

Т.В. Волобуева

***ИНФОРМАТИКА***

**Основы алгоритмизации**

Учебное пособие

Воронеж 2019

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
Высшего образования  
«Воронежский государственный технический университет»

Т.В.ВОЛОБУЕВА

***ИНФОРМАТИКА***

**Основы алгоритмизации**

Учебное пособие

Воронеж 2019

УДК 69.003:658.5.012.22

ББК 32.973

В68

*Рецензенты:*

*Кафедра высшей математики и информационных технологий  
Воронежского государственного университета инженерных технологий;*

*Корелина Т.В., К.т.н*

**Волобуева, Т.В.**

**В68 Информатика. Часть 1** : учебное пособие / Т.В.Волобуева ,  
«Воронежский государственный технический университет». –Воронеж,  
Изд-во ВГТУ. 2019. – 96 с.

ISBN

Изложен теоретический материал по основам алгоритмизации.  
Приведены примеры расчета задач и задания для самостоятельной работы.

Ил. 77 Табл. 1. Библиограф.: 17 названий

УДК 69.003:658.5.012.22

ББК 32.973

*Печатается по решению учебно-методического совета*

*Воронежского государственного технического университета*

ISBN

© Волобуева Т.В. 2019

©ФГБОУ ВО «Воронежский  
государственный технический  
университет», 2019

## **ВВЕДЕНИЕ**

Алгоритмизация – это раздел информатики, изучающий методы создания и свойства алгоритмов. Решение любого рода задач на ЭВМ основано на применении алгоритмов.

Цель данного пособия состоит в ознакомлении специалистов профилей «Теплоэнергетика и теплотехника» и «Нефтегазовое дело» с основами алгоритмизации и формировании у них умений по использованию основных приемов алгоритмизации в профессиональной деятельности.

Пособие предназначено для генерирования у студентов начальных навыков алгоритмизации, которые образуют фундаментальную базу для изучения основ программирования. В нем рассмотрены базовые структуры алгоритмов и стандартные приемы их построения для решения вычислительных задач. Приведены примеры решения широкого круга задач, иллюстрирующих изложенный теоретический материал, и задания для самостоятельной подготовки. Пособие адаптировано для студентов - первокурсников

После изучения изложенного материала будущий специалист должен владеть основами алгоритмизации решения инженерных задач.

## **ТЕХНОЛОГИЯ РЕШЕНИЯ ЗАДАЧ НА КОМПЬЮТЕРЕ**

Решение задачи на ЭВМ представляет собой процесс получения искомой информации на основе обработки исходных данных с помощью программы. В законе РФ от 23 сентября 1992 г «О правовой охране программ для электронных вычислительных машин и баз данных» программа для ЭВМ определена как объективная форма представления совокупности данных и команд, предназначены для функционирования

электронных вычислительных машин и других компьютерных устройств с целью получения определенного результата.

Создание такой программы решения задачи на ЭВМ предполагает выполнение ряда последовательных этапов технологического процесса

### **Основные этапы компьютерного решения задачи**

**Первым этапом** процесса подготовки решения задачи на ЭВМ является *постановка задачи*.

Постановка задачи включает формулирование условия задачи, определение конечных целей решения задачи, подробное описание содержания задачи. Описываются характер и сущность всех величин задачи (тип данных, их структуру, диапазон изменения величин задействованных в решении задачи, т.е. ограничения, налагаемые на данные предметной областью и т.д.), форма представления ожидаемых результатов решения задачи, определяются условия, при которых задача имеет решение. Важно, чтобы количество исходной информации было достаточным для получения решения задачи.

Постановка задач должна включать описание контрольного примера, который должен демонстрировать алгоритм решения задачи штатным способом, а так же отражать весь спектр возможных вариантов исходных данных задачи. В контрольном примере должны быть перечислены все нештатные ситуации, которые могут возникнуть в процессе решения задачи, а так же должен быть описан алгоритма действий пользователя в каждой конкретной нештатной ситуации.

Конечный пользователь разрабатываемой программы, хорошо знающий ее предметную область, может слабо представлять возможности использования ЭВМ для ее решения. В свою очередь предметная область пользователя может быть незнакома разработчику программы, реализующему алгоритм ее решения, хотя он хорошо представляет себе

возможности ЭВМ, которые можно использовать при реализации разрабатываемого алгоритма. Под предметной областью понимаем часть реального мира, данные о котором используются при решении задачи.

Это противоречие может стать причиной возникновения ошибок при реализации данного этапа решения задачи с использованием ЭВМ. По данным экспертов, на этот этап приходится более 50% общего числа ошибок, обнаруживаемых в процессе разработки программ решения задач вычислительного и инженерного характера. При исправлении подобного рода ошибок расходуется в среднем 80% всего времени разработчиков на поиск и устранение ошибок в программе. Таким образом, постановка задачи – важнейший этап, который должен обеспечить корректную и полную постановку задачи, так как от нее во многом зависят все последующие действия, а также однозначность ее понимания, как разработчиком программы, так и пользователем этой программы, в качестве которого обычно выступает постановщик задачи. В противном случае разрабатываемый программный продукт не будет соответствовать требованиям конечного потребителя.

На этапе постановки задачи должно быть четко определено, что должна делать программа, в чем реальные проблемы, решению которых должна способствовать программа, что представляют собой исходные данные. Какими должны быть выходные данные, какими ресурсами располагает разработчик.

**Вторым этапом** в технологии разработки программы является ***анализ и математическое описание (моделирование, формализация) задачи и выбор метода ее решения.***

В результате анализа задачи определяется объем исходных данных, устанавливается принадлежность задачи к одному из известных классов задач, выбирается математический аппарат ее решения.

Моделью называется представление основных элементов системы в виде совокупности физических или математических объектов. Математическая модель позволяет свести реальную задачу к ее математическому представлению. Для достижения этой цели условие задачи записывается в виде системы уравнений, либо в виде последовательности формул, необходимых для получения искомого результата. Математическая модель – приближенное описание реального объекта или явления. Степень адекватности модели реальному объекту проверяется экспериментально, в случае необходимости - модель уточняется.

Математическая модель задачи должна обеспечивать однозначное ее понимание пользователем и разработчиком программы, которая выполняет решение этой задачи.

Результатом второго этапа технологии разработки программы должна быть **формализованная** (в виде формул) постановка задачи.

Например.

Фабрика раскраивает листы фанеры, получая в результате заготовки А, В, С. Существует два способа раскроя. При первом способе с одного листа получается 2 заготовки А, 5 – В, 2 – С. При втором способе раскроя с одного листа получается 6 заготовок А, 4 - В, 3 – С. Заказ: необходимо произвести не менее 24 заготовок А, 31 – В, 18 – С.

При первом способе раскроя получается 12 единиц отходов с листа, при втором – 16. Сколько листов и каким способом нужно раскроить, чтобы выполнить заказ при минимальном отходе?

Построим математическую модель.

Сначала определимся с количеством неизвестных задачи – их два.

Пусть  $x_1$  – количество листов, которые нужно раскроить первым способом, чтобы выполнить заказ, а  $x_2$  – количество листов раскроенных

вторым способом

Функция отходов:  $12x_1$  – количество отходов при раскрое первым способом  $x_1$  листов, а  $16x_2$  – при раскрое вторым способом  $x_2$  листов.

Общее количество отходов  $12x_1 + 16x_2$ .

В задаче требуется добиться  $\min(12x_1 + 16x_2)$ .

Существуют ограничения в задаче на количество производимых заготовок. Сформулируем их математически. При раскрое  $x_1$  листов первым способом получится  $2x_1$  заготовок А. При раскрое  $x_2$  листов вторым способом получится  $6x_2$  заготовок А. Всего, таким образом, имеем  $2x_1 + 6x_2$  заготовок А. И по условию задачи их должно быть не менее 24. Т.е.

$$2x_1 + 6x_2 \geq 24$$

Аналогично для заготовок В и С получаем

$$5x_1 + 4x_2 \geq 31$$

$$2x_1 + 3x_2 \geq 18$$

И ограничение на не отрицательность решения

$$x_1 \geq 0$$

$$x_2 \geq 0$$

Итак, формализованная постановка задачи (ее математическая модель):

найти  $\min(12x_1 + 16x_2)$ .

При условии, что

$$\begin{cases} 2x_1 + 6x_2 \geq 24 \\ 5x_1 + 4x_2 \geq 31 \\ 2x_1 + 3x_2 \geq 18 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{cases}$$



Третий этап процесса подготовки решения задачи на ЭВМ представляет собой *алгоритмизацию ее решения*, т.е. выбор метода решения задачи и разработку собственного или применение уже известного алгоритма реализации выбранного метода.

В большинстве случаев математическую постановку задачи бывает трудно записать в виде машинного кода, поэтому возникает необходимость найти способ, т.е. алгоритм решения таких задач. Для некоторых классов задач, например, нахождение корней квадратного уравнения, существуют точные методы решения, которые можно представить в виде последовательности математических и логических действий. Для многих задач, например, вычисления интегралов, дифференциальных уравнений, системы уравнений и т.д. точное решение либо неизвестно, либо оно приводит к громоздким формулам. Приближенное решение такого рода задач с требуемой точностью можно найти применив специальные численные методы.

Для тех задач, которые удастся описать математически, необходимо выбрать и обосновать один из известных численных методов их решения, либо построить математическую модель их решения, а для нечисловых задач – принципиальную схему их решения (т.е. последовательность выполнения элементарных математических и логических операций).

После того, как численный метод выбран, приступают к построению алгоритма решения задачи.

Алгоритмизация – это процесс построение новых или преобразование уже известных алгоритмов к условиям конкретных технических заданий. Постановка задачи и ее алгоритмизация занимают в среднем 20 - 30% общего времени на разработку программных средств решения задачи. Для решения одной и той же задачи или уже существует множество различных алгоритмов, или существует возможность построения нескольких новых алгоритмов. Эти алгоритмы могут

отличаться друг от друга уровнем сложности, объемами вычислительных работ, оригинальность решения, другими факторами, влияющими на эффективность выбранного способа решения задачи.

"Час, потраченный на выбор алгоритма, стоит пяти часов программирования" (Д. Ван-Тассел).

Процесс алгоритмизации решения задачи в общем случае реализуется по следующей схеме:

- разбиение процесса решения поставленной задачи на отдельные этапы (части, блоки) обычно с одним входом и одним выходом;
- формализованное (в виде формул) описание шагов (действий), выполняемых на каждом из выделенных этапов;
- проверка правильности реализации выбранного алгоритма на различных контрольных примерах решения для поставленной задачи.

Название "алгоритм" происходит от латинизированного воспроизведения арабского имени узбекского математика Аль-Хорезми, жившего в конце VIII – начале IX вв., Он первым сформулировал правила, позволяющие систематически составлять и решать квадратные уравнения.

Трактовка алгоритма в соответствии с ГОСТ 19.004-80: «алгоритм – это точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату». Или более развернутое определение: алгоритм – это точное и понятное предписание возможному исполнителю совершить определенную последовательность действий для получения решения задачи за конечное число шагов.

Упрощая можно сказать, что алгоритм описывает процесс преобразования исходных данных в результаты, т.к. для решения любой задачи необходимо:

1. Ввести исходные данные;

2. Преобразовать исходные данные в ожидаемое решение задачи (выходные данные);

3. Вывести результаты.

На этапе разработки алгоритма рекомендуется придерживаться следующих правил его составления:

1. Алгоритм должен быть как можно более простым и понятным.

2. Алгоритм должен состоять из элементарных шагов (интуитивно понятных).

3. Сложную задачу необходимо разбивать на простые, легко понимаемые части.

4. Логика алгоритма должна основываться на минимальном числе базовых управляющих структур из всех возможных вариантов.

### **Свойства и формы записи алгоритмов**

Любой алгоритм должен обладать следующим набором основных свойств: понятностью для исполнителя, дискретностью, массовостью, результативностью, определенностью.

Понятность для исполнителя означает, что исполнитель должен понимать, как выполнять предписанные алгоритмом действия при любом варианте исходных данных

Дискретность (прерывность, раздельность) – алгоритм должен представлять процесс решения задачи как последовательность выполнения простых шагов

Определенность означает, что каждое действие алгоритма должно быть однозначным, не оставляющим места для произвольного толкования

Результативность (конечность) состоит в том, что за конечное число шагов алгоритм либо должен приводить к решению задачи либо после

выполнения конечного числа шагов останавливаться из-за невозможности получения решения задачи с выдачей соответствующего сообщения

Массовость означает, что алгоритм разрабатывается в общем виде, т.е. должен быть применим к классу задач, различающихся только исходными данными

Существует несколько способов описания алгоритмов:

- словесный,
- формульно-словесный,
- графический,
- средствами специального языка операторных схем,
- с помощью таблиц решений и др.

Словесный способ описания алгоритма отражает содержание выполняемых действий средствами естественного языка. К достоинствам этого способа описания следует отнести его общедоступность, а также возможность описывать алгоритм с любой степенью детализации. К главным недостаткам этого способа относится достаточно громоздкое описание (и, как следствие, низкая наглядность представления вычислительного процесса), а так же отсутствие строгой формализации в силу неоднозначности восприятия естественного языка, вытекающего из свойств синонимии, омонимии, полисемии.

Пример словесного способа записи алгоритма нахождения наибольшего общего делителя (НОД) двух натуральных чисел.

1. Задать два натуральных числа;
2. Если числа равны, то взять любое из них в качестве ответа и остановиться, в противном случае выполнить пункт 3 алгоритма;
3. Определить большее из чисел;
4. Заменить большее из чисел разностью большего и меньшего из чисел;
5. Повторить алгоритм с пункта 2.

Формульно-словесный способ описания алгоритма основан на записи содержания выполняемых действий с использованием математических символов и выражений, в сочетании со словесными пояснениями. Данный способ, обладает всеми достоинствами словесного способа. Кроме того он более лаконичен, а значит, и более нагляден, имеет большую формализацию.

В качестве примера рассмотрим алгоритма решения задачи по геометрии.

Дано: основание треугольника  $AC = 5$  см, высота треугольника  $BD = 5$  см. (См. рис.1).

Найти: площадь  $S$  треугольника  $ABC$ .

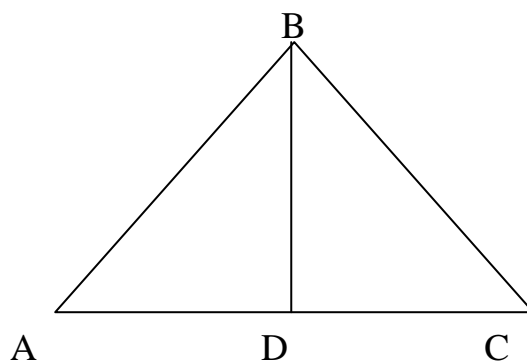


Рис.1

Решение:




1. Площадь треугольника находится по формуле  $S = AC * BD$  ,
2. Подставим исходные данные задачи в формулу:  $S = 5 * 5$  ,
3. Результат решения задачи: площадь  $S$  треугольника  $ABC$  равна 25 см.

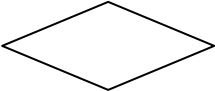



Графический способ описания алгоритмов.

При графическом представлении алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий. Такое представление называется блок-схемой. В блок-схеме каждому типу действий (вводу исходных данных, вычислению значений выражения, проверке условия и т.д.) соответствует блочный символ (геометрическая фигура). Блочные символы соединяются линиями переходов, определяющих очередность выполнения действий алгоритма.

Таблица 1.

Функциональное назначение блочных символов

Геометрическая фигура, обозначающая функциональный блок на схеме	Действие алгоритма, которому она соответствует
	Начало или конец алгоритма
	Ввод или вывод данных (преобразование данных в форму пригодную для обработки(ввод) или отображения результатов обработки (вывод))
	Процесс. Выполнение операций или группы операций, в результате которых изменяется значение, форма представление или расположение данных. (арифметический блок)

	Решение Выбор направления выполнения алгоритма в зависимости от некоторых условий (логический блок)
	Модификация. Выполнение операций, меняющих команды или группу команд, изменяющих алгоритм
	Соединитель (установление связи между прерванными линиями потока)
	Межстраничный соединитель (установление связей между разорванными частями схем алгоритма расположенных на разных листах)

Составление графического изображения алгоритмов осуществляется в соответствии с требованиями ГОСТ 19701-90 – «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения» и ГОСТ 19.003-80 «Схемы алгоритмов и программ. Обозначения условные графические».

Изображение блок-схем алгоритмов осуществляется по определенным правилам, которые должны повысить их наглядность и однозначность восприятия всеми пользователями и исполнителями. Следование этим правилам облегчит обнаружение логических ошибок в программах. В соответствии с этими правилами каждая блок-схема должна начинаться и заканчиваться соответствующими символами, обозначающими начало и окончание алгоритма.

Все функциональные блоки в блок-схеме должны располагаться в последовательности сверху вниз и слева направо. Линии потока призваны объединить функциональные блоки между собой. Направление сверху вниз и слева направо – это нормальное направление линий потока (т.е. естественный порядок следования этапов процесса решения задачи). В этом случае направление линий потока не указывают с помощью стрелок. Другие направления процесса решения задачи должны быть указаны с помощью стрелок.

Необходимым условием корректности блок-схем алгоритмов является недопустимость более чем одного выхода из функциональных блоков, обозначающих обрабатываемые операции, и не менее двух выходов из функциональных блоков, обозначающих логические операции по проверке выполнения условий.

Для обозначения направлений передачи управления алгоритмом от блоков логических операций над соответствующими линиями потока могут присутствовать слова ДА либо НЕТ

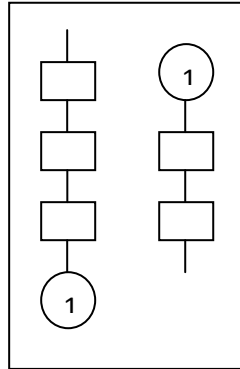
Для отражения связей между удаленными друг от друга функциональными блоками соединяющие их линии потока можно разрывать. В этом случае в конце и начале обрыва линии потока изображаются специальные символы:

– при отражении разрыва связей между функциональными блоками, расположенными на одной и той же странице внутри соединителей, отмечающих разрыв одной и той же линии потока, ставятся одинаковая буквенная, цифровая или буквенно-цифровая маркировка. Иллюстрация этой ситуации приведена на рис. 2

– при отражении разрыва связей между функциональными блоками, расположенными на разных страницах внутри соединителей, отмечающих разрыв одной и той же линии потока, требуется указать маркировку блока соединителя (может быть цифра, буква или буквенно-цифровая

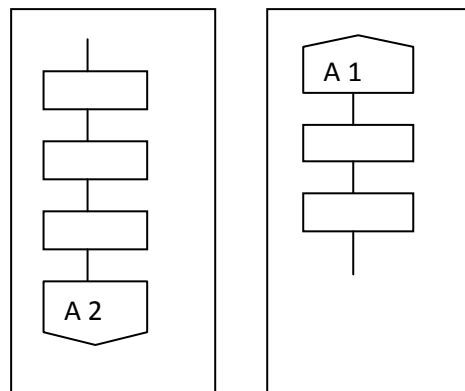


последовательность), а также номер следующего за разрывом или предшествующего разрыву листа. Иллюстрация этой ситуации приведена на рис. 3



Лист 1

Рис.2 Вид разрыва линий перехода на одном листе



Лист 1

Лист 2

Рис 3. Вид разрыва линий перехода на различных листах

## **АЛГОРИТМИЗАЦИЯ ЗАДАЧ НА ОСНОВЕ ТИПОВЫХ АЛГОРИТМОВ**

Решение задач, реализуемое с использованием ЭВМ, осуществляется, посредством составления программы. Основой программы для вычислительной машины является алгоритм решения поставленной задачи, т.е. точное предписание о последовательности действий, которые должны быть произведены для получения результата.

Программа для вычислительной машины – это запись алгоритма решения некоторой задачи в виде, пригодном для данной вычислительной машины. Следовательно, основная сущность процесса решения задач с помощью ЭВМ – это разработка алгоритмов.

Графическое изображение алгоритма широко используется перед программированием задачи вследствие его наглядности. Зрительное восприятие алгоритма значительно облегчает процесс написания программы и ее последующей корректировки при возможных ошибках.

Когда алгоритм решения задачи разработан и записан в виде блок-схемы, он без труда может быть представлен на любом языке программирования с учетом особенностей конкретных вычислительных машин или предметной области решаемой задачи.

Вывод: главное в решении задач на ЭВМ – это алгоритмизация. Вычислительные процессы, применяемые для реализации различного рода алгоритмов на ЭВМ в общем виде можно разделить на три большие группы – линейные, разветвляющиеся и циклические

Логическую структуру алгоритма любой сложности можно представить в виде комбинации этих трех базовых алгоритмических структур – следование (линейный алгоритм), ветвление, цикл.

### **Методика разработки алгоритмов**

При разработке алгоритма используют следующие основные методики:

– проектирование алгоритма снизу вверх. Сначала разрабатываются алгоритмы решения конкретных задач, а затем они объединяются в более крупное решение;

– проектирование сверху вниз. В этом случае крупная задача разбивается на более мелкие подзадачи до тех пор, пока каждую из них можно будет решить наиболее простым способом.

### Линейный алгоритм

Линейным называется такой алгоритмический процесс, при котором все элементарные действия, приводящие к решению задачи, выполняются друг за другом, т.е. в естественном порядке следования записи этих действий. Элементарным называем действие, которое не требует большей детализации вследствие своей очевидности. Порядок выполнения действий (шагов алгоритма) не зависит от исходных данных алгоритма решения задачи и от результатов выполнения предыдущих этапов (шагов) решения задачи. В общем виде блок-схема линейного алгоритма приведена на рис. 4.



Рис. 4. Фрагмент блок схемы линейного алгоритма

#### Пример 1 (поэтапного решения задачи)

1 этап (постановка задачи)

В поезде имеется  $a$  купейных мест и  $b$  мест класса люкс. Непроданными оказались  $c$  билетов на купейные места и  $d$  билетов класса

люкс. Билеты на купейные места в два раза дешевле, чем билеты на места класса люкс. Найти общую сумму, полученную компанией от продажи билетов.

2 этап (построение математической модели)

Для нахождения решения нужно вычислить значение следующего выражения

$$S = (b - d)x + \frac{(a - c)x}{2}$$

Здесь  $a, b, c, d$  – целые величины

Величины  $S, x$  – вещественного типа

$x$  – цена билетов класса люкс

$S$  – искомая сумма, полученная от продажи билетов

3 этап (запись алгоритма, в графической форме)

Блок схема алгоритма приведена на рис. 5

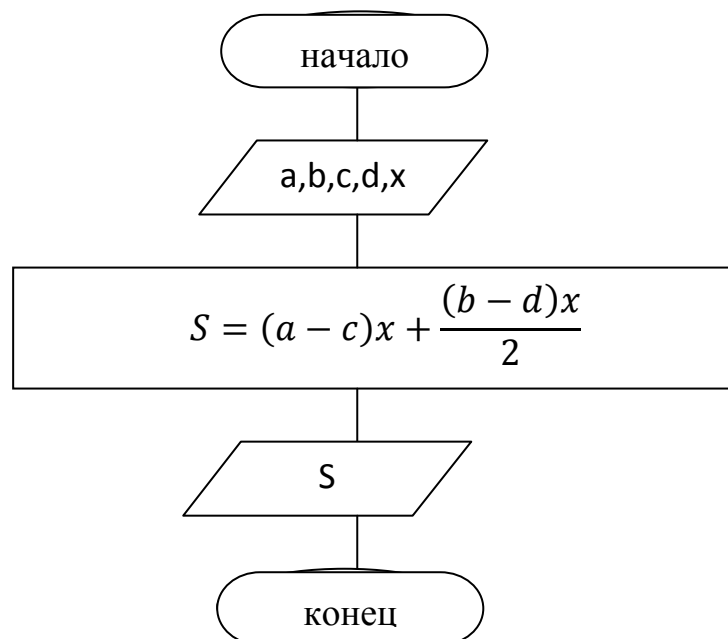


Рис. 5. Блок схема алгоритма решения задания примера 1

Исполняемые действия алгоритма:

1. Ввести количество купейных мест, количество мест класса люкс, количество непроданных купейных мест, количество непроданных мест класса люкс, цену билета класса люкс ( $a, b, c, d, x$ ).

2. Вычислить сумму дохода компании  $S$ .

3. Вывести вычисленный результат.

**Пример 2.** Вычислить функцию

$$y = \frac{x}{1 - \frac{x^2}{3 - \frac{x^2}{5 - \frac{x^2}{7}}}}$$

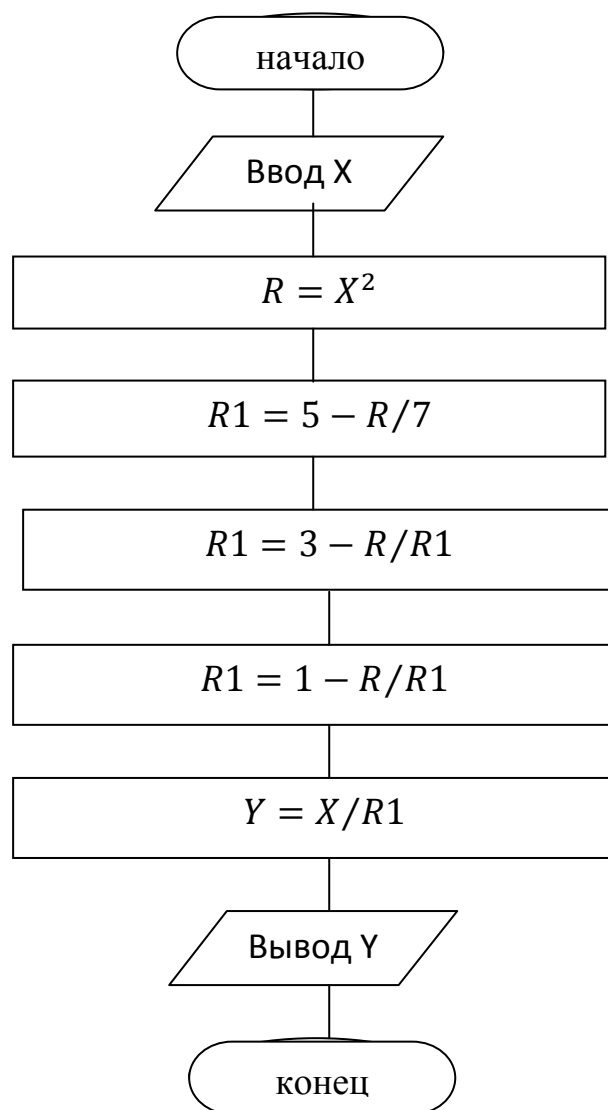


Рис. 6. Блок схема алгоритма решения задания примера 2

Исполняемые действия алгоритма:

1. Ввести любое число  $X$
2. Вычислить вспомогательную переменную  $R$  (Обоснование – многократно встречается выражение  $X^2$  в функции  $Y$ ).
3. Блоки с 4 по 6 – Вычислить знаменатель очередной дроби.
4. Вычислить функцию  $Y$ .
5. Вывести результат.

### Задания для самостоятельной работы

1. Даны любые пять чисел:  $a, b, c, d, e$ . Найти среднее арифметическое и среднее геометрическое заданных чисел.
2. Вычислить функцию  
 $z = x^5 - 0,2x^4 + 5,5x^3 + 0,1x - 2,5$  в точках  $x = 1,7$ ;  $x = -2,6$
3. Даны катеты прямоугольного треугольника  $a$  и  $b$ . Найти гипотенузу  $c$  и прилежащие углы прямоугольного треугольника  $\alpha, \beta$ .
4. Известен диаметр окружности  $B$ . Требуется найти длину окружности  $L$ .
5. Известны два числа  $A$  и  $B$ , ни одно из них не равно нулю. Найти их сумму  $S$ , разность  $R$ , произведение  $P$  и частное. Результаты вывести на печать.
6. Известны катеты прямоугольного треугольника  $A$  и  $B$ . Требуется вычислить его гипотенузу  $C$ .
7. Вычислить функцию  $y = 0,8x^4 + 2,6x^5 - 2,3x^2 + x^6 + 1,57$  в точках  $x = 1,1$ ;  $x = 2,1$
8. Поменять местами содержимое ячеек  $A$  и  $B$  двумя способами.
9. Известны координаты точек  $A$  и  $B$  на плоскости:  $A(x_1, y_1), B(x_2, y_2)$ . Требуется вычислить длину отрезка  $AB$ .
10. Известна диагональ квадрата  $A$ . Вычислить площадь  $S$  квадрата.

11. Известны три числовые величины  $A_1$ ,  $A_2$ ,  $A_3$ . Требуется осуществить перенос содержимого величины  $A_2$  в  $A_1$ , содержимого величины  $A_3$  в  $A_2$ , содержимого величины  $A_1$  в  $A_3$ . Результат перестановки вывести на печать.

12 Известна длина окружности  $L$ . Требуется вычислить радиус этой окружности  $R$ .

13. Известна сторона квадрата  $A$ . Вычислить длину диагонали  $D$  этого квадрата. Результат вывести на печать.

14. Даны три точки на координатной плоскости:  $A(x_1, y_1)$ ,  $B(x_2, y_2)$  и  $C(x_3, y_3)$ . Требуется вычислить длины отрезков  $AB$ ,  $BC$  и  $CA$ . Результат вычисления вывести на печать.

15. Известна гипотенуза  $c$  и один из прилежащих углов  $\alpha$  прямоугольного треугольника. Найти площадь прямоугольного треугольника  $S$  и прилежащий угол  $\beta$ .

### **Разветвляющийся алгоритм**

Вычислительный процесс называется *разветвляющимся*, если для его реализации предусмотрено несколько альтернативных направлений решения задачи. Выбор одного из путей решения задачи, то есть выбор определенного способа обработки информации зависит от результатов вычисления логического выражения (логического условия). Для обозначения действия «проверить условие (логическое выражение)» в блок-схеме используют логический блок. Каждое отдельное направление обработки информации, появляющееся в результате проверки логического условия, называется ветвью.

В зависимости от характера логического условия разветвляющийся процесс может состоять из двух и более ветвей. Для разветвляющейся алгоритмической структуры характерно, что в любой конкретный момент ее реализации осуществляется обработка информации только по одной из

ветвей, а выполнение операции по другим ветвям исключается. Поэтому, для обеспечения проверки корректности разработанного алгоритма решения конкретной задачи, контрольный пример должен содержать данные, предусматривающие проверку всех ветвей алгоритма.

Итак, базовая структура ветвления обеспечивает в зависимости от результатов проверки логического условия (истина/ложь) выбор одного из нескольких возможных ветвей работы алгоритма. Каждая из этих ветвей должна приводить к общему выходу из структуры ветвления, поэтому работа алгоритма будет продолжена вне зависимости от того какая из ветвей решения была выбрана. Любая стандартная базовая алгоритмическая структура должна иметь один вход и один выход.

Структура ветвления может существовать в четырех вариантах

### 1. Если-то (рис. 7)

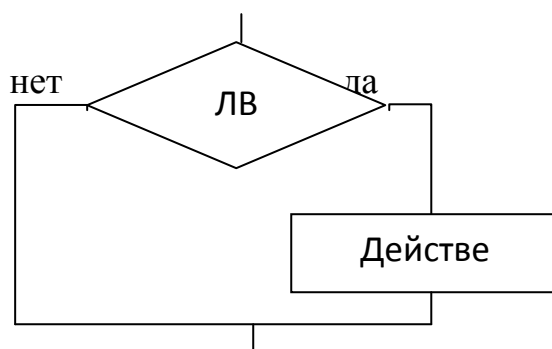


Рис. 7. Вариант базовой алгоритмической структуры ветвление

Здесь ЛВ – логическое выражение (условие), в результате вычисления которого можем получить только две величины истина (да) или ложь (нет)

Читать эту структуру нужно так:

**Если** логическое выражение (ЛВ) истина (да), **то** нужно выполнить «действие». В качестве этого действия может выступать любая из базовых алгоритмических структур (следование, ветвление, цикл). Поскольку любая базовая алгоритмическая структура должна иметь один вход и один



выход, т.е. сколько бы ветвей не образовалось в процессе ветвления (в данном случае две) все они должны прийти к общему выходу из структуры ветвления (соединиться в общей точке). Необходимо следить, чтобы не возникло оборванных ветвей.

## 2. Если то иначе (рис. 8)

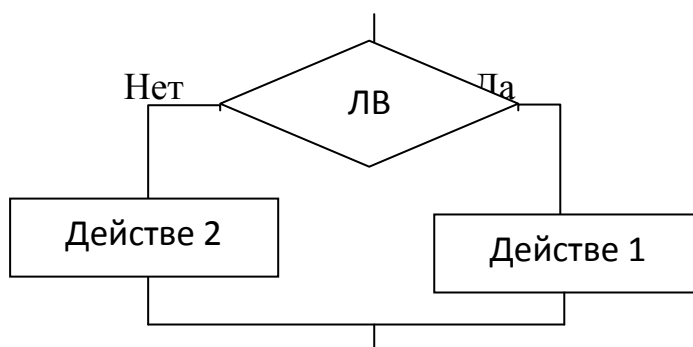


Рис. 8. Вариант базовой алгоритмической структуры ветвление

Читать эту структуру нужно так:

**Если** логическое выражение (ЛВ) в результате вычисления примет значение истина (т.е. ответ на поставленный в условии вопрос – да), **то** нужно выполнить «действие 1» **иначе** (т.е. логическое выражение в результате вычисления принимает значение ложь, иными словами ответ на поставленный в условии вопрос – нет) необходимо выполнить «действие 2».

## 3. Выбор (рис. 9)

Читать эту структуру нужно так:

Если логическое выражение (ЛВ1) в результате вычисления примет значение истина (выход «да» из первого логического блока), то нужно выполнить «действие 1», иначе (т.е. в случае если ЛВ1 в результате вычисления приняло значение ложь (на схеме это ветвь «нет» из первого логического блока) нужно проверить еще одно логическое выражение ЛВ2. Если ЛВ2 истина (выход «да» из второго логического блока), то необходимо выполнить «действие 2», иначе (т.е. ЛВ2 приняло значение

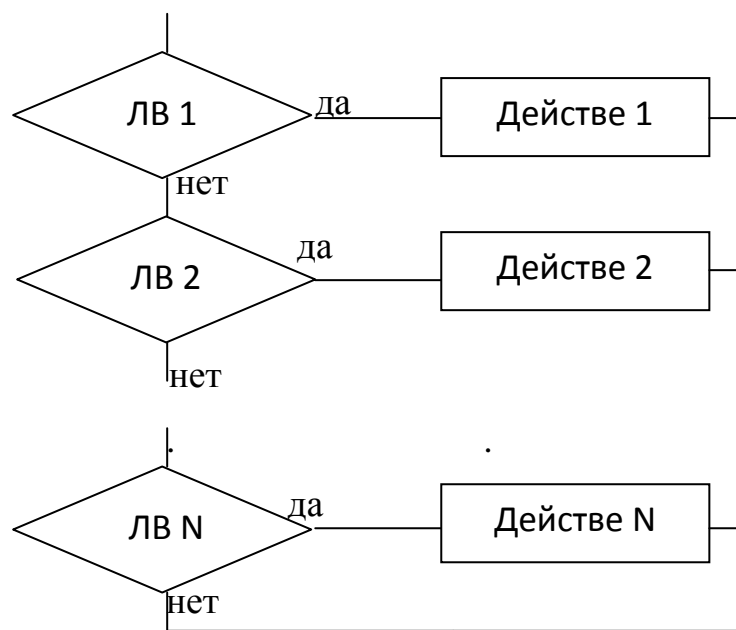


Рис .9 Вариант базовой алгоритмической структуры ветвление

ложь, это выход «нет» из второго логического блока) необходимо проверить следующее логическое выражение и так далее. Вложенность не ограничена. Если ни одно из логических выражений ЛВ1, ЛВ2,..., ЛВN не является истиной, тогда ничего не делать (выход «нет» из последнего логического блока). Все ветви, образующиеся в результате этого вложенного процесса ветвления должны приводить к общему выходу из структуры ветвления (т.е. соединяться в одной точке, что будет означать – структура ветвления закончена). При этом варианте ветвления действие выбирается в зависимости от истинности соответствующего ему логического выражения.

#### 4. Выбор иначе (рис.10)

Эта структура отличается от структуры выбора только тем, что предусмотрено действие «действие N+1», которое выполнится в случае, если ни одно из логических выражений ЛВ1, ЛВ2,..., ЛВN не является истиной (выход «нет» из последнего логического блока).

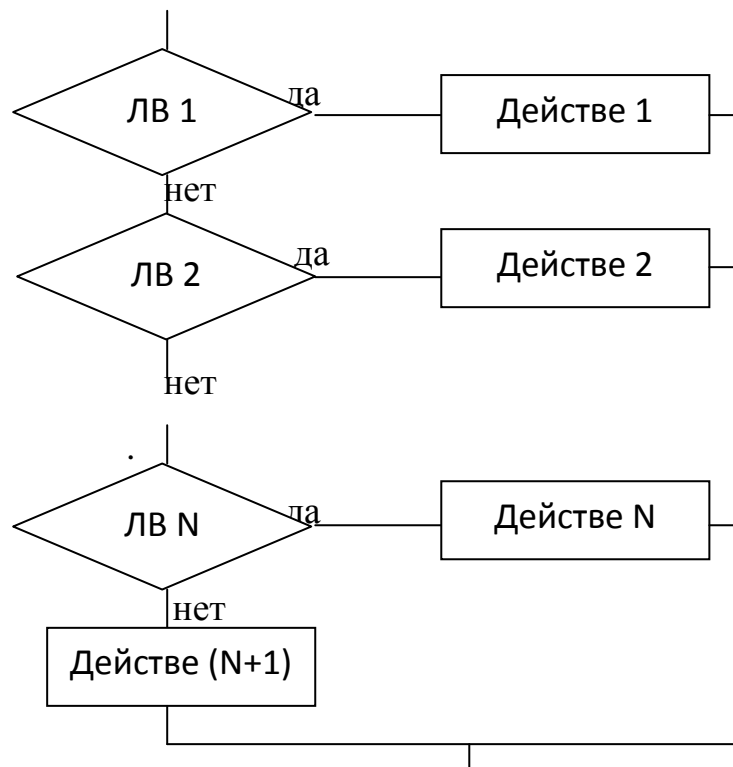


Рис. 10. Вариант базовой алгоритмической структуры ветвление

**Пример 3.** Вводится число, если оно положительное требуется его удвоить. Разработать блок схему алгоритма

Математическая модель:

$$x = 2x \quad \text{если} \quad x > 0$$

Блок схема алгоритма решения приведена на рис. 11.

$x$  – любое введенное число.

Исполняемые действия алгоритма:

1. Вводим любое число  $X$ .
2. Поскольку у задачи несколько путей решения в зависимости от результатов проверки условия  $X > 0$ , то начинается разветвляющийся процесс. Проверяем условие – «положительное ли введенное число?».

3. Если условие пункта 2 является истинным, то введенное число удваивается, в противном случае ничего не происходит. Разветвление закончено. Все ветви пришли в одну точку.

4. Выводим результат.

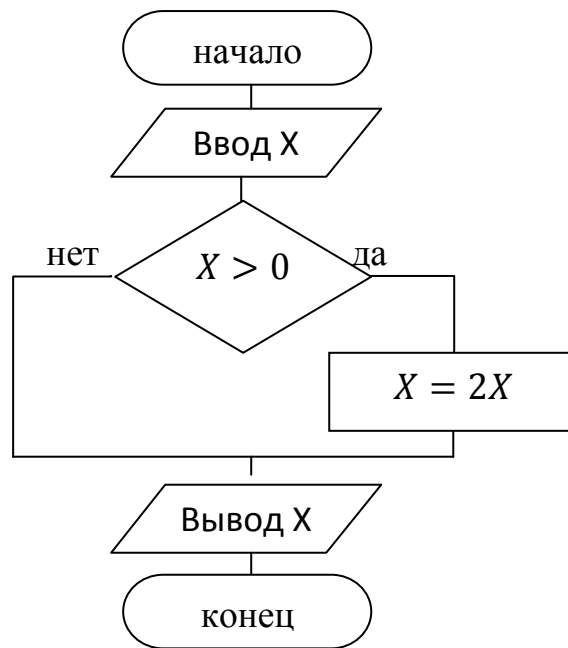


Рис. 11. Блок схема алгоритма решения задания примера 3

**Пример 4.** Вводится число, если оно положительное требуется его удвоить, в противном случае утроить. Разработать блок схему алгоритма.

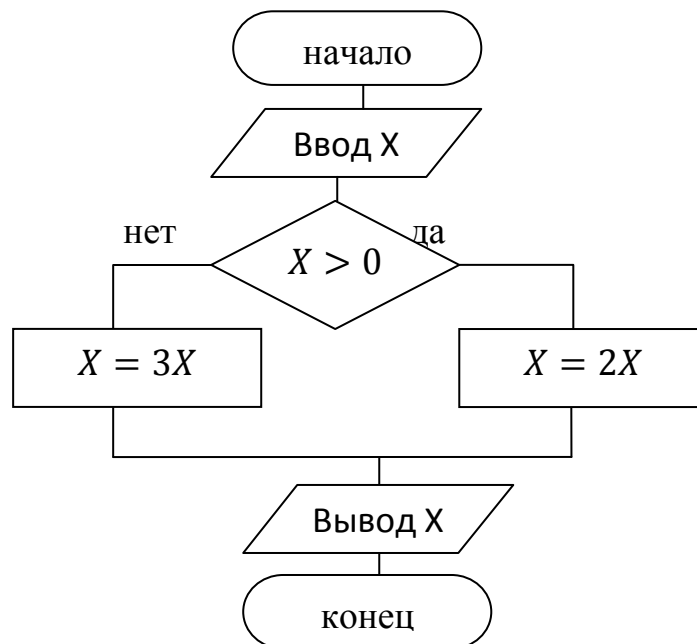


Рис. 12. Блок схема алгоритма решения задания примера 4

Математическая модель

$$X = \begin{cases} 2X, & \text{если } X > 0 \\ 3X, & \text{если } X \leq 0 \end{cases}$$

Блок схема алгоритма решения приведена на рис. 12.

Исполняемые действия алгоритма:

1. Вводим любое число X.
2. Начинается разветвляющийся процесс. Проверяем условие «является ли положительным введенное число?».
3. Если условие пункта 2 является истиной, то введенное число удвоим, в противном случае утроим. Разветвление закончено.
4. Выводим результат.

**Пример 5.** Вводится число, если оно положительное требуется его удвоить, если отрицательное утроить, если это ноль оставить без изменения. Разработать блок схему алгоритма

Математическая модель:

$$X = \begin{cases} 2X, & \text{если } X > 0 \\ 3X, & \text{если } X < 0 \\ X, & \text{если } X = 0 \end{cases}$$

Блок схема алгоритма решения приведена на рис. 13.

Исполняемые действия алгоритма:

1. Вводится любое число X.
2. Начинается разветвляющийся процесс. Проверяем условие «введенное число положительное?».
3. Если условие пункта 2 в результате вычисления дает истину, то введенное число удвоим.
4. В противном случае (введенное число не положительное), введенное число либо отрицательное, либо ноль, а в зависимости от этого, согласно условию, существуют два пути решения, поэтому проверяем еще одно условие «введенное число отрицательное?»

5. Если условие пункта 4 – истина, то число утроим.

6. Если же условие пункта 4 – ложь, т.е. введенное число оказалось равным нулю, ничего не нужно делать, т.е. число остается без изменения.

Разветвление закончено. Все ветви пришли в одну точку.

7. Выводим результат.

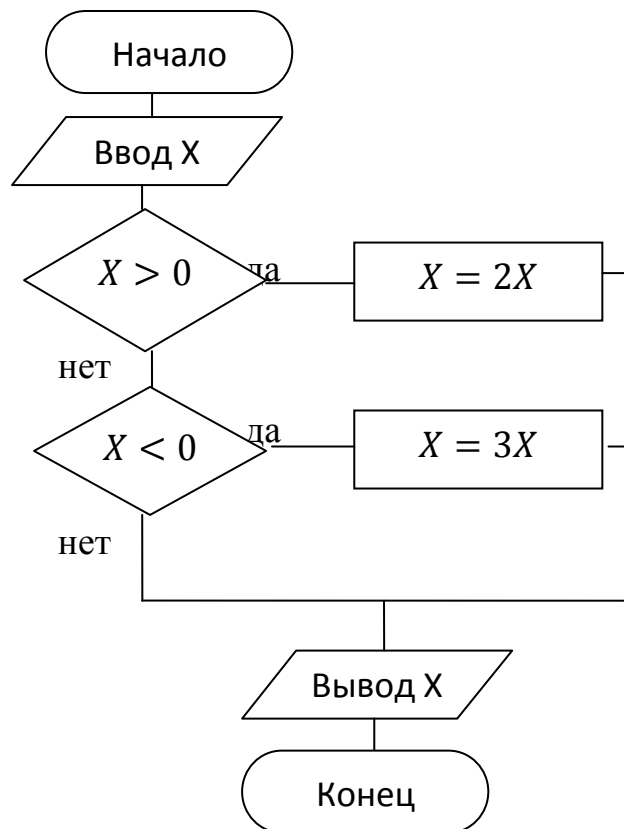


Рис. 13. Блок схема алгоритма решения задания примера 5

**Пример 6.** Вводится число, если оно положительное требуется его удвоить, если отрицательное утроить, если это ноль увеличить на единицу.

Разработать блок схему алгоритма.

Математическая модель:

$$X = \begin{cases} 2X, & \text{если } X > 0 \\ 3X, & \text{если } X < 0 \\ X + 1, & \text{если } X = 0 \end{cases}$$

Блок схема алгоритма решения приведена на рис. 14.

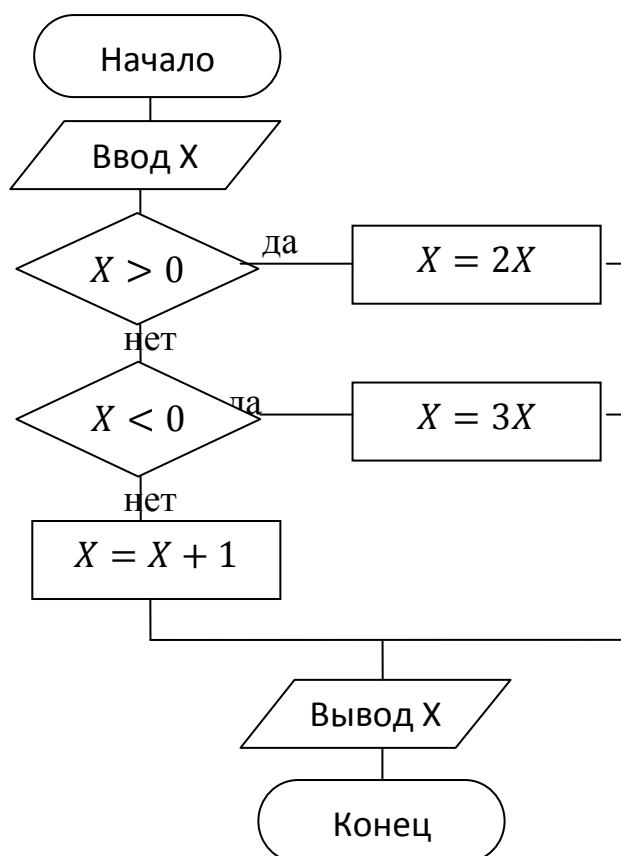


Рис. 14. Блок схема алгоритма решения задания примера 6

#### Исполняемые действия алгоритма

1. Вводим любое число  $X$ .
2. Начинается разветвление. Проверяем условие «введенное число положительное?»
3. Если условие пункта 2 истина, то число удваивается.
4. В противном случае (т.е. число оказалось либо отрицательным, либо нулем) проверим условие «введенное число отрицательное?»
5. Если условие пункта 4 истина, то введенное число утроится.
6. В случае если условие пункта 4 ложь (введенное число ноль), увеличим введенное число на 1. Разветвление закончено. Все ветви пришли в одну точку.
7. Выводим результат.

**Пример 7.** Вводится число, если оно положительное требуется его удвоить, если отрицательное утроить. Разработать блок схему алгоритма

Математическая модель:

$$X = \begin{cases} 2X, & \text{если } X > 0 \\ 3X, & \text{если } X < 0 \end{cases}$$

Блок схема алгоритма решения приведена на рис. 15.

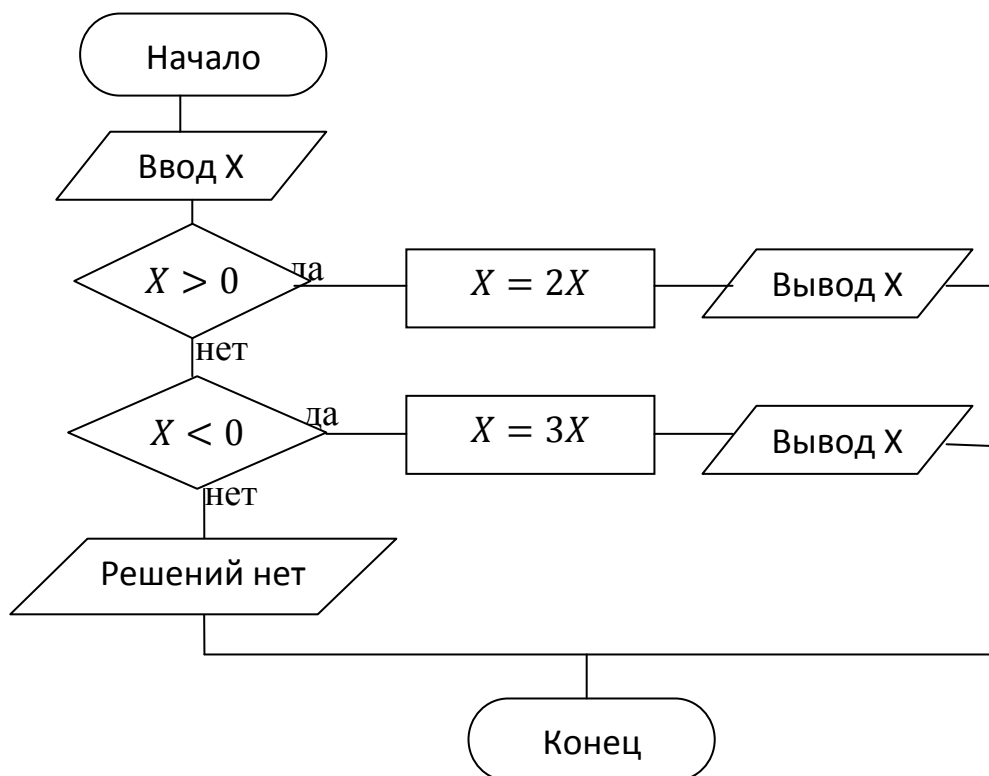


Рис. 15. Блок схема алгоритма решения задания примера 7

Исполняемые действия алгоритма:

1. Вводим любое число  $X$ .
2. Начинается ветвление. Проверим условие «введенное число положительное?»
3. Если да, число удвоим и выведем результат вычисления.



4. Если же условие пункта 2 примет значение ложь (введенное число отрицательное либо ноль) проверим условие «введенное число отрицательное?»

5. Если условие пункта 4 – истина, число утроим и выведем результат вычисления.

6. Если же условие пункта 5 примет значение ложь (введенное число 0) выведем сообщение о том, что при таких начальных данных у задачи нет решения. Разветвление закончилось. Все ветви пришли в одну точку.

Схему этого же алгоритма можно изобразить и в виде, приведенном на рис. 16

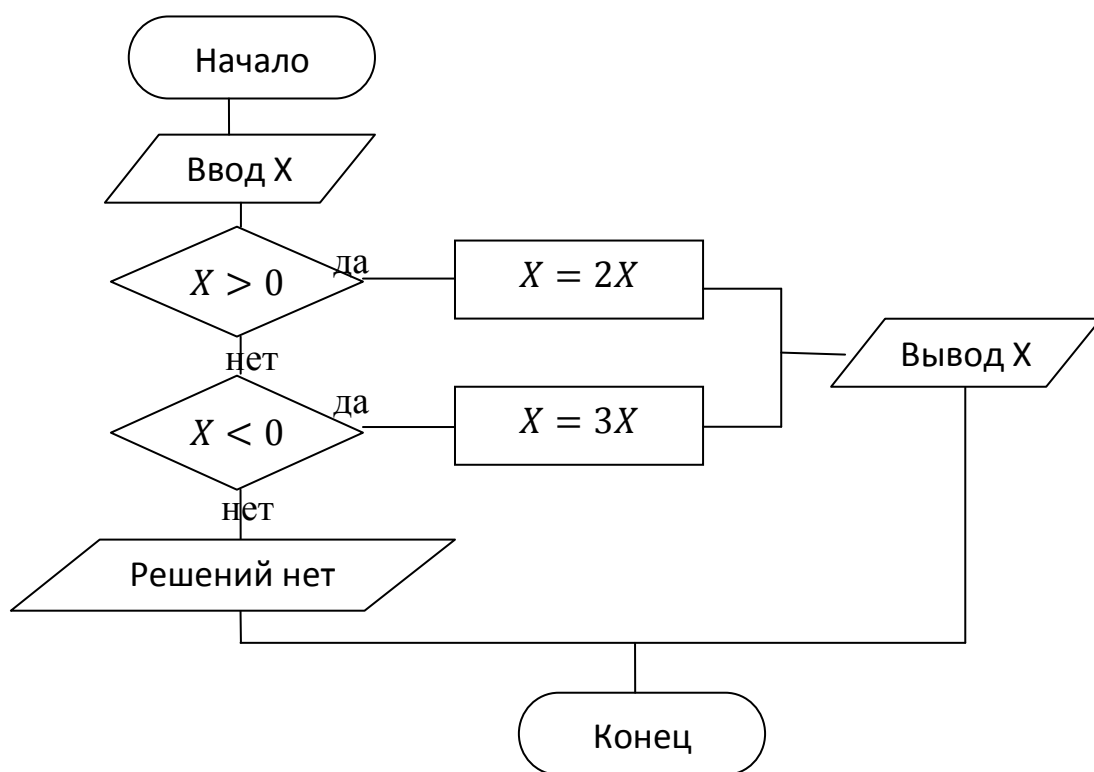


Рис. 16. Блок схема алгоритма решения задания примера 7

Два параллельных отрезка ветвей пункта 3, 5 накладываются, но не становятся от этого короче. Выход из структуры ветвления – в точке соединения всех ветвей (на схеме это место перед блоком “конец”). Т.е. в эту точку по прежнему приходят три альтернативных пути решения алгоритма.

Одна задача может иметь несколько способов (алгоритмов) решения. Рассмотрим это на примере

**Пример 8.** Найти максимальное из трех заданных чисел

Обозначения:

$X, Y, Z$  – введенные числа;

$MAX$  – максимальное из чисел.

Блок схема алгоритма 1 решения приведена на рис. 17.

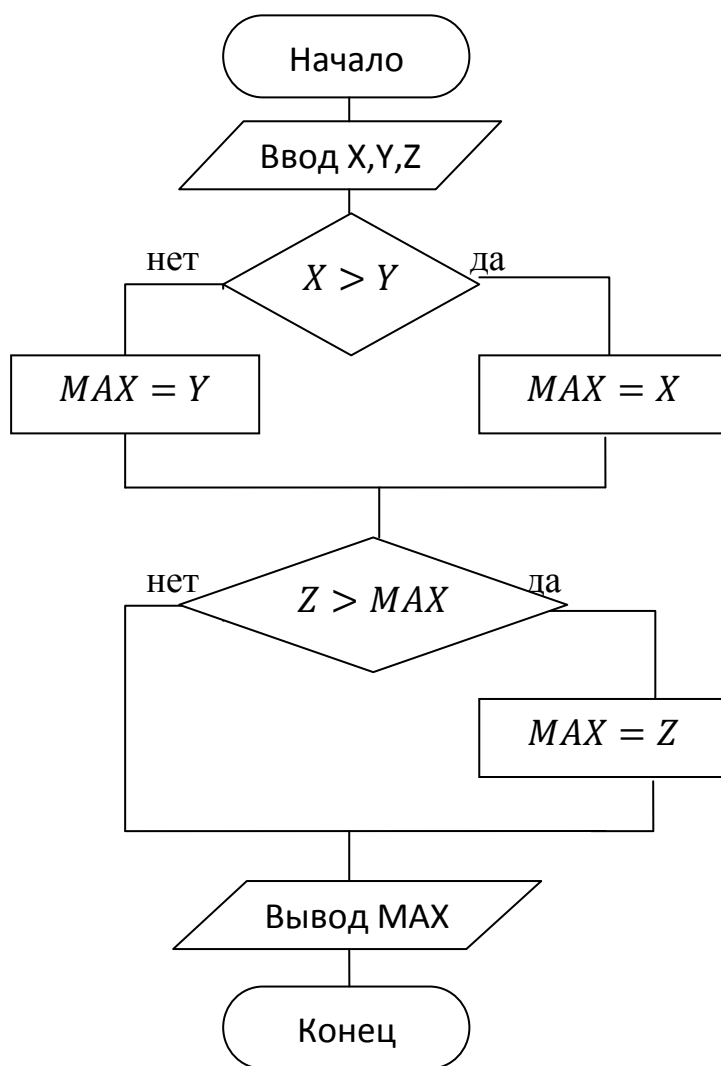


Рис. 17. Блок схема алгоритма 1 решения задания примера 8

Исполняемые действия алгоритма 1:

1. Вводим три числа.
2. Начало разветвления. Найдем максимальное число из первых двух введенных чисел. Конец разветвления. Ветви, образовавшиеся в процессе ветвления, пришли в одну точку.
3. Начало разветвления. Найдем максимум опять из двух чисел, найденного в предыдущем пункте алгоритма максимума и третьего введенного числа. Конец разветвления. Ветви, образовавшиеся в процессе ветвлении, пришли в одну точку.
4. Выведем результат – максимум из трех введенных чисел.

Блок схема алгоритма 2 решения приведена на рис. 18.

Исполняемые действия алгоритма 2:

1. Вводим три числа.
2. Начинается разветвление. Проверяем условие «Первое число больше второго?»
3. Если условие пункта 2 истина, то первое число сравним с третьим «первое число больше третьего?»
4. Если условие пункта 3 истина, то первое число – искомый максимум.
5. Если условие пункта 3 – ложь, то – искомый максимум – третье число.
6. Если в результат проверки условия в пункте 2 этого алгоритма – ложь, то проверим условие «второе число больше третьего?»
7. Если условие пункта 6 – истина, то искомый максимум – второе число.
8. Если условие пункта 6 – ложь, то третье число – максимальное. Разветвление закончено. Все ветви должны прийти в общую точку.
9. Выводим результат

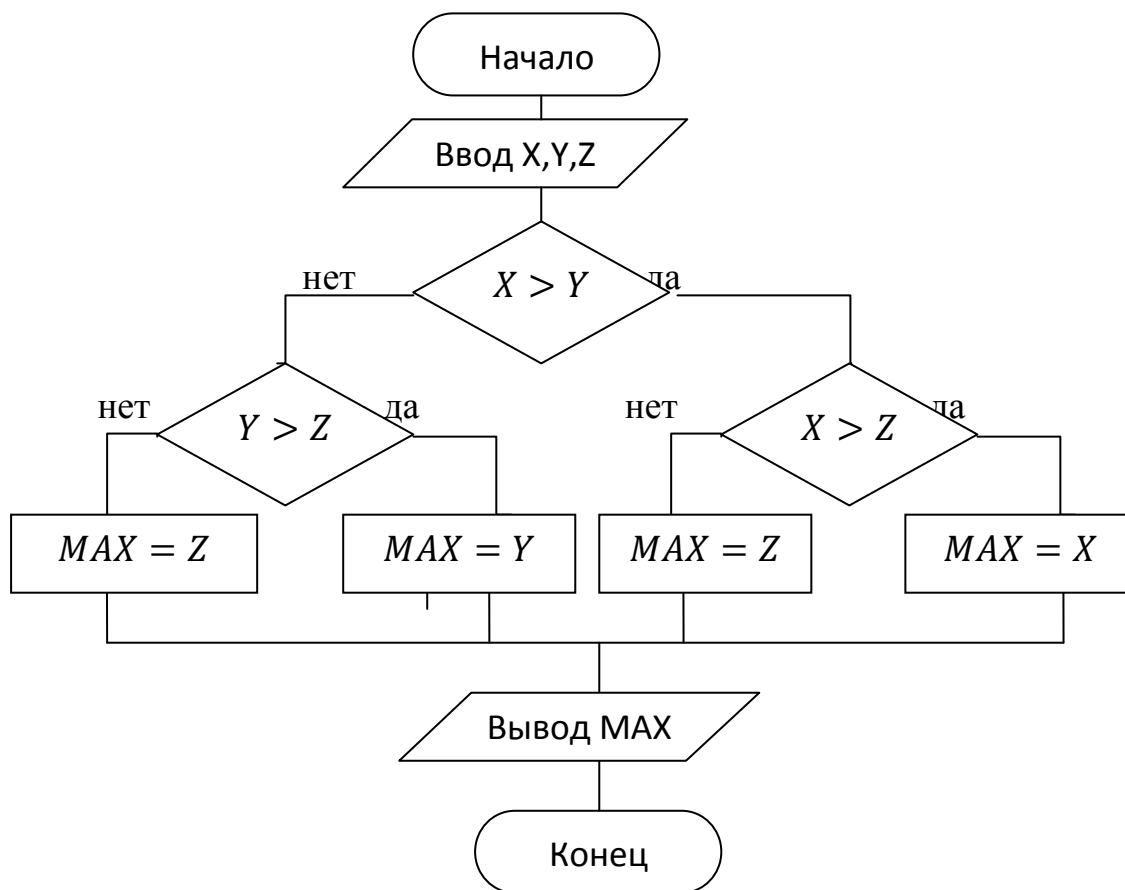


Рис. 18. Блок схема алгоритма 2 решения задания примера 8

**Пример 9.** Упорядочить три числа по возрастанию.

Обозначения:

$X, Y, Z$  – введенные числа;

$K$  – вспомогательная переменная (для хранения копии числа при обмене содержимого двух ячеек).

Блок схема алгоритма решения приведена на рис .19.

Исполняемые действия алгоритма:

Реализуем метод пузырька, т.е. выталкивания вперед наименьшего из введенных чисел.

1. Вводим три числа.
2. Начинается разветвление. Проверяем условие «первое число больше второго?»»

3. Если условие пункта 2 в результате вычисления – истина, то меняем местами первое и второе число. В противном случае ничего не делаем (т.е. нет перестановки). Разветвление закончилось.

4. Пункт 3 повторить еще один раз, только теперь сравниваем первое число с третьим.

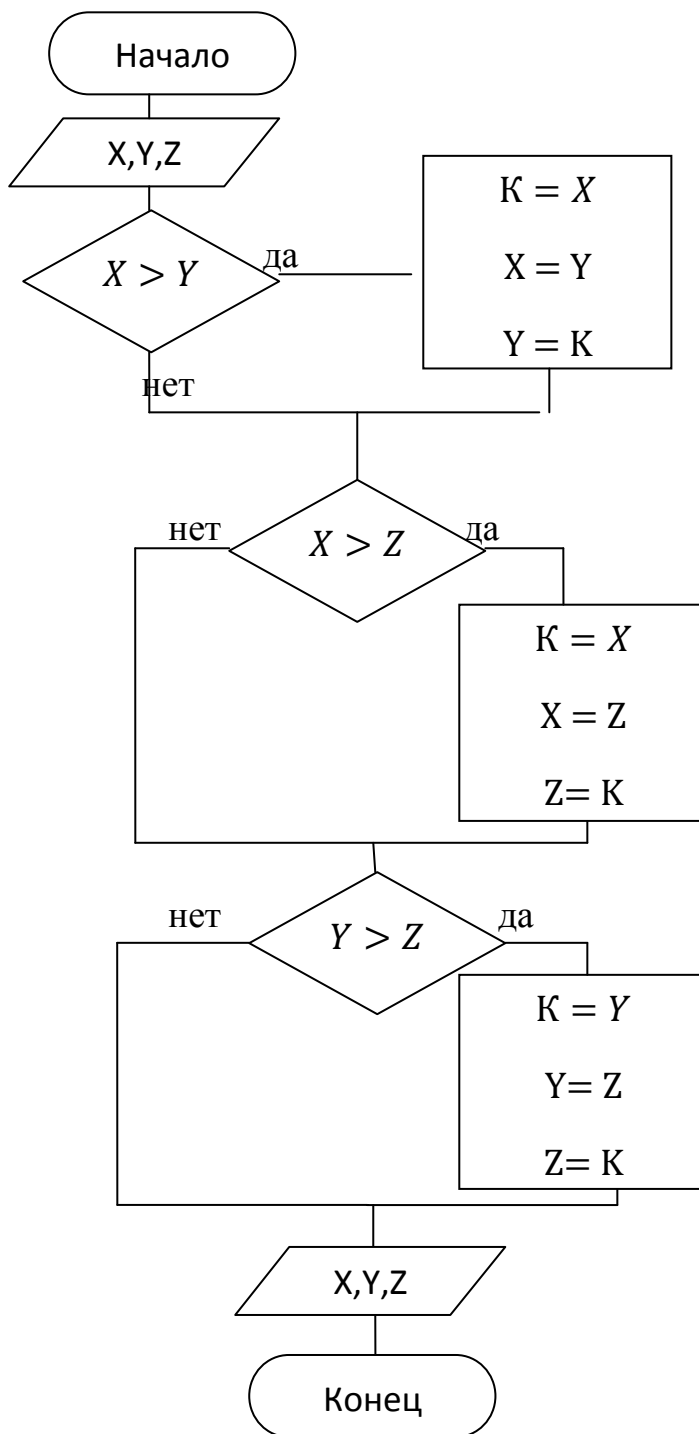


Рис. 19. Блок схема решения задания примера 9

5. Пункт 3 повторить еще один раз, только теперь сравниваем второе число с третьим.

6. Выводим числа после перестановок.

**Пример 10.** Вычислить

$$y = \begin{cases} x, & \text{если } -3 < x < 3 \\ x^2, & \text{если } 3 \leq x < 5 \\ x^3, & \text{если } x \geq 5 \end{cases}$$

Блок схема алгоритма решения приведена на рис. 20.

Алгоритм:

1. Вводим любое число  $X$ .
2. Начинается ветвление. Проверяем условие  $-3 < x < 3$ .
3. Если условие пункта 2 – истина, то функция  $Y$  становится равной  $X$ .
4. Выводим результат вычислений.
5. Если условие пункта 2 – ложь, то проверяем условие  $3 \leq x < 5$ .
6. Если условие пункта 5 – истина, то функция  $Y$  становится равной  $x^2$ .
7. Выводим результат вычисления.
8. Если условие пункта 5 при вычислении даст – ложь, то проверяем условие  $x \geq 5$ .
9. Если в результате проверки условия пункта 8 – истина, функция станет равной  $x^3$ .
10. Выводим результат вычислений.
11. Если и условие пункта 8 в результате проверки даст – ложь, то нужно вывести сообщение, что решений у задачи нет при таком  $X$ . Разветвление закончилось. Все ветви пришли в одну точку.

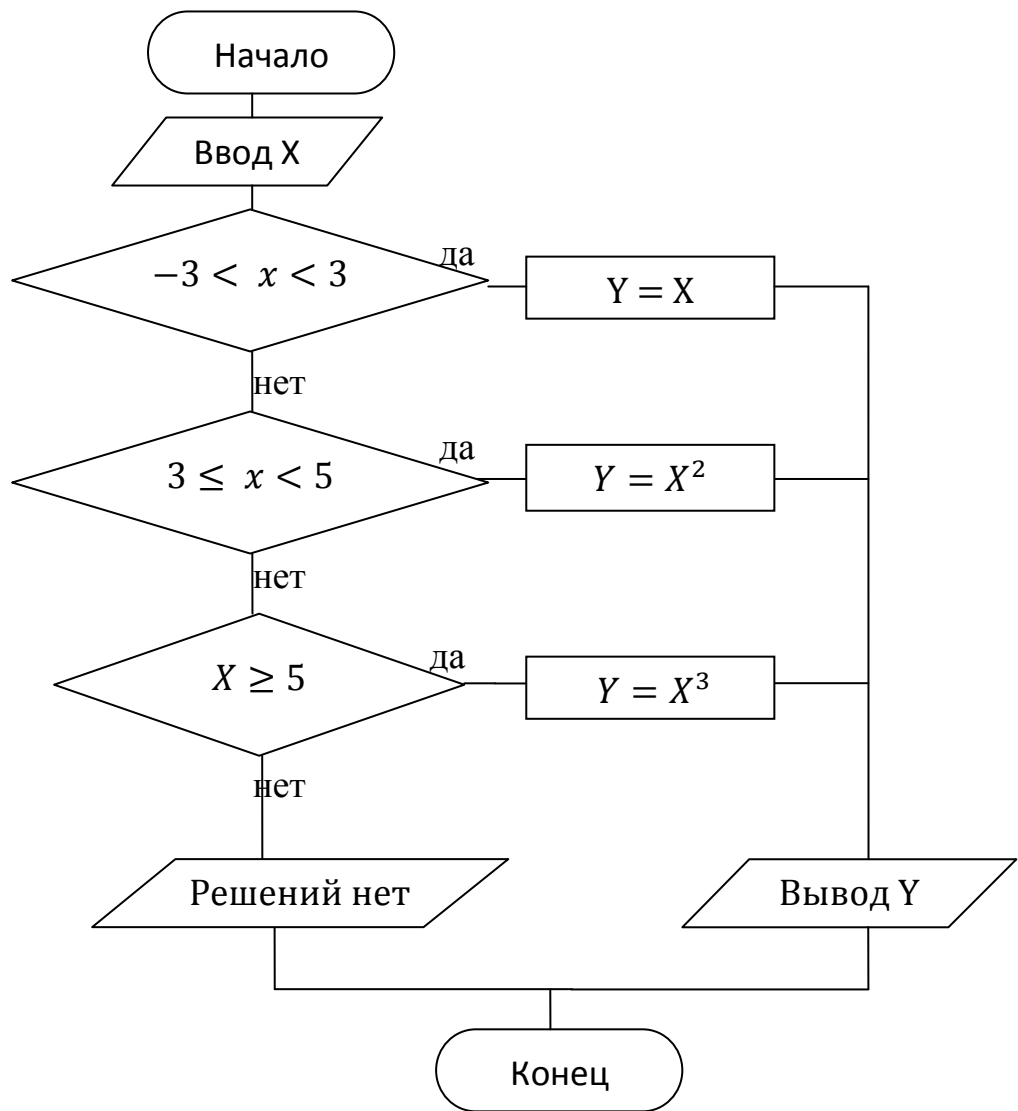


Рис. 20. Блок схема алгоритма решения задания примера 10

Еще один вариант алгоритма решения этой же задачи. Блок схема приведена на рис. 21.

Отличается от предыдущего только порядком проверки условий.

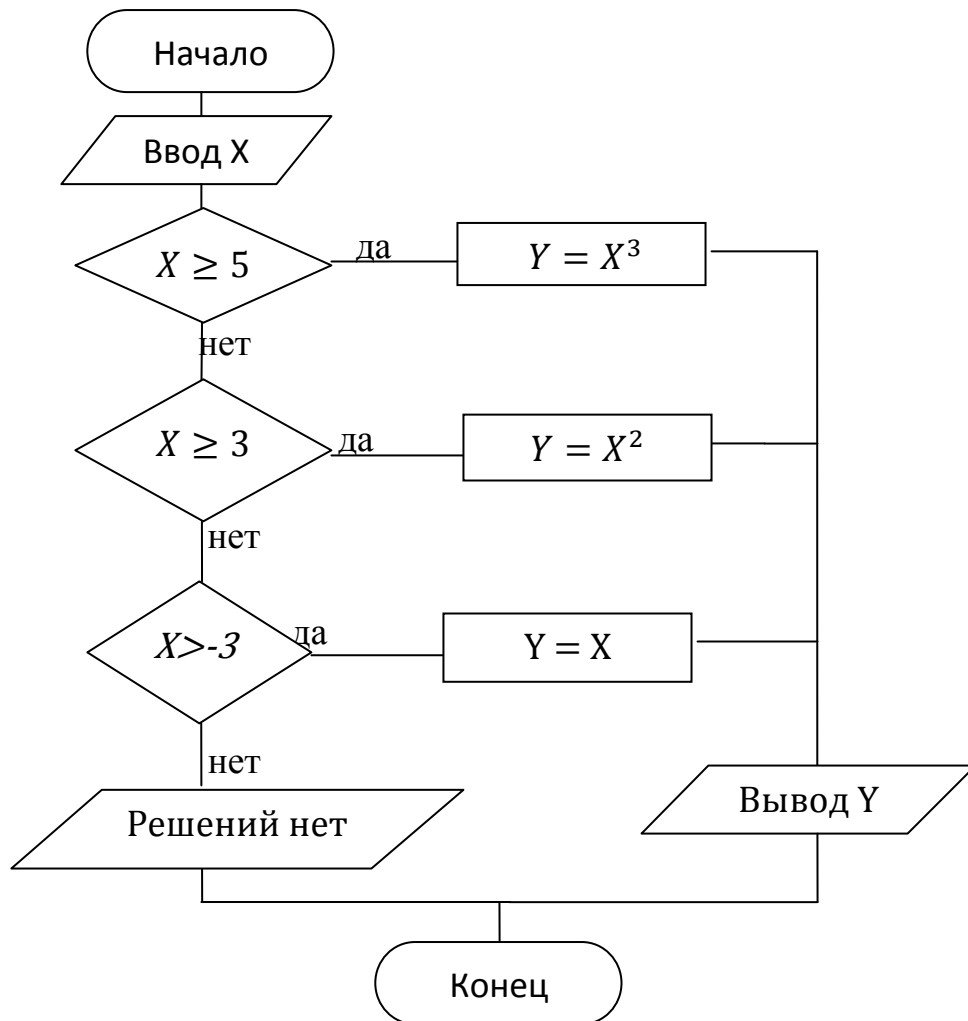


Рис. 21. Блок схема алгоритма решения задания примера 10

**Пример 11.** Даны вещественные числа  $x$  и  $y$ . Определить, принадлежит ли точка с координатами  $(x, y)$  заштрихованной области (рис. 22.).

Заданная область ограничена линиями  $x = -1$ ,  $x = 3$ ,  $y = -2$ ,  $y = 4$ .

Значит, точка с координатами  $(x, y)$  будет принадлежать заданной области, если будут выполняться следующие условия

$$x \geq -1, x \leq 3, y \geq -2, y \leq 4.$$



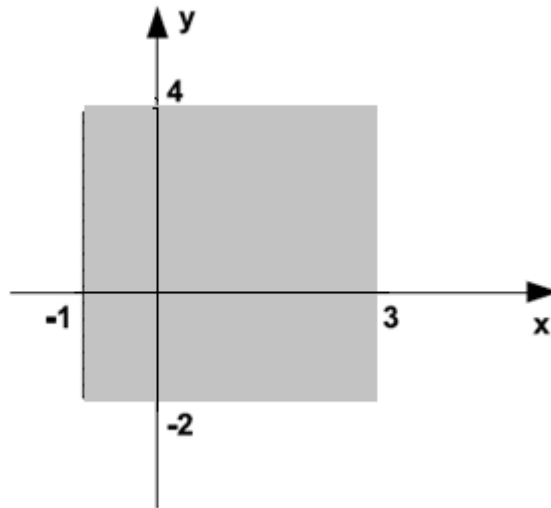


Рис. 22

Блок схема алгоритма решения приведена на рис .23.

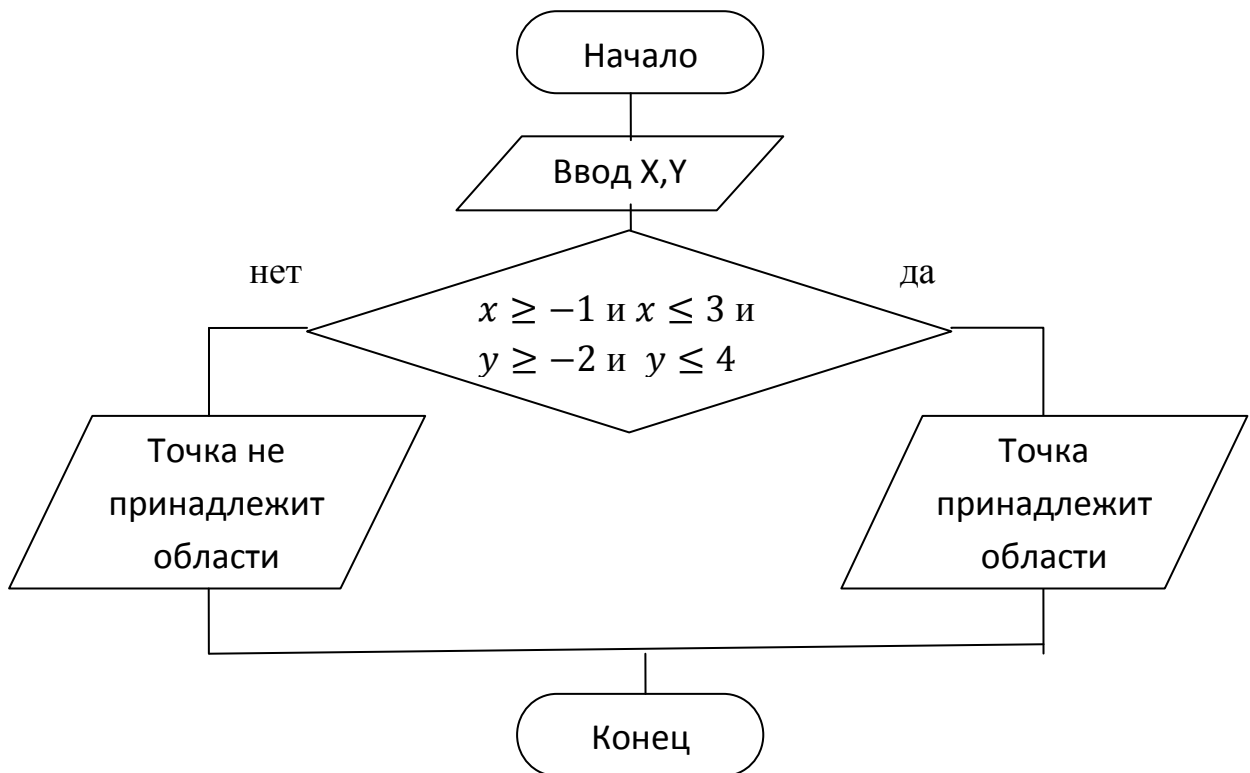


Рис .23. Блок схема алгоритма решения задания примера 11

Исполняемые действия алгоритма:

1. Вводятся координаты точки  $x$ ,  $y$ .

2. Начало разветвления. Проверяется условие  $x \geq -1$  и  $x \leq 3$  и  $y \geq -2$  и  $y \leq 4$ .

3. Если это условие будет истинным, то выводим сообщение - точка принадлежит заданной области.

4. В противном случае выводим сообщение – точка не принадлежит области. Разветвление закончено. Все ветви, образовавшиеся в процессе ветвления, пришли в одну точку.

**Пример 12.** Даны вещественные числа  $x$  и  $y$ . Определить, принадлежит ли точка с координатами  $(x, y)$  закрашенной области (рис. 24).

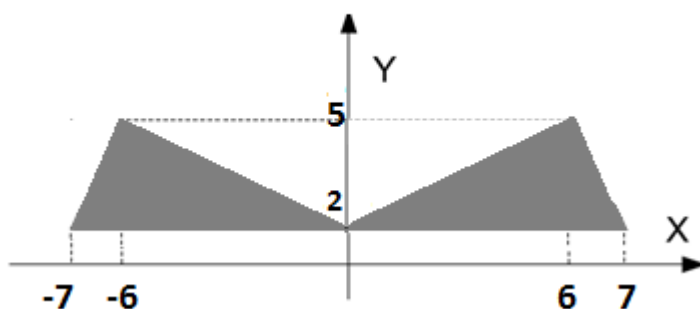


Рис. 24

Составим уравнения линий, ограничивающих заданные области. В общем виде уравнение прямой, проходящей через точки с координатами  $(x_1, y_1)$  и  $(x_2, y_2)$ , имеет вид:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$$

Треугольник в первой четверти координатной плоскости ограничен линиями, проходящими через точки

1.  $(0,2) - (6,5)$
2.  $(6,5) - (7,2)$
3.  $(7,2) - (0,2)$

Следовательно, уравнение линий, его образующих:

– первая:

$$\frac{x - 0}{6 - 0} = \frac{y - 2}{5 - 2}, \quad y = 2 + \frac{1}{2}x$$

– вторая:

$$\frac{x - 6}{7 - 6} = \frac{y - 5}{2 - 5}, \quad y = -3x + 23$$

– третья:

$$y = 2$$

Линии? образующие второй треугольник? проходят через точки:

1.  $(0,2) - (-6,5)$

2.  $(-6,5) - (-7,2)$

3.  $(-7,2) - (0,2)$

Следовательно, уравнение линий, его образующих :

– первая;

$$y = 2 - \frac{1}{2}x$$

– вторая:

$$y = 3x + 23$$

– третья:

$$y = 2$$

Условие попадания точки в область

$$\left\{ \begin{array}{l} y \leq 2 + \frac{1}{2}x \\ y \leq -3x + 23 \\ y \leq 2 \end{array} \right. \text{ или } \left\{ \begin{array}{l} y \leq 2 - \frac{1}{2}x \\ y \leq 3x + 23 \\ y \leq 2 \end{array} \right.$$

Исполняемые действия алгоритма

1. Вводим координаты точек – вершины треугольников.

2. Начинается разветвление. Если выполняется условие попадания точки в область – вывести сообщение – точка в области.

3. В противном случае вывести сообщения – точка не попала в область. Разветвление закончено.

Блок схему изобразить самостоятельно.

### Задания для самостоятельной работы

1. Дано вещественное число  $B$ . Вычислить значения функции в заданной точке  $y = f(B)$ . Графики функции  $y = f(B)$  приведены на рис. 25-48.

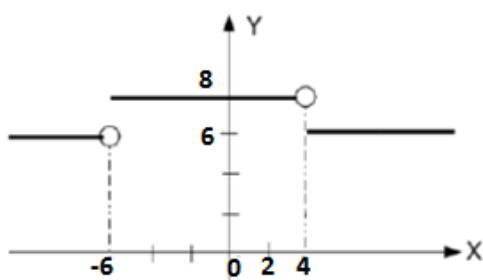


Рис. 25

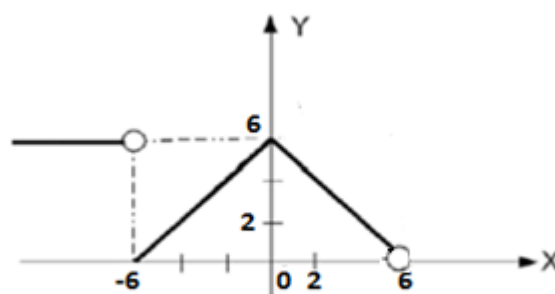


Рис. 26

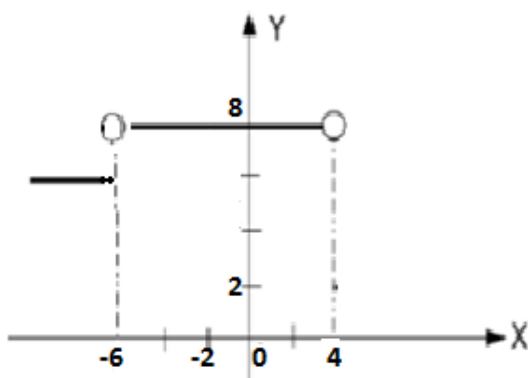


Рис. 27

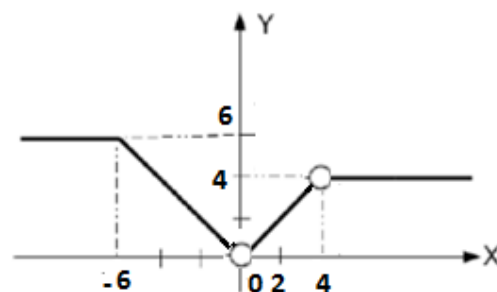


Рис. 28

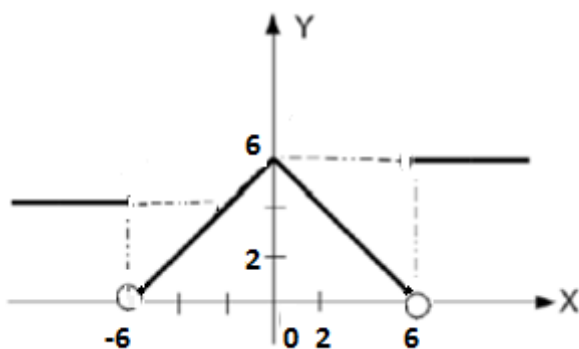


Рис. 29

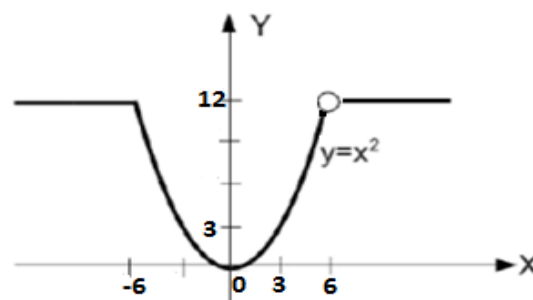


Рис. 30

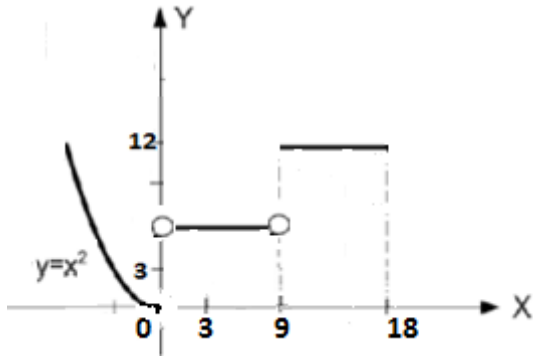


Рис. 31

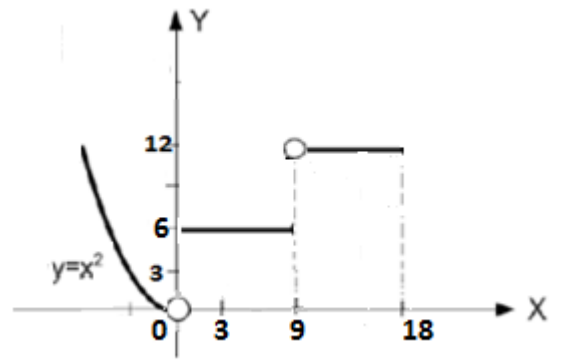


Рис. 32

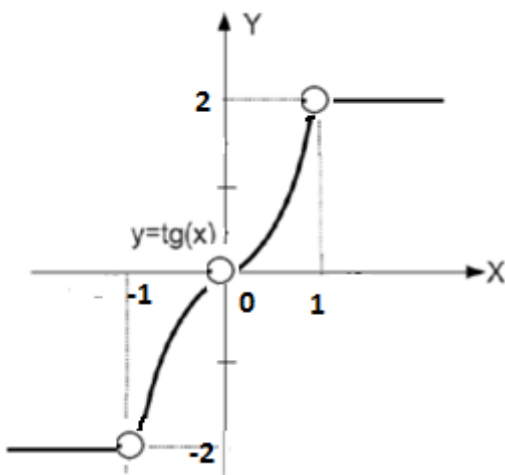


Рис. 33

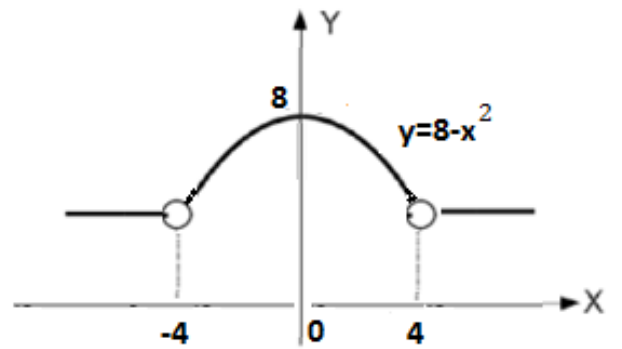


Рис. 34

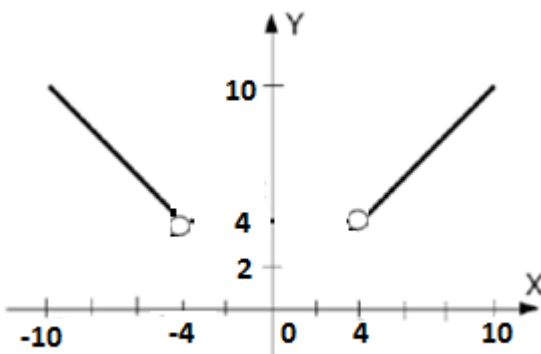


Рис. 35

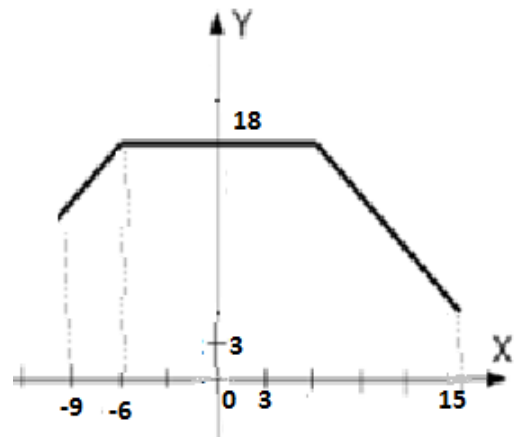


Рис. 36

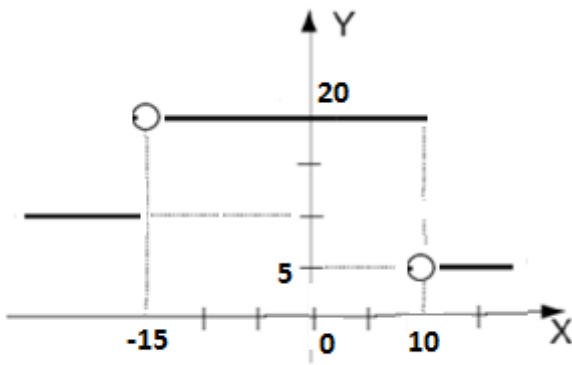


Рис. 37

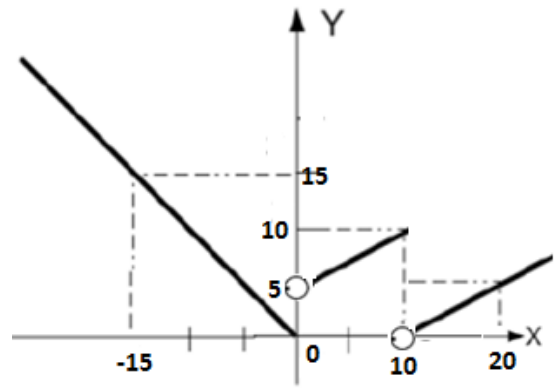


Рис. 38

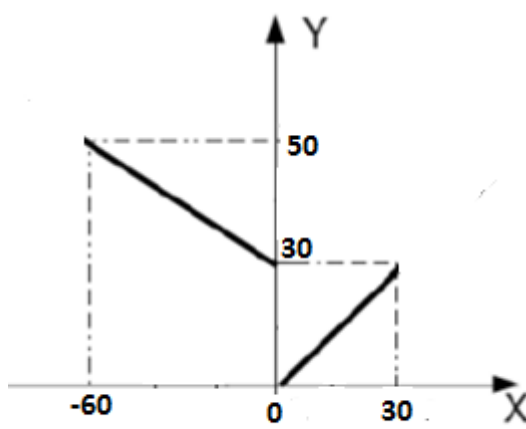


Рис. 39

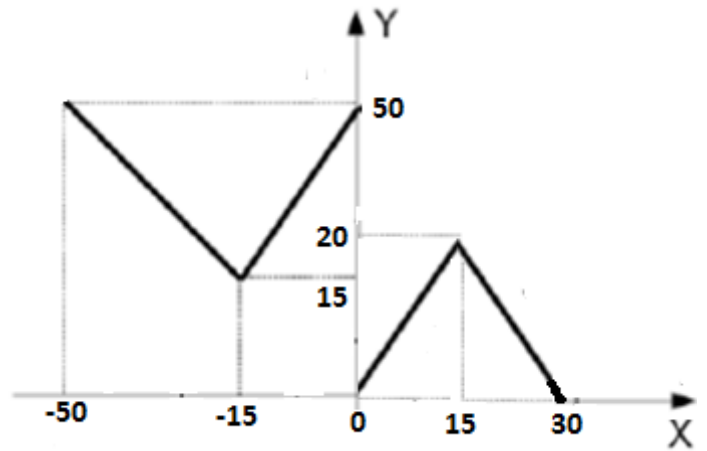


Рис. 40

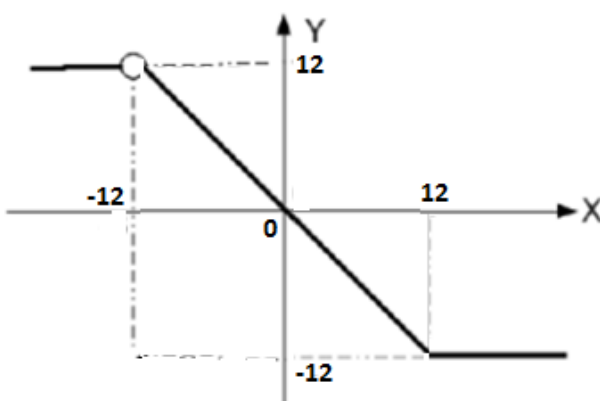


Рис. 41

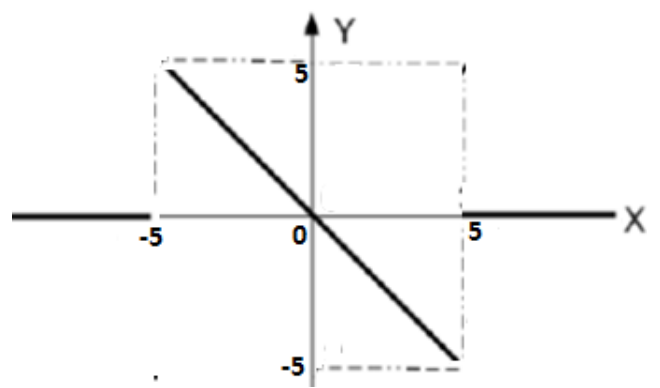


Рис. 42

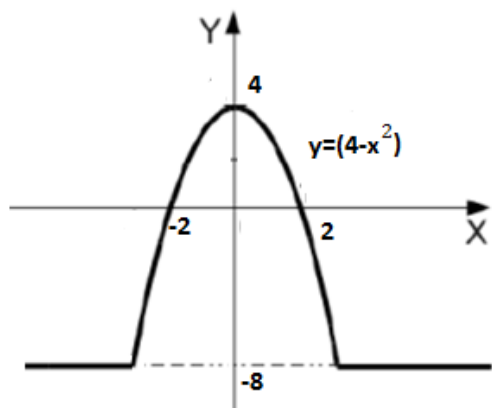


Рис. 43

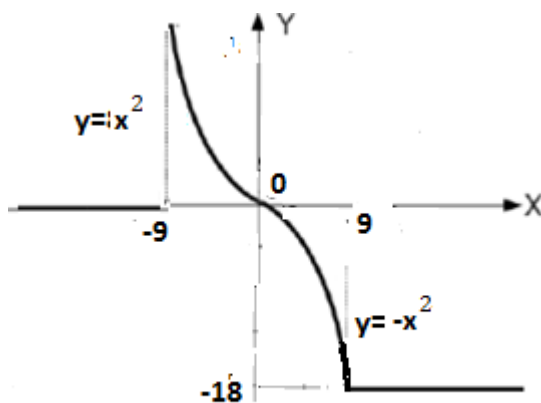


Рис. 44

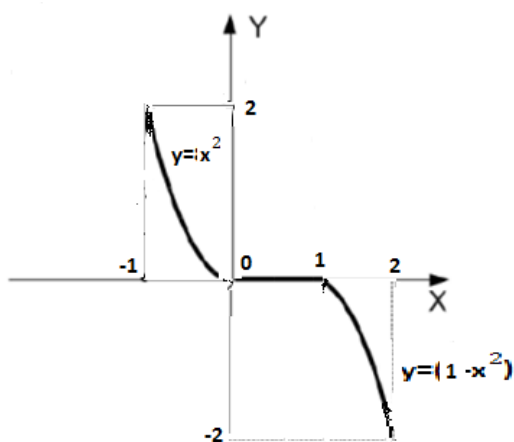


Рис. 45

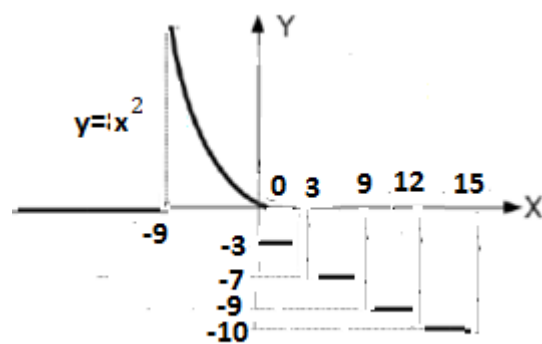


Рис. 46

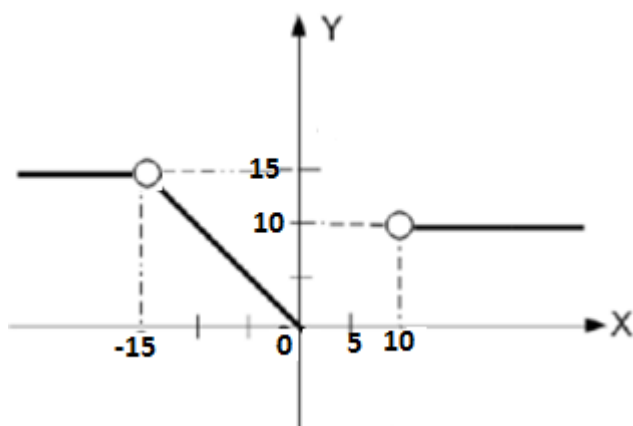


Рис. 47

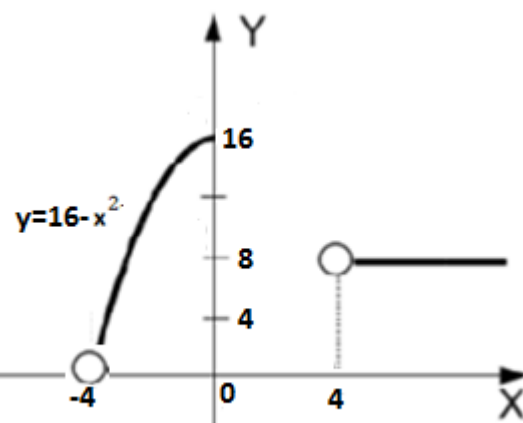


Рис. 48

2. Дана точка A с координатами (x,y) на плоскости. Определить, принадлежит ли эта точка закрашенной области. Варианты заданий на рис. 49-57.

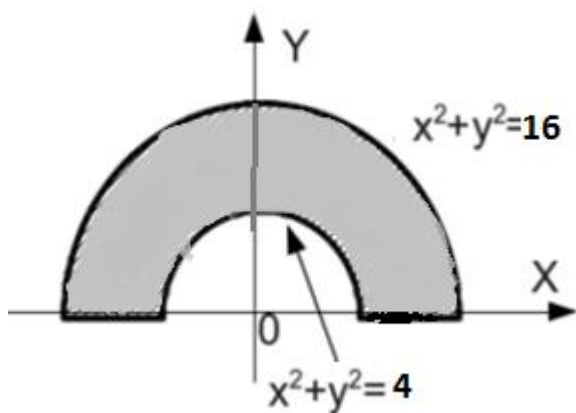


Рис.49

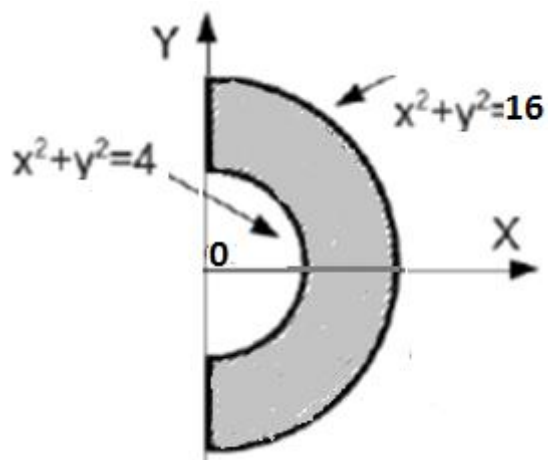


Рис. 50

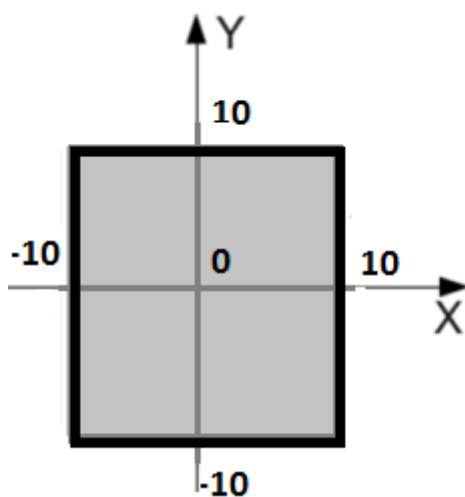


Рис. 51

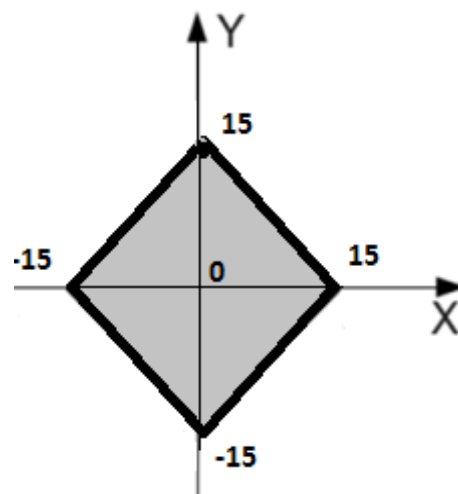


Рис. 52

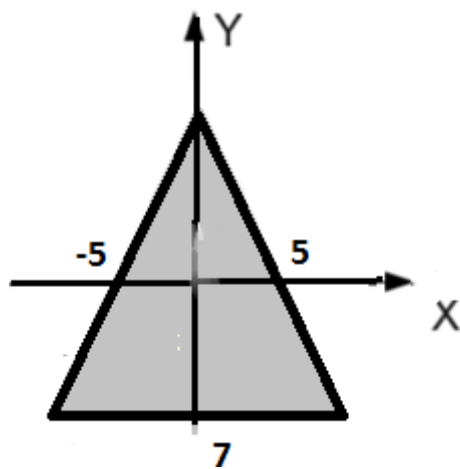


Рис. 53

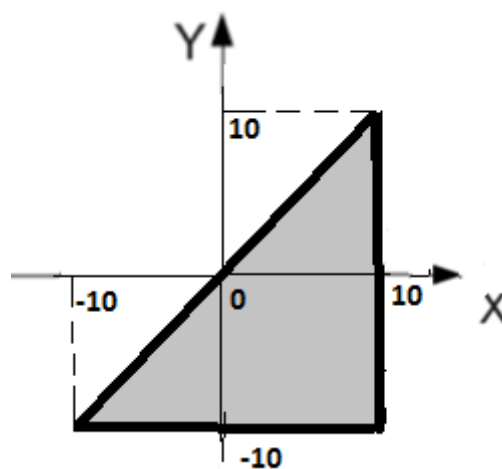


Рис. 54



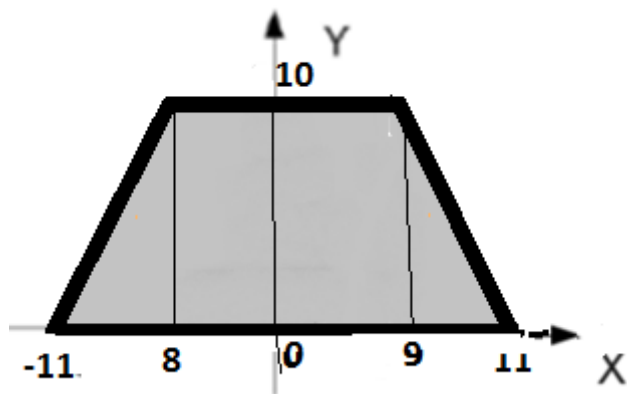


Рис. 55

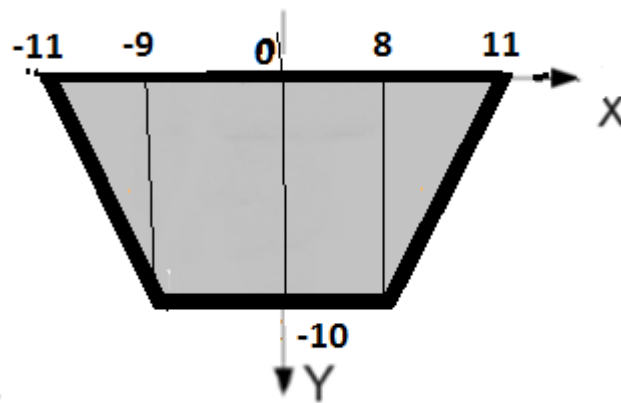


Рис. 56

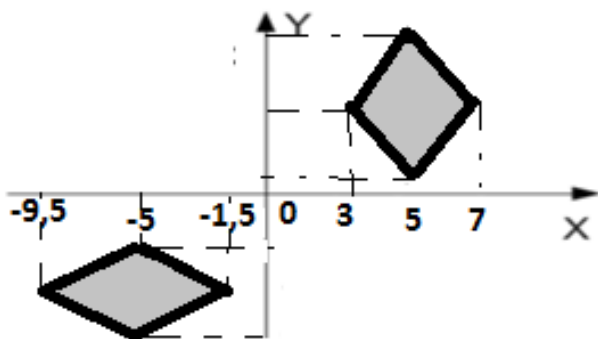


Рис. 57

3. Задана окружность. Центр окружности – в точке  $(x, y)$ , ее радиус –  $r$ . Определить, пересекается ли заданная окружность с осью абсцисс и если пересекается – найти координаты точек пересечения.

4. Задана окружность. Центр окружности – в точке  $(x, y)$ , ее радиус –  $r$ . Определить, пересекается ли заданная окружность с осью ординат, и если пересекается – найти координаты точек пересечения.

5. Вычислить функцию:

$$Z = \begin{cases} \frac{3X + 5Y}{X^2 + Y^2}, & \text{если } x^2 + y^2 \geq 1 \\ 7X^3 + 2Y + X, & \text{если } x^2 + y^2 < 1, \quad x \geq 0 \\ 3Y^3 - 5Y + X, & \text{если } x^2 + y^2 < 1, \quad x < 0 \end{cases}$$

6. Вычислить функцию:

$$Y = \begin{cases} 3\lg\left(\frac{x}{2} + 1\right), & \text{если } x > 2 \\ e^x - 2x^2, & \text{если } x < 2 \end{cases}$$

### Циклический алгоритм

Циклический вычислительный процесс представляет собой алгоритмическую структуру, реализующую многократное повторение некоторой совокупности действий. Совокупность действий, которую необходимо многократно повторить, называется телом цикла. В качестве тела цикла могут выступать линейные, ветвящиеся или другие циклические структуры, а также сочетания этих структур.

Простыми называются циклы, которые не содержащие внутри себя других циклов. Циклы, которые содержат внутри себя, по крайней мере, хотя бы еще одну циклическую структуру называют вложенными. При этом циклы, тело которых содержит другие циклы, называются внешними, а циклы, входящие в тело внешних циклов, – внутренними.

Циклы бывают с переадресацией и без переадресации.

Циклы без переадресаций по способу задания данных бывают 2 видов: итерационными и с заданным числом повторений

Итерационными называются циклы, число повторений которых заранее (т.е. из условия задачи) не известно. Выход из итерационного цикла осуществляется в случае выполнения (невыполнения) некоторого заданного условия. При каждом новом повторении такого цикла происходит последовательное приближение к искомому результату и проверка условия его достижения.

При разработке итерационного цикла требуется следить за тем, чтобы значение проверяемого условия окончания (продолжения) цикла было обязательно достигнуто в ходе исполнения циклического процесса. В противном случае циклический процесс будет исполняться бесконечно, т.е. произойдет заикливание программы.

По способу организации проверки условия продолжения (окончания) цикла различают две разновидности итерационных циклов:

- с проверкой условия продолжения цикла до реализации тела цикла;
- с проверкой условия окончания цикла после реализации тела цикла.

### **Циклы с проверкой условия продолжения цикла до реализации тела цикла**

Циклы с проверкой условия до выполнения тела цикла называют циклами с предшествующим условием или циклы типа “пока”.

Они предписывают выполнять тело цикла до тех пор, пока выполняется некое условие, называемое условием продолжения цикла. В общем виде на блок-схеме это представлено на рис. 58.

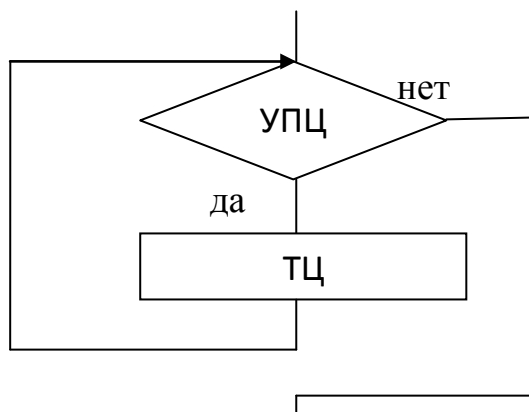


Рис. 58. Блок схема цикла с предшествующим условием

Читать эту структуру нужно так:

Пока УПЦ (условие продолжения цикла) истина (да), необходимо выполнять ТЦ (тело цикла) в противном случае, т.е. когда условие продолжения цикла ложь (нет) цикл заканчивает свою работу (выход из цикла). Особенность цикл в том, что он может не выполниться ни разу.

Чтобы избежать заикливания необходимо следить, чтобы как минимум одна из величин, присутствующих в условии продолжении цикла, меняла свое значение в теле цикла.

При решении любой задачи на циклы следует придерживаться такой тактики, перед началом цикла всегда описывается состояние всех величин алгоритма в начальный момент времени, а в теле цикла они могут изменять свои значения от этого начального состояния.

**Пример 13.** Вычислить функцию  $y = x^2$  в точках заданного интервала  $x \in [a. b]$ , точки на интервале выбираются с заданным шагом  $h$ . Блок схема алгоритма 1 решения приведена на рис. 59.

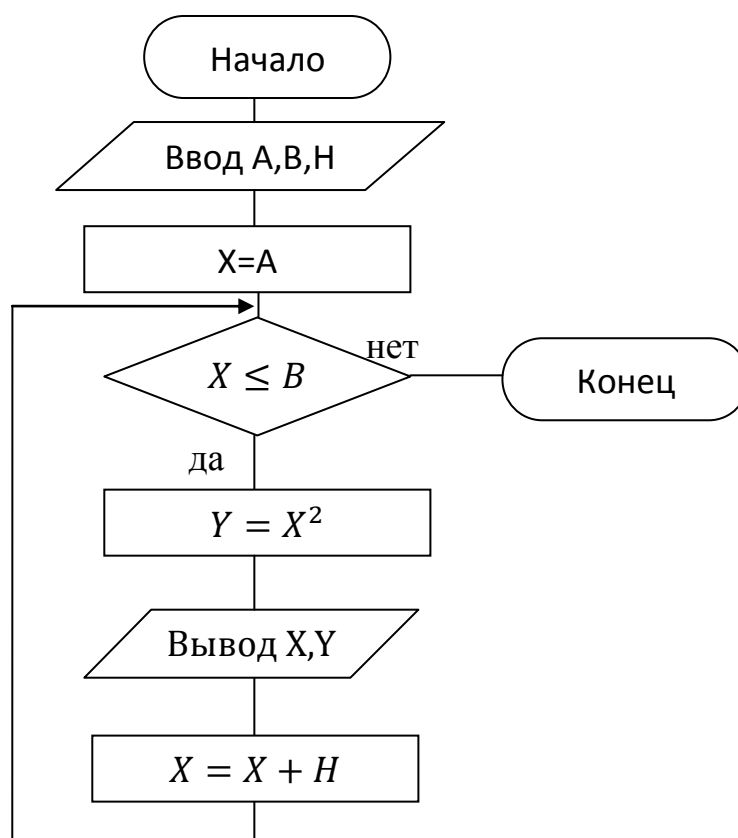


Рис. 59. Блок схема алгоритма 1 решения задания примера 13

Обозначения:

$A, B$  – начало и конец заданного интервала;

$H$  – шаг, с которым выбирается точка на интервале;

$X$  – текущая точка на интервале;

$Y$  – значение функции вычисленное в текущей точке интервала.

Алгоритм 1 (цикл с предшествующим условием).

Решать задачу этим способом целесообразно, если автор алгоритма хочет быть уверен, чтобы ошибки при вводе данных не повлияли на ход выполнения алгоритма, т.е. например, пользователь перепутает при вводе начало и конец отрезка. Если такое случится, алгоритм сразу завершит свою работу.

1. Ввести  $A, B, H$ .

2. Вычислим текущую точку на отрезке (целесообразно выбрать ее в начале отрезка). Т.е. описываем начальный момент времени, состояние переменных до цикла.

3. Начинается цикл с предшествующим условием, т.е. сразу для первой точки проверяется условие продолжение цикла (принадлежит ли выбранная точка отрезку, для этого ее нужно сравнить со вторым концом отрезка), а именно: «текущая точка меньше или равна конца заданного отрезка».

4. Если условие пункта 3 – истина, то выполняем тело цикла, которое в данном случае содержит три действия (три следующих пункта алгоритма 5-7).

5. Вычисляется функция  $Y$  в текущей точке.

6. Выводится вычисленный результат  $(X, Y)$ .

7. Вычисляется следующая текущая точка  $X$ , которая находится на расстоянии  $H$  от предыдущей текущей точки  $X$ .

8. Все циклически повторяется, начиная с пункта 3 (стрелка вверх). Т.е. для текущей точки опять проверяется условие пункта 3.

9. Цикл завершается, когда условие, вычисленное в пункте 3 – станет ложь.

10. Конец алгоритма.

## Циклы с проверкой условия окончания цикла после реализации тела цикла

Циклы с проверкой условия окончания цикла после тела цикла называют циклом с последующим условием. Они предписывают завершить выполнения цикла в случае истинности некоего условия, называемого условием окончания цикла.

В общем виде на схеме эта структура представлена на рис. 60.

Здесь:

ТЦ – тело цикла;

УОЦ – условие окончания цикла.

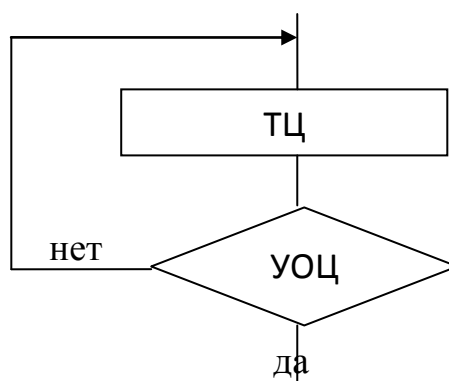


Рис. 60. Блок схема цикла с последующим условием

Отличается от предыдущей конструкции тем, что тело цикла обязательно выполнится один раз. Цикл завершится, когда УОЦ станет истиной.

Чтобы избежать заикливания необходимо следить, чтобы как минимум одна из величин присутствующих в условии окончания цикла меняла свое значение в теле цикла

**Пример 13а).** Задание, рассмотренное в примере 13 может иметь несколько алгоритмов решения. Приведем второй алгоритм решения задачи 13.

Вычислить значения функции  $y = x^2$  в точках заданного интервала  $x \in [a, b]$ , точки на интервале выбираются с заданным шагом  $h$ .

Теперь пусть это будет алгоритм с применением циклов с последующим условием. Даже если пользователь введет в качестве начала и конца отрезка одно и то же значение, то все равно ведь одна точка на отрезке есть, и, значит, можно воспользоваться этим типом циклов.

Обозначения этого алгоритма такие же, как в задаче 13.

Блок схема алгоритма 2 решения приведена на рис. 61.

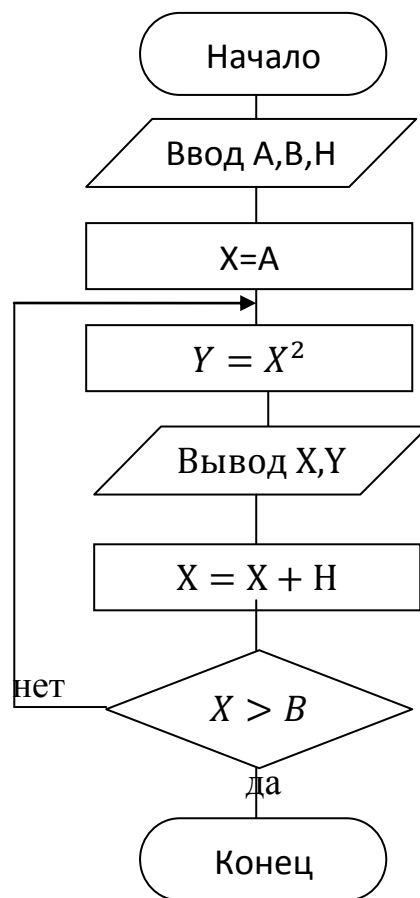


Рис. 61. Блок схема алгоритма 2 решения задания примера 13а)

Алгоритм 2 для задачи 13:

1. Ввести А,В,Х

2. Вычислим текущую точку на отрезке (целесообразно выбрать ее в начале отрезка). Т.е. вычисляем значение  $X$  в начальный момент времени, до цикла.

3. Начинается цикл с последующим условием. Его начало на схеме (конец стрелки  $\longrightarrow$ , т.е. с этого места начинается повтор, в это место вернулась стрелка, указывающая следующее действие алгоритма) Вычисляется функция  $Y$  в текущей точке  $X$ .

4. Выводится вычисленный результат  $(X, Y)$ .

5. Вычисляется следующая текущая точка  $X$ , которая находится на расстоянии  $H$  от предыдущей текущей точки  $X$ .

6. Проверяется условие окончания цикла: «текущая точка вышла за конец интервала»

7. Если условие пункта 6 – ложь, все циклически повторяется, начиная с пункта 3 (стрелка вверх).

8. Цикл завершается, когда условие в пункте 4 – станет истиной.

9. Конец алгоритма.

**Пример 14.** Рассмотрим известный алгоритм нахождения НОД (наибольшего общего делителя) двух натуральных чисел (алгоритм Евклида).

Наибольший общий делитель (НОД) – это число, которое делит без остатка два натуральных числа и делится само без остатка на любой другой делитель данных двух чисел. Проще говоря, это самое большое число, на которое можно без остатка разделить два числа, для которых ищется НОД.

Математическая модель  $\text{НОД}(A, B) = \text{НОД}(\min(A, B), [A - B])$ .

Обозначения:

$A, B$  – заданные натуральные числа.



Алгоритм 1 (цикл с предшествующим условием).

Алгоритм применим к любым натуральным числам и за конечное число шагов должен приводить к решению.

Поскольку количество повторений заранее неизвестно, в алгоритме следует применить цикл "пока" с предшествующим условием:

1. Ввести два натуральных числа и начать цикл с предшествующим условием.

2. Проверим условие продолжения этого цикла «числа не равны?»

3. Начинается разветвление в цикле. Определяем большее из чисел и заменяем его разностью большего и меньшего из чисел. Разветвление закончено.

4. Повторить алгоритм с шага 2.

5. Если условия пункта 2 – ложь, т.е., если числа равны, то взять любое из них и есть искомый ответ, цикл завершается.

6. Вывести вычисленный результат.

7. Конец алгоритма.

Блок схема алгоритма 1 представлена на рис. 62.

Алгоритм 2 (нахождения НОД делением):

1. Задать два числа.

2. Большее число делим на меньшее.

3. Если делится без остатка, то меньшее число и есть НОД (следует выйти из цикла).

4. Если есть остаток, то большее из чисел заменяем на остаток от деления.

5. Переходим к пункту 2.

Например:

Найти НОД для 30 и 18.

$$30/18 = 1 \text{ (остаток 12)}$$

$18/12 = 1$  (остаток 6)

$12/6 = 2$  (остаток 0). Конец: НОД – это делитель. НОД (30, 18) = 6

Предлагается этот алгоритм изобразить графически самостоятельно.

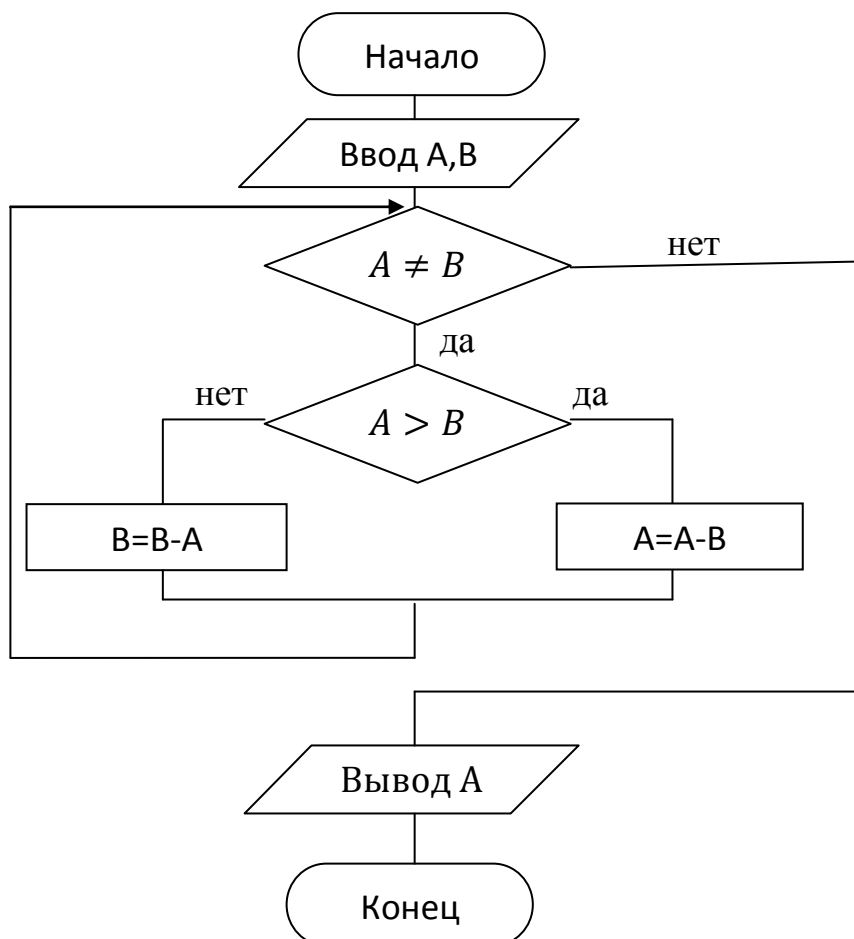


Рис. 62. Блок схема алгоритма 1 решения задания примера 14

Рассмотрим еще один пример итерационного цикла

**Пример 15.** Вычислить:

$$S = 1 + x + \frac{x^2}{2!} + \dots + \frac{x^n}{n!} + \dots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

Задача на суммирование – типичная циклическая задача

Известно свойство сходящихся степенных рядов – любое слагаемое ряда превышает оставшуюся бесконечную сумму «хвоста» ряда. Поэтому

точность вычисления может быть оценена значением очередного слагаемого сходящегося степенного ряда.

Условие окончания суммирования – очередное слагаемое по модулю меньше заданной точности  $\varepsilon$  (т.е. это слагаемое достаточно малое, т.е. практически ничего не добавляющее в сумму), т.е

$$|u_n| = \left| \frac{x^n}{n!} \right| < \varepsilon$$

Здесь  $u_n$  – n-ое слагаемое суммы  $S$

Для «экономии вычислений» степеней, факториалов и других регулярных выражений целесообразно вывести рекуррентную формулу. Она позволит вычислить текущее слагаемое вычисляемой суммы через предыдущее. Это можно сделать аналитически, разделив в общем виде выражение  $u_{n+1}$  на  $u_n$ , где  $u_{n+1}$  это n+1 слагаемое суммы  $S$

$$R = \frac{U_{n+1}}{U_n} = \frac{X^{N+1}N!}{(N+1)!X^N} = \frac{X}{N+1}$$

$R$  – рекуррентный множитель

Т.е.

$$U_{N+1} = \frac{U_n X}{N+1}$$

Блок схема алгоритма решения этой задачи представлена на рис. 63.

Обозначения:

$X$  – любое число;

$\varepsilon$  – точность;

$N$  – номер слагаемого в сумме;

$S$  – текущее значение суммы;

$U$  – текущее слагаемое в сумме.

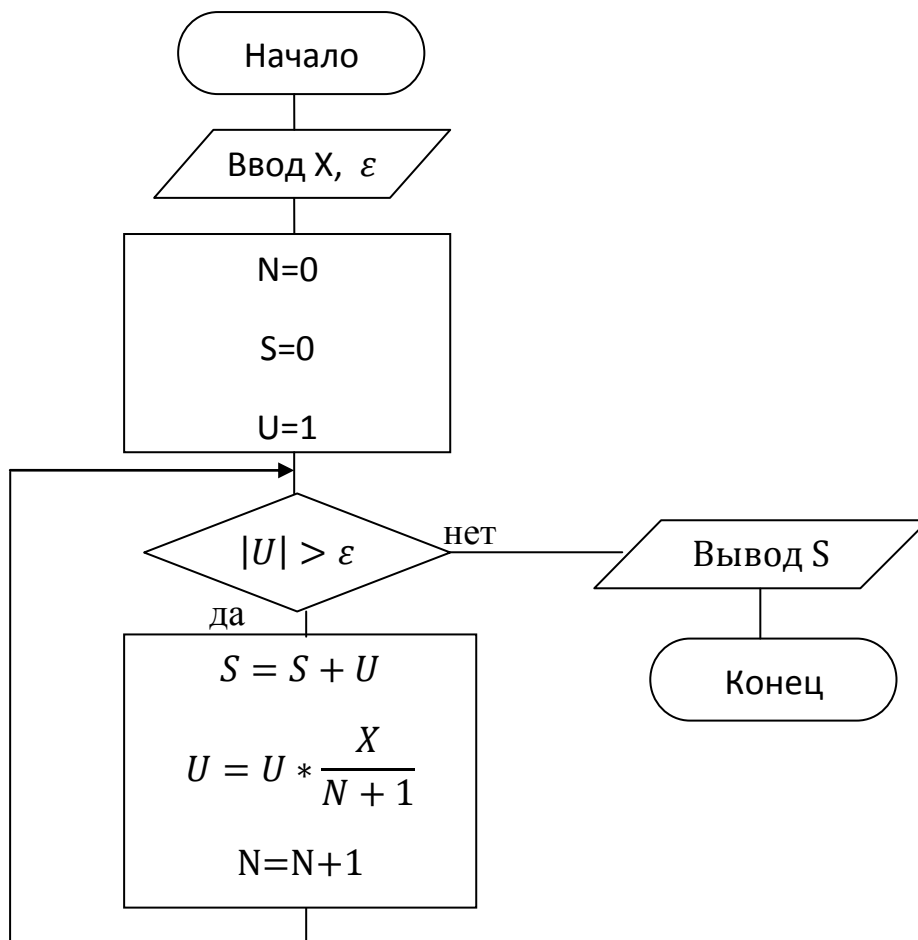


Рис 63. Блок схема алгоритма решения задания примера 15

Алгоритм:

1. Вводим любое число  $X$  и точность  $\varepsilon$ .
2. Вычислим значения  $N$ ,  $S$ ,  $U$  в начальный момент времени (т. е. их состояние до цикла). Номер слагаемого = 0, сумма = 0 (т.е. ее накопление начинается с нуля, т.к. добавляя слагаемые к этому начальному нулю мы не испортим конечный результат), нулевое слагаемое =1 (из заданной в условии формулы суммы).
3. Точка входа в цикл с предшествующим условием. Проверим условие продолжение цикла “текущее слагаемое больше заданной точности” (цикл будет выполняться пока это условие – истина).

4. Если условие пункта 3 – истина (выход «да» на схеме), то выполняем тело цикла, состоящее из трех действий. Текущее слагаемое добавляем к текущей сумме.

5. Вычисляем следующее слагаемое  $U$ , используем вычисленный заранее рекуррентный множитель  $\frac{x}{N+1}$ .

6. Изменяем номер слагаемого на единицу.

7. Все циклически повторяется, начиная с пункта 3 этого алгоритма.

8. Когда результатом вычисления условия пункта 3 станет – ложь цикл останавливается (выход «нет» на схеме). Выводим вычисленную сумму.

9. Конец алгоритма.

### Цикл с параметром

Цикл с заданным числом повторений (с параметром) целесообразно использовать в тех случаях, где количество повторений тела цикла известно из условия задачи. Алгоритм решения задачи будет короче.

В общем виде схема такого алгоритма представлена на рис. 64.

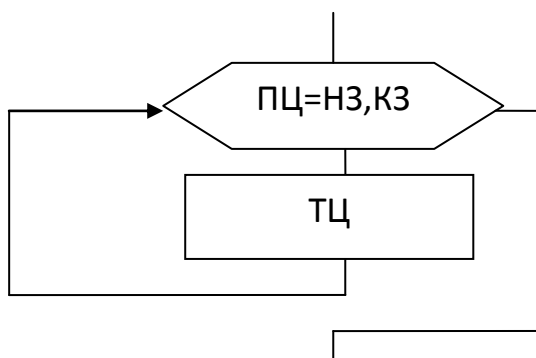


Рис. 64. Блок схема цикла с заданным числом повторений

Здесь:

ПЦ – переменная цикла (параметр цикла, счетчик повторения цикла);

НЗ – начальное значение переменной цикла;

КЗ – конечное значение переменной цикла;

ТЦ – тело цикла.

Алгоритм выполнения цикла:

1. При первичном входе (т.е. сверху) в блок модификации (изменения) переменной цикла, переменной цикла (ПЦ) присваивается начальное значение (НЗ).

2. В блоке модификации переменной цикла проверяется условие, текущее значение переменной цикла меньше или равно конечному значению (КЗ) переменной цикла.

3. Если условие пункта 2 истина, то выполняется тело цикла.

4. Возвращаемся в блок модификации переменной цикла (стрелочка возвращается в блок модификации слева, при этом переменная цикла модифицируется, т.е. изменяет свое значение на единицу).

5. Все циклически повторяется с пункта 2 этого алгоритма.

6. Цикл завершается, когда переменная цикла станет больше, чем конечное значение (выход справа из блока модификации).

**Пример 16.** Вычислить:

$$S = \sum_{i=1}^n \frac{1}{i^2}$$

Сначала применим уже имеющиеся знания, т.е. применим для вычисления этой конечной суммы один из циклов с условием (предшествующим или последующим). Это не является оптимальным выбором цикла для данной задачи, т.к. потребует вычислять начальное значение номера слагаемого  $I=1$ , проверять условие окончания цикла (т.е. не превысил ли этот номер конечного значения  $N$  ( $I \leq N$ )), и увеличивать номер слагаемого  $I=I+1$ , что удлинит алгоритм. Блок-схема этого алгоритма представлена на рис 65.

Обозначения:

N – Количество слагаемых в сумме;

I – номер слагаемого;

S – текущая сумма.

Но, поскольку количество повторений цикла известно из условия задачи, целесообразнее применить цикл с параметром при составлении алгоритма решения этой задачи.

Блок схема с использованием этого типа цикла для решения задачи 16 приведена на рис. 66.

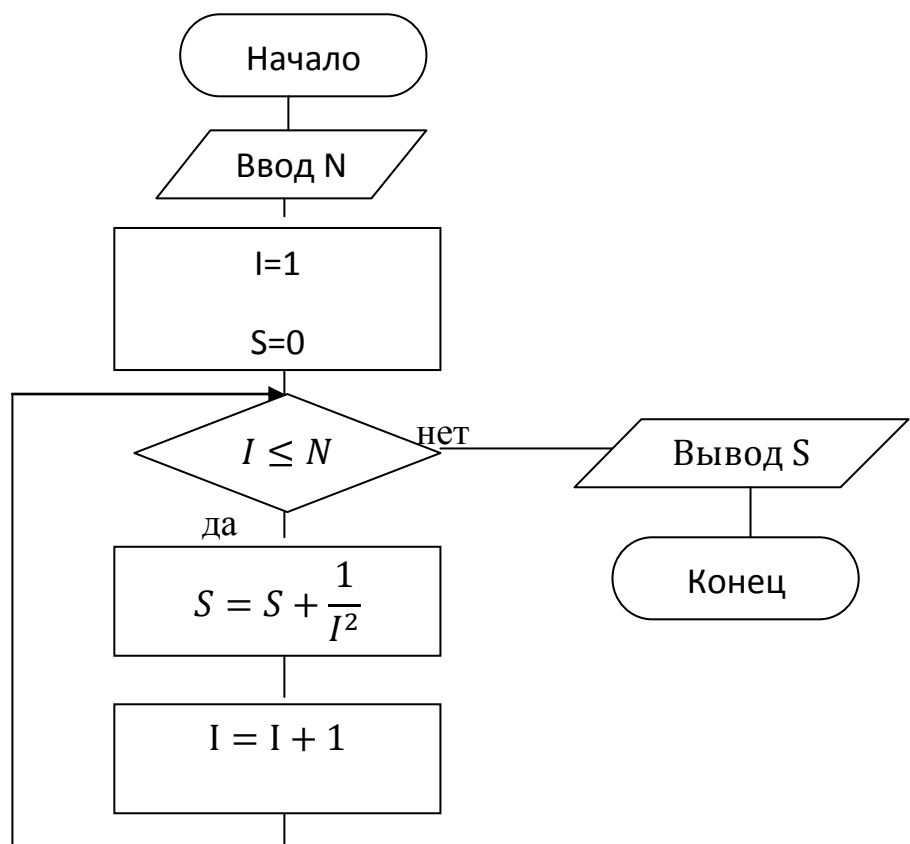


Рис. 65. Блок схема алгоритма решения задания примера 16 с применением цикла с предшествующим условием

Алгоритм (применяем цикл с параметром):

1. Вводим количество слагаемых в сумме – N.
2. Описываем начальное состояние, т.е. обнуляется сумма.

3. Далее цикл с параметром. Параметром цикла (т.е. счетчиком повторений цикла в данном случае является номер слагаемого  $I$ , т.к. цикл надо повторить столько раз, сколько слагаемых мы будем добавлять в сумму). В блоке модификации переменной цикла указываем имя переменной цикла ( $I$ ), ее начальное ( $1$ ) и конечное ( $N$ ) значение. При первичном входе в блок модификации (сверху) переменной цикла  $I$  присвоится значение  $1$ , и пока выполняется условие “текущее значение переменной  $I \leq N$ ”, будет выполняться тело цикла, состоящее из одного действия.

4. Тело цикла. Вычисляем текущую сумму, т.е. добавляем к ней текущее слагаемое. И все повторяется (стрелочка вверх). При возврате в блок модификации переменная цикла модифицируется на  $1$ , и опять проверяется условие пункта 3.

5. Цикл завершается, когда параметр цикла станет больше  $N$  (выход из блока модификации справа). Выводится вычисленный результат.

6. Конец алгоритма.

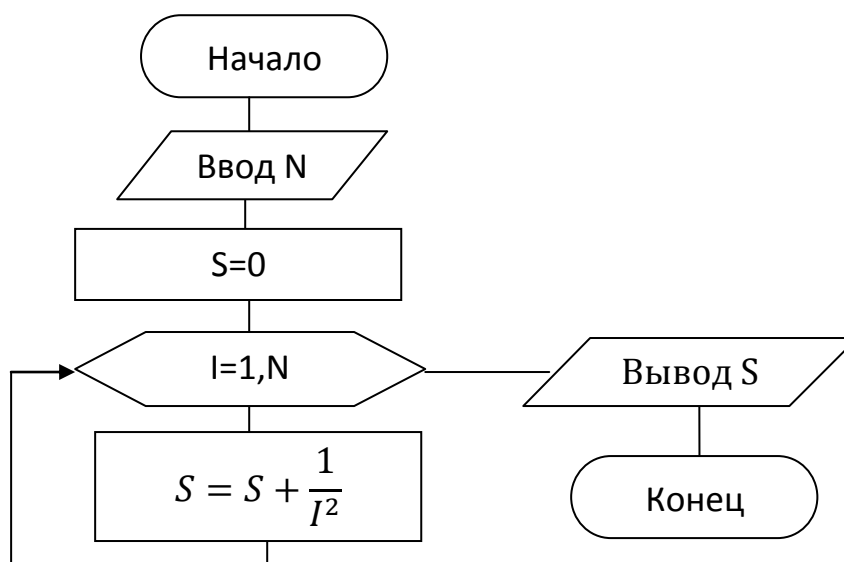


Рис. 66. Блок схема алгоритма решения задания примера 16 с применением цикла с параметром



## Циклы с переадресацией

Особенность этих циклов в том, что они работают с массивами переменных

Массив – упорядоченное множество однородных величин, например коэффициенты полинома, или коэффициенты при неизвестных в системе уравнений.

Рассмотрим систему уравнений

$$\begin{cases} a_{11}x + a_{12}y = b_1 \\ a_{21}x + a_{22}y = b_2 \end{cases}$$

Здесь  $A$  матрица коэффициентов при неизвестных

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

и  $B$  вектор свободных членов

$$B = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Количество измерений массива – количество индексов стоящих у переменной  $a, b$ .

$A$  – двумерный массив, т.е. чтобы указать место его элемента нужно задать два индекса, в какой строке и в каком столбце находится элемент

$B$  – одномерный массив. Для указания места его элемента нужно одно число – порядковый номер элемента.

Рассмотрим полином

$$a_0x^n + a_1x^{n-1} + \dots + a_{n-1}x + a_n$$

$$A = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \\ a_n \end{pmatrix}, \quad B = \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_1 \\ a_0 \end{pmatrix}$$

$A$  и  $B$  – различные массивы, хотя элементы их одинаковы (коэффициенты полинома)

**Пример 17.** Вычислить сумму:

$$S = \sum_{i=1}^n x_i y_i$$

Т.е. вычислить

$$x_1 y_1 + x_2 y_2 + \dots + x_{n+1} y_{n+1}.$$

Обозначения:

X, Y – одномерные массивы;

N – количество слагаемых в сумме, и размерность массивов X и Y;

S – искомая сумма;

i – номер элемента в массиве.

Блок схема алгоритма решения задачи приведена на рис. 67.

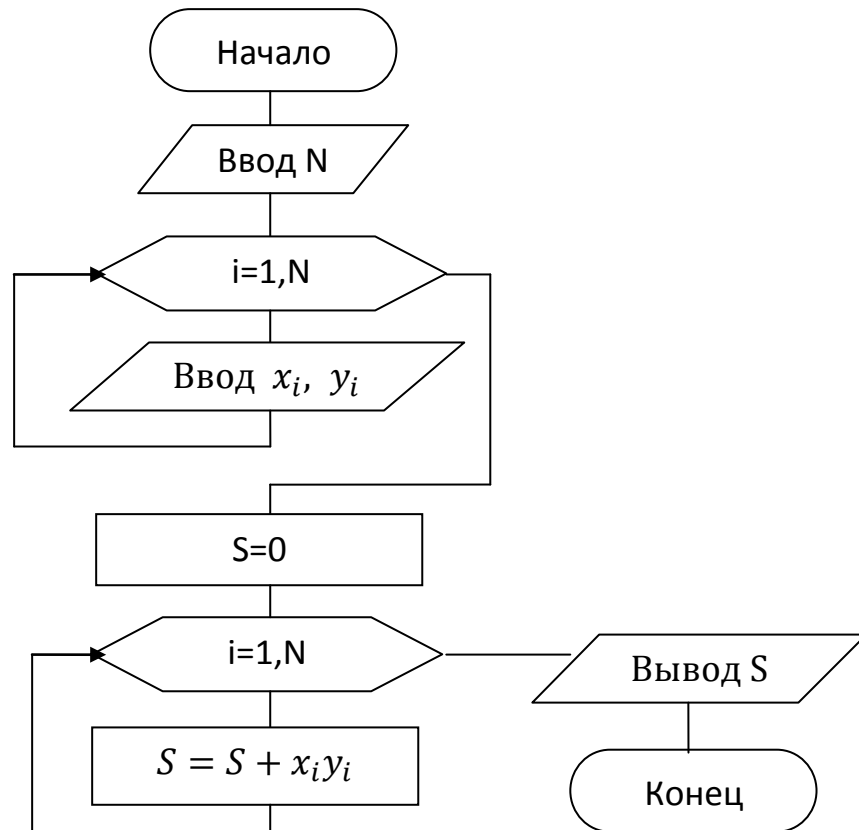


Рис. 67. Блок схема алгоритма решения задания примера 17

Массивы вводятся и выводятся поэлементно, а поскольку количество

элементов массива известно, то целесообразно использовать для этих целей цикл с параметром

Алгоритм решения задачи:

1. Введем размерность массивов  $X$  и  $Y$  (т.е. количество элементов в массивах).

2. Цикл с параметром, который выполнится  $N$  раз (пока счетчик повторения цикла (он же номер элемента в массиве)  $i$ , будет изменяться в диапазоне от 1 до  $N$ ).

3. Тело цикла содержит одно действие ввод элементов массива  $X$  и  $Y$  с номером  $i$  (т.е.  $x_i, y_i$ ).

4. Описываем начальный момент времени – обнуляем сумму.

5. Далее – цикл с параметром, в котором будет накапливаться сумма. Для всех значений параметра цикла  $i$  (номер слагаемого суммы), изменяющегося в диапазоне от 1 до  $N$ , будет выполняться тело цикла.

6. Тело цикла состоит из одного действия – вычисления суммы. При каждом новом проходе цикла к текущей сумме будет добавлять новое слагаемое – произведение элементов массива  $X$  и  $Y$  с номером  $i$ .

7. После окончания цикла (параметр  $i$  стал больше чем  $N$ ), выводим вычисленную сумму  $S$ .

8. Конец алгоритма.

**Пример 18.** Вычислить количество положительных и отрицательных элементов в массиве  $A$  из  $N$  элементов.

Обозначения:

$N$  – размерность массива ;

$i$  – номер элемента в массиве;

$A$  – одномерный массив;

КР – количество положительных элементов в массиве;

КО – количество отрицательных элементов в массиве.

Блок схема алгоритма решения задания приведена на рис. 68.

Алгоритм решения задачи:

1. Введем размерность массива  $A$  (т.е. количество элементов в массиве).

2. Далее – цикл с параметром, который выполнится  $N$  раз (пока счетчик повторения цикла (он же номер элемента в массиве)  $i$ , будет изменяться в диапазоне от 1 до  $N$ , будет выполняться тело цикла.

3. Тело цикла содержит одно действие ввод элемента массива  $A$  с номером  $i$  (т.е.  $A_i$ ).

4. Описываем начальный момент времени – обнуляем количество положительных и отрицательных элементов массива.

5. Далее – цикл с параметром, в котором будем обрабатывать каждый элемент массива  $A$  и выяснять положительный он или отрицательный и в зависимости от этого изменять счетчики положительных или отрицательных элементов массива  $KP$  и  $KO$ . Для всех значений параметра цикла  $i$  (номер слагаемого суммы), изменяющегося в диапазоне от 1 до  $N$ , будет выполняться тело цикла.

6. Тело цикла содержит разветвляющуюся структуру. Разветвление начинается – проверяется условие «элемент массива  $A$  с номером  $i$  ( $A_i$ ) положительный?»

7. Если условие в пункте 6 будет истинным, то счетчик положительных элементов массива  $KP$  увеличим на единицу.

8. Если условие в пункте 6 – ложь, то проверим еще одно условие – «элемент массива  $A$  с номером  $i$  ( $A_i$ ) отрицательный?»

9. Если условие в пункте 8 будет истинным, то счетчик отрицательных элементов массива  $KO$  увеличим на единицу.

10. Если условие в пункте 8 – ложь. То никаких действий производить не требуется. Ветвление закончилось. Т.е. все альтернативные пути решения пришли в одну точку.

11. Алгоритм продолжается – номер элемента массива (переменная цикла) изменяется (увеличивается) на единицу – (стрелка вверх к блоку модификации переменной цикла) и, если, этот номер не превысил  $N$ , то все циклически повторяется, начиная с пункта 6 этого алгоритма.

12. Когда  $i$  станет больше  $N$  – цикл заканчивается – выход справа из блока модификации. Выводим вычисленный результат –  $KP$ ,  $KO$ .

13. Конец алгоритма.

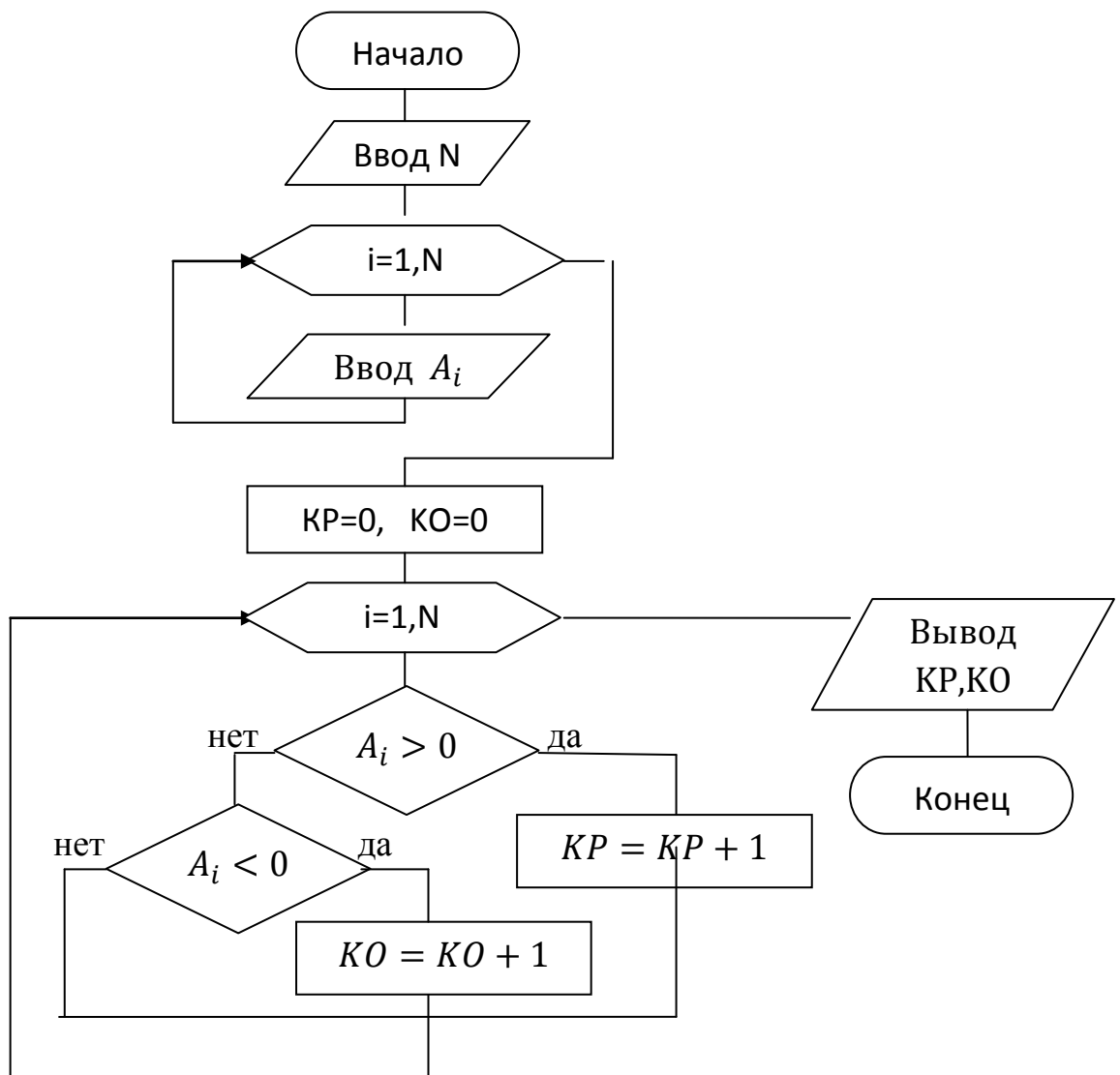


Рис. 68. Блок схема алгоритма решения задания примера 18

**Пример 19.** Найти максимальный из положительных элементов одномерного массива.

Обозначения:

$A$  – массив;

$N$  – количество элементов в массиве;

$i$  – номер элемента в массиве;

$MAX$  – максимальный из положительных элементов массива.

Блок схема алгоритма решения этого задания приведена на рис. 69.

Алгоритм решения:

1. Вводим размерность массива.
2. Цикл с параметром (на схеме блок модификации), который выполнится  $N$  раз. Тело этого цикла содержит одно действие. Организуется цикл для поэлементного ввода массив  $A$ .
3. Тело цикла, описанного в пункте 2 – вводится элемент массива  $A_i$ . Цикл завершится, когда параметр  $i$  станет больше  $N$  (выход справа из блока модификации).
4. Описываем начальный момент времени. Начальному максимуму присваиваем первый элемент массива  $A$ . В этот момент это предположительный максимум и он может быть равен любому элементу массива  $A$ . Но, поскольку, минимальное число  $N$ , которое может ввести пользователь – это 1, то элемент в массиве с таким номером будет всегда, какое бы число  $N$  не было введено пользователем. Значит наше предположение на этом этапе алгоритма о том, что  $MAX = A_1$  будет справедливо для любого введенного массива. Можно было на этом этапе задать начальный максимум равным и  $A_N$  ( $MAX = A_N$ ). Это тоже будет справедливо для любого введенного массива.
5. Цикл с параметром предназначенный для обработки введенного массива (на схеме блок модификации). Он выполнится  $N$  раз (столько

элементов в массиве) и параметр  $i$  (счетчик этих элементов) изменяется от 1 до  $N$ . Тело этого цикла содержит разветвляющуюся структуру.

6. Тело цикла, описанного в пункте 5. Начинается разветвляющаяся структура проверкой логического выражения «элемент массива  $A$  с номером  $i$  положительный?»

7. Если логическое выражение в пункте 6 – истина, то проверим еще одно логическое выражение «элемент массива  $A$  с номером  $i$  больше чем текущий максимум?» (т.к. нам нужно искать максимальный элемент среди положительных элементов введенного массива).

8. Если условие в пункте 7 будет истинным, то изменим текущий максимум, присвоив ему значение элемента массива  $A$ , удовлетворяющего условию пункта 7 (т.к. найден элемент в массиве больший текущего максимума).

9. Если логическое выражение пункта 7 в результате вычисления даст значение – ложь (т.е. элемент массива окажется не большим текущего максимума), то никаких действий производить не нужно.

10. Если логическое выражение в пункте 6 в результате вычисления даст значение – ложь (т.е. элемент массива окажется не положительным), то никаких действий производить не нужно. Разветвление закончилось. Все ветви, образовавшиеся в процессе ветвления, пришли в одну точку.

11. Цикл, описанный в пункте 5-10. заканчивается, когда параметр  $i$  станет больше чем  $N$ . Т.е. будут обработаны все  $N$  элементов массива. (на схеме выход справа из блока модификации). Теперь проверим условие  $MAX > 0$ . Это необходимо сделать т.к. не исключена возможность, что введенный массив вообще не содержит положительных элементов. И тогда предположительный максимум, заданный нами в начальный момент времени не является положительным числом, и он не изменится в ходе выполнения алгоритма.

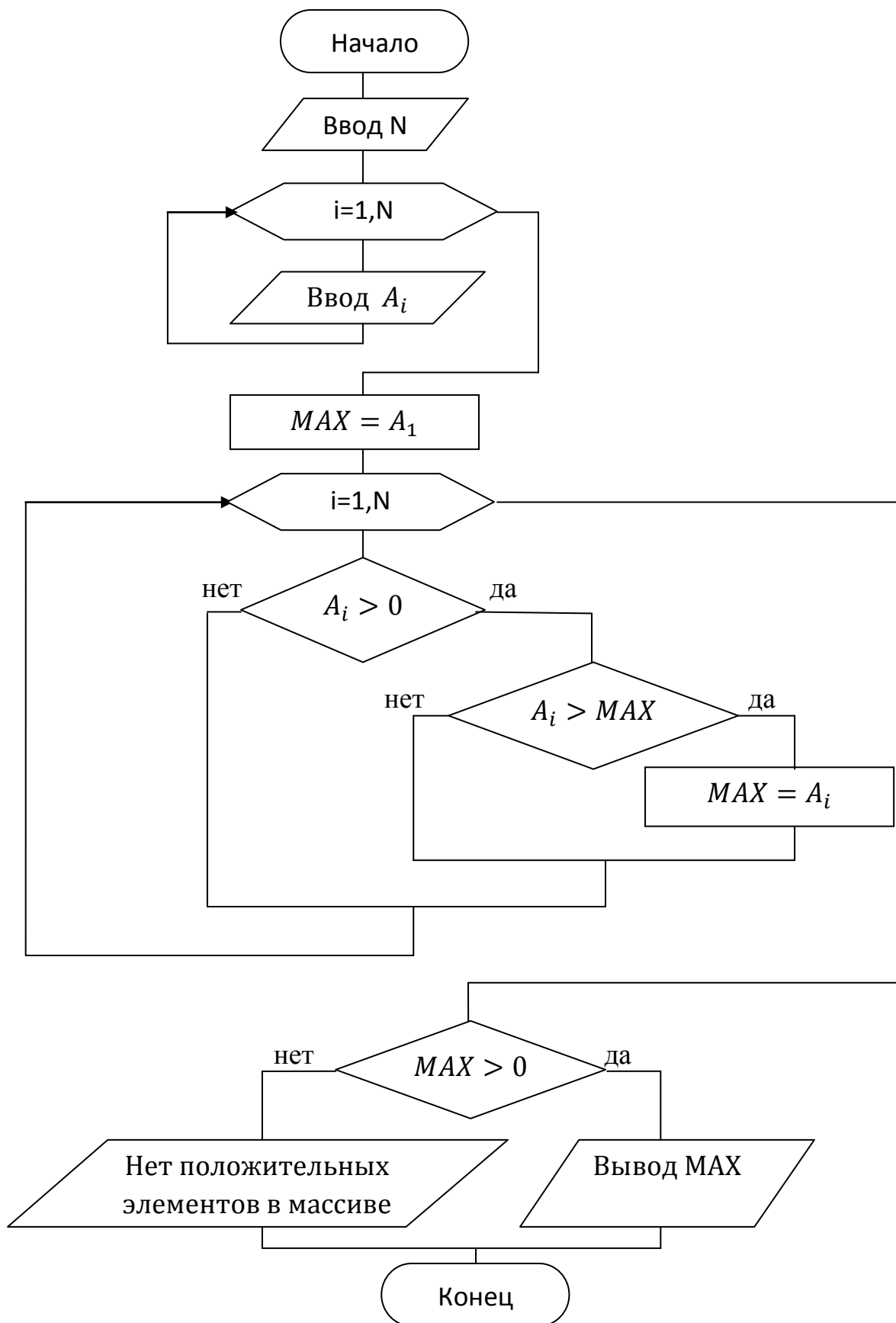


Рис. 69. Блок схема алгоритма решения задания примера 19



12. Если условие в пункте 11 – истина, то реальный максимум найден, и мы его выводим.

13. Если условие пункта 2 – ложь, то необходимо вывести сообщение о том, что в массиве не было положительных элементов, а значит, задача не имеет решения. Разветвление закончено.

14. Конец алгоритма.

### **Задания для самостоятельной работы**

1. Вычислить сумму натуральных чётных чисел не больших заданного натурального числа  $N$ .

2. Вычислить факториал натурального числа  $N$ .

3. Дано натуральное число  $N$ . Определить  $K$  – количество делителей этого числа, меньших чем само заданное число  $N$  (Например, для  $N=12$  делители 1, 2, 3, 4, 6. Количество  $K=5$ ).

4. Дано натуральное число  $N$ . Определить, является ли оно простым. Натуральное число  $N$  называется простым, если оно делится без остатка только на единицу и на само себя. Число 19 - простое, так как делится только на 1 и 19, а число 12 не является простым, так как делится на 1, 2, 3, 4, 6 и 12.

5. Из  $N$  заданных вещественных чисел определить наибольшее число.

6. Целые числа вводятся до тех пор, пока не ввели число 0. Найти наименьшее число среди положительных. Если найденный минимум – не единственный (есть повторы), определить, сколько их.

7. Водятся числа до тех пор, пока не введет число 0. Определить среднее арифметическое всех введенных чисел.

8. Водятся числа до тех пор, пока не введет число 0. Вычислить процент положительных и отрицательных чисел.
9. Вводится последовательность из  $N$  положительных целых чисел. Найти наименьшее число среди чётных элементов последовательности.
10. Определить, является ли последовательность из  $N$  произвольных чисел строго возрастающей (каждый следующий элемент больше предыдущего).
11. Водятся числа до тех пор, пока не введет число 0. Определить, является ли эта последовательность строго убывающей (каждый следующий элемент меньше предыдущего).
12. Водятся числа до тех пор, пока не введет число 0. Вычислить среднее значение чётных элементов последовательности.
13. Водится одномерный массив из  $N$  произвольных чисел, найти среднее значение отрицательных элементов этого массива.
14. Водится одномерный массив из  $N$  произвольных чисел, Определить, содержит ли массив хотя бы два соседних одинаковых числа.
15. Водятся числа до тех пор, пока не введет число 0. Найти наибольшее число среди чисел больших, чем заданное число  $K$ .
16. Водятся числа до тех пор, пока не введет число 0. Вычислить сумму и количество отрицательных чисел среди введенных.
17. Задан массив из  $N$  действительных чисел. Найти сумму положительных и сумму отрицательных элементов этого массива.
18. Задан массив из  $N$  элементов. Вычислить, сколько раз элементы в массиве меняют знак.
19. Дано  $K$  массивов из  $N$  элементов,  $N \geq 2$ . Определить количество массивов, элементы которых возрастают.
20. В массиве из  $N$  вещественных чисел определить количество простых чисел.

21. Вывести на экран таблицу значений утроенных косинусов в интервале от  $-B$  до  $B$  с шагом  $H$ .

22. Ввести целое положительное число  $K$ . Вычислить сумму натуральных нечётных чисел, не превышающих это число и произведение натуральных чётных чисел, не превышающих число  $K$ .

23. Ввести целое положительное число  $K$ . Вычислить количество натуральных чисел кратных двойке и не превышающих число  $K$ .

24. Задано целое положительное число  $n$ . Определить значение выражения:

$$p = \frac{n!}{\sum_{i=1}^n i}$$

25. Задано целое положительное число  $n$ . Определить значение выражения:

$$p = \frac{\sum_{i=1}^n i^2}{n!}$$

26. Задано целое положительное число  $n$ . Определить значение выражения:

$$p = \frac{\sum_{i=1}^n i - 2}{(n + 1)!}$$

27. Задано целое положительное число  $n$ . Определить значение выражения:

$$p = \frac{\sum_{i=1}^n 3^i}{(5)!}$$

28. Задано целое положительное число  $n$ . Определить значение выражения:

$$p = \sum_{i=0}^{10} 2^i$$

29. Задано целое положительное число  $n$ . Определить значение выражения:

$$P = \frac{\sum_{i=5}^{15} i}{(2n + 1)!}$$

30. Вводится целое число  $N$ , ( $N > 1$ ), Составить алгоритм вывода наименьшего целого числа  $K$ , при котором выполняется неравенство  $5K > N$ .

31. Вводится целое число  $N$ , ( $N > 1$ ). Составить алгоритм вывода наибольшего целого числа  $K$ , при котором выполняется неравенство  $5K < N$ .

32. Найти минимальное число большее 100, которое нацело делится на 15.

33. Найти минимальное из трех заданных чисел.

34. Найти произведения двух наименьших чисел из трех заданных.

### **БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. Кнут Д. Искусство программирования для ЭВМ. Т. 1. М.: Мир, 1976.
2. Хьюз Дж., Мичтом Дж. Структурный подход к программированию. М.: Мир, 1980.

3. Лингер Р., Миллс Х., Уитт Б. Теория и практика структурного программирования. М.: Мир, 1982.
4. Керниган, Плджер. Элементы стиля программирования. М.: Радио и связь, 1984.
5. Ван Тассел Д. Стил, разработка, эффективность, отладка и испытание программ. М.: Мир, 1985.
6. Зуев Е.А. Программирование на языке Turbo Pascal 6.0, 7.0. М.: Радио и связь, 1993.
7. Касьянов В.Н., Сабельфельд В.К. Сборник задач по практикуму на ЭВМ. М.: Наука, 1986.
8. Грогоно П. Программирование на языке Паскаль. М.: Мир, 1982.
9. Аляев Ю. А. Алгоритмизация и языки программирования Pascal, C++, Visual Basic : учеб.-справ. пособие для вузов / Ю. А. Аляев. - М. : Финансы и статистика, 2004. - 320 с.
10. Аляев Ю. А. Практикум по алгоритмизации и программированию на языке Паскаль : учеб. пособие / Ю. А. Аляев, В. П. Гладков, О. А. Козлов. – М. : Финансы и статистика, 2004. – 528 с.
11. Златопольский Д. М. Сборник задач по программированию / Д. М. Златопольский. – 2-е изд., перераб и доп. – СПб. : БХВ-Петербург, 2007. – 240 с.
12. Информатика : учебник / Б. В. Соболев [и др.]. – Изд. 3-е, дополн. и перераб. – Ростов н/Д : Феникс, 2007. – 446 с.
13. Информатика и информационные технологии / под ред. Ю. Д. Романовой. – М. : Эксмо, 2005. – 592 с.
14. Князева М. Д. Алгоритмика: от алгоритма к программе : учеб. пособие / М. Д. Князева. – М. : КУДИЦ-ОБРАЗ, 2006. – 191 с.
15. Колдаев В. Д. Основы алгоритмизации и программирования : учеб. пособие / В. Д. Колдаев ; под ред. проф. Л. Г. Гагариной. – М. : ФОРУМ: ИНФРА – М, 2006. – 416 с.

16. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. – М. : МЦНМО, 2002. – 955 с.

17. Алексеев Е.Р., Злобин Г.Г., Костюк Д.А., Чеснокова О.В., Чмыхало А.С. Программирование на языке С++ в среде Qt Creator/ М.: ALT Linux, 2015. — 448 с.

## **ОГЛАВЛЕНИЕ**

ВВЕДЕНИЕ.....	4
ТЕХНОЛОГИЯ РЕШЕНИЯ ЗАДАЧ НА КОМПЬЮТЕРЕ .....	4
Основные этапы компьютерного решения задачи .....	5

Свойства и формы записи алгоритмов .....	11
<b>АЛГОРИТМИЗАЦИЯ ЗАДАЧ НА ОСНОВЕ ТИПОВЫХ АЛГОРИТМОВ</b> .....	<b>17</b>
Методика разработки алгоритмов.....	18
Линейный алгоритм .....	19
Задания для самостоятельной работы.....	22
Разветвляющийся алгоритм .....	23
Задания для самостоятельной работы.....	44
Циклический алгоритм .....	50
Циклы с проверкой условия продолжения цикла до реализации тела цикла.....	51
Циклы с проверкой условия окончания цикла после реализации тела цикла.....	54
Цикл с параметром.....	61
Циклы с переадресацией .....	65
Задания для самостоятельной работы.....	73
<b>БИБЛИОГРАФИЧЕСКИЙ СПИСОК.....</b>	<b>76</b>

Учебное издание

**Волобуева Татьяна Витальевна**

Информатика.  
Основы алгоритмизации

Учебное пособие

Редактор

Подписано в печать 20019. Формат 60×84 1/16.

Бумага для множительных аппаратов.

Усл. печ. л. Бумага писчая. Тираж экз. Заказ №

---

ФГБОУ ВО «Воронежский государственный технический университет»

394026, Воронеж, Московский проспект 14

Участок оперативной полиграфии издательства ВГТУ

394026, Воронеж, Московский проспект 14