- Слово "Алгоритм" происходит от algorithmi - латинского написания имени аль-Хорезми, под которым в средневековой Европе знали величайшего математика из Хорезма (город в современном Узбекистане) Мухаммеда бен Мусу, жившего в 783-850 гг. В своей книге "Об индийском счете" он сформулировал правила записи натуральных чисел с помощью арабских цифр и правила действий над ними столбиком.
- В дальнейшем алгоритмом стали называть **точное предписание, определяющее последовательность действий, обеспечивающую получение требуемого результата из исходных данных.** Алгоритм может быть предназначен для выполнения его человеком или автоматическим устройством. Создание алгоритма, пусть даже самого простого, - процесс творческий. Он доступен исключительно живым существам, а долгое время считалось, что только человеку. Другое дело - реализация уже имеющегося алгоритма. Ее можно поручить субъекту или объекту, который не обязан вникать в существо дела, а возможно, и не способен его понять.

- Такой субъект или объект принято называть **формальным исполнителем.** Примером формального исполнителя может служить стиральная машина-автомат, которая неукоснительно исполняет предписанные ей действия, даже если вы забыли положить в нее порошок. Человек тоже может выступать в роли формального исполнителя, но в первую очередь формальными исполнителями являются различные автоматические устройства, и компьютер в том числе. **Каждый алгоритм создается в расчете на вполне конкретного исполнителя.** Те действия, которые может совершать исполнитель, называются его **его допустимыми действиями**. Совокупность допустимых действий образует **систему команд исполнителя.** Алгоритм должен содержать только те действия, которые допустимы для данного исполнителя.

- Объекты, над которыми исполнитель может совершать действия, образуют так называемую **среду исполнителя.** Для алгоритмов, встречающихся в математике, средой того или иного исполнителя могут быть числа разной природы - натуральные, действительные и т.п., буквы, буквенные выражения, уравнения, тождества и т.п.
- Данное выше определение алгоритма нельзя считать строгим - не вполне ясно, что такое "точное предписание" или "последовательность действий, обеспечивающая получение требуемого результата". Поэтому обычно формулируют несколько общих свойств алгоритмов, позволяющих отличать алгоритмы от других инструкций.

- Такими свойствами являются:
- **Дискретность (прерывность, раздельность)** - алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов. Каждое действие, предусмотренное алгоритмом, исполняется только после того, как закончилось исполнение предыдущего.
- **Определенность** - каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвола. Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.
- **Результативность (конечность)** - алгоритм должен приводить к решению задачи за конечное число шагов.
- **Массовость** - алгоритм решения задачи разрабатывается в общем виде, то есть, он должен быть применим для некоторого класса задач, различающихся только исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется **областью применимости алгоритма.**

- На основании этих свойств иногда дается определение алгоритма, например:
- *“Алгоритм – это последовательность математических, логических или вместе взятых операций, отличающихся детерменированностью, массовостью, направленностью и приводящая к решению всех задач данного класса за конечное число шагов.”*
- Такая трактовка понятия “алгоритм” является неполной и неточной. Во-первых, неверно связывать алгоритм с решением какой-либо задачи. Алгоритм вообще может не решать никакой задачи. Во-вторых, понятие “массовость” относится не к алгоритмам как к таковым, а к математическим методам в целом.
- Решение поставленных практикой задач математическими методами основано на абстрагировании – мы выделяем ряд существенных признаков, характерных для некоторого круга явлений, и строим на основании этих признаков математическую модель, отбрасывая несущественные признаки каждого конкретного явления. В этом смысле любая математическая модель обладает свойством массовости. Если в рамках построенной модели мы решаем задачу и решение представляем в виде алгоритма, то решение будет “массовым” благодаря природе математических методов, а не благодаря “массовости” алгоритма.

- Алгоритм – искусственная конструкция, которую мы сооружаем для достижения своих целей. Чтобы алгоритм выполнил свое предназначение, его необходимо строить по определенным правилам. Поэтому нужно говорить не о свойствах алгоритма, а о правилах построения алгоритма, или о требованиях, предъявляемых к алгоритму.

- **Первое правило** – при построении алгоритма прежде всего необходимо задать множество объектов, с которыми будет работать алгоритм. Формализованное (закодированное) представление этих объектов носит название данных. Алгоритм приступает к работе с некоторым набором данных, которые называются входными, и в результате своей работы выдает данные, которые называются выходными. Таким образом, алгоритм преобразует входные данные в выходные.
- Это правило позволяет сразу отделить алгоритмы от “методов” и “способов”. Пока мы не имеем формализованных входных данных, мы не можем построить алгоритм.

- **Второе правило** – для работы алгоритма требуется память. В памяти размещаются входные данные, с которыми алгоритм начинает работать, промежуточные данные и выходные данные, которые являются результатом работы алгоритма. Память является дискретной, т.е. состоящей из отдельных ячеек. Поименованная ячейка памяти носит название переменной. В теории алгоритмов размеры памяти не ограничиваются, т. е. считается, что мы можем предоставить алгоритму любой необходимый для работы объем памяти.

- **Третье правило** – дискретность. Алгоритм строится из отдельных шагов (действий, операций, команд). Множество шагов, из которых составлен алгоритм, конечно.
- **Четвертое правило** – детерменированность. После каждого шага необходимо указывать, какой шаг выполняется следующим, либо давать команду остановки.
- **Пятое правило** – сходимость (результативность). Алгоритм должен завершать работу после конечного числа шагов. При этом необходимо указать, что считать результатом работы алгоритма.

- Любая работа на компьютере – это есть обработка информации. Работу компьютера можно схематически изобразить следующим образом:
- “Информация” слева и “информация” справа – это разные информации. Компьютер воспринимает информацию извне и в качестве результата своей работы выдает новую информацию. Информация, с которой работает компьютер, носит название “данные”.
- Компьютер преобразует информацию по определенным правилам. Эти правила (операции, команды) заранее занесены в память компьютера. В совокупности эти правила преобразования информации называются алгоритмом. Данные, которые поступают в компьютер, называются входными данными. Результат работы компьютера – выходные данные. Таким образом, алгоритм преобразует входные данные в выходные:

- Трактовка работы алгоритма как преобразования входных данных в выходные естественным образом подводит нас к рассмотрению понятия “постановка задачи”. Для того, чтобы составить алгоритм решения задачи, необходимо из условия выделить те величины, которые будут входными данными и четко сформулировать, какие именно величины требуется найти. Другими словами, условие задачи требуется сформулировать в виде “Дано … Требуется” – это и есть постановка задачи.

- Виды алгоритмов как логико-математических средств отражают указанные компоненты человеческой деятельности и тенденции, а сами алгоритмы в зависимости от цели, начальных условий задачи, путей ее решения, определения действий исполнителя подразделяются следующим образом:
- **Механические алгоритмы**, или иначе детерминированные, жесткие (например алгоритм работы машины, двигателя и т.п.);
- **Гибкие алгоритмы**, например стохастические, т.е. вероятностные и эвристические.
- Механический алгоритм задает определенные действия, обозначая их в единственной и достоверной последовательности, обеспечивая тем самым однозначный требуемый или искомый результат, если выполняются те условия процесса, задачи, для которых разработан алгоритм.

- **Вероятностный (стохастический) алгоритм** дает программу решения задачи несколькими путями или способами, приводящими к вероятному достижению результата.
- **Эвристический алгоритм** (от греческого слова “эврика”) – это такой алгоритм, в котором достижение конечного результата программы действий однозначно не предопределено, так же как не обозначена вся последовательность действий, не выявлены все действия исполнителя. К эвристическим алгоритмам относят, например, инструкции и предписания. В этих алгоритмах используются универсальные логические процедуры и способы принятия решений, основанные на аналогиях, ассоциациях и прошлом опыте решения схожих задач.
- **Линейный алгоритм** – набор команд (указаний), выполняемых последовательно во времени друг за другом.

- **Разветвляющийся алгоритм** – алгоритм, содержащий хотя бы одно условие, в результате проверки которого ЭВМ обеспечивает переход на один из двух возможных шагов.
- **Циклический алгоритм** – алгоритм, предусматривающий многократное повторение одного и того же действия (одних и тех же операций) над новыми исходными данными. К циклическим алгоритмам сводится большинство методов вычислений, перебора вариантов.
- **Цикл программы** – последовательность команд (серия, тело цикла), которая может выполняться многократно (для новых исходных данных) до удовлетворения некоторого условия.

- **Структурная (блок-, граф-) схема алгоритма** – графическое изображение алгоритма в виде схемы связанных между собой с помощью стрелок (линий перехода) блоков – графических символов, каждый из которых соответствует одному шагу алгоритма. Внутри блока дается описание соответствующего действия.
- Графическое изображение алгоритма широко используется перед программированием задачи вследствие его наглядности, т.к. зрительное восприятие обычно облегчает процесс написания программы, ее корректировки при возможных ошибках, осмысливание процесса обработки информации.

- Принцип программирования “сверху вниз” требует, чтобы блок-схема поэтапно конкретизировалась и каждый блок “расписывался” до элементарных операций. Но такой подход можно осуществить при решении несложных задач. При решении сколько-нибудь серьезной задачи блок-схема “расползется” до такой степени, что ее невозможно будет охватить одним взглядом.
- Блок-схемы алгоритмов удобно использовать для объяснения работы уже готового алгоритма, при этом в качестве блоков берутся действительно блоки алгоритма, работа которых не требует пояснений. Блок-схема алгоритма должна служить для упрощения изображения алгоритма, а не для усложнения.

- Разработка программы – это не только написание программы. Написание программы является одним из этапов. Для начала перечислим все этапы разработки программ, а затем подробно расскажем о них.

- **Этапы разработки программ:**
- **1. Постановка задачи**
- 1. Формулировка и анализ физической задачи
- 2. Составление математической модели
- 3. Составление алгоритма задачи
- **2. Создание программы**
- 1. Составление текста программы
- 2. Ввод текста программы в компьютер
- 3. Синтаксическая отладка программы
- 4. Тестирование и семантическая отладка
- 5. Документирование программы
- **3. Запуск готовой программы и анализ полученных результатов**

- **Формулировка и анализ физической задачи**
- *Формулировка задачи* – это само её объявление, её постановка.
- Но просто формулировка ничем не поможет программистам. Для этого и существует второй подэтап – это анализ задачи.
- *Анализ задачи* – это подробный просмотр задачи с определением и выявлением входной и выходной информации. (*Входная информация по задаче* — это данные, поступающие на вход задачи и используемые для её решения. *Выходная информация* – это результат.)
- После проведения анализа поставленной задачи программисту более или менее понятно, с какими проблемами ему придется столкнуться.

- **Составление математической модели**

- Для более четкого понимания рассмотрим определения математической модели:
- *«Математическая модель - система уравнений и концепций, используемых для описания и прогнозирования данного феномена или поведения объекта.»*
- Математические модели находят как практическое, так и теоретическое применение (иногда одновременно). Практические задачи, в которых используются математические модели, включают создание новых материалов, предсказание погоды, проверку прочности мостов, самолетов и тому подобного» - это определение используется в физике, химии и математической биологии.
- *Математическая модель в программировании* – это система математических соотношений, приближенно отражающий сформулированную задачу. И она позволяет осуществить предварительный выбор оптимальных вариантов решений по определенным критериям.
- Создание математической модели не занимает много времени, т.к. задача подробно разобранна по предыдущему пункту.

- **Составление алгоритма задачи**
- У алгоритма есть 2 обязательных условия:
- 1) Алгоритм должен быть представлен в форме, понятной человеку, который его разрабатывает.
- 2) Алгоритм должен быть представлен в форме, понятной тому объекту (в том числе и человеку), который будет выполнять описанные в алгоритме действия.
- Так же у алгоритмов есть свойства:
- 1. Дискретность
- 2. Детерминированность
- 3. Конечность
- 4. Массовость
- 5. Результативность
- В мире существует несколько видов алгоритмов:
- · Линейный алгоритм (описание действий, которые выполняются однократно в заданном порядке);
- · Циклический алгоритм (описание действий, которые должны повторятся указанное число раз или пока не выполнено условие);
- · Разветвляющий алгоритм (алгоритм, в котором в зависимости от условия выполняется либо одна, либо другая последовательность действий);

- **Составление текста программы**
- Это, наверное, самый сложный из этапов, требующий наибольшего внимания. По сути, составление текста программы – это запись алгоритма задачи при помощи одного из языков программирования. Чтобы этот текст был понятен пользователю и составителю, используются комментарии.
-

- **Синтаксическая отладка программы**
- *Отладка программы* — это специальный этап в разработке программы, состоящий в выявлении и устранении программных ошибок, факт существования которых уже установлен.
- *Синтаксическая отладка* – поиск синтаксических ошибок в тексте программы. Обнаружив ошибку, транслятор выводит сообщение, указывая на место ошибки в программе и ее характер. Получив такое сообщение, программист должен исправить ошибку и снова повторить трансляцию. Так продолжается до тех пор, пока не будут исправлены все синтаксические ошибки.
- Если вы сталкиваетесь с синтаксической ошибкой, то чаще всего вы можете решить проблему с помощью справочной системы, из которой можно получить дополнительную информацию об ошибке, и исправить эту ошибку, уделив дополнительное внимание точному синтаксису используемых вами функций, объектов, методов и свойств.

- **Тестирование и семантическая отладка**
- Тестирование – это динамический контроль программы, т.е. проверка правильности программы при ее выполнении на компьютере. На этот этап приходится около 50% общей стоимости разработки программного обеспечения.

- Основные принципы организации тестирования:
- **1.** необходимой частью каждого теста должно являться описание ожидаемых результатов работы программы, чтобы можно было быстро выяснить наличие или отсутствие ошибки в ней;
- **2.** следует по возможности избегать тестирования программы ее автором, т.к. кроме уже указанной объективной сложности тестирования для программистов здесь присутствует и тот фактор, что обнаружение недостатков в своей деятельности противоречит человеческой психологии (однако отладка программы эффективнее всего выполняется именно автором программы);

- **3.** по тем же соображениям организация - разработчик программного обеспечения не должна “единолично ” его тестировать (должны существовать организации, специализирующиеся на тестировании программных средств);
- **4.** должны являться правилом доскональное изучение результатов каждого теста, чтобы не пропустить малозаметную на поверхностный взгляд ошибку в программе;
- **5.** необходимо тщательно подбирать тест не только для правильных (предусмотренных) входных данных, но и для неправильных (непредусмотренных);
- **6.** при анализе результатов каждого теста необходимо проверять, не делает ли программа того, что она не должна делать;

- **7.** следует сохранять использованные тесты (для повышения эффективности повторного тестирования программы после ее модификации или установки у заказчика);
- **8.** тестирования не должно планироваться исходя из предположения, что в программе не будут обнаружены ошибки (в частности, следует выделять для тестирования достаточные временные и материальные ресурсы);
- **9.** следует учитывать так называемый “принцип скопления ошибок” : вероятность наличия не обнаруженных ошибок в некоторой части программы прямо пропорциональна числу ошибок, уже обнаруженных в этой части;
- **10.** следует всегда помнить, что тестирование - творческий процесс, а не относиться к нему как к рутинному занятию.