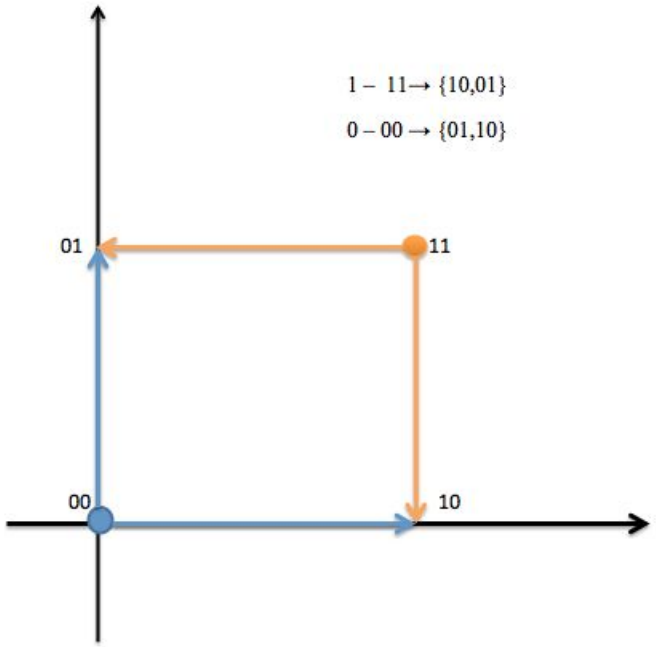
Коды Хемминга

Коды Хемминга обнаруживают и исправляют ошибки.

Передается двоичная последовательность. В процессе передачи возможны ошибки: 1→ 0; 0→1.

Предполагается, что возникает только **одна** ошибка.

Например 100111000011000→1**1**0111000011000. Как обнаружить ошибку? – добавить проверочные символы:



Мы видим, что невозможно исправить ошибку, поскольку множества ошибок совпадают.

Добавим ещё один проверочный символ:

Кодирование

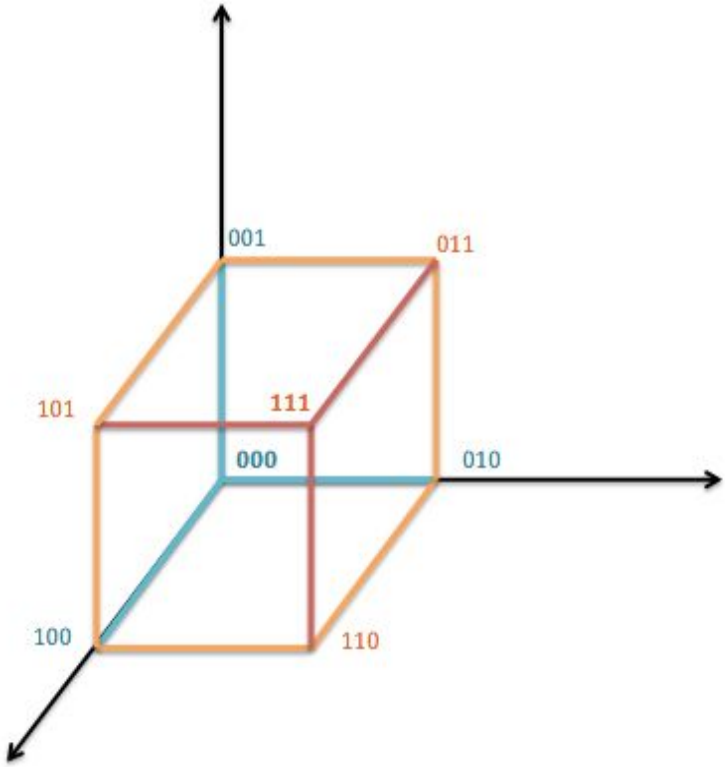
1→ 111→ {011,101,110}

0→ 000→ {100,010,001}

Декодирование

{011,101,110} → 111 → 1

{100,010,001}→ 000 → 0



Таким образом по полученному ошибочному сообщению возможно обнаружить и исправить одну ошибку.

Алгоритм

- **Слово «алгоритм» появилось в результате искаженного перевода с арабского на европейские языки имени узбекского ученого IX века Аль-Хорезми, который изложил правила арифметических действий над числами в позиционной десятичной системе. Эти правила и назвали алгоритмами (Альхорезми «имя»+ Аритмос «число»= алгоритм)**

«**Алгори́тм** — набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное число действий, при любом наборе исходных данных.» *Википедия*

Требования к алгоритму:

- **Конечность(результативность)** алгоритма означает, что за конечное число шагов должен быть получен результат;
- **Дискретность** алгоритма означает, что алгоритм должен быть разбит на последовательность выполняемых шагов;
- **Понятность** алгоритма означает, что алгоритм должен содержать только те команды, которые входят в набор команд, который может выполнить конкретный исполнитель;
- **Точность** алгоритма означает, что каждая команда должна пониматься однозначно;
- **Массовость** алгоритма означает, что однажды составленный алгоритм должен для решения подобных задач с разными исходными данными;
- **Детерминированность (определенность)**. Алгоритм обладает свойством детерминированности, если для одних и тех же наборов исходных данных он будет выдавать один и тот же результат, т.е. результат однозначно определяется исходными данными.

Примеры алгоритмов

Пример 1 – нарезание апельсина на дольки:

Начало

1. достать нож;
2. нарезать апельсин на дольки (Именно апельсин, а не любой другой фрукт. За это отвечает ТОЧНОСТЬ);
3. достать тарелку;
4. выложить на тарелке;
5. подать к столу.

Конец

Пример 2 – Зная длины трех сторон треугольника, вычислить площадь и периметр треугольника.

Пусть **a, b, c** - длины сторон треугольника. Необходимо найти **S** - площадь треугольника, **P** - периметр.

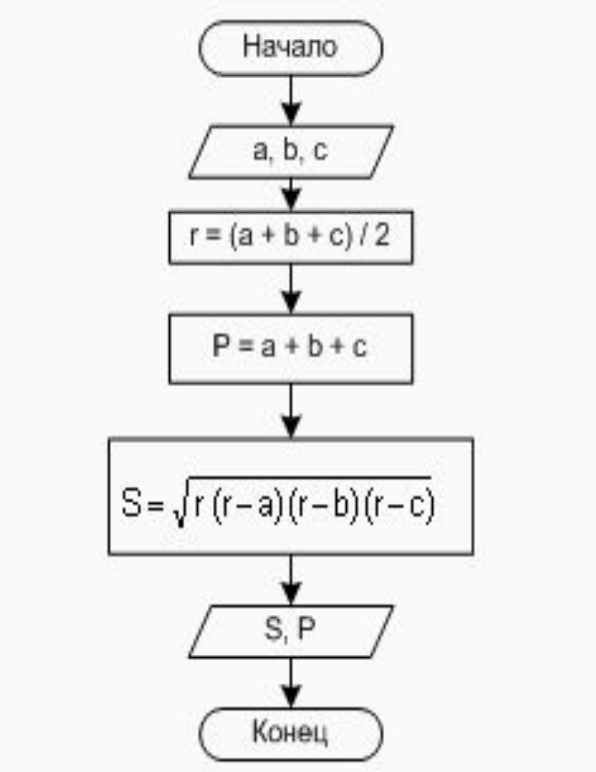
Для нахождения площади можно воспользоваться формулой Герона:

$$\sqrt{r(r-a)(r-b)(r-c)}\text{, где } r \text{ - полупериметр.}$$

Входные данные: **a, b, c**.

Выходные данные: **S, P**.

Удобно представить графически:

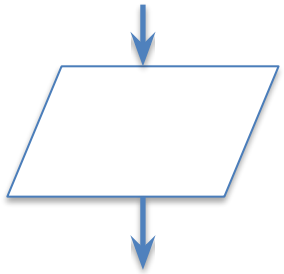


Найдите неточность в схеме!

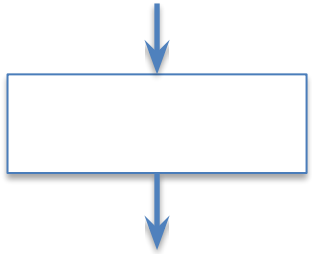
Блок-схемы

- **Блок-схемой** называется наглядное графическое изображение алгоритма, когда отдельные его этапы изображаются при помощи различных геометрических фигур - блоков, а связи между этапами (последовательность выполнения этапов) указываются при помощи стрелок, соединяющих эти фигуры. Блоки сопровождаются надписями. Типичные действия алгоритма изображаются следующими геометрическими фигурами:
- **Блок начала-конца алгоритма**. Надпись на блоке: "начало" ("конец").

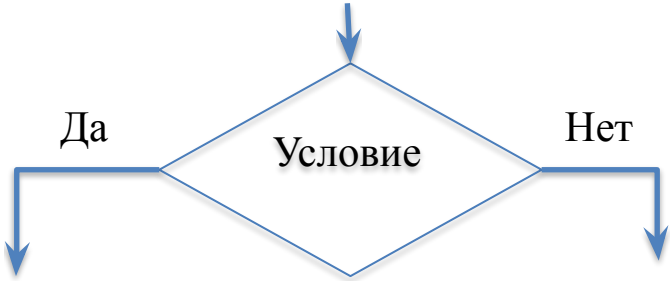
- **Блок ввода-вывода данных** . Надпись на блоке: слово "ввод" ("вывод" или "печать") и список вводимых (выводимых) переменных.



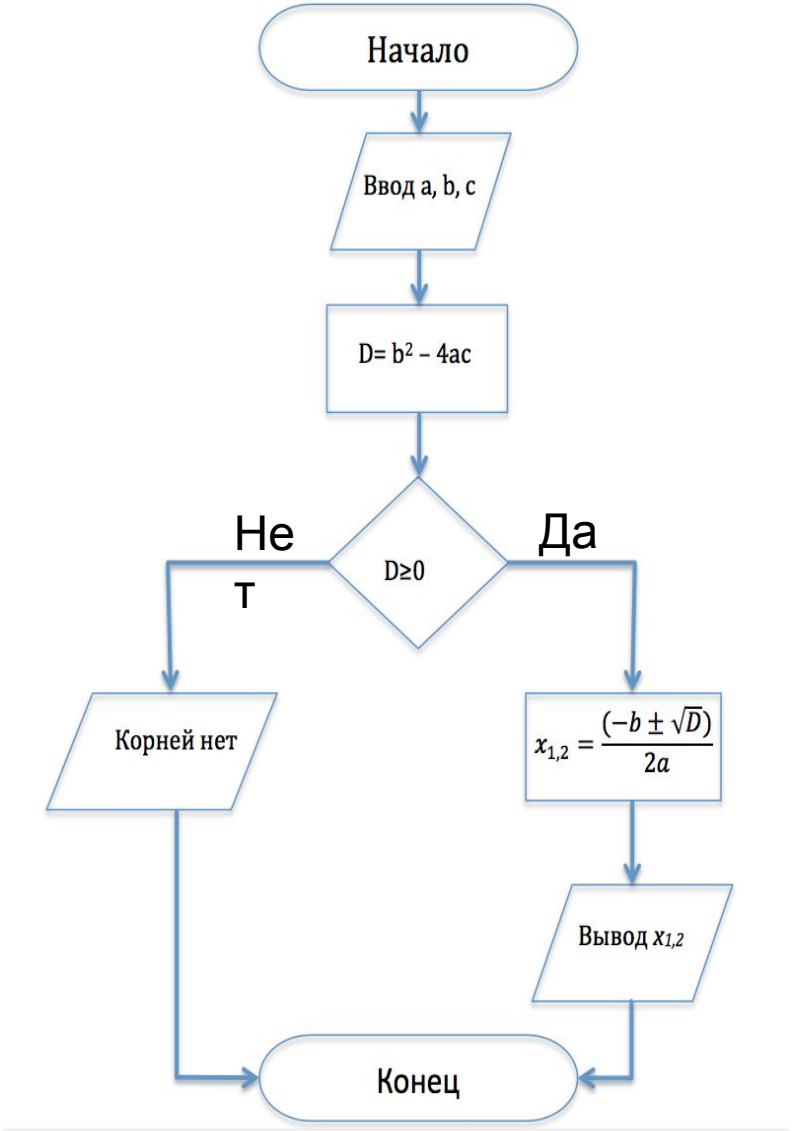
- **Блок решения** или **арифметический** (. Надпись на блоке: операция или группа операций.



Условный блок. Надпись на блоке: условие. В результате проверки условия осуществляется выбор одного из возможных путей (ветвей) вычислительного процесса. Если условие выполняется, то следующим выполняется этап по ветви «Да", если условие не выполняется, то выполняется этап по ветви «Нет".

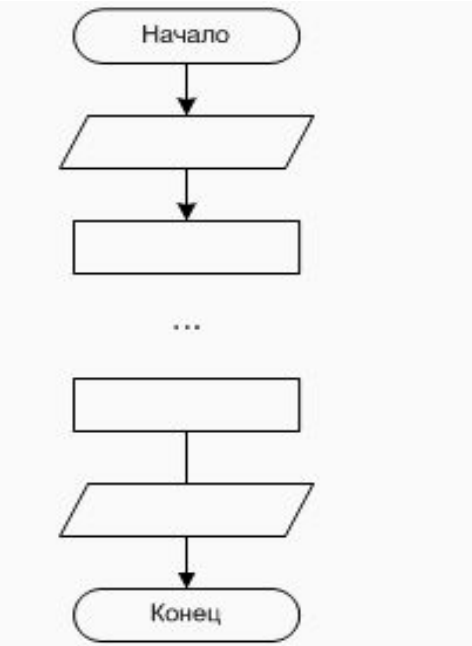


Блок-схема алгоритма решения квадратного уравнения



Виды алгоритмов

- **Линейный алгоритм -** это такой, в котором все операции выполняются последовательно одна за другой

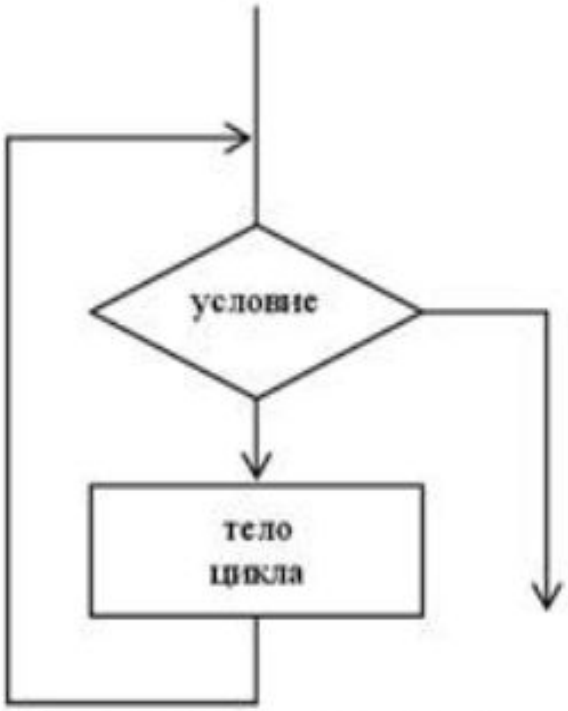


- Алгоритмы **разветвленной** структуры применяются, когда в зависимости от некоторого условия необходимо выполнить либо одно, либо другое действие. В блок-схемах разветвленные алгоритмы изображаются так:

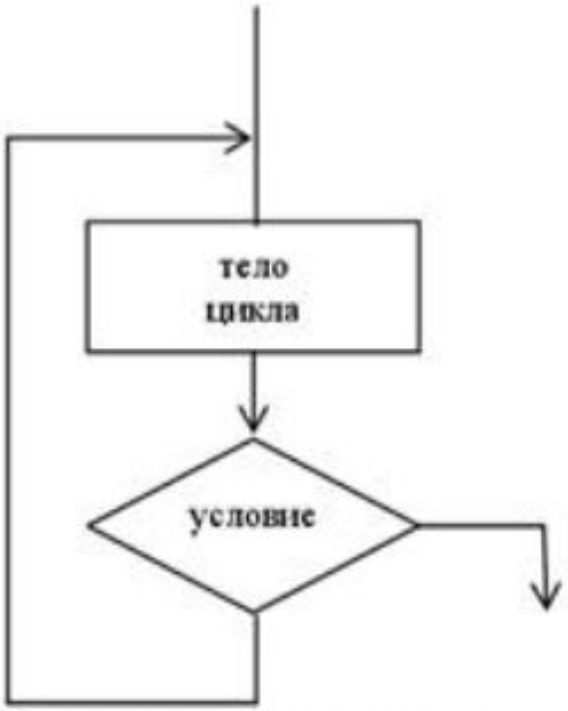
- Или так:

Вариант обозначения «Да» - «+»; «Нет» – «-».

- Циклическая структура алгоритма позволяет организовывать повторные однотипные вычисления:



Цикл с предусловием

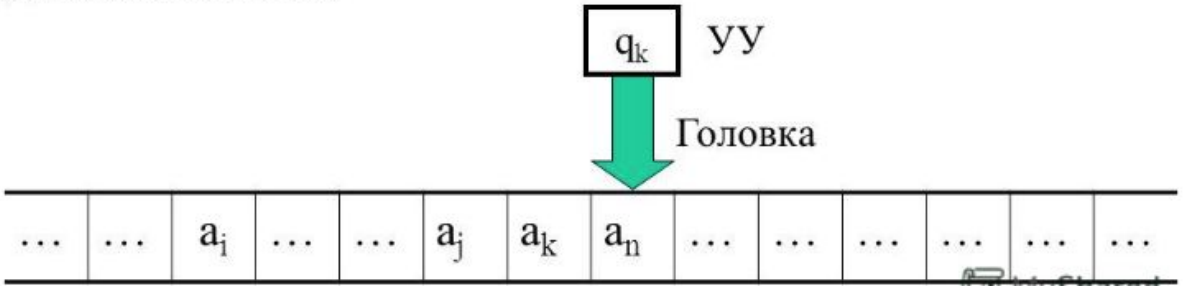


Цикл с постусловием

Существуют другие уточнения понятия алгоритма

Машины Тьюринга

- Лента бесконечна в обе стороны.
- В ячейке может быть записан только один символ.
- Число возможных символов конечно и образует алфавит машины $A=\{a_1,\ldots,a_m\}$.
- Головка может находиться в одном состоянии из конечного множества возможных состояний $Q=\{q_1,\ldots,q_n\}$.
- Среди состояний выделяются начальное - q_1 и конечное - q_n.
- Набор правил задается таблицей: $q_i a_j \rightarrow q_i' a_j' d_k$, где d_k - движение головки.



Алгоритмы Маркова

Примером схемы нормального алгоритма в пятибуквенном алфавите $|*abc$ может служить схема

$$\begin{cases} |b & \to & ba| \\ ab & \to & ba \\ b & \to & \\ *| & \to & b* \\ * & \to & c \\ |c & \to & c \\ ac & \to & c| \\ c & \to & \cdot \end{cases}$$

Машины Поста

Система команд машины, включающая шесть действий

№ п/п	Команда	Запись команды	Описание действий
1	Шаг вправо	X→y	Сдвиг головки на одну секцию вправо
2	Шаг влево	X←y	Сдвиг головки на одну секцию влево
3	Установить метку	XMy	В обозреваемую секцию ставится метка
4	Стереть метку	XCy	Из обозреваемой секции удаляется метка
5	Передача управления	x → y_1, x → y_2	При отсутствии метки в обозреваемой секции управление передается команде y_1, при наличии – команде y_2
6	Остановка	x стоп	Прекращение работы машины

- Алан Тьюринг высказал предположение (известное как Тезис Чёрча — Тьюринга), что любой алгоритм в интуитивном смысле этого слова может быть представлен эквивалентной машиной Тьюринга. Уточнение представления о вычислимости на основе понятия машины Тьюринга (и других эквивалентных ей понятий) открыло возможности для строгого доказательства алгоритмической неразрешимости различных массовых проблем (то есть проблем о нахождении единого метода решения некоторого класса задач, условия которых могут варьироваться в известных пределах). Простейшим примером алгоритмически неразрешимой массовой проблемы является так называемая проблема применимости алгоритма (называемая также проблемой остановки). Она состоит в следующем: требуется найти общий метод, который позволял бы для произвольной машины Тьюринга (заданной посредством своей программы) и произвольного начального состояния ленты этой машины определить, завершится ли работа машины за конечное число шагов, или же будет продолжаться неограниченно долго.