

Алгоритмы основные понятия, свойства и виды.

- 1. История возникновения понятия «алгоритм»**
- 2. Цели, задачи и практическое применение теории алгоритмов**
- 3. Формализация понятия и свойства алгоритмов**
- 4. Виды алгоритмов**
 - 4.1. Линейные алгоритмы**
 - 4.2. Разветвляющиеся алгоритмы**
 - 4.3. Циклические алгоритмы**

1. История возникновения понятия «алгоритм»

- «Алгоритм – это конечный набор правил, который определяет последовательность операций для решения конкретного множества задач и обладает пятью важными чертами: конечность, определённость, ввод, вывод, эффективность» (Д. Э. Кнут).
- «Алгоритм – это всякая система вычислений, выполняемых по строго определённым правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи» (А. Колмогоров).
- «Алгоритм – это точное предписание, определяющее вычислительный процесс, идущий от варьируемых исходных данных к искомому результату» (А. Марков).
- «Алгоритм – точное предписание о выполнении в определённом порядке некоторой системы операций, ведущих к решению всех задач данного типа» (Философский словарь под ред. М. М. Розенталя).

2. Цели, задачи и практическое применение теории

алгоритмов:

- формализация понятия «алгоритм» и исследование формальных алгоритмических систем;
- формальное доказательство алгоритмической неразрешимости задач;
- классификация задач, определение и исследование сложностных классов;
- получение явных функций трудоемкости в целях сравнительного анализа алгоритмов;
- разработка критериев сравнительной оценки качества алгоритмов.

ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ РЕЗУЛЬТАТОВ ТЕОРИИ АЛГОРИТМОВ

Теоретический аспект

При исследовании некоторой задачи позволяет ответить на вопросы:
1) является задача в принципе алгоритмически разрешимой?
При утвердительном ответе, анализируют временные затраты на получения точного решения для больших размерностей исходных данных.

Практический аспект

Методы и методики теории алгоритмов позволяют осуществить:

- рациональный выбор из известного множества алгоритмов решения задачи с учетом особенностей их применения (например, при ограничениях на размерность исходных данных или объема дополнительной памяти);
- получение временных оценок решения сложных задач;
- разработку и совершенствование эффективных алгоритмов решения задач в области обработки информации на основе практического анализа.

3. Формализация понятия и свойства алгоритмов

Формализация понятия алгоритма

Определение 1. Алгоритм – это заданное на некотором языке конечное предписание, задающее конечную последовательность выполнимых элементарных операций для решения задачи, общее для класса возможных исходных данных.

Определение 2. Алгоритм – точный набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное время.

Свойства алгоритмов

Дискретность — алгоритм должен представлять процесс решения задачи как последовательное выполнение некоторых простых шагов. Для выполнения каждого шага алгоритма требуется конечный отрезок времени, то есть преобразование исходных данных в результат осуществляется во времени дискретно.

Детерминированность — определённость. В каждый момент времени следующий шаг работы однозначно определяется состоянием системы. Таким образом, алгоритм выдаёт один и тот же результат (ответ) для одних и тех же исходных данных.

Понятность — алгоритм для исполнителя должен включать только те команды, которые ему (исполнителю) доступны, которые входят в его систему команд.

Завершаемость (конечность) — при корректно заданных исходных данных алгоритм должен завершать работу и выдавать результат за конечное число шагов. Вероятностный алгоритм может и никогда не выдать результат, но вероятность этого равна 0.

Массовость — универсальность. Алгоритм должен быть применим к разным наборам исходных данных.

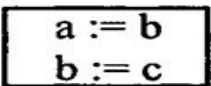
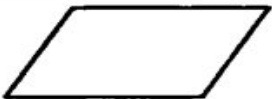
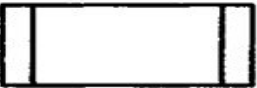
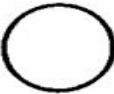

Формы записи алгоритмов. Алгоритмы можно записывать по-разному. Форма записи, состав и количество операций алгоритма зависят от того, кто будет исполнителем этого алгоритма. Если задача решается с помощью ЭВМ, алгоритм решения задачи должен быть записан в понятной для машины форме, т. е. в виде программы. Всякий алгоритм может быть:

- записан на естественном языке (примеры записи алгоритма на естественном языке приведены при определении понятия алгоритма);
- изображен в виде блок-схемы;
- записан на алгоритмическом языке.

Запись алгоритмов в виде блок-схем. Схема алгоритма — графическое представление алгоритма. Каждый пункт алгоритма отображается на схеме некоторой геометрической фигурой — блоком — и дополняется элементами словесной записи. Правила выполнения схем алгоритмов регламентирует ГОСТ 19.002—80 (единая система программной документации, см. табл. 1.1)

Блоки на схемах соединяются линиями потоков информации. Основное направление потока информации идет сверху вниз и слева направо (стрелки могут не указываться), снизу вверх и справа налево — стрелка обязательна. Количество входящих линий для блока не ограничено. Выходящая линия должна быть одна (исключение составляет логический блок).

Таблица 1.1. Основные элементы блок-схем

№ п/п	Символ	Наименование	Содержание
1.		Блок вычислений	Вычислительные действия или последовательность действий
2.		Логический блок	Выбор направления выполнения алгоритма в зависимости от некоторого условия
3.		Блоки ввода-вывода данных	1. Общие обозначения ввода (вывода) данных (вне зависимости от физического носителя) 2. Вывод данных, носителем которых является документ
4.	Начало (конец)	Начало или конец алгоритма, вход или выход в программу	
5.		Процесс пользователя (подпрограмма)	Вычисление по стандартной программе или подпрограмме
6.		Блок модификации	Функция выполняет действия, изменяющие пункты (например, заголовок цикла) алгоритма
7.		Соединитель	Указание связи прерванными линиями между потоками информации в пределах одного листа.
8.		Межстраничные соединения	Указание связи между информацией на разных листах

В линейном вычислительном процессе все действия выполняются в строгой последовательности друг за другом. Таким образом, существует только один путь, по которому можно пройти из блока «Начало» в блок «Конец» алгоритма, т.е. выполнить алгоритм.

Пример 1.

Составить алгоритм вычисления площади треугольника с заданными сторонами a , b и c .

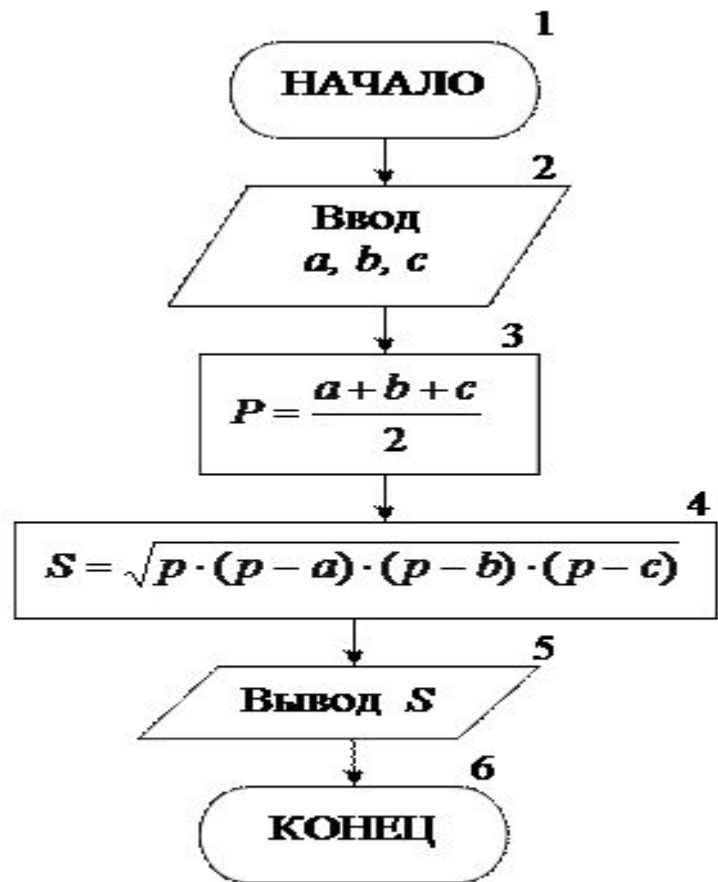


Рисунок 1. Алгоритм линейной структуры

Как известно из курса геометрии площадь треугольника, заданного длинами сторон, можно вычислить, используя формулу Герона. Разрабатываемый алгоритм (рис. 1) должен обеспечивать ввод исходных данных, т.е. значений длин сторон треугольника a , b и c (блок 2). Затем, используя введенные значения, вычислять полупериметр P (блок 3) и значение площади треугольника S (блок 4). После завершения вычислений необходимо вывести результат, т.е. полученное значение площади треугольника S (блок 5).

4.2. Разветвляющийся вычислительный процесс

Позволяет выбрать один из нескольких вариантов решения поставленной задачи в зависимости от выполнения некоторых условий. Таким образом, существует несколько различных путей, по которым можно пройти из блока «Начало» в блок «Конец» алгоритма, т.е. выполнить алгоритм.

Для реализации процесса выбора одного из двух вариантов решения используется логический блок (блок проверки условий), приведенный на рисунке 2.



Рисунок 2. Логический блок

При входе в этот блок выполняется проверка логического условия (обычно математического неравенства). Если результат проверки условия «Истина», т.е. условие выполняется, то происходит переход к выполнению блоков, стоящих по ветви «+». В противном случае, т.е. когда проверяемое условие не выполняется, происходит переход к выполнению блоков, стоящих по ветви «-».

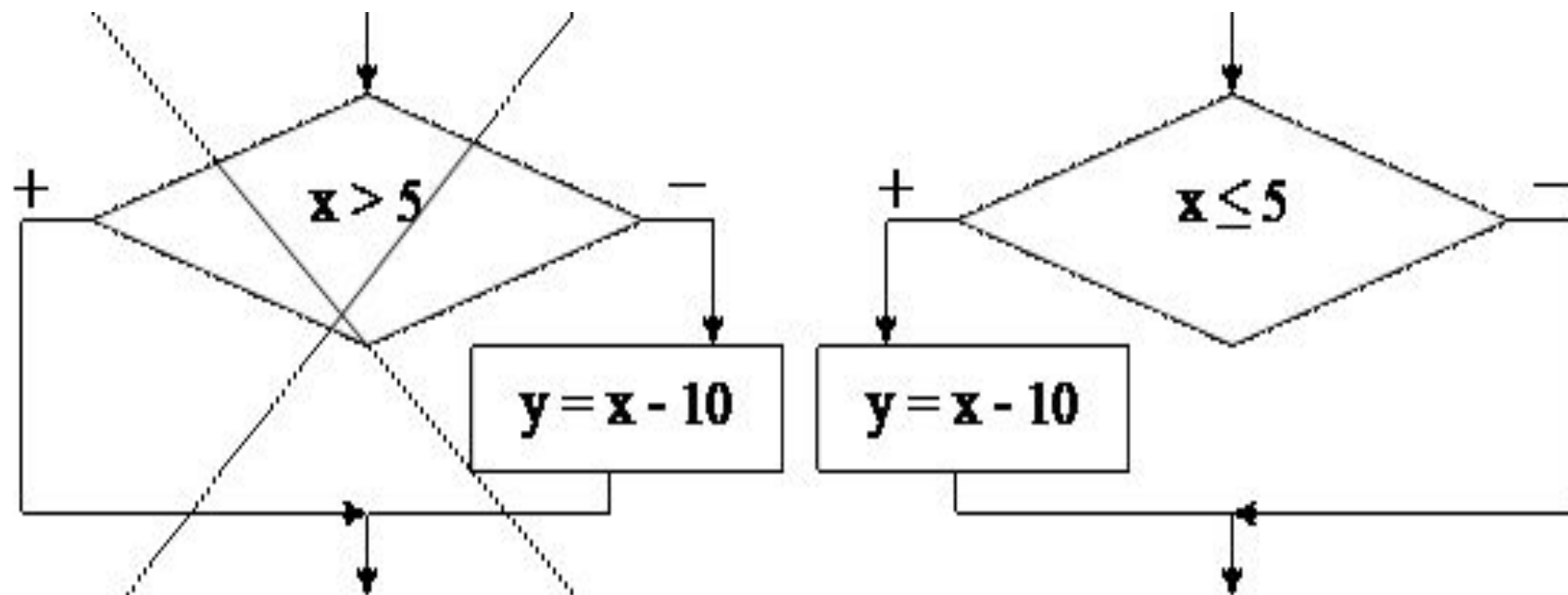


Рисунок 3. Рациональное использование логического блока

Если требуется выбрать один из трех и более вариантов решения, то необходимо использовать вложенные логические блоки. В случае, когда действия должны выполняться только по одной из ветвей логического блока, их необходимо реализовывать по ветви «+», подобрав соответствующее условие (рис. 3).

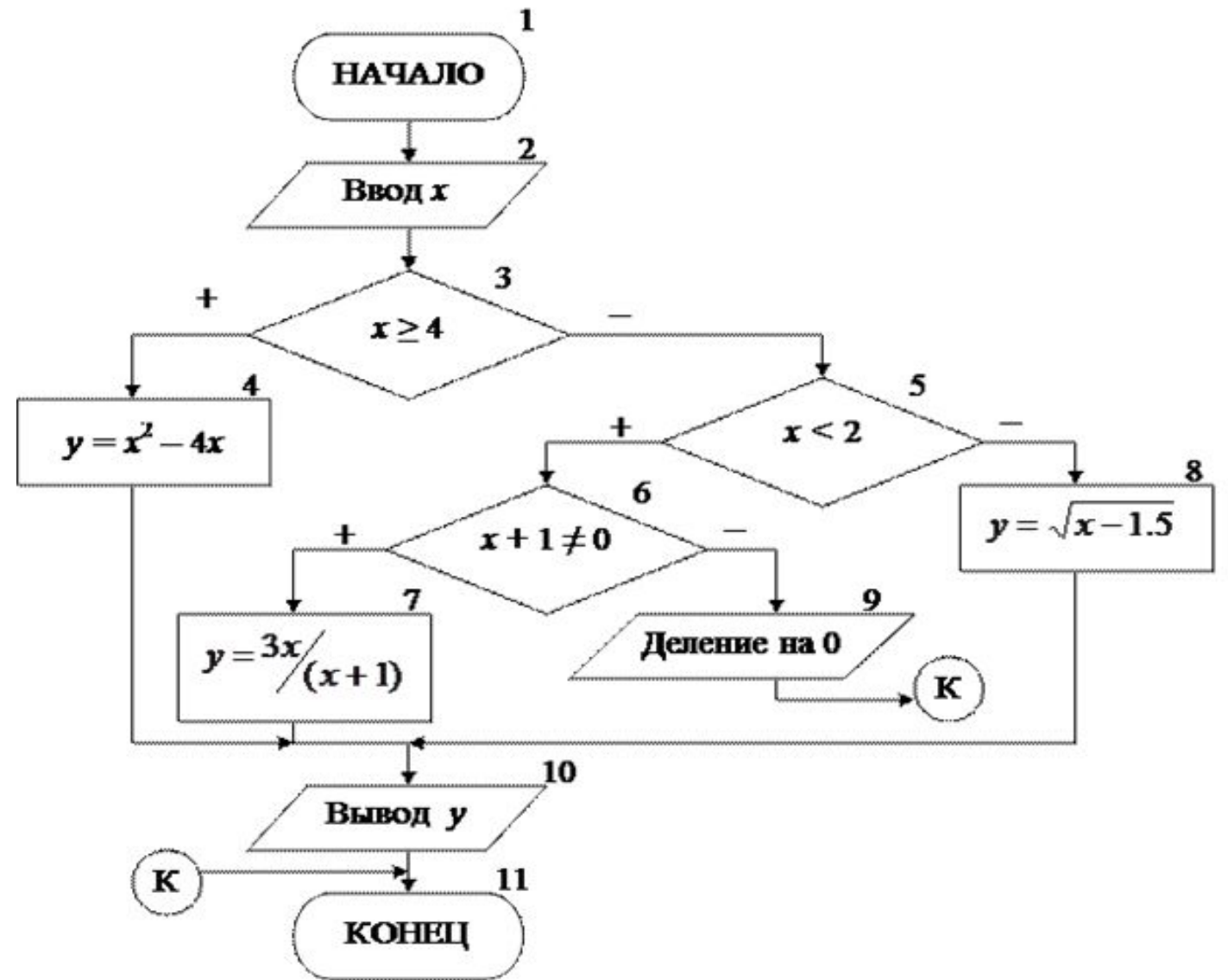


Рисунок 4 . Алгоритм разветвляющейся структуры

При выполнении вычислений необходимо учитывать область определения математических функций. Следовательно, вначале необходимо проверить возможность вычисления данного математического выражения при текущих значениях исходных данных, т.е. проверить «аномалию». К наиболее часто встречающимся «аномалиям» относятся: операция деления (на 0 делить нельзя), вычисление квадратного корня (подкоренное выражение должно быть ≥ 0), вычисление логарифма (выражение под знаком логарифма должно быть > 0), вычисление tg , ctg . В случае возникновения «аномалии» (невозможно выполнить вычисления) необходимо пропустить все действия, которые зависят от вычисляемой величины, и перейти в ту часть алгоритма, где можно продолжить вычисления.

Пример 2. Составить блок-схему алгоритма, вычисляющего значение y по одной из трех формул, в зависимости от значения x .

$$y = \begin{cases} x^2 - 4x, & \text{если } x \geq 4 \\ \sqrt{x - 1.5}, & \text{если } 2 \leq x < 4 \\ \frac{3x}{x + 1}, & \text{если } x < 2 \end{cases}$$

Исходными данными для решения поставленной задачи является значение x , которое вводится в блоке 2. Затем в блоке 3 проверяется 1-е ограничение ($x \geq 4$), в случае его выполнения происходит переход к блоку 4, в котором вычисляется значение y по 1-й формуле. Если результатом проверки 1-го условия является «ЛОЖЬ» (это означает, что $x < 4$), то необходимо проверить одно из двух оставшихся условий. Обычно выбирается более «короткое» ограничение (в блоке 5 проверяется 3-е ограничение). Если $x < 2$, то необходимо вычислять y по 3-й формуле, в которой может возникнуть «аномалия» – деление на 0, поэтому в блоке 6 проверяется неравенство знаменателя дроби нулю. И только после этого в блоке 7 вычисляется значение y . В том случае, когда невозможно вычислить значение y по 3-й формуле, выводится сообщение о возникновении ошибки (блок 9) и выполняется переход на конец алгоритма. И, наконец, если значение x не удовлетворяет ни 1-му, ни 3-му ограничению, значит, выполняется 2-е ограничение и y вычисляется по 2-й формуле (блок 8). Эта формула также содержит «аномалию» – подкоренное выражение должно быть ≥ 0 . Однако, проверять её не надо, так как при любом значении x , попадающем в интервал $[2; 4[$, подкоренное выражение всегда > 0 . Не зависимо от того, по какой формуле будет вычислено значение y его надо вывести на экран, поэтому выходы блоков 4, 7 и 8 объединяются и происходит переход к блоку 10, в котором выводится y .

4.3. В алгоритмах циклической структуры выполнение одних и тех же действий может повторяться несколько раз. Организация циклического вычислительного процесса выполняется в несколько этапов:

1-й этап – подготовка к выполнению цикла. На этом этапе задаются начальные значения для параметра цикла и переменных, используемых для хранения накапливающихся величин (сумма, количество или произведение вычисляемых величин). **Параметр цикла** – это переменная, на основе которой строится цикл. Она должна удовлетворять трем условиям: являться исходной величиной для выполнения вычислений; изменяться по определенному закону (чаще всего это закон арифметической прогрессии); оказывать влияние на условие завершения повторяющихся вычислений.

2-й этап – тело цикла. Оно содержит арифметические и логические действия, которые могут повторяться определенное количество раз. В конце тела цикла обязательно должен быть блок, в котором изменяется значение параметра цикла.

3-й этап – условие выхода из цикла, которое предотвращает бесконечное выполнение цикла. С помощью этого условия проверяется надо ли повторять вычисления, или выходить из цикла.

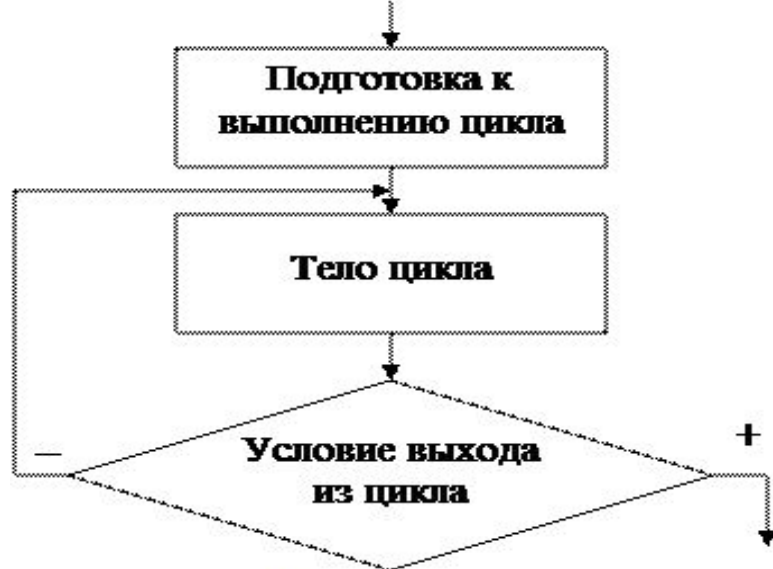


Рисунок 5.1. Организация цикла с постусловием

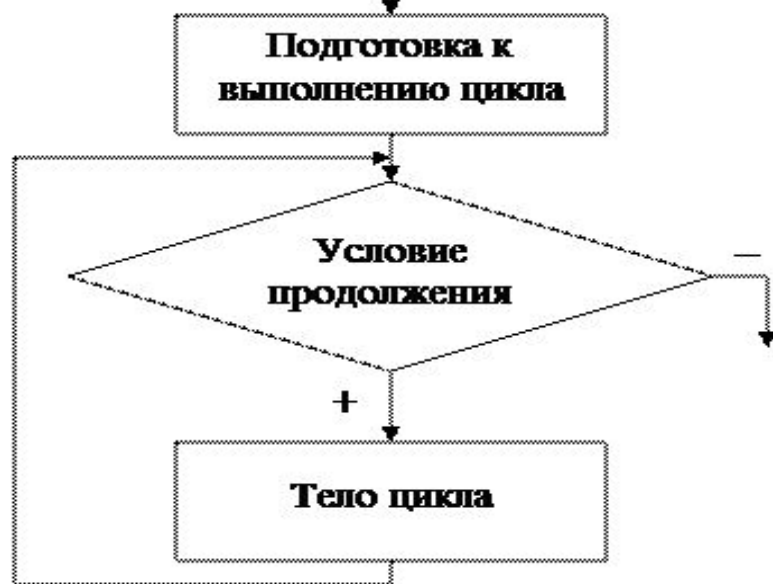


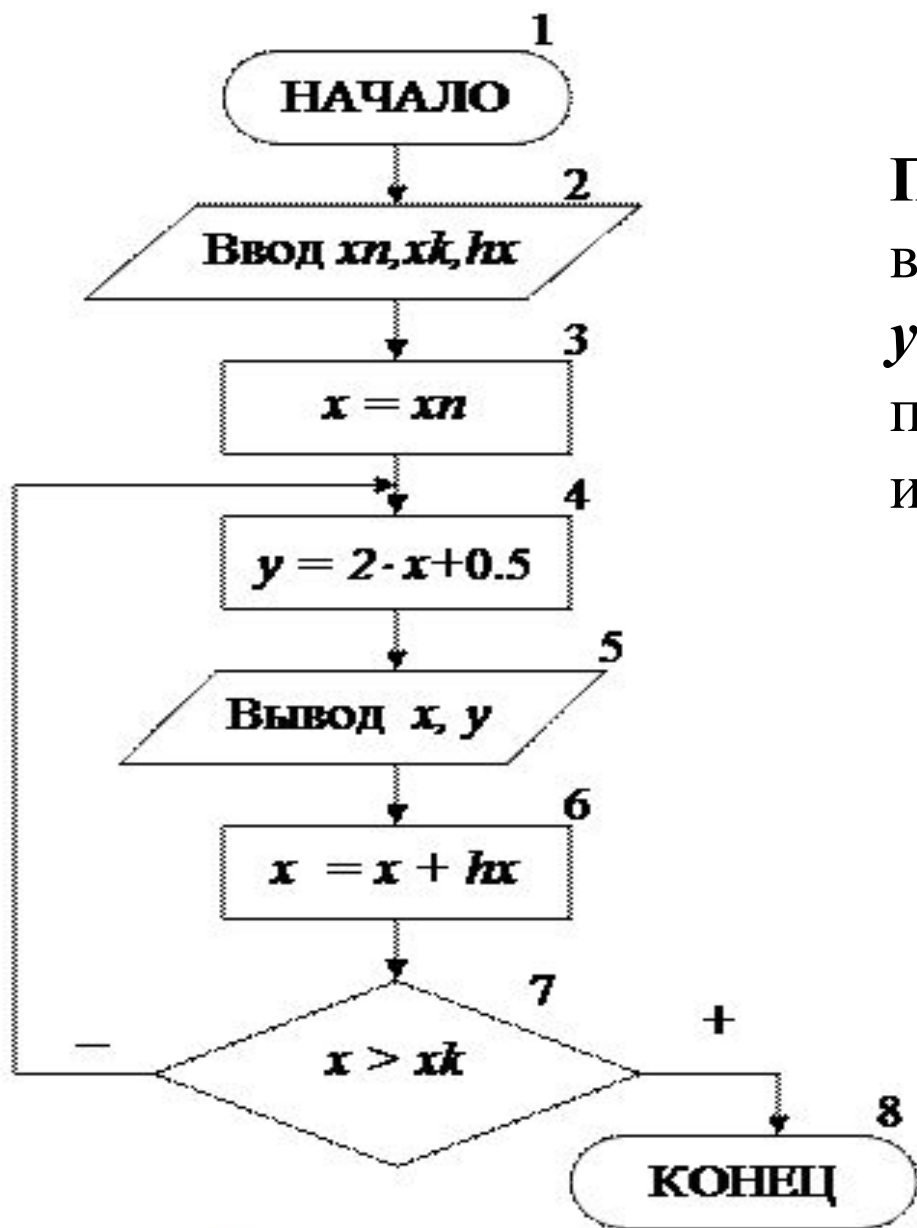
Рисунок 5.2. Организация цикла с предусловием

Блок-схема изображающая этапы организации цикла представлена на рис. 5.1. Такая организация циклического вычислительного процесса называется **циклом с постусловием**. В этом случае, после каждого выполнения тела цикла, проверяется условие выхода из цикла. Если оно не выполняется, то происходит возврат на начало тела цикла и повторение вычислений. Когда условие выхода будет выполнено, произойдет завершение работы и выход из цикла. Наличие линии возврата в блок-схеме является основным признаком циклического вычислительного процесса. Кроме цикла с постусловием также существуют цикл с предусловием и цикл «Для».

В **цикле с предусловием** (рис. 5.2) перед началом тела цикла проверяется **условие продолжения цикла**. При этом если условие продолжения цикла является истиной, то выполняются действия, составляющие тело цикла, и происходит переход в начало на проверку условия. Цикл завершает свою работу в том случае если условие продолжения цикла не выполняется – становится ложью. Таким образом, в цикле с постусловием в отличие от цикла с предусловием, тело цикла всегда выполнится хотя бы один раз.

Цикл «Для» реализуется на основе блока модификации и представляет собой вариант цикла с предусловием, в котором предполагается, что часть действий по организации цикла выполняется автоматически. Работа цикла «Для» более подробно будет рассмотрена в последующих разделах.

Цикл, в состав которого не входят другие циклы, называется простым.

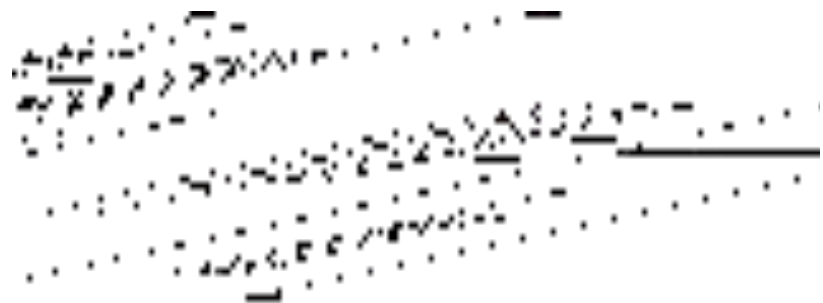


Пример 3. Составить блок-схему алгоритма, вычисляющего значения функции $y = 2 \cdot x + 0.5$, при различных значениях x , принадлежащих интервалу $1 \leq x \leq 3$, $h \cdot x = 0.5$.

Рисунок 5.3. Блок-схема для примера 3.

Таблица 1. Пошаговое выполнение цикла

№ шага	Текущее значение x (в начале цикла)	Вычисленное значение y	Новое значение x (в конце цикла)	Результат проверки условия выхода из цикла ($x > x_k$)
1	1 (x_n)	2,5	1,5	Ложь
2	1,5	3,5		Ложь
3		4,5	2,5	Ложь
4	2,5	5,5		Ложь
5	3 (x_k)	6,5	3,5	Истина



Где $[]$ обозначают целую часть выражения.

Циклические вычислительные процессы, для которых можно вычислить количество шагов цикла без выполнения алгоритма, называются **циклами с известным числом повторений**. Для реализации циклов с известным числом повторений можно равноценно использовать любой из трех стандартных типов цикла (с постусловием, с предусловием, «Для»).

Если в цикле при выполнении вычислений может возникнуть «**аномалия**», то завершать выполнение алгоритма, как в разветвляющемся вычислительном процессе, не надо. В этом случае необходимо пропустить все операции, использующие переменную, значение которой невозможно вычислить, и выполнить переход к блоку, в котором изменяется значение параметра цикла, т.е. продолжить работу цикла. При следующем значении параметра цикла «аномалия» может не возникнуть, и работа цикла пойдет естественным путем.

Часто наряду с циклическим вычислением значений величины или совокупности величин необходимо определить сумму, произведение или количество всех получаемых значений или некоторых из них.

При выполнении суммирования и вычислении произведения или количества рационально использовать принцип постепенного накапливания значений величин, получаемых на каждом шаге цикла. Для хранения таких «накапливаемых» величин используются дополнительные переменные, которым предварительно необходимо задать начальные значения. Обычно это делается на этапе подготовки к выполнению цикла. **В качестве начального значения для суммы и количества используется ноль, для произведения – единица.**

Пример 5. Составить блок-схему алгоритма, вычисляющего значения функции $y = \ln(1 + 0.2 \cdot x)$,

при различных значениях параметра x , изменяющегося от $x_n \leq 3$ с шагом $hx = -2$. При этом вычислять y до тех пор, пока выражение под знаком логарифма остается больше 0. Определить количество (k) вычисленных значений y .

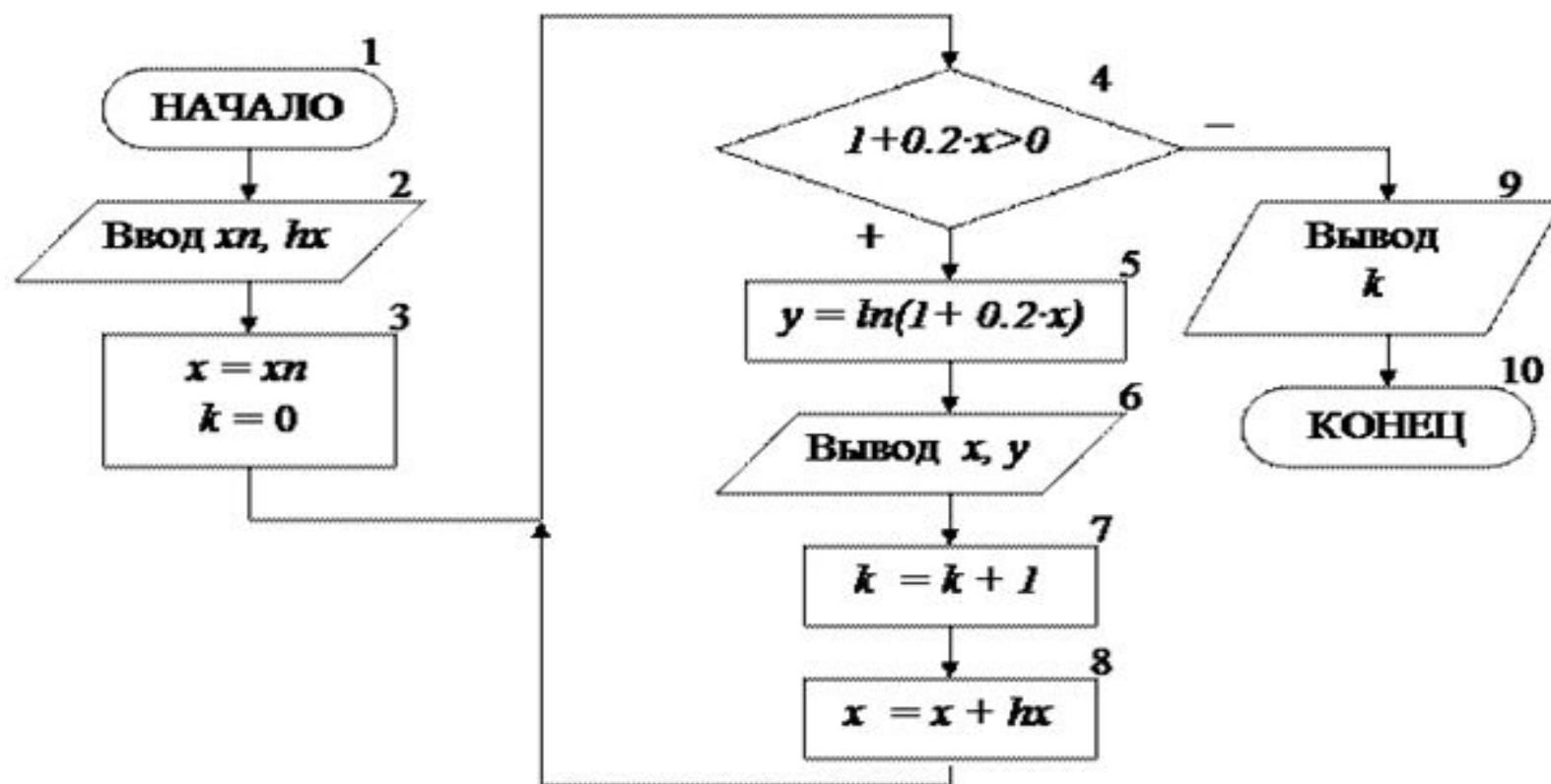


Рисунок 5.5. Блок-схема алгоритма для примера 5.