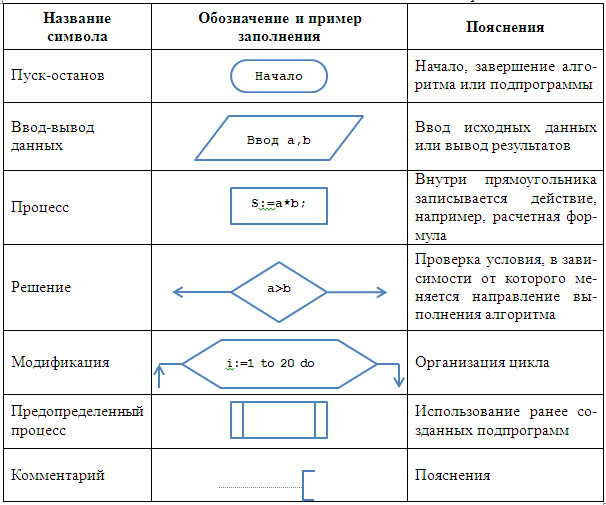
**Алгоритмы. Ветвление в алгоритмах**

**Алгоритм** – это ясный перечень действий, который направлен на решение какой-либо задачи. Одно из свойств алгоритма — **дискретность**. Дискретность связана с наличием в алгоритмической последовательности ряда операций (этапов, действий), выполняемых пошагово, то есть дискретно. Алгоритм обладает свойством дискретности, так как он представляет собой процесс решения задачи в виде последовательного выполнения простых шагов. И каждое действие исполняется лишь после окончания исполнения предыдущего. Также предполагается наличие определённых исходных данных и результата выполнения.

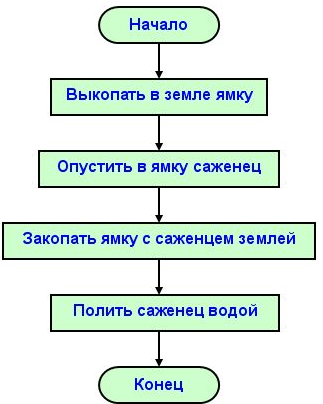
**Блок-схема** — графический способ описания алгоритмов. Графическое представление обеспечивает наглядность и упрощает запись, делая последовательность более понятной. При использовании схемы каждому действию соответствует определённая геометрическая фигура (эти фигуры называют блоками). Вот наиболее часто употребляемые:



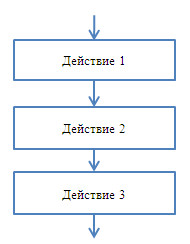
**Ещё раз о линейности**

Линейная последовательность — самая простая из возможных структур. При наличии линейности команды выполняются в чёткой последовательности и в порядке их записи, то есть друг за другом. Вот линейная алгоритмическая последовательность посадки дерева: 1) выкапывание ямки в земле; 2) размещение в ямке саженца; 3) закапывание ямки; 4) поливание места посадки водой.

Такой линейный алгоритм имеет следующую блок-схему:



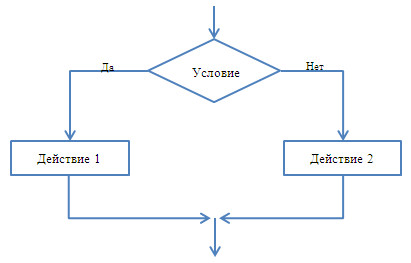
А вот и общая схема линейного алгоритма:



**Ветвление в алгоритмических последовательностях**

От значения условий зависит дальнейшее поведение. Когда условие выполняется, оно принимает значение «истина», когда нет — «ложь». Иногда анализ ситуации и выбор не вызывают особых затруднений, а иногда принять решение очень трудно. А всё потому, что принимающий решение пытается продумать каждый из вариантов и предугадать последствия выбора. Нельзя не вспомнить гроссмейстера, который анализирует позицию на ходы вперёд, прежде чем передвинуть фигуру на шахматной доске.

Компьютерные программы и игры тоже построены на выборе действий. А блок-схема при наличии ветвления приобретает иной вид:



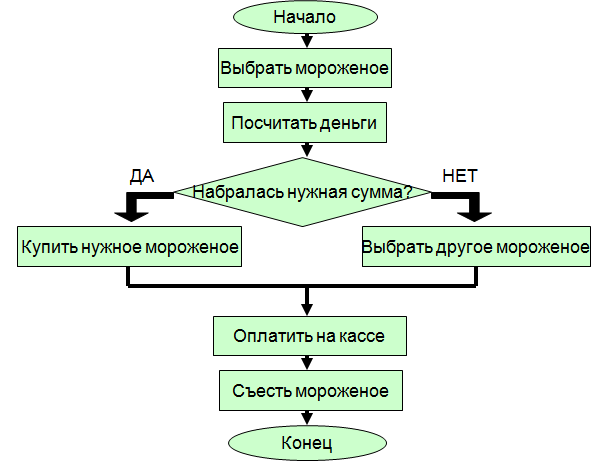
**Логика разветвляющих алгоритмов**

Логику можно описать следующим образом:

ЕСЛИ <условие истинно> ТО <действие 1> ИНАЧЕ <действие 2>

**Ветвление** — метод и форма организации действий, когда в зависимости от выполнения определённого условия совершается та либо иная последовательность шагов.

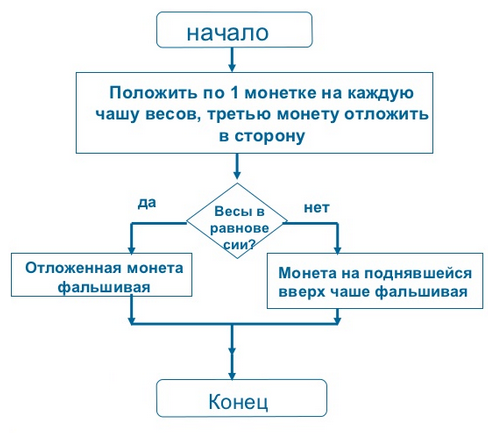
В результате совсем несложно составить алгоритм покупки мороженого с учётом наличия необходимой суммы денег. Описать эту алгоритмическую последовательность с помощью схемы и блоков тоже не составит труда:



Для закрепления можно решить задачу.

*Есть 3 монеты одинакового достоинства. Одна из монет фальшивая (известно, что она имеет меньший вес). Найдите фальшивую монету на чашечных весах без гирь с помощью только одного взвешивания.*

Решение легко описывается посредством схематических блоков:



Следующий пример легко экстраполируется в жизнь. Речь идёт об алгоритме для перехода дороги при наличии светофора. Он имеет следующий вид: 1. Подходим к светофору. 2. Смотрим, какой горит свет. 3. Если зелёный, переходим дорогу. 4. Если красный, ждём, пока загорится зелёный, а потом переходим дорогу.

Соответствующая блок-схема:



**Программный способ записи**

Чтобы алгоритм было понятен компьютеру, машине и любой другой цифровой системе, следует оформить его в таком виде, который эта система способна воспринимать. То есть надо написать программу, используя для этого команды из **СКИ**. СКИ — это список команд исполнителя — перечень команд, ему понятных. А любой исполнитель способен исполнить лишь те команды, которые включены в его СКИ, а если говорить человеческим языком — входят в набор его компетенций.

Для примера можно реализовать алгоритм на языке программирования **Pascal**. Исходя из вышесказанного, следует использовать команды, входящие в терминологию Pascal.

Простейший пример описания алгоритма с разветвляющейся структурой — **условный оператор IF**. Полная конструкция этого условного оператора имеет следующий вид:

if<логическое выражение>then<оператор 1>else<оператор 2>

Здесь if — это «если», then — это «то», else — «иначе».

Условный оператор работает просто: — вычисляется значение логического выражения, которое расположено после служебного слова IF; — если результат — истина, выполняется оператор 1, который размещён после THEN, причём действие после ELSE пропускается; — если результат — ложь, пропускается уже действие после THEN, а действие после ELSE выполняется с помощью оператора 2.

Теперь можно вспомнить пресловутого витязя на распутье и написать простую программу, реализующую этот алгоритм с помощью соответствующих условных операторов.

program Algoritm\_vetvlenia;

Var x :string;

Begin

WriteLn ('Витязь, куда путь держишь?');

ReadLn (x);

If x='Направо' then writeLn ('Направо пойдёшь — жену найдёшь');

If x='Налево' then writeLn ('Налево пойдешь — богатым будешь');

If x='Прямо' then writeLn ('Прямо пойдёшь — смерть найдёшь');

ReadLn;

End.

Попробовать этот алгоритм в работе можно на любом онлайн-компиляторе, поддерживающим Pascal. Но не стоит на этом останавливаться — лучше всего написать собственную программу, что позволит получить максимальную пользу от урока.