ОСНОВЫ АЛГОРИТМИЗАЦИИ

Этапы решения задачи на ЭВМ

Работа по решению любой задачи с использованием компьютера делится на следующие этапы:

1. Постановка задачи.
2. Формализация задачи.
3. Построение алгоритма.
4. Составление программы на языке программирования.
5. Отладка и тестирование программы.
6. Проведение расчетов и анализ полученных результатов.

Постановка задачи

На этапе постановки задачи должно быть четко сформулировано, *что дано и что требуется найти.* Здесь очень важно определить *полный набор исходных данных*, необходимых для получения решения.

Формализация задачи

На этом этапе чаще всего задача переводится на язык математических формул, уравнений, отношений.

Если решение требует математического описания какого-то реального объекта, явления или процесса, то формализация равносильна получению соответствующей *математической модели.*

Построение алгоритма

- выбор метода проектирования алгоритма;
- выбор формы записи алгоритма (блок-схемы, псевдокод и др.);
- выбор тестов и метода тестирования;
- проектирование алгоритма.

Составление программы на языке программирования

- выбор языка программирования;
- уточнение способов организации данных;
- запись алгоритма на выбранном языке программирования.

Тестирование и отладка

- синтаксическая отладка;
- отладка семантики и логической структуры;
- тестовые расчеты и анализ результатов тестирования;
- совершенствование программы.

Проведение расчетов и анализ полученных результатов

На этом этапе выполняется анализ результатов решения задачи и уточнение в случае необходимости математической модели с повторным выполнением этапов 2-5.

Алгоритм

Алгоритм

Алгоритмом называется точная инструкция исполнителю в понятной для него форме, определяющая процесс достижения поставленной цели на основе имеющихся исходных данных за конечное число шагов.

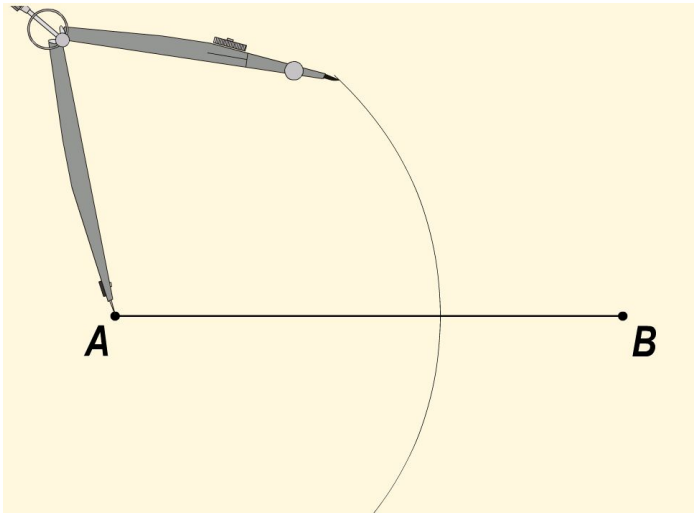
Алгоритм записывается на формальном языке, исключающем неоднозначность толкования.

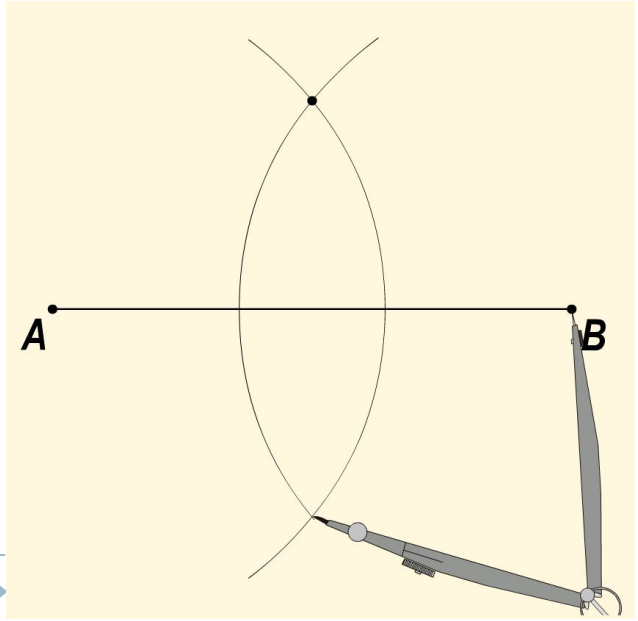
Исполнитель - это человек, компьютер, автоматическое устройство и т.п. Он должен уметь выполнять все команды, составляющие алгоритм, причем механически, «не раздумывая».

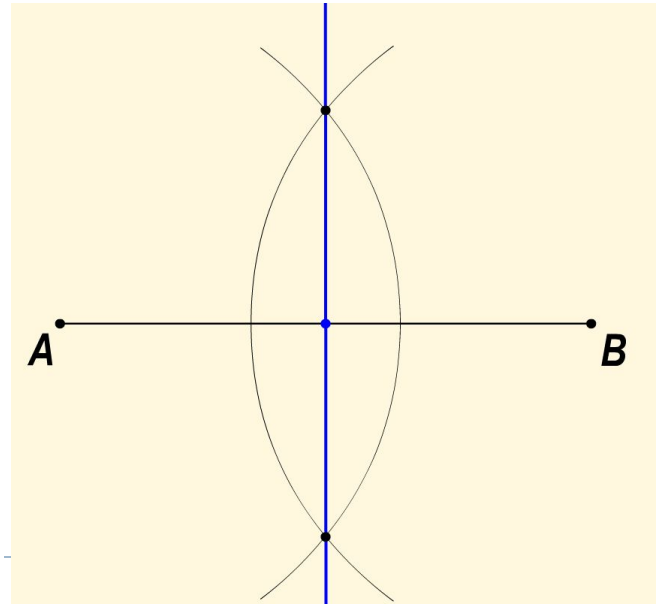
Алгоритм

- Слово алгоритм происходит от algorithmi – латинской формы написания имени великого математика IX в. Аль Хорезми, который сформулировал правила выполнения арифметических действий.
- Первоначально под алгоритмами и понимали только правила выполнения четырех арифметических действий над многозначными числами. В дальнейшем это понятие стали использовать вообще для обозначения последовательности действий, приводящих к решению поставленной задачи.

Алгоритм деления отрезка АВ пополам







Алгоритм деления отрезка АВ пополам

Пример. Алгоритм деления отрезка АВ пополам:

1) поставить ножку циркуля в точку А;
2) установить раствор циркуля равным больше половины длины отрезка АВ;
3) провести дугу;
4) поставить ножку циркуля в точку В;
5) провести дугу;
6) через точки пересечения дуг провести прямую;
7) отметить точку пересечения этой прямой с отрезком АВ.

Система команд исполнителя

- Анализ примеров различных алгоритмов показывает, что запись алгоритма распадается на отдельные указания исполнителю выполнить некоторое законченное действие. Каждое такое указание называется командой.
- Команды алгоритма выполняются одна за другой. После каждого шага исполнения алгоритма точно известно, какая команда должна выполнятся следующей.
- Совокупность команд, которые могут быть выполнены исполнителем, называется системой команд исполнителя.

Свойства алгоритма

Основными свойствами алгоритмов являются:

1. Универсальность (массовость) - применимость алгоритма к различным наборам исходных данных.
2. Дискретность - процесс решения задачи по алгоритму разбит на отдельные действия.
3. Однозначность - правила и порядок выполнения действий алгоритма имеют единственное толкование.
4. Конечность - каждое из действий и весь алгоритм в целом обязательно завершаются.
5. Результативность - по завершении выполнения алгоритма обязательно получается конечный результат.

ОСНОВЫ АЛГОРИТМИЗАЦИИ

Способы записи алгоритмов

Способы записи алгоритмов

Выделяют следующие основные способы записи алгоритмов:

- вербальный, когда алгоритм описывается на человеческом языке;
- символьный, когда алгоритм описывается с помощью набора символов;
- графический, когда алгоритм описывается с помощью набора графических изображений.

Выбор средства для записи алгоритма определяется типом исполняемого алгоритма.

Способы записи алгоритмов

На практике чаще всего встречаются следующие формы представления алгоритмов:

- словесная – записывается на естественном языке;
- графическая – с помощью изображения из графических символов;
- псевдокоды – полуформализованные описания алгоритмов на некотором условном алгоритмическом языке, которые включают в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.;
- программная – тексты на языках программирования.

Пример словесной записи алгоритма

Правило деления обыкновенных дробей:

1. Числитель первой дроби умножить на знаменатель второй дроби.
2. Знаменатель первой дроби умножить на числитель второй дроби.
3. Записать дробь, числитель которой есть результат выполнения пункта 1, а знаменатель — результат выполнения пункта 2.

$$\frac{a}{b} : \frac{c}{d} = \frac{a \cdot d}{b \cdot c} = \frac{m}{n}$$

Пример словесной записи алгоритма

1. Начало алгоритма.
2. Выполнить некоторое действие (оператор) s1.
3. Если выполнено условие "Усл1", то выполнить операторы s2, s3 и перейти к п. 4. Иначе - перейти к пп. 3.1.
   - 3.1. Пока выполняется условие "Усл2", выполнять пп. 3.2 и 3.3. Иначе - перейти к п. 4.
   - 3.2. Если выполнено условие "Усл3", то выполнить оператор s4, иначе — выполнить оператор s5.
   - 3.3. Выполнить оператор s6.
4. Пока выполняется условие "Усл4", выполнять оператор s7. Иначе — перейти к п. 5.
5. Выполнить оператор s8.
6. Конец алгоритма.

Псевдокоды

Примером псевдокода является школьный алгоритмический язык.

- Общий вид алгоритма, записанного на АЯ

алг название алгоритма (аргументы и результаты)
дано условия применимости алгоритма
надо цель выполнения алгоритма
нач описание промежуточных величин,
последовательность команд (тело алгоритма)
кон

Пример алгоритма на АЯ

алг площадь треугольника (**арг вещ** *a*, *b*, **рез вещ** *s*)

нач

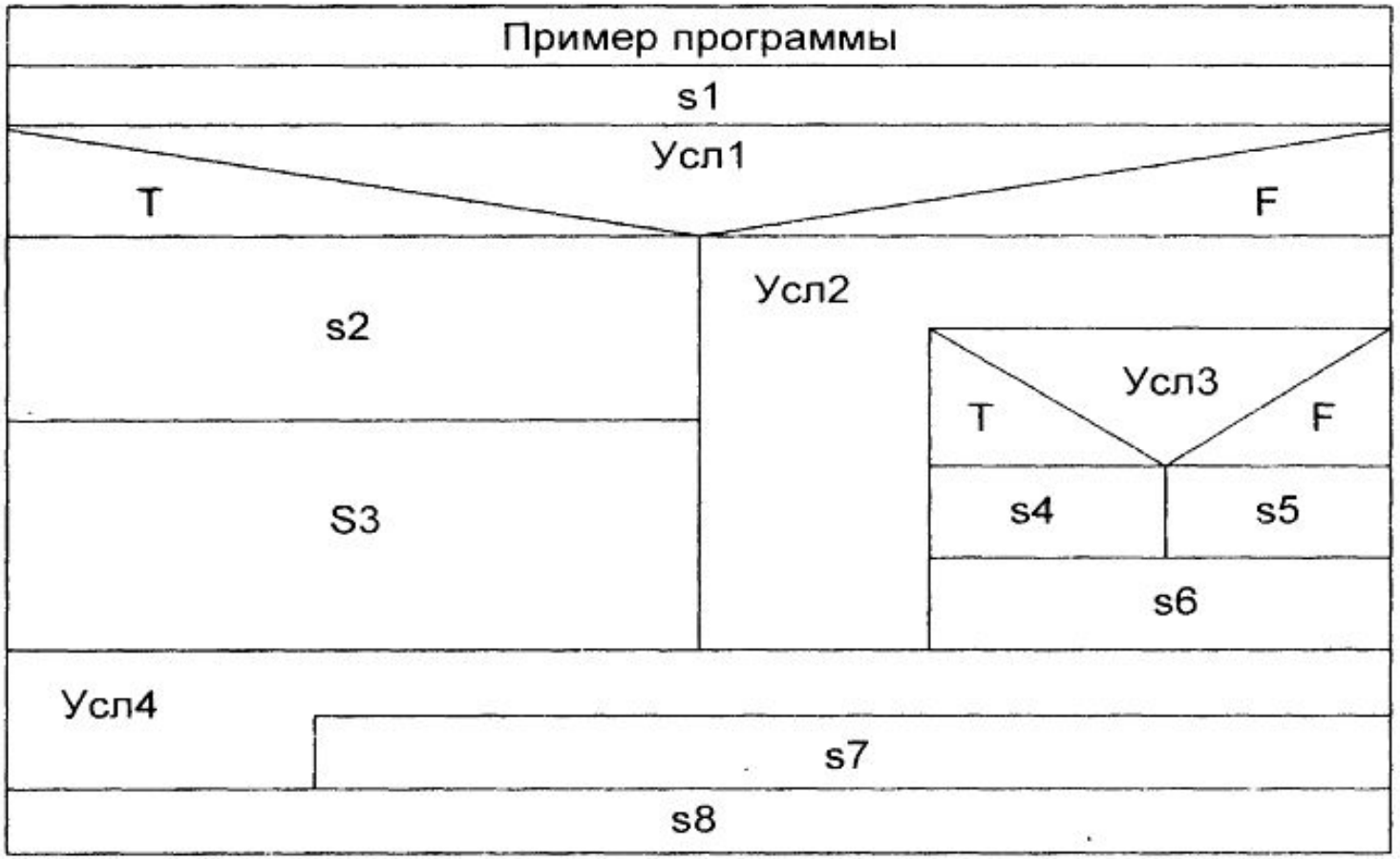
вещ *h*

h:=*a*/2

s:=*h*·*b*

кон

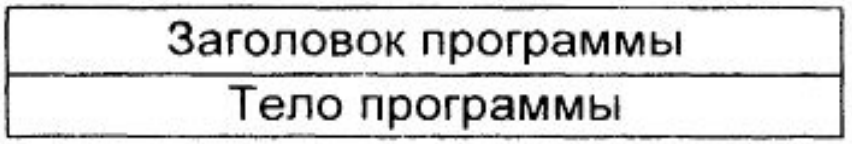
Графическая запись алгоритма с помощью диаграммы Нэсси-Шнейдермана



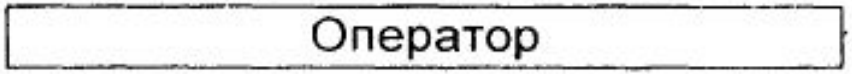
Графическая запись алгоритма с помощью диаграммы Нэсси-Шнейдермана

Графические элементы диаграммы Нэсси-Шнейдермана

Программа

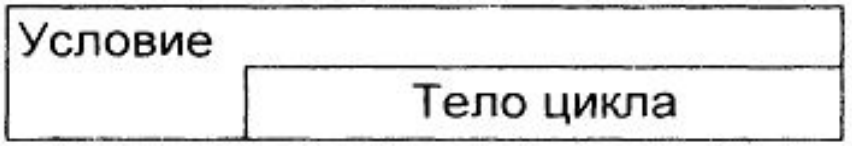


Простая конструкция



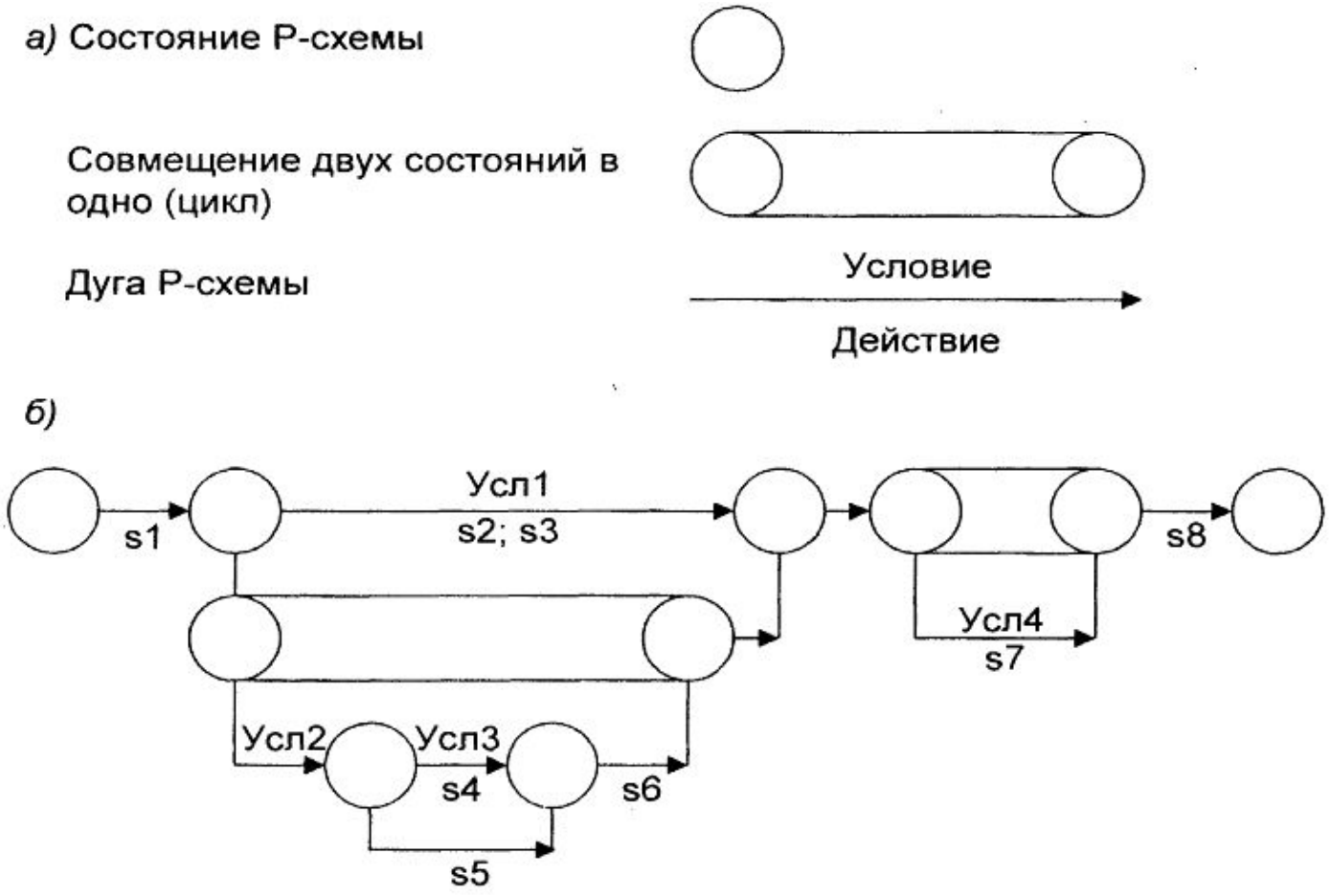
Условная конструкция

Циклическая конструкция



Графическая запись алгоритма с помощью Р-схемы

Р-технология программирования разработана в Институте Кибернетики АН УССР.



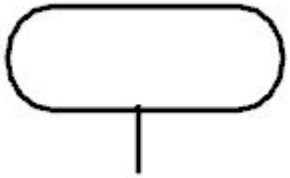
Р-схема:
а) графические элементы; *б)* пример использования

- Общепринятыми способами записи являются графическая запись с помощью блок-схем и символьная запись с помощью какого-либо алгоритмического языка.

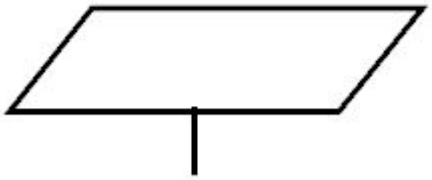
Графическая запись с помощью блок-схем

- Описание алгоритма с помощью блок схем осуществляется рисованием последовательности геометрических фигур, каждая из которых подразумевает выполнение определенного действия алгоритма.
- Порядок выполнения действий указывается стрелками.
- Написание алгоритмов с помощью блок-схем регламентируется ГОСТом. (**ГОСТ 19.701-90, ГОСТ 19.002-80, ГОСТ 19.003-80)**

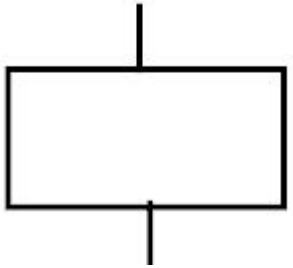
Основные условные обозначения, используемые при записи алгоритма с помощью блок-схем



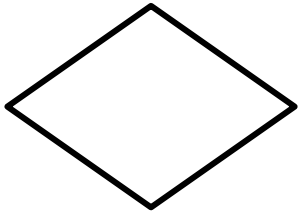
Начало программы



Операции ввода-вывода

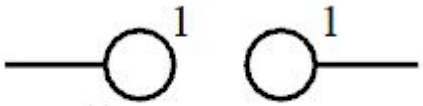


Обработка данных

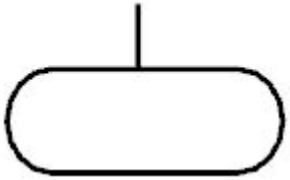


Ветвление, выбор

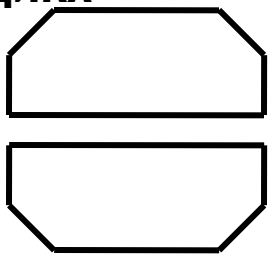
Счетный цикл



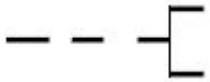
Соединитель



Завершение программы

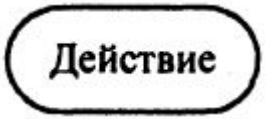


Граница цикла

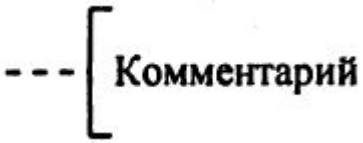


Комментарий

Основные условные обозначения, используемые при записи алгоритма с помощью блок-схем

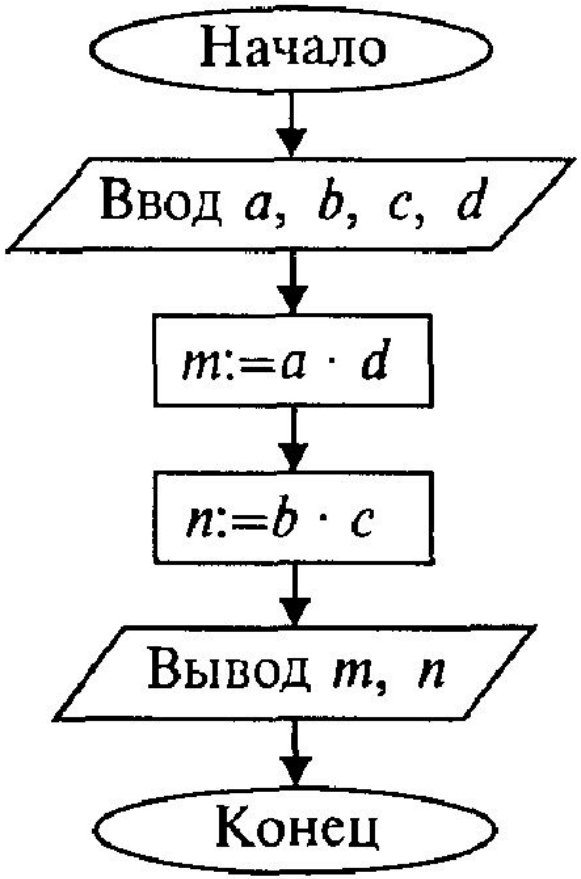
Название блока	Обозначение	Назначение блока
Терминатор		Начало, завершение программы или подпрограммы
Процесс		Вычислительное действие ли последовательность действий
Данные		Операции ввода-вывода
Решение		Ветвления, выбор, итерационные и поисковые циклы
Подготовка		Счетные циклы
Граница цикла		Любые циклы

Основные условные обозначения, используемые при записи алгоритма с помощью блок-схем

Название блока	Обозначение	Назначение блока
Предопределенный процесс		Вызов процедур
Соединитель		Маркировка разрыва линии и продолжения ее в другом месте блок-схемы
Комментарий		Пояснения к операциям
Линии потока данных		Отображает поток данных или управления

Пример записи алгоритма с помощью блок-схем

- Правило деления обыкновенных дробей:



Использование блок-схем дает возможность:

- наглядно отобразить базовые конструкции алгоритма;
- сосредоточить внимание на структуре алгоритма, а не на синтаксисе языка;
- анализировать логическую структуру алгоритма;
- преобразовывать алгоритм методом укрупнения (сведения к единому блоку) или детализации – разбиения на ряд блоков;
- использовать принцип блочности при коллективном решении сложной задачи;
- осуществить быструю проверку разработанного алгоритма (на уровне идеи);

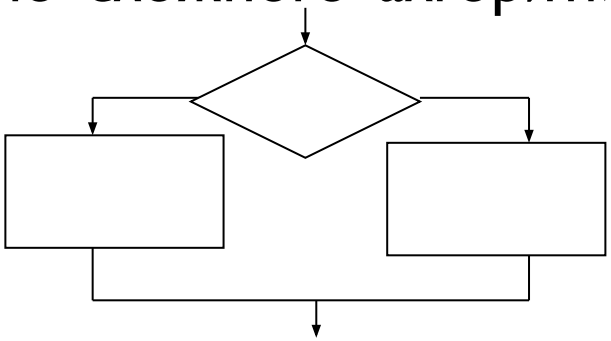
ОСНОВЫ АЛГОРИТМИЗАЦИИ

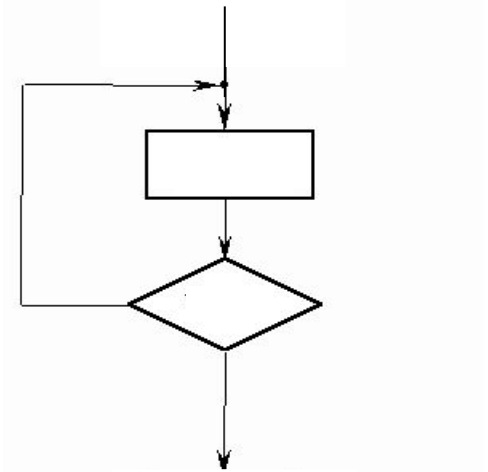
Базовые алгоритмические структуры

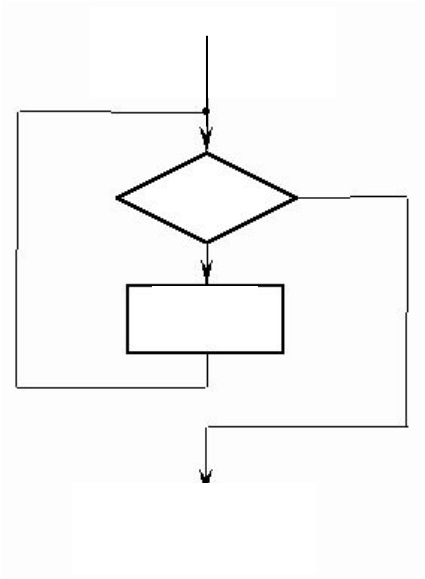
Базовые алгоритмические структуры

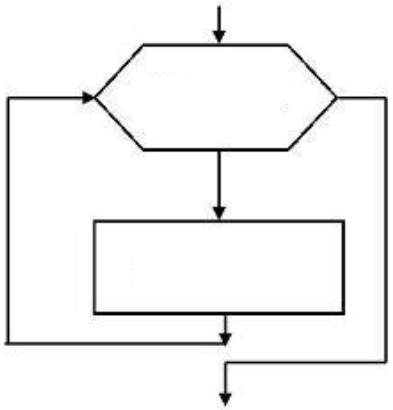
В теории программирования доказано, что для записи любого сколь угодно сложного алгоритма достаточно трех базовых структур:

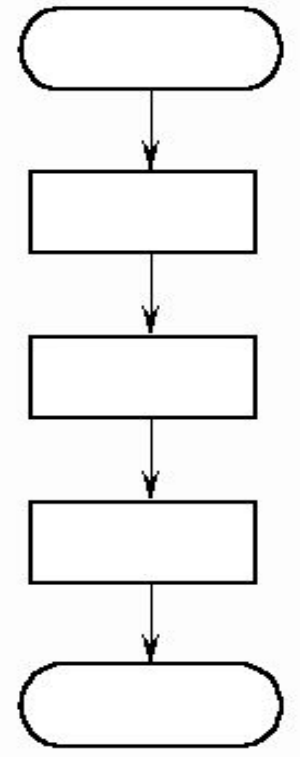
- *следование,*
- *ветвление,*
- *повторение.*











Следование

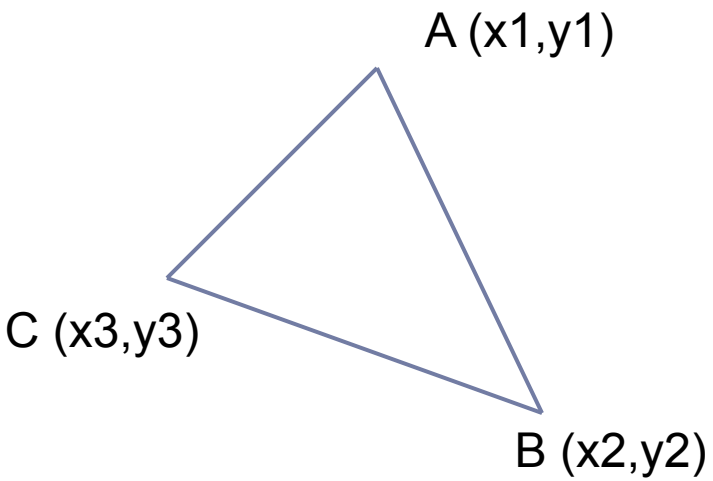
Базовая структура "следование". Образуется последовательностью действий, следующих одно за другим:

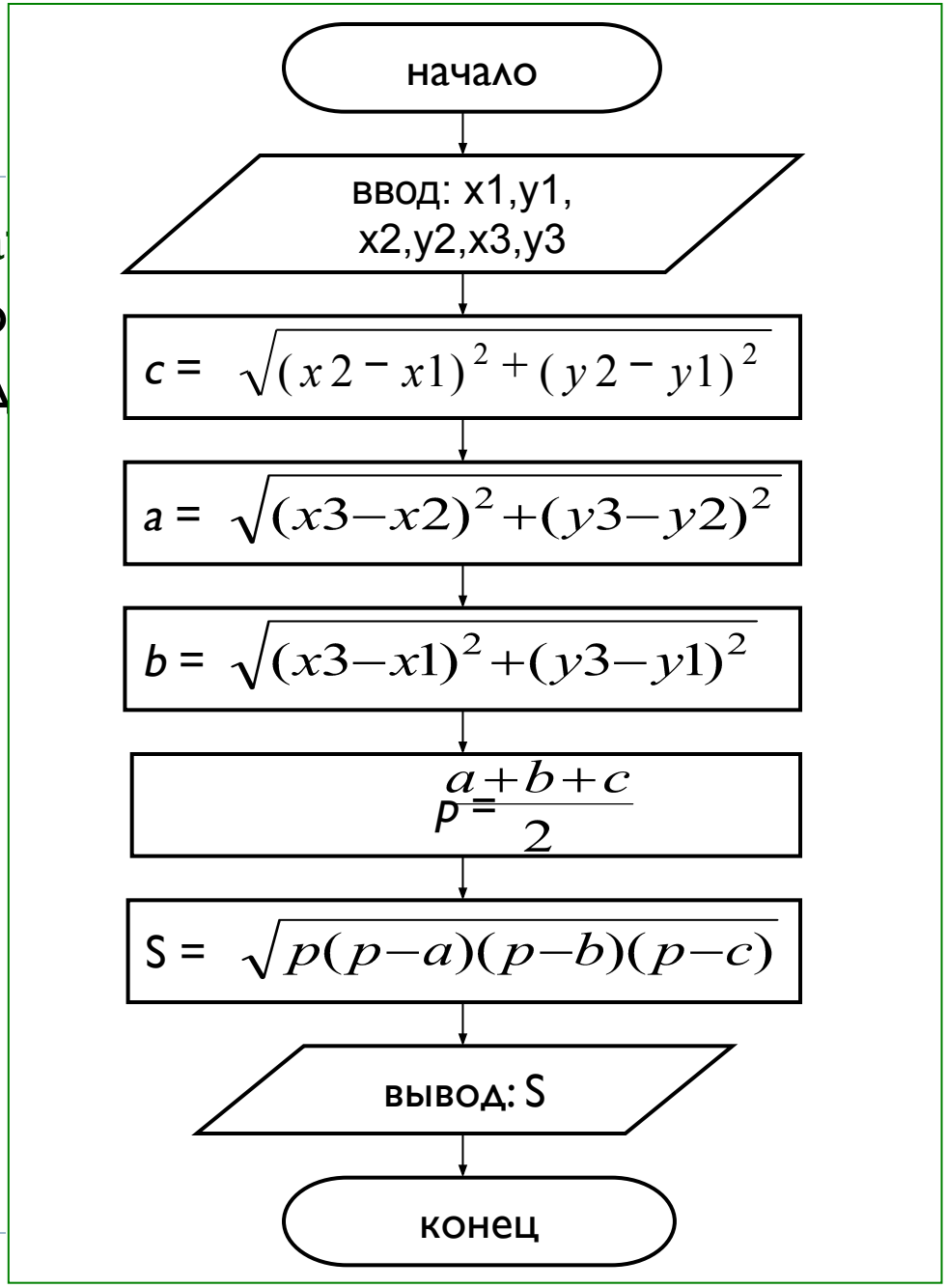
Следование. Примеры

- **Задача:** Даны координаты точек A и B. Найти длину отрезка AB.

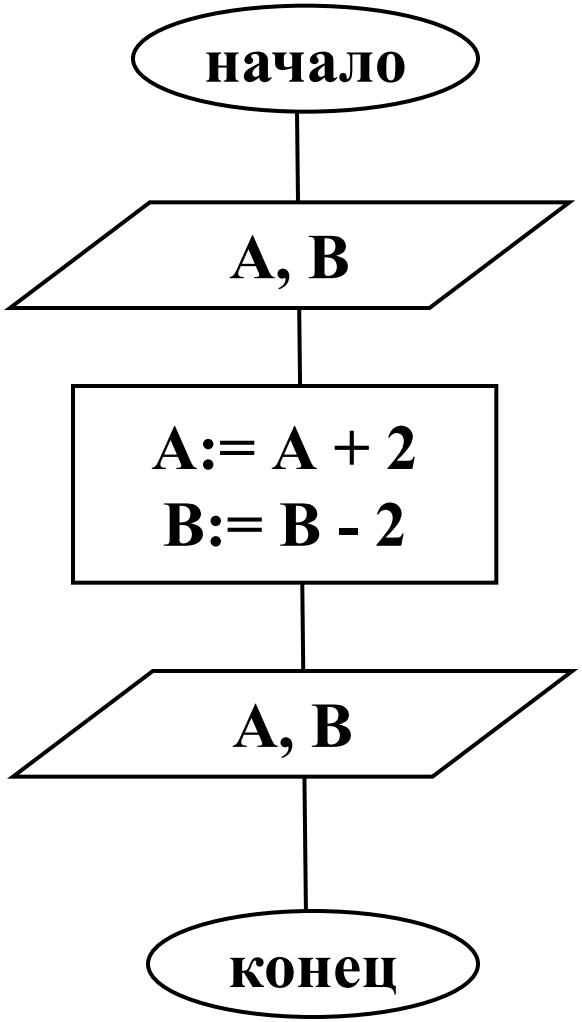
Следование

- **Задача:** Даны координа
  Найти его площадь. Со
  решения поставленной зад





Найти, чему будут равны значения A и B после выполнения алгоритма, если были введены A=5 и B=7.



Ветвление

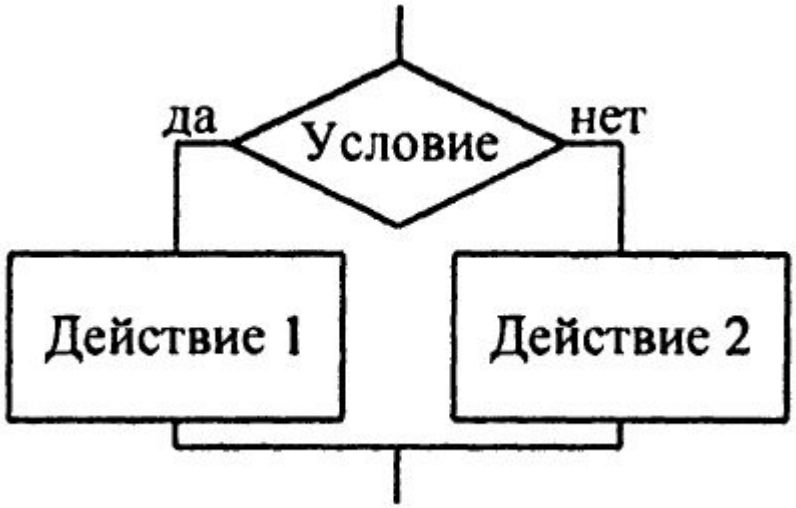
Базовая структура "ветвление". Обеспечивает в зависимости от результата проверки условия (да или нет) выбор одного из альтернативных путей работы алгоритма.

Каждый из путей ведет к общему выходу, так что работа алгоритма будет продолжаться независимо от того, какой путь будет выбран. Структура ветвление существует в четырех основных вариантах:

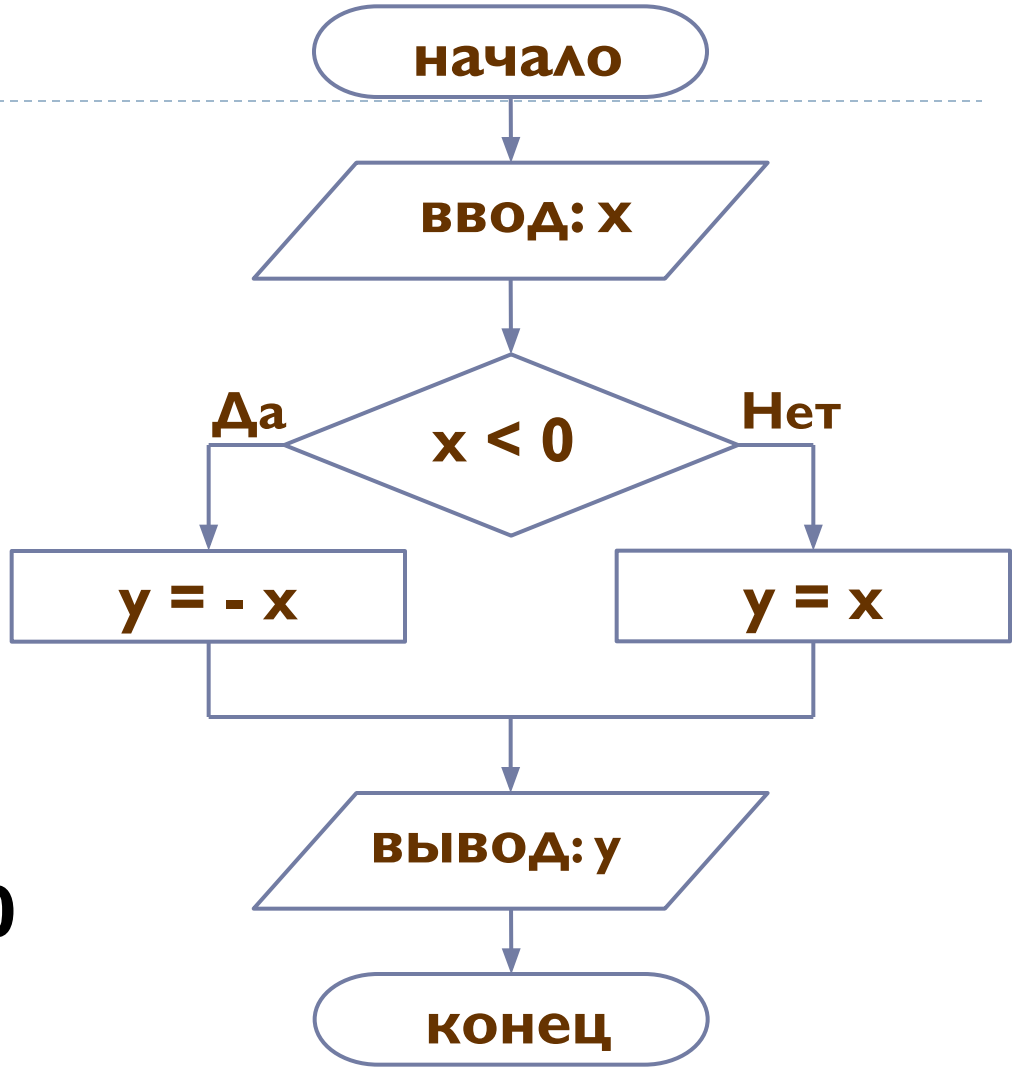
- если—то—иначе;
- если—то;
- выбор;
- выбор—иначе.

Полная команда ветвления

- если—то—иначе;



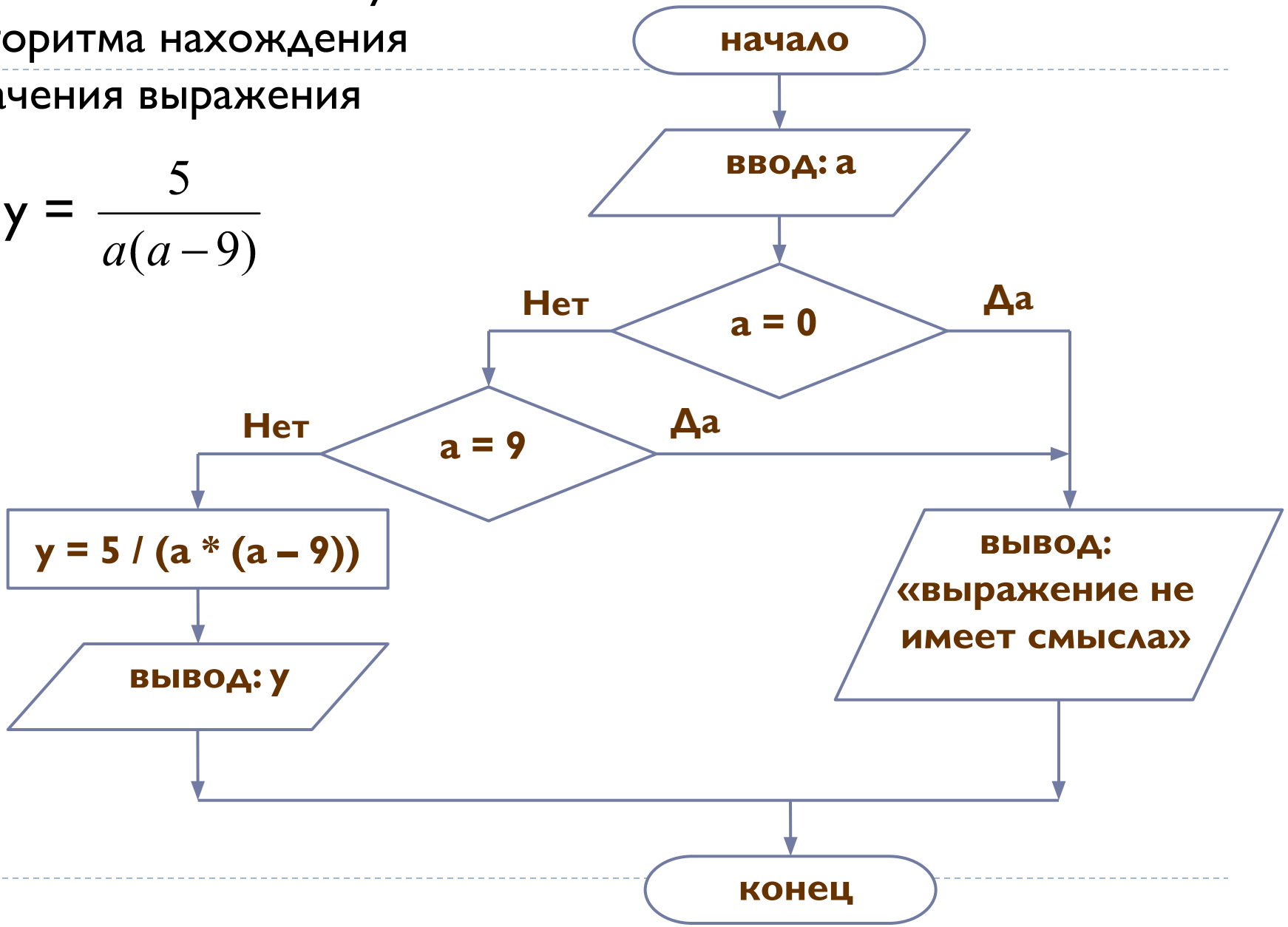
Составить блок-схему алгоритма вычисления абсолютной величины числа



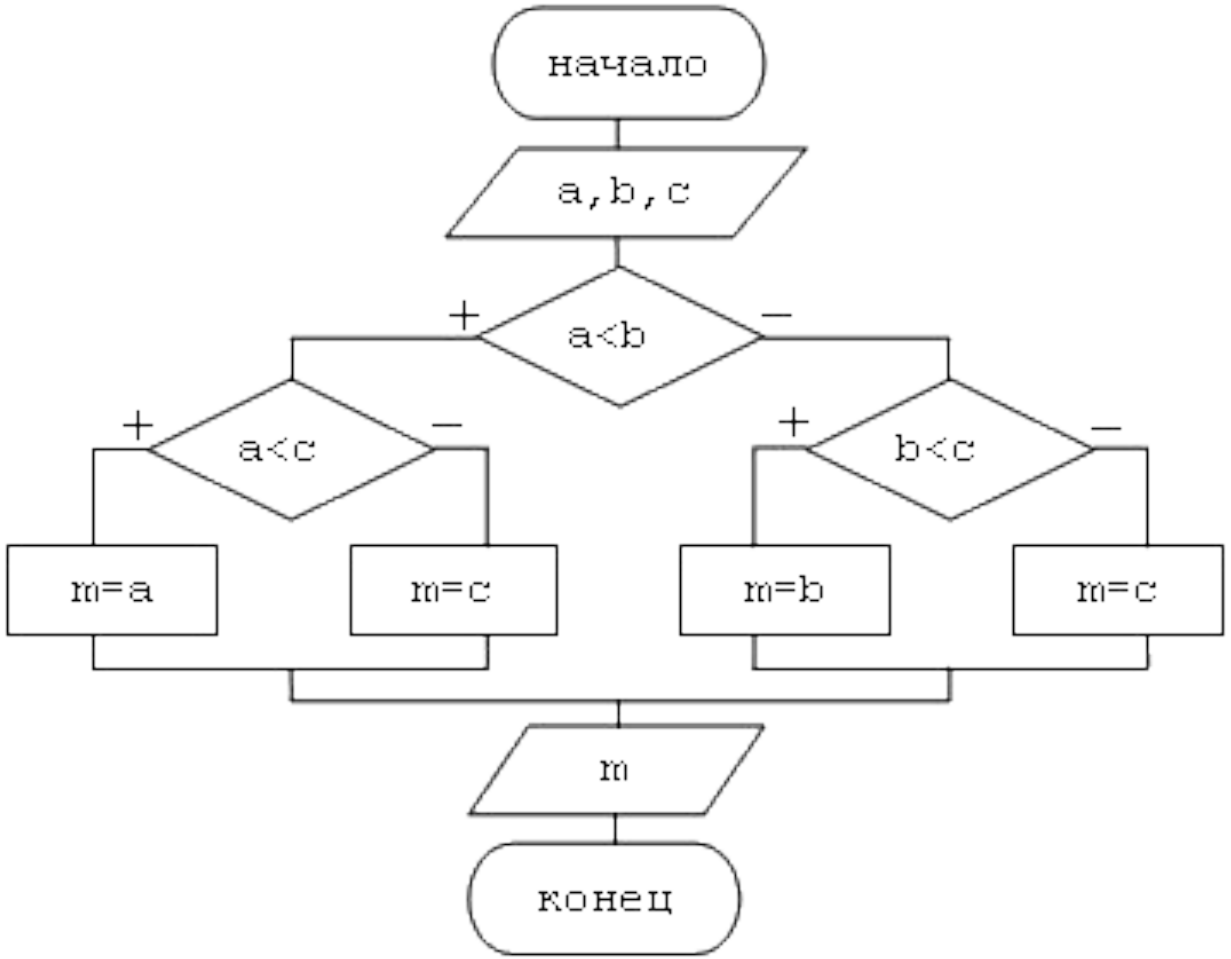
$$y = |x| = \begin{cases} x & \text{при } x \geq 0 \\ -x & \text{при } x < 0 \end{cases}$$

Составьте блок-схему алгоритма нахождения значения выражения

$$y = \frac{5}{a(a-9)}$$



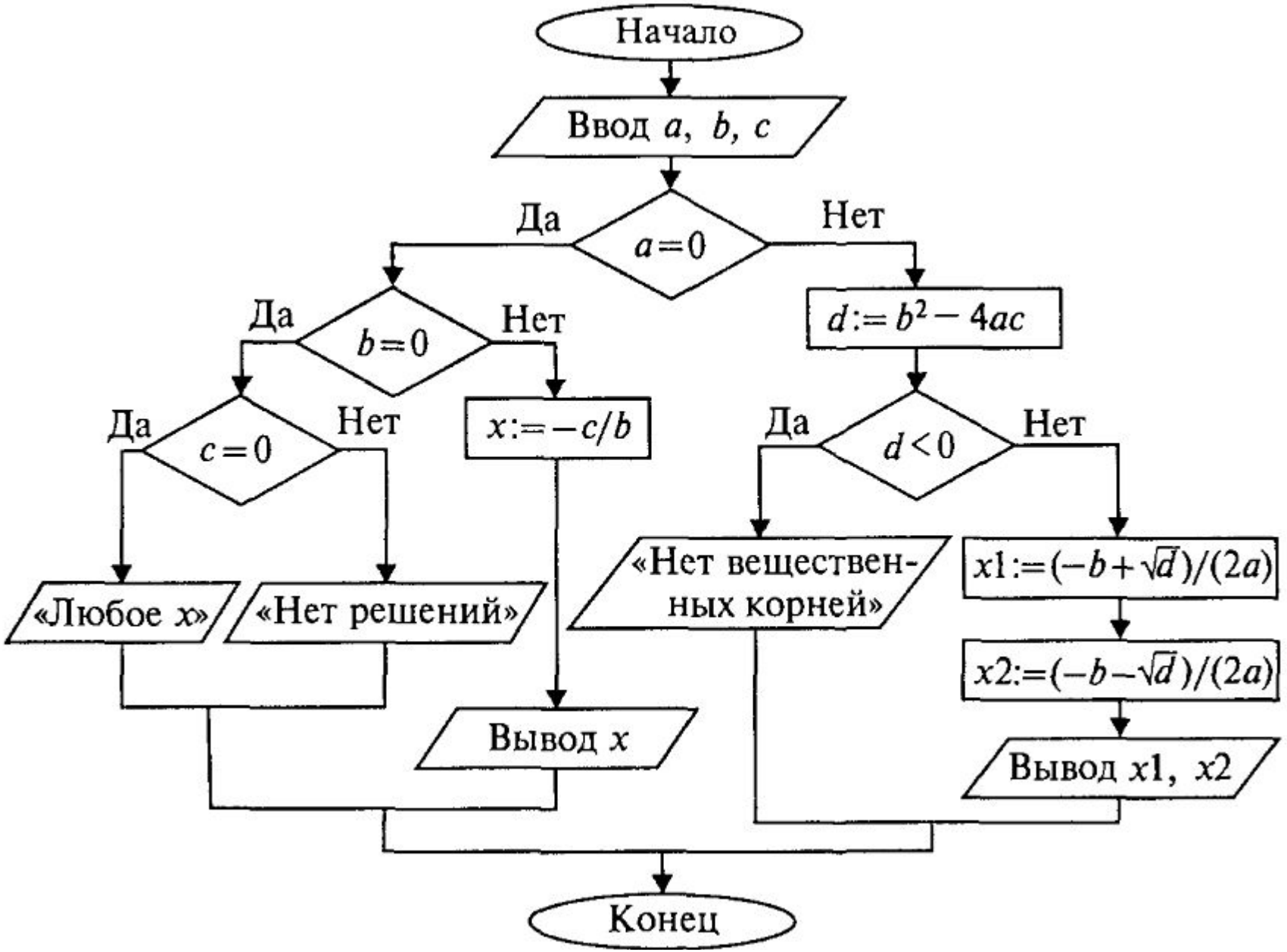
Пример: найти наименьшее из трех чисел.



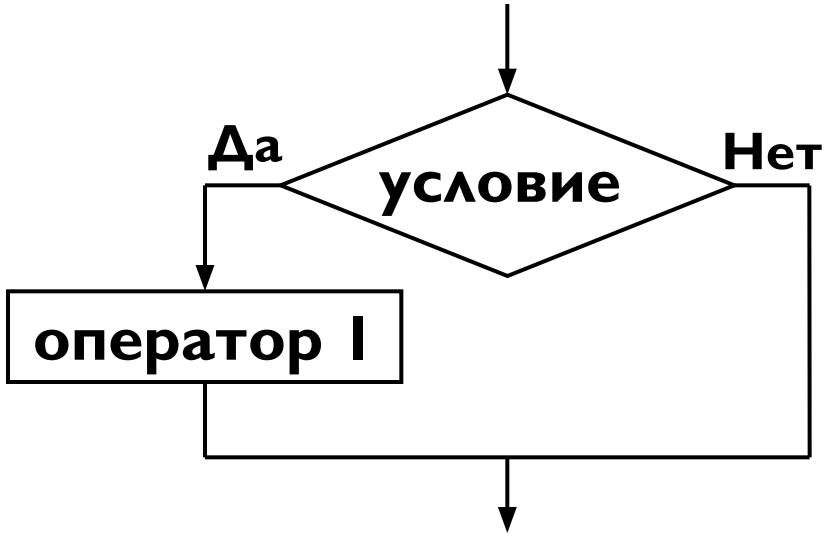
Задача. Составить алгоритм начисления зарплаты согласно следующему правилу:

- если стаж работы сотрудника менее 5 лет, то зарплата 10 тыс.руб.,
- при стаже работы от 5 до 15 лет – 18 тыс. руб.,
- при стаже свыше 15 лет зарплата повышается с каждым годом на 1 тыс. рублей.

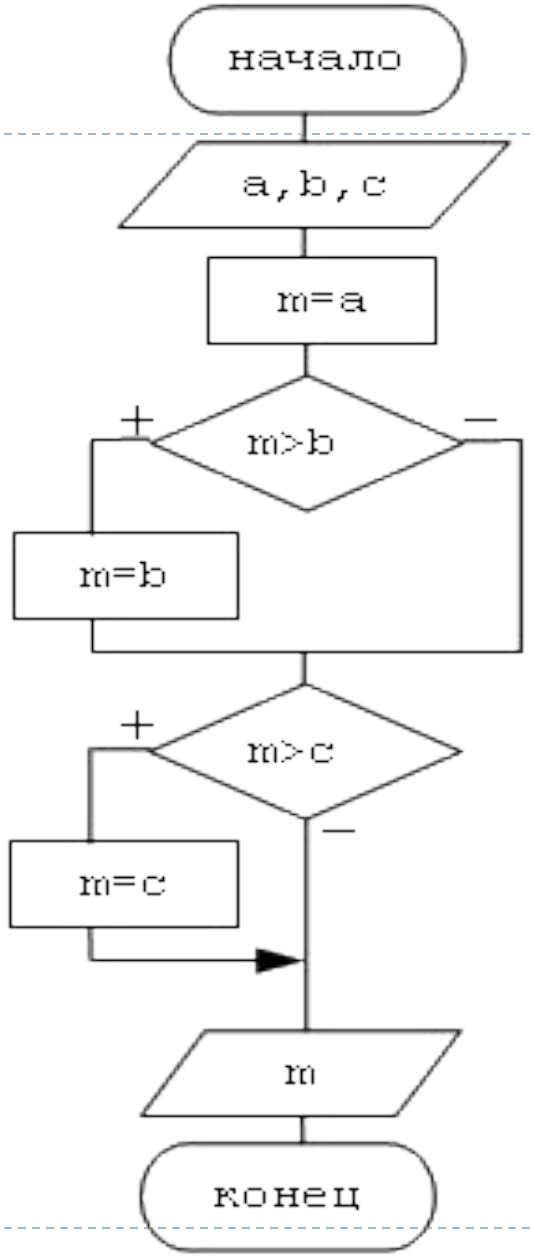
Алгоритм нахождения корней квадратного уравнения



Неполная команда ветвления

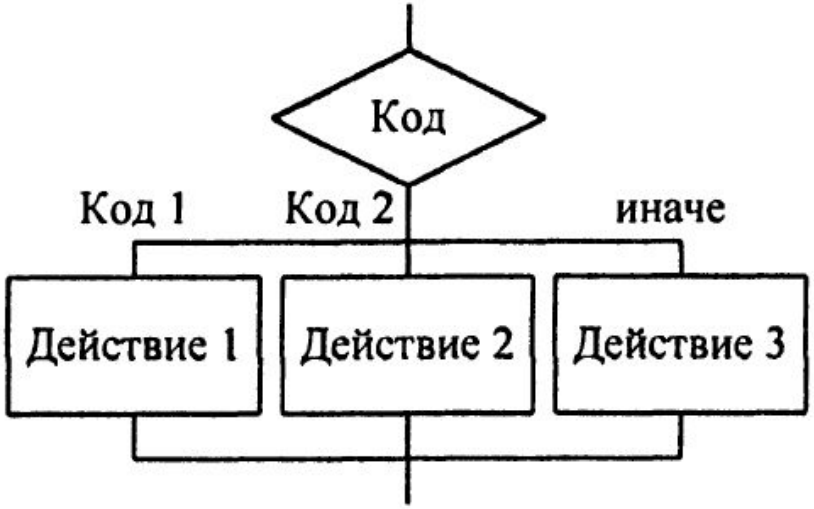


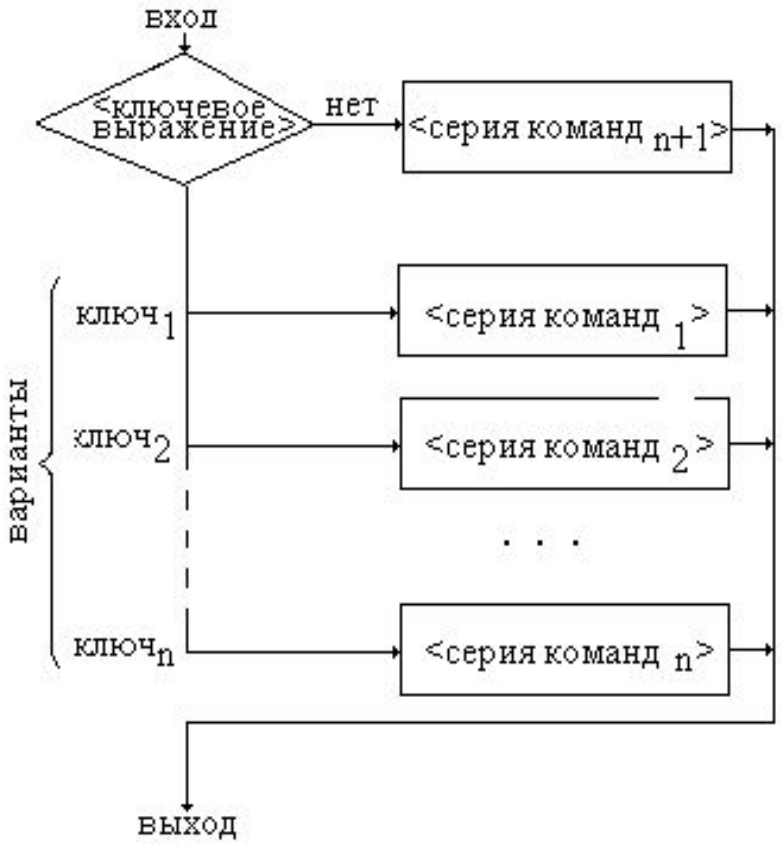
Пример: найти наименьшее из трех чисел.



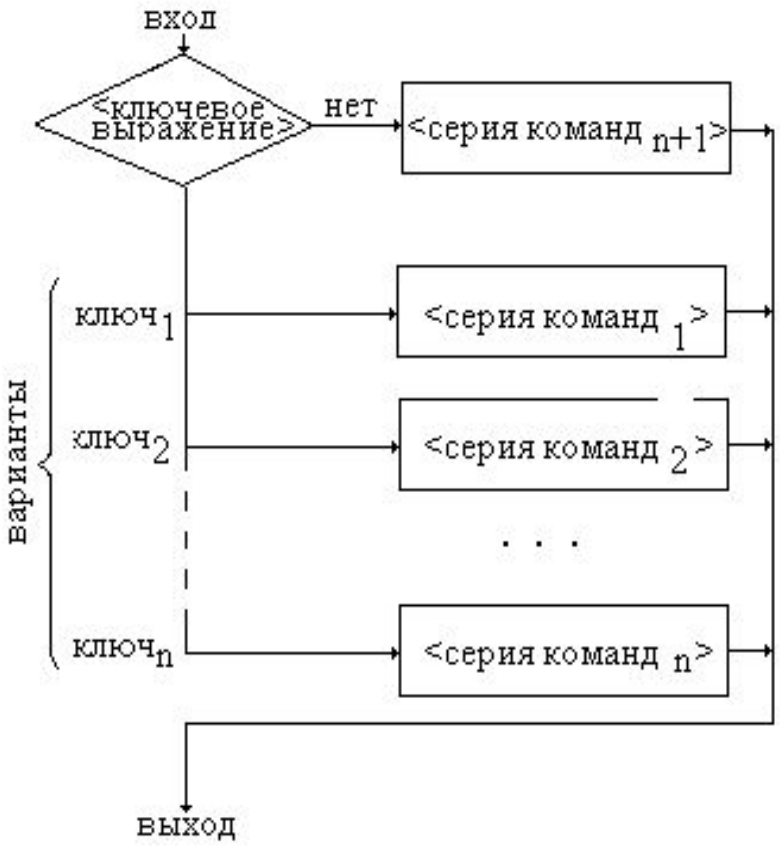
Выбор

Выбор - выбор одного варианта из нескольких в зависимости от значения некоторой величины



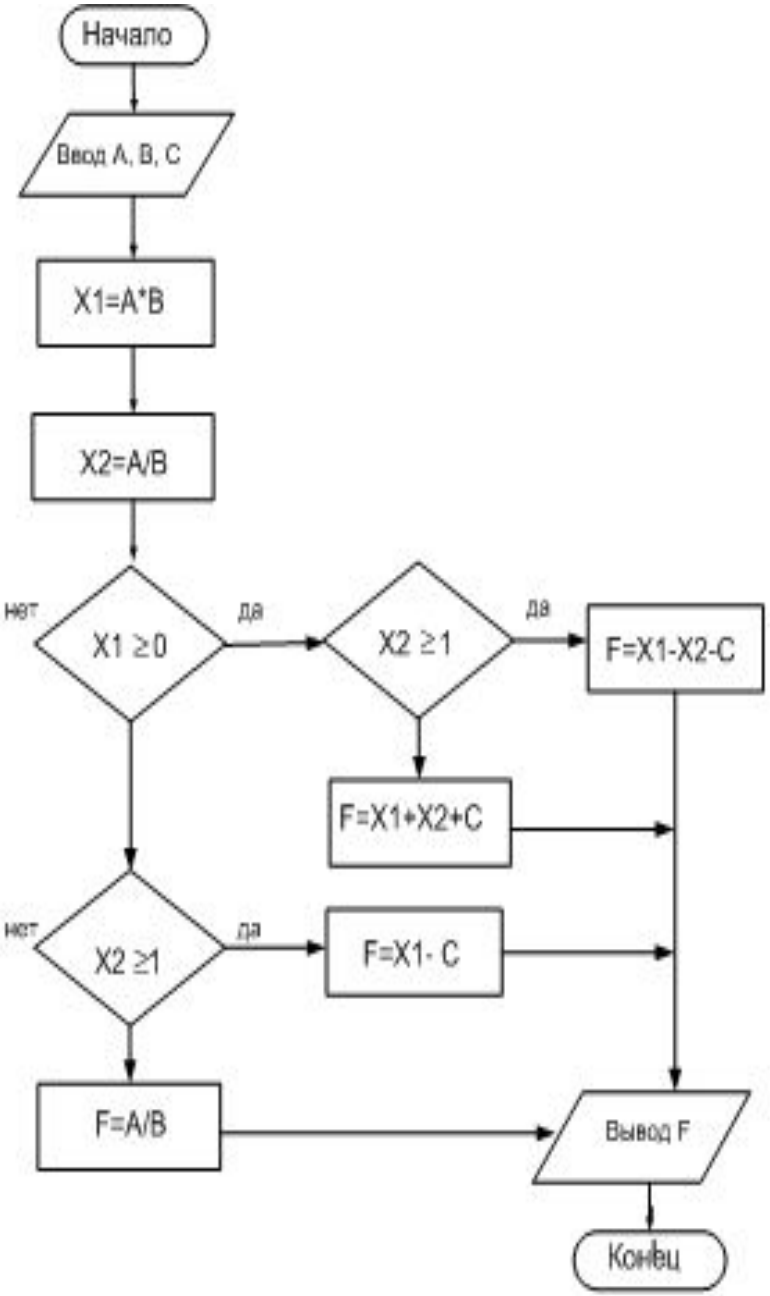


Выбор



Структура выбора используется в алгоритмах, в которых при разных значениях одного и того же выражения (которое называют ключевым выражением или кодом) необходимо выполнять различные варианты действий, каждый из которых имеет свой уникальный ключ варианта.

Выполняется тот вариант, для которого значение ключевого выражения и константа, представляющая ключ варианта, совпадают.


Начало
Ввод A, B, C
X1=A*B
X2=A/B
нет
X1 ≥0
да
X2 ≥1
да
F=X1-X2-C
F=X1+X2+C
нет
X2 ≥1
да
F=X1- C
F=A/B
Вывод F
Конец

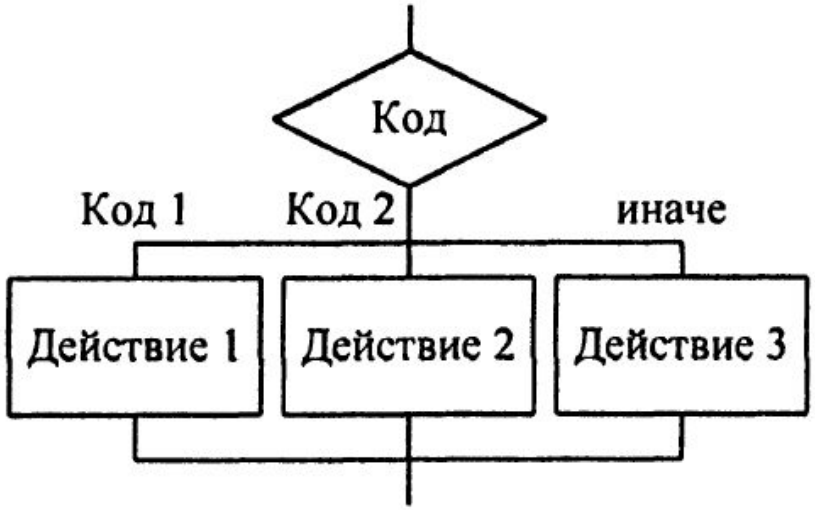
- Составить алгоритм и написать программу, которые запрашивают у пользователя номер дня недели, затем выводят название дня недели или сообщение об ошибке, если введены неверные данные.

```
алг Определение дня недели (арг int nd, рез лит "День недели")
  нач
    ввод nd
    выбор
        при nd = 1  вывод "Понедельник"
        при nd = 2  вывод "Вторник"
        при nd = 3  вывод "Среда"
        при nd = 4  вывод "Четверг"
        при nd = 5  вывод "Пятница"
        при nd = 6  вывод "Суббота"
        при nd = 7  вывод "Воскресенье"
        иначе       вывод "Число должно быть в диапазоне 7"
    все
  кон
```

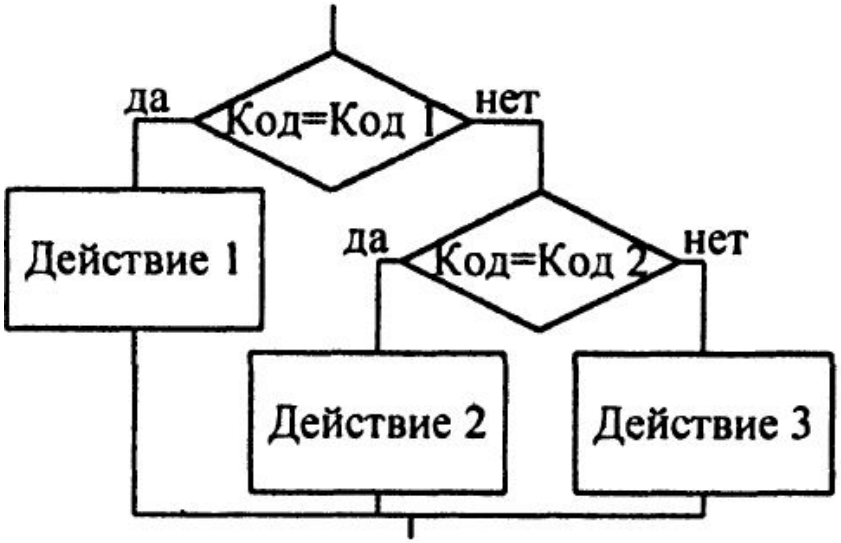
Пример реализации множественного выбора на ЯП Паскаль

```
program year;
var
month: integer; {номер месяца}
begin
    write (’Введите номер месяца:’);
    readln (month);
    writeln (‘Время года:’);
    case month of
        1, 2, 12: writeln (’зима’);
        3..5: writeln (’весна’);
        6..8: writeln (’лето’);
        9..11: writeln (’осень’);
        else writeln (’число должно быть от 1 до 12’);
    end;
end.
```

Выбор



Дополнительная структура **выбор**



Реализация структуры **выбор** через базовую структуру **если-то-иначе**

Повторение

Пример. Составить алгоритм варки картофеля.

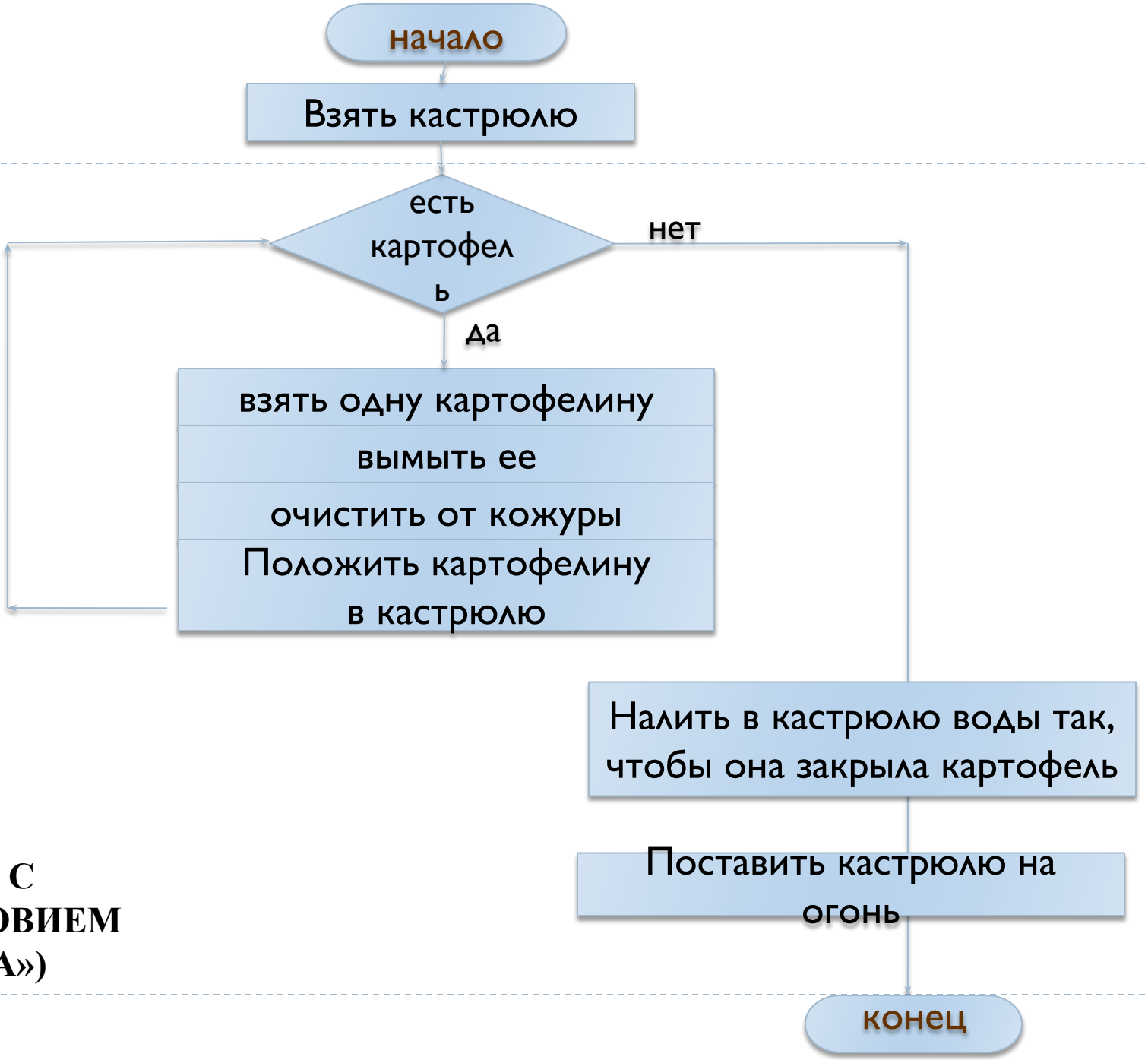
Решение.

1. Взять кастрюлю такого объема, чтобы в нее вместился картофель, который требуется сварить.

2. Пока есть картофель, повторять:
   1) взять одну картофелину;
   2) вымыть ее;
   3) очистить от кожуры;
   4) положить вычищенную картофелину в кастрюлю.

3. Налить в кастрюлю воды так, чтобы она закрыла картофель.

4. Поставить кастрюлю на огонь.



ЦИКЛ С ПРЕДУСЛОВИЕМ («ПОКА»)

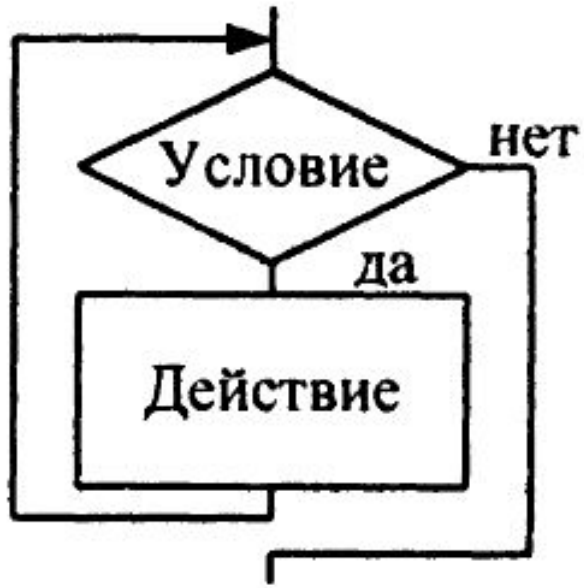
Повторение

Цикл – это многократное повторение некоторой совокупности действий, которая называется телом цикла.

ТИПЫ ЦИКЛИЧЕСКИХ КОНСТРУКЦИЙ

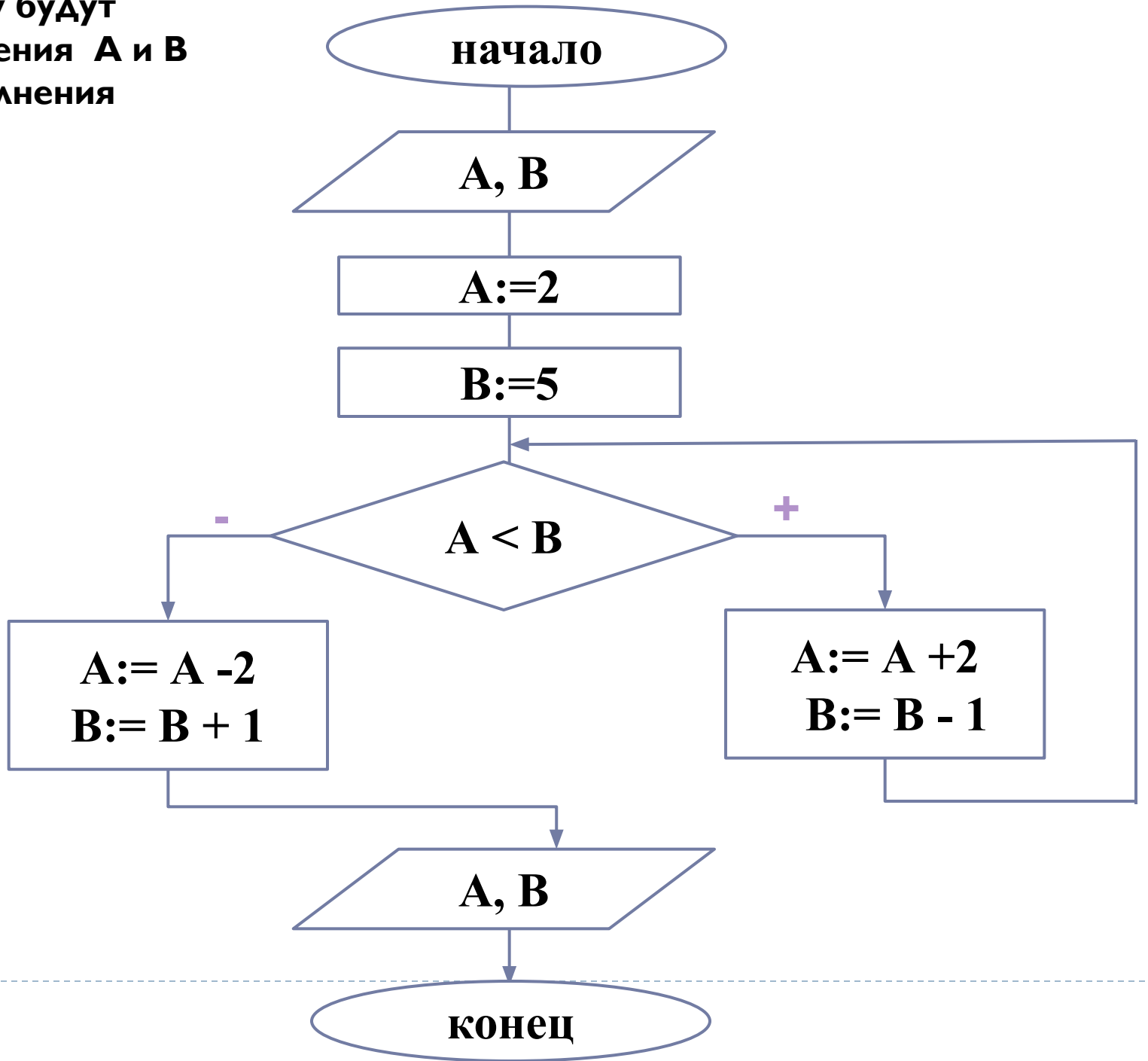
Цикл с предусловием (цикл «пока»)

Цикл «пока» - определяет повторение действий, пока не будет нарушено условие, выполнение которого проверяется в начале цикла



1. Здесь *Действие* называют *телом цикла.*
2. Цикл работает до тех пор, пока условие ИСТИННО; как только условие становится ложным, цикл заканчивает работу. В частности, этот цикл может не выполниться ни разу, если при первой же проверке условие ложно.
3. В теле цикла обязательно должно быть действие, которое влияет на изменение условия. В противном случае может произойти «зацикливание» (бесконечный цикл).

Найти, чему будут равны значения A и B после выполнения алгоритма.

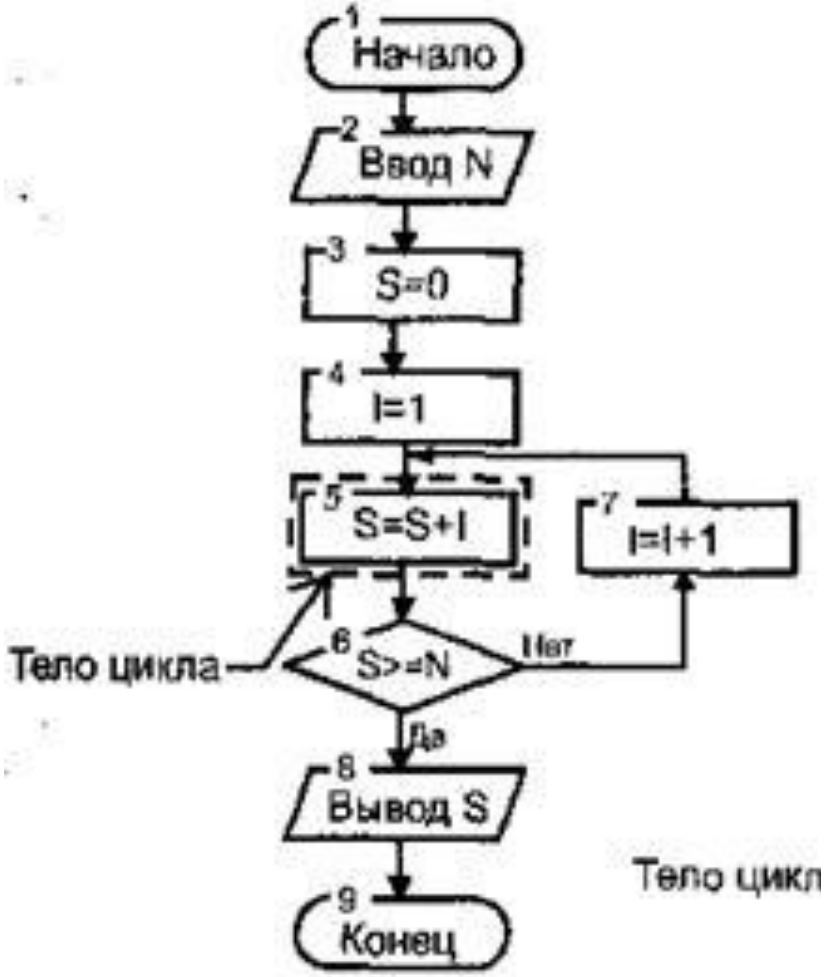


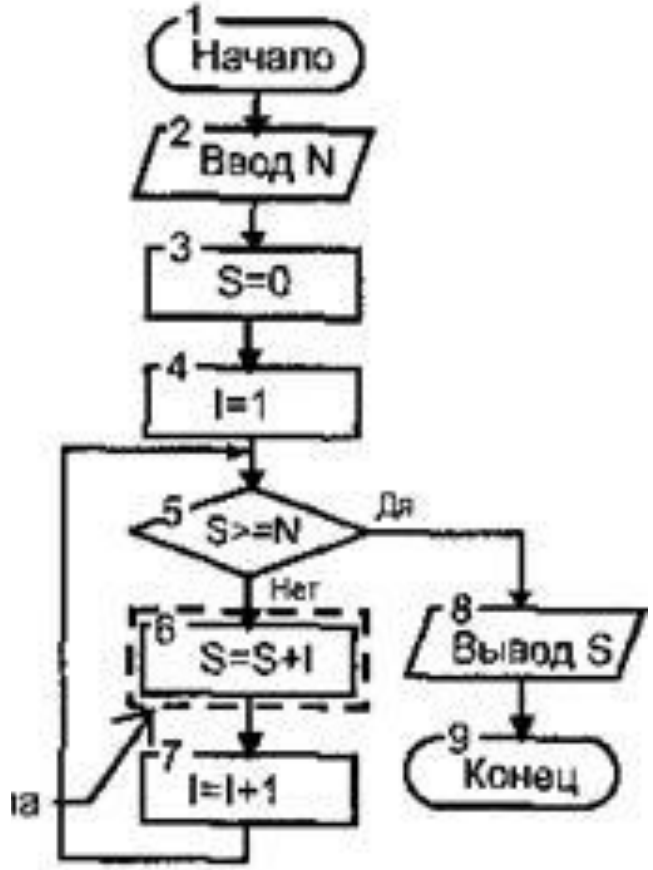
Цикл с постусловием (Цикл «до»)

Цикл «до» - повторение некоторых действий до выполнения заданного условия, проверка которого осуществляется после выполнения действий в цикле

1. Здесь *Действие* называют *телом цикла.*
2. Цикл работает до тех пор, пока условие ЛОЖНО; как только условие становится истинным, цикл заканчивает работу. Этот цикл выполняется как минимум один раз, так как условие стоит после тела цикла.
3. В теле цикла обязательно должно быть действие, которое влияет на изменение условия. В противном случае может произойти «зацикливание» (бесконечный цикл).

Пример алгоритма нахождения суммы первых членов натурального ряда. Вычисление суммы прекратить, как только ее значение будет равно или превысит заданное N.



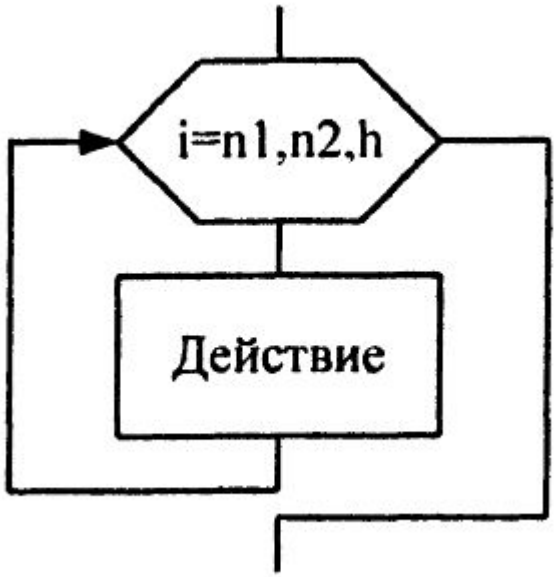


Цикл с постусловием;

Цикл с предусловием;

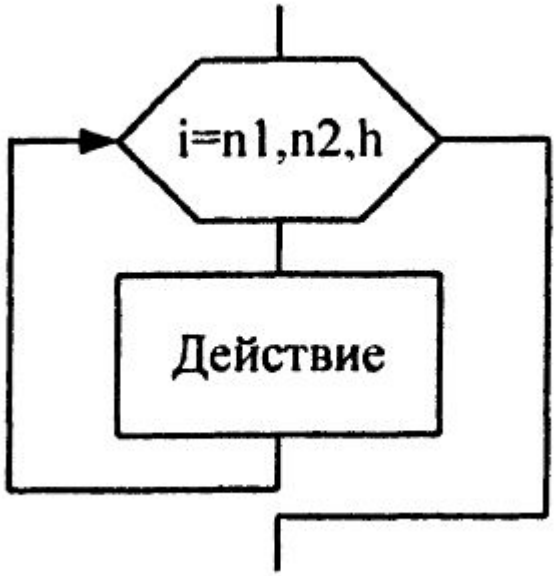
Цикл с параметром (цикл «для»)

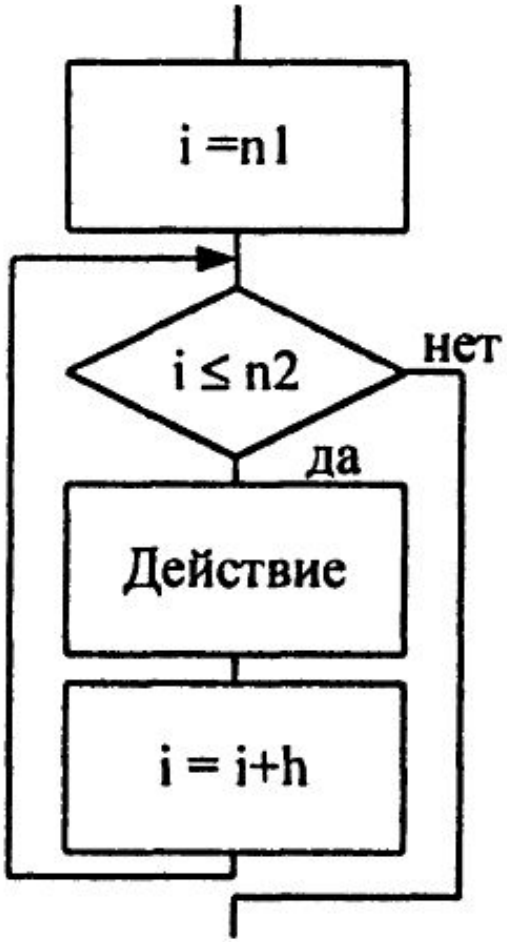
цикл с заданным числом повторений (счетный цикл) – повторение некоторых действий указанное число раз



1. Здесь переменную i называют счетчиком цикла, n1 – начальное значение счетчика, n2 – конечное значение счетчика.
2. Переменная i последовательно принимает все значения от n1 до n2, автоматически увеличиваясь на h – шаг цикла.
3. Действие цикла заканчивается как только i становится больше n2.
4. Этот цикл используют в задачах, в которых заранее известно количество повторений.

Реализация структуры **цикл с параметром** через базовую структуру **цикл пока**





Цикл с параметром (цикл «для»)

Пример. Составить алгоритм и написать программу, которые вычисляют сумму первых n целых положительных целых чисел. Количество суммируемых чисел должно вводится во время работы программы.

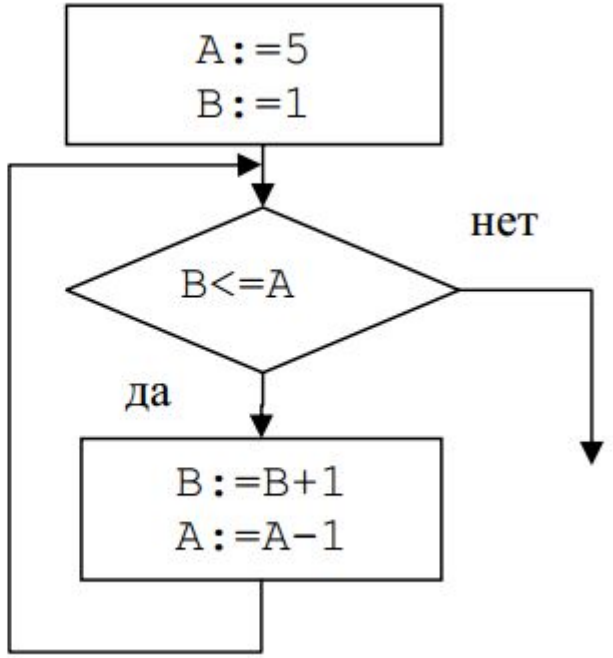
алг Сумма первых n целых положительных чисел (**арг** **цел** n, **рез** **цел** summ)
нач
цел i
ввод n
нц для i от 1 до n
summ = summ + i
кц
вывод summ
кон

Трассировочная таблица

- Трассировочные таблицы используются для анализа свойств алгоритма и проверки его соответствия решаемой задаче. В такой таблице для конкретных значений исходных данных по шагам прослеживается изменение переменных, входящих в алгоритм.
- В заголовке трассировочной таблицы помещают имена всех переменных, используемых в алгоритме. В отдельном столбце записывают команды и логические выражения (условия), которые выполняются.
- Каждая строка таблицы соответствует одному шагу алгоритма, при котором изменяются значения переменных или выражений.

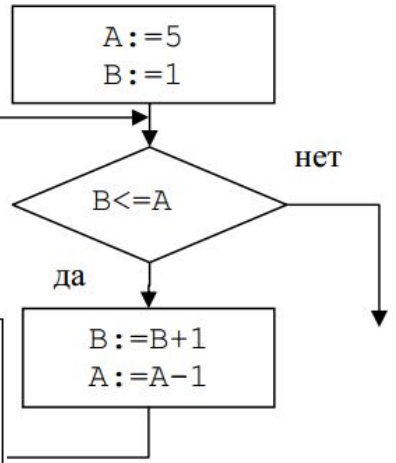
- Пример: Для фрагмента алгоритма составить трассировочную таблицу.

```
A:=5
B:=1
Пока B<=A
нц
      B:=B+1
      A:=A-1
кц
```



Определить,

- сколько раз выполняется тело цикла,
- значения переменных A и B после его завершения.



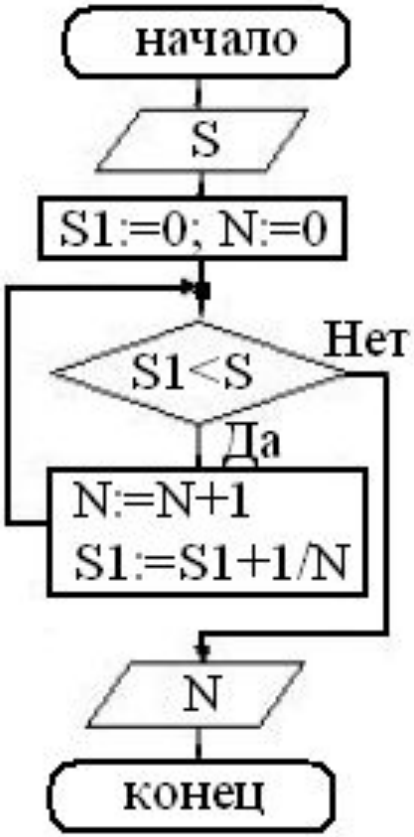
№	Команда или условие (логическое выражение)	Вычисление правой части команды присваивания или условия (логического выражения)	A	B
1	A:=5	5	5	
2	B:=1	1		1
3	B<=A	(1<=5)=да		
4	B:=B+1	1+1=2		2
5	A:=A-1	5-1=4	4	
6	B<=A	(2<=4)=да		
7	B:=B+1	2+1=3		3
8	A:=A-1	4-1=3	3	
9	B<=A	(3<=3)=да		
10	B:=B+1	3+1=4		4
11	A:=A-1	3-1=2	2	
12	B<=A	(4<=2)=нет		
Результат			**2**	**4**

Из таблицы видно, что тело цикла выполнилось трижды (строки 4-5, 7-8, 10-11), переменные приняли значения A=2, B=4.

Задачи

Пример Какое значение будет иметь N на выходе, если:

а) S=1,1; б) S=2,09?



а) S=1,1

Шаг	1	2	3
Значение S1	0	1	1,5
Значение N	0	1	2
Тело цикла	0 < 1,1 (Да)	1 < 1,1 (Да)	1,5 < 1,1 (Нет)
Результат выполнения	N=0+1=1 S1=0+1/1=1	N=1+1=2 S1=1+1/2=1,5	N = 2
Вывод значения N			2

Задачи

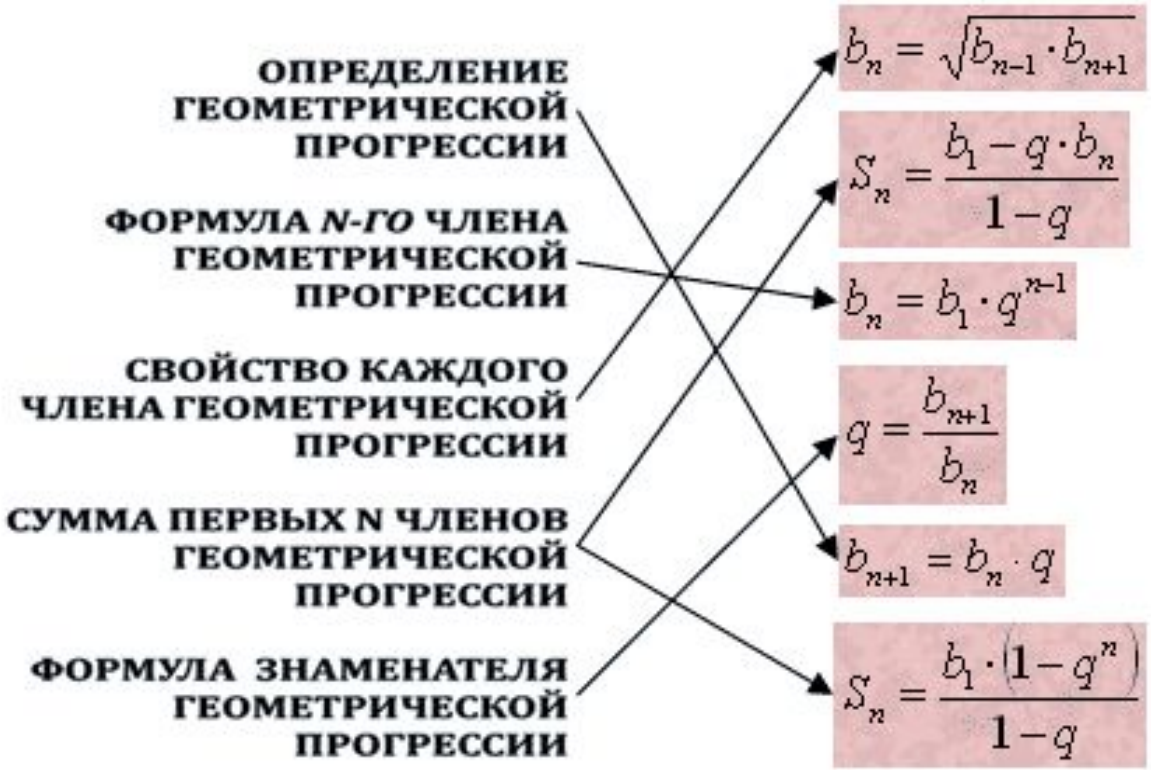
- б) S=2,09

Шаг	1	2	3	4	5	6
Значение S1	0	1	1,5	1,8333	2,0833	2,2833
Значение N	0	1	2	3	4	5
условие	0 < 2,09 (Да)	1 < 2,09 (Да)	1,5<2,09 (Да)	1,83333<2,09 (Да)	2,0833<2,09 (Да)	2,2833 < 2,09 (Нет)
Результат выполне-ния	N=0+1=1 S1=0+1/1=1	N=1+1=2 S1=1+1/2 = 1,5	N=2+1=3 S1=1,5+1/3= 1,8333	N=3+1=4 S1=1,8333+1/4 = 2,0833	N=4+1=5 S1=2,0833+1/5= 2,2833	N = 5
Вывод значения N						5

Задачи

Построить алгоритм нахождения N первых членов геометрической прогрессии по известному первому члену и знаменателю.

$$b_1,\ b_2 = b_1 q,\ b_3 = b_2 q,\ \ldots,\ b_n = b_{n-1} q$$



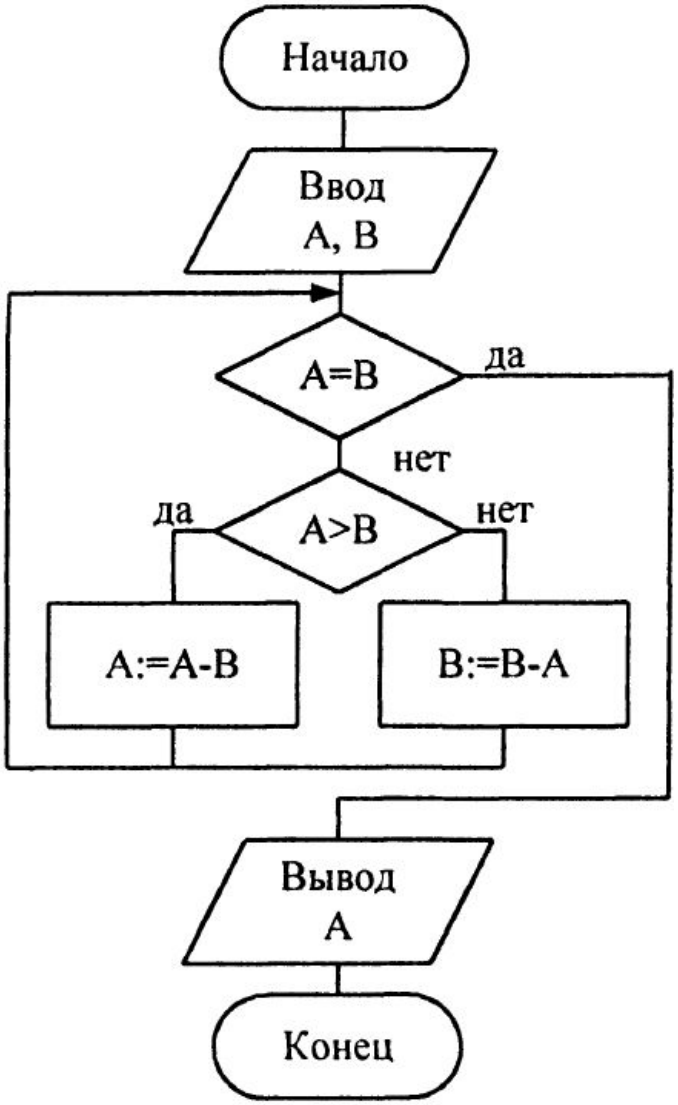
Алгоритм Евклида

- **Задача.** Определение наибольшего общего делителя двух натуральных чисел.
- Самым простым способом нахождения НОД является так называемый алгоритм Евклида. Суть этого метода заключается в последовательной замене большего из чисел на разность большего и меньшего. Вычисления заканчиваются, когда числа становятся равны.

A		B
225		125
225-125 = 100		125
100		125-100 = 25
100-25 = 75		25
75-25 = 50		25
50-25 = **25**	=	**25**

A		B
13		4
13-4=9		4
9-4=5		4
5-4=1		4
1		4-1 = 3
1		3-1 = 2
1	=	2-1 = **1**

Алгоритм Евклида



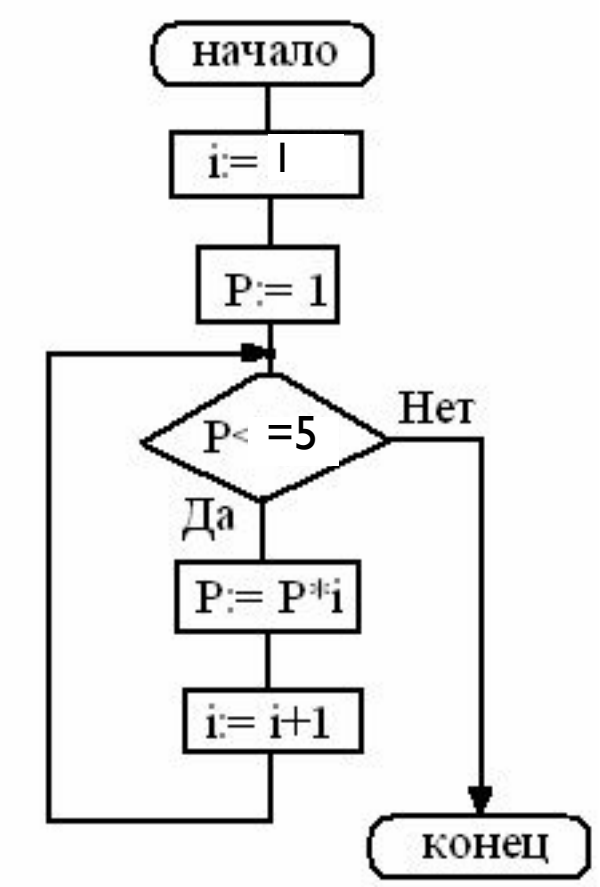
да

да

Шаг	A	B	A= B	A>B	A:=A-B	B:=B-A	Вывод A

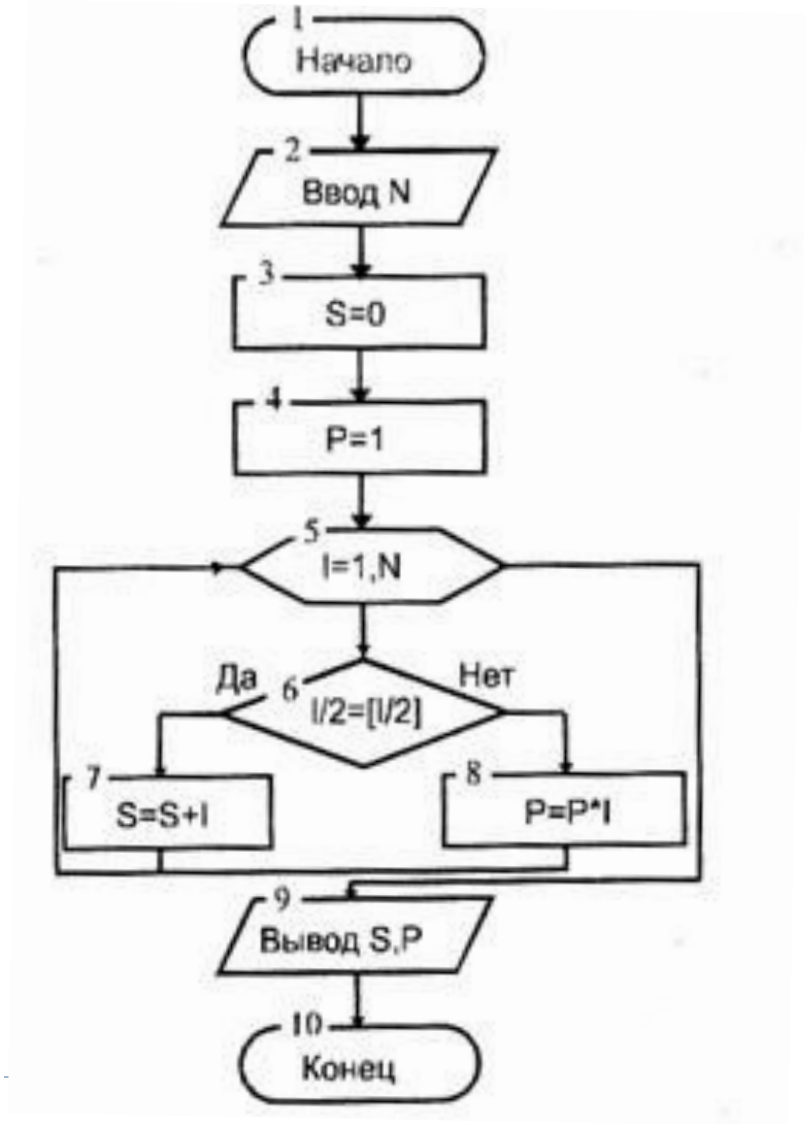
Задачи

Найти значение переменных P и i после исполнения алгоритма.



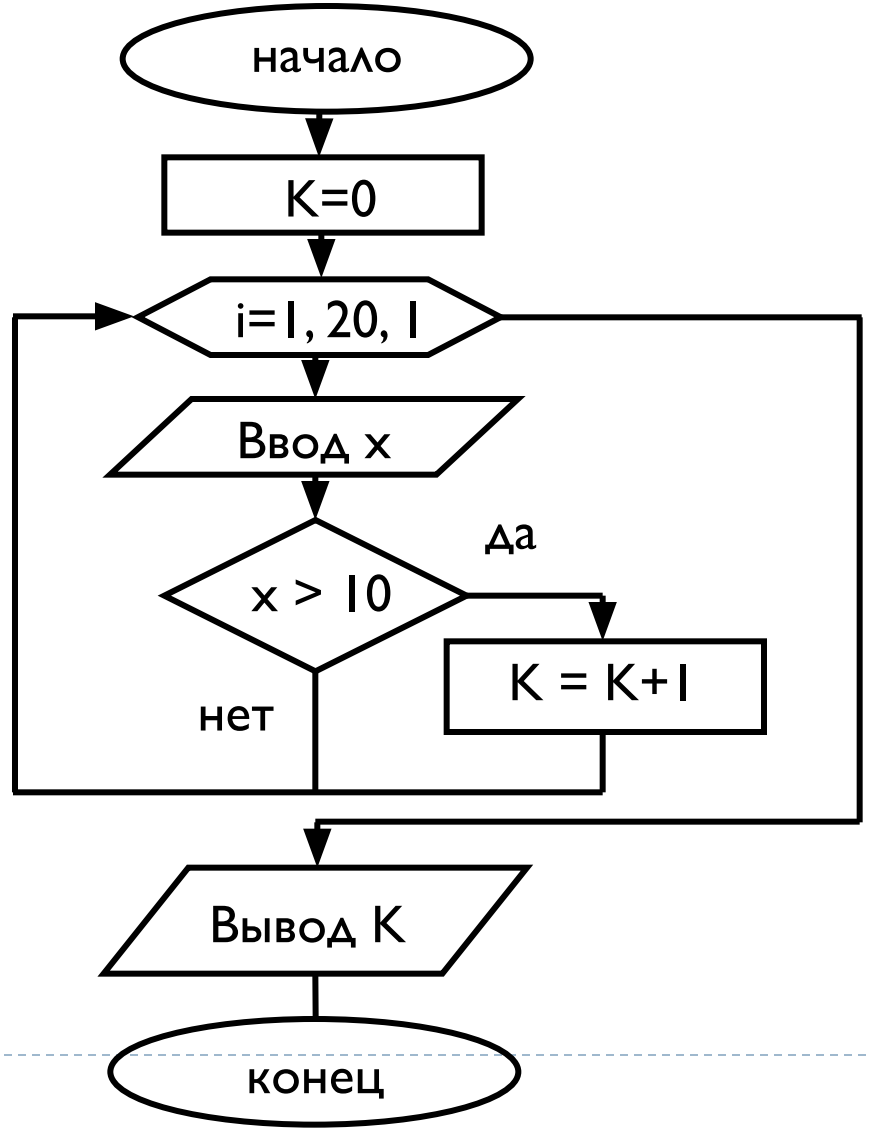
Задачи

- Дан натуральный ряд чисел от 1 до N. Вычислить сумму четных и произведение нечетных чисел этого ряда.



Задачи

Пример. Задано 20 чисел. Сколько среди них чисел, больших 10?



Задачи

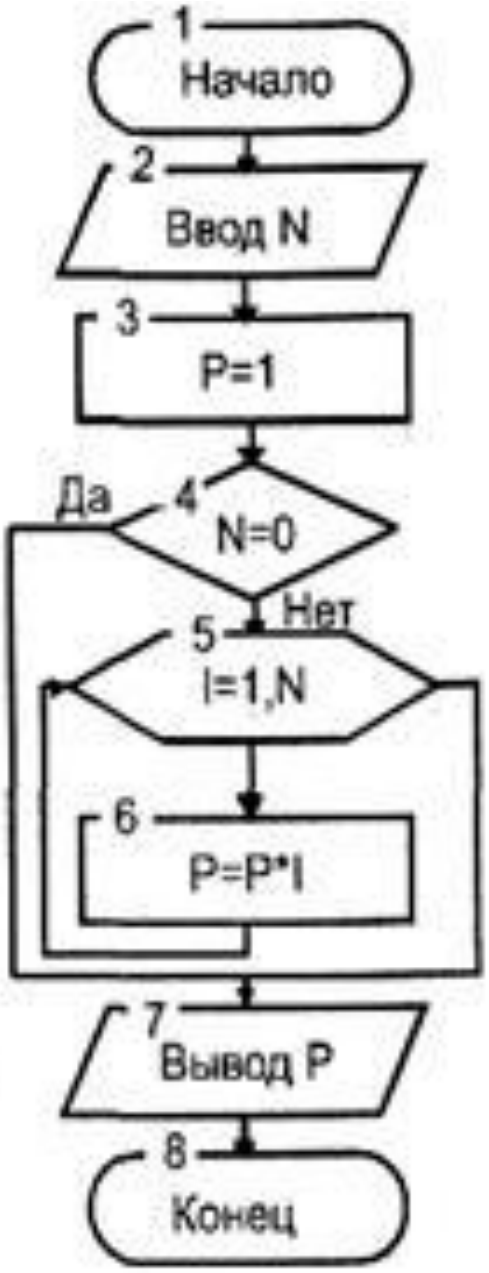
- **Пример**. Составить алгоритм и написать программу, которые вычисляют среднее арифметическое последовательности чисел, вводимых с клавиатуры. Найти минимальное и максимальное число последовательности. Количество чисел последовательности должно задаваться с клавиатуры до ввода последовательности.

Нахождение фактори

Задача. Построить блок-схему факториала числа.

Опр. Факториалом числа n назы натуральных чисел до n включитель

$$n! = \begin{cases} 1, & \text{если } n = 0; \\ 1 \cdot 2 \ldots n, & \text{если } n \geq 1. \end{cases}$$



Рекурсия

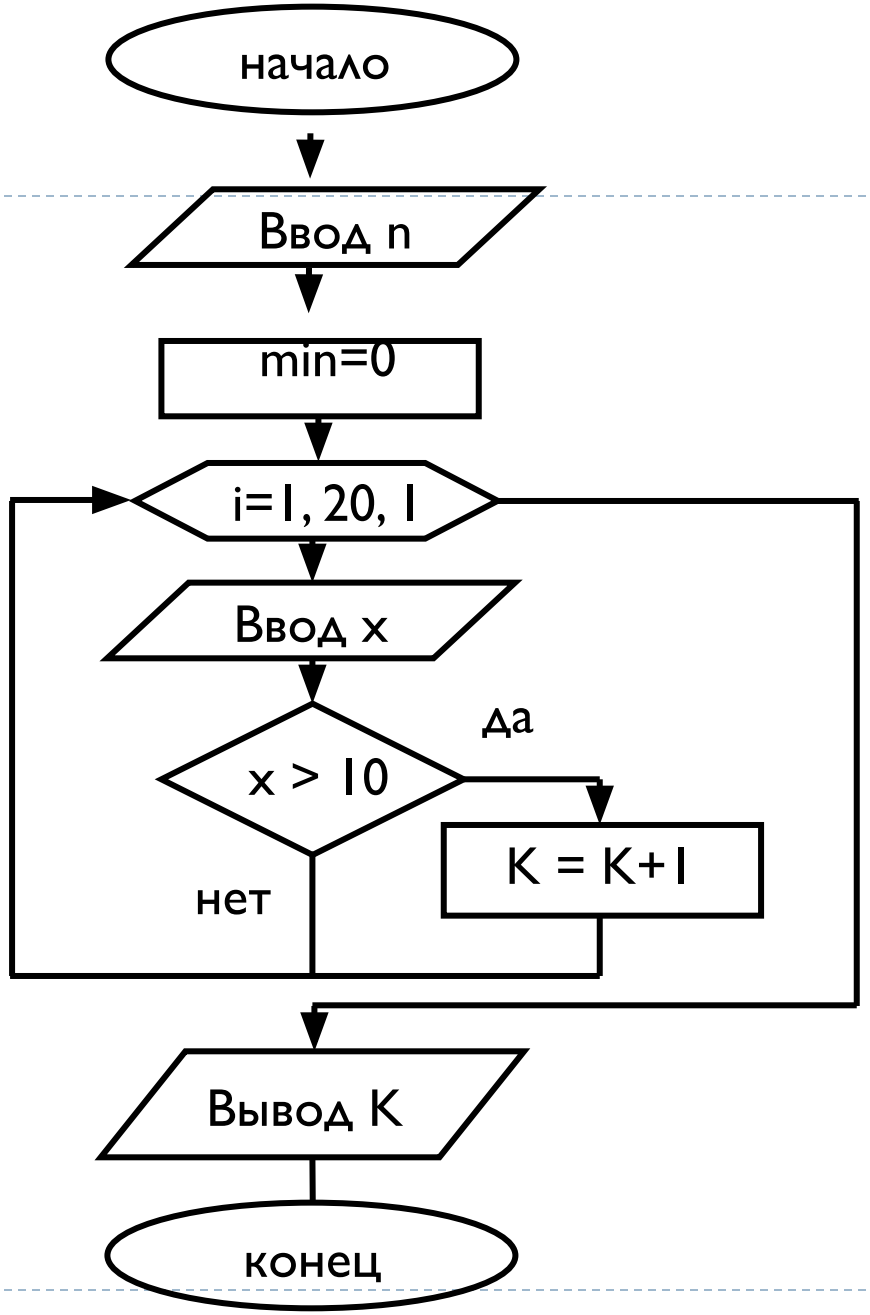
- ***Рекурсия*** – это метод определения или выражения функции, процедуры, языковой конструкции или решения задачи посредством той же функции, процедуры и т. д.

- *Факториальная функция* определяется рекурсивно следующим образом:
  - 0! = 1, нерекурсивно определенное начальное значение
  - $n! = n * (n-1)!$, если $n > 0$
- Для каждой рекурсивной функции нужно хотя бы одно начальное значение, в противном случае ее нельзя вычислить в явном виде.

```
Нач
если n=0 то  Fakt:=1
иначе  Fakt:=n*Fakt(n-1);
кон
```

```
begin
if (n=0) then
    Fakt:=1
else
    Fakt:=n*Fakt(n-1);
end;
```

- какую работу надо выполнить, чтобы вычислить n?. Для этого необходимо произвести *рекурсивное обращение* и вычислить (*n*-1)! Это в свою очередь требует другого рекурсивного обращения для вычисления (*n*-2)! и т.д. Таким образом, для того чтобы вычислить *n*! нужно произвести *n* рекурсивных обращений, последнее из которых выполняется для 0! = 1. Принято говорить, что *глубина рекурсии*, требуемая для вычисления *n*!, равна *n*.


начало
Ввод n
min=0
i=1, 20, 1
Ввод x
да
x > 10
K = K+1
нет
Вывод K
конец