1.6. Определение и свойства алгоритма

Понятие алгоритма является одним из основных понятий математики и информатики. Слово “алгоритм” возникло от *algorithmi* – латинской формы написания имени великого узбекского математика Мухаммеда ибн аль-Хорезми (полное имя Абу Джафар Мухаммед ибн Муса аль-Хорезми), который в IX веке изложил правила выполнения четырех арифметических действий над числами в десятичной системе счисления. Эти правила до сих пор служат примерами простейших математических алгоритмов.

Алгоритмами являются правила дорожного движения, инструкции по пользованию различными приборами, кулинарные рецепты и т. д. Каждый из нас ежедневно пользуется различными алгоритмами, даже не зная, что это такое.

Дадим ***интуитивное*** определение алгоритма:

Алгоритм – *строгая и четкая конечная система правил, которая определяет последовательность действий над некоторыми объектами и после конечного числа шагов приводит к достижению поставленной цели.*

Это определение не является определением в математическом смысле. Точное определение алгоритма будет сформулировано в курсе математической логики, которую вы также будете изучать в университете.

Всем алгоритмам присущи некоторые общие свойства. Перечислим эмпирические (подмеченные на практике) свойства:

понятность (доступность) – все действия, описанные в алгоритме должны быть понятны исполнителю, то есть должны принадлежать системе действий данного исполнителя;

определенность (детерминированность) – каждое действие должно быть четко и однозначно определено". Точное предписание", то есть, предписание, задающее алгоритм, должно выполняться однозначно и последовательно для получения конкретного и однозначного результата;

конечность – выполнение алгоритма должно завершиться за конечное число шагов;

результативность (сходимость) – достижение после конечного числа шагов искомого результата;

дискретность (дискретная структура) – исполнение алгоритма должно состоять из отдельных шагов; Алгоритм представляет собой упорядоченное конечное множество шагов для получения результата. А всякое множество обладает свойством дискретности, то есть в любом алгоритме для каждого шага (кроме последнего), можно указать следующий за ним шаг;

массовость – состоит в том, что алгоритм служит не для решения какой-то одной задачи, а для целого класса однотипных задач. Алгоритм - это единый метод, позволяющий по любому исходному объекту из определенного бесконечного множества исходных объектов получить искомый результат.

конструктивность объектов. Исходные объекты, промежуточные и конечные результаты - это конструктивные объекты, которые могут быть построены целиком или допускают кодирование в каких-то алфавитах. Не все математические объекты конструктивны. Например, иррациональные числа, выражающиеся бесконечными десятичными непериодическими дробями, не являются конструктивными объектами, так как иррациональное число не удается ни построить целиком, ни закодировать в каком-то алфавите.

Еще одно, согласованное с предыдущим, неформальное определение алгоритма:

Под алгоритмом понимается единый метод решения определенного класса однотипных задач, обладающий свойством дискретности, массовости, определенности, результативности и оперирующий конструктивными объектами.

Для разработанного алгоритма характерна оценка его качества (эффективности), которая определяется скоростью его сходимости (количеством шагов, необходимых для получения результата), временем выполнения, удобством обращения к алгоритму, простотой и удобочитаемостью.

Формальные определения алгоритма

Формальное уточнение определения алгоритма было предложено Тьюрингом в 30-ые годы двадцатого столетия. Тьюринг описал некоторую гипотетическую машину и формализовал правила выполнения действий алгоритма этой машиной. Она получила название машины Тьюринга.

Тезис Тьюринга - "Всякий алгоритм может быть задан посредством Машины Тьюринга". Таким образом, машину Тьюринга мы можем рассматривать как точное математическое описание алгоритма.

Точное математическое описание алгоритма, получившее название Нормального Алгоритма, было сформулировано Марковым А.А., создавшим теорию нормальных алгоритмов.

Тезис Маркова А.А. - "Всякий алгоритм представим в виде нормального алгоритма".

Класс алгоритмов в форме Машины Тьюринга и класс нормальных алгоритмов Маркова совпадают, эти алгоритмы равносильны.

Другое направление при решении задачи формализации определения алгоритма - частично-рекурсивные функции, и связанный с ними тезис Черча: "Всякая вычислимая арифметическая функция является частично-рекурсивной функцией".

Все эти определения алгоритмов эквивалентны.

Алгоритм должен быть независимым как от языка программирования, так и от программного и технического обеспечения.

Прежде, чем разрабатывать алгоритм, рекомендуется осуществить анализ задачи и найти наиболее оптимальный метод ее решения.

Например, пусть требуется найти сумму и произведение чисел от 0 до 100. К решению этой задачи можно подойти по-разному. Приведем два способа решения и оценим быстродействие каждого из способов.

Первый способ:

Второй способ:

Сумма=0+1+2+3+...+99+100=5050

Для нахождения этой суммы требуется 100 операций сложения

Применим другой алгоритм: 0 - можно исключить; (1+100)+(2+99) +(3+98) + (50+51) - всего 50 скобок, значение каждой скобки=101. Тогда Сумма = 101*50 = 5050

Для выполнения этого алгоритма требуется только одна операция - умножения

Произведение=0*1*2*...*99*100=0

При последовательном перемножении мы выполним 10 операций умножения.

Произведение = 0*любое число = 0

Поэтому, мы можем сразу же получить результат - 0. Других действий выполнять не надо.

1.7. Способы описания алгоритмов

Для описания алгоритмов существуют различные способы:

обычная словесная запись, которая используется для описания бытовых алгоритмов и адресуется человеку;

математическая запись;

графическая запись в виде схем, структурограмм и графов;

запись на искусственном алгоритмическом языке (псевдокоде);

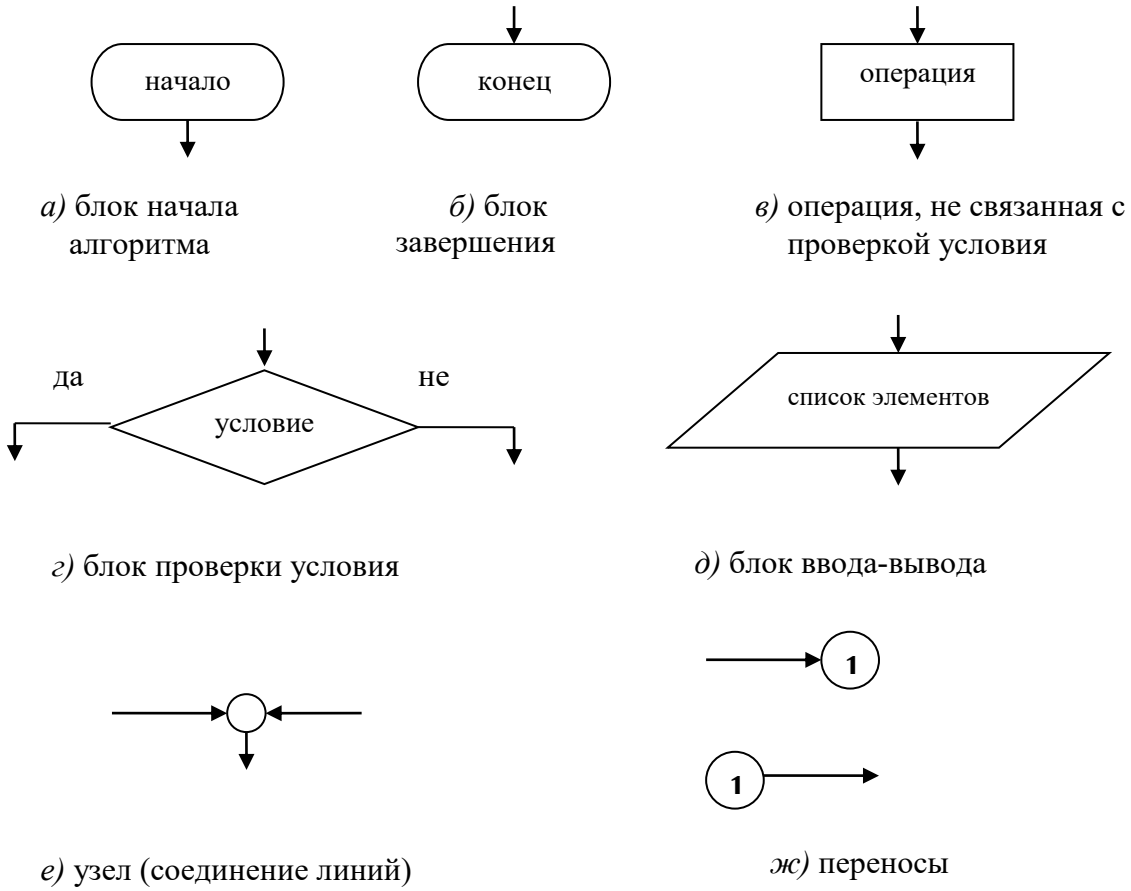
запись на одном из языков программирования.

Чаще применяются описание алгоритма на псевдокоде и в виде блок-схем.

Псевдокод – способ описания алгоритма на ограниченном подмножестве естественного языка, он занимает промежуточное положение между естественным и машинным языком.

Блок-схема – графическое представление последовательности шагов алгоритма.

На блок-схемах каждая элементарная операция обозначается специальным блоком, а последовательность, в которой выполняются эти операции – стрелками между блоками. Основные блоки, используемые для построения алгоритма, приведены на рис.1.



а) блок начала алгоритма

б) блок завершения

в) операция, не связанная с проверкой условия

г) блок проверки условия

д) блок ввода-вывода

е) узел (соединение линий)

ж) переносы

Рис. 1. Элементы блок-схем

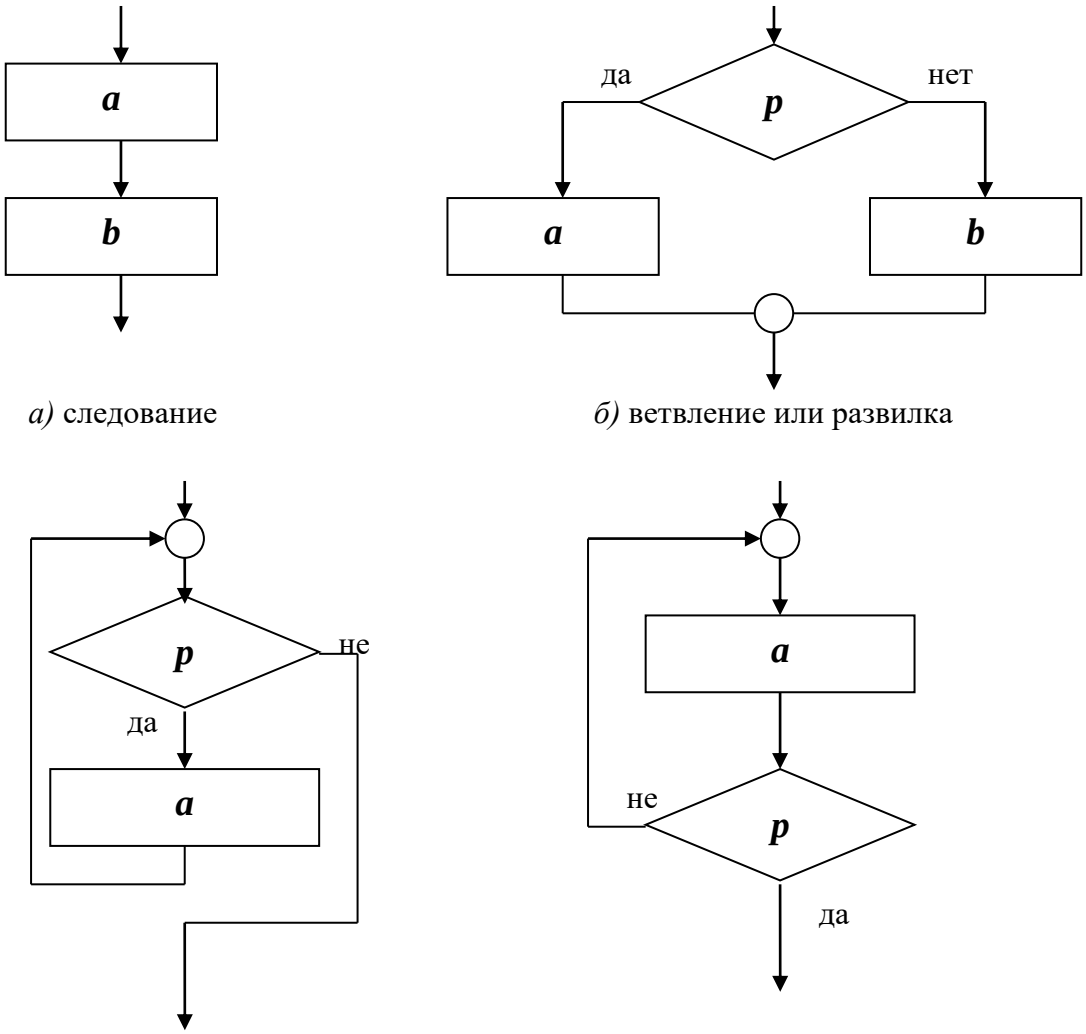
Рассмотрим далее три основные базовые структуры (управляющие конструкции алгоритмов), из которых можно построить блок-схему для любого алгоритма (в 1968 году была доказана теорема Бома-Джакопини о том, что любой алгоритм можно представить в виде комбинаций этих трех конструкций):

следование;

ветвление;

цикл.

Они показаны на рис.2.



а) следование *б)* ветвление или развилка

в) цикл с предусловием *г)* цикл с постусловием

Рис. 2. Основные базовые структуры блок-схем

Следование означает последовательное выполнение действий. Если алгоритм содержит только такую структуру, он называется *линейным.*

Ветвление или *развилка* содержит блок проверки условия. Словесно эту структуру можно описать так: если ***p*** истинно, то выполнить действие ***a***, иначе – действие ***b***. Одно из действий может отсутствовать. Алгоритм, имеющий такую структуру, называется *ветвящимся.*

Повторение или *цикл* имеет два варианта: цикл с предусловием (ЦИКЛ – ПОКА) и цикл с постусловием (ЦИКЛ – ДО). Алгоритм, содержащий структуры *повторения*, называется *циклическим*.

Действия структуры ЦИКЛ – ПОКА формулируются следующим образом: пока условие ***p*** истинно, то выполнять действие ***a***. Цикл с предусловием может быть не выполнен ни разу.

Действия структуры ЦИКЛ – ДО: выполнять действие ***a*** до тех пор, пока условие ***p*** не станет истинным. Такой цикл выполнится, по меньшей мере, один раз.

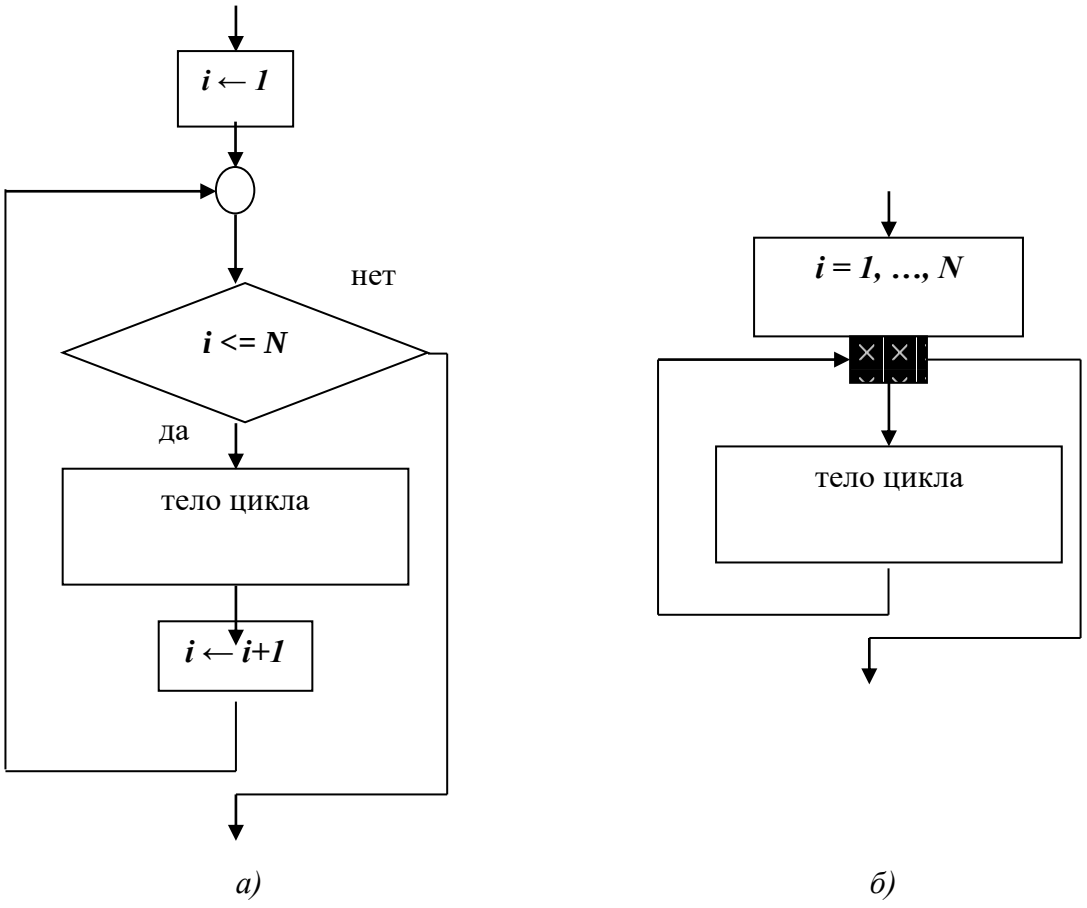


Рис. 3. Цикл, выполняемый определенное количество раз:
а) стандартная запись; *б)* упрощенная запись того же цикла

В большинстве языков программирования существует цикл, выполняемый определенное количество раз (цикл с параметром). Этот цикл является

разновидностью цикла с предусловием. Иногда в литературе его изображают так, как показано на рис. 3. (В некоторых языках параметр может изменяться не только на 1).

С течением времени и развитием языков программирования управляющие конструкции совершенствовались, предлагались новые, и к настоящему времени их состав фактически стандартизован и стал классическим. Кроме рассмотренных выше основных управляющих конструкций существуют конструкции выбора, безусловного (бесконечного) цикла, цикла с выходом из середины и некоторые другие. С ними вы познакомитесь на практических занятиях.

Каждая из базовых структур имеет один вход и один выход. Следовательно, и блок-схема, построенная из этих базовых структур, будет иметь один вход и один выход.

Рассмотрим несколько примеров построения блок-схем:

1) Алгоритм нахождения минимума из трех целых чисел.

Несмотря на кажущуюся простоту этой задачи, для ее решения можно предложить, по крайней мере, два различных алгоритма, представленные на рис. 4 и рис. 5.

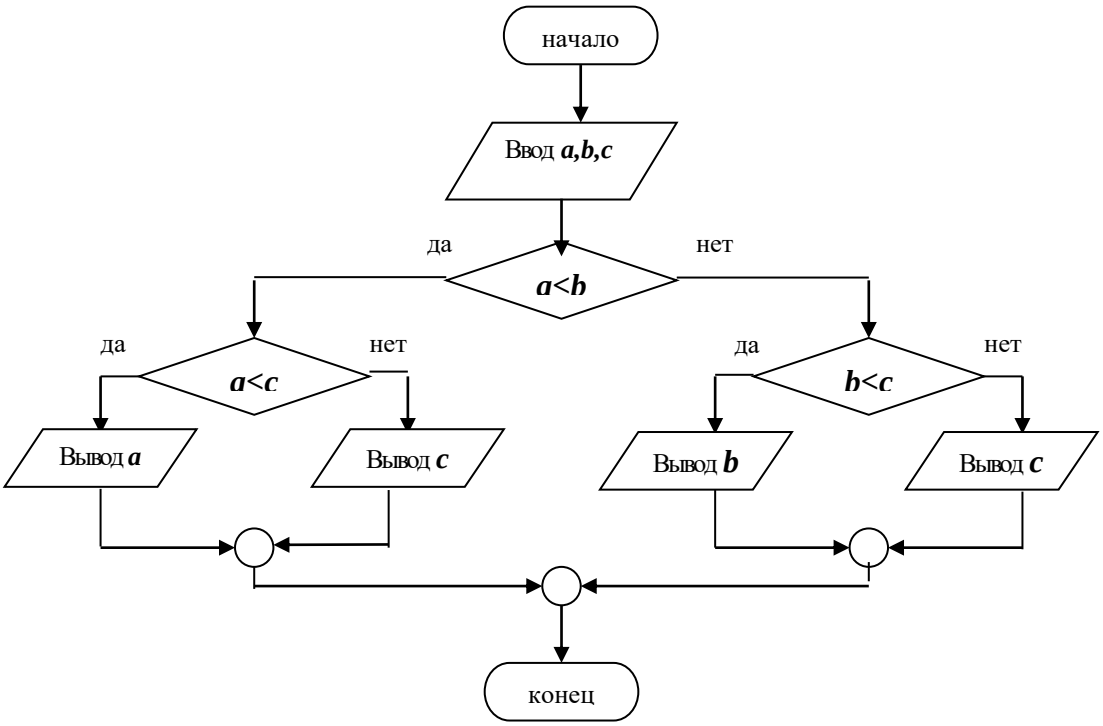


Рис. 4. Первый алгоритм нахождения минимума из трех целых чисел

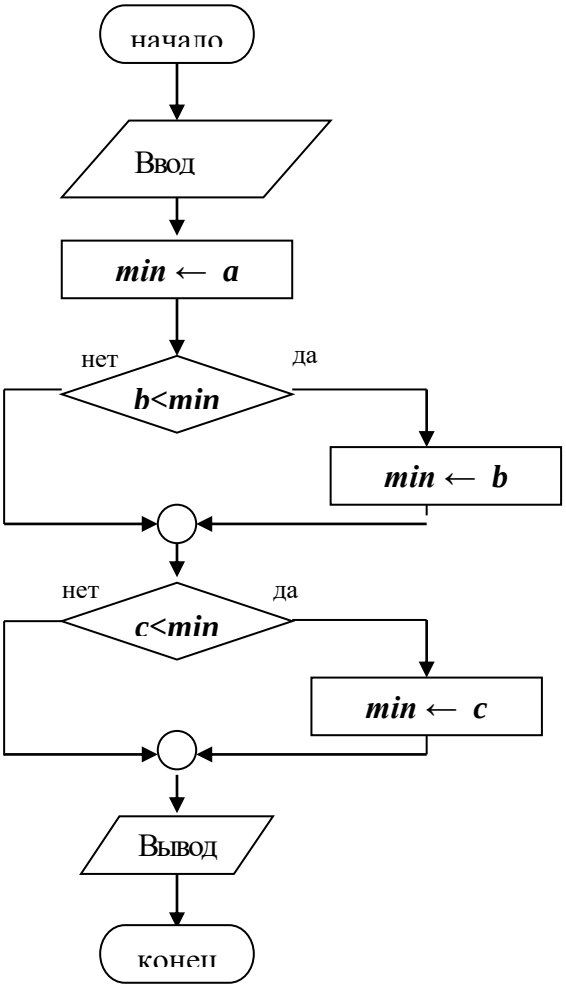


Рис. 5. Второй алгоритм нахождения минимума из трех целых чисел

Заметим, что второй алгоритм является более удобным, поскольку он допускает обобщение на произвольное количество чисел. Изменение же первого алгоритма в случае поиска минимума не из трех, а из четырех или более чисел, затруднительно

2) Найти наибольший общий делитель двух натуральных чисел.

Авторство одного из наиболее простых алгоритмов для решения этой задачи принадлежит Евклиду: из большего числа вычитается меньшее до тех пор, пока числа не станут равными; получившееся число и будет наибольшим общим делителем. Этот алгоритм в классической формулировке на псевдокоде записывается так:

Рассмотреть A как первое число, B – как второе.

Сравнить первое и второе числа. Если они равны, то перейти к п.5, если нет – к п.3.

Если первое число меньше второго, то переставить их. Перейти к п.4.

Вычесть из первого числа второе и рассмотреть полученную разность, как первое число. Перейти к п.2.

Рассмотреть первое число как результат. СТОП.

Упражнение1:

Нарисовать соответствующую блок-схему.

Измененный алгоритм Евклида немного короче:

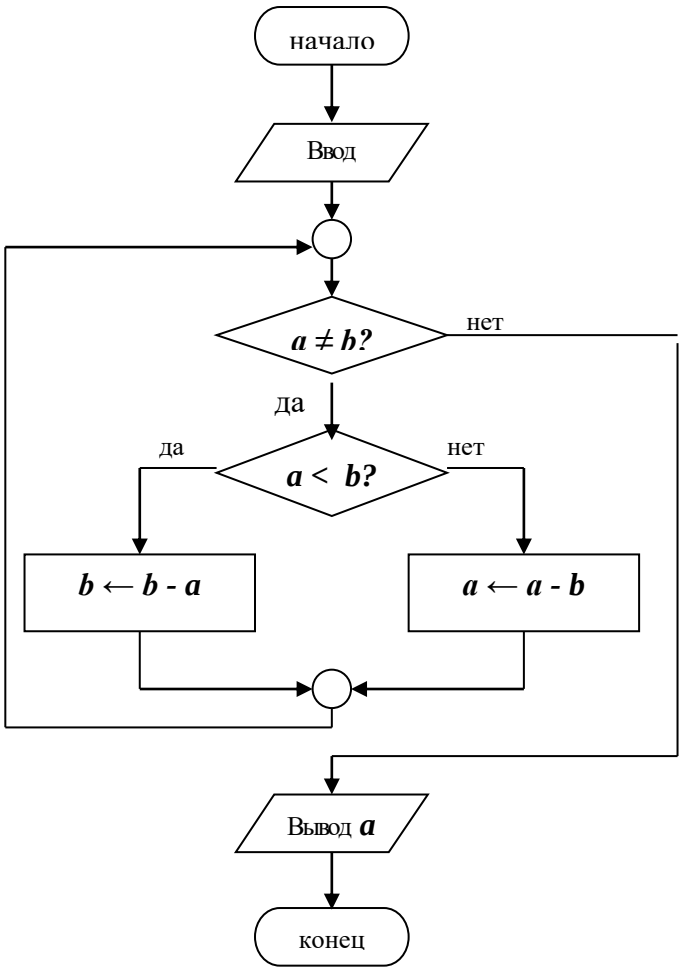


Рис. 6. Алгоритм нахождения наибольшего общего делителя двух чисел

Для того, чтобы убедиться в правильности алгоритма Евклида нам понадобятся некоторые свойства НОД.

Пусть a и b – отличные от нуля натуральные числа, причем a>b, тогда:

НОД (a, b) = НОД (a-b, b)

НОД (a,a) = a

НОД (a,0) = a

Свойства 2) и 3) очевидны.

Докажем свойство 1). Оно будет справедливо, если мы докажем, что множество общих делителей чисел a и b совпадает с множеством общих делителей чисел a-b и b, т.е. всякий делитель X чисел a и b является делителем чисел a-b и b, и наоборот.

Если X - делитель X чисел a и b, то a = k*X и b = l*X для некоторых k и l. Тогда a-b = (k-l) * X, т.е. X является общим делителем чисел a-b и b.

С другой стороны, пусть a-b = m * X и b = n * X для некоторых m и n. Тогда складывая, получаем a = (m+n) * X. Таким образом, X является общим делителем a и b.

Построенный нами алгоритм позволяет путем последовательных вычитаний (используется свойство 1) уравнять два числа, после чего НОД находится по свойству 2).

Упражнение 2. Составить блок-схему модифицированного варианта алгоритма Евклида, использующего соотношения

НОД (a, b) = НОД (a mod b, b) при a >= b,

НОД (a, b) = НОД (a, b mod a) при b >= a.

Упражнение 3. Составить блок-схему нахождения НОК двух чисел с использованием НОД. Обосновать решение.

3) Найти сумму N первых натуральных чисел.

В данном случае логично использовать цикл с параметром (рис.7).

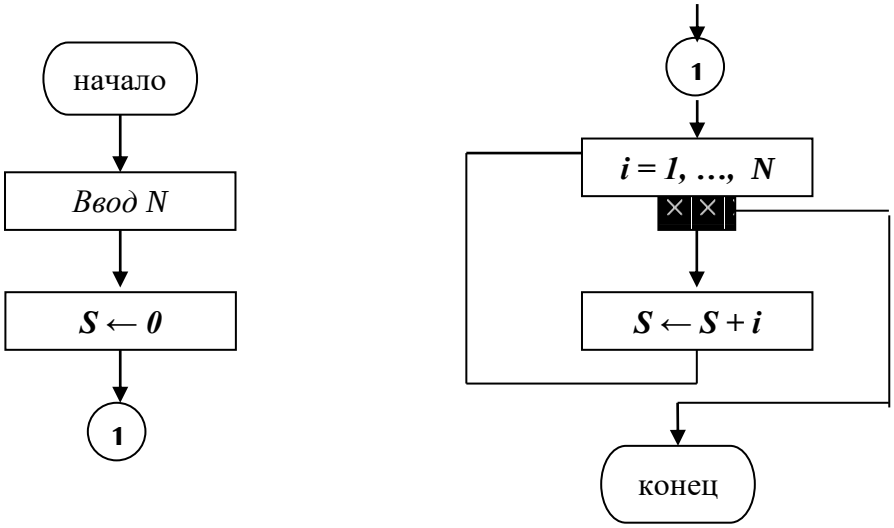


Рис. 7. Алгоритм нахождения суммы первых N натуральных чисел