

Раздел I. Основы алгоритмизации

1.1. Типы алгоритмов.

Алгоритм - конечная последовательность точно определенных действий, приводящих к однозначному решению поставленной задачи. Главная особенность любого алгоритма - *формальное исполнение*, позволяющее выполнять заданные действия (команды) не только человеку, но и различным техническим устройствам (исполнителям). Процесс составления алгоритма называется *алгоритмизацией*.

Свойства алгоритмов

Дискретность – значения новых величин (данных) вычисляются по определенным правилам из других величин с уже известными значениями.

Определенность (детерминированность) – каждое правило из набора однозначно, а сами данные однозначно связаны между собой, т.е. последовательность действий алгоритма строго и точно определена.

Результативность (конечность) – алгоритм решает поставленную задачу за конечное число шагов.

Массовость – алгоритм разрабатывается так, чтобы его можно было применить для целого класса задач, например, алгоритм вычисления определенных интегралов с заданной точностью. **Способы описания алгоритмов**

Алгоритмы могут быть заданы: словесно, таблично, графически (с помощью блок-схем). *Словесное* задание описывает алгоритм с помощью слов и предложений. *Табличное* задание служит для представления алгоритма в форме таблиц и расчетных формул. *Графическое* задание, или *блок-схема*, - способ представления алгоритма с помощью геометрических фигур, называемых *блоками*. Последовательность блоков и соединительных линий образуют блок-схему. Описание алгоритмов с помощью блок-схем является наиболее наглядным и распространенным способом задания алгоритмов. Блок-схемы располагаются сверху вниз. Линии соединения отдельных блоков показывают направление процесса обработки в схеме. Каждое такое направление называется ветвью. Алгоритм независимо от его структуры всегда имеет по одному блоку «Начало» и «Конец». Его ветви должны в конце сойтись, и по какой бы ветви не было бы начато движение, оно всегда должно привести к блоку «Конец».

При задании алгоритма с помощью блок-схемы используются строго определенные блоки. Основные типы блоков приведены в таблице 1. Следует отметить, что все блоки нумеруются. В этом случае номера проставляются вверху слева от блока (блоки «Начало», «Конец» и соединительные блоки не нумеруются). Стрелки на соединяющих линиях обычно не ставят при направлении сверху вниз и слева направо; если направление противоположное, то его показывают стрелкой на линии. отметить, что все блоки нумеруются. В этом случае номера проставляются вверху слева от блока (блоки «Начало», «Конец» и соединительные блоки не нумеруются).



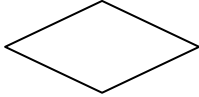



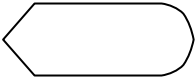
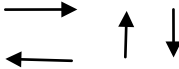
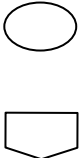
Стрелки на соединяющих линиях обычно не ставят при направлении сверху вниз и слева направо; если направление противоположное, то его показывают стрелкой на линии.

Графическое описание алгоритма

Графическое изображение алгоритма – это представление его в виде схемы, состоящей из последовательности блоков (геометрических фигур), каждый из которых отображает содержание очередного шага алгоритма. А внутри фигур кратко записывают действие, выполняемое в этом блоке. Такую схему называют блок-схемой или структурной схемой алгоритма, или просто схемой алгоритма.

Правила изображения фигур сведены в единую систему программной документации (дата введения последнего стандарта ГОСТ 19.701.90 – 01.01.1992).

По данному ГОСТу графическое изображение алгоритма – это схема данных, которая отображает путь данных при решении задачи и определяет этапы их обработки.

Наименование	Обозначение	Пояснение
1	2	3
Пуск – останов		Начало, конец алгоритма, останов, вход, выход в подпрограмму
Процесс		Вычислительная операция или группа операций
Решение		Разветвление в алгоритме, проверка условий
Предопределенный процесс		Программа, стандартная подпрограмма
Ввод-вывод		Ввод-вывод в общем виде
Документ		Вывод результатов на бумагу
Дисплей		Ввод-вывод данных на дисплей
Линии потока		Соединительные линии между блоками алгоритмов
Соединители		Разрыв линий потока на странице, на разных страницах

Типы алгоритмов

Алгоритмы бывают *линейные, разветвляющиеся и циклические*.

Линейный алгоритм не содержит логических условий, имеет одну ветвь обработки и изображается линейной последовательностью связанных друг с другом блоков. Условное изображение линейного алгоритма может быть представлено на рис. 1.1

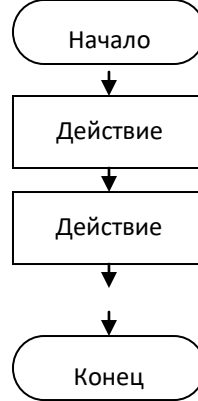


Рис. 1.1. Условное изображение линейного алгоритма.

Пример простейшего линейного процесса

Наиболее часто в практике программирования требуется организовать расчет некоторого арифметического выражения при различных исходных данных. Например, такого:

$$z = \frac{tg^2x}{\sqrt{x^2+m^2}} + x^{(m+1)}\sqrt{x^2 + m^2},$$

где $x > 0$ – вещественное, m – целое.

Разработка алгоритма обычно начинается с составления схемы. Продумывается оптимальная последовательность вычислений, при которой, например, отсутствуют повторения. При написании алгоритма рекомендуется переменным присваивать те же имена, которые фигурируют в заданном арифметическом выражении либо иллюстрируют их смысл.

Для того чтобы не было «длинных» операторов, исходное выражение полезно разбить на ряд более простых. В нашей задаче предлагается схема вычислений, представленная на рис. 1.2.

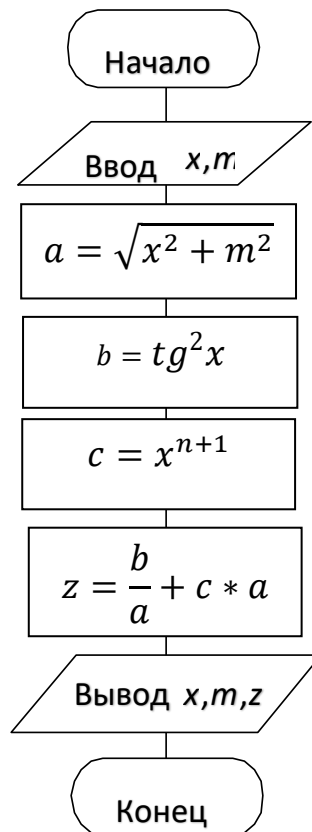


Рис. 1.2. Схема линейного процесса

Она содержит ввод и вывод исходных данных, линейный вычислительный процесс, вывод полученного результата. Заметим, что выражение $\sqrt{x^2 + m^2}$ вычисляется только один раз. Введя дополнительные переменные a, b, c , мы разбили сложное выражение на ряд более простых.

Разветвляющийся алгоритм содержит одно или несколько логических условий и имеет несколько ветвей обработки.

Условное изображение разветвления представлено на рис. 1.3. Структура РАЗВЕТВЛЕНИЕ предусматривает проверку условия, после которого вычислительный процесс развивается по одной из двух ветвей (в зависимости от ответа на поставленный в условии вопрос). Каждый из путей (ветвей) ведет к общему выходу.

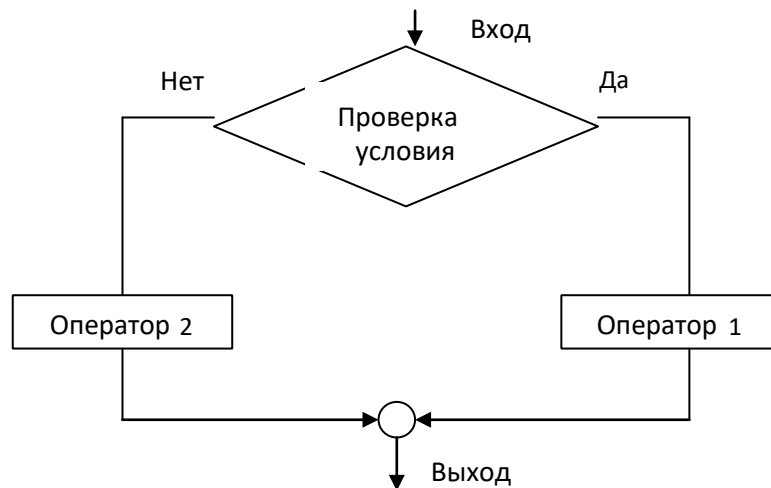


Рис. 1.3. Условное изображение разветвляющегося алгоритма

Пример разветвляющегося процесса

Построить алгоритм вычисления значения функции y , заданной формулой

$$y = \begin{cases} x^2 + 2, & \text{если } x \leq 1 \\ x^2 - 2x + 4, & \text{если } x > 1 \end{cases}$$

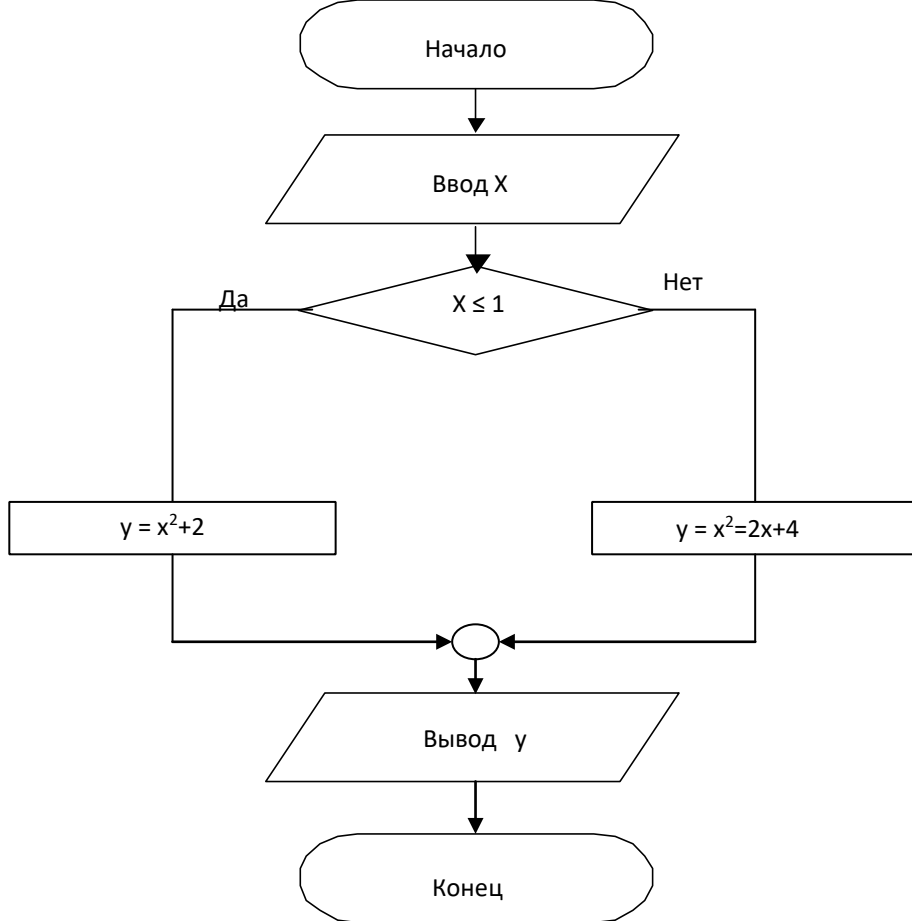


Рис. 1.4. Блок-схема примера

Если $x > 1$, то y вычисляется по формуле $y = x^2 - 2x + 4$. Блок-схема алгоритма решения задачи приведена на рис. 1.4.

Циклический алгоритм содержит один или несколько циклов – многократно повторяемых частей алгоритма. Цикл, не содержащий внутри себя других циклов, называют простым. Если он содержит внутри себя другие циклы или разветвления, то цикл называют сложным или вложенным. Любой цикл характеризуется одной или несколькими переменными, называемыми параметрами цикла, от анализа значений которых зависит выполнение цикла. *Параметр цикла* – переменная, принимающая при каждом вхождении в цикл новое значение. Условное изображение циклического алгоритма представлено на рис. 1.5.

Базовую структуру Цикл с предусловием можно использовать для описания циклического процесса при решении любой задачи, т.к. она предусматривает возможность обхода этого цикла в случае невыполнения условия при первой проверке условия. Базовую структуру Цикл с постусловием можно использовать только в тех случаях, когда из условия задачи следует, что, хотя бы один раз цикл обязательно должен выполняться.

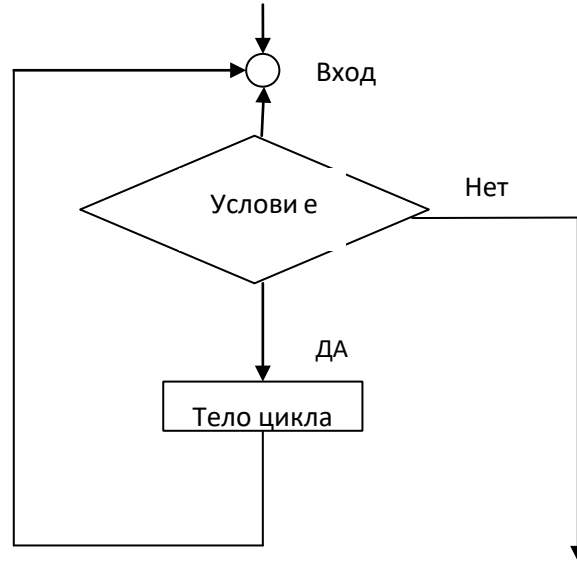


Рис. 1.5 Условное изображение циклического алгоритма

Пример циклического процесса

Вычислить значение функции $y = \sin x$, представленной в виде разложения в ряд, с заданной точностью, т.е. до тех пор, пока разность между соседними слагаемыми не станет меньше заданной точности:

$$y = \sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

Схема алгоритма, приведенная на рис. 1.7, реализует циклический процесс, в состав которого (в блоке проверки $|E| < eps$) входит участок разветвления.

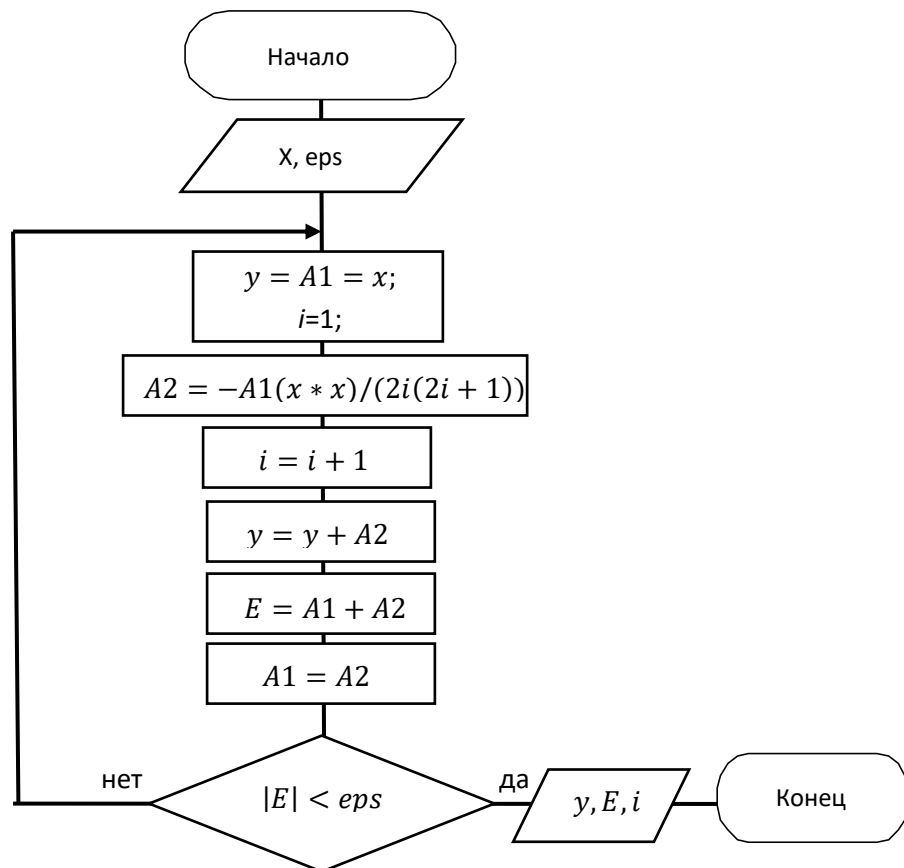


Рис. 1.6. Схема циклического алгоритма