Понятие алгоритма и формы записи алгоритмов

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 допустимыми действиями. Совокупность допустимых действий образует систему команд исполнителя. Алгоритм должен содержать только те действия, которые допустимы для данного исполнителя.

Свойства алгоритмов

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.

Правила выполнения арифметических операций или геометрических построений представляют собой алгоритмы. При этом остается без ответа вопрос, чем же отличается понятие алгоритма от таких понятий, как «метод», «способ», «правило». Можно даже встретить утверждение, что слова «алгоритм», «способ», «правило» выражают одно и то же (т.е.

являются синонимами), хотя такое утверждение, очевидно, противоречит “свойствам алгоритма”.

Само выражение «свойства алгоритма» не совсем корректно. Свойствами обладают объективно существующие реальности. Можно говорить, например, о свойствах какого-либо вещества. Алгоритм – искусственная конструкция, которую мы сооружаем для достижения своих целей. Чтобы алгоритм выполнил свое предназначение, его необходимо строить по определенным правилам. Поэтому нужно говорить все же 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.

В школьной «теории алгоритмов» эти два правила не рассматриваются. В то же время практическая работа с алгоритмами (программирование) начинается именно с реализации этих правил. В языках программирования распределение памяти осуществляется декларативными операторами (операторами описания переменных). В языке Бейсик не все переменные описываются, обычно описываются только массивы. Но все равно при запуске программы транслятор языка анализирует все идентификаторы в тексте программы и отводит память под соответствующие переменные.

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.

Итак, алгоритм – неопределяемое понятие теории алгоритмов. Алгоритм каждому определенному набору входных данных ставит в соответствие некоторый набор выходных данных, т. е. вычисляет (реализует) функцию. При рассмотрении конкретных вопросов в теории алгоритмов всегда имеется в виду какая-то конкретная модель алгоритма.

Виды алгоритмов и их реализация

Алгоритм применительно к вычислительной машине – точное предписание, т.е. набор операций и правил их чередования, при помощи которого, начиная с некоторых исходных данных, можно решить любую задачу фиксированного типа.

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.

•Вспомогательный (подчиненный) алгоритм (процедура) – алгоритм, ранее разработанный и целиком используемый при алгоритмизации конкретной задачи. В некоторых случаях при наличии одинаковых последовательностей указаний (команд) для различных данных с целью сокращения записи также выделяют вспомогательный алгоритм.

Методы изображение алгоритмов

На практике наиболее распространены следующие формы представления алгоритмов:

•словесная (записи на естественном языке);

•графическая (изображения из графических символов);

•псевдокоды (полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.);

•программная (тексты на языках программирования).

Словесное описание алгоритма

Данный способ получил значительно меньшее распространение из-за его многословности и отсутствия наглядности.

Рассмотрим пример на алгоритме нахождение максимального из двух значений:

Определим форматы переменных X, Y, M, где X и Y – значения для сравнения, M – переменная для хранения максимального значения;

- получим два значения чисел X и Y для сравнения;
- сравним X и Y.
- если X меньше Y, значит большее число Y.
- Поместим в переменную M значение Y.
- Если X не меньше (больше) Y, значит большее число X.
- Поместим в переменную M значение X.

Словесный способ не имеет широкого распространения по следующим причинам:

- такие описания строго не формализуемы;
- страдают многословностью записей;
- допускают неоднозначность толкования отдельных предписаний.

Блок-схема алгоритма

А этот способ оказался очень удобным средством изображения алгоритмов и получил широкое распространение в научной и учебной литературе.

> *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.

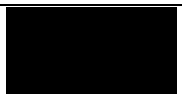
Можно встретить даже такое утверждение: «Внешне алгоритм представляет собой схему – набор прямоугольников и других символов, внутри которых записывается, что вычисляется, что вводится в машину и что выдается на печать и другие средства отображения информации «. Здесь форма представления алгоритма смешивается с самим алгоритмом.

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.

В таблице приведены наиболее часто употребляемые символы.

Название символа	Обозначение и пример заполнения	Пояснение
Процесс		Вычислительное действие или последовательность действий
Решение		Проверка условий

Модификация		Начало цикла
Предопределенный процесс		Вычисления по подпрограмме, стандартной подпрограмме
Ввод-вывод		Ввод-вывод в общем виде
Пуск-останов		Начало, конец алгоритма, вход и выход в подпрограмму
Документ		Вывод результатов на печать

Блок «процесс» применяется для обозначения действия или последовательности действий, изменяющих значение, форму представления или размещения данных. Для улучшения наглядности схемы несколько отдельных блоков обработки можно объединять в один блок. Представление отдельных операций достаточно свободно.

Блок «решение» используется для обозначения переходов управления по условию. В каждом блоке «решение» должны быть указаны вопрос, условие или сравнение, которые он определяет.

Блок «модификация» используется для организации циклических конструкций. (Слово модификация означает видоизменение, преобразование). Внутри блока записывается параметр цикла, для которого указываются его начальное значение, граничное условие и шаг изменения значения параметра для каждого повторения.

Блок «предопределенный процесс» используется для указания обращений к вспомогательным алгоритмам, существующим автономно в

виде некоторых самостоятельных модулей, и для обращений к библиотечным подпрограммам.

Рисунок. Пример блок - схемы алгоритма нахождения максимального из двух значений.

Псевдокод

Псевдокод представляет собой систему обозначений и правил, предназначенную для единообразной записи алгоритмов. Он занимает промежуточное место между естественным и формальным языками.

С одной стороны, он близок к обычному естественному языку, поэтому алгоритмы могут на нем записываться и читаться как обычный текст. С другой стороны, в псевдокоде используются некоторые формальные конструкции и математическая символика, что приближает запись алгоритма к общепринятой математической записи.

В псевдокоде не приняты строгие синтаксические правила для записи команд, присущие формальным языкам, что облегчает запись алгоритма на стадии его проектирования и дает возможность использовать более широкий набор команд, рассчитанный на абстрактного исполнителя. Однако в псевдокоде обычно имеются некоторые конструкции, присущие формальным языкам, что облегчает переход от записи на псевдокоде к записи алгоритма на формальном языке. В частности, в псевдокоде, так же, как и в формальных языках, есть служебные слова, смысл которых определен раз и навсегда. Они выделяются в печатном тексте жирным шрифтом, а в рукописном тексте подчеркиваются. Единого или формального определения псевдокода не существует, поэтому возможны различные псевдокоды, отличающиеся набором служебных слов и основных (базовых) конструкций.

Примером псевдокода является школьный алгоритмический язык в русской нотации, описанный в учебнике А.Г. Кушниренко и др. «Основы информатики и вычислительной техники».

Пример записи алгоритма на школьном алгоритмическом языке:

```
алг Сумма квадратов (арг цел n, рез цел S)
дано | n > 0
надо | S = 1*1 + 2*2 + 3*3 + ... + n*n
нач цел i
  ввод n; S:=0
  нц для i от 1 до n
    S:=S+i*i
  кц
  вывод "S = ", S
кон
```

Программное представление алгоритма

При записи алгоритма в словесной форме, в виде блок-схемы или на псевдокоде допускается определенный произвол при изображении команд.

Вместе с тем такая запись точна настолько, что позволяет человеку понять суть дела и исполнить алгоритм.

Однако на практике в качестве исполнителей алгоритмов используются специальные автоматы — компьютеры. Поэтому алгоритм, предназначенный для исполнения на компьютере, должен быть записан на «понятном» ему языке. И здесь на первый план выдвигается необходимость точной записи команд, не оставляющей места для произвольного толкования их исполнителем.

Следовательно, язык для записи алгоритмов должен быть формализован. Такой язык принято называть языком программирования, а запись алгоритма на этом языке — программой для компьютера.

Порядок разработки иерархической схемы реализации алгоритмов

К основным методам структурного программирования относится, прежде всего, отказ от бессистемного употребления оператора непосредственного перехода и преимущественное использование других структурированных операторов, методы нисходящего проектирования разработки программы, идеи пошаговой детализации и некоторые другие соглашения, касающиеся дисциплины программирования.

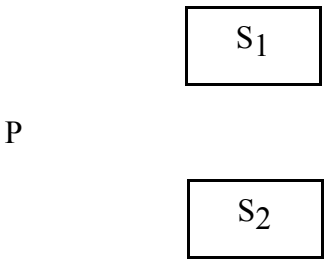
Всякая программа, в соответствии с структурным подходом к программированию, может быть построена только с использованием трех основных типов блоков.

- Функциональный блок, который на блок-схеме изображается в виде прямоугольников с одним входом и одним выходом:

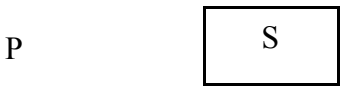
Функциональному блоку в языках программирования соответствуют операторы ввода и вывода или любой оператор присваивания.

В виде функционального блока может быть изображена любая последовательность операторов, выполняющихся один за другим, имеющая один вход и один выход.

- Условная конструкция. Этот блок включает проверку некоторого логического условия (P), в зависимости от которого выполняется либо один (S1), либо другой (S2) операторы:



- Блок обобщенного цикла. Этот блок обеспечивает многократное повторение выполнения оператора S пока выполнено логическое условие P:



При конструировании программы с использованием рассмотренных типов блоков эти блоки образуют линейную цепочку так, что выход одного блока подсоединяется к входу следующего. Таким образом, программа имеет линейную структуру, причем порядок следования блоков соответствует порядку, в котором они выполняются.

Такая структура значительно облегчает чтение и понимание программы, а также упрощает доказательство ее правильности. Так как линейная цепочка блоков может быть сведена к одному блоку, то любая программа может, в конечном итоге, рассматриваться как единый функциональный блок с один входом и одним выходом.

При проектировании и написании программы нужно выполнить обратное преобразование, то есть этот блок разбить на последовательность подблоков, затем каждый подблок разбить на последовательность более мелких блоков до тех пор, пока не будут получены «атомарные» блоки, рассмотренных выше типов. Такой метод конструирования программы принято называть нисходящим («сверху вниз»).

При нисходящем методе конструирования алгоритма и программы первоначально рассматривается вся задача в целом. На каждом последующем этапе задача разбивается на более мелкие подзадачи, каждая подзадача, в конечном итоге на еще более мелкие подзадачи и так до тех пор, пока не будут получены такие подзадачи, которые легко кодируются на выбранном языке программирования. При этом на каждом шаге уточняются все новые и новые детали («пошаговая детализация»).

В процессе нисходящего проектирования сохраняется строгая дисциплина программирования, то есть разбиение на подзадачи осуществляется путем применения только рассмотренных типов конструкций (функциональный блок, условная конструкция, обобщенный цикл), поэтому, в конечном итоге, получается хорошо структурированная программа.

Автоматизация деятельности человека на основе алгоритмизации

Автоматизация сопровождает человеческое общество с момента его зарождения. Она внутренне присуща его развитию. В методологии ее определяют как замещение процессов человеческой деятельности процессами технических устройств. Любопытство заставляло наших предков изучать окружающий мир. Как только они познавали какой-нибудь элемент его, лень толкала их к созданию устройств, которые выполняли бы работу за них. Даже пещерный человек, взяв палку в руки, освободил себя от необходимости залезать на дерево. С каждым новым открытием, человек

снимал с себя какую-нибудь обязанность и перекладывал ее на подручные средства, на животных, потом на машины.

Сегодня любое, предприятие имеет дело с потоками различной информации, которые нуждаются в быстрой и оперативной обработке. Количество информации зависит в основном от размера предприятия и вида деятельности, чем больше предприятие, тем больше объём и уровень сложности обрабатываемой информации. Огромную помощь здесь оказывают современные компьютерные информационные технологии, профессионально разработанная компьютерная информационная система может существенно облегчить жизнь бухгалтерии и руководителям, позволит вести оперативный учёт на предприятии быстро и точно, предоставит широкие возможности анализа, автоматизировав учётные операции, избавит от огромного количества лишней бумаги.

Проектирование информационной системы является, пожалуй, самым важным элементом автоматизации деятельности предприятия. Правильно спроектировать систему означает обеспечить большую часть успеха всего проекта автоматизации. Очень частой ошибкой является внедрение информационной системы при отсутствии какой-либо четко сформулированной системы управления. То есть выражение «создать систему правления» воспринимается как «внедрить нечто компьютерное». Нужно четко осознавать, что система управления первична, а уже создание информационной системы на ее основе, или, попросту говоря, ее реализация в компьютерном виде – вторична.

Многие компании верят в то, что одна только автоматизация приведет к улучшению финансово-экономической ситуации, и начинают усилия по реализации информационных систем непосредственно с автоматизации, пропуская критические шаги понимания и упрощения своих бизнес процессов. Но нередко эти процессы настолько неупорядочены, что в общем

создают впечатление хаоса на предприятии. Как известно, автоматизировать хаос далеко не просто, если невозможно. Поэтому прежде чем создавать информационную систему следует пересмотреть систему управления в организации. Изменение бизнес процессов называют реинжинирингом (business processes reengineering). Так, для начала нужно упорядочить схему бизнес процессов и систему управления организации в целом:

- определиться с организационной штатной структурой,
- разработать механизм финансово-экономического управления компанией (в том числе определить центры ответственности),
- произвести выделение основных технологических потоков (процессов),
- разработать механизмы организационного управления технологическими потоками,
- на основании созданных механизмов управления сформировать технологию финансового анализа и управления деятельностью технологических потоков.

Если будут иметься вышеперечисленные технологии, будет значительно легче разработать информационную систему. Однако, часто приходится упрощать бизнес процессы на предприятии, для того, чтобы было проще описать их на языке компьютеров.

Организация – это набор правил и процедур. Информационная система это тоже набор правил и процедур, поэтому следует понимать какие инструкции и процедуры какими заменить. Не следует также забывать о человеческом факторе при создании информационной системы. Во-первых, именно людям придется работать с системой – одна работать она в любом случае не сможет. Во-вторых, служащие могут улучшить (или упростить) процессы, с которыми они ежедневно встречаются. Автоматизация должна

происходить только после того, как служащие поймут процесс и примут решение о необходимости автоматизации.

После проведения формирования четкой системы управления, начинается непосредственно процесс проектирования информационной системы. Важно, чтобы в проектировании системы участвовали по возможности все сотрудники, которые будут с ней работать. Это позволит определить небольшие особенности и частные потребности в работе каждого отдела организации, поскольку только пользователи будущей системы лучше всего знают, что им нужно.

В проектировании информационной системы также должны участвовать ее разработчики, то есть те, кто будет ее создавать. К выбору разработчика информационной системы нужно подходить очень осторожно. Основными критериями в выборе разработчика являются опыт работы в области создания информационных систем, количество успешно внедренных данной компанией систем на российских предприятиях.

Финансовый менеджер и руководство предприятия должны относиться к автоматизации, как к проекту, то есть определить все стадии, характеристик, временные рамки и бюджет. Основными этапами работы над проектом по автоматизации являются:

- Проведение обследования с целью описания бизнес процессов организации.
- Разработка технического задания на систему автоматизации.
- Разработка технического проекта системы.
- Разработка системы (иногда называемая настройкой).
- Различные стадии и этапы внедрения, опытной и промышленной эксплуатации.

- Выполнение доработок в соответствии с изменившимися потребностями организации.

Результатом проектирования системы является строго формализованное описание, как объекта ее автоматизации, так и ее самой – это и есть алгоритм деятельности предприятия, а значит и деятельности людей, которые на нем трудятся.

Значение алгоритмов при решении повседневных задач

Информатика, как и арифметика, тоже дает явно необходимые знания для выживания человека в современном мире. Например, умение программировать домашнюю бытовую технику: видеомагнитофон (составление списка записываемых телепередач), магнитофон (составление списка записываемых дорожек аудио-CD), сотовый телефон (запоминание номеров, установка параметров, управление роумингом, управление голосовой почтой), часы и таймеры в любом бытовом приборе (выставление и корректировка времени), микроволновые печи, кофеварки, хлебопечки, телевизоры, наконец, компьютеры.

Здесь под программированием понимается составление плана дальнейших действий домашнего прибора - составление алгоритма, запись этого алгоритма на языке прибора (кодирование в соответствии с прилагаемой инструкцией) с последующей загрузкой составленной программы в компьютер (домашнего) прибора.

Пусть надо запрограммировать запись на видеомагнитофоне - на 4 канале с 10.00 утра до 11.25. Это программа в голове у человека кодируется примерно так:

```
ПОКА НЕ 10.00 - НИЧЕГО НЕ ДЕЛАТЬ
УСТАНОВИТЬ КАНАЛ НОМЕР 4
ВКЛЮЧИТЬ ЗАПИСЬ
ПОКА НЕ 11.25 - НИЧЕГО НЕ ДЕЛАТЬ
ВЫКЛЮЧИТЬ ЗАПИСЬ
```

Далее эта программа должна быть перекодирована на язык видеомагнитофона:

```
ВЫБРАТЬ СВОБОДНОЕ МЕСТО
УСТАНОВИТЬ "ДАТА ЗАПИСИ" = СЕГОДНЯ
УСТАНОВИТЬ "НАЧАЛО ЗАПИСИ" = 10:00
УСТАНОВИТЬ "ОКОНЧАНИЕ ЗАПИСИ" = 11:25
УСТАНОВИТЬ "НОМЕР ТЕЛЕКАНАЛА" = 4
```

Загрузка данной программы в видеомагнитофон состоит в нажатии на пульте видеомагнитофона соответствующих кнопок для каждой строки программы.

Компьютер - это такой очень сложный и универсальный домашний прибор. Компьютерная программа является планом дальнейших действий компьютера так же, как программа домашнего прибора является планом дальнейших действий этого прибора. Вывод: программирование компьютеров ничем не отличается от программирования в быту.

Может ли человек, не прошедший никакого курса информатики в школе, разобраться с этим набором современных домашних помощников? Это очень трудный вопрос. На него нельзя ответить однозначно. Известно, что люди старшего поколения сталкиваются с определенными трудностями при проведении даже элементарных действий по программированию современной домашней техники. Конечно, проще всего это объяснить старческим маразмом или отсутствием современной техники в домах «пожилых родителей». Но это не так - программированию можно учить. А когда вокруг все техническое окружение становится программируемым - нужно учить!

Как научить человека узнавать, правильно ли составлена программа для домашнего помощника? Для этого человеку надо представить себя «домашним прибором» с полным набором функций-инструкций и исполнить

(«прокрутить» у себя в голове) составленную программу. А приборов много, каждый имеет свой язык, и приходится постоянно быть выполнителем программ, составленных на разных языках для разных приборов.

Программы из двух-трех шагов можно просто запомнить и считать своими рефлексами: «хочу кушать - жму кнопку два, когда загорится лампочка - можно кушать». Но жить, зазубривая все нужные программы, - не получится. Программируемых приборов так много, инструкции к ним так объемны, требуемые программы так длинны, запоминать команды на языках приборов так лень. Для телевизора, например, нельзя благоприобрести рефлекс: НАЖАТЬ КНОПКУ ОДИН, ДОКРУТИТЬ РУЧКУ ДВА, ПОВТОРИТЬ ВСЕ СНАЧАЛА ДЛЯ КАНАЛОВ 1-32, ЕСЛИ ТЕЛЕКАНАЛЫ УЖЕ НАСТРОЕНЫ, НИЧЕГО НЕ ДЕЛАТЬ. Как минимум в данной инструкции нужно понимать, как менять номера каналов.

Без умения программировать разнообразные устройства человеку сегодня жить трудно, а завтра будет просто невозможно.

Роль информационных технологий сегодня

Недаром нынешнее время многие называют «веком прогресса». Прогресса технического, научного, интеллектуального. Мы оказались под влиянием знаний, порождающих развитие все новых современных технологий, разработку продуктов, создание которых ранее считалось невозможным, интеллектуальное развитие специалистов, воплощающих многочисленные «умные» идеи в жизнь. Спорным вопросом, дебаты по которому разгораются все чаще, однако, по-прежнему оказывается вопрос о роли информационных технологий в развитии данного прогресса. Многие полагают, что Интернет способствует лишь безопасному сохранению интеллектуальной собственности, в то время как другие уверены в том, что функция информационных технологий гораздо более значительная…

За последние 100 лет человечество сделало значительный шаг вперед. Аналитики утверждают, что если бы теми знаниями, которые были использованы в этом столетии специалистами со всего мира в различных областях науки и техники, обладали наши предки хотя бы несколько столетий назад, сегодня на своих авто мы бы уж точно ездили на чистой воде, а не на бензине, и вовсе не по дорогам, а по воздуху. Лучшим объяснением такой идеи послужили бы точные статистические данные, однако в случае подсчета объема использованных знаний точные данные привести, разумеется, невозможно. Что ж, обратимся хотя бы к приблизительным.

В конце девяностых годов XX века значительно выросло число заявок на получение патентов на различные изобретения. В 1997 году, например, в США их насчитывалось лишь 124068, в 1998 – 163147, а в 1999 уже порядка 170000. Таким образом, лишь за два года рост числа заявок вырос на 36%. А следовательно, ровно на столько выросло и число принципиально новых разработок. Сотни тысяч «умных» книг издаются ежегодно, причем из года в год их число все возрастает. Автор одной из них, Роджер Хендрикс, назвал свое произведение «Экономика идей». Именно так, по его мнению, можно охарактеризовать сегодняшнюю экономику.

В чем-то с ним нельзя не согласиться. В наши дни можно работать и зарабатывать деньги многими оригинальными способами, например, продавая патенты и лицензии через Интернет, чем, к слову, занимаются создатели Веб-сайта The Patent & License Exchange (www.pl-x.com), которые, фактически, превратили интеллектуальную собственность в продукт купли-продажи. Впрочем, они не первые и не последние. Многие крупные и мелкие компании и фирмы тесно связали между собой два понятия: «интеллектуальная собственность» и «информационные технологии». Но вот оперируют ими по-разному.

В результате опросов агентств маркетинговых исследований выяснилось, что довольно много образованных людей уверено в том, что информационные технологии вовсе не нужны при решении многих задач, связанных с применением и хранением знаний, преобразованных в «электронный формат». По их утверждениям внедрение информационных технологий обходится дорого и влечет за собой ряд проблем. Именно поэтому многие стараются прибегать к помощи современных разработок только в случае защиты хранящейся информации от несанкционированного доступа к ней. Однако такую категорию людей современные эксперты назвали старомодными. Те же, кто рассуждает более «прогрессивно» осознают, что информационные технологии сегодня значительно облегчают жизнь всем, в частности в вопросах работы с информационными данными. Именно поэтому в настоящее время все более активно развивается довольно молодое направление «разработки программного обеспечения для управления знаниями». Посредством таких систем практически любой сегодня может проконсультироваться в режиме онлайн со специалистом, найти интересующую информацию в Сети, ответить на вопрос, который изводил на протяжении уже долгого времени. Разумеется, с функцией «информационного сторожа» современные технологии справляются также безупречно (или почти безупречно), но понимать их основную роль все-таки следует немного по-иному.

Взять, например, историю Texaco, руководителю группы информационного управления которой, Джону Олду, пришлось нос к носу столкнуться с полезностью информационных технологий в вопросах организации интеллектуальной поддержки работников предприятия.

Пару лет назад многие сотрудники начали жаловаться на то, что иногда им приходилось сталкиваться с подчас неразрешимыми задачами, зачастую решения, которых знали их коллеги. Однако в силу того, что компания огромная, на выяснение всех подробностей зачастую уходило слишком

много времени. Олд задумался над этим. Как-то раз Джон заметил, что сотрудники Texaco очень активно пользуются услугами электронной почты. В ходе переписки они подчас перебрасываются фразами, много значащими для остальных. Однако как сделать так, чтобы полезная информация оказалась доступной всем и при этом имя ее автора осталось в тени? Олд обратился к своему другу, соучредителю Giga Information Group, который вскоре предложил решение проблемы. Основав компанию Tacit Knowledge Systems, он представил на рынок первый продукт – систему KnowledgeMail, которая могла находить в множестве сообщений ключевые слова и фразы, оставляя при этом имя автора сообщения анонимным. И теперь любой работник, у которого возникла проблема с тем же сверлильным аппаратом мог просто ввести в строке запроса ключевую фразу «застряло сверло», а система сама изучала базу данных и находила возможно имеющийся ответ на его вопрос.

Решение похожей проблемы пришло в голову и специалистам по информационным технологиям компании Procter&Gamble, когда ее работники, разбросанные по всему миру, стали жаловаться на затрудненный поиск в огромной Интернет сети предприятия интересующей их информации. Начав использовать программный пакет компании Plumtree Corp., P&G обеспечила всем своим работникам простой доступ к тщательно структурированной внутренней информационной системе, содержащей на самом более миллиона страниц данных.

Однако во многих случаях информационные технологии, как выяснилось, не только обеспечивают доступ к информации, к знаниям, но и являются их мощным источником. Так, специалистами «коммуникационного общества» компании IEEE совместно с программистами MindCrossing была разработана программа «по представлению интеллектуальной собственности в Сети». Под «интеллектуальной собственностью» в данном случае следует понимать многочисленные труды экспертов компании – статьи,

аналитические материалы, обзоры. Десятки тысяч, опубликованные лишь в одном 1999 году, были размещены на Веб-сайте в Интернет. Любой посетитель портала может и сегодня легко (в отличии от все более сложных поисковых двигателей) найти любую, интересующую его статью по каталогу, а также проконсультироваться с экспертом компании. Но если первая услуга бесплатная, совет специалиста может обойтись недешево.

Таким образом, в данном случае компания использовала информационные технологии для размещения материала, который создавался ее сотрудниками. Однако многие пользуются их услугами лишь для претворения уже существующей идеи в жизнь. Наиболее яркий, опять же, пример – Aventis Pharmaceuticals. Производитель лекарственных препаратов на протяжении довольно долгого времени вынашивал идею создания собственного Веб-портала, целью которого стало бы более тесное взаимодействие больных с изготовителем лекарств. Наконец, идея получила свою реализацию. Отныне, клиенты компании могут также проконсультироваться со специалистами, не выходя из дома, получить всю необходимую информацию о том или ином препарате или рекомендацию людей, его применяющих, на Веб-страницах Aventis. Таким образом, компания выработала реальную модель по созданию собственного электронного бизнеса, хотя многие пока недоумевают, как именно Aventis планирует окупить воплощение своей мечты.

Что ж, фактов в защиту «информационно-интеллектуального будущего» предостаточно. Можно не боясь утверждать, что информационные технологии, как это ни «проблематично» и «дорого», все-таки играют большую, чем им отводится многими, роль. Так что, наверное, единственное, что можно порекомендовать «современным старомодникам» - открыть более широко глаза и последовать примеру тех,

кто благодаря высоко-технологичному информационному настоящему добивается действительно блестящих успехов.