



# ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ

## НАЧАЛА ПРОГРАММИРОВАНИЯ

**8 класс**



ИЗДАТЕЛЬСТВО

**БИНОМ**

# Ключевые слова

- **условный оператор**
- **неполный условный оператор**
- **составной оператор**
- **вложенные ветвления**



# Общий вид условного оператора

Полная форма условного оператора:

**if** <условие> **then** <оператор\_1> **else** <оператор\_2>

Неполная форма условного оператора:

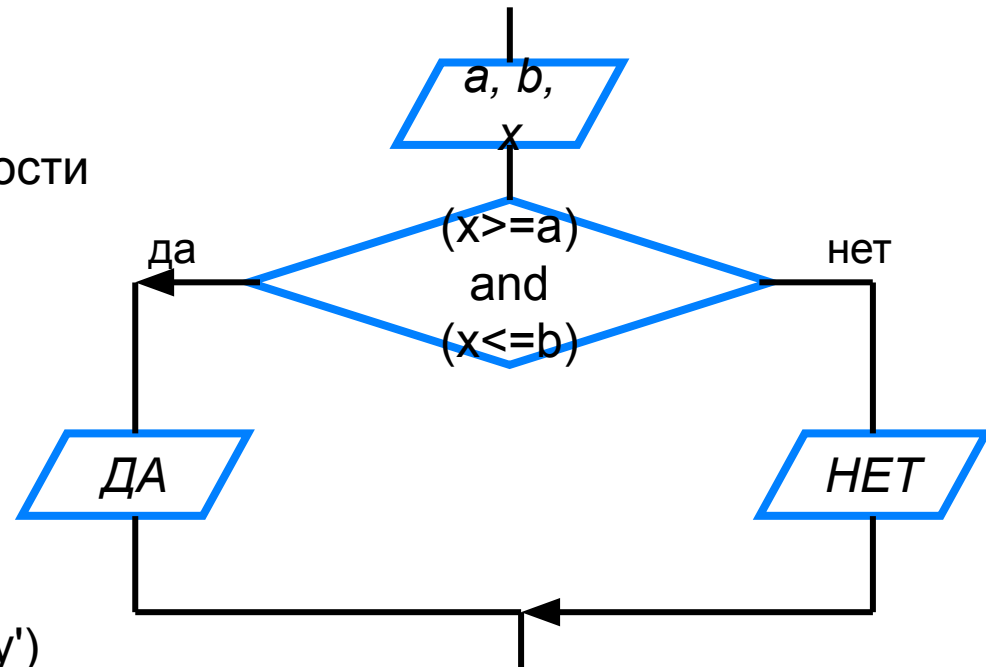
**if** <условие> **then** <оператор>

**!** Перед **else** знак «;» не ставится.



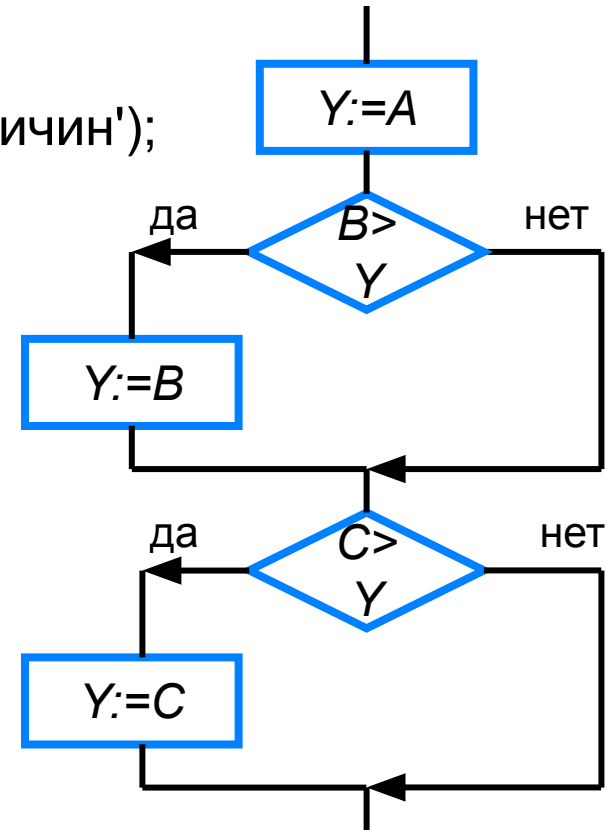
# Условный оператор

```
program n_9;  
  var x, a, b: real;  
begin  
  writeln ('Определение принадлежности  
           точки отрезку');  
  write ('Введите a, b>>');  
  readln (a, b);  
  write ('Введите x>>');  
  readln (x);  
  if (x>=a) and (x<=b) then  
    writeln ('Точка принадлежит отрезку')  
  else writeln ('Точка не принадлежит отрезку')  
end.
```



# Неполный условный оператор

```
program n_10;  
  var y, a, b, c: integer;  
begin  
  writeln ('Нахождение наибольшей из трёх величин');  
  write ('Введите a, b, c>>');  
  readln (a, b, c);  
  y:=a;  
  if (b>y) then y:=b;  
  if (c>y) then y:=c;  
  writeln ('y=', y)  
end.
```

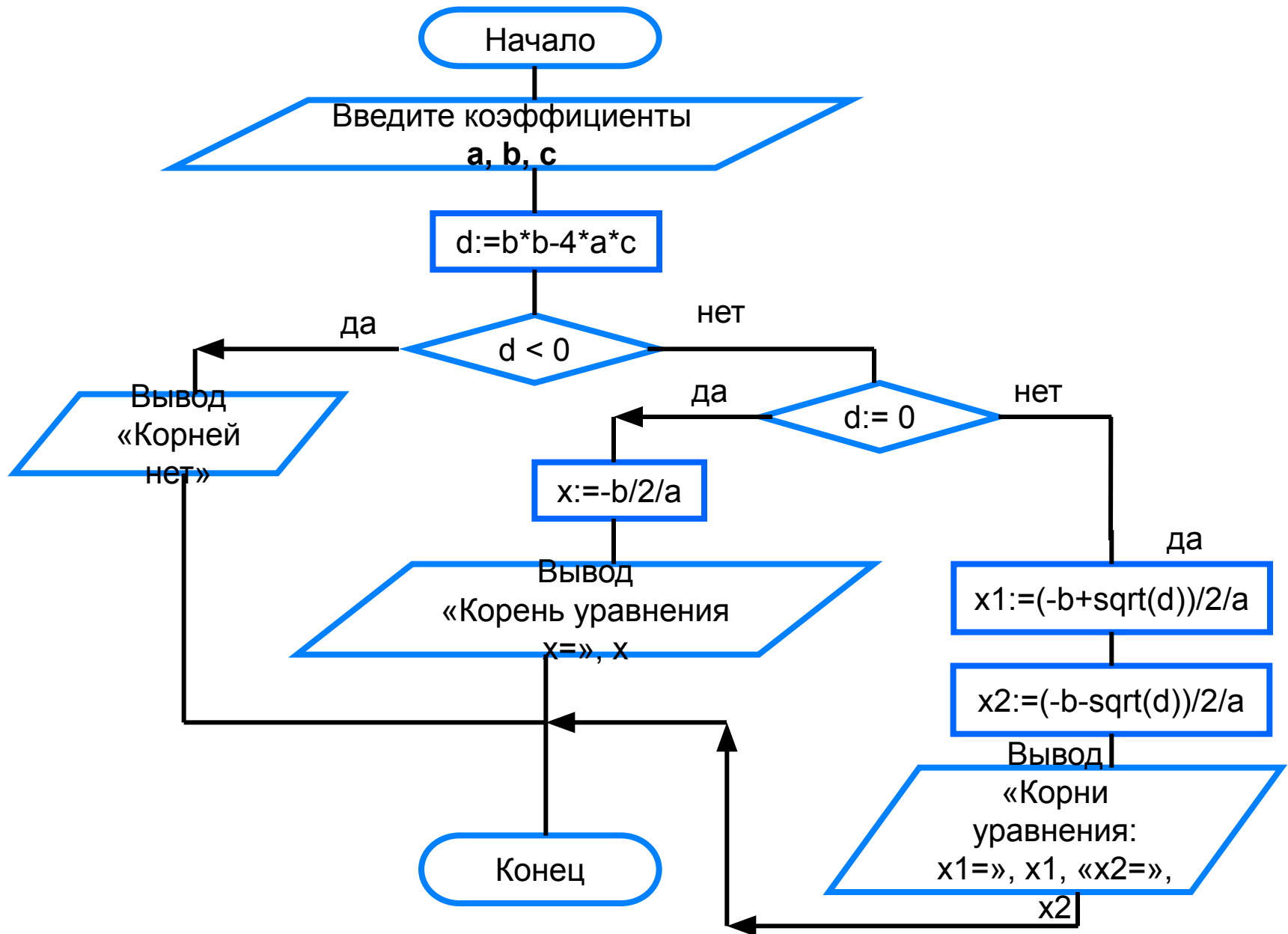


# Составной оператор

В условном операторе и после **then**, и после **else** можно использовать **только один оператор**.

Если в условном операторе после **then** или после **else** нужно выполнить **несколько операторов**, то используют **составной оператор** – конструкцию вида:  
**begin** <последовательность операторов> **end**

# Блок-схема решения КВУР



```
program n_11;  
  var a, b, c: real;  
  var d: real;  
  var x, x1, x2: real;
```

```
begin
```

```
  writeln ('Решение квадратного уравнения');
```

```
  write ('Введите коэффициенты a, b, c >>');
```

```
  readln (a, b, c);
```

```
  d:=b*b-4*a*c;
```

```
  if d<0 then writeln ('Корней нет')
```

```
    else
```

```
      if d=0 then
```

```
        begin
```

```
          x:=-b/2/a;
```

```
          writeln ('Корень уравнения x=', x:9:3)
```

```
        end
```

```
      else
```

```
        begin
```

```
          x1:=(-b+sqrt(d))/2/a;
```

```
          x2:=(-b-sqrt(d))/2/a;
```

```
          writeln ('Корни уравнения:');
```

```
          writeln ('x1=', x1:9:3);
```

```
          writeln ('x2=', x2:9:3)
```

```
        end
```

```
end.
```



# Вложенные ветвления

```
if <условие1> then
```

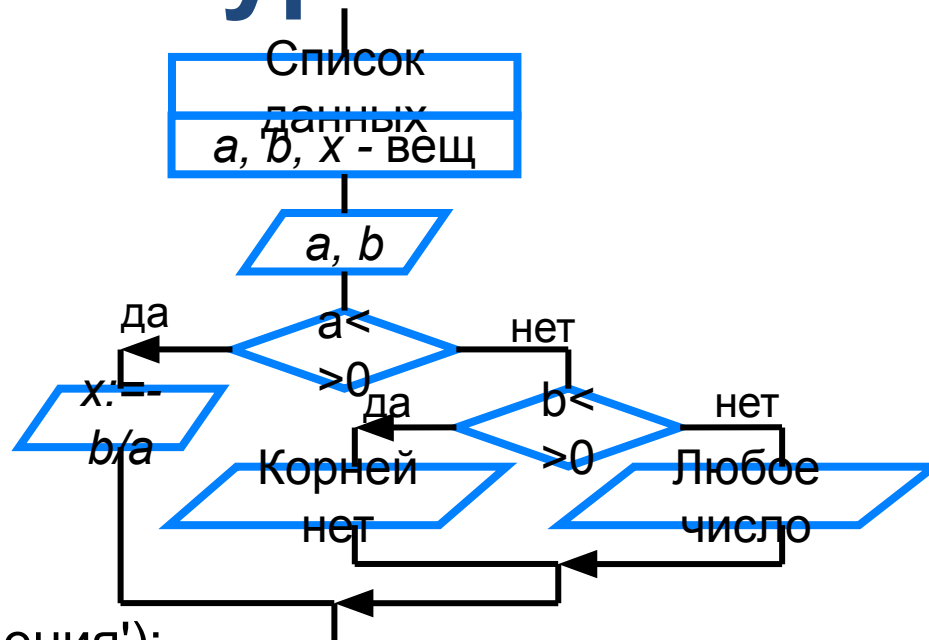
```
    if <условие2> then <оператор1>
```

```
        else <оператор2>
```

```
    else <оператор3>
```

**!** **else** всегда относится к ближайшему оператору **if**

# Решение линейного уравнения



```
program n_12;  
  var a, b, x: real;  
begin
```

```
  writeln ('Решение линейного уравнения');  
  write ('Введите коэффициенты a , b>>');  
  readln (a, b);
```

```
  if a <> 0 then  
    begin  
      x := -b/a;  
      writeln ('Корень уравнения x=', x:9:3)  
    end  
  else if b <> 0 then writeln ('Корней нет')  
    else writeln ('x – любое число');
```

```
end.
```

# Самое главное

При записи на языке Паскаль разветвляющихся алгоритмов используют условный оператор:

***if*** <условие> ***then*** <оператор\_1> ***else*** <оператор\_2>

Для записи неполных ветвлений используется неполный условный оператор:

***if*** <условие> ***then*** <оператор>

Если при некотором условии требуется выполнить определённую последовательность операторов, то их объединяют в один составной оператор, имеющий вид:

***begin*** <последовательность операторов> ***end***



# Вопросы и задания

1. Проверить на наличие ошибок в программе, которая проверяет, является ли введенное натуральное число простым. Программа производит обработку введенных данных, выводя на экран результат. Если число простое, то выводится сообщение "Число простое", в противном случае - "Число не простое". Программа должна работать корректно для всех возможных входных данных. Проверить работу программы на тестовых данных.

Пример входных данных: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50. Пример выходных данных: Число простое, Число не простое.

2. Проверить работу программы, которая проверяет, является ли введенное натуральное число простым. Программа производит обработку введенных данных, выводя на экран результат. Если число простое, то выводится сообщение "Число простое", в противном случае - "Число не простое". Программа должна работать корректно для всех возможных входных данных. Проверить работу программы на тестовых данных.

Пример входных данных: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50. Пример выходных данных: Число простое, Число не простое.

3. Проверить работу программы, которая проверяет, является ли введенное натуральное число простым. Программа производит обработку введенных данных, выводя на экран результат. Если число простое, то выводится сообщение "Число простое", в противном случае - "Число не простое". Программа должна работать корректно для всех возможных входных данных. Проверить работу программы на тестовых данных.

Пример входных данных: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50. Пример выходных данных: Число простое, Число не простое.

4. Проверить работу программы, которая проверяет, является ли введенное натуральное число простым. Программа производит обработку введенных данных, выводя на экран результат. Если число простое, то выводится сообщение "Число простое", в противном случае - "Число не простое". Программа должна работать корректно для всех возможных входных данных. Проверить работу программы на тестовых данных.

Пример входных данных: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50. Пример выходных данных: Число простое, Число не простое.

5. Проверить работу программы, которая проверяет, является ли введенное натуральное число простым. Программа производит обработку введенных данных, выводя на экран результат. Если число простое, то выводится сообщение "Число простое", в противном случае - "Число не простое". Программа должна работать корректно для всех возможных входных данных. Проверить работу программы на тестовых данных.

Пример входных данных: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50. Пример выходных данных: Число простое, Число не простое.

# Опорный конспект

Условный оператор

Полная форма

*if* <условие> *then* <оператор\_1> *else* <оператор\_2>

Неполная форма

*if* <условие> *then* <оператор>

Составной оператор

*begin* <последовательность операторов> *end*