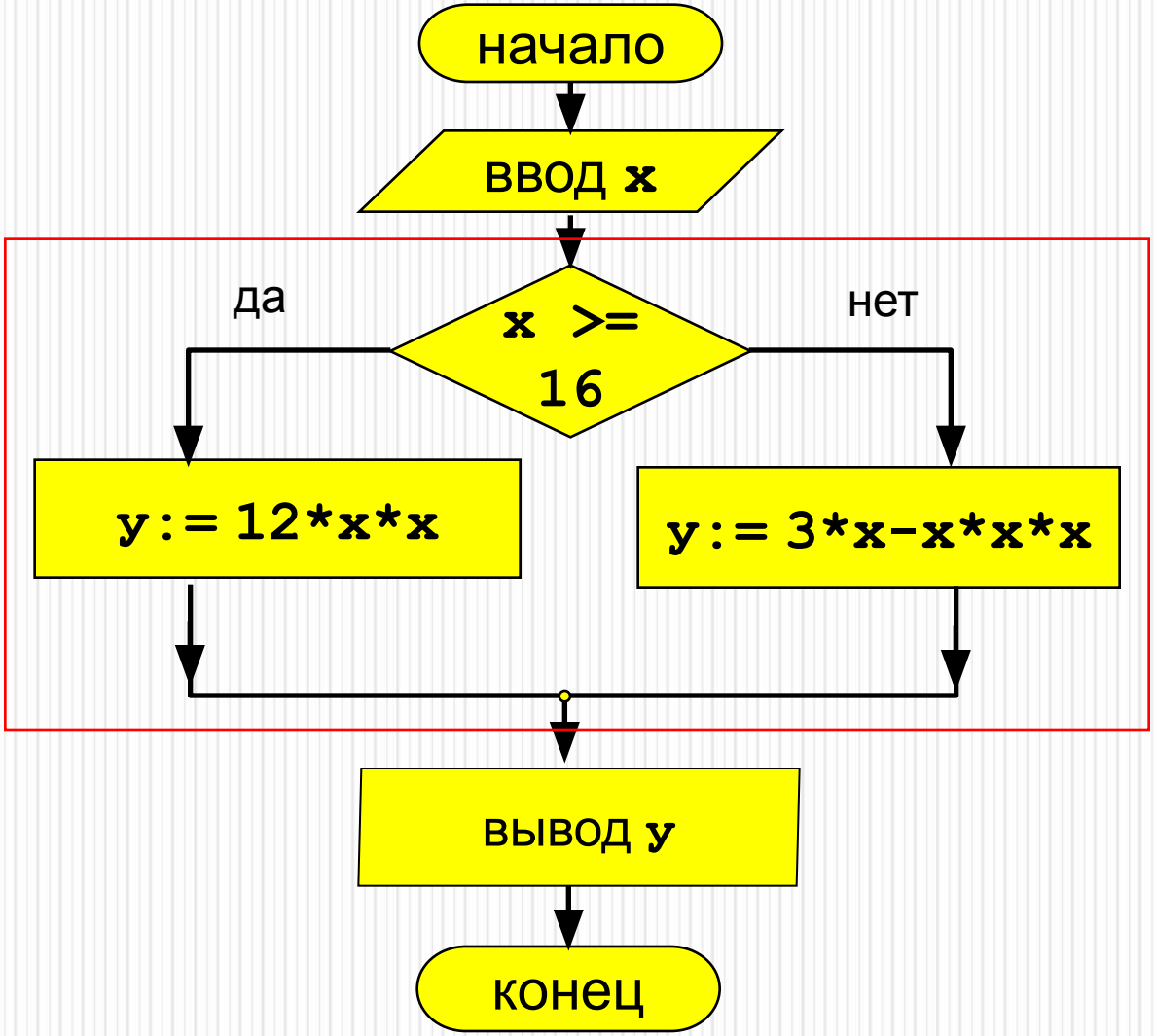
Если x<=16, то y=12*x*x.

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.
```

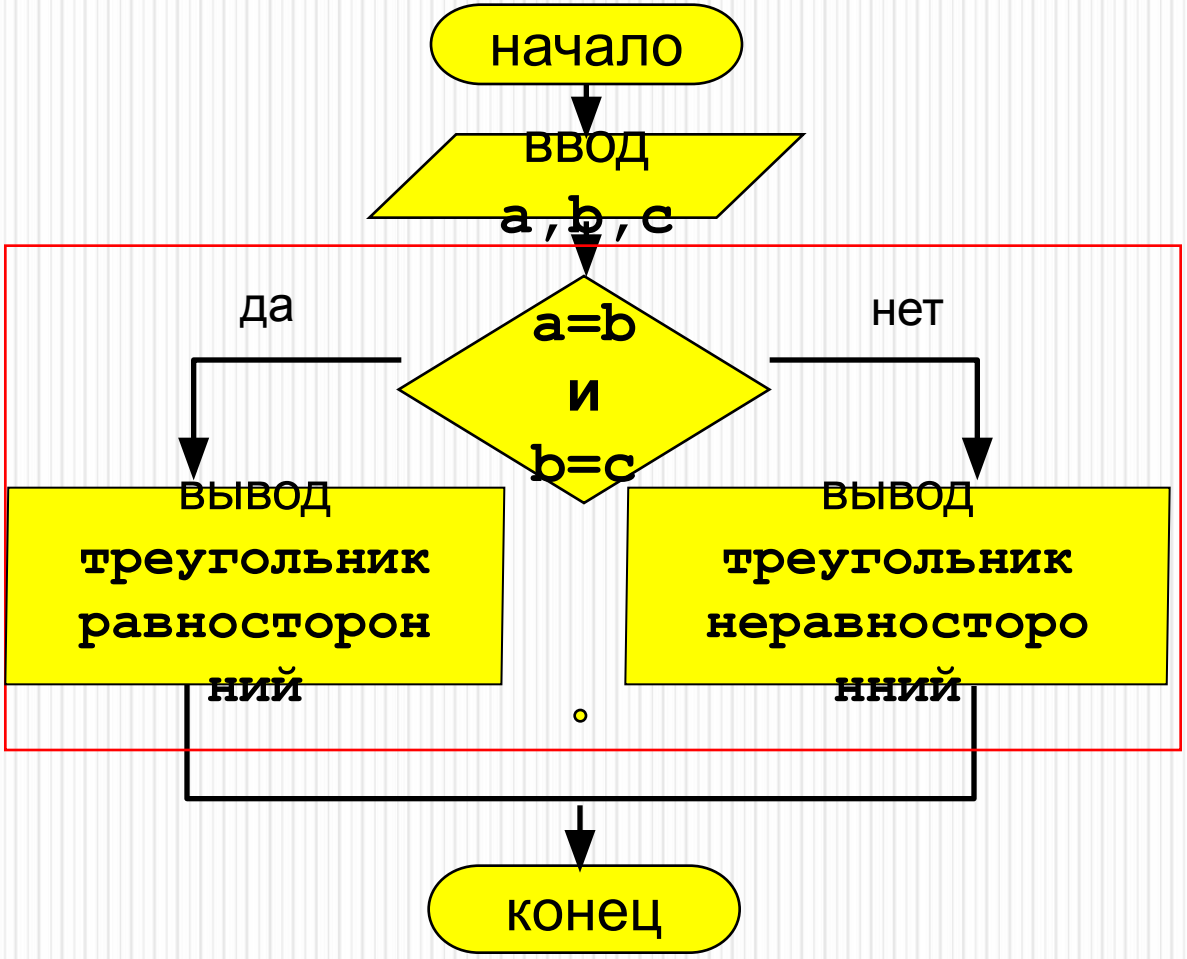
2. Определить является ли треугольник со сторонами a, b, c равносторонним треугольником.

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.
```

Пример программы:

```
Program uslov;
Var a: integer;
Begin
Writeln('введите  число');
Write('a=');
Readln(a);
If a mod 2=0 then writeln('a –четное')
else writeln('a –нечетное');
End.
```

Если в качестве блока Действие1 (Действие2) должна выполниться серия операторов, то эти операторы заключаются в операторные скобки Begin – End.

Задание на дом: оформить задачи в виде кода программы на языке Паскаль и блок-схемы

1. Дано целое число. Если оно является положительным, то прибавить к нему 1, в противном случае вычесть из него два. Вывести полученное число.
2. Даны три стороны одного треугольника и три стороны другого треугольника. Определить, будут ли эти треугольники равновеликими, т. е. имеют ли они равные площади.
3. Составьте программу вычисления функции:

$$Y = \begin{cases} -x, если & x <= 0 \\ 5 + x, если & x > 0 \end{cases}$$

4. Написать алгоритм вычисления значения z, если

$$z = \begin{cases} a^3 + \sqrt{a} - 2a, если & a \geq 0 \\ 2\sqrt{a}, если & a < 0 \end{cases}$$

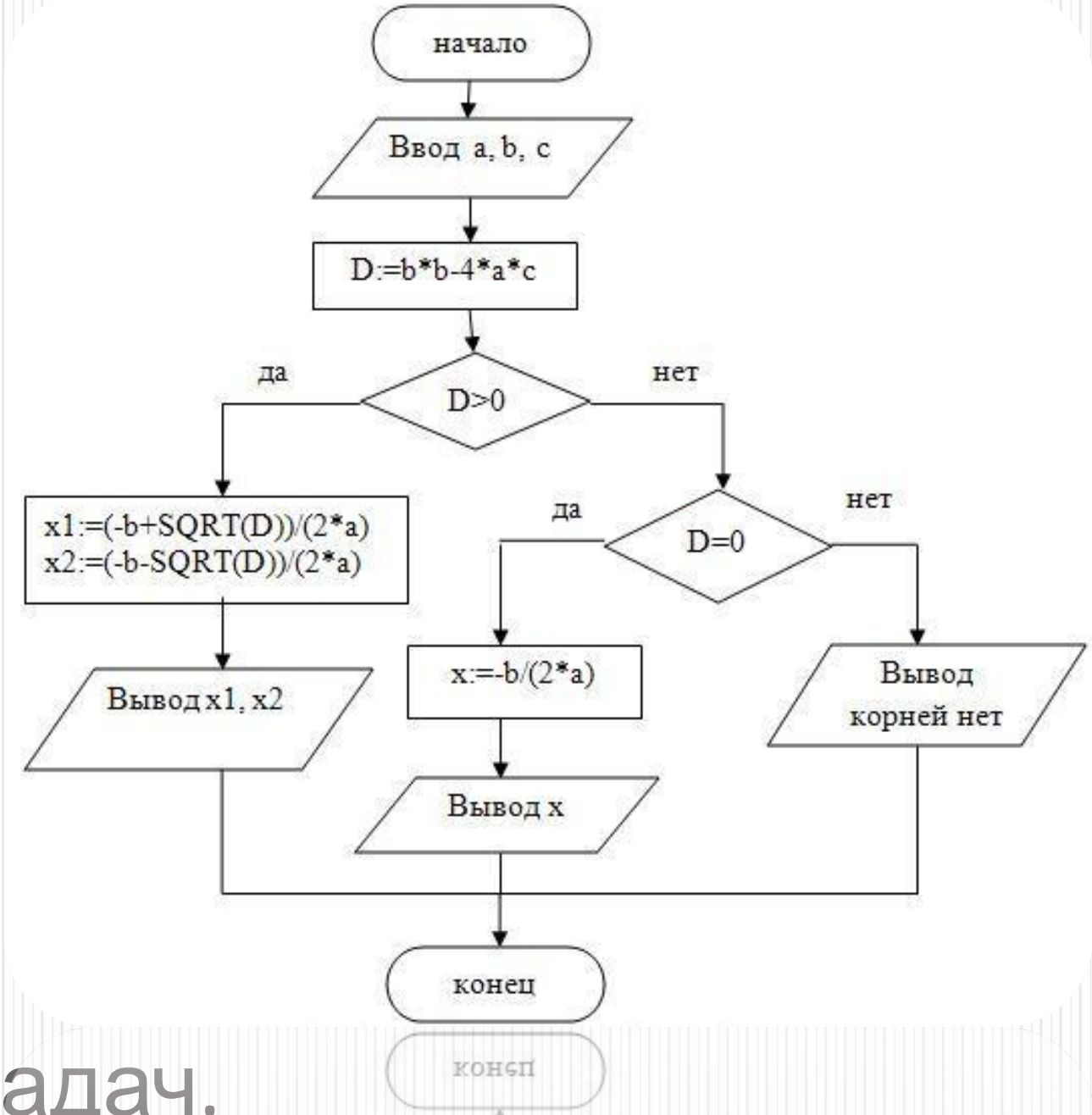
5. Определить является ли треугольник со сторонами a, b, c равнобедренным треугольником.
6. Определить является ли треугольник со сторонами a, b, c прямоугольным треугольником.

- В качестве оператора в команде ветвления может быть другой условный оператор. В этом случае получаем вложенные ветвления. Рассмотрим на примере.

- **Задача. Составить программу для решения квадратного уравнения $ax^2 + bx + c = 0$.**

Решение задач.

Решение задач.



```
PROGRAM  zadacha2;
VAR a,b,c,D,x,x1,x2:REAL;
BEGIN
        WRITE('a='); READLN(a);
        WRITE('b='); READLN(b);
        WRITE('c='); READLN(c);
        D:= b*b - 4*a*c;
        WRITE ('Корни уравнения: ');
        IF D>0 THEN
                BEGIN
                        x1:=(-b+SQRT(D))/(2*a);
                        x2:=(-b-SQRT(D))/(2*a);
                        WRITELN ('x1= ',x1:5:2,'x2= ',x2:5:2);
                END
                ELSE
                        IF D=0 THEN
                            BEGIN
                                    x:= -b/(2*a);
                                    WRITELN ('x= ',x:5:2);
                            END
                            ELSE
                                    WRITELN ('Корней нет');
END.
```

Решение задач.

Для записи в тетрадь

- ***Разветвляющимся*** называется алгоритм, в котором выбирается одна из нескольких возможных серий команд. Каждый подобный путь называется *ветвью алгоритма*.

- Признаком разветвляющегося алгоритма является наличие операций **проверки условия**.

- Слайды 36-50 –записать в тетрадь

Условие – это логическое выражение, которое может быть записано в операторе явно или вычислено в программе. Для записи *простых условий* используются операции отношения:

<	меньше	x>y
>	больше	a>5
<=	меньше или равно	n<=0
>=	больше или равно	t>=r
<>	не равно	a+b<>0
=	равно	s mod 2 = 0

В ветвлении можно проверять несколько условий одновременно. Для этого условия связываются между собой логическими операциями. Получается *сложное условие*.

Если необходимо проверить одновременное выполнение нескольких условий, для их связи используют логическую операцию **AND (И)**

Например, условие $0<x<5$, в ветвлении будет выглядеть так:

(x>0) **and** (x<5)

Если же нужно чтобы выполнялось хотя бы одно из нескольких условий, то для их связи используют операцию **OR (ИЛИ)**

Например, условия *y<0 или y>9* будет выглядеть следующим образом:

(y<0) **or** (y>9)

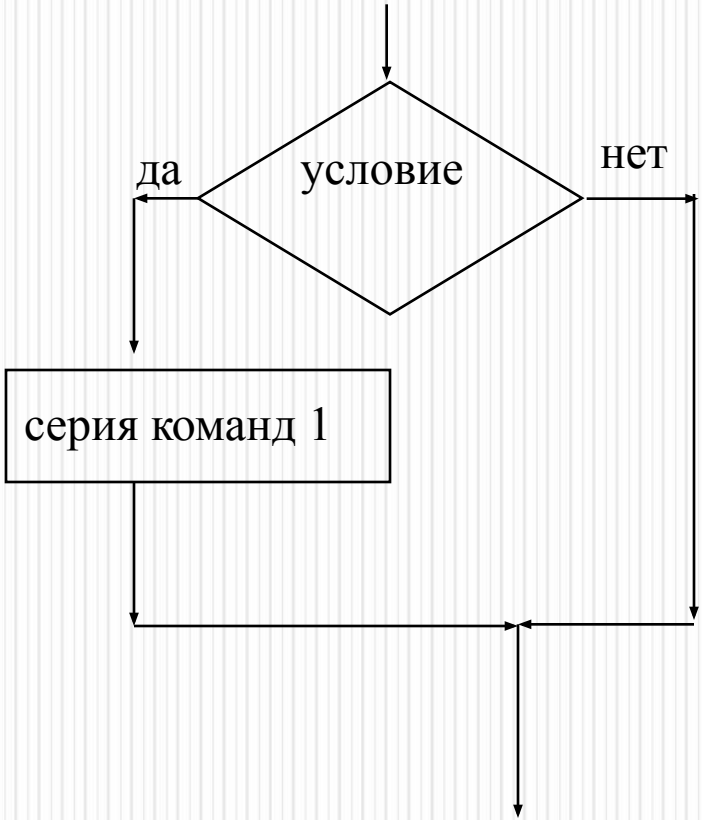
Логическая операция **NOT (НЕ**) меняет значение условия на противоположное.
Например, необходимо взять *все значения x, кроме 1*:

not (x=1)

При связывании нескольких условий логическими операциями, необходимо заключать простые условия в скобки.

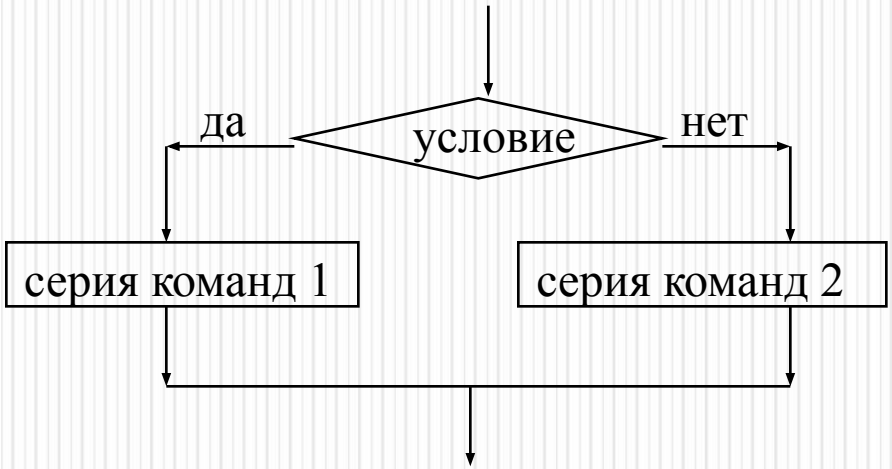
Основные варианты структуры ветвления:

если - то;



неполное ветвление

если – то - иначе;



полное ветвление

Запись команды ветвления на языке программирования Pascal.

IF (условие)

THEN (оператор 1);

IF (условие) THEN

```
BEGIN
    <оператор 1>;
    <оператор 2>;
    …
    <оператор n>;
END;
```

Если должна выполниться серия команд, то эти операторы заключаются в операторные скобки Begin – End.

Запись полного ветвления

на языке программирования Pascal

IF (условие)

THEN (оператор 1)

ELSE (оператор 2);

```
IF (условие) THEN
        BEGIN
            <оператор 1>;
            <оператор 2>;
            …
            <оператор n>;
        END
    ELSE
        BEGIN
            <оператор 1>;
            <оператор 2>;
            …
            <оператор n>;
        END;
```

ЗАДАЧА 1. *Из двух чисел A и B найти набольшее.*

```
PROGRAM  zadacha1;
  VAR A, B, max: INTEGER; {описываем переменные A , B и max целыми числами}
BEGIN
  WRITE('A='); {Вводим с клавиатуры числа A и B}
  READLN(A);
  WRITE('B=');
  READLN(B);
  {Если A>B , то наибольшее число A, иначе наибольшее число  B}
  IF A>B THEN max :=A
            ELSE max :=B;
  WRITELN ('Большее число = ', max );
END.
```

ЗАДАЧА 2. *Из двух чисел A и B найти набольшее и наименьшее.*

```
PROGRAM  zadacha2;
  VAR  A, B, max, min: INTEGER;
BEGIN
  WRITE('A='');  READLN(A);
  WRITE('B='');  READLN(B);
   IF A>B THEN begin
                        max :=A;
             min :=B;
                      end
         ELSE begin
                         max :=B;
              min :=A;
                       end;
 WRITELN ('max=', max, '  min=', min);
END.
```

Определение четности числа

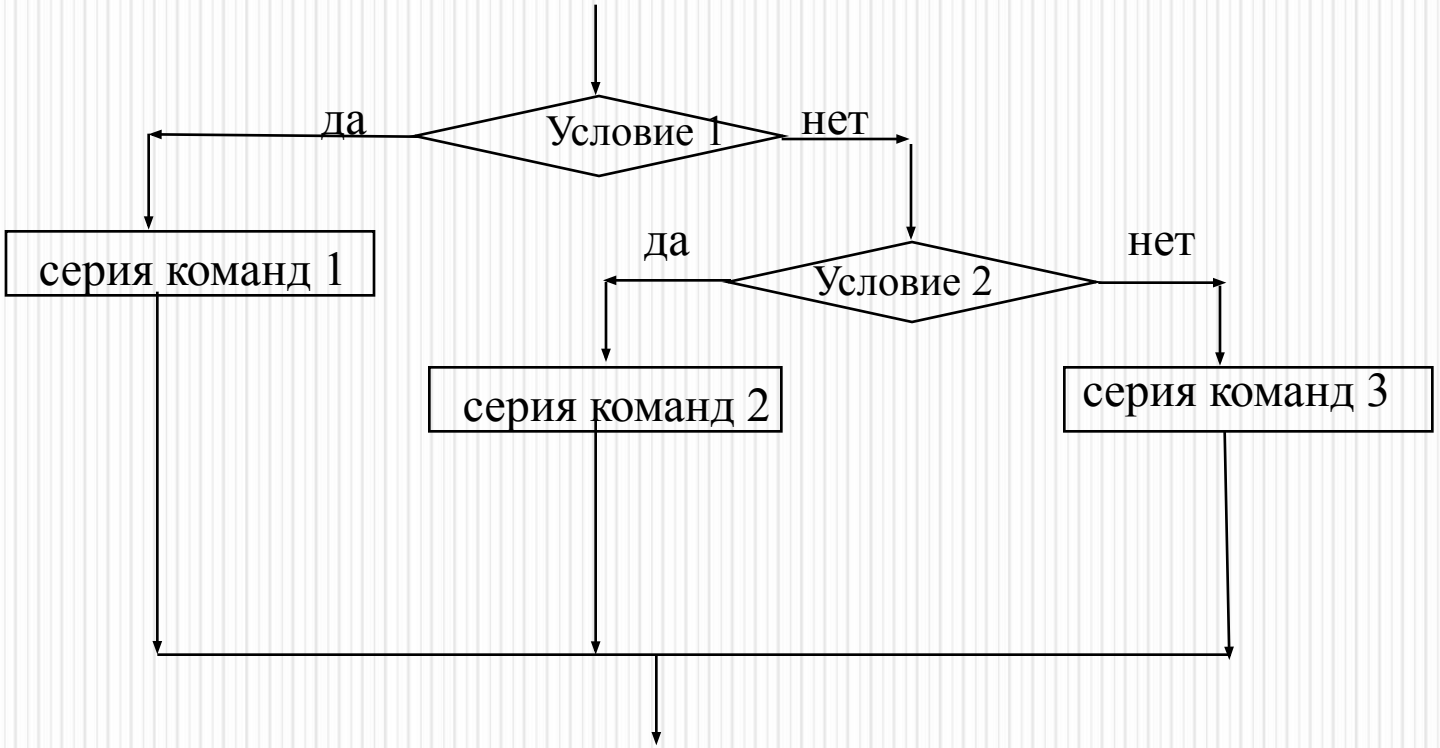
Пример программы:

```
Program uslov;
Var a: integer;
Begin
Writeln('введите  число');
Write('a=');
Readln(a);
If a mod 2=0 then writeln('a –четное')
else writeln('a –нечетное');
End.
```

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»).

```
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.
```

Вложенное ветвление:



IF (условие)

THEN (оператор 1)

ELSE IF (условие) THEN (оператор 1)

ELSE (оператор 2);

Задача 4. Найти наибольшее из трёх данных чисел a, b, c.

```
Program zadacha3;
 Var   a, b, c, max: Integer;
Begin
  writeln('введи числа');
  readln(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('большее число равно ', max);
End.
```