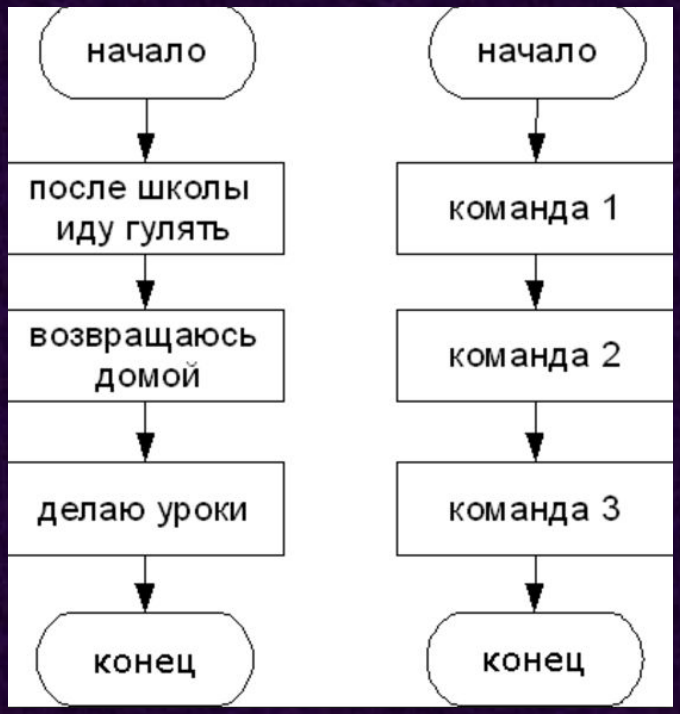
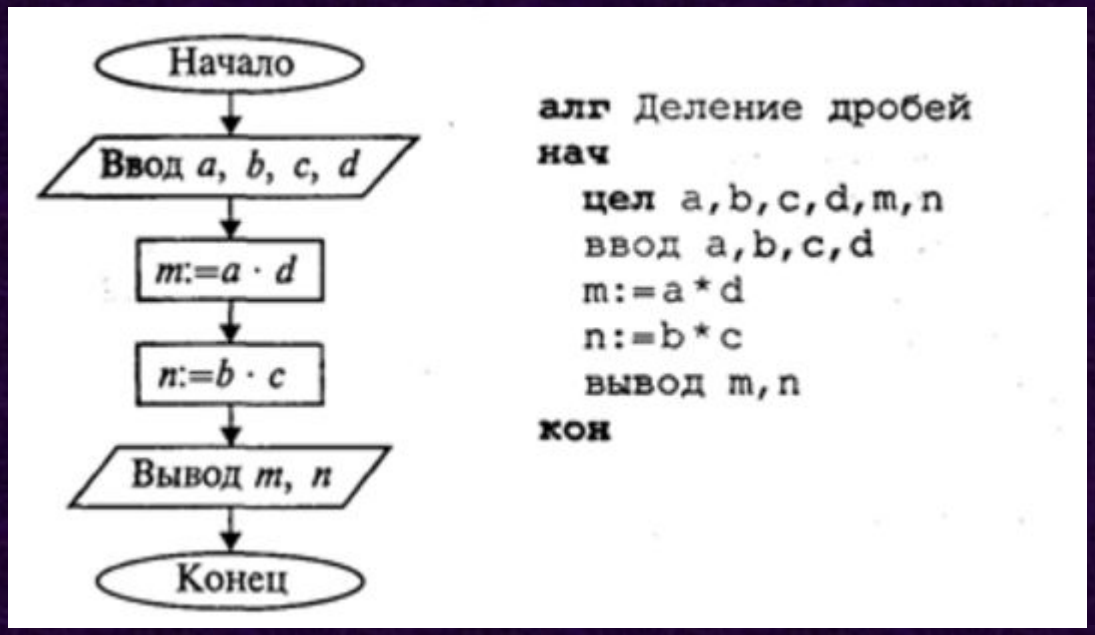
Алгоритмы, структуры алгоритмов, структурное программирование

Понятие об алгоритме:

Алгоритм - это предписание исполнителю (человеку или автомату) выполнить точно определенную последовательность действий, направленных на достижение заданной цели.





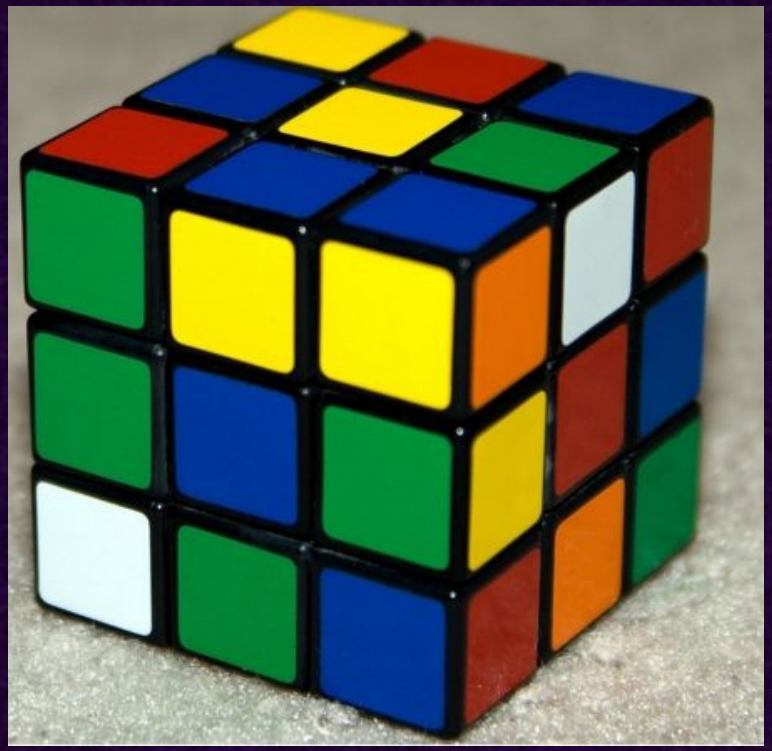
Алгоритм - это сформулированное на некотором языке правило, указывающее на действия, последовательное выполнение которых приводит от исходных данных к искомому результату. Значение слова алгоритм очень схоже со значением слов рецепт, процесс, метод, способ. Однако любой алгоритм, в отличие от рецепта или способа, обязательно обладает следующими свойствами.

Свойства алгоритма

Основные свойства алгоритма

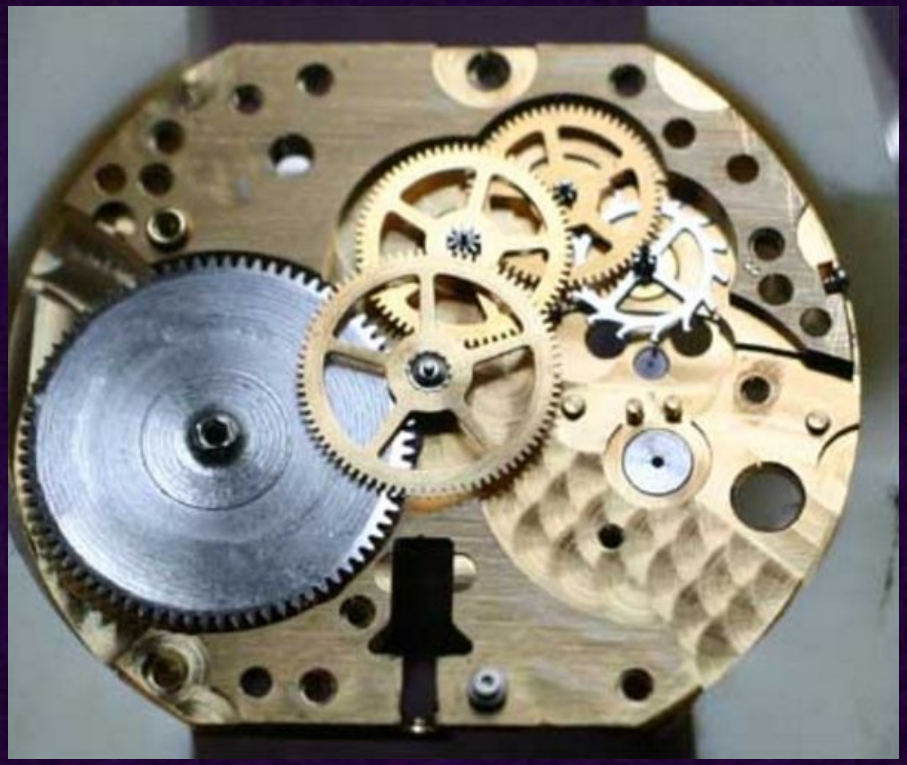
1. Дискретность

- разбиение алгоритма на ряд отдельных законченных действий - шагов. Выполнение алгоритма разбивается на последовательность законченных действий - шагов. Каждое действие должно быть закончено исполнителем алгоритма прежде, чем он приступит к исполнению следующего действия.



2. Точность

- однозначные указания. На каждом шаге однозначно определено преобразование объектов среды исполнителя, полученной на предыдущих шагах алгоритма. Если алгоритм многократно применяется к одному и тому же набору исходных данных, то на выходе он получает каждый раз один и тот же результат. Запись алгоритма должна быть такой, чтобы на каждом шаге его выполнения было известно, какую команду надо выполнять следующей.



3. Понятность

- однозначное понимание и исполнение каждого шага алгоритма его исполнителем. Алгоритм должен быть записан на понятном для исполнителя языке.

4. Результативность

- обязательное получение результата за конечное число шагов. Каждый шаг (и алгоритм в целом) после своего завершения дает среду, в которой все объекты однозначно определены. Если это по каким-либо причинам невозможно, то алгоритм должен сообщать, что решение задачи не существует. Работа алгоритма должна быть завершена за конечное число шагов. Информатика оперирует только с конечными объектами и конечными процессами, поэтому вопрос о рассмотрении бесконечных алгоритмов остается за рамками теории алгоритмов.

5. Массовость

- применение алгоритма к решению целого класса однотипных задач. Исходные данные могут отличаться.

Исполнитель алгоритма

Исполнитель - это некоторый объект (человек, животное, техническое устройство), способный выполнять определённый набор команд.

Данные и величины

Величины – различные информационные объекты (числа, символы, коды и пр.), с которыми работает универсальный исполнитель алгоритмов – компьютер.

- константы
- переменные

Данные – совокупность величин.

- исходные
- промежуточные
- результаты

Этапы решения задачи на компьютере

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

1.Постановка задачи.

2.Формализация задачи.

3.Построение алгоритма.

4.Составление программы на языке программирования.

5.Отладка и тестирование программы.

6.Проведение расчетов и анализ полученных результатов.

Часто эту последовательность называют технологической цепочкой решения задачи на компьютере.

Разработка алгоритма

Алгоритм – модель деятельности исполнителя алгоритмов

Независимо от того , на каком языке программирования будет написана программа , алгоритм решения любой задачи на компьютере может быть составлен из команд:

- Присваивания;
- Ввода;
- Вывода;
- Обращения в вспомогательному алгоритму (подпрограмме);
- Цикла;
- Ветвления.

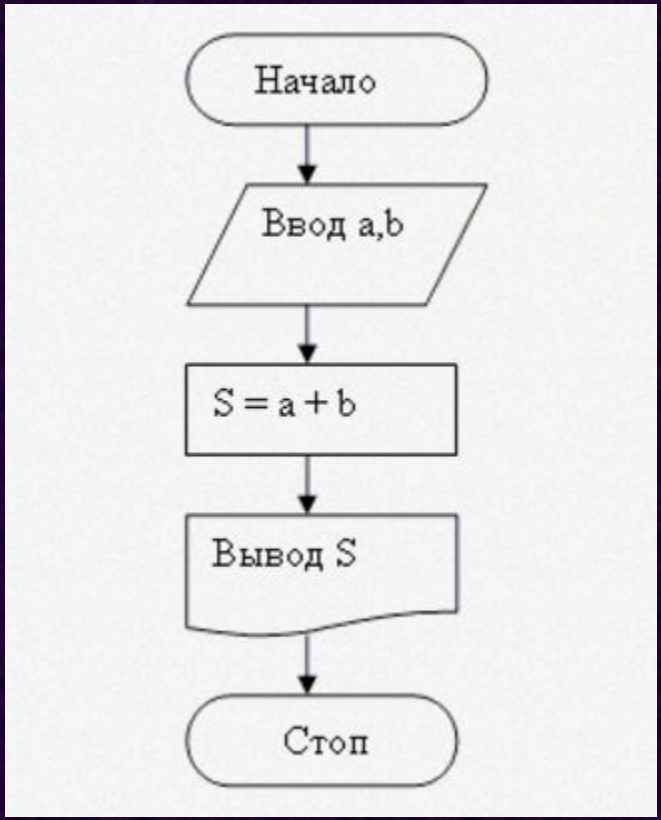
Алгоритмы и величины.

Способы описания алгоритмов.

Выше отмечалось, что один и тот же алгоритм может быть записан по-разному. Можно записывать алгоритм **естественным языком.** В таком виде мы используем рецепты, инструкции и т.п. Для записи алгоритмов, предназначенных формальным исполнителям, разработаны специальные **языки программирования**. Любой алгоритм можно описать **графически в виде блок-схемы**. Для этого разработана специальная система обозначений:

Обозначение	Описание	Примечания
	Начало и конец алгоритма	
	Ввод и вывод данных.	Вывод данных иногда обозначают иначе: 
	Действие	В вычислительных алгоритмах так обозначают присваивание
	Развилка	Развилка - компонент, необходимый для реализации ветвлений и циклов
	Начало цикла с параметром	
	Типовой процесс	В программировании - процедуры или подпрограммы
	Переходы между блоками	

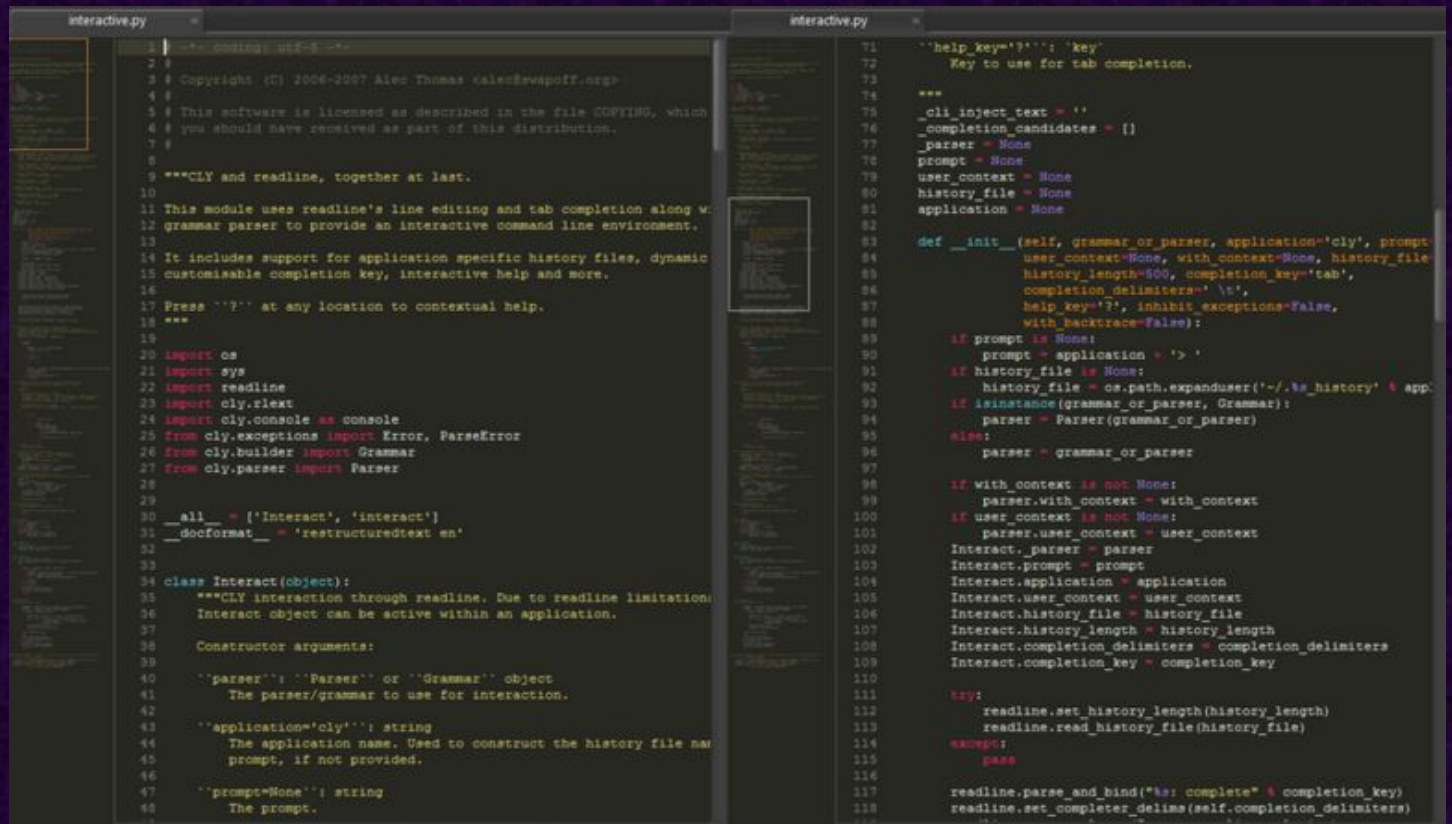
Пример описания алгоритма суммирования двух величин в виде блок-схемы:



Такой способ описания алгоритм наиболее нагляден и понятен человеку. Поэтому, алгоритмы формальных исполнителей обычно разрабатывают сначала в виде блок-схемы, и только затем создают программу на одном из языков программирования.

Типовые алгоритмические структуры.

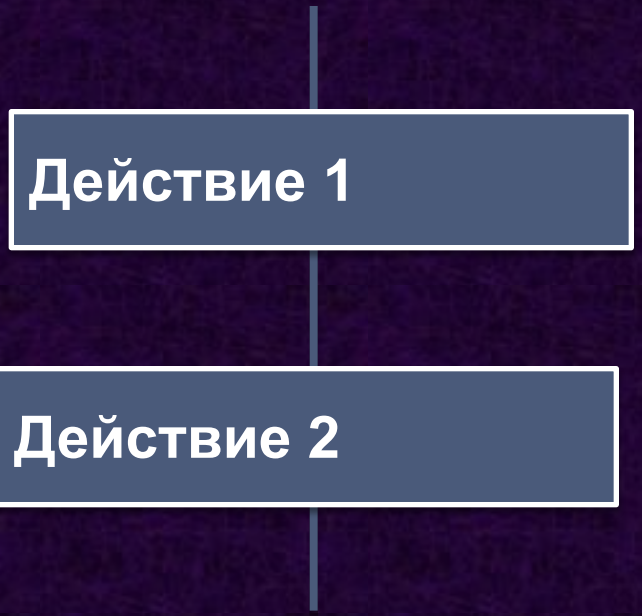
Программист имеет возможность конструировать и использовать нетипичные алгоритмические структуры, однако, в этом нет необходимости. **Любой сколь угодно сложный алгоритм может быть разработан на основе трёх типовых структур: следования, ветвления и повторения. При этом структуры могут располагаться последовательно друг за другом или вкладываться друг в друга.**



Следование

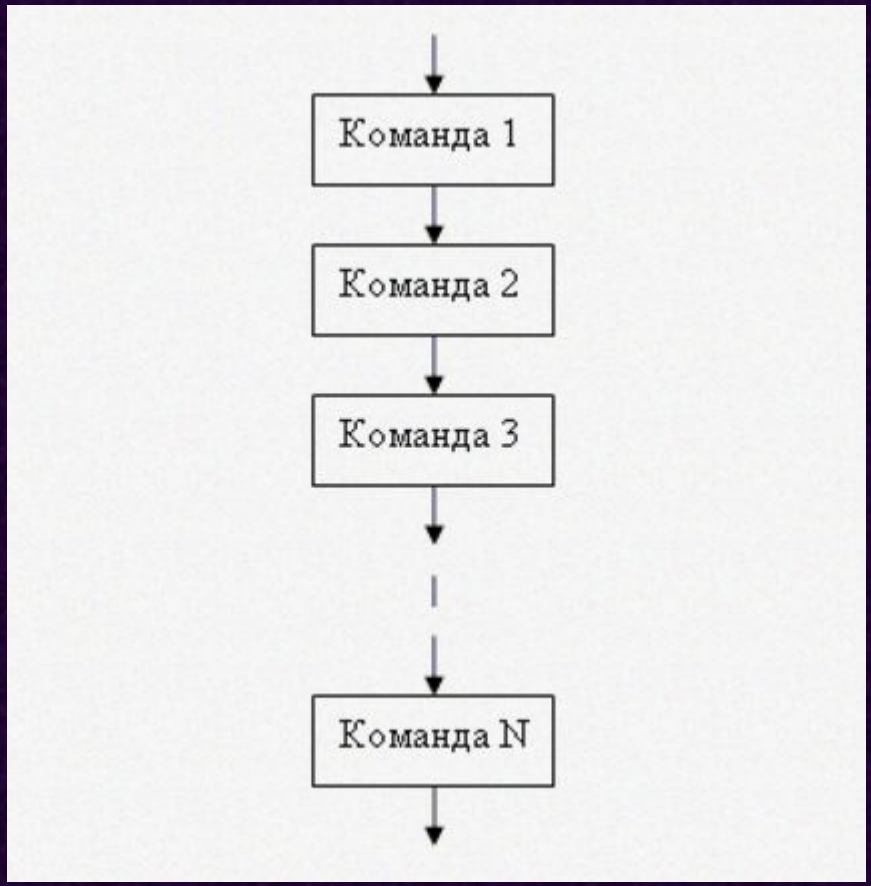
Следование - алгоритмическая конструкция, отображающая естественный, последовательный порядок действий.

Алгоритмы, в которых используется только структура «следование», называются ***линейными***.



Линейная структура (следование)

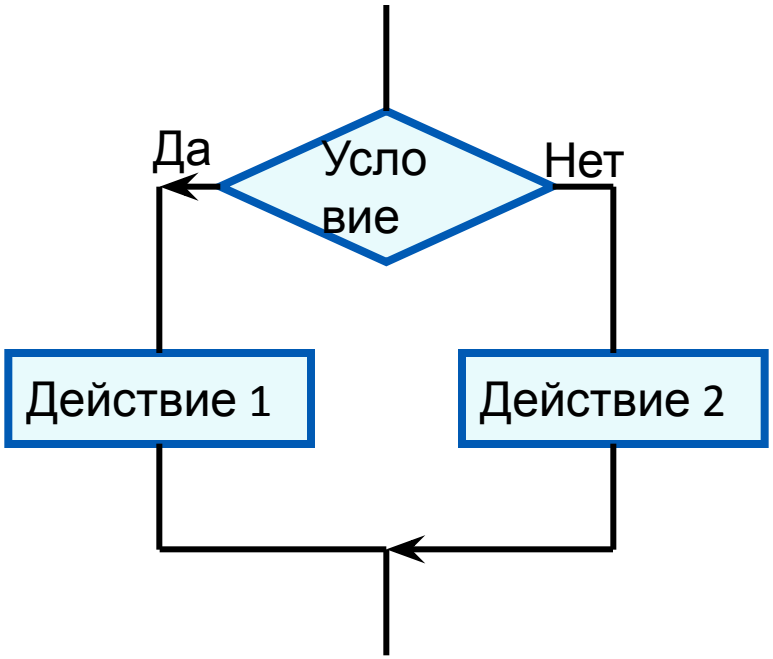
Наиболее простой алгоритмической структурой является **линейная**. *В ней все операции выполняются один раз в том порядке, в котором они записаны.*



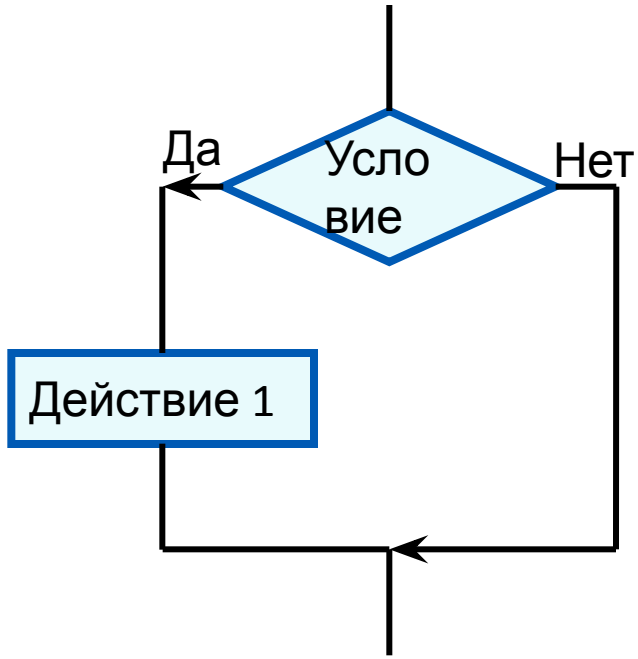
Ветвление

Ветвление - алгоритмическая конструкция, в которой в зависимости от результата проверки условия (да или нет) предусмотрен выбор одной из двух последовательностей действий (ветвей).

Алгоритмы, в основе которых лежит структура «ветвление», называют ***разветвляющимися***.



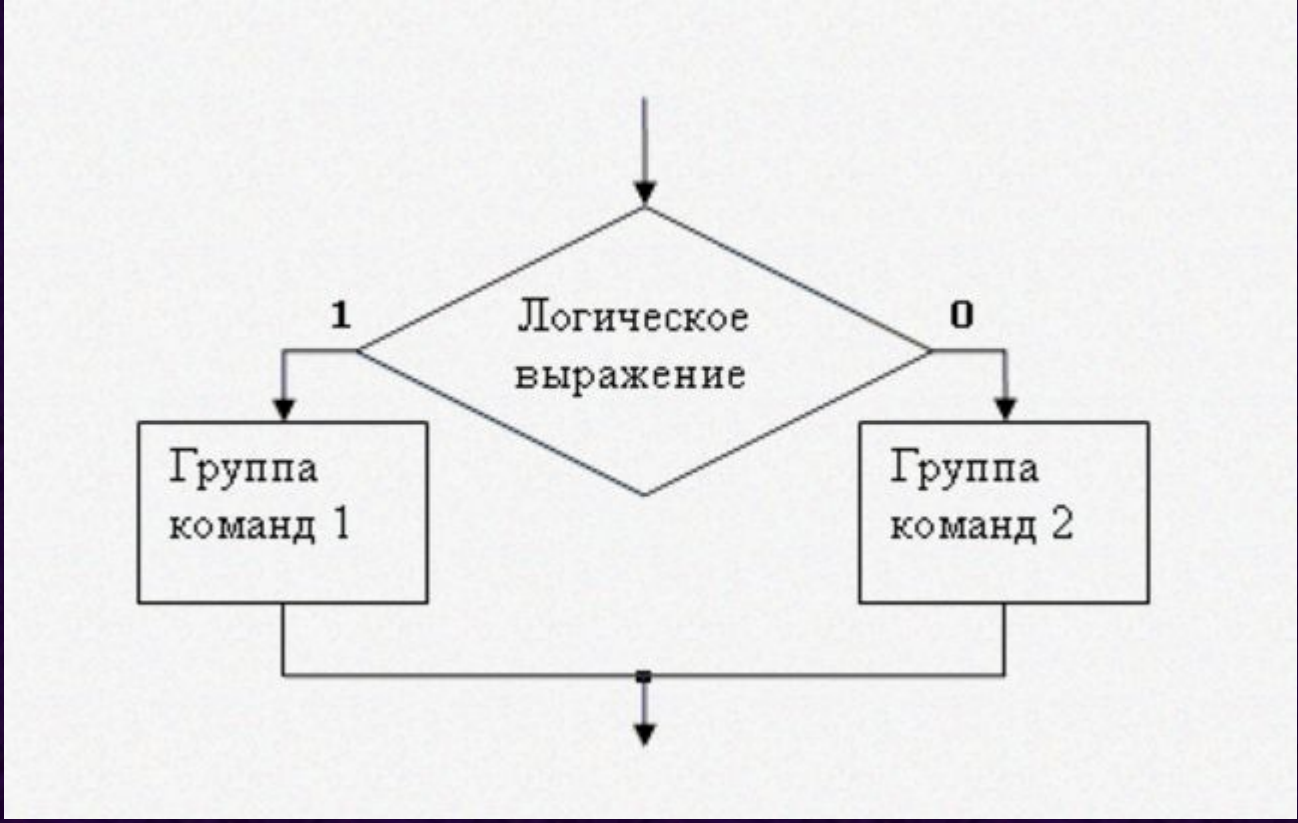
Полная форма ветвления



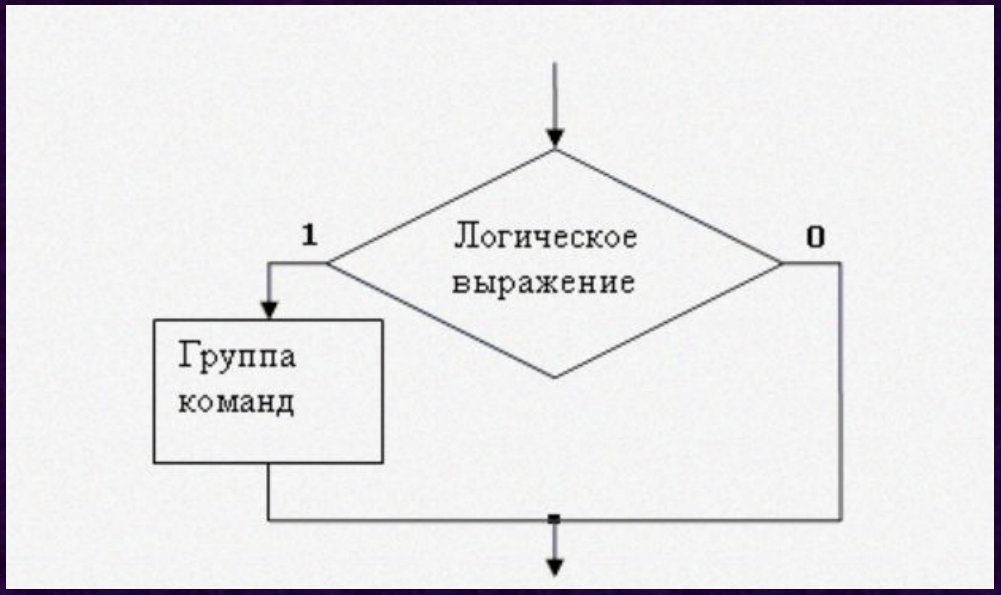
Неполная форма ветвления

Ветвление

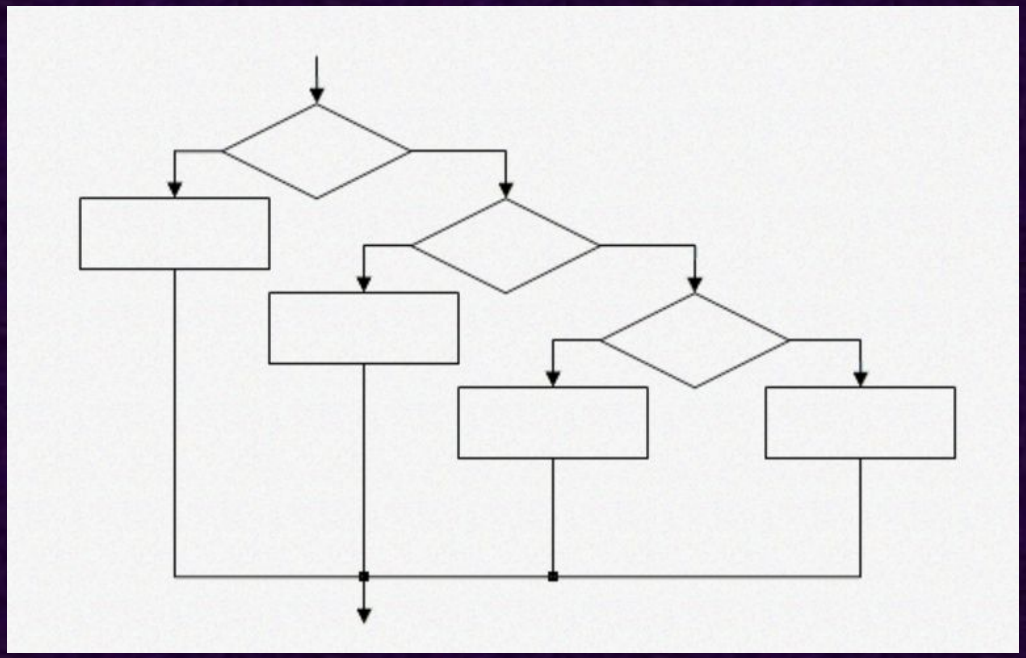
В **полном ветвлении** предусмотрено два варианта действий исполнителя в зависимости от значения логического выражения (условия). *Если условие истинно, то выполняться будет только первая ветвь, иначе только вторая ветвь.*



Вторая ветвь может быть пустой. Такая структура называется **неполным ветвлением или обходом**.



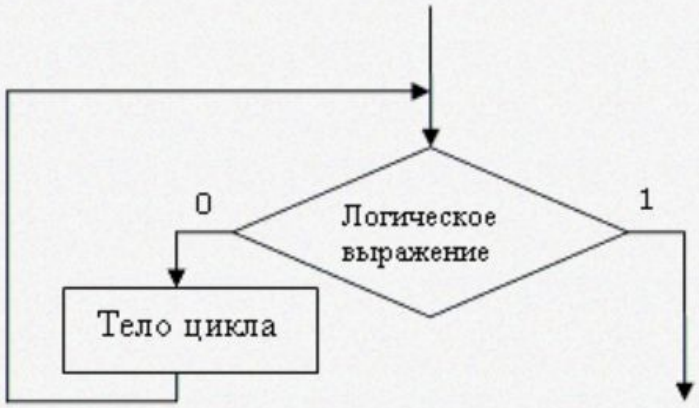
Из нескольких ветвлений можно сконструировать структуру «**выбор**» (множественное ветвление), которая будет выбирать не из двух, а из большего количества вариантов действий исполнителя, зависящих от нескольких условий. Существенно, что выполняется только одна ветвь - в такой структуре важное значение приобретает порядок следования условий: если выполняются несколько условий, то сработает только одно из них - первое сверху.

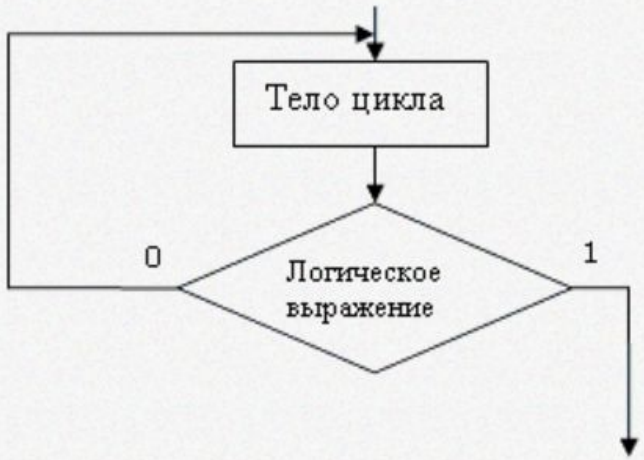


Цикл (повторение)

Цикл *позволяет организовать многократное повторение одной и той же последовательности команд* - она называется телом цикла. В различных видах циклических алгоритмов количество повторений может зависеть от значения логического выражения (условия) или может быть жестко задано в самой структуре. Различают циклы : **«до», «пока», циклы со счётчиком.** В циклах «до» и «пока» логическое выражение (условие) может предшествовать телу цикла (**цикл с предусловием**) или завершать цикл (**цикл с послеусловием**).

Циклы «до» - повторение тела цикла **до выполнения условия:**





Цикл (повторение)

Повторение - алгоритмическая конструкция, представляющая собой последовательность действий, выполняемых многократно.

Алгоритмы, содержащие конструкцию «повторение», называют ***циклическими*** или ***циклами***.

Последовательность действий, многократно повторяющаяся в процессе выполнения цикла, называется ***телом цикла***.