

Конспект занятия "Основы алгоритмики"

Лекция:

Цели текущего занятия:

1. Познакомиться с алгоритмами. Узнать, что это такое и как с этим работать.
2. Познакомиться с блок-схемами. Научиться их читать.

По плану мы сначала изучим сами алгоритмы, а затем уже их представление в виде блок-схем.

Итак, что такое алгоритм? Алгоритм — это последовательность действий, которая направлена на достижение окончательного решения проблемы наиболее оптимальными и эффективными способами.

Разберем определение по порядку. Алгоритм - это последовательность действий, то есть действия идут друг за другом. Далее, это не просто последовательность действий, а последовательность, которая направлена на достижение окончательного решения проблемы. Получается, что при выполнении каких-то действий в определенном порядке мы должны решить определенную проблему. Причем не просто решить, а решить наиболее оптимальным и эффективным способом. Что значит оптимальным? Проще говоря, оптимальный алгоритм - потому что решает вашу задачу лучшим способом. А эффективный, потому что для того способа, который вы выбрали, все его действия выбраны и отлажены так, чтобы результат был получен максимально удовлетворительный.

Например, вы хотите выкопать большую яму. Яму можно выкопать самостоятельно с помощью лопаты, нанять рабочих или вызвать экскаватор. Лучший способ или оптимальный - это выбрать тот вариант, который удовлетворяет вас по времени и затратам. Например, выкопать самостоятельно яму - дешево, но придется потратить много личного времени. Поручить выкапывание отряду рабочих - с одной стороны быстро, с другой стороны есть риски с тем, что выкопаю не там или захватят с собой что-то с вашего участка на память. Если же вызывать экскаватор, то это максимально быстро и безопасно, но и очень дорого.

Получается, оптимальный выбор будет зависеть от входных данных и допустимых рисков, которые подразумевает задача.

А что же с эффективностью? Допустим, мы решили копать яму самостоятельно. Нам надо вырыть её как можно скорее и мы можем копать, не переставая, 5 часов подряд. А можем делать перерывы. В обоих случаях яма будет вырыта, согласно одному и

тому же алгоритму, но вот затраты энергии и даже удовлетворенность результатом будут разными. Получается, эффективность - это баланс действий внутри алгоритма.

Алгоритмы существуют не сами по себе — они предназначаются для конкретного исполнителя. Кто может выступать таким исполнителем:

- человек;
- роботизированное/автоматизированное устройство, механизм;
- компьютер;
- язык программирования и т. д.

Отличительная черта исполнителя — способность выполнять команды, которые включены в алгоритм. Это становится возможным, благодаря описанию последнего на формальном языке, который исключает неоднозначность толкования. Действия, которые должен выполнить исполнитель, называют элементарными действиями, а сама запись алгоритмической последовательности на формальном языке — это программа.

Существует несколько видов алгоритмов, рассмотрим основные из них: линейный, алгоритм ветвления и циклический.

Линейный алгоритм — это последовательное выполнение инструкций в строгой очередности их расположения. То есть все действия выполняются строго друг за другом. Сначала первое, потом второе и так далее, не нарушая порядок.

Алгоритм ветвления — это последовательность действий в соответствии с определенными условиями. Иначе говоря, если условие выполняется, то делаем такие-то действия, если не выполняется, то делаем другие действия. Подобные действия, зависящие от каких-то факторов, и отражает алгоритм ветвления.

Циклические алгоритмы — это последовательность действий, которую необходимо повторять несколько раз для достижения положительного результата. То есть если мы должны выполнить одни и те же действия несколько раз для достижения конечного результата. Эти повторяющиеся действия - цикл - описывает именно циклический алгоритм.

Посмотрим на примеры из жизни, где применяются подобные алгоритмы. Начнём с линейного алгоритма. Например, человеку нужно постирать одежду. Построим линейный алгоритм для решения этой задачи:

1. Собрать грязную одежду
2. Положить одежду в стиральную машину
3. Засыпать в машинку стиральный порошок
4. Залить кондиционер
5. Выбрать программу для стирки
6. Нажать на кнопку Пуск

Все действия в этом алгоритме выполняются друг за другом. Вряд ли мы сначала включим машину, а потом попытаемся положить туда одежду. И в результате

последовательного выполнения данных действий мы можем достичь результата - выстиранная одежда.

Следующий алгоритм - алгоритм ветвления. Пример из жизни можно привести следующий: лампочка в подъезде включается, когда срабатывает датчик движения. Наверняка каждый сталкивался с таким поведением включения света, по какому алгоритму он может работать? Предположим, что в основе алгоритма следующее поведение:

1. Датчик получил сигнал о движении?
 - a. Если да, то надо зажечь лампочку
 - b. Если нет, то ничего не делать

Получается, что при получении сигнала датчик зажигает лампочку, если же сигнал не получен или получен сигнал не о движении, а о чем-то ещё, то лампочка не зажигается.

И последний алгоритм - циклический. Пример для него из жизни может быть таким: Для телевизора в детской комнате установили таймер. Теперь дети могут смотреть телевизор только 1 час, затем телевизор выключается. Как можно описать подобный алгоритм? Мы знаем, что у телевизора есть какая-то программа, которая имеет переменную в виде таймера, значение которой увеличивается каждую минуту. Пусть алгоритм будет следующим:

1. Поставить таймер после включения экрана
2. Увеличивать таймер на 1 каждую минуту
3. Повторять действие 2, пока таймер не достигнет 1 часа

Таким образом под изученные нами виды алгоритмов можно найти множество примеров из обычной жизни.


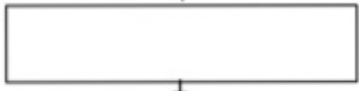
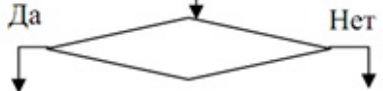
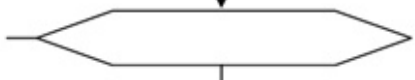
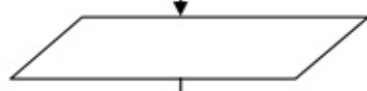
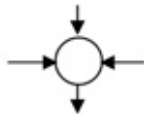
Кроме того, что все алгоритмы объединены одним определением, они также имеют общие свойства:

- Конечность - алгоритм всегда должен заканчиваться за конечное число шагов, но это число не ограничено сверху.
- Массовость - алгоритм применяется к некоторому классу входных данных (чисел, пар чисел, набору букв и тому подобному).
- Дискретность - каждая алгоритмическая последовательность действий делится на шаги, а процесс решения задачи — это последовательное исполнение данных шагов. Также дискретность означает, что для выполнения каждого этапа потребуется конечный временной отрезок.
- Понятность - в алгоритм должны быть включены лишь те команды, которые доступны и понятны исполнителю, то есть входят в систему его команд.
- Определенность — каждый шаг алгоритма должен быть строго определенным, то есть различные толкования должны быть исключены. Также все шаги имеют конкретное место в последовательном выполнении.

Это лишь некоторые свойства алгоритмов, которые важно знать начинающему тестировщику. Чтобы узнать больше, можно обратиться к ссылкам в доп. материалах.

Алгоритмы, которые, мы с вами рассматривали, были описаны простым текстом. Но чем больше и сложнее алгоритм, тем сложнее описать его текстом так, чтобы это было понятно другим людям. По этой причине были придуманы блок-схемы - схематические представления алгоритмов.

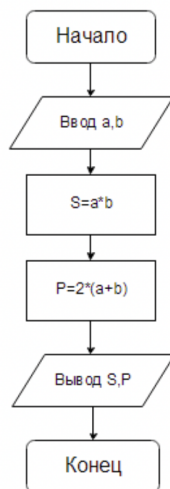
Блок-схемы могут состоять из нескольких разных блоков:

Фигура	Характер действий
	Начало, конец алгоритма
	Любые действия, чаще всего арифметические; блоки действий
	Проверка условий
	Начало, конец цикла
	Ввод, вывод данных
	Соединитель в разветвляющихся алгоритмах. Ссылка при переносе части алгоритма на другую страницу (хотя этого следует избегать)

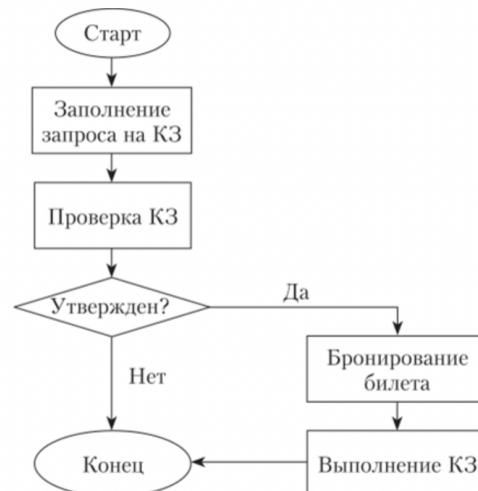
Эти блоки иногда могут незначительно менять свой внешний вид, в зависимости от принятых в компании стандартов описания блок-схем. Однако в целом их суть остается такой же.

Давайте попробуем прочитывать следующие блок-схемы:

Линейный алгоритм



Алгоритм ветвления



На первой блок-схеме изображен линейный алгоритм для определения площади и периметра прямоугольника. У него есть блоки начало и конец (это обязательно для любого алгоритма), ввод и вывод данных и блоки действий. Блок-схема читается сверху вниз. Таким образом, данный алгоритм можно описать так:

1. Ввести значения переменных a и b (стороны)
2. Посчитать площадь по формуле
3. Посчитать периметр
4. Вывести полученные значения площади и периметра.

Второй алгоритм чуть сложнее. Он описан не на языке программы (без переменных и арифметических операций), но тоже может быть представлен в виде блок-схемы. В данном алгоритме идет речь об оформлении командировки. Действия "Заполнение запроса" и "Проверка КЗ" представлены в блоках действий, а далее идёт условие "Утвержден?". Если условие выполняется, то алгоритм продолжает выполняться по правой ветке. Если не выполняется, то по левой.

Подведем итог. На сегодняшнем занятии мы познакомились с понятием алгоритма, рассмотрели, какие виды алгоритмов бывают и изучили основные элементы блок-схем. Всё это пригодится нам для понимания, как алгоритмы превращаются в код. На практическом занятии мы ещё раз повторим, как описываются элементы блок-схем на практике и попробуем решить несколько задач.

Задачи:

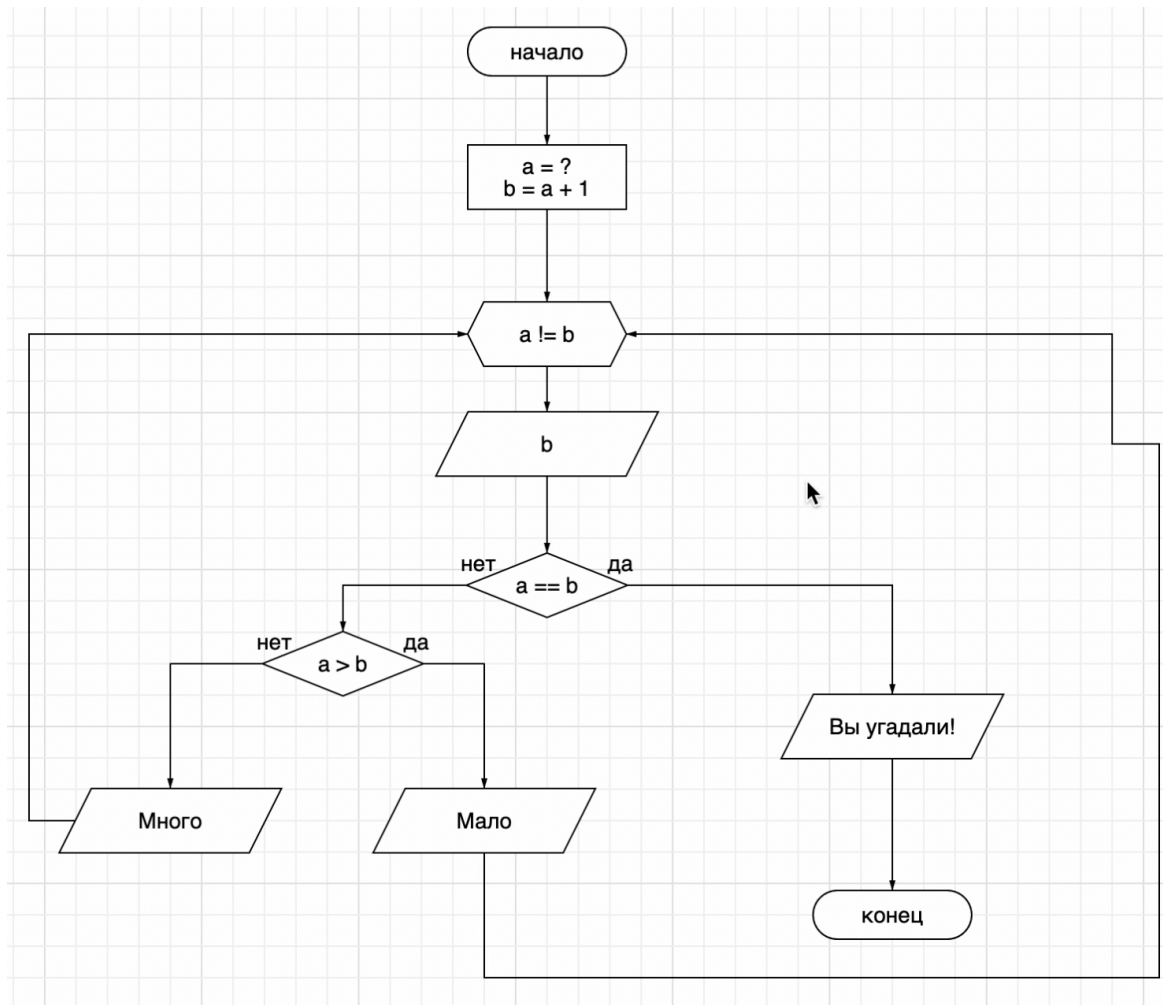
На практическом занятии мы составили блок-схемы для следующих задач:

1. Игра с компьютером в угадывание числа:
 - Компьютер загадывает число A
 - Игрок вводит число B
 - Если $B > A$ – сообщение «Много», игрок снова вводит B
 - Если $B < A$ – сообщение «Мало», игрок снова вводит B

- Если $B = A$ – сообщение «Вы угадали», завершение работы

Решаем с помощью цикла, так как программа завершается только при выигрыше пользователя. Изначально задаем значение для b , чтобы провести первую проверку, далее это значение меняется на значение, введенное пользователем.

Итоговая блок схема:



2. Известен вес боксера-любителя.

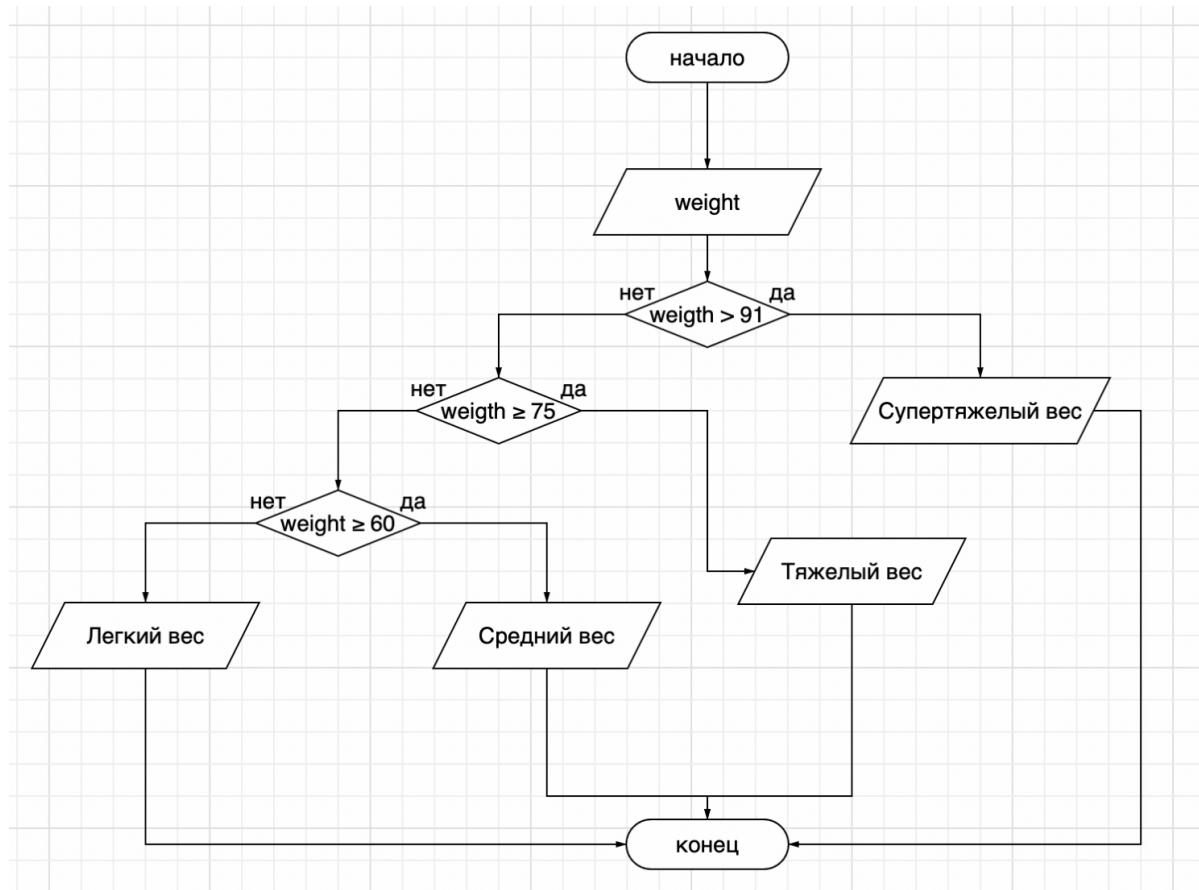
Известно, что вес таков, что боксер может быть отнесен к одной из четвертых весовых категорий:

- легкий вес — до 60 кг;
- средний вес — до 75 кг;
- тяжелый вес — до 91 кг;
- супертяжелый вес — с 91 кг.

Определить, в какой категории будет выступать данный боксер.

Решаем с помощью ветвления, обязательно учитываем все варианты веса, обращаем внимание, что знак $<$ означает, что число после него не включено в проверку. То есть сравнение $a < 75$ вернет “нет”, если $a = 75$ или больше него.

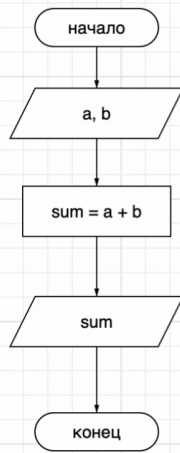
Итоговая блок схема:



Также в процессе изучения элементов мы составили простые схемы для каждого из видов алгоритмов:

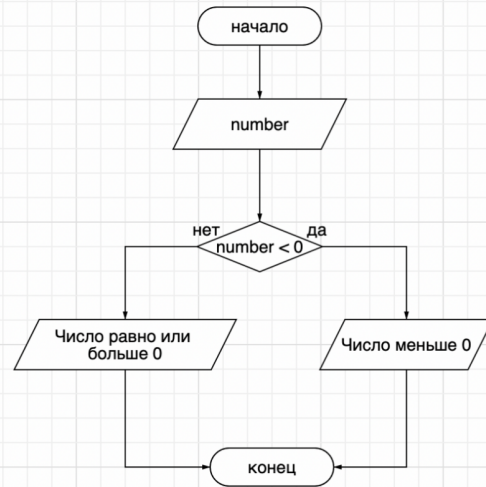
Линейный алгоритм

Пользователь вводит два числа, программа выводит их сумму



Алгоритм ветвления

Пользователь вводит число, если оно меньше нуля, выводится соответствующее сообщение, если равно или больше 0, то выводится другое соответствующее сообщение



Циклический алгоритм

Пользователь вводит число
Программа выводит это число 5 раз

