

# Алгоритмы и структуры данных.

## Часть 1. Основы алгоритмизации

## ***Список литературы к теме***

1. Вирт Н. Алгоритмы и структуры данных. СПб.: Невский диалект, 2001.
2. Кормен Т., Лейзерсон Ч. Ривест Р., Штайн К. Алгоритмы. Построение и анализ. Вильямс, 2019.
3. Анисимов А. Е. Введение в алгоритмизацию задач: учебно-методическое пособие. Ижевск: Изд-во «Удмуртский университет», 2017.
4. Бхаргава А. Грокаем алгоритмы. СПб.: Питер, 2019.

*В конечном счёте программы  
представляют собой конкретные,  
основанные на некотором реальном  
представлении и строении данных  
воплощения абстрактных алгоритмов.*

*Н. Вирт*

**АЛГОРИТМЫ  
+ СТРУКТУРЫ ДАННЫХ  
= ПРОГРАММЫ**

## План темы "Часть 1. Основы алгоритмизации"

1. Определение алгоритма
2. Свойства алгоритма
3. Формы представления алгоритма
4. Три вида алгоритма
5. Линейные алгоритмы
6. Разветвляющиеся алгоритмы
7. Циклические алгоритмы
8. Несколько примеров алгоритмов

# 1. Определение алгоритма

- **Алгоритмом** называется точное и понятное исполнителю предписание (инструкция), описывающее порядок действий для достижения указанной цели или решения поставленной задачи.

Абу Абдуллах Мухаммад ибн Муса **аль-Хорезми** (783 – 850) - персидский учёный, математик, географ, историк. Автор трактатов "о сложении и вычитании" (алгоритмы), о "восполнении и противопоставлении" (алгебра) и др.



***Исполнитель алгоритма*** - тот, кто способен понимать и исполнять действия, предписываемые ему алгоритмом.

**Примеры:**

- дрессированная собака
- автоматы по продаже напитков
- лифт в доме
- солдат в строю
- процессор компьютера



## Свойства исполнителя:

- **не принимает** самостоятельных решений (не обладает волей)
- обладает **системой команд**

## Пример

Исполнитель	Система команд
Служебная собака	«Сидеть», «Лежать», «Вперед», «Фас», ...
Солдат	«Стройся», «Смирно», «Шагом марш», «Налево», «Разойдись», ...
Процессор ЭВМ	«Сложить содержимое регистра АХ с содержимым регистра ВХ, результат записать в АХ», «Записать в память по адресу 1000 значение 0», ...
Лифт в доме	«Открыть дверь», «Закрыть дверь», «Двигаться вверх», «Двигаться вниз», «Остановиться»



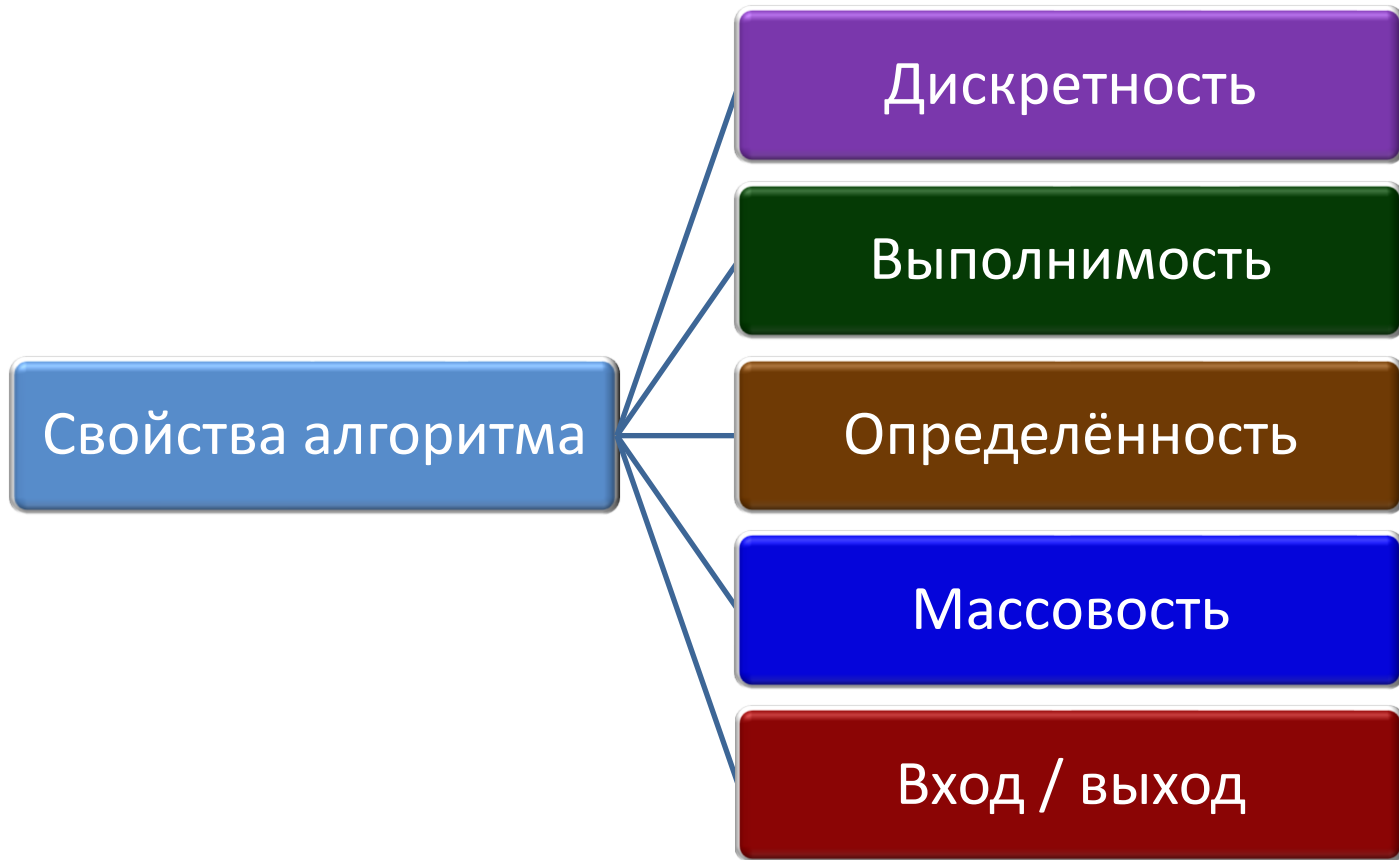
## Примеры алгоритмов:

- рецепт приготовления борща
- инструкция по эксплуатации холодильника
- решение квадратного уравнения  $ax^2+bx+c=0$
- нахождение максимума функции
- сортировка массива чисел

## Контрольные вопросы к п. 1:

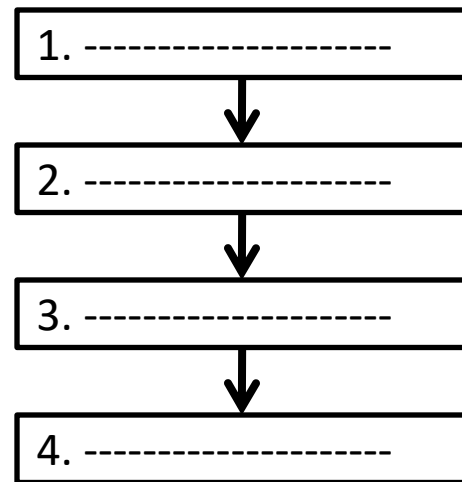
1. Что такое алгоритм?
2. Кто или что такое исполнитель алгоритма?
3. Как соотносятся понятия «алгоритм» и «исполнение алгоритма»?
4. Почему у каждого исполнителя своя система команд?
5. Чем отличается формальный исполнитель от неформального?
6. Приведите примеры различных алгоритмов.

## 2.Свойства алгоритма



## Дискретность

Алгоритм состоит из конечной последовательности отдельных шагов-команд, которые можно пронумеровать



- Исполняются команды последовательно, если иное не указано в самом алгоритме.
- Исполнение очередного шага-команды начинается только после завершения исполнения предыдущего шага.

## Пример.

### Алгоритм ЗавариваниеЧая

#### Начало алгоритма

1. Ополоснуть заварочный чайник кипятком.
2. Засыпать в чайник заварку из расчета одна чайная ложка на одну чашку чая.
3. Залить в чайник кипятком на  $2/3$  объема.
4. Подождать 5 минут.
5. Заварка готова.

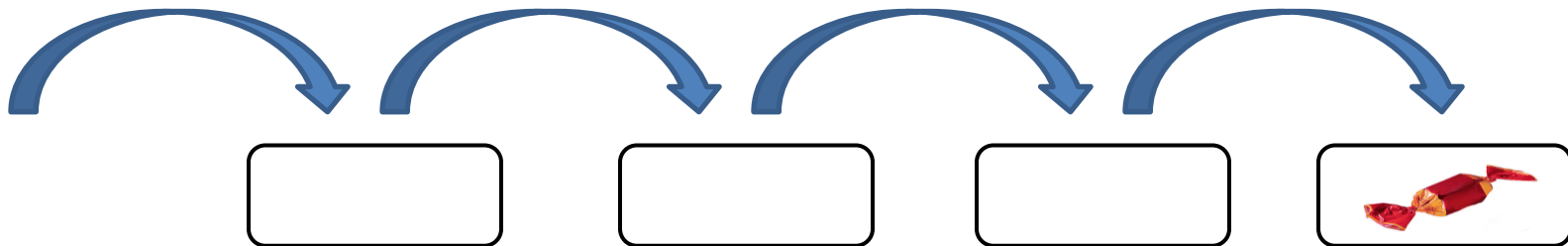
#### Конец алгоритма



## Выполнимость

Исполнение алгоритма  
исполнителем  
заканчивается за конечное  
число действий.

Не может быть выполнен  
алгоритм (то есть успешно  
получен результат) если  
количество шагов  
бесконечно!



## Пример

Алгоритм Песня(Невыполнимый Алгоритм)

Начало алгоритма

1. Жила-была бабка у самой речки
2. Захотелось бабке искупаться в речке
3. Она купила мыло мочало
4. Эта песня хороша, начинай сначала

Конец алгоритма



# Определённость

При исполнении алгоритма не должно возникать ситуаций неопределенности, когда исполнитель «не знает», что делать дальше или не может правильно «понять» очередной шаг

- То есть алгоритм рассчитан на чисто механическое исполнение, так как исполнитель не имеет права на самостоятельное принятие решений.
- Все действия исполнителя полностью определяются в самом алгоритме.



# Массовость

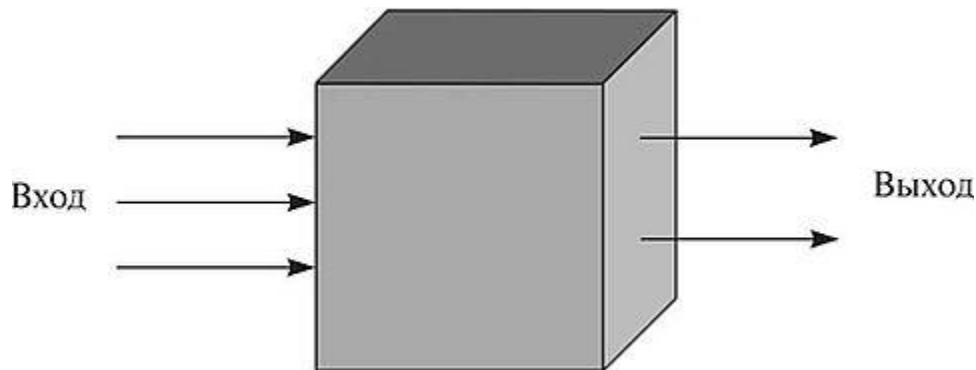
Алгоритм можно  
использовать для решения  
любой задачи из  
некоторого класса  
однотипных задач

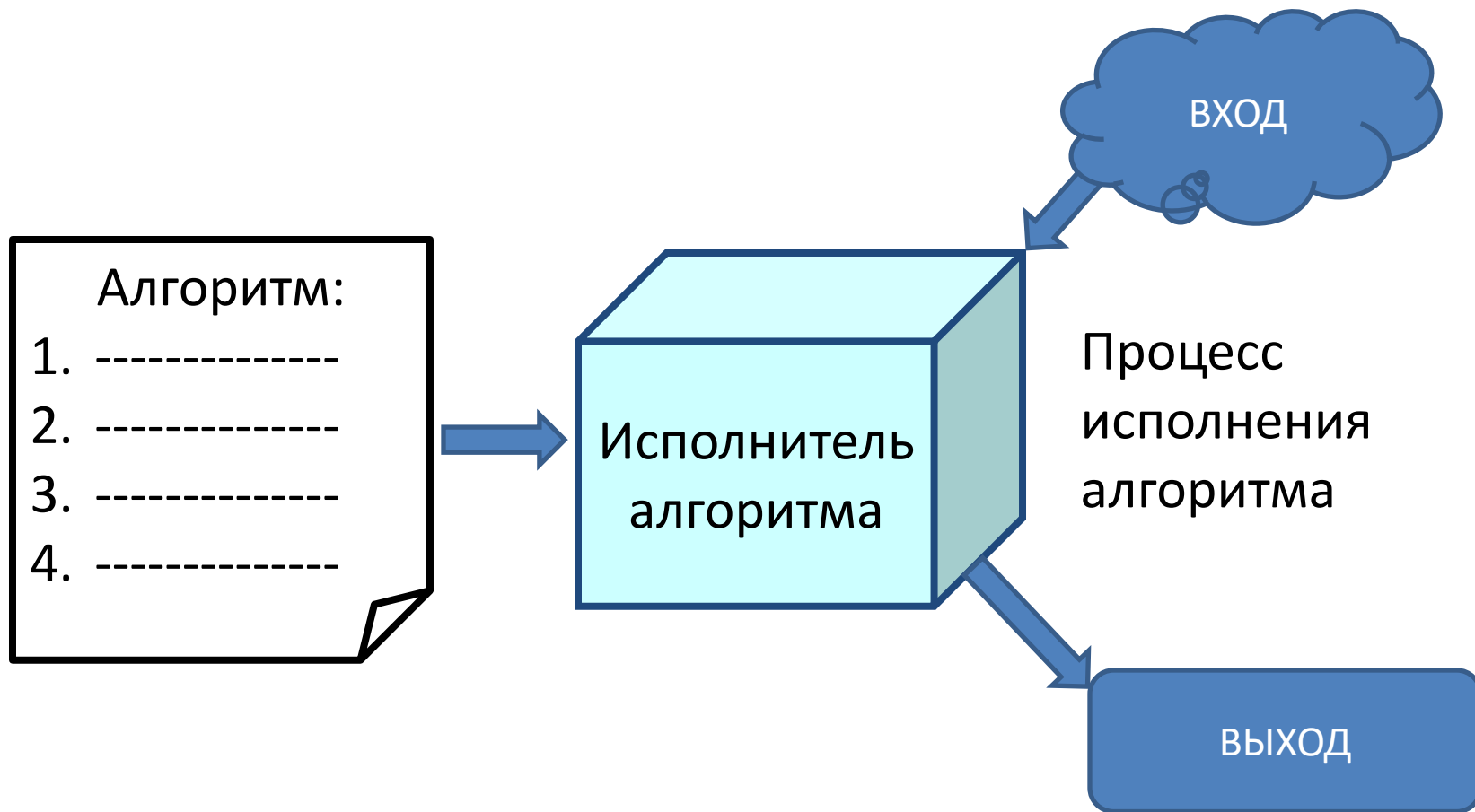


## Вход / выход

***Входные данные*** – это данные, задаваемые исполнителю для выполнения алгоритма.

***Выходные данные*** – полученный результат.







Значения входных данных должны принадлежать заранее определенному **множеству допустимых входных значений**. Это гарантирует, что выполнение алгоритма исполнителем **приведет к успешному решению** задачи.

В противном случае алгоритм может и не быть успешно исполненным.

## Примеры входных и выходных данных.

### 1. Решение квадратного уравнения $ax^2+bx+c=0$ .

*Вход:* действительные числа  $a, b, c$  ( $a \neq 0$ ).

*Выход:* множество решений уравнения

$(\{\}, \{x_1\}, \{x_1, x_2\})$

*В данном случае множество допустимых входных значений определяется так:*  $a \in (-\infty; 0) \cup (0; +\infty); b, c \in (-\infty; +\infty)$

### 2. Нахождение НОД ( $a, b$ )

*Вход:* натуральные числа  $a, b$ .

*Выход:* натуральное число, делящее  $a$  и  $b$  без остатка и являющееся наибольшим среди всех таких чисел

*Множество допустимых входных значений:*  $a, b \in \mathbb{N}$

- **Тестом** называется набор значений входных данных алгоритма, для которого заранее известен набор выходных значений.
- **Тестированием** алгоритма будем называть формальное исполнение алгоритма на заранее подготовленной системе тестов с целью обнаружения ошибок, допущенных при составлении алгоритма.

## Примеры тестов.

### 1. Тесты для решения квадратного уравнения

№ теста	Вход			Выход
	a	b	c	
1.	1	2	-3	$\{-3; 1\}$
2.	1	-2	1	$\{1\}$
3.	1	2	5	$\emptyset$

### 2. Тесты для нахождения НОД

№ теста	Вход		Выход
	a	b	
1.	35	21	7
2.	45	60	15
3.	37	38	1



## Контрольные вопросы к п. 2:

1. Какие свойства алгоритмов вы знаете?
2. Что такое «вход» и «выход» алгоритма?
3. Чем может закончиться исполнение алгоритма, если на вход были заданы данные, не входящие в множество допустимых значений?
4. Что такое тест? Что такое тестирование? Какова цель тестирования?
5. Приведите пример алгоритма и обоснуйте все его свойства.

## 3.Формы представления алгоритма

# Существует различные формы представления алгоритмов

1. Словесная форма	На естественном языке, например, русском или английском.	+понятность, простота -неоднозначность
2. Графическая форма	При описании структуры алгоритма используются графические обозначения, блок-схемы	+легкость восприятия человеком -трудность применения на практике для больших алгоритмов
3. Язык программирования	Формальный язык для составления компьютерных программ	+строгая формализация +однозначность -требование специальных знаний
и др.		

## Пример.

Рассмотрим на примере следующего алгоритма три формы представления.

**Алгоритм Евклида. Даны два натуральных числа  $a$ ,  $b$ . Найти наибольший общий делитель этих чисел:  $\text{НОД}(a, b)$  .**

Таблица тестов алгоритма Евклида

№ теста	Вход		Выход
	a	b	
1.	21	35	7
2.	17	16	1
3.	100	100	100

# 1. Словесная форма

**Алгоритм** Евклида

**Вход:**  $a, b \in \mathbb{N}$ , где  $\mathbb{N}$  – множество натуральных чисел;

**Выход:** НОД( $a, b$ )

**Начало алгоритма**

1. Если  $a \neq b$ , то перейти к шагу 2, иначе – перейти к шагу 5.
2. Найти большее из  $a$  и  $b$ .
3. Заменить большее на разность большего и меньшего.
4. Перейти к шагу 1.
5. Выход:  $a$ .

**Конец алгоритма.**

Рассмотрим пошагово процесс исполнения алгоритма Евклида для теста № 1 ( $a = 21$ ,  $b = 35$ ).

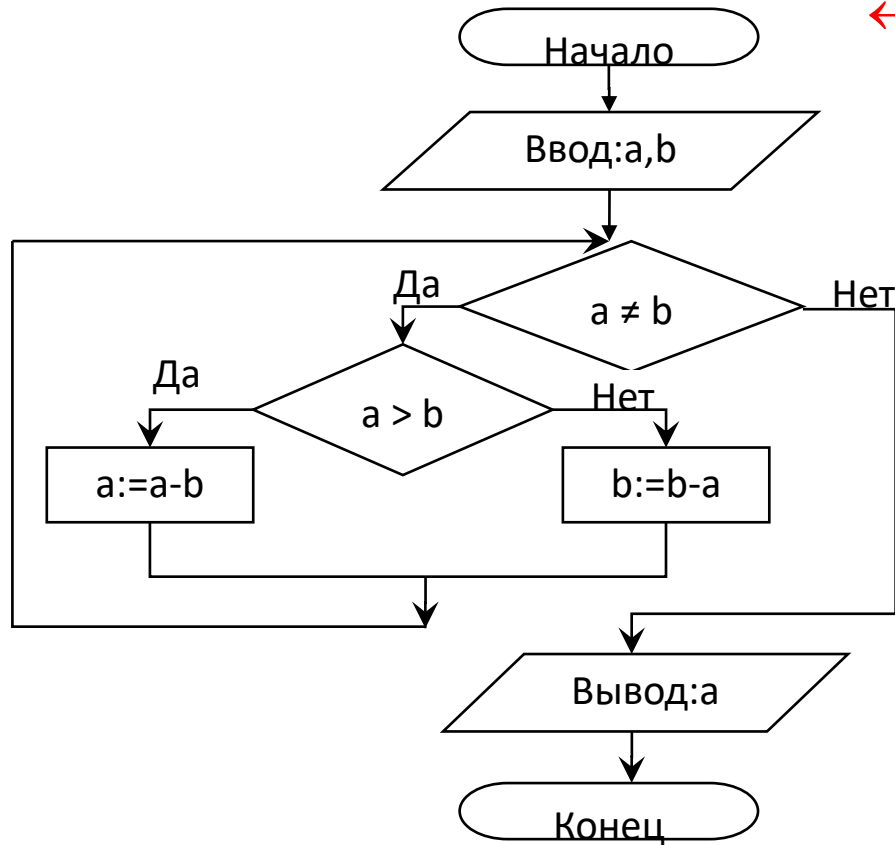
### Алгоритм Евклида

1. Если  $a \neq b$ , то перейти к шагу 2, иначе – перейти к шагу 5.
2. Найти большее из  $a$  и  $b$ .
3. Заменить большее на разность большего и меньшего.
4. Перейти к шагу 1.
5. Выход:  $a$ .

Процесс исполнения теста 1 алгоритма Евклида			
№ шага	a	b	Выход
1.	21	35	
2.		14	
3.	7		
4.		7	
5.			7

*\*Другие тесты проделать самостоятельно*

## 2. Графическая форма



←Сравните→

### Алгоритм Евклида

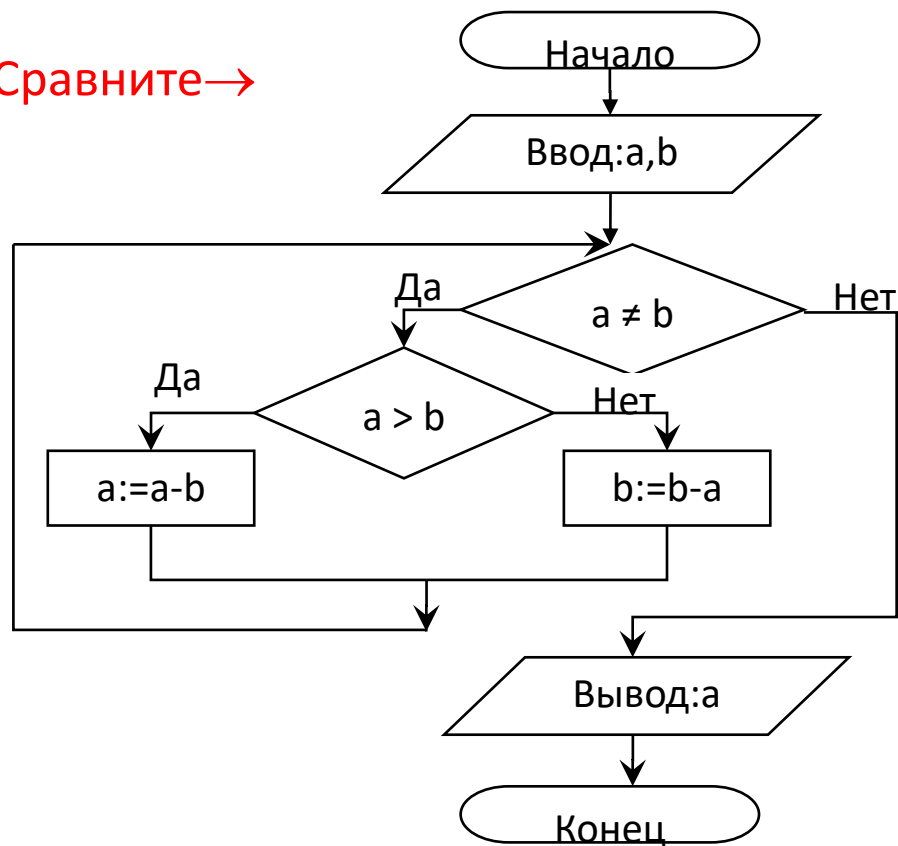
1. Если  $a \neq b$ , то перейти к шагу 2, иначе – перейти к шагу 5.
2. Найти большее из  $a$  и  $b$ .
3. Заменить большее на разность большего и меньшего.
4. Перейти к шагу 1.
5. Выход:  $a$ .



### 3. Язык программирования

```
program Euklid;  
var  
  a, b : integer;  
begin  
  readln(a, b);  
  while a <> b do  
    if a > b then  
      a := a - b;  
    else  
      b := b - a;  
    writeln(a);  
  end.
```

←Сравните→



### Контрольные вопросы к п. 3:

- Перечислите формы представления алгоритма.
- В чем особенность каждой из форм представления алгоритма?
- Всегда ли можно алгоритм, представленный в одной из форм записи, преобразовать в другую форму?

## 4. Три вида структур алгоритма

# Виды структуры алгоритма

```
graph TD; A[Виды структуры алгоритма] --> B[Линейный]; A --> C[Разветвляющийся]; A --> D[Циклический]; B --> E[последовательность действий без ветвлений и возвратов]; C --> F[в зависимости от условия выполняется либо одна, либо другая последовательность действий]; D --> G[последовательность действий, которая выполняется многократно];
```

## Линейный

последовательность действий без ветвлений и возвратов

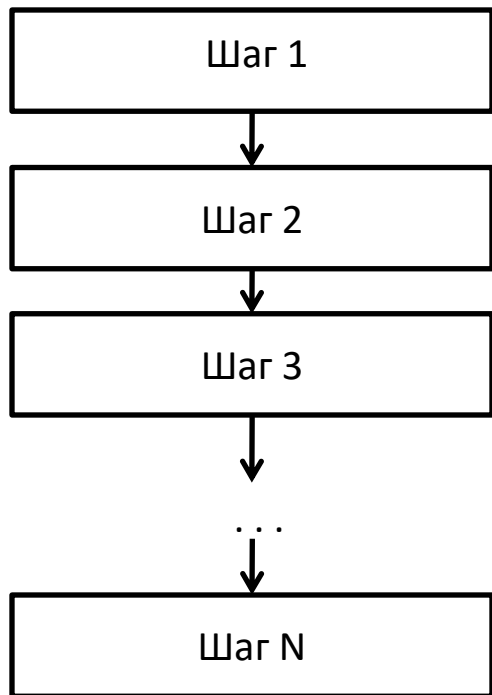
## Разветвляющийся

в зависимости от условия выполняется либо одна, либо другая последовательность действий

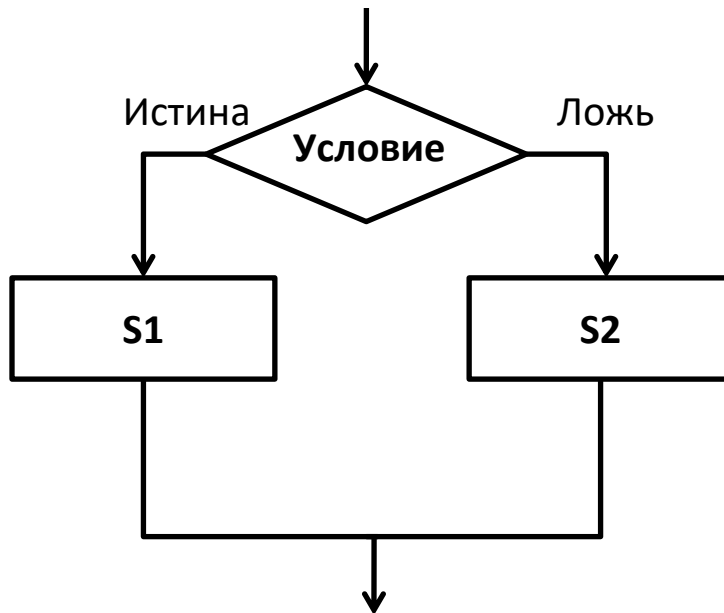
## Циклический

последовательность действий, которая выполняется многократно

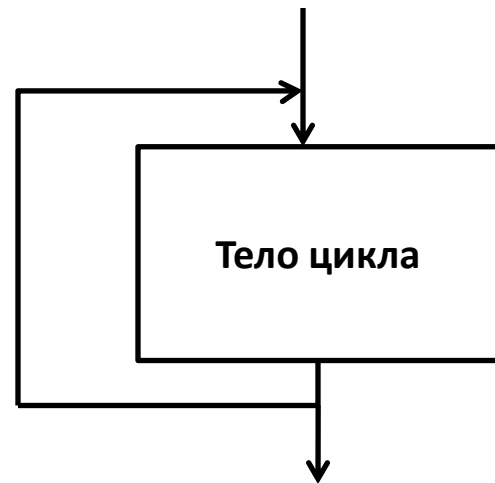
## Линейный



## Разветвляющийся



## Циклический




***Логическая структура*** любого алгоритма может быть представлена комбинацией трех видов алгоритмических структур :

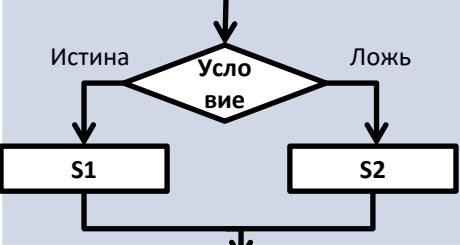

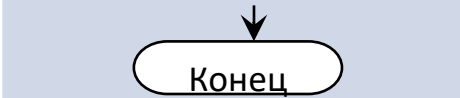
- линейный алгоритм (простое следование);
- разветвляющийся алгоритм (развилка);
- циклический алгоритм (повтор).



# Условные обозначения блок-схем (1)

Условное графическое обозначение	Название	Комментарий
	Стрелка	Означает, к исполнению какого действия следует перейти. С помощью стрелки обозначают последовательность исполнения команд
	Начало	Точка блок-схемы, с которой начинается исполнение алгоритма
	Ввод или Вывод	Блок, означающий, что в этом месте алгоритма необходимо произвести ввод или вывод данных
	Простое действие (присваивание)	Один элементарный шаг алгоритма. Присваивание значения выражения переменной (в данном случае переменной х присвоено значение 1)

# Условные обозначения блок-схем (2)

Условное графическое обозначение	Название	Комментарий
	Условие	Исполнение предполагает вычисление условия – логического выражения. Если условие истинно (Истина, True), то необходимо перейти к действию по стрелке помеченной Истина (Т), если условие ложно – то по стрелке Ложь (F , False)
	Модификация (цикл со счетчиком)	Повторение исполнения тела цикла для каждого из последовательных значений целочисленного счетчика i от 1 до n.
	Конец	Точка блок-схемы, дойдя до которой прекращается процесс исполнения алгоритма.

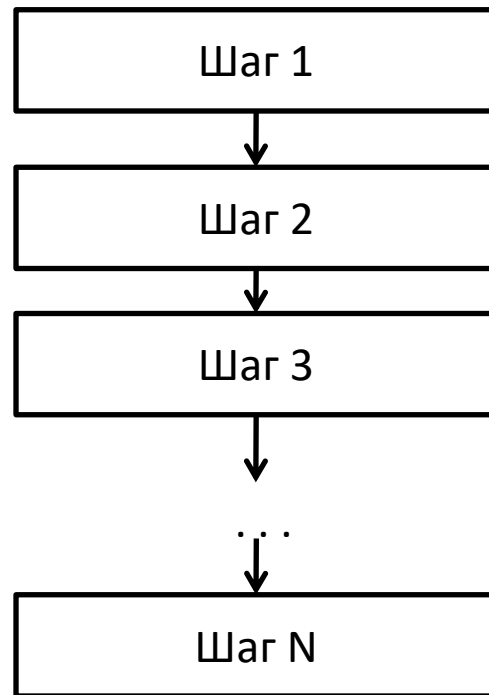


## 5. Линейные алгоритмы

## Линейный алгоритм

простая последовательность шагов, которые исполняются в том порядке, в котором они перечислены в алгоритме.

В линейном алгоритме не может быть ветвлений и возвратов.



### Пример 1.

Составить блок-схему алгоритма, решающего следующую задачу. *Даны три вещественных положительных числа  $a$ ,  $b$  и  $c$ . Найти площадь треугольника, стороны которого равны  $a$ ,  $b$  и  $c$ .*

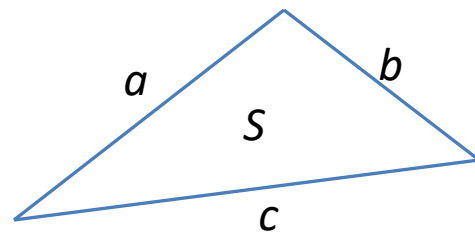
### Решение.

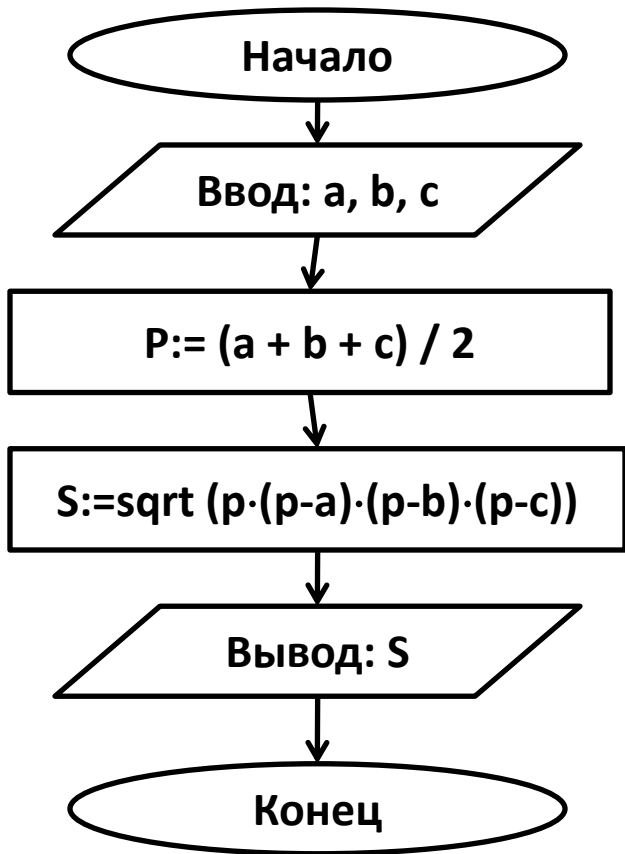
Для нахождения площади треугольника по трем его известным сторонам воспользуемся формулой Герона:

$$S = \sqrt{p(p-a)(p-b)(p-c)}$$

где  $p$  – полупериметр треугольника, равный

$$p = \frac{a+b+c}{2}$$





### Примечания

- Здесь важен порядок шагов!
- Не рассматривается условие существования треугольника

## Пример 2.

Даны значения двух действительных переменных  $a$  и  $b$ . Обменять местами их значения, то есть добиться, чтобы  $a$  получила значение, которое изначально имела переменная  $b$ , а  $b$  – получила бы значение  $a$ .

### Решение.

В данном случае для обмена значениями двух переменных необходима третья – дополнительная переменная

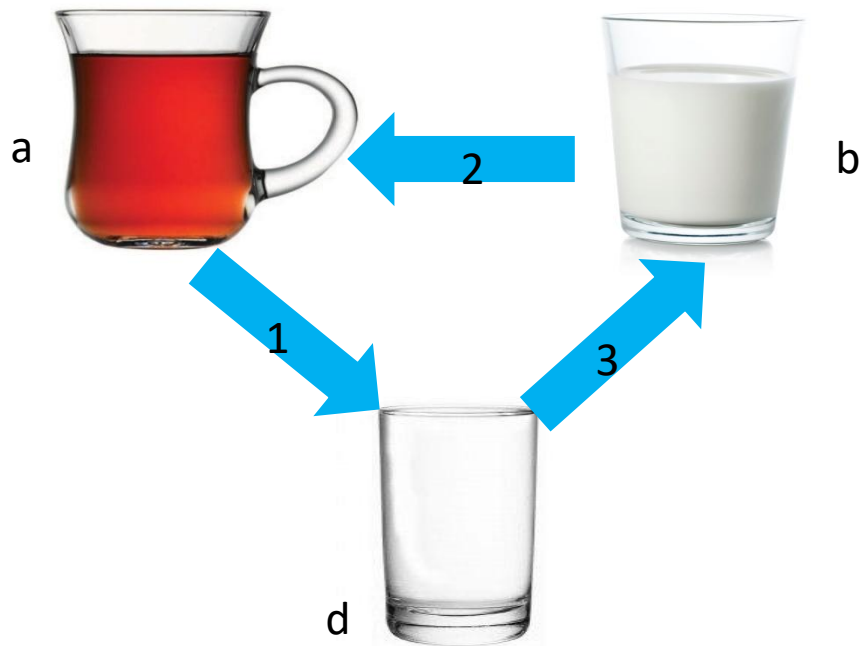
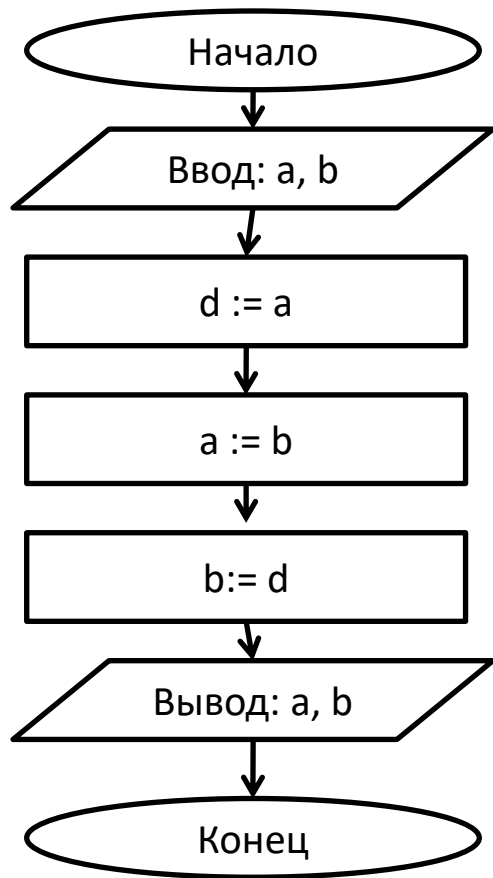
1.  $d := a$



2.  $a := b$

3.  $b := d$

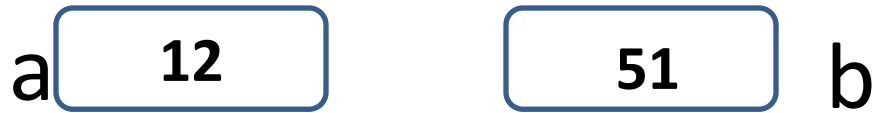
## решение примера 2



### Пример 3.

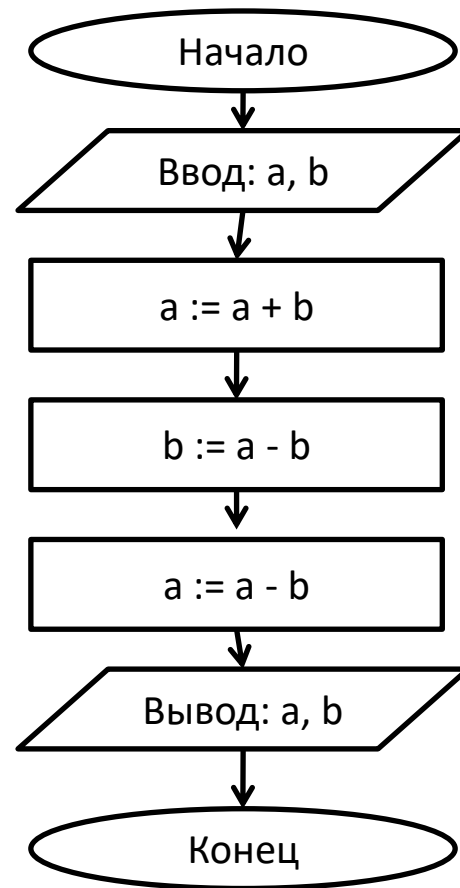
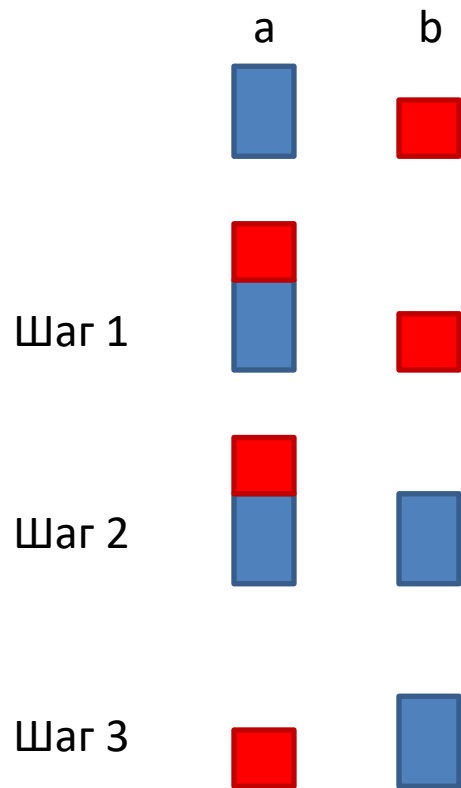
Даны значения двух действительных переменных  $a$  и  $b$ .  
Обменять местами их значения, при этом нельзя  
использовать никаких дополнительных переменных.

**Решение.**



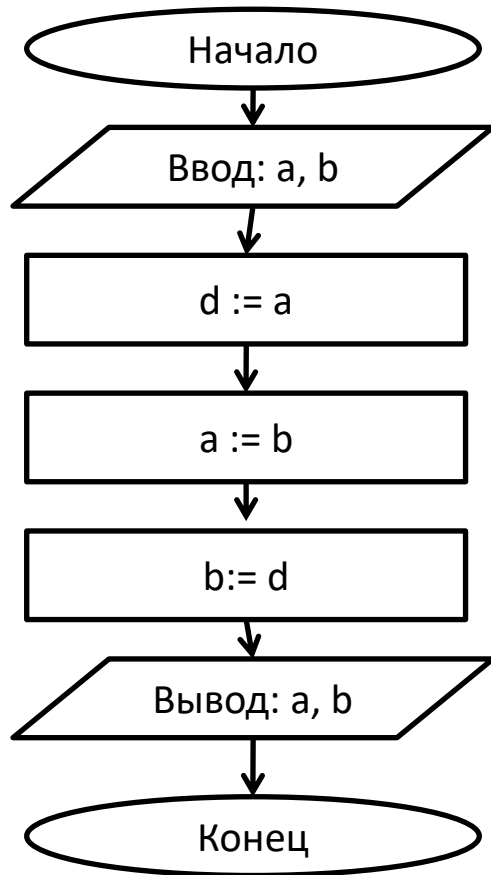
*Теряем исходное значение  $a$ !*

### Решение примера 3





## Решение примера 2

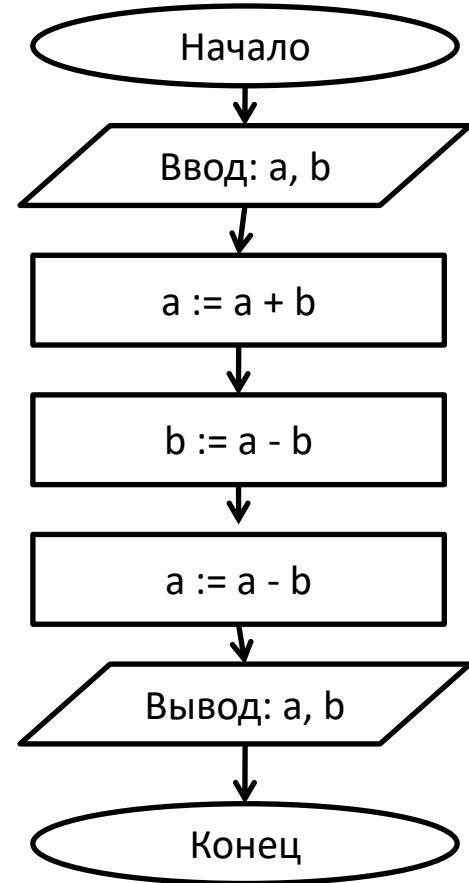


←Сравнить→

←Лучше по времени

Лучше по памяти→

## Решение примера 3



## Упражнения к п. 5:

Составить блок-схемы:

1. Даны значения трех действительных переменных  $a$ ,  $b$  и  $c$ . Обменять местами их значения так, чтобы
  - $a$  получила бы исходное значение  $b$ ;
  - $b$  получила бы исходное значение  $c$ ;
  - $c$  получила бы исходное значение  $a$ .
2. Решить ту же задачу без использования дополнительных переменных.
3. Дано  $a$ . Найти  $a^{10}$  за четыре операции умножения.

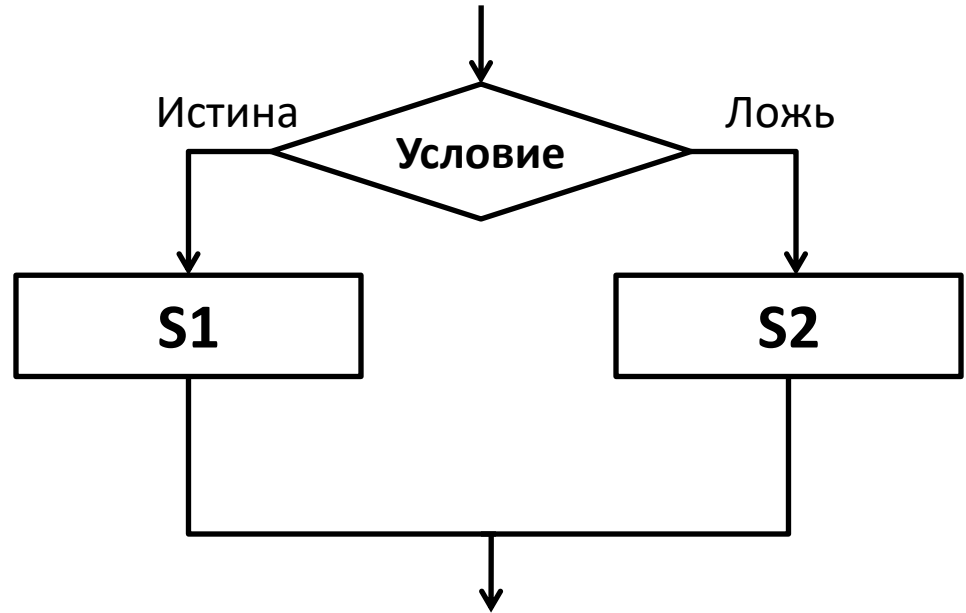
## 6. Разветвляющиеся алгоритмы

## Разветвляющийся алгоритм

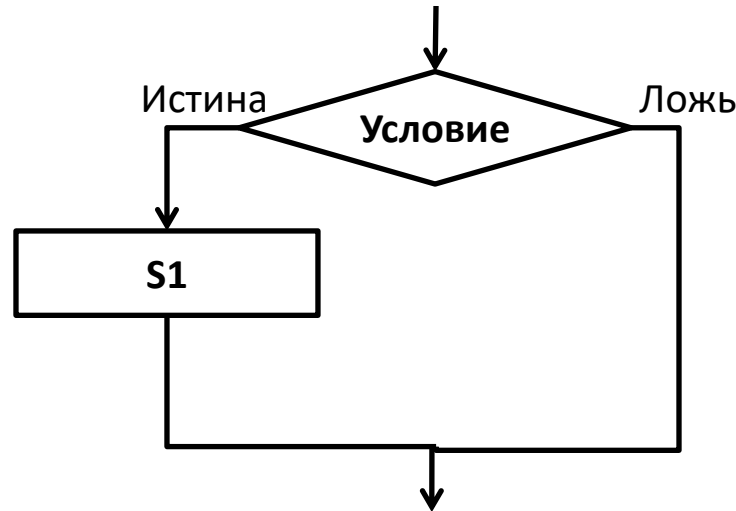
это алгоритм, в котором в зависимости от условия выполняются либо одни, либо другие действия.

Если условие имеет значение «Истина» (true, T), то будет исполнено действие S1,

«Ложь» (false, F), то будет исполнено действие S2.



- При исполнении ветвления будет исполнены либо действия S1, либо действия S2. Одновременно пройти по обеим ветвям или не пройти ни по одной *нельзя!*
- Иногда одна из ветвей алгоритма с ветвлением отсутствует. Этот случай называют **неполным ветвлением**:

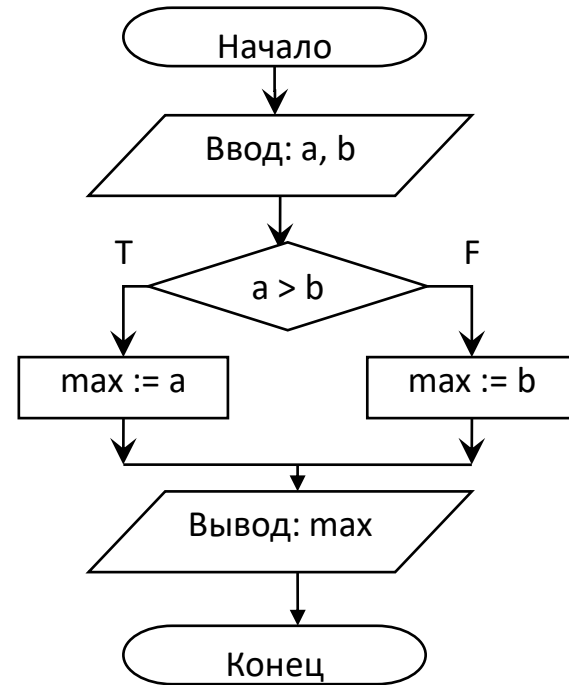


## Пример 4.

Даны значения двух действительных переменных  $a$  и  $b$ . Найти наибольшее значение из  $a$  и  $b$ .

Таблица тестов

№ теста	Вход		Выход
	$a$	$b$	
1.	2	6	6
2.	4	0	4
3.	8	8	8

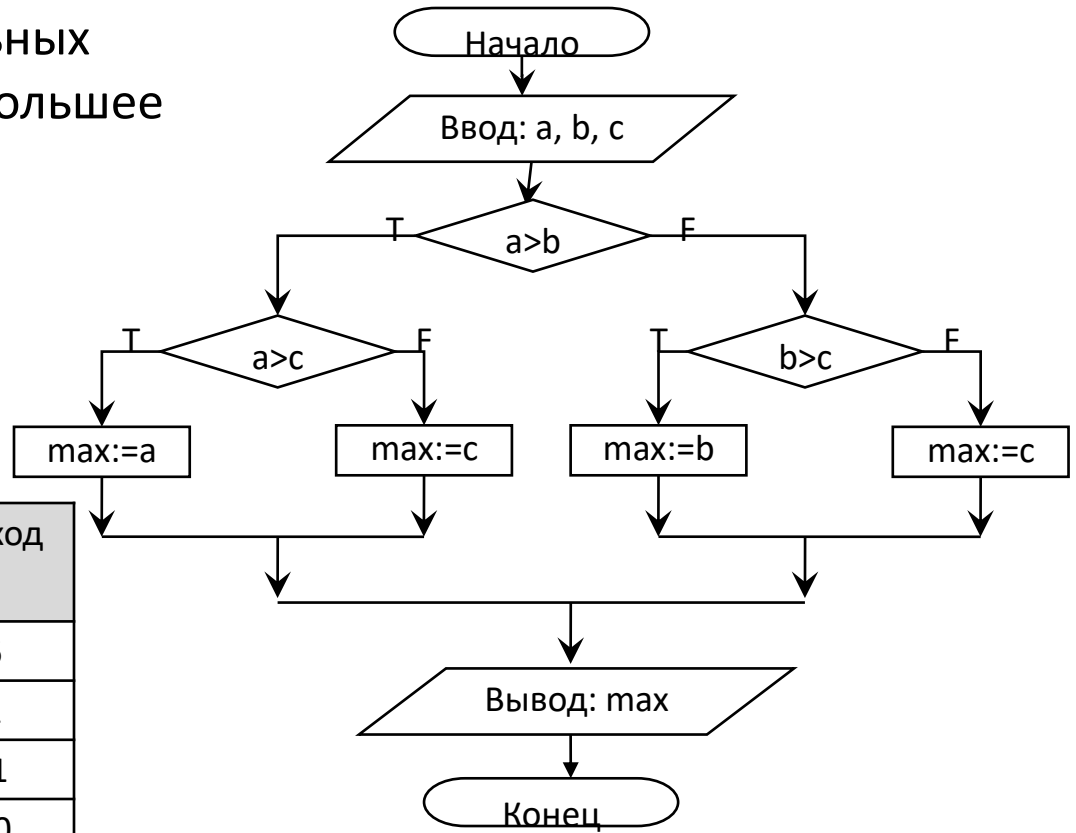


Пример 5.

Даны значения трех действительных переменных a, b и c. Найти наибольшее значение из a, b и c.

Таблица тестов

№ теста	Вход			Выход
	a	b	c	
1.	2	6	4	6
2.	1	0	1	1
3.	-1	-2	-3	-1
4.	25	10	30	30
5.	8	8	8	8

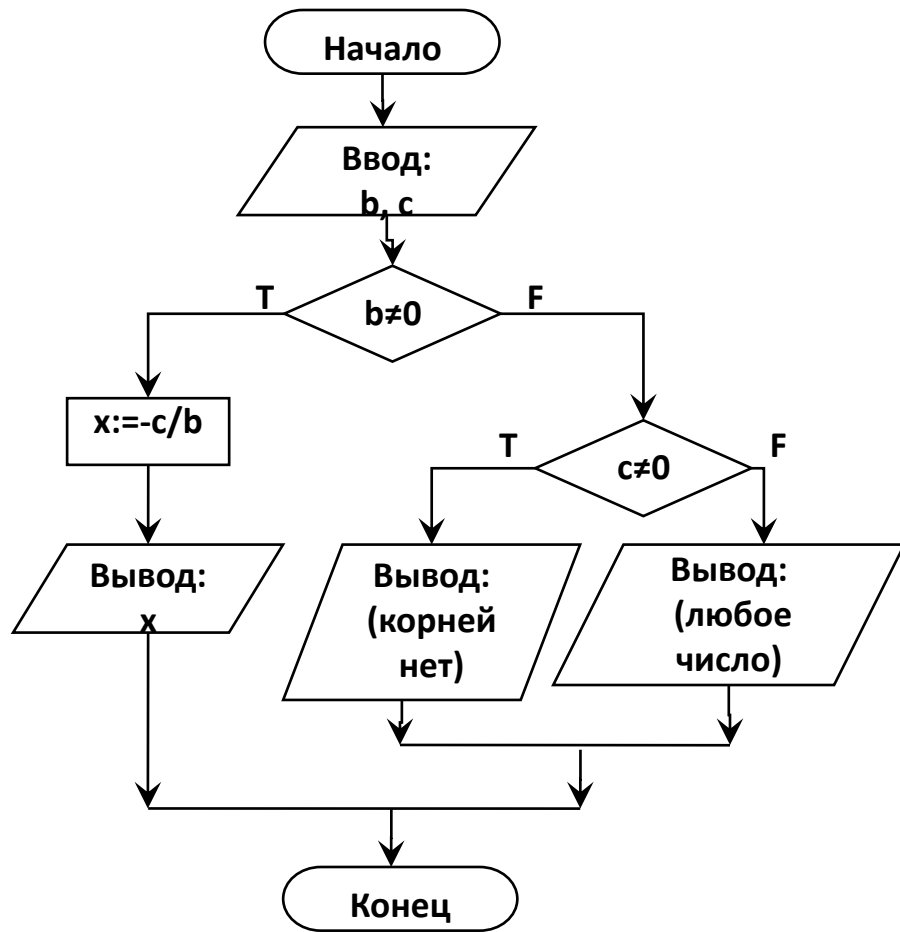


## Пример 6.

Даны значения действительных переменных  $b$  и  $c$ . Решить линейное уравнение  $b x + c = 0$ .

Таблица тестов

№ теста	Вход		Выход
	$b$	$c$	
1.	1	-2	2
2.	2	1	-0,5
3.	10	0	0
4.	0	1	корней нет
5.	0	0	любое число



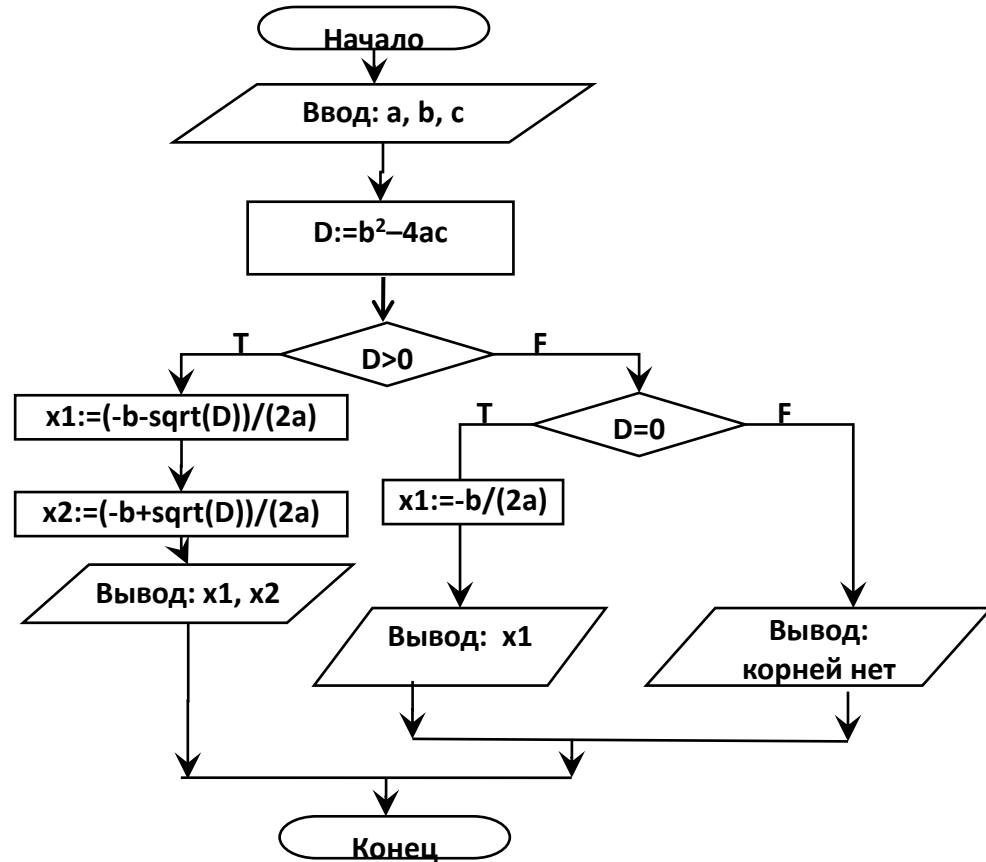


## Пример 7.

Даны значения действительных переменных  $a$ ,  $b$  и  $c$ , причем  $a \neq 0$ .  
Решить уравнение  $ax^2+bx+c=0$ .

Таблица тестов

№ теста	Вход			Выход
	a	b	c	
1.	1	2	-3	-3; 1
2.	2	8	8	-2
3.	2	-1	4	корней нет



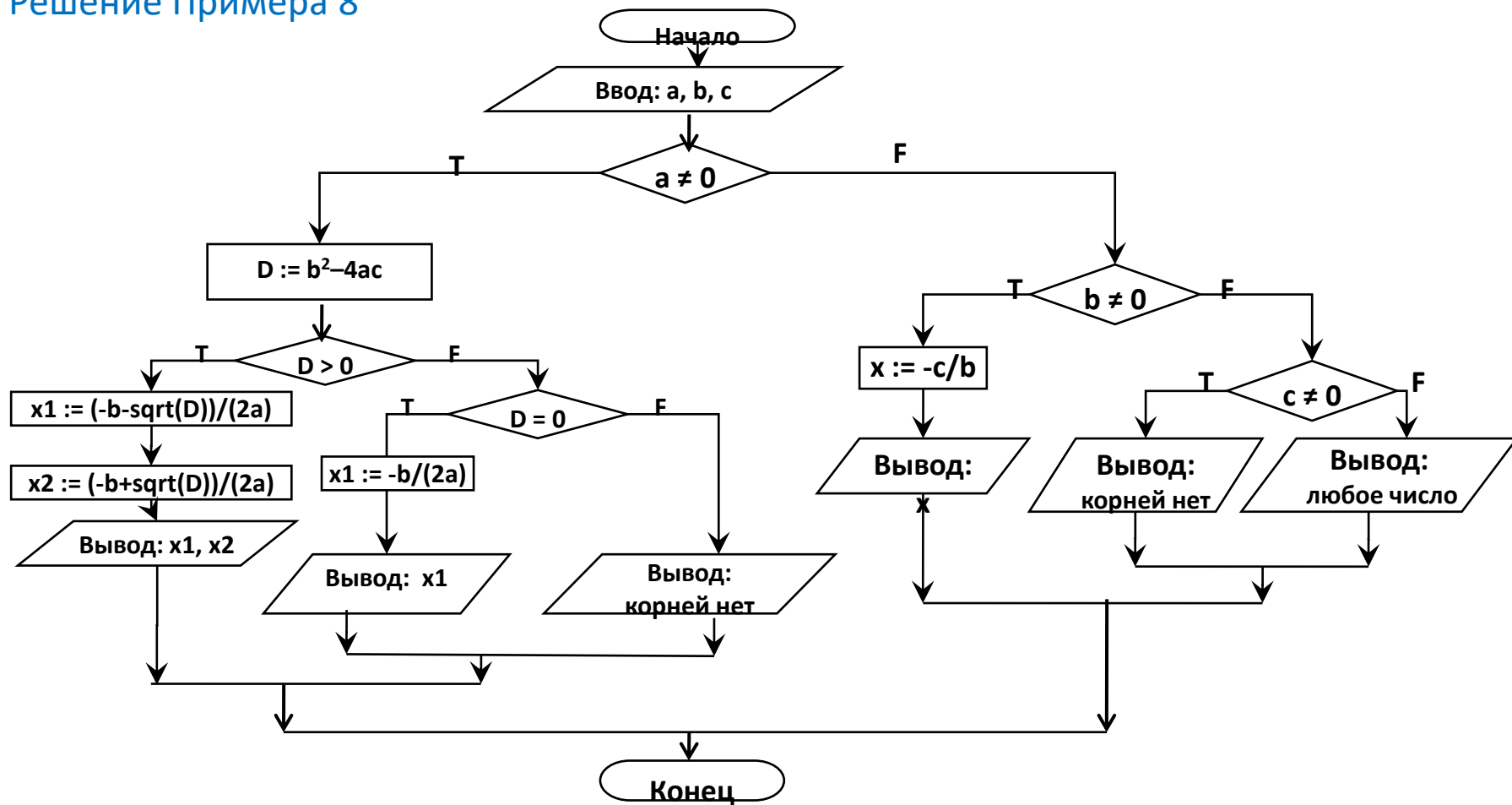
## Пример 8 (самостоятельно).

Даны значения действительных переменных  $a$ ,  $b$  и  $c$ . Решить уравнение  $ax^2+bx+c=0$ .

Таблица тестов

№ теста	Вход			Выход
	a	b	c	
1.	1	2	-3	-3; 1
2.	2	8	8	-2
3.	2	-1	4	нет корней
4.	0	1	-2	2
5.	0	0	1	нет корней
6.	0	0	0	любое число

## Решение Примера 8



### Упражнения к п. 6:

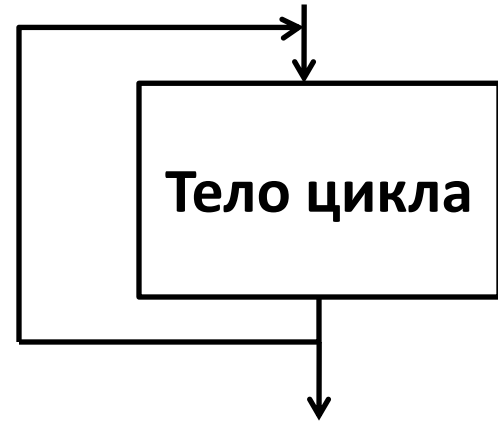
Составить блок-схемы:

1. Даны три точки вещественной оси  $a$ ,  $b$  и  $c$ . Указать две из них, наиболее удаленные друг от друга.
2. Даны три различные вещественные числа  $a$ ,  $b$  и  $c$ . Найти то из них, которое на числовой оси лежит между двумя другими.
3. Даны три вещественные числа  $a$ ,  $b$  и  $c$  ( $a \neq 0$ ). Решить биквадратное уравнение  $ax^4 + bx^2 + c = 0$ .

## 7. Циклические алгоритмы

## Циклический алгоритм

это алгоритм, в котором некоторая последовательность действий (тело цикла) выполняется многократно.



- тело цикла указано в алгоритме один раз, но исполняться оно может многократно;
- каждое исполнение тела цикла называется ***итерацией***;
- выражение, определяющее будет ли продолжаться исполнение очередной итерации или цикл завершится, называется ***условием цикла***;
- существуют **несколько разных видов** циклических алгоритмов, отличающихся друг от друга расположением условия относительно тела, способом выхода из цикла, наличием счетчика итераций и др.

# Основные виды циклических алгоритмов

```
graph TD; A[Основные виды циклических алгоритмов] --> B[Цикл с параметром]; A --> C[Цикл с предусловием]; A --> D[Цикл с постусловием];
```

Цикл с параметром

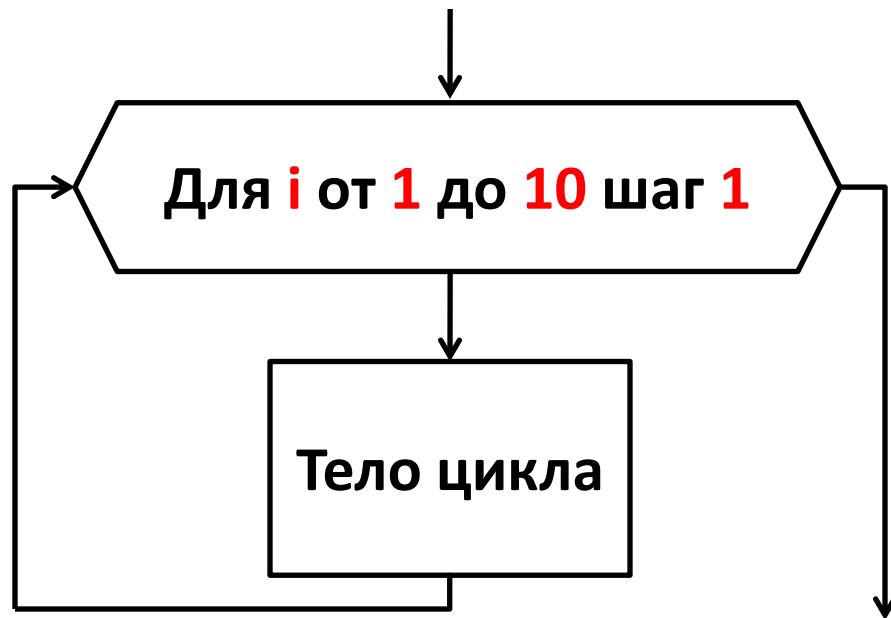
Цикл с предусловием

Цикл с постусловием



## Цикл с параметром

Эта разновидность цикла содержит специальную переменную (*счетчик*, параметр), которая последовательно изменяет свое значение от заданного **начального** до **конечного значения** с некоторым **шагом**, при этом для каждого значения счетчика тело цикла выполняется один раз



## Цикл с параметром

- Начальное, конечное значения и шаг цикла вычисляются *до* *исполнения* цикла, поэтому можно *заранее* подсчитать количество итераций (*циклом с заранее известным числом итераций*)
- Чаще всего используют циклы с шагом 1
- В языках программирования обозначается оператором **for**

## Пример 9

Дано натуральное значение  $n$ . Найти сумму всех натуральных чисел от 1 до  $n$ .

*Например, при  $n = 1$  сумма равна*  
 $1 = 1$

*При  $n = 3$  сумма равна*  
 $1 + 2 + 3 = 6$

*При  $n = 5$  сумма равна*  
 $1 + 2 + 3 + 4 + 5 = 15$

Таблица тестов

№ теста	Вход	Выход
	$n$	
1.	1	1
2.	3	6
3.	5	15
4.	100	5050

Процесс вычисления  $1+2+3+4+5$  «вручную на калькуляторе»:



*Включили калькулятор*

**+** **1** **=**

**+** **2** **=**

**+** **3** **=**

**+** **4** **=**

**+** **5** **=**

0

1

3

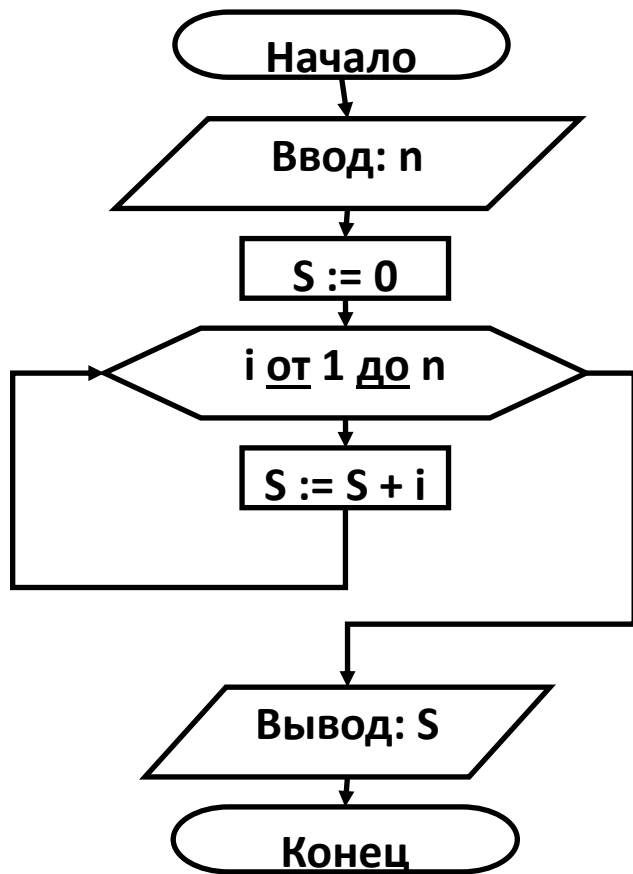
6

10

15

*Слагаемые*

*Суммы*



Процесс исполнения теста № 3 (n = 5)				
№ шага	n	i	S	Выход
1.	5			
2.			0	
3.		1		
4.			1	
5.		2		
6.			3	
7.		3		
8.			6	
9.		4		
10.			10	
11.		5		
12.			15	
13.				15

Слагаемые
Суммы

## Пример 10

Дано натуральное значение  $n$ . Найти  $n!$   
("n факториал")

*При  $n = 1$*

$$1! = 1.$$

*При  $n = 3$*

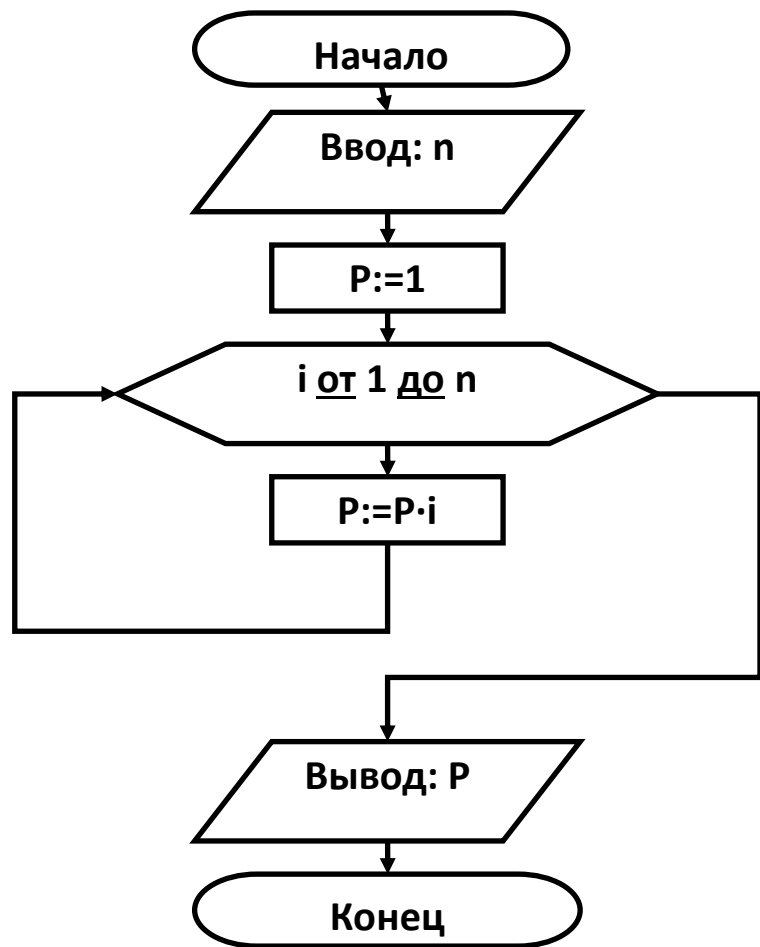
$$3! = 1 \cdot 2 \cdot 3 = 6$$

*При  $n = 5$*

$$5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$$

Таблица тестов

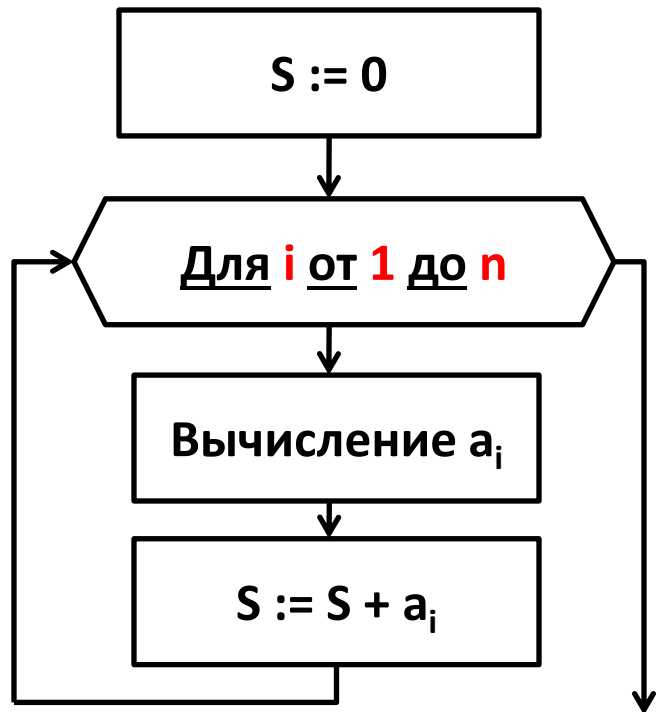
№ теста	Вход	Выход
	n	
1.	1	1
2.	3	6
3.	5	120



Процесс исполнения теста № 3 (n = 5)				
№ шага	n	i	P	Выход
1.	5			
2.			1	
3.		1		
4.			1	
5.		2		
6.			2	
7.		3		
8.			6	
9.		4		
10.			24	
11.		5		
12.			120	
13.				120

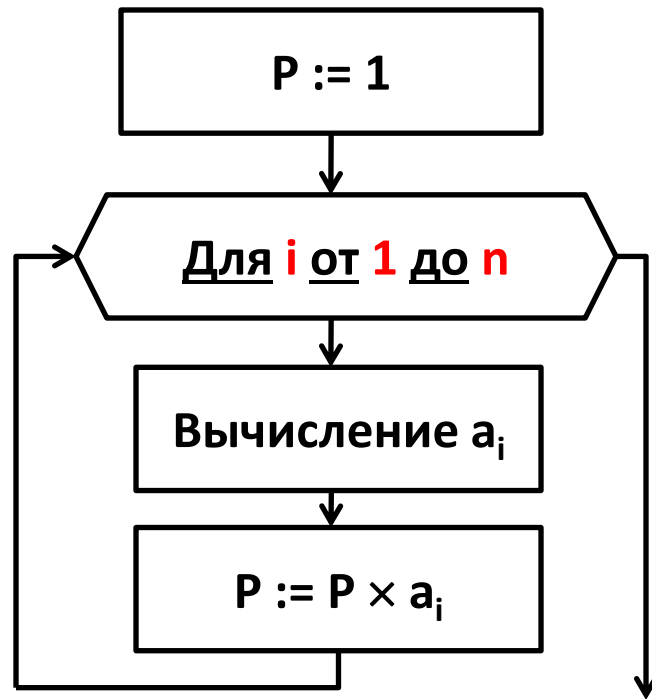
## Алгоритм накопления суммы

$$S = a_1 + a_2 + \dots + a_n$$



## Алгоритм накопления произведения

$$P = a_1 \cdot a_2 \cdot \dots \cdot a_n$$





## Пример 11

Дано натуральное значение  $n$ . Найти

$$\sum_{i=1}^n i! = 1! + 2! + \dots + n!$$

При  $n = 1$

$$1! = 1.$$

При  $n = 3$

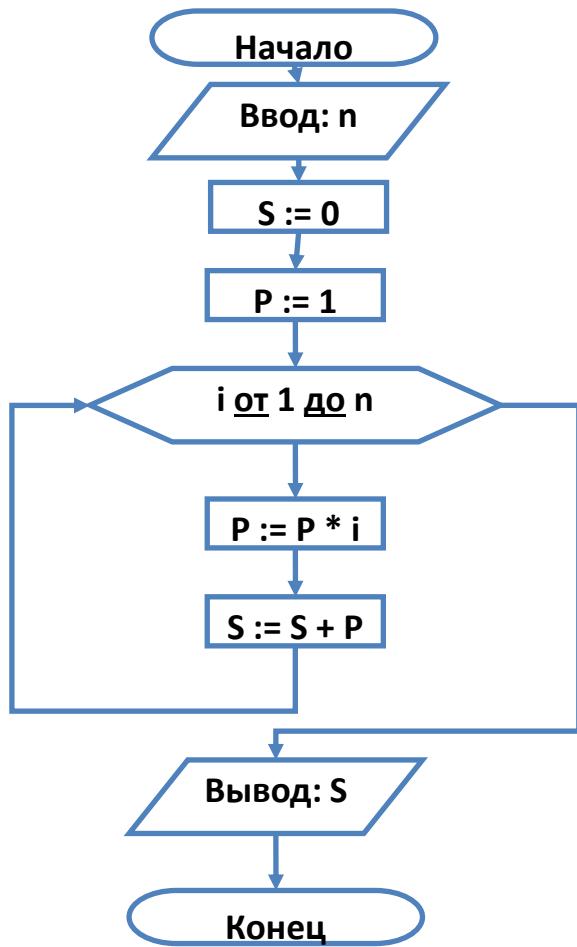
$$1! + 2! + 3! = 1 + 2 + 6 = 9$$

При  $n = 5$

$$1! + 2! + 3! + 4! + 5! = 1 + 2 + 6 + 24 + 120 = 153$$

Таблица тестов

№ теста	Вход	Выход
	n	
1.	1	1
2.	3	9
3.	5	153



Процесс исполнения теста № 2 ( $n = 3$ )					
№ шага	$n$	$i$	$P$	$S$	Выход
1.	3				
2.				0	
3.			1		
4.		1			
5.			1		
6.				1	
7.		2			
8.			2		
9.				3	
10.		3			
11.			6		
12.				9	
13.					9

## Пример 12

Дано натуральное значение  $n$ . Найти -  $n$ -ое число Фибоначчи.

**Опр.** Бесконечная числовая последовательность

$a_1, a_2, a_3, a_4, a_5, a_6, \dots$

задаваемая по правилам

$$a_1 = a_2 = 1;$$

$$a_i = a_{i-1} + a_{i-2} \quad \text{для } i = 3, 4, 5, 6, \dots$$

называется последовательностью **чисел Фибоначчи**.

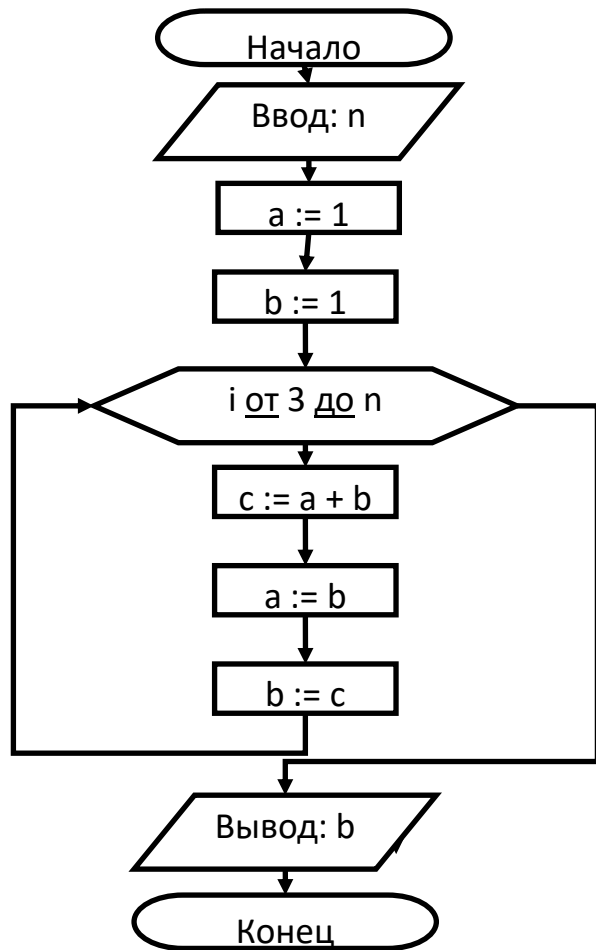


1 1 2 3 5 8 13 21 34 55 89 144 233 ....

Таблица тестов

№ теста	Вход	Выход
	n	
1.	1	1
2.	5	5
3.	7	13
4.	13	233

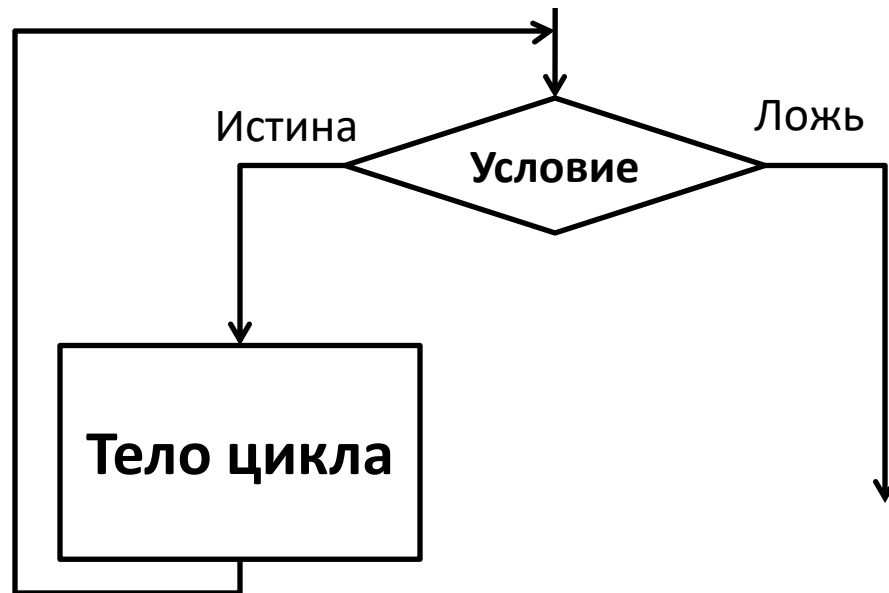
**1 1 2 3 5 8 13 21 34 55 89 144 233 ....**



Процесс исполнения теста № 2 (n = 7)						
№ шага	n	i	a	b	c	Выход
1.	7					
2.			1			
3.				1		
4.		3				
5.					2	
6.			1			
7.				2		
8.		4				
9.					3	
10.			2			
11.				3		
12.		5				
13.					5	
14.			3			
15.				5		
16.		6				
17.					8	
18.			5			
19.				8		
20.		7				
21.					13	
22.			8			
23.				13		
24.						13

## Цикл с предусловием

– это цикл, исполнение которого продолжается, пока истинно условие цикла, проверяемое *до* исполнения очередной итерации.



## Цикл с предусловием

- Если окажется, что с самого начала условие цикла ложно, то тело цикла *ни разу не будет исполнено*.
- Если в процессе исполнения цикла условие всегда принимает значение «истина», то цикл начинает исполняться бесконечно – происходит *зацикливание*.
- В языках программирования обозначается оператором **while**.

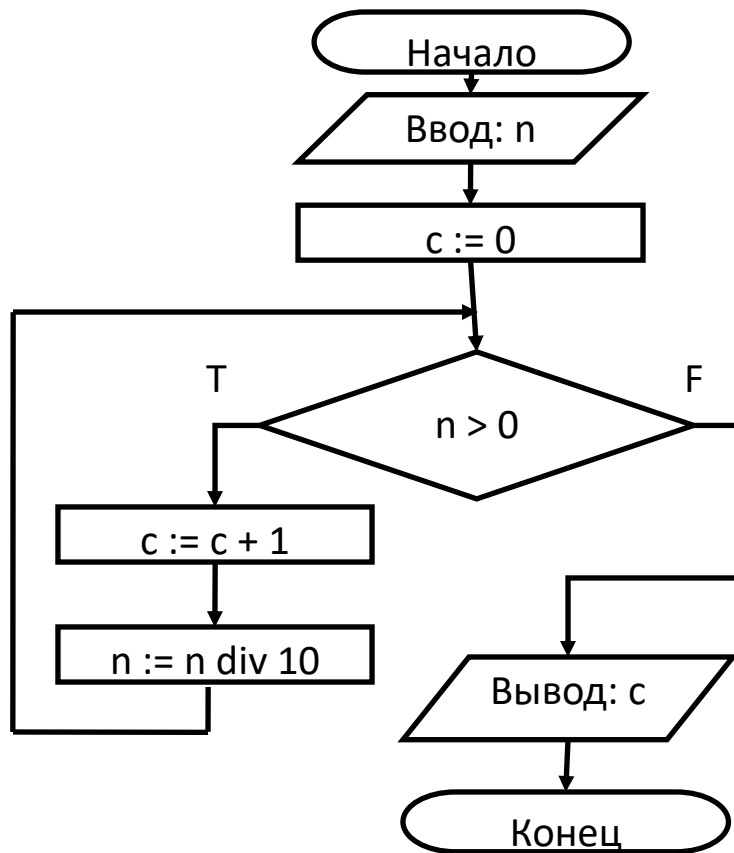
## Пример 13

Дано натуральное число  $n$ .  
Найти количество его цифр.

Таблица тестов

№ теста	Вход $n$	Выход
	$n$	
1.	1234	4
2.	100000	6
3.	1	1

В алгоритме будем использовать операцию **деления нацело на 10 ( $\text{div}$ )**, которая "отбрасывает" последнюю цифру из числа, например  $1234 \text{ div } 10 = 123$





## Пример 14

Даны действительные числа  $a$  и  $b$  ( $a > 1$ ).

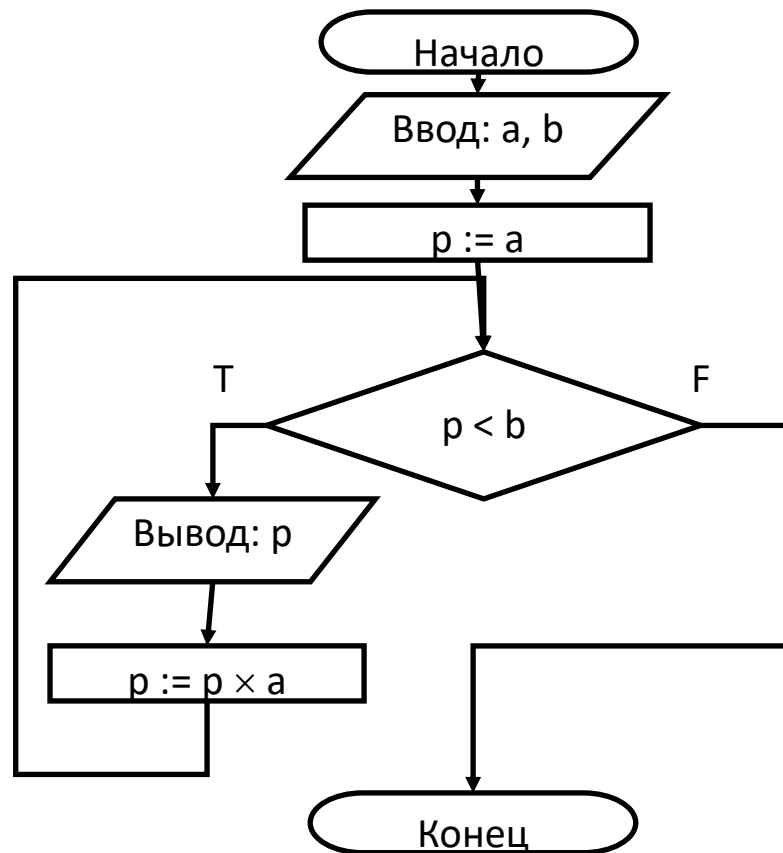
Найти все члены последовательности

$a^1, a^2, a^3, a^4, a^5, \dots$

меньшие  $b$ .

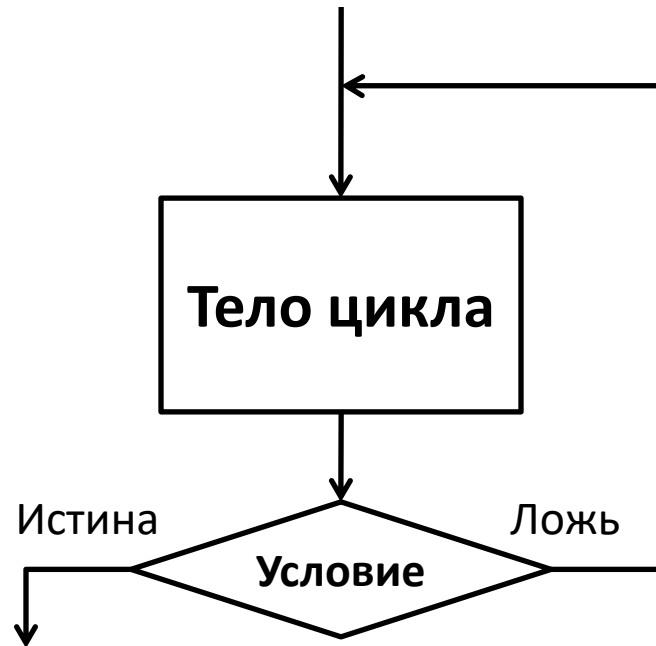
Таблица тестов

№ теста	Вход		Выход
	a	b	
1.	2	10	2; 4; 8
2.	7	50	7; 49
3.	1,5	0	-



## Цикл с постусловием

- это цикл, исполнение которого продолжается, пока *ложно* условие цикла, проверяемое *после* исполнения тела цикла.



## Цикл с постусловием

- независимо от начального значения постусловия тело цикла будет исполнено ***по крайней мере один раз***
- возможна ситуация ***зацикливания***, когда при исполнении постусловие всегда принимает значение «ложь»
- В языках программирования обозначается оператором **do-while** (либо repeat-until).

## Пример 15

Дано натуральное число  $M$ .

Найти первое из чисел

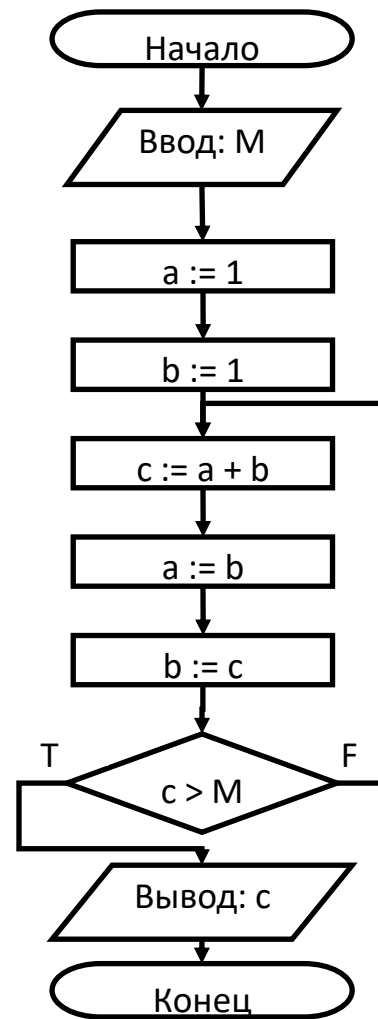
Фибоначчи, большее  $M$ .

1 1 2 3 5 8 13 21 34 55 89 144 233 ....

Таблица тестов

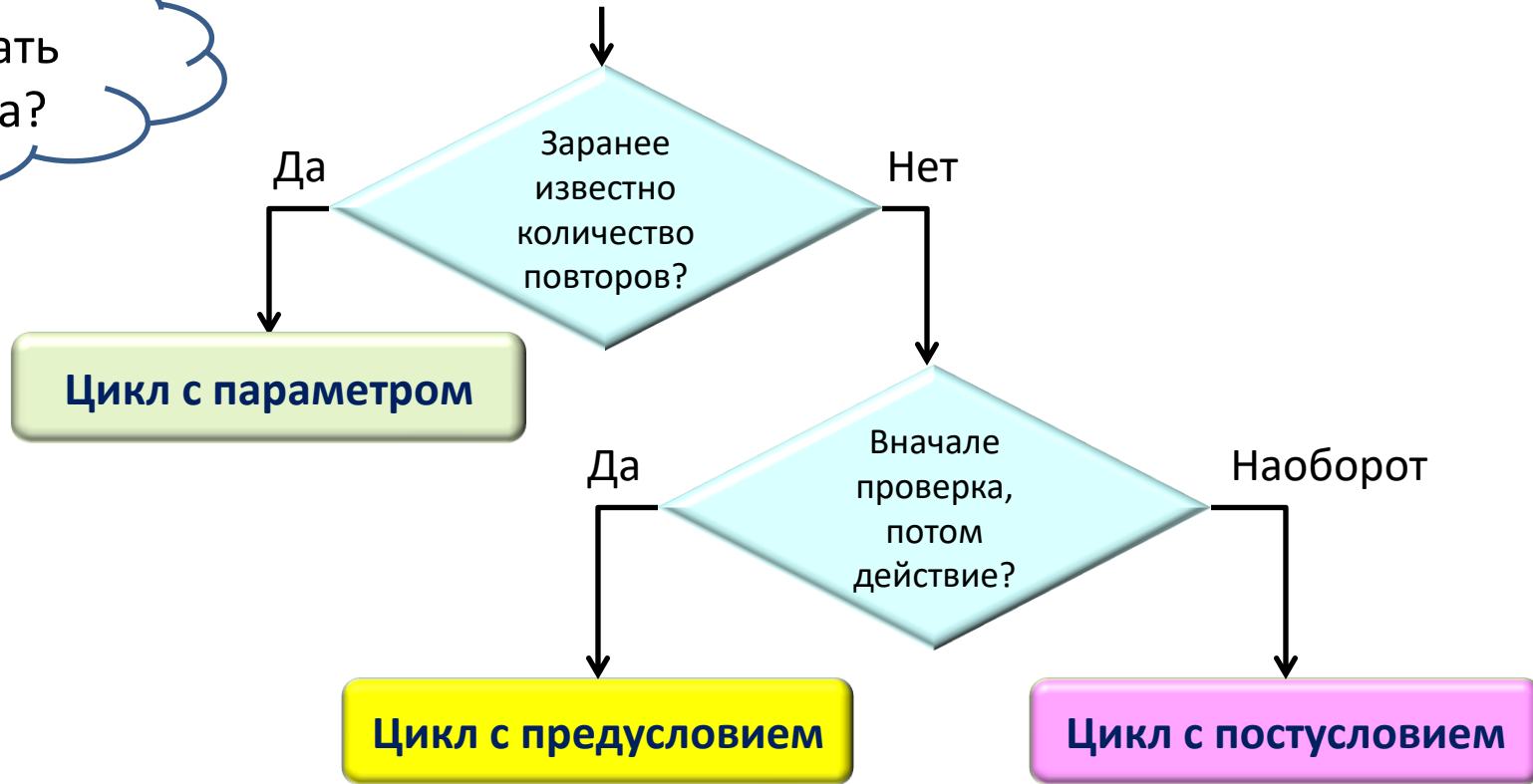
№ теста	Вход	Выход
	$M$	
1.	1	2
2.	10	13
3.	100	144

Вначале находим очередное число Фибоначчи, а затем проверяем, не стало ли оно больше  $M$ . Вначале действие, затем проверка условия. Подходит цикл с постусловием.



# Вывод по теме 7. Циклические алгоритмы

Как выбрать  
вид цикла?



## Упражнения к п. 7:

Составить блок-схемы:

1. Дано натуральное число  $n$ . Найти наименьший среди квадратов натуральных чисел, которые больше  $n$ .
2. Дано натуральное число  $n$ . Найти все натуральные числа  $a$  и  $b$ , представимые в виде  $a^2 + b^2 = n$ .

## 8. Несколько примеров алгоритмов

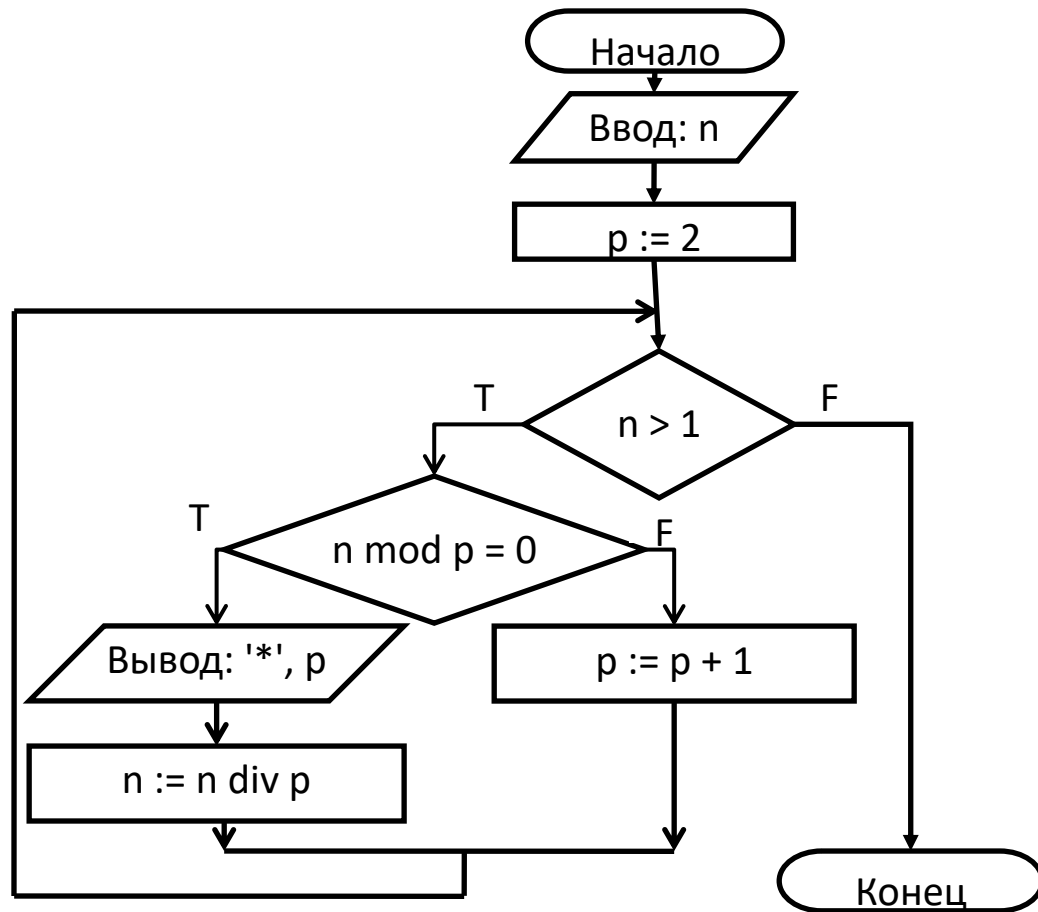
## Пример 16

Дано натуральное число  $n > 1$ .  
Разложить его в произведение  
простых множителей.

Таблица тестов

№ теста	Вход	Выход
	$n$	
1.	60	$*2*2*3*5$
2.	128	$*2*2*2*2*2*2*2$
3.	17	$*17$

В алгоритме будем использовать  
операции частное от деления нацело  
(div) и остаток от деления нацело  
(mod).





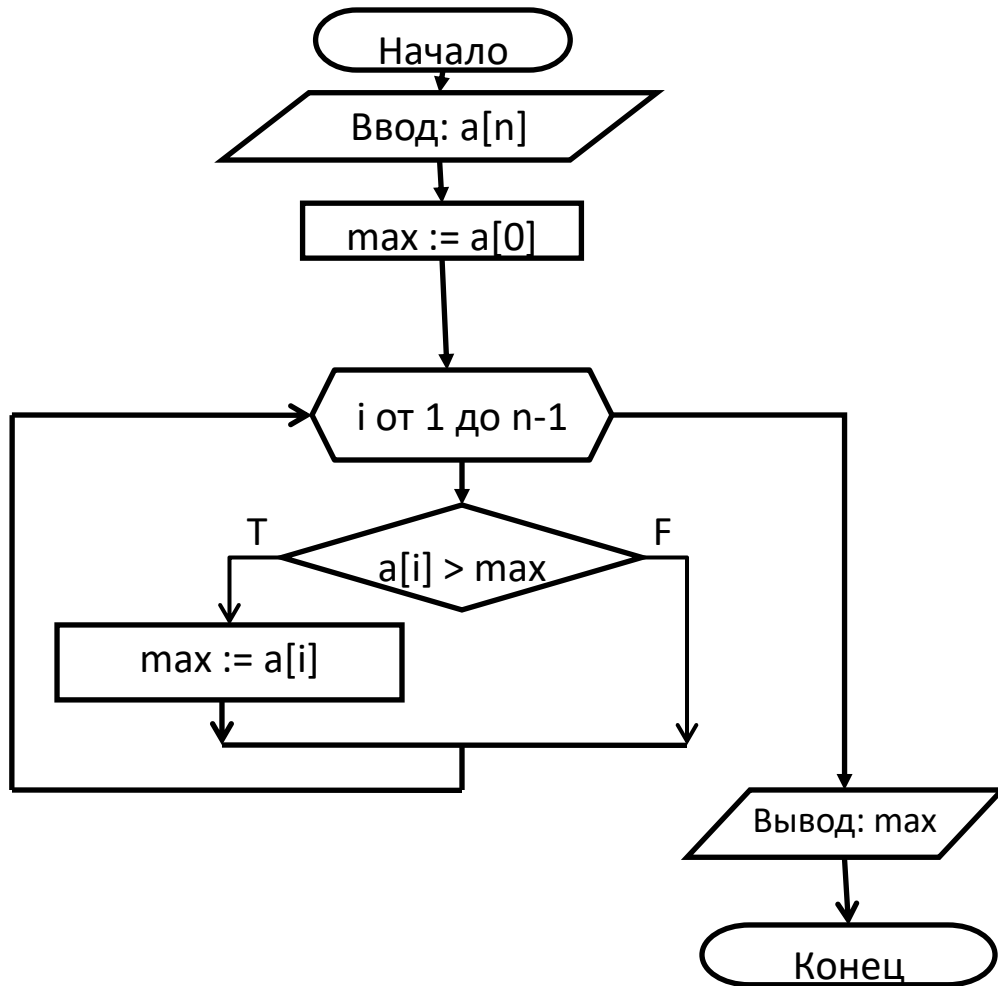
## Пример 17

Дан вещественный массив  $a[0], a[1], a[2], \dots, a[n-1]$ .

Найти максимальное значение среди элементов массива.

0	1	2	3	4
29	-5	17	61	39

max = 61



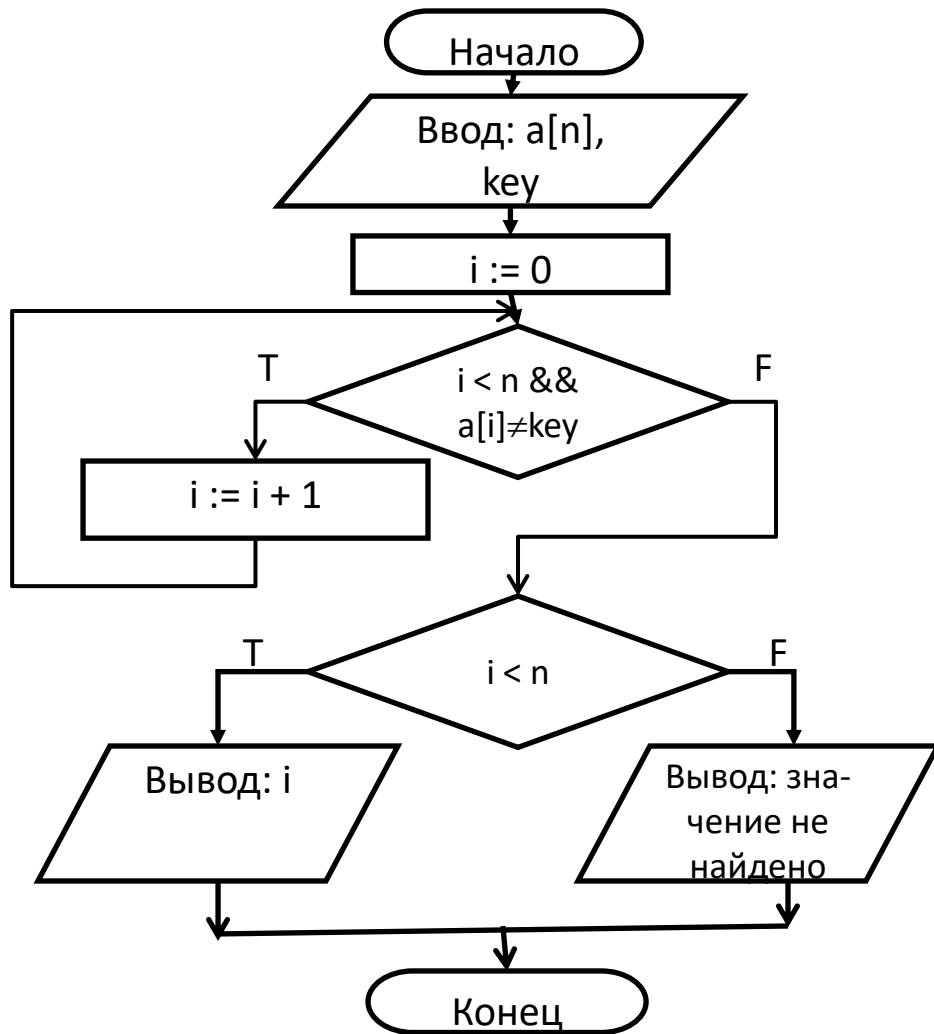
## Пример 18

Дан вещественный массив  $a[0], a[1], a[2], \dots, a[n-1]$ , дано вещественное значение  $key$ . Найти индекс элемента, значение которого равно  $key$ .

0	1	2	3	4
29	-5	17	61	39

1)  $key = 17$   
 $index = 2$

2)  $key = 40$   
значение не найдено



## Пример 19

Дан вещественный массив  $a[0], a[1], a[2], \dots, a[n-1]$ .

Расположить значения в массиве по не убыванию (пузырьковая сортировка)

0	1	2	3	4
29	-5	17	61	39

Отсортировано:

0	1	2	3	4
-5	17	29	39	61

