Московский государственный технический университет
имени Н.Э. Баумана

А.М. Губарь

Начальный курс информатики

Конспект лекций

Часть 1

Рекомендовано редсоветом МГТУ им. Н.Э. Баумана
в качестве учебного пособия

М о с к в а
Издательство МГТУ им. Н.Э. Баумана
2009

Предисловие

Предлагаемое вниманию читателя учебное пособие написано в соответствии с программой дисциплины «Информатика», которая преподается студентам первого курса на кафедре «Компьютерные системы и сети» МГТУ имени Н.Э. Баумана. Информатика как предмет входит в естественнонаучный цикл дисциплин российской высшей школы и является базовым компонентом федерального государственного образовательного стандарта учебных планов подготовки дипломированных специалистов по направлению «Информатика и вычислительная техника».

В настоящее время преподавание информатики в средней школе проводится с разной степенью детализации изучения отдельных разделов. Наряду с различной квалификацией учителей и разнородным оснащением кабинетов информатики это приводит к тому, что первокурсники технических вузов по-разному подготовлены к обучению в таком вузе, хотя практически все они имеют дома персональные компьютеры. Разумеется, при написании учебного пособия было учтено данное обстоятельство, поэтому каждая новая тема излагается по принципу «от простого к сложному», с подробным объяснением основных понятий и многочисленными примерами. Вместе с тем автор надеется, что эта работа не покажется искушенному читателю слишком простой, и он найдет в ней довольно много интересной для себя информации.

Во введении анализируется предмет информатики и определяется ее место в ряду других научных дисциплин.

Первая глава посвящена рассмотрению основного понятия информатики, а именно информации. В ней конкретизируются ее свойства, изучаются различные подходы к измерению количества информации. Здесь же вводится

понятие энтропии и исследуется ее связь с информацией. Наконец, устанавливается различие между терминами «информация» и «данные», которые в обиходе мы привыкли считать синонимами, и рассматриваются различные типы данных.

Во второй главе изучаются системы счисления, их взаимосвязь и способы перевода чисел из одной системы счисления в другую. Рассматриваются машинные коды чисел, применяемые для представления последних и реализации арифметических действий в компьютере. Приводятся способы размещения чисел в разрядной сетке компьютера, а также основные методы выполнения арифметических операций.

В третьей главе изучены основные понятия алгебры логики. В качестве примеров использования этого математического аппарата рассмотрены логические элементы, реализующие булевы функции, а также приведены функциональные схемы некоторых блоков компьютера.

В конце каждой главы предлагаются контрольные вопросы, правильные и уверенные ответы на которые позволят читателю убедиться в том, что он твердо усвоил основное содержание соответствующего раздела. Автор также надеется, что этому будет способствовать и словарь основных терминов, приведенный в конце учебника.

Содержание остальных глав будет отражено в следующих частях учебного пособия. В них читатель познакомится с основами алгебры логики и теории автоматов – математического аппарата, с помощью которого в формализованном виде описывается функционирование основных блоков и компьютера в целом. Будут рассмотрены вопросы представления и обработки информации с целью формирования у читателя общего понятия о функционировании компьютера, способы и устройства машинного хранения информации, а также системы, функционирование которых основано на использовании больших информационных хранилищ. Особое внимание будет

уделено проблемам передачи информации, а также информационным сетям, их типам и функционированию вычислительных сетей.

Введение

Термин «информатика» (informatique) возник во Франции в конце 60-х годов XX века путем слияния двух слов: информация (information) и автоматика (automatique) и подразумевает компьютерную обработку информации. В США и в англоязычных странах для обозначения области информационной деятельности с помощью человеко-машинных систем переработки информации несколько ранее был принят термин «вычислительная наука» (computer science). В нашей стране под информатикой первоначально понималась лишь «научная дисциплина, изучающая структуру и общие свойства научной информации, а также закономерности всех процессов научной коммуникации – от неформальных процессов обмена научной информацией при непосредственном устном и письменном общении ученых и специалистов до формальных процессов обмена посредством научной литературы». (Словарь по кибернетике, 1979 год).

В некотором смысле предшественницей информатики можно считать кибернетику – науку об управлении, получении, преобразовании и передаче информации в кибернетических системах, под которыми понимаются системы любой природы: административные, биологические, социальные, технические и др. Можно точно указать время появления нового научного направления в современном понимании – в 1948 году вышла сразу же ставшая научным бестселлером книга американского математика Норберта Винера «Кибернетика, или Управление и связь в животном и машине». В ней речь идет о

возможности создания общей теории управления, а проблемы управления и связи для различных систем рассматриваются с единых позиций.

Кибернетика – слово греческого происхождения и может быть переведено как «искусство управления». Однако этот термин в научном смысле впервые использовал еще в первой половине XIX века французский физик Ампер, разрабатывая единую систему классификации всех наук. Он обозначил так тогда еще не существовавшую гипотетическую науку управления людьми и обществом, которая, по его мнению, обязательно должна была появиться.

Необходимо отметить, что развитие кибернетики в нашей стране искусственно тормозилось почти все 50-е годы XX века. Например, первое издание упомянутой книги Винера на русском языке появилось только в 1958 году, а в философском словаре 1959 года издания кибернетика все еще определялась как «буржуазная лженаука». Это замедлило развитие вычислительной техники в СССР, хотя именно в те же годы у нас были реализованы передовые по тем временам проекты создания вычислительных машин под руководством С.А. Лебедева.

Возникновение кибернетики совпало по времени с построением электронных цифровых вычислительных машин первого поколения, благодаря которым стало возможным решение очень сложных вычислительных задач. Универсальность компьютерных вычислений позволяла надеяться на открытие универсальных схем управления, но этого в полной мере не произошло. Тем не менее, полученные при кибернетическом подходе знания о разнообразных системах управления, общие принципы их функционирования, которые частично удалось при этом выявить, оказались весьма продуктивными. Идеи кибернетики оказались плодотворными для биологии, химии и многих других наук.

Во многом благодаря кибернетике возникла структурная лингвистика с разделением последней на математическую и прикладную лингвистику.

Следует выделить такое направление как техническая кибернетика, в состав которой входит теория автоматического управления – теоретический фундамент автоматики. Исследовательская и практическая работа в этом направлении позволила получить важнейшие результаты, без которых в современном обществе был бы невозможен технический прогресс. Сегодня кибернетику можно рассматривать как прикладную информатику при создании различных автоматических и автоматизированных систем управления, от управления автономным объектом до мощных систем управления отраслями промышленности, коллективами людей и т.д. Таким образом, источниками современной информатики, прежде всего, являются документалистика, изучающая и оптимизирующая документы и документальные системы, и кибернетика.

Становление информатики совпало по времени с бурным развитием вычислительной техники, с появлением все более мощных и совершенных электронных вычислительных машин, а затем и персональных компьютеров. Современный компьютер – мощный инструмент обработки разнородной информации, а информация, в свою очередь, – главный объект изучения информатики. Отсюда понятно положительное влияние постоянного быстрого совершенствования вычислительных средств на темпы развития современной информатики и на ее содержание. С другой стороны, достижения информатики благотворно воздействуют на прогресс в области вычислительной техники.

Информатику в самом общем виде можно определить как науку о способах обработки информации с помощью компьютеров для использования последней в различных сферах человеческой деятельности. Под обработкой информации понимается ее сбор,

хранение, поиск, преобразование, передача и выдача. К настоящему времени в информатике выделились следующие составные части.

Теоретическая информатика с использованием математических методов изучает структуру и общие свойства информации и протекания информационных процессов. Она включает в себя такие дисциплины как математическая логика, вычислительные методы, моделирование, теория автоматов, теория алгоритмов, теория информации и ее кодирования и передачи, теория формальных грамматик и языков, исследование операций, искусственный интеллект. Последний из указанных разделов занимается компьютерной лингвистикой, машинным переводом, распознаванием образов, моделированием рассуждений, созданием экспертных систем и находится на стыке с психологией, физиологией, лингвистикой и другими науками.

Технические и программные средства информатизации позволяют воплотить теоретические достижения информатики на прикладном уровне. К ним относятся вычислительные устройства, вычислительные системы, а также системы обработки и передачи данных. В программном обеспечении выделяют системные, сетевые, универсальные и профессионально ориентированные средства с пакетами прикладных программ.

Информационные технологии и системы в рамках рассматриваемой классификации являются достаточно универсальными. Они занимаются решением вопросов анализа и оптимизации информационных потоков в различных системах, реализацией принципов структурирования, хранения и поиска информации. К ним относятся информационно-справочные, информационно-поисковые системы, а также глобальные системы хранения и поиска информации, включая интернет.

Наконец, следует назвать социальную информатику, которая сравнительно недавно стала выделяться в отдельный раздел

информатики. Она занимается изучением информационных ресурсов как факторов социально-экономического и культурного развития современного информационного общества.

Из рассмотрения содержания информатики становится ясно, что она представляет собой очень широкую область научных знаний и расположена на пересечении нескольких фундаментальных и прикладных дисциплин. Она связана:

- с математикой – через математическую логику, дискретную математику, теорию алгоритмов, математическое моделирование;
- с физикой, химией, биологией, электроникой, радиотехникой – через разработку аппаратных средств информатизации;
- с кибернетикой – через теорию информации и теорию управления;
- с лингвистикой – через теорию формальных языков и знаковых систем;
- с философией и психологией – через теорию познания.

Важная роль информатики заключается в том, что она, по существу, является научным фундаментом процесса информатизации современного общества. Потребность в ней как образовательной дисциплине заключается в социальном заказе на подготовку специалистов с новым мировоззрением. Оно основано на понимании роли информации, знании новейших и перспективных информационных и вычислительных технологий, систем и сетей.

В предлагаемом курсе будут рассмотрены основные темы, характеризующие содержание информатики.

1. Основные понятия теории информации

1.1. Понятие информации, ее свойства

Со словом ИНФОРМАЦИЯ каждый из нас встречается очень часто. Человек живет среди себе подобных, окружающий мир является для нас постоянным источником разных сведений, которые мы получаем при общении с другими людьми, с животными, от различных приборов, предметов, из книг и газет, наблюдая происходящие явления и процессы и т.д. При этом восприятие осуществляется с помощью пяти известных органов чувств: зрение, слух, вкус, обоняние, осязание; главными в данном процессе являются глаза – свыше 80% информации поступает человеку через них.

Что же такое информация? Ведь есть люди, которых мы никогда не встретим, страны, в которых никогда не побываем, книги, которые никогда не будут прочитаны нами. А это все – потенциальные источники информации. Следовательно, информация существует не сама по себе, а становится для нас таковой только после того, как мы получим ее.

Термин информация происходит от латинского слова informatio, что означает осведомление, сообщение, сведения о ком или о чем-либо. Можно сказать, что *информация – это сведения или знания, которыми живые существа или приборы обмениваются в процессе своего функционирования*. При этом необходимо учитывать важный аспект, связанный с получением информации: воспринимая ее, мы тем самым узнаем что-то новое о конкретной предметной области, другими словами, уменьшаем степень неполноты наших знаний.

Информация принадлежит к исходным, неопределяемым понятиям науки. Точно так же, например, в планиметрии не определяются такие

базовые понятия как точка, прямая и плоскость. Являясь отражением процессов реального мира, сущность информации раскрывается в связи с действиями, в которых она принимает самое непосредственное участие: передача, прием, хранение, преобразование, выдача. В рамках нашего рассмотрения мы остановимся на такой формулировке:

Информация – получаемые сведения об объектах и явлениях, которые уменьшают степень неполноты знаний о них.

При таком подходе ясно, что очень важными являются способы получения и обработки разнородной информации, которая постоянно сопутствует нашей жизнедеятельности. Однако существует ряд свойств, которые присущи *любой* информации, независимо от способов ее получения и представления, то есть являются *универсальными*. Что же это за свойства, делающие информацию таковой?

Предположим, вы читаете предложение: «Дважды два – четыре». Вряд ли для кого-то из вас здесь содержится информация, потому, что этот факт вы и так знаете уже давно. Следовательно, информация должна быть *новой*.

А теперь другое высказывание: «Дважды два – пять». И здесь нет никакой информации, так как это не соответствует действительности. Следовательно, информация должна быть *достоверной*.

Если бы сейчас читатель-первокурсник стал детально знакомиться с правилами начисления пенсий, то почти ничего не запомнил бы – эти сведения для него совершенно неинтересны, так как *сейчас* и в обозримом будущем они вряд ли ему понадобятся. Следовательно, информация должна быть *своевременной*.

Предложение: «Ich wurde in Potsdam geboren» для некоторых из вас также не содержит никакой информации, поскольку, не зная немецкого языка, вы не поймете, о чем в нем идет речь. Следовательно, информация должна быть *понятной*.

Пятое свойство информации является идеальным, то есть таким, к которому нужно стремиться, но которого достичь нельзя. Действительно, вряд ли кто-то из нас может узнать все обо всем или даже абсолютно все о чем-то конкретном, но любознательный человек всегда старается узнать как можно больше об интересующем его предмете или явлении. Следовательно, информация должна быть *всеобъемлющей* или *полной*.

Итак, на «бытовом» уровне мы выделили пять свойств информации, а именно: она должна быть новой, достоверной, своевременной, понятной и всеобъемлющей. Вообще-то этих свойств больше, приведем их перечень, вовсе не претендующий на полноту.

Адекватность – определенный уровень соответствия образа, создаваемого на основе полученной информации, реальному объекту или явлению.

Актуальность – степень сохранения полезности информации к моменту ее использования.

Достаточность – свойство полученной информации содержать минимальный, но достаточный для ее использования набор показателей.

Достоверность – свойство информации отражать с заданной точностью реальные объекты или явления.

Доступность – свойство информации соответствовать уровню ее восприятия пользователем.

Репрезентативность – свойство информации, связанное с правильностью ее отбора для всестороннего отражения свойств объекта или явления.

Своевременность – свойство информации поступать к моменту ее использования.

Содержательность – свойство информации, которое можно определить как отношение ее количества в сообщении к объему обрабатываемых при этом данных.

Точность – степень близости получаемой информации к реальному состоянию объекта или явления.

Устойчивость – свойство информации реагировать на изменения исходных данных без уменьшения требуемой точности.

Перечисленные свойства информации требуют некоторых уточнений. Во-первых, их набор характеризует *качество информации* – совокупность ее потребительских показателей, определяющих возможность эффективного использования информации. Во-вторых, следует различать такие свойства как адекватность, достоверность, репрезентативность, точность и устойчивость, актуальность и своевременность, поскольку перечисленные характеристики определяются как на этапе проектирования, так и при функционировании информационных систем, то есть на разных стадиях использования и обработки информации. В-третьих, присутствует некоторая терминологическая неоднозначность, связанная с неисчерпаемостью самого понятия информации, например, вместо доступности точнее выглядит термин понятность информации, так как в первом случае речь может идти о невозможности по каким-либо причинам получения требуемой информации. Наконец, в-четвертых, предложенный список свойств, как отмечалось, не является полным; так можно справедливо утверждать, что информация, например, должна быть полезной, интересной, ведь если она кажется человеку таковой, то он легче и прочнее запоминает ее.

1.2. Измерение информации

Мы привыкли к единицам измерения различных величин, с которыми нам часто приходится сталкиваться: километр, грамм, час, рубль, доллар и т.д. А как и в каких единицах измерять знания? Ведь информация – это знания, которыми…

Предположим, перед читателем две книги, в которых, очевидно, содержатся определенные знания. Пусть в первой книге триста страниц, а во второй – сто. Значит ли это, что в более толстой книге для вас в три раза больше различных сведений, чем во второй? Вряд ли. Например, первая книга – это сборник русских народных сказок, многие из которых знакомы вам с детства, а вторая книга – самоучитель игры на гитаре, которую вы хотите освоить. Ясно, что в этом случае во второй книге для вас содержится гораздо больше информации, то есть при измерении последней нельзя действовать прямолинейно, а надо учитывать различные параметры, например, содержание.

Информация передается в пространстве и времени от источника к получателю в виде *сообщения*, под которым можно понимать некоторую форму ее представления с помощью определенных знаков. Такое сообщение может быть выражено средствами естественных или искусственных языков. Первые являются языками общения, возникшими естественным путем и представляющими собой совокупность алфавита, лексики, грамматики, а также фонетики. Вторые являются специально созданными *семиотическими* системами (*семиотика* – наука о свойствах знаков и знаковых систем) и выступают в роли специализированных знаковых систем для записи информации.

С точки зрения семиотики информационное сообщение рассматривается на трех уровнях. На синтаксическом уровне исследуются внутренние свойства сообщений, а именно отношения, которые сложились между знаками и отражают структуру существующей знаковой системы. (*Синтактика* – раздел семиотики, изучающий синтаксис знаковых систем). Внешние свойства изучаются на семантическом и прагматическом уровнях. В первом случае анализируются отношения между знаками и обозначаемыми ими понятиями – предметами, действиями, качествами и т.д. Другими словами, в роли объекта изучения на данном этапе выступает *смысловое*

содержание информационного сообщения, а также его связь с источником информации. (*Семантика* – раздел семиотики, изучающий интерпретацию высказываний знаковых систем). Во втором случае анализируется *потребительское* содержание сообщения, то есть его связь с получателем информации. (*Прагматика* – раздел семиотики, изучающий восприятие осмысленных выражений знаковых систем как средств общения между источником и потребителем информации).

В соответствии с этим формируются и три направления решения проблем представления и передачи информации, а также измерения ее количества. Необходимо отметить, что современная *теория информации* занимается в основном проблемами синтаксического уровня, абстрагируясь от смыслового содержания. При этом центральным понятием является «количество информации», под которым понимается мера частоты использования знаков для формирования сообщений. Мы также сосредоточимся именно на данном направлении, тем более что оно гораздо легче поддается формализации. Однако сначала приведем классификацию методов измерения информации (рисунок 1.1) и дадим обзор двух других уровней рассмотрения внешних свойств информационных сообщений с точки зрения измерения количества информации, содержащейся в них.

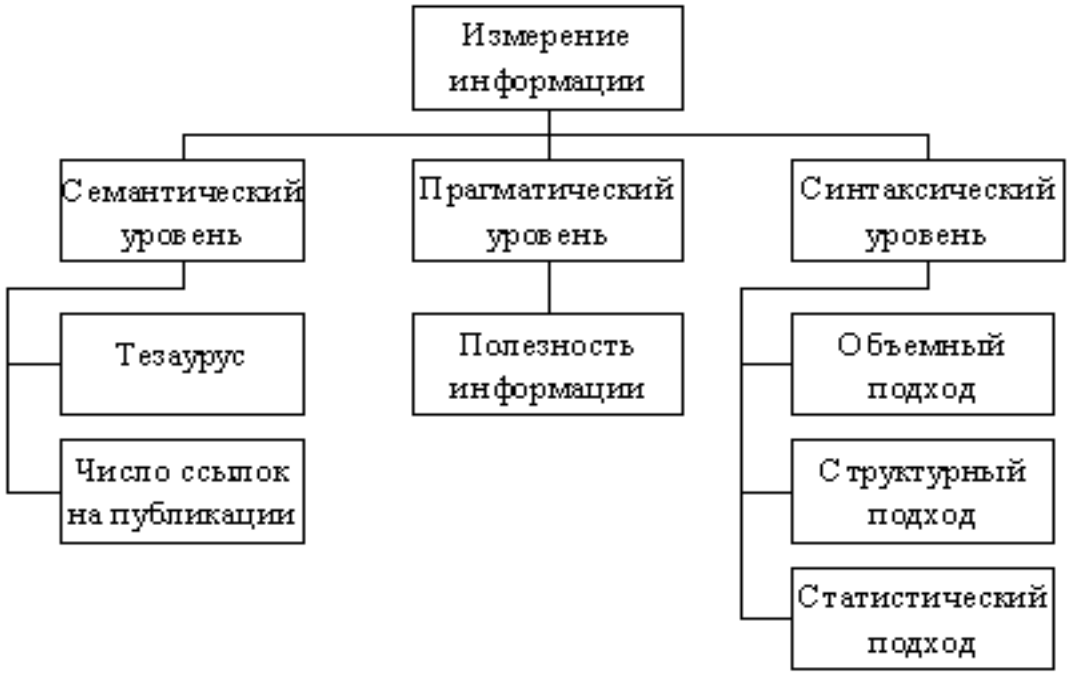


Рис. 1.1. Методы измерения информации

Центральным понятием *семантического* уровня является *тезаурус*, под которым в широком смысле понимается совокупность накопленных человеком знаний, а также сведений, которыми располагает пользователь или система. В этом случае от объема тезауруса пользователя зависит количество воспринимаемой им семантической информации. Характер этой зависимости можно интерпретировать следующим образом. В двух крайних случаях, когда получатель демонстрирует полное незнание предмета или когда он «все знает» в данной предметной области, количество информации, которую потребитель извлекает из поступающих сообщений и затем включает в свой тезаурус, близко к нулю. Ведь в первом случае он не понимает смысла того, что получил, а во втором – полученное сообщение не представляет для него никакого интереса. Ясно, что между этими двумя предельными значениями объема тезауруса пользователя существует некоторая оптимальная величина, при которой смысловое содержание получаемой информации будет практически полностью соотноситься с объемом тезауруса, и тогда

количество воспринимаемой информации будет максимально. Следовательно, до достижения этого «наилучшего» значения количество полезной информации будет расти, а затем уменьшаться. Кстати, одно и то же информационное сообщение является содержательным для подготовленного потребителя и в то же время абсолютно бессмысленно для некомпетентного пользователя. Можно констатировать, что в рассмотренном случае в качестве относительной меры количества семантической информации I_c, воспринимаемой получателем, удобно использовать коэффициент содержательности или информативности C, который есть отношение I_c к ее объему V:

$$C = I_c/V.$$

Иное направление оценки количества информации на семантическом уровне разрабатывается в рамках науковедения и состоит в том, что основным показателем ценности семантической информации, содержащейся в сообщении, которым в данном случае является опубликованная работа, служит количество ссылок на него в других публикациях. Соответствующие значения определяются методами математической статистики.

Прагматическая мера информации определяет ее *полезность* в том смысле, насколько ее получение приближает пользователя к достижению поставленной перед ним цели. За эту меру можно принять количество информации, которое необходимо для реализации целевой функции. Полезность информации также может оказаться совершенно иной для разных групп потребителей, к тому же на нее оказывают существенное влияние внешние обстоятельства, например, своевременность ее получения. К скорости доставки информации предъявляются высокие требования, потому что принятие решений и выработка на их основе управляющих воздействий осуществляются в действующих системах в реальном масштабе времени, со скоростью изменения параметров управляемых объектов. Таким образом, рассматриваемая прагматическая мера непосредственно связана с

практическим использованием полученной информации. Следует отметить, что полезность может иметь и отрицательное значение, когда полученная информация отдаляет ее получателя от желаемой цели, – тогда такая информация является *дезинформацией*.

Теперь перейдем к проблемам, связанным с измерением количества информации, которые разрешаются на *синтаксическом* уровне.

Рассмотрим такую ситуацию: вы подбрасываете монету и смотрите, как она упала на стол – «орлом» или «решкой» вверх (аверсом или реверсом, как говорят нумизматы). Заведомо маловероятные случаи мы отбрасываем, то есть считаем, что монета не может встать на ребро на поверхности гладкого стола, быть проглоченной бросающим и т. д.

Итак, возможен только один из двух результатов такого бросания. Если монета упала «орлом», обозначим такое событие за единицу, если она упала «решкой» – за ноль. Подбросив монету и посмотрев, как она упала, вы получаете определенную порцию информации (другое дело – насколько эта информация вам нужна). Учитывая наши обозначения, вы узнаете, 1 или 0 выпали в результате вашего действия. Подбросив монету еще раз и опять посмотрев, как она упала, вы снова получаете определенное количество информации (такое же, как и в первом случае независимо от того, как теперь упала монета – «орлом» или «решкой»).

В результате ваших действий вы каждый раз будете узнавать о появлении 0 или 1. Забегая вперед или вспомнив о двоичной системе счисления, можно сказать, что вы узнаете о последовательном появлении двоичной цифры в разрядах двоичного числа. Например, подбросив монету пять раз и столько же раз получив информацию о результатах бросания, вы можете сформировать пятиразрядное двоичное число.

Получаемое вами каждый раз при этом количество информации минимально, оно носит название 1 *бит*. Другими словами, *один бит – это*

минимальное количество информации, содержащееся в одном двоичном разряде.

Этот термин образован следующим образом. В переводе с английского языка binary digit означает двоичная цифра или двоичная единица. Взяв две первые буквы первого слова и последнюю букву второго слова (или сжав слова), получили новообразование bit.

Рассмотрим еще один пример. Предположим, кто-то задумал одно из следующих чисел:

0, 1, 2, 3, 4, 5, 6, 7,

а другой пытается узнать задуманное число, формулируя вопросы так, что первый будет отвечать на них только «да» или «нет». Пусть отгадывающий решил ограничиться простым методом *перебора*. Тогда в лучшем для себя случае он может сразу угадать требуемое число, получив, например, ответ «да» на вопрос: «Это – 0?». Однако если он снова последовательно пойдет с «левого конца», а задумано число 7, то в этом наихудшем для себя случае ему для достижения результата придется задать семь вопросов.

Алгоритм поиска можно усложнить, за счет чего требуемое количество задаваемых вопросов всегда будет существенно меньше семи, хотя и больше одного. Для этого надо формулировать их таким образом, чтобы после каждого ответа, причем «да» или «нет» – не имеет значения, пространство поиска сокращалось бы вдвое, то есть применить принцип *дихотомического деления*.

Один из возможных вариантов формулирования вопросов при реализации указанного подхода выглядит так. Пусть задумано число 6. Первый вопрос: «Это число больше трех?», ответ: «Да». Второй вопрос: «Это число больше пяти?», ответ: «Да». Третий вопрос: «Это число больше шести?», ответ: «Нет». Следовательно, задумано число 6, чтобы узнать это, потребовалось задать три вопроса.

Если ответы «да» кодировать единицей, а ответы «нет» – нулем, то в процессе нахождения требуемого числа получается следующая последовательность цифр: 110, а это – десятичное число 6, записанное в двоичной системе счисления. Таким образом, отвечая на поставленные вопросы, «хранитель информации» последовательно выдает трехразрядный двоичный код задуманного им десятичного числа. Ясно, что в этом случае для узнавания числа необходимо обладать информацией объемом в 3 бита, которая поступает тремя порциями по одному биту.

Попробуйте ответить на два следующих вопроса:

1. Если исходных чисел не 8, а 16 (от нуля до пятнадцати включительно), то сколько потребуется задать вопросов, чтобы узнать задуманное число, действуя в соответствии с рассмотренным алгоритмом?
2. Если в распоряжении отгадывающего 5 вопросов, то одно из скольких задуманных чисел он может узнать?

Аналогично тому, как в метрической системе мер мы используем различные единицы измерения, кратные степеням десятки: километр, метр, дециметр, тонна, центнер, килограмм, грамм и т.д., в информатике кроме бита нашли применение и другие единицы измерения информации. Только здесь в качестве соответствующего коэффициента используется степень числа 2, ближайшая к тысяче, то есть $2^{10} = 1024$, поскольку мы уже связали бит с двоичной системой счисления. Так

$$1 \text{ килобит} = 2^{10} \text{ бит}, \quad 1 \text{ мегабит} = 2^{20} \text{ бит},$$

$$1 \text{ гигабит} = 2^{30} \text{ бит}, \quad 1 \text{ терабит} = 2^{40} \text{ бит}.$$

Для тех же целей применяется и *байт* (от английского byte) – *единица измерения информации, равная восьми битам*. Соответственно используются также 1 килобайт = 2^{10} байт, 1 мегабайт = 2^{20} байт, 1 гигабайт = 2^{30} байт, 1 терабайт = 2^{40} байт.

При измерении информации в указанных единицах часто говорят об *объемном* подходе к определению количества информации. Он очень удобен, так как для определения, например, объема информации, записанной в двоичном представлении в памяти компьютера или на внешнем носителе, достаточно подсчитать количество требуемых для такой записи двоичных символов, причем всегда будем получать целое число. Однако существуют и другие методы измерения количества такого многогранного понятия как информация, к изучению которых мы и приступим.

1.3. Количество информации

Собственно говоря, рассматривая ситуацию с угадыванием числа, мы уже затронули вопрос о подсчете количества информации. Однако следует отметить, что специфика примера позволила нам формально подойти к решению этой задачи. Ведь ясно, что в реальных условиях поступления информации всегда надо учитывать *вероятностный* характер этого процесса: те или иные сведения могут быть получены или нет в силу целого ряда различных причин. Поэтому в целях дальнейшего изложения сначала придется кратко коснуться основных положений теории вероятностей, вернувшись к ситуации с подбрасыванием монеты.

Теория вероятностей – это математическая дисциплина, изучающая закономерности случайных процессов. Ее центральным понятием является *вероятность* – числовая характеристика степени возможности наступления при определенных условиях случайного события, то есть *отношение числа наступивших событий к общему числу всех возможных событий.* Такие события интерпретируются как результаты опыта, воспроизводимого многократно. Пусть в нашем примере было произведено N подбрасываний

монеты, из которых она n_1 раз упала «орлом» вверх и n_2 – «решкой», причем все эти испытания являются независимыми. Под независимостью испытаний в данном случае понимается тот факт, что необходимо разнообразить условия проведения опыта: подбрасывать монету с разной силой, с разной высоты и т.д., то есть обеспечить случайный характер протекания данного процесса. Тогда $p_1 = n_1/N$ – вероятность выпадения «орла» и $p_2 = n_2/N$ – вероятность выпадения «решки». Важно отметить, что поскольку $n_1 + n_2 = N$, то

$$p_1 + p_2 = 1.$$

Из определения вероятности также следует, что она всегда неотрицательна и не больше 1.

В двух крайних случаях, когда $p = 0$ и $p = 1$, соответствующих ситуациям, когда случайное событие никогда не происходит или, наоборот, всегда имеет место, неопределенность ситуации отсутствует. Очевидно, что мы сталкиваемся с наибольшей неопределенностью, когда в нашем примере оба из двух возможных результатов опыта равновероятны, то есть $p_1 = p_2 = 0{,}5$.

Рассмотрим еще два примера. Вы подбрасываете игральную кость – кубик, на шести гранях которого «закодированы» числа от одного до шести. Пусть событие состоит в том, что выпало четное число. Очевидно, что вероятность этого события $p = 0{,}5$, так как четных и нечетных чисел – равное количество – по три.

Наконец, снова пример с подбрасыванием монеты. Какова вероятность того, что при трехкратном ее подбрасывании один раз выпадет «орел»? Всего возможны восемь исходов такого бросания. С учетом введенных обозначений они будут выглядеть так: 000, 001, 010, 011, 100, 101, 110 и 111. Следовательно, три исхода 001, 010 и 100 соответствуют однократному выпадению «орла». Теперь становится понятным, что искомая вероятность $p = 3/8$.

Итак, вероятность события, наступившего в результате некоторого эксперимента, есть отношение числа исходов этого эксперимента, благоприятствующих наступлению данного события, к числу всех возможных исходов проводимого эксперимента.

Теперь можно перейти к рассмотрению понятия количества информации, под которым понимается мера величины информации, содержащейся в одной случайной величине относительно другой. Это понятие тесно связано с понятием *энтропии – количественной меры неопределенности ситуации*. Ведь уже отмечалось, что получение информации уменьшает степень неполноты наших знаний. Насколько бóльший объем знаний мы получаем в процессе познания, настолько уменьшается наше незнание в конкретной предметной области. Другими словами, в ходе этого процесса уменьшается неопределенность ситуации, поэтому на практике необходимо уметь численно оценивать степень неопределенности разнообразных опытов, чтобы можно было сравнивать их по этому параметру.

Сначала рассмотрим эксперимент, имеющий *k равновероятных* исходов. Ясно, что степень неопределенности такого опыта определяется именно числом k: при $k = 1$ единственный исход не является случайным, то есть неопределенность ситуации отсутствует или равна нулю, а с ростом k количество разных исходов также увеличивается и предсказать результат эксперимента весьма сложно, так как неопределенность ситуации возрастает. Следовательно, искомая числовая характеристика степени неопределенности представляется в виде функции $f(k)$ числа k, причем она обладает следующими свойствами: при $k = 1$ $f(k) = 0$, а с ростом k функция также возрастает.

Теперь рассмотрим ситуацию с двумя *независимыми* опытами α и β, независимыми в том смысле, что любые сведения об исходе первого из них никак не влияют на вероятности исходов второго опыта. Пусть они имеют k и l равновероятных исходов соответственно. Проанализируем сложный

эксперимент $\alpha\beta$, который заключается в одновременном выполнении опытов α и β. Неопределенность исхода такого сложного опыта будет больше, чем неопределенность опыта α, ведь к последней здесь необходимо еще добавить неопределенность исхода опыта β. Естественно предположить, что степень неопределенности рассматриваемого сложного эксперимента $\alpha\beta$ равна сумме неопределенностей опытов α и β. Эксперимент $\alpha\beta$ имеет kl равновероятных исходов, поскольку каждый из k возможных исходов опыта α комбинируется с l исходами опыта β. Таким образом, в данном случае искомая функция должна удовлетворять следующему условию:

$$f(kl) = f(k) + f(l).$$

Это выражение позволяет принять за меру неопределенности опыта, имеющего k равновероятных исходов, величину $\log_a k$, так как хорошо известно, что $\log_a(kl) = \log_a k + \log_a l$. Так же известно, что логарифмическая функция является единственной непрерывной функцией аргумента k, которая удовлетворяет полученному условию и для которой $f(1) = 0$ и $f(k) > f(l)$ при $k > l$ ($a > 1$). Отметим также, что выбор основания системы логарифмов в данном случае не является существенным, потому что в соответствии с формулой

$$\log_b k = \log_b a \cdot \log_a k$$

переход от одной системы логарифмов к другой сводится лишь к умножению найденной функции $f(k) = \log_a k$ на постоянный множитель $\log_b a$, называемый модулем перехода. Другими словами, этот процесс равносилен простому изменению единицы измерения степени неопределенности.

Полученная общая неопределенность эксперимента, имеющего k равновероятных исходов, равна $\log_a k$, поэтому можно считать, что каждый отдельный исход, вероятность которого равна $1/k$, вносит неопределенность

$$(1/k) \cdot \log_a k = -(1/k) \cdot \log_a(1/k).$$

Пусть теперь $k = 3$, а соответствующие вероятности трех исходов равны 1/2, 1/3 и 1/6. Тогда можно считать, что в результат такого опыта эти исходы вносят неопределенность, равную соответственно $-(1/2) \cdot \log_a(1/2)$, $-(1/3) \cdot \log_a(1/3)$ и $-(1/6) \cdot \log_a(1/6)$, а общая неопределенность этого опыта будет равна

$$-(1/2) \cdot \log_a(1/2) - (1/3) \cdot \log_a(1/3) - (1/6) \cdot \log_a(1/6).$$

Аналогично этому можно положить, что в общем случае для эксперимента с вероятностями его исходов p_1, p_2, …, p_k мера неопределенности $H(\alpha)$, называемая энтропией эксперимента α, равна:

$$H(\alpha) = -p_1 \cdot \log_a p_1 - p_2 \cdot \log_a p_2 - \ldots - p_k \cdot \log_a p_k.$$

Приведенные рассуждения позволяют перейти к рассмотрению двух подходов к измерению количества информации, широко используемых на синтаксическом уровне в информатике и различных технических приложениях.

Структурный метод не учитывает содержания сообщения и связан с подсчетом числа символов в нем, то есть с его длиной. Если m – количество символов, принятых для представления информации (или основание *системы счисления*), а n – количество позиций, необходимых и достаточных для представления чисел заданной величины, то $N = m^n$ – количество чисел, которые можно представить при наличии данных параметров. Введя логарифмическую меру, количество информации $I(n)$ можно вычислить так:

$$I(n) = n\log_a m.$$

Это – формула Р. Хартли, которая получена при условии равной вероятности всех возможных событий. Если в качестве основания логарифма принять $a = m$, то $I(n) = n$. В этом случае при $m = 10$ единицей измерения информации является *дит*, а для двоичной системы имеем $m = a = 2$, при этом искомая величина количества информации эквивалентна количеству в сообщении двоичных символов 0 или 1 и измеряется в битах. Следовательно,

1 бит информации соответствует одному элементарному событию, которое может произойти или нет. Другими словами, за единицу измерения степени неопределенности в этом случае (при а = 2) принимается неопределенность, содержащаяся в эксперименте, имеющем два равновероятных исхода, как это было в опыте с подбрасыванием монеты. Кстати отметим, что один дит примерно в 3,3 раза больше одного бита, так как в данном случае упомянутый модуль перехода равен $\log_2 10 \approx 3{,}32$.

Итак, количество информации в двоичном сообщении длиной n как раз равно n битам. Используя формулу $n = \log_2 2^n = n\log_2 2$, мы можем определить количество информации в сообщении длиной n из m произвольных символов (число m-итов): $I(n) = n\log_m m$. Таким образом, формула Хартли является обобщением упомянутого факта, когда за основание логарифма берется произвольное a.

Следует отметить, во-первых, что полученная мера количества информации удобна тем, что ею можно оперировать как числом. Во-вторых, при наличии нескольких источников общее количество информации, получаемой от них, равно сумме количеств информации от каждого источника:

$$I(n_1, n_2, \ldots, n_n) = I(n_1) + I(n_2) + \ldots + I(n_n).$$

Тогда становится понятным другое название меры Хартли – *аддитивная* мера, поскольку слово addition с английского переводится как суммирование.

Статистический подход, принятый в теории информации, учитывает содержание информационного сообщения и основан на том, что представления получателя информации о наступлении того или иного события недостоверны и выражаются вероятностями, с которыми он их ожидает. Эта мера неопределенности зависит от указанных вероятностей, а количество информации в сообщении определяется тем, насколько данная мера уменьшается с получением сообщения. Если все сообщения

равновероятны, а их число конечно, то в качестве меры неопределенности используется мера Хартли.

К. Шеннон обобщил эту ситуацию на случай, когда количество информации зависит не только от n, но и от вероятностей выбора того или иного сообщения или получения сигнала, введя уже рассмотренное нами понятие энтропии H, то есть меры неопределенности ситуации вида

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i,$$

где p_i – вероятности возможных событий.

Данное соотношение иначе можно получить следующим образом. Попробуем определить *среднее* значение количества информации $I_{\text{ср.}}$, получаемой при одном опыте, если предполагается N его возможных исходов n разных типов, а i-й исход повторяется n_i раз и предоставляет количество информации, равное I_i. Тогда при данных обозначениях получим

$$I_{\text{ср.}} = (n_1 I_1 + n_2 I_2 + \ldots + n_n I_n)/N.$$

Количество информации, выражаемое в битах и получаемое от каждого исхода опыта, К. Шеннон связал с вероятностью его появления p_i следующим соотношением:

$$I_i = \log_2(1/p_i) = -\log_2 p_i.$$

Тогда с учетом того, что $p_i = n_i/N$, так как соответствующие отношения представляют собой частоты повторения исходов опытов и могут быть заменены их вероятностями, получим

$$I_{\text{ср.}} = p_1(-\log_2 p_1) + p_2(-\log_2 p_2) + \ldots + p_n(-\log_2 p_n),$$

и окончательно:

$$I_{\text{ср.}} = H = -\sum_{i=1}^{n} p_i \log_2 p_i.$$

Если все события равновероятны и независимы, то оценки количества информации, полученные при структурном и статистическом подходах,

совпадают. В случае неравных вероятностей статистический подход дает меньшее значение. Этот вывод иллюстрируется следующим примером.

Т а б л и ц а 1.1

Частотные вероятности русских букв

i	Символ	p_i	i	Символ	p_i	i	Символ	p_i
1	Пробел	0,175	13	К	0,028	25	Ч	0,012
2	О	0,090	14	М	0,026	26	Й	0,010
3	Е	0,072	15	Д	0,025	27	Х	0,009
4	Ё	0,072	16	П	0,023	28	Ж	0,007
5	А	0,062	17	У	0,021	29	Ю	0,006
6	И	0,062	18	Я	0,018	30	Ш	0,006
7	Т	0,053	19	Ы	0,016	31	Ц	0,004
8	Н	0,053	20	З	0,016	32	Щ	0,003
9	С	0,045	21	Ь	0,014	33	Э	0,003
10	Р	0,040	22	Ъ	0,014	34	Ф	0,002
11	В	0,038	23	Б	0,014			
12	Л	0,035	24	Г	0,013			

В таблице 1.1 указаны вероятности частоты употребления различных символов русского алфавита, полученные путем анализа весьма объемных текстов. Сначала подсчитаем количество информации, связанное с появлением любого символа в сообщениях, записанных на русском языке, по формуле Хартли:

$$I = \log_2 34 \approx 5 \text{ бит}.$$

Полученная величина I является максимальным количеством информации, которое могло бы приходиться на один символ. Если подсчитать этот же параметр с учетом приведенных в таблице вероятностей по формуле Шеннона, то получим $I = H \approx 4{,}72$ бит.

Следует подчеркнуть, что первые результаты, связанные с измерением количества информации и оказавшиеся очень полезными при решении ряда практических задач, были получены Р. Хартли. Однако предложенная им мера степени неопределенности, в то же время, часто оказывается мало пригодной, так как она совершенно не учитывает различие между имеющимися

исходами эксперимента: ведь почти невероятному исходу придается такое же значение, как и очень правдоподобному.

К. Шеннон устранил этот недостаток, предложив учитывать вероятностный характер рассматриваемых процессов и введя понятие энтропии в теорию информации. При таком подходе в качестве меры неопределенности $H(\alpha)$ всего эксперимента α принимается *среднее* значение неопределенности отдельных исходов. Этот результат, оказавшийся весьма продуктивным для многих приложений, связанных, прежде всего, с передачей сообщений по линиям связи, однако, также не может претендовать на универсальность. Ведь полученная мера $H(\alpha)$ зависит лишь от вероятностей различных исходов опыта, но не учитывает «природу» этих исходов, другими словами, являются ли они в некотором смысле «близкими» или «далекими» друг от друга. Например, рассматриваемая «степень неопределенности» оказывается одинаковой для оценки предполагаемого выступления шахматиста в двух турнирах, в обоих из которых его две партии закончатся ничейным исходом, но оставшиеся восемь партий он в первом турнире выиграет, а во втором – проиграет.

1.4. Информация и энтропия

Понятие энтропии в XIX веке в научный обиход ввел Р. Клаузиус – один из основателей термодинамики и молекулярно-кинетической теории теплоты, что позволило сформулировать следующие постулаты: для всякой материальной системы энтропия есть мера близости системы к состоянию равновесия; если энтропия в системе максимальна, то никакая работа в ней не может совершаться. Затем Л. Больцман дал кинетическое обоснование

второго закона термодинамики, экспериментально доказав, что энтропия H замкнутого пространства определяется соотношением:

$$H = -(1/N)\sum_{i=1}^{N} n_i \ln(n_i/N),$$

где N – общее число молекул в данном пространстве;

n_i – число молекул, обладающих скоростью $u_i + \Delta u_i$.

Поскольку $p_i = n_i/N$ есть вероятность того, что молекула имеет указанную скорость, выражение для H примет следующий вид:

$$H = -\sum_{i=1}^{N} p_i \ln p_i,$$

Полученная формула эквивалентна формуле Шеннона – в обоих случаях величина энтропии характеризует степень разнообразия системы.

Теперь становится понятным, почему Шеннон полученную им меру неопределенности назвал энтропией. Однако надо осторожно относиться к выявленной формальной аналогии, поскольку формула Больцмана определяет энтропию статистической системы материальных частиц, а формула Шеннона была получена специально для решения некоторых практических вопросов, связанных с передачей сообщений по линиям связи, поэтому она оказалась особенно удобна именно для такого технического использования.

Энтропия обладает следующими свойствами:

1. Она всегда неотрицательна, так как вероятности $0 \le p_i \le 1$, а их логарифмы не больше нуля.
2. Она равна нулю, когда одна из вероятностей $p_i = 1$, а все остальные вероятности равны 0. Это соответствует случаю, когда неопределенность ситуации отсутствует, и результат опыта не дает никакой информации (он заранее известен).
3. Она имеет наибольшее значение, когда все вероятности равны между собой, то есть все $p_i = 1/k$, и $H_{max} = \log_2 k$. При этом ясно, что источник с

двумя возможными сообщениями, вероятность каждого из которых равна 1/2, обладает энтропией в один бит.

Характер зависимости величины энтропии от значений вероятностей для ситуации с двумя возможными исходами представлен на рисунке 1.2.

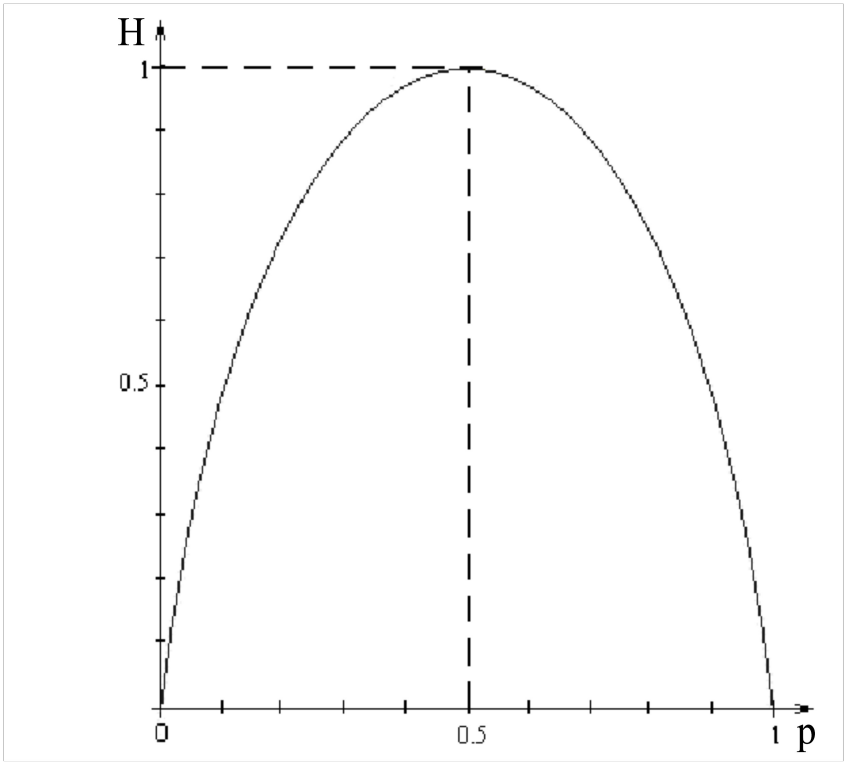


Рис. 1.2. Зависимость энтропии от вероятностей для ситуации с двумя возможными исходами

4. Энтропия сложного события (например, рассмотренный ранее сложный эксперимент $\alpha\beta$, заключающийся в одновременном выполнении двух независимых опытов α и β) определяется следующим правилом сложения энтропий:

$$H(\alpha\beta) = H(\alpha) + H(\beta).$$

5. Для более общего случая, когда опыты α и β не являются независимыми, энтропия сложных событий также определяется правилом сложения энтропий, которое теперь выглядит так:

$$H(\alpha\beta) = H(\alpha) + H_{\alpha}(\beta),$$

где $H_{\alpha}(\beta)$ – условная энтропия опыта β при условии выполнения опыта α. Эта величина является неотрицательным числом. $H_{\alpha}(\beta) = 0$ в том и только в том случае, если при любом исходе α результат β становится полностью определенным, при этом, конечно, $H(\alpha\beta) = H(\alpha)$. Если же α и β являются независимыми, то $H_{\alpha}(\beta) = H(\beta)$, и получается предыдущая формула сложения; причем для условной энтропии всегда имеет место соотношение:

$$0 \le H_{\alpha}(\beta) \le H(\beta).$$

Введенное понятие условной энтропии позволяет трактовать разность $H(\beta) - H_{\alpha}(\beta)$ следующим образом. Она показывает, насколько обоснованнее мы можем судить об исходе опыта β, осуществив опыт α. Следовательно, выражение

$$I(\alpha,\beta) = H(\beta) - H_{\alpha}(\beta)$$

определяет количество информации о β, содержащейся в α, и появляется возможность численного измерения информации.

Итак, получая информацию в результате некоторого процесса познания, ее потребитель тем самым уменьшает степень своего незнания, то есть неопределенность ситуации для него становится меньше. В этом случае количество информации будет равно разности между начальной $H_н$ и конечной $H_к$ энтропией или между априорной (до опыта) и апостериорной (после опыта) энтропией:

$$I = H_н - H_к.$$

Если эта неопределенность снимается полностью (апостериорная энтропия превратилась в нуль), то только тогда количество полученной информации равно энтропии $H_н$, причем оно будет максимальным, если

априорная энтропия была наибольшей, когда все вероятности возможных событий были одинаковы.

1.5. Информация и данные

На первый взгляд может показаться, что два этих понятия – синонимы, но это не так. Данные можно трактовать как зарегистрированные результаты наблюдений за объектами окружающего нас материального мира и происходящими в нем событиями. Такие результаты являются сигналами, имеющими энергетическую природу, поскольку их появление всегда связано с любым обменом энергией и ее переходом из одной формы в другую.

Итак, *данные* – это каким-либо образом зафиксированные сигналы, которые хранятся в определенном виде для их последующего применения. Когда они начинают использоваться для уменьшения существующей неопределенности в той или иной предметной области, то превращаются в информацию. Таким образом, информация рассматривается как более общее понятие, ведь ею являются используемые данные.

Информация, как и данные, может быть *структурирована*. В этом случае в ней всегда содержатся и различные *отношения*, например, причинно-следственные, в результате чего она образует единую систему, которую мы привыкли называть *знанием*. Когда данные воспринимаются человеком, они, как мы уже установили, становятся для него информацией. Но ведь разные люди в зависимости от своего образовательного уровня и по другим причинам по-разному воспринимают один и тот же набор фактов или могут вовсе не понять смысл, заложенный в них. Следовательно, присущее данным свойство превращаться в информацию является потенциальным, так как может быть реализовано или нет различными людьми.

Необходимо также отметить, что *компьютер всегда* работает с данными, а не с информацией, поскольку он не может мыслить и, значит, интерпретировать ее. Между тем, в устной речи и разнообразных текстах (в том числе и в этом учебном пособии) мы часто по привычке смешиваем эти два понятия, но установленные между ними различия всегда надо иметь в виду.

Любая компьютерная *программа*, реализующая определенный *алгоритм*, при функционировании осуществляет преобразование исходных данных в конечные, в том числе с использованием промежуточных результатов, поэтому большое значение приобретают способы представления и организации данных. В ходе эволюции средств вычислительной техники и программирования также повышались и возможности представления данных. В настоящее время кроме простейших неструктурированных данных широко используются различные типы структурированных данных, которые так называются потому, что образуются в результате разнообразных комбинаций простых данных и обладают определенной организационной структурой. К ним относятся массивы, записи, списки, стеки и т.д.

Простейший тип *неструктурированных* данных соответствует простому правилу: одно имя – одно значение. Как известно, в математике используются целые, вещественные или действительные и комплексные числа, а также логические величины. В информатике, а точнее в языках программирования, по аналогии с этим любая константа, переменная, выражение или функция также относятся к определенному типу данных, характеризующему множество значений, которые они могут принимать. Этот тип явно указывается в описании соответствующей величины, и над данными каждого типа могут выполняться определенные операции.

Например, над *целыми* числами могут выполняться операции сложения, вычитания и умножения. Чтобы результат деления не вышел за пределы множества целых чисел, используются две разновидности этой операции:

определение целой части или остатка от деления одного числа на другое. Аналогично эти операции над целыми числами выполняются и в компьютере. Результат вычислений также оказывается абсолютно точным. Единственное отличие заключается в том, что компьютерные числа имеют ограниченный диапазон – это связано с их представлением в ячейках памяти, поэтому при любой системе кодирования всегда существуют самое большое (положительное, $M_{+\infty}$) допустимое и самое малое (отрицательное, $M_{-\infty}$) числа. Как правило, между ними выполняются такие соотношения:

$$M_{+\infty} + 1 = M_{-\infty} \text{ и } M_{-\infty} - 1 = M_{+\infty},$$

которые станут более понятными, когда мы рассмотрим представление отрицательных чисел в разрядной сетке компьютера с помощью обратного кода, о чем речь пойдет в следующей главе, а сейчас поговорим о так называемых числах *конечной точности*.

Предположим, для записи целых положительных чисел в нашем распоряжении только три десятичных разряда. В этом случае мы можем представить тысячу чисел, от 000 до 999, но в этот ряд не войдут числа определенных типов, а именно: числа больше 999 и отрицательные числа, дроби, иррациональные и комплексные числа. В результате сложения, вычитания и умножения целых чисел всегда также получаются целые числа, поэтому говорят, что набор этих чисел замкнут по отношению к этим трем действиям. Однако по отношению к делению набор целых чисел не замкнут, так как в результате выполнения этой операции далеко не всегда получаются целые числа.

Числа конечной точности не замкнуты относительно всех четырех арифметических действий. Ведь в результате сложения или умножения трехразрядных десятичных чисел может получиться слишком большой результат (число больше 999), в результате вычитания – слишком маленький результат (отрицательное число), наконец, при делении – не целое число.

Приведенные примеры можно разбить на два класса: первые три случая относятся к одному из них, а четвертый – к другому. Ошибки переполнения или из-за потери значимости возникают, когда результат выполнения операции выходит за пределы представления чисел; ошибочный результат также получается, когда он не принадлежит заданному ряду чисел, то есть является числом другого типа. Компьютеры работают с числами конечной точности, поэтому в них предусмотрены специальные аппаратные средства для обнаружения и исправления ошибок переполнения.

Осталось отметить, что ассоциативный и дистрибутивный законы также не работают с числами конечной точности.

Пусть $a = 600$, $b = 400$, $c = 100$. Вычислим обе части выражения

$$a + (b - c) = (a + b) - c.$$

В левой части сначала вычисляется разность, равная 300, а затем сумма; в итоге получим 900. При вычислении в правой части суммы трехразрядных чисел произойдет переполнение, так как результат 1000 выходит за пределы рассматриваемого набора целых чисел; в итоге после вычитания не получится 900.

Теперь попробуем вычислить обе части выражения

$$a \cdot (b - c) = a \cdot b - a \cdot c$$

при $a = 5$, $b = 200$, $c = 100$. В левой части получим результат 500, которого не достичь для правой части, потому что снова произойдет переполнение при умножении a на b. Таким образом, порядок выполнения действий с такими числами имеет важное значение.

Над *вещественными* числами могут выполняться все четыре арифметические операции. Однако надо иметь в виду, что при компьютерной реализации точность получаемого результата никогда не превышает некоторого фиксированного значения, так как числа в памяти компьютера имеют ограниченную длину.

Символьные или *литерные* данные кодируются числовыми кодами, поэтому их можно сравнивать. Другими словами, они обладают свойством упорядоченности, например: “*a*” < “*b*”, “*b*” < “*c*” и т.д., кроме того, обычно по символу можно получить его числовой код и наоборот.

Над *строчными* величинами выполняется единственная операция конкатенации или сложения строк, например,

“*ab*” + “*cde*” = “*abcde*”.

Как правило, можно определить длину строки, т.е. количество входящих в нее символов, удалить из нее фрагмент и т. п.

К *логическим* данным, которые могут принимать только одно из двух возможных значений – «истина» («true») или «ложь» («false»), применимы основные логические операции: конъюнкция (логическое И, «and»), дизъюнкция (логическое ИЛИ, «or») и отрицание (логическое НЕ, «not»). Об этом мы поговорим подробнее в третьей главе.

Структурированные данные предоставляют программисту гораздо больше возможностей при обработке информации, хотя надо иметь в виду, что различные языки программирования по-разному реализуют структуризацию данных. Однако к настоящему времени многие такие сложные структуры стали практически стандартными и реализованы в большинстве используемых алгоритмических языков. К рассмотрению этих структур, образуемых из простых типов данных, мы и перейдем, но сначала дадим их общую классификацию, проведенную по четырем основным признакам и представленную на рисунке 1.3.

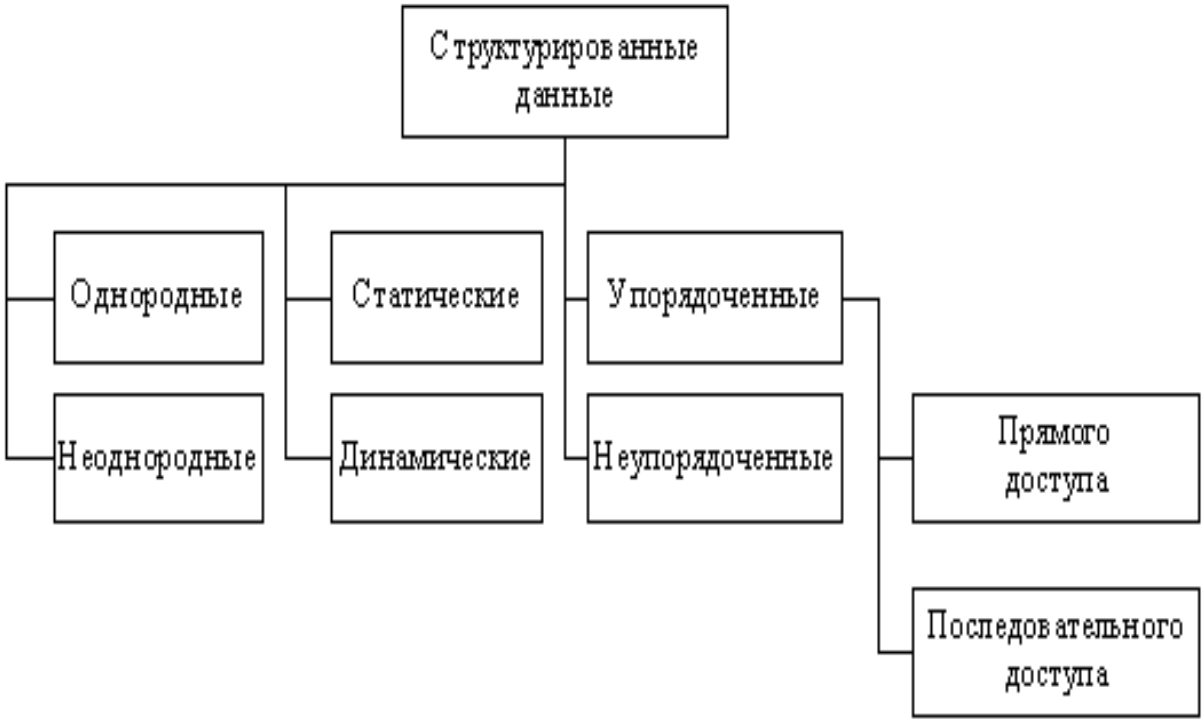


Рис. 1.3. Классификация структурированных данных

Сложная структура может состоять из однотипных или разнотипных простых данных. В первом случае такая структура является *однородной*, во втором – *неоднородной*. Структура считается *упорядоченной*, если между всеми входящими в нее элементами определен порядок следования; в противном случае, естественно, структура *не упорядочена*. Упорядоченные структуры по способу доступа к их элементам подразделяются на структуры *прямого* и *последовательного* доступа. При наличии возможности прямого доступа любой элемент такой структуры сразу доступен для использования, если же этой возможности нет, то, чтобы «добраться» до нужного элемента, необходимо перебрать все предыдущие. Наконец, если размер структуры, то есть ее длина или количество входящих в нее элементов, строго фиксирован заранее и не может быть изменен в процессе обработки, то такая структура является *статической*. У *динамической* структуры размер определяется во время работы с ней и при необходимости может изменяться.

Пожалуй, наиболее распространенным типом структурированных данных является *массив* – в нашей терминологии это однородная упорядоченная статическая структура прямого доступа. Компоненты массива представляют собой его однотипные элементы, которые объединяются общим именем – *идентификатором* массива. Благодаря этому и с помощью индекса они, имея одно и то же имя, но различаясь по индексам, могут идентифицироваться или адресоваться, то есть любая компонента массива доступна для обработки как одиночная *переменная*.

Наиболее простой структурой рассматриваемого типа является *одномерный* массив *линейной* структуры, содержащий фиксированный набор определенных величин. Однако компонентами массива, в свою очередь, могут быть и более сложные структуры, например другие массивы. В этом случае формируется как бы массив массивов, то есть *многомерный* массив, для индексации элементов которого требуется уже более одного индекса. Так двумерный массив, обычно называемый *матрицей* и представленный в виде таблицы 1.2, состоит из трех одномерных массивов, в каждом из которых по четыре компоненты.

Т а б л и ц а 1.2

Двумерный массив

	1	2	3	4
1	a_{11}	a_{12}	a_{13}	a_{14}
2	a_{21}	a_{22}	a_{23}	a_{24}
3	a_{31}	a_{32}	a_{33}	a_{34}

В этой таблице 3×4 три строки и четыре столбца. Любой из двенадцати элементов такого массива представляется в виде a_{ij}, где первый индекс i указывает номер строки таблицы и изменяется от одного до трех, а второй индекс j – номер столбца и изменяется от одного до четырех. На пересечении определенных строки и столбца в клетке таблицы и находится

соответствующая компонента. Этот двумерный массив можно было бы трактовать и несколько иначе: он состоит из четырех одномерных массивов, в каждом из которых по три элемента. Все зависит от того, как мы будем «просматривать» таблицу: по строкам или по столбцам. Важно, что в обоих случаях наш массив содержит 12 компонент.

Запись является неоднородной упорядоченной статической структурой прямого доступа, используемой для хранения логически связанной информации. Она представляет собой набор, как правило, разнотипных *полей*, объединенных общим именем, которые могут идентифицироваться или адресоваться как по имени записи, так и по именам полей. При работе с совокупностью записей имя поля состоит из двух частей (имя записи, а также собственное имя этого поля), то есть является составным. В терминологии *баз данных* запись и поля иногда определяют как кортеж и атрибуты соответственно. По сравнению с массивом запись можно считать более универсальным типом структурированных данных, поскольку она является неоднородной структурой. С другой стороны, массив – более гибкая структура, поскольку индексы его элементов можно вычислять, а имена полей записи – нельзя.

Принципиально иным типом структурированных данных является *множество*, которое трактуется в математике как неупорядоченная совокупность неповторяющихся объектов. Над множествами определены следующие операции: проверка принадлежности элемента данному множеству, объединение множеств (аналог операции сложения), пересечение множеств (операция умножения) и теоретико-множественная разность (вычитание множеств). Следует отметить, что размер множества заранее не фиксируется, хотя обычно ограничивается конкретной компьютерной реализацией (например, в нем может содержаться не больше, чем 255 элементов). Кроме того, доступ к любому элементу множества

осуществляется единственным путем: проверкой его принадлежности данному множеству.

Теперь перейдем к рассмотрению более сложных типов структурированных данных и начнем с файлов или очередей, но сначала нам придется уточнить само понятие «файл».

Дело в том, что этот распространенный термин применяется в информатике в различных смыслах. Первоначально им стали обозначать последовательный набор данных, а также команд, то есть программу, хранящихся на внешних носителях, основными из которых в то время была магнитная лента. Но в английском языке это слово имеет несколько значений: архив, картотека, комплект, очередь, подшитые бумаги, ряд и т.д. В данном случае мы идентифицируем файл с понятием очереди.

Итак, *очередь* – это однородная линейно упорядоченная динамическая структура *последовательного* доступа, благодаря чему существенное значение приобретает так называемая дисциплина обслуживания очереди. Естественно предложить два варианта пополнения очереди компонентами (записи) и их извлечения (чтения) из очереди. В первом случае новый элемент добавляется в «хвост», а считывание компонентов осуществляется с головы очереди. Это соответствует принципу «первым пришел – первым обслужен» (английская аббревиатура FIFO – «first-in, first-out»). Во втором случае новый элемент также добавляется, конечно, в «хвост», но обслуживание очереди начинается именно с него, то есть первая компонента очереди «выходит» из нее последней. Другими словами, при этом реализуется принцип «последним пришел – первым обслужен» или LIFO – «last-in, first-out». Структура, в которой реализован последний вариант обслуживания, называется *стеком* или *магазином* (по аналогии с магазином стрелкового оружия), поэтому структуру, соответствующую первому случаю, иногда называют обратным магазином.

До сих пор мы рассматривали сложные структуры, элементы которых, в определенном смысле, были равноправны. Однако часто специфика задачи требует использования данных, которые связаны между собой отношением подчинения. Хорошим примером такой структуры является генеалогическое дерево, корень которого – имя конкретного человека, на следующем уровне располагаются имена его родителей, затем – имена родителей его родителей и т.д. Описанная структура представляет собой *двоичное дерево*, поэтому подобные структуры данных называются *древовидными* или *иерархическими*.

Список также является примером подобного рода структур, в нем кроме собственно данных хранятся и адреса элементов, поэтому любой элемент списка состоит из информационной и адресной частей. Первая часть содержит данные, вторая – указатели на следующие элементы. Список представляет собой основную конструкцию языка Лисп (это название происходит от английского Lisp, list processing – обработка списков). Он является упорядоченной последовательностью, элементами которой могут быть как простейшие данные, называемые в Лиспе атомами, так и другие списки или подсписки. Эта структура данных позволяет представлять иерархическую связь с помощью системы строго соответствующих друг другу открывающих и закрывающих скобок или с использованием определенной точечной нотации. Благодаря этому списки являются основным средством представления знаний, например, с помощью вложенных списков можно записать анкетные данные любого человека.

Наконец, следует отметить, что некоторые языки программирования позволяют образовывать из рассмотренных структур разнообразные суперпозиции, то есть совмещать их. Например, такая суперпозиция как *файл записей* обычно используется при формировании баз данных и работе с ними.

Контрольные вопросы

1. Как осуществляется восприятие информации человеком?
2. Что характеризует совокупность свойств информации?
3. Что понимается под информационным сообщением?
4. Какие существуют уровни рассмотрения сообщений?
5. Что такое тезаурус и как его объем влияет на количество воспринимаемой человеком информации?
6. Что такое дезинформация?
7. Что является минимальным количеством информации?
8. Как принцип дихотомического деления можно использовать при поиске информации?
9. Какие коэффициенты используются при переходе от одной единицы измерения информации к другой?
10. Как определяется вероятность события, наступившего в результате некоторого эксперимента?
11. Как энтропия связана с неопределенностью ситуации?
12. Почему логарифмическая функция принимается за меру неопределенности опыта?
13. Чем структурный подход к измерению количества информации отличается от статистического подхода?
14. Что такое условная энтропия?
15. Чем данные отличаются от информации?
16. Что такое числа конечной точности?
17. Как можно классифицировать структурированные данные?
18. Почему двумерный массив обычно называется матрицей?
19. Какие существуют дисциплины обслуживания очередей?
20. Что хранит любой элемент списка?
21. Что такое суперпозиция структур данных?

2. Числа в компьютере и действия над ними

2.1. Непозиционные и позиционные системы счисления

С давних времен люди выполняли элементарные подсчёты, связанные с житейскими потребностями. Наиболее древние числительные, к которым можно отнести один, два, пять, десять, двадцать, видимо, связаны с самыми естественными счётными приспособлениями – пальцами рук и ног.

В древнем Египте стали использоваться привычные для нас числовые компоненты – единицы, десятки, сотни и тысячи, а числа, обозначаемые иероглифами, представлялись в виде суммы стандартных слагаемых.

Древнеримская система, построенная по такому же принципу, представляет собой более сложную модель, в которой используются следующие обозначения:

1 = I, 5 = V, 10 = X, 50 = L, 100 = C, 500 = D, 1000 = M.

Если меньшее число расположено справа, то оно увеличивает значение предыдущего слагаемого, а если слева – то уменьшает. Например,

MCMXLVI = 1000+900+40+5+1 = 1946.

Запись чисел не очень сложна, но арифметические действия в древнеримской системе выполнять весьма затруднительно. В настоящее время она иногда используется для представления натуральных чисел.

Под *системой счисления* можно понимать искусственный язык, специально предназначенный для представления чисел. Различают *непозиционные* и *позиционные* системы счисления. В непозиционных системах, к которым относятся упомянутые древнеегипетская и древнеримская системы, значение каждой цифры в изображении числа *не зависит* от позиции, которую эта цифра занимает в записи числа, в позиционных – зависит.

Древнейшая из известных нам позиционных систем счисления – вавилонская *шестидесятеричная система*, основание которой равно 60, то есть одна и та же цифра (а всего их 60) в последующем (левом) разряде значит в 60 раз больше, чем в предыдущем. В дальнейшем древние вавилоняне упростили свою систему, перейдя к *двенадцатеричной системе счисления*. Поскольку они обладали большими познаниями в области астрономии, так ими было вычислено движение всех пяти известных тогда планет солнечной системы, то, вероятно поэтому, отголоски их систем счисления дошли до наших дней. Ведь в году у нас 12 месяцев, а не 10, циферблат часов содержит 12 чисел, в одном часе 60 минут и т.д. Кроме того, в некоторых языках существует специальное слово для обозначения именно цифры 12: например, в русском языке – это дюжина, в английском – dozen.

В настоящее время общеупотребительной является позиционная *десятичная система счисления*, «завезенная» в Европу арабами, которые восприняли ее у индусов. Она имеет основание 10, в ней, как известно, для записи чисел используются 10 различных цифр, а любое число может быть представлено в виде суммы степеней числа 10, умноженных на соответствующие коэффициенты, взятые из записи числа, например:

$$123{,}45 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2}.$$

Итак, в десятичной системе разряду единиц числа соответствует нулевая степень десяти, разряду десятков – первая степень, разряду сотен – вторая, разряду десятых – минус первая степень и т.д., а любая цифра изменяет свой вес в 10 раз при перемещении в числе на один разряд вправо или влево.

С развитием вычислительной техники нашли применение позиционные системы счисления с другими основаниями, например, двоичная, троичная, восьмеричная, шестнадцатеричная. Алфавиты цифр этих систем приведены в таблице 2.1:

Таблица 2.1

Алфавиты цифр систем счисления

N	Цифры														
2	0	1													
3	0	1	2												
8	0	1	2	3	4	5	6	7							
10	0	1	2	3	4	5	6	7	8	9					
16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Элементы, применяемые в компьютере для представления чисел, являются двухпозиционными, то есть обладают двумя устойчивыми состояниями, одно из которых обозначается как 0, другое – как 1, поэтому числа в нем представляются в *двоичной системе*. Технический элемент, «понимающий» и «запоминающий» эти состояния, должен распознавать лишь два сигнала: высокий уровень сигнала может соответствовать единице, а низкий – нулю. Ясно, что строить элементы, распознающие 10 различных состояний, соответствующих десяти разным цифрам, неизмеримо сложнее, и построение вычислительного устройства в этом случае потребовало бы больших затрат.

Итак, в двоичной системе применяются лишь две цифры – 0 и 1, а число представляется в виде суммы степеней двойки, умножаемых на 0 или 1:

$$1010_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 8 + 2 = 10_{10}.$$

Поскольку теперь мы будем работать с разными системами счисления, в качестве индекса числа будем указывать основание системы, в которой это число записано.

Троичная система также имеет отношение к вычислительной технике, поэтому следует упомянуть о ней. Дело в том, что, с одной стороны, чем меньше основание системы счисления, тем меньшее количество различных цифр используется для записи чисел (это видно из таблицы 2.1), а для кодирования последних – меньше различных физических состояний. В этом

смысле наиболее удобной для применения в компьютерах является двоичная система. С другой стороны, чем больше основание системы, тем короче запись числа. Этот вывод подтверждается последним примером, в котором число десять в десятичной системе является двухразрядным, а в двоичной системе – четырехразрядным. Следовательно, в этом случае требуется меньше аппаратуры для представления чисел. Было показано, что оптимальной с точки зрения наименьшего количества используемой аппаратуры является система счисления с основанием $e \approx 2{,}718$ (рис. 2.1).

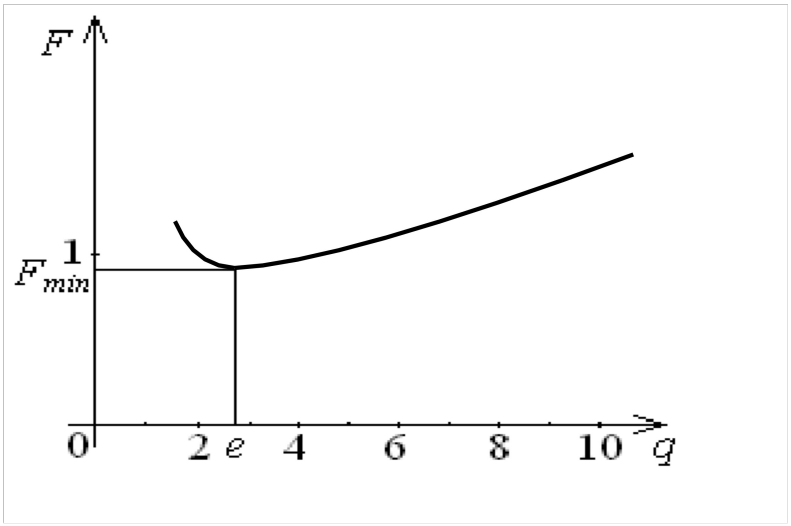


Рис. 2.1. Зависимость количества аппаратуры от системы счисления

Такую систему неудобно применять, поэтому, округляя до ближайшего целого, получим, что троичная система наилучшим образом удовлетворяет поставленной задаче минимизации количества аппаратуры (без учета других соображений). И действительно, на начальном этапе развития были построены несколько компьютеров, работающих с использованием троичной арифметики. Однако от этой идеи быстро отказались, так как в пользу двоичной системы «перевесили» такие факторы, как простота и быстродействие используемых элементов, а также степень сложности выполнения арифметических и логических операций.

В *восьмеричной системе* для записи чисел используются 8 цифр:

$$0, 1, 2, 3, 4, 5, 6, 7,$$

а любое число может быть представлено в виде суммы степеней восьмерки, умноженных на эти цифры (коэффициенты), например:

$$145_8 = 1 \cdot 8^2 + 4 \cdot 8^1 + 5 \cdot 8^0 = 101_{10}.$$

Восьмеричная система занимает промежуточное место между двоичной и десятичной системами и может использоваться для удобства ручного перевода чисел из одной системы в другую. Кроме того, в «старых» ЭЦВМ (электронных цифровых вычислительных машинах – такая аббревиатура была раньше общеупотребительна в нашей стране, позднее она трансформировалась в ЭВМ) команды, составляющие программу, записывались в восьмеричной системе счисления, что сокращало длину их записи, поскольку при этом каждая тройка двоичных цифр (двоичная триада) заменялась одним символом.

Шестнадцатеричная система также является вспомогательной и предназначена для удобства представления относительно длинных двоичных чисел, поскольку при этом каждая четверка двоичных цифр (двоичная тетрада) заменяется одним символом. Некоторая сложность ее использования заключается в том, что в ней для записи чисел используются 16 цифр (а мы привыкли к десяти), поэтому десятичные числа от десяти до пятнадцати включительно в шестнадцатеричной системе представляются в виде цифр. Для обозначения недостающих цифр используются начальные заглавные буквы латинского алфавита:

$$10 = A,\ 11 = B,\ 12 = C,\ 13 = D,\ 14 = E,\ 15 = F.$$

Тогда шестнадцатеричное число можно расписать по степеням шестнадцати следующим образом:

$$3C4_{16} = 3 \cdot 16^2 + C \cdot 16^1 + 4 \cdot 16^0 = 3 \cdot 256 + 12 \cdot 16 + 4 \cdot 1 = 964_{10}.$$

В таблице 2.2 приведены примеры записи некоторых целых десятичных чисел в рассмотренных системах счисления.

Т а б л и ц а 2.2

Числа систем счисления

Десятичное	Двоичное	Восьме-ричное	Шестнадца-теричное
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
30	11110	36	1E
40	101000	50	28
50	110010	62	32
60	111100	74	3C
70	1000110	106	46
80	1010000	120	50
90	1011010	132	5A
100	1100100	144	64

В общем случае любое смешанное число A в позиционной системе счисления с основанием N может быть представлено в виде полинома по основанию N:

$$A_N = k_{n-1} \cdot N^{n-1} + \ldots + k_1 \cdot N^1 + k_0 \cdot N^0 + k_{-1} \cdot N^{-1} + \ldots + k_{-m} \cdot N^{-m}.$$

Более компактно эта же запись выгляди так:

$$A_N = \sum_{i=-m}^{n-1} k_i \cdot N^i .$$

Итак, существуют несколько позиционных систем счисления, нашедших применение в вычислительной технике. В связи с этим возникает проблема перевода чисел из одной системы в другую.

2.2. Перевод чисел из одной системы счисления в другую

Сначала рассмотрим способы перевода *целых чисел*.

Чтобы перевести исходное число из двоичной (восьмеричной или шестнадцатеричной) системы счисления в десятичную систему, его надо представить в виде суммы степеней числа 2 (чисел 8 или 16 соответственно, если речь идет о восьмеричной или шестнадцатеричной системах), умноженных на соответствующие коэффициенты, взятые из записи числа, и полученные произведения сложить. При этом надо учитывать, что разряд единиц целого числа соответствует нулевой степени, следующий разряд десятков – первой степени и т.д.

Фактически примеры такого перевода чисел рассмотрены ранее, когда шла речь о представлении чисел в той или иной позиционных системах.

Обратный перевод осуществляется следующим образом.

Чтобы перевести целое число из десятичной системы счисления в двоичную (восьмеричную или шестнадцатеричную) систему, его надо последовательно нацело делить на 2 (8 или 16), запоминая остатки, до тех пор, пока в последнем частном не получим 1 (число, меньшее восьми или

шестнадцати соответственно). Затем, начиная с этого последнего частного, выписываются все остатки.

Например:

```
35 | 2                    Или  35 | 1_____________
-34  17 | 2                        17 | 1
  1 -16  8 | 2                     8 | 0
     1 -8  4 | 2                   4 | 0
        0 -4  2 | 2                2 | 0
           0 -2  1.                1 |
              0
```

Наконец, эту же схему перевода можно представить и так:

Частные: 35:2→17:2→8:2→4:2→2:2→1.

Остатки: 1 1 0 0 0.

Результат, естественно, во всех трех случаях будет один и тот же:

$$35_{10} = 100011_2.$$

Рассмотрим еще несколько примеров перевода с использованием последней схемы:

Частные: 687:16→42:16→2.

Остатки: 15 10.

Результат:

$$687_{10} = 2AF_{16}.$$

Частные: 101:2→50:2→25:2→12:2→6:2→3:2→1.

Остатки: 1 0 1 0 0 1.

Таким образом, получаем, что

$$101_{10} = 1100101_2.$$

Аналогично имеем:

101:8→12:8→1 и $101_{10} = 145_8$ или 101:16→6 и $101_{10} = 65_{16}$.
5 4 5

Для быстрого перевода десятичных чисел в двоичную систему полезно знать степени числа 2, которые часто используются. В этом случае схема перевода может быть представлена по-другому:

Показатель: 0 1 2 3 4 5 6 7 8 9 10.

Степень: 1 2 4 8 16 32 64 128 256 512 1024.

Предположим, надо перевести десятичное число 241 в двоичную систему. Замечаем, что 241 = 128 + 64 + 32 + 16 + 1. Тогда получим:

Степени двойки: 128 64 32 16 8 4 2 1.

Двоичные разряды: 1 1 1 1 0 0 0 1,

то есть

$$241_{10} = 11110001_2.$$

Теперь рассмотрим, как переводить числа из двоичной системы счисления в восьмеричную или шестнадцатеричную системы и обратно.

Чтобы перевести целое двоичное число в восьмеричную или шестнадцатеричную системы, его, начиная с младшего разряда, надо разбить на триады или тетрады, то есть на группы по три или четыре двоичных разряда, и каждую триаду (тетраду) представить цифрой восьмеричного (шестнадцатеричного) числа, например:

$$11001101_2 = 11'001'101_2 = 315_8 \text{ или } 11001101_2 = 1100'1101_2 = CD_{16}.$$

Обратный перевод производится аналогично: каждая восьмеричная или шестнадцатеричная цифры представляются двоичной триадой или тетрадой соответственно, например:

$$567_8 = 101'110'111_2 = 101110111_2,$$

$$A19_{16} = 1010'0001'1001_2 = 101000011001_2.$$

В заключение рассмотрим два примера "полного" перевода целых чисел.

Пример 1:

$$1101110101_2 \rightarrow a_8 \rightarrow b_{10} \rightarrow c_{16} \rightarrow d_2.$$

$d_2 \rightarrow a_8$:

$$1'101'110'101_2 = 1565_8;$$

$a_8 \rightarrow b_{10}$:

$$1565_8 = 1 \cdot 8^3 + 5 \cdot 8^2 + 6 \cdot 8^1 + 5 \cdot 8^0 = 512 + 320 + 48 + 5 = 885_{10};$$

$b_{10} \rightarrow c_{16}$:

$$885:16=55:16=3 \text{ и } 885_{10} = 375_{16};$$
$$5 \quad 7$$

$c_{16} \rightarrow d_2$:

$$375_{16} = 11'0111'0101_2 = 1101110101_2 .$$

Таким образом, $1101110101_2 = 1565_8 = 885_{10} = 375_{16} = 1101110101_2$.

Пример 2:

$$AB4_{16} \rightarrow a_{10} \rightarrow b_8 \rightarrow c_2 \rightarrow d_{16} .$$

$d_{16} \rightarrow a_{10}$:

$$AB4_{16} = A \cdot 16^2 + B \cdot 16^1 + 4 \cdot 16^0 = 10 \cdot 256 + 11 \cdot 16 + 4 = 2740_{10};$$

$a_{10} \rightarrow b_8$:

$$2740:8=342:8=42:8=5 \text{ и } 2740_{10} = 5264_8;$$
$$4 \quad 6 \quad 2$$

$b_8 \rightarrow c_2$:

$$5264_8 = 101'010'110'100 = 101010110100_2;$$

$c_2 \rightarrow d_{16}$:

$$101010110100_2 = 1010'1011'0100 = AB4_{16} .$$

Окончательно получаем: $AB4_{16} = 2740_{10} = 5264_8 = 101010110100_2 = AB4_{16}$.

Теперь займемся переводом *правильных дробей* и *смешанных чисел*.

Для перевода правильной десятичной дроби ее необходимо умножить на основание той системы, в которую она переводится, полученная при этом целая часть произведения является первой цифрой после запятой искомого числа. Затем дробные части получающихся произведений также умножаются с получением новых цифр в виде целых частей до тех пор, пока не будет получена требуемая точность. Например:

$0{,}125_{10} \rightarrow a_2$:

$$
\begin{array}{r}
0{,}125 \\
\times \quad 2 \\
\hline
0{,}250 \\
\times \quad 2 \\
\hline
0{,}500 \\
\times \quad 2 \\
\hline
1{,}000
\end{array}
$$

$0{,}125_{10} = 0{,}001_2$.

В данном примере процесс перевода завершился с получением точного результата, так как дробная часть последнего произведения оказалась нулевой. И действительно

$$0{,}001_2 = 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = 1/8 = 0{,}125_{10} .$$

Однако ясно, что этого можно достичь далеко не всегда (например, при исходном числе 0,126). Вот поэтому была упомянута требуемая точность: в общем случае процесс перевода надо продолжать, пока не будет получено нужное количество цифр после запятой. А это связано с понятием разрядной сетки, о чем речь пойдет в дальнейшем.

Для перевода смешанного числа по рассмотренным правилам отдельно переводятся его целая и дробная части, а результаты объединяются. Например:

$$110110{,}11_2 \rightarrow a_{10} .$$

Сначала переведем целую часть:

$$
\begin{array}{rl}
110110 & \big| \; 1010 \\
-1010 & \quad 101 \\
\hline
1110 & \\
-1010 & \\
\hline
100 &
\end{array}
$$

Итак, $110110_2 = 54_{10}$,

а $0{,}11_2 = 0{,}75_{10}$.

Переводим дробную часть числа:

$$
\begin{array}{r}
0{,}11 \\
\times \quad 1010 \\
\hline
1010 \\
+ \quad 1010 \\
\hline
111{,}10 \\
\times \quad 1010 \\
\hline
0000 \\
+ \quad 1010 \\
\hline
101{,}00
\end{array}
$$

Таким образом, $110110{,}11_2 = 54{,}75_{10}$.

Разумеется, этот перевод можно было сделать проще, не выполняя деление и умножение чисел в двоичной системе счисления:

$$110110{,}11_2 = 2^5 + 2^4 + 2^2 + 2^1 + 2^{-1} + 2^{-2} = 54\ 3/4 = 54{,}75_{10}\ .$$

Возможен и третий вариант перевода, обоснованность и алгоритм которого попробуйте сформулировать сами:

$$110110{,}11_2 \rightarrow 128 + 64 + 16 + 8 + 2 + 1 = 219 \rightarrow 219/4 = 54\ 3/4 = 54{,}75_{10}.$$

2.3. Машинные коды чисел

Арифметические операции в компьютере выполняются в двоичной системе счисления, причем все четыре действия сводятся к операциям сложения и сдвига чисел. Чтобы это обеспечить, для представления чисел используются прямой, обратный и дополнительный коды.

Прямой код $[x]_{пр.}$ правильной двоичной дроби определяется условиями:

$$[x]_{пр.} = x, \text{ при } x \geq 0 \text{ или } [x]_{пр.} = 1 - x, \text{ при } x < 0,$$

$$\text{то есть } [x]_{пр.} = 0{,}x_1x_2\ldots x_n \text{ или } [x]_{пр.} = 1{,}x_1x_2\ldots x_n.$$

Здесь x_i обозначают двоичные разряды числа, а 0 или 1 проставляются в знаковом разряде для обозначения положительных или отрицательных чисел соответственно. Например:

$$x = -\,0{,}101001 \rightarrow [x]_{пр.} = 1 - (-\,0{,}101001) = 1{,}101001.$$

Следовательно, прямой код двоичного числа совпадает с записью самого числа, а в знаковом разряде проставляются 0 или 1 для положительных и отрицательных чисел соответственно.

Обратный код $[x]_{обр.}$ определяется условиями:

$$[x]_{обр.} = x, \text{ при } x \geq 0 \text{ или } [x]_{обр.} = 10 + x - 10^n, \text{ при } x < 0,$$

где n – количество разрядов.

Например:

$$x = -\,0{,}1101 \rightarrow [x]_{обр.} = 10 + (-\,0{,}1101) - 0{,}0001 = 1{,}0010.$$

Разумеется, здесь 10 – «два» в двоичной системе счисления.

Следовательно, обратный код отрицательного числа образуется путем записи в знаковый разряд 1, а цифровые разряды инвертируются.

Дополнительный код $[x]_{доп.}$ определяется условиями:

$$[x]_{доп.} = x, \text{ при } x \geq 0 \text{ или } [x]_{доп.} = 10 + x, \text{ при } x < 0.$$

Например:

$$x = -0{,}1101 \rightarrow [x]_{доп.} = 10 + (-0{,}1101) = 1{,}0011.$$

Следовательно, дополнительный код отрицательного числа образуется путем записи в знаковый разряд 1, цифровые разряды инвертируются, а к младшему разряду прибавляется 1.

Рассмотренные коды чисел используются для замены вычитания сложением. Однако, например, при сложении двух правильных дробей результат может оказаться больше единицы, что приведет к переполнению уже упомянутой разрядной сетки. Переполнение означает, что результат операции вышел за диапазон чисел, представляемых в данной разрядной сетке. При этом теряются старшие разряды результата, и он может изменить свой знак. Для быстрого обнаружения этого факта и возможного исправления его последствий применяются *модифицированные* обратный $[x]^{M}_{обр.}$ и дополнительный $[x]^{M}_{доп.}$ коды. Они образуются по таким же правилам, лишь под знак числа отводятся два разряда: сочетание цифр 00 в знаковых разрядах свидетельствует о том, что число положительное, цифр 11 – оно отрицательное. Например:

$$x = -0{,}10101 \rightarrow [x]^{M}_{обр.} = 11{,}01010;\ [x]^{M}_{доп.} = 11{,}01011.$$

Признаком переполнения разрядной сетки служит появление в знаковых разрядах суммы сочетания цифр 01 или 10.

2.4. Размещение чисел в разрядной сетке

Совокупность двоичных разрядов, предназначенных для хранения и обработки чисел, образует *разрядную сетку* компьютера. Применяются две формы представления чисел: естественная – с фиксированной запятой (точкой) и нормальная или полулогарифмическая – с плавающей запятой (точкой).

При естественной форме представления двоичных чисел *с фиксированной запятой* последняя фиксируется перед старшим разрядом (при представлении правильных дробей) или после младшего (при представлении целых чисел), отделяя целую часть числа от дробной. Под знак числа отводится нулевой бит (0 – «+», 1 – «–»), остальные разряды заполняются числом, младшая цифра которого размещается справа. Например, десятичное число 3105, которое сначала удобнее перевести в шестнадцатеричную систему счисления, а затем уже – в двоичную, в 16-тиразрядной сетке будет размещено следующим образом (рис. 2.2).

$$3105_{10} = C21_{16} = 110000100001_2.$$

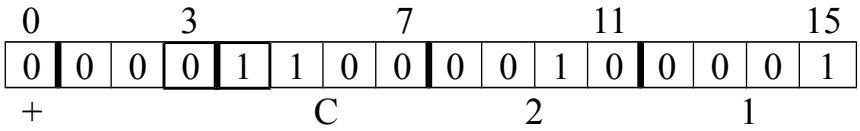


Рис. 2.2. Размещение целого числа в разрядной сетке по форме с фиксированной запятой

При таком представлении наибольшее число, которое может быть записано в данной разрядной сетке, равно $2^{15} – 1$. Следует также отметить, что здесь старшая тетрада является неполной (разряды 1, 2 и 3).

При нормальной (полулогарифмической) форме представления чисел *с плавающей запятой* они изображаются в виде

$$N = m \cdot q^p,$$

где m – мантисса числа ($|m| < 1$);

q – основание *системы счисления*;

p – порядок числа, который указывает положение запятой в числе, при разных порядках положение запятой различно – отсюда и название – с плавающей запятой. Например:

$$123_{10} = 0{,}123 \cdot 10^3 = 0{,}0123 \cdot 10^4 = \ldots$$

Запись такого вида называется полулогарифмической потому, что в логарифмической форме представляется не все число, а только его часть q^p .

Различают *нормализованные* и *ненормализованные* числа, первые – это такие, у которых абсолютная величина мантиссы удовлетворяет условию

$$1/q \le |m| < 1.$$

Ясно, что первое из приведенных чисел (с порядком $p = 3$) является нормализованным, а второе – нет. Для двоичной системы имеем $1/2_{10} = 0{,}1_2$, то есть число будет нормализованным, если после запятой сразу идет значащая цифра, а если в старшем разряде 0, то это ненормализованное число. Операция нормализации осуществляется в компьютере автоматически путем сдвига мантиссы влево с соответствующим уменьшением порядка. Например: $N_2 = 0{,}00101 \cdot 10^{101} = 0{,}10100 \cdot 10^{11}$.

Число $555{,}441_{10} = 22\text{B}{,}70\text{E}5_{16} = 0{,}22\text{B}70\text{E}5 \cdot 10^3$ в 32-хразрядной сетке будет размещено следующим образом (рис. 2.3).

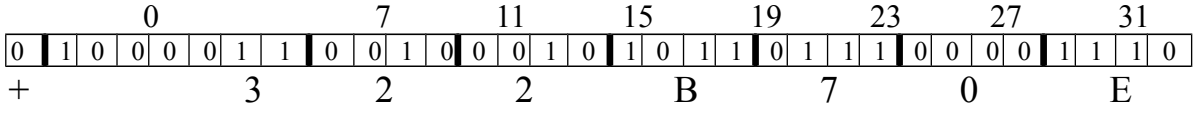


Рис. 2.3. Размещение смешанного числа в разрядной сетке по форме с плавающей запятой

Необходимо обратить внимание на два момента.

Во-первых, младший разряд нашего шестнадцатеричного числа (или младшая тетрада двоичной мантиссы) не поместился в разрядной сетке, то есть мы получили приближенное число. Однако существуют возможности оперирования с числами двойной и даже большей точности, например, путем

увеличения количества разрядов в разрядной сетке. К тому же, поскольку в разрядной сетке компьютера всегда записываются нормализованные числа, у которых старший разряд мантиссы в двоичном представлении всегда равен единице, то он не хранится, хотя и учитывается при выполнении операций (так называемый скрытый разряд), и таким образом выгадывается еще один двоичный разряд для представления числа.

Во-вторых, для записи порядка числа используются разряды с первого по седьмой, и в первом появилась единица. В результате получился так называемый смещенный порядок. Это делается для того, чтобы порядок можно было представлять без знака, таким способом упрощая действия с порядками чисел.

2.5. Выполнение арифметических операций

Выполнение арифметических операций компьютером отличается от того, как эти действия реализует человек. Это связано с тем, что компьютер оперирует с числами, точность которых в общем случае конечна и фиксирована (об этом шла речь в первой главе). Данное ограничение определяется фиксированным размером разрядной сетки. Хотя программист может работать с числами в два, три и более раз большими, чем позволяет размер разрядной сетки, но эта проблема все равно остается. Нам же, в принципе, все равно, сколько десятичных разрядов содержат используемые нами числа, поскольку никогда не возникнет проблемы нехватки бумаги для их записи. Последнее соображение можно проиллюстрировать историей вычисления числа π.

Число π (отношение длины окружности к ее диаметру) является трансцендентным, то есть таким числом, которое не может получиться в результате решения любых алгебраических уравнений с рациональными

коэффициентами. Кстати, установление этого факта в 1882 г. «закрыло» знаменитую проблему квадратуры круга – построения при помощи циркуля и линейки квадрата, равновеликому данному кругу, история которой насчитывает более четырех тысячелетий. Так вот, еще Архимед установил, что число π заключено между 3 10/71 и 3 1/7. В XVI веке π было вычислено сначала с 9-ю, а затем с 32-я знаками (последний результат потребовал от его автора десятилетнего труда). Наконец, в XIX веке один англичанин, затратив более 20-ти лет жизни, вычислил 707 знаков числа π. К счастью для «рекордсмена», уже после его смерти было обнаружено, что он ошибся в 520-м знаке, и все последующие цифры полученного выражения неверны. Для вычисления π использовались бесконечные ряды или произведения, сходящиеся к π, например ряд, открытый Лейбницем:

$$\pi = 4 \cdot (1 - 1/3 + 1/5 - 1/7 + 1/9 - \ldots).$$

Однако следует подчеркнуть, что для получения, например, шестизначного π необходимо взять в этом ряду $2 \cdot 10^6$ членов.

С появлением компьютеров число π было вычислено в 1949 году сначала с точностью свыше 2000 знаков, затем точность постоянно увеличивалась и к нашему времени достигла миллиона знаков. Разумеется, это представляет лишь технический, но вряд ли научный или практический интерес, поскольку для практических целей вполне достаточно знать, что $\pi \approx 3{,}14159265$.

Итак, сначала рассмотрим выполнение операций над числами с фиксированной запятой. При выполнении *сложения* машинные коды чисел просто складываются, при этом коды отрицательных чисел преобразуются в обратный или дополнительный коды. Таким образом, *вычитание* сводится к сложению путем определения для кода вычитаемого дополнения, которое прибавляется к уменьшаемому. Единица переноса из знакового разряда при использовании дополнительного кода не учитывается, при использовании

обратного кода она прибавляется к младшему разряду суммы, реализуя так называемый циклический перенос. Например:

$$x = 0{,}1101,\ y = -0{,}0110,\ x + y = ?$$

$$[x]^{М}_{доп.} = 00{,}1101 \quad \text{или} \quad [x]^{М}_{обр.} = 00{,}1101$$

$$+\ \underline{[y]^{М}_{доп.} = 11{,}1010} \qquad +\ \underline{[y]^{М}_{обр.} = 11{,}1001}$$

$$100{,}0111 \qquad\qquad 100{,}0110$$

$$+\ \underline{\qquad 1}$$

$$00{,}0111$$

Итак, x + y = 00,0111. И действительно: 13 + (– 6) = 7.

Необходимо отметить, что суммирование обратных кодов чисел выполняется за два шага, включая циклический перенос, что замедляет получение результата. По этой причине предпочтительнее представление и обработка чисел в компьютере в дополнительных кодах.

Умножение сводится к последовательности операций суммирования и сдвига (влево или вправо). Под сдвигом понимается перемещение содержимого разрядной сетки на определенное количество разрядов в одну сторону, при этом освобождающиеся разряды заполняются нулями. Различают арифметический и логический сдвиги: в первом случае сдвигаются только цифровые разряды, во втором – все разряды двоичного кода. Следует заметить, что сдвиг целого числа на один разряд влево увеличивает его в два раза, сдвиг на два разряда – увеличивает в четыре раза и т.д. Аналогичным образом сдвиг вправо эквивалентен делению на 2, 4 и т.д. Максимальная разрядность произведения определяется суммой разрядов сомножителей (n + n = 2n), а знак произведения – путем сложения знаковых разрядов сомножителей. Например:

$$x = 0{,}110,\ y = 1{,}101,\ x \cdot y = ?$$

Определяем знак произведения: 0 + 1 = 1 – произведение отрицательно.

Выполняем умножение:

$$
\begin{array}{r}
0{,}110 \\
\times\ \underline{0{,}101} \\
110 \\
+\ 000 \\
\underline{110} \\
11110
\end{array}
$$

Итак, x · y = 1,11110. И действительно: 6 ·(– 5) = – 30.

Деление состоит в последовательности операций вычитания и сдвигов, причем делитель вычитается из делимого или очередного остатка. При делении без восстановления остатка полученный в результате вычитания остаток сдвигается на разряд влево. Если он положительный, из него вычитается делитель и в разряд частного записывается 1, если отрицательный – к нему прибавляется делитель и в разряд частного записывается 0. Например:

x = 0,00100, y = 0,10100, x : y = ?

$[y]_{доп.}$= 1,01100.

0.00100	1.11000 → 0	1.01000 → 0	Проверка:
+1.01100	+ 0.10100	+ 0.10100	$x_{10} = 4$, $y_{10} = 20$, x : y = 0,2.
1.10000	0.01100	1.11100	0,2
←	←	←	× 2
1.00000 → 0	0.11000 → 1	1.11000 → 0	0,4
+ 0.10100	+ 1.01100	+ … и т.д.	× 2
1.10100	0.00100		0,8
←	←		× 2
1.01000 → 0	0.01000 → 1		1,6
+ 0.10100	+ 1.01100		× 2
1.11100	1.10100		1,2
←	←		× 2
Таким образом, x : y = 0.00110011…			0,4 …

Теперь рассмотрим выполнение операций над числами с плавающей запятой. *Сложение* и *вычитание* выполняются как над мантиссами, так и над порядками. Сначала порядки выравниваются путем увеличения меньшего,

для чего мантисса соответствующего числа сдвигается вправо на число разрядов, равное разности порядков. Затем производится сложение мантисс и при необходимости нормализация результата. Например:

$$x = 0{,}101001 \cdot 10^{110},\ y = -0{,}101110 \cdot 10^{101},\ x + y = ?$$

Порядок первого числа на 1 больше порядка второго числа, поэтому после выравнивания порядков имеем

$$y = -0{,}010111 \cdot 10^{110}.$$

Складываем мантиссы:

$$\begin{aligned} [m_x]^{M}_{\text{пр.}} &= 00{,}101001 \\ + [m_y]^{M}_{\text{доп.}} &= \underline{11{,}101001} \\ & \quad 00{,}010010\ (p = 110). \end{aligned}$$

Нормализуем результат с уменьшением порядка на 1:

$$x + y = 0{,}10010 \cdot 10^{101}.$$

И действительно: 41 + (– 23) = 18.

Умножение и *деление* выполняются, в общем, аналогично тому, как и эти операции над числами с фиксированной запятой. Например, выполним умножение двоичных чисел (для упрощения расчетов их порядки, которые при умножении складываются, а при делении вычитаются, представлены в обычной двоичной форме):

$(0.11110 \times 10^{101}) \times (0.1001 \times 10^{100}) = (0.11110 \times 0.1001) \times 10^{(101\ +\ 100)} =$

$= 0.100001110 \times 10^{1001}$.

И действительно: $30 \times 9 = 270$.

Мы рассмотрели простейшие способы выполнения арифметических операций в компьютере. Существуют и более сложные методы, позволяющие уменьшить время их выполнения, которые изучаются в курсах «Организация ЭВМ» и «Схемотехника». Отметим в заключение, что в персональном компьютере арифметика с плавающей точкой имеет алгоритмическую и аппаратную поддержки, причем последняя реализуется в виде сопроцессора и обеспечивает большую скорость выполнения арифметических операций.

Контрольные вопросы

1. Что такое система счисления?
2. В чем различие между непозиционной и позиционной системами счисления?
3. Почему числа в компьютере представляются в двоичной системе?
4. В чем сложность использования шестнадцатеричной системы?
5. Как осуществляется перевод целых и дробных чисел из одной системы счисления в другую?
6. К каким операциям сводится выполнение арифметических действий в компьютере?
7. Как образуются обратный и дополнительный коды двоичных чисел?
8. Как образуются модифицированные коды чисел и для чего они применяются?
9. Что такое разрядная сетка компьютера?
10. Как двоичное число представляется по форме с фиксированной запятой?
11. Почему нормальная форма представления числа с плавающей запятой еще называется полулогарифмической?
12. Что такое нормализованные и ненормализованные числа?
13. Для чего нужна операция нормализации числа?
14. Как образуется смещенный порядок числа?
15. В чем отличие операций сложения чисел с фиксированной и плавающей запятыми?
16. По какому правилу может осуществляться операция деления чисел?

3. Элементы алгебры логики

3.1. Высказывания и логические связки

Алгебра логики – это раздел математической логики, в котором изучаются законы, выражаемые логическими формулами, построенными из логических переменных и символов логических связок. Алгебру логики также называют булевой алгеброй – в честь англичанина Джорджа Буля, опубликовавшего в 1847 г. трактат «Математический анализ логики», в котором уже известные тогда логические правила были представлены в математической форме.

В любом естественном языке, на котором общаются люди, основной конструкцией фразы является предложение, простое или сложное (составное). Сложные предложения состоят из нескольких простых, соединенных (или разделенных) знаками препинания, союзами или словосочетаниями «И», «ИЛИ», «ЕСЛИ ... ТО» и другими.

В алгебре логики основной конструкцией является высказывание, которое иногда называют утверждением. ***Под высказыванием понимается такое предложение, о котором можно сказать, что оно истинно или ложно.*** Например:

«Семь – простое число» – истинное высказывание,

«Дважды два – пять» – ложное высказывание.

Наконец, предложения: «Который час?», «Пошел вон!» высказываниями не являются.

Постарайтесь определить, какое из двух следующих высказываний является истинным, а какое – ложным:

1. Всякое высказывание является предложением.
2. Всякое предложение является высказыванием.

Сложные высказывания, как и сложные предложения, также составляются из простых, а роль знаков препинания, союзов или оборотов при этом играют *логические связки*, основные из которых обозначаются следующим образом:

знак ¬ или $\bar{}$ – аналог частицы «НЕ»;

знак $\wedge$ – аналог союза «И»;

знак $\vee$ – аналог союза «ИЛИ»;

знак $\rightarrow$ – аналог словосочетания «ЕСЛИ ...ТО»;

знак $\leftrightarrow$ – аналог словосочетания «ТОГДА И ТОЛЬКО ТОГДА, КОГДА».

Используя введенные обозначения, высказывание: «Два есть простое, а девять – составное число» символически можно записать следующим образом. Это сложное высказывание состоит из двух простых: «Два – простое число», «Девять – составное число», соединенных союзом «а», который в данном случае играет роль союза «и». Обозначив эти простые высказывания заглавными буквами латинского алфавита P и Q соответственно, получим $P \wedge Q$.

В качестве следующих примеров «перевода» используем некоторые из мыслей и афоризмов Козьмы Пруткова. Высказывания «Наука изощряет ум; ученье вострит память» и «Хитрость есть оружие слабого и ум слепого» имеют аналогичную структуру, поэтому оба они представляются как $P \wedge Q$. Действуя подобным образом, высказывания «Если хочешь быть счастливым, будь им» и «Если хочешь быть красивым, поступи в гусары» символически можно записать так: $P \rightarrow Q$. С переводом на язык алгебры логики афоризма «Щелкни кобылу в нос – она махнет хвостом» могут возникнуть некоторые трудности. Дело в том, что на первый взгляд равноценны две записи этого составного высказывания: $P \wedge Q$ и $P \rightarrow Q$. Второй вариант предпочтительней, так как он отражает причинно-следственную связь произведенных действий. Наконец, чтобы символически записать высказывание «Бросая в воду

камешки, смотри на круги, ими образуемые; иначе такое бросание будет пустою забавою», его предварительно необходимо преобразовать без изменения смысла. В результате получим: «Если ты бросаешь в воду камешки и не смотришь на образуемые ими круги, то такое бросание будет пустою забавою». Нетрудно понять, что символическая запись такого высказывания будет выглядеть так: $(P \wedge (\neg Q)) \rightarrow R$.

Однако последний пример ставит вопрос о применении скобок. Чтобы уменьшить их количество при записи символических выражений, приняты следующие *соглашения о «старшинстве» логических связок*, аналогичные соглашениям о порядке выполнения арифметических операций.

Связка $\neg$ является слабейшей, то есть если сразу после нее нет скобки, то она относится только к тому простому высказыванию, непосредственно перед которым находится. Среди оставшихся слабейшей будет связка $\wedge$, затем связки $\vee$ и $\rightarrow$; наконец, сильнейшей является связка $\leftrightarrow$. Другими словами, приоритет логических связок убывает в том порядке, в каком они были определены выше. Такой порядок называется естественным, если его необходимо нарушить, следует использовать скобки.

Ясно, что с учетом принятых соглашений реализация последнего примера может не содержать скобок: $P \wedge \overline{Q} \rightarrow R$.

Теперь постараемся решить обратную задачу: нам надо будет перевести на русский язык сложное высказывание, символически записанное на языке алгебры логики, если известен «перевод» составляющих его простых высказываний. Например, пусть P – «Сегодня воскресенье», Q – «Сегодня понедельник», R – «Сегодня вторник». Тогда составное высказывание $P \rightarrow \neg (Q \wedge R)$ можно перевести так: «Если сегодня воскресенье, то сегодня не понедельник и не вторник».

3.2. Основные логические операции

Сравнительно просто определить, является истинным или ложным простое высказывание: для этого достаточно руководствоваться здравым смыслом и учитывать конкретную ситуацию. Ведь тот факт, что данное высказывание является истинным или ложным, в общем случае не определяется самим высказыванием, а зависит от ситуации. Например, высказывание «Меня зовут Саша» будет истинным, если его произносит человек с таким именем, и ложным в противном случае. Впрочем, существуют и высказывания, которые являются истинными (или ложными) во всех возможных ситуациях. Такие высказывания называются абсолютно истинными (или абсолютно ложными). Примером первого из них является высказывание «Казимир Малевич написал картину «Черный квадрат»», примером второго – «Казимир Малевич написал картину «Три богатыря»».

Итак, простые (как и составные) высказывания могут принимать только одно из двух возможных значений – «истина» или «ложь», которые могут обозначаться как И или Л, true или false, 1 или 0 соответственно. В алгебре логики не рассматривается зависимость истинности высказывания от ситуации, а изучается лишь, как зависит значение истинности составного высказывания от значений входящих в него простых высказываний. При этом логические связки рассматриваются как операции на множестве из двух значений 0 и 1, то есть {0,1}.

И все-таки, как быть с составными высказываниями? Например, как оценить истинность высказывания «Или я – студент, или у меня две головы», если рассматривать его как одно целое? Для этого необходимо изучить *логические операции*, каждая из которых задается соответствующей ей *таблицей истинности*, и правила их выполнения.

Обозначение таких операций соответствует обозначению логических связок, основные из которых мы рассмотрели, а правила их выполнения и оценки истинности или ложности следующие.

Сложное высказывание вида $P \wedge Q$ истинно только тогда, когда истинны и P, и Q одновременно, в остальных случаях оно ложно.

Это определение может быть представлено и в табличном виде (таблица 3.1).

Т а б л и ц а 3.1

Таблица истинности конъюнкции

P	Q	P∧Q
0	0	0
0	1	0
1	0	0
1	1	1

Такая таблица называется *таблицей истинности*, а соответствующая логическая операция – *конъюнкцией* или *логическим произведением*. Последнее название объясняется тем, что в этой таблице фактически записаны правила умножения чисел в двоичной системе. Другие названия этой операции приведены ниже.

Составное высказывание вида $P \vee Q$ ложно только тогда, когда ложны и P, и Q, в остальных случаях оно истинно. Соответствующая таблица истинности (таблица 3.2) выглядит следующим образом:

Т а б л и ц а 3.2

Таблица истинности дизъюнкции

P	Q	P∨Q
0	0	0
0	1	1
1	0	1
1	1	1

Эта логическая операция называется дизъюнкцией, другие ее названия также приведены ниже.

Сложное высказывание вида $P \rightarrow Q$ ложно только тогда, когда P истинно, а Q ложно, в остальных случаях оно истинно (таблица 3.3).

Т а б л и ц а 3.3

Таблица истинности импликации

P	Q	P→Q
0	0	1
0	1	1
1	0	0
1	1	1

Это – *импликация*. Иногда говорят, что результат импликации ложен, если посылка верна, а утверждение – нет.

Последняя таблица истинности может вызвать некоторые трудности для ее понимания, в то время как две предыдущие хорошо согласуются с содержательным смыслом логических связок. Поэтому проиллюстрируем ее следующими двумя примерами.

Очевидно, что высказывание «Если x делится на 4, то x делится на 2» истинно при любом x. Обозначив простые высказывания «x делится на 4» и «x делится на 2» через $P(x)$ и $Q(x)$ соответственно, получим составное высказывание $P(x) \rightarrow Q(x)$, истинное при любом x. При $x = 3$ имеем $P(3) = Q(3) = 0$; при $x = 2$ – $P(2) = 0$, $Q(2) = 1$; при $x = 4$ – $P(4) = Q(4) = 1$. Таким образом, реализуются все строки таблицы истинности для импликации, в которых значение импликации есть 1.

Высказывание «Если ведьмы существуют, то все они – брюнетки» является истинным в том случае, если ведьмы не существуют. Вообще-то «правильность» этого утверждения можно установить, пытаясь выяснить у знакомых брюнеток, не являются ли они ведьмами.

Рассмотренные три логические операции называются *бинарными*, потому что их результат зависит от значений *двух* переменных, но есть ещё и *унарная* операция, содержащая *одну* логическую переменную, и соответствующая ей таблица истинности (таблица 3.4).

Т а б л и ц а 3.4

Таблица истинности инверсии

A	$\bar{A}$
0	1
1	0

Это – *отрицание*, *инверсия* или *операция НЕ*. Из таблицы истинности видно, что если простое высказывание А ложно, то его отрицание – истинно и наоборот. Кстати, вы, очевидно, уже заметили, что в приведенных таблицах истинности «ЛОЖЬ» обозначается как 0, а «ИСТИНА» – как 1.

Необходимо отметить, что *n*-арная логическая операция содержит *n* переменных.

Каждая из рассмотренных бинарных операций содержит четыре набора значений аргументов, поскольку в общем случае с помощью *n* булевых переменных, а их у нас здесь две, можно задать 2^n различных наборов их значений. Понятно, что четырем наборам значений двух аргументов можно сопоставить $2^4 = 16$ разных наборов значений логических операций. Поэтому, вообще говоря, всего существует 16 бинарных логических операций и, соответственно, логических функций, представленных в таблице 3.5, но только 10 из них имеют самостоятельное значение, так как существенно зависят от обоих аргументов. К тому же, кроме введенных, часто используются и другие обозначения логических связок и функций, как и их названия, которые мы и приведем. Отметим также, что в общем случае на *n* аргументах можно задать $2^{2\wedge n}$ логических функций.

Таблица 3.5

Бинарные логические функции

Аргументы		Функции															
x	y	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Функция f_0 – константа 0, абсолютная ложь.

Функция $f_1 = x \wedge y = x \,\&\, y = x \cdot y = xy$ – «x и y» – конъюнкция, логическое произведение, логическое «и», функция совпадения.

Функция $f_2 = \neg (x \rightarrow y) = x \wedge \overline{y}$ – «неверно, что x влечет y; x, но не y» – отрицание импликации, антиимпликация, запрет y.

Функция $f_3 = x$ – переменная x.

Функция $f_4 = \neg (y \rightarrow x) = \overline{x} \wedge y$ – «неверно, что y влечет x; не x, но y» – отрицание обратной импликации, обратная антиимпликация, запрет x.

Функция $f_5 = y$ – переменная y.

Функция $f_6 = x \oplus y = \neg (x \leftrightarrow y)$ – «либо x, либо y; x не эквивалентно y» – сумма по модулю 2, отрицание эквивалентности, неравнозначность, разделительная дизъюнкция.

Функция $f_7 = x \vee y$ – «x или y; либо x, либо y; либо (x и y)» – дизъюнкция, логическая сумма, логическое «или», функция разделения.

Функция $f_8 = x \downarrow y = \neg (x \vee y)$ – «ни x, ни y» – стрелка Пирса, отрицание дизъюнкции, антидизъюнкция, функция Вебба.

Функция $f_9 = x \leftrightarrow y = x \equiv y = x \sim y$ – «x тогда и только тогда, когда y; x эквивалентно y» – эквивалентность, равнозначность.

Функция $f_{10} = \overline{y}$ – «не y» – переменная $\overline{y}$, инверсия от y.

Функция $f_{11} = y \rightarrow x$ – «y влечет x; если y, то x» – обратная импликация.

Функция $f_{12} = \overline{x}$ – «не x» – переменная $\overline{x}$, инверсия от x.

Функция $f_{13} = x \rightarrow y$ – «если x, то y; x влечет y; x имплицирует y» – импликация, логическое следование.

Функция $f_{14} = x \mid y = \neg (xy)$ – «x и y несовместны; неверно, что x и y» – штрих Шеффера, отрицание конъюнкции, антиконъюнкция.

Функция f_{15} – константа 1, абсолютная истина.

Рассмотренные бинарные логические операции и соответствующие им функции позволяют сделать следующие выводы.

Две функции (f_0 и f_{15}) задают константы 0 и 1, функции f_{10} и f_{12} задают унарные инверсии, наконец, функции f_3 и f_5 задают унарные переменные. Таким образом, действительно только 10 из имеющихся 16-ти бинарных логических функций существенно зависят от обоих аргументов.

Возможно, кто-то уже заметил, что каждая из функций имеет свой «антипод» – функцию, значения которой противоположны ей на всех наборах значений аргументов. При $m + n = 15$ для любой функции f_m существует такая функция f_n, что $f_m = \neg f_n$.

3.3. Оценка составных высказываний

С помощью рассмотренных таблиц истинности можно *оценивать*, (то есть определять истинность или ложность) любые сложные высказывания, содержащие эти операции. Для этого сначала надо записать все возможные комбинации переменных (другими словами, ввести исходные данные), а затем последовательно выполнить действия, содержащиеся в логической формуле. При этом надо учитывать "старшинство" операций, которые

соответствуют логическим связкам. Первой должна выполняться инверсия, затем – конъюнкция, дизъюнкция, а последней выполняется импликация. Такой порядок выполнения называется естественным, если его необходимо нарушить, следует применять скобки. Например, оценим истинность составного высказывания $A \rightarrow B \vee \neg C$ (таблица 3.6).

Таблица 3.6

Пример оценки составного высказывания

A	B	C	$\neg C$	$B \vee \neg C$	$\rightarrow$
0	0	0	1	1	1
0	0	1	0	0	1
0	1	0	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	1	1

Сначала заполняются три первых столбца таблицы. Фактически, в них надо записать восемь трехразрядных двоичных чисел от нуля до семи включительно. Если с этим все еще возникают проблемы, то можно воспользоваться другим правилом: в первом столбце записываются четыре нуля и четыре единицы, во втором столбце нули и единицы чередуются через два, а в третьем – через один. Очевидно, что в случае четырех переменных, содержащихся в исходной формуле, потребовалось бы заполнить шестнадцать строк таблицы, поскольку $2^4 = 16$, а первый столбец (из четырех) будет содержать восемь нулей и восемь единиц. Затем последующие столбцы таблицы заполняются по мере выполнения логических действий. В данном случае сначала выполняется отрицание, затем – дизъюнкция и, наконец, – импликация.

Таким образом, данное высказывание ложно в единственном случае: когда A и C истинны, а B ложно, и истинно при всех других комбинациях A, B, и C, то есть можем записать $f(1,0,1) = 0$.

3.4. Логические функции

Последняя запись очень напоминает представление обычных алгебраических функций, но есть и существенные отличия. Например, в выражении $f(x) = 3x$ переменная x является аргументом, то есть независимой переменной, которая в данном случае может принимать любые значения, а затем по формуле вычисляются соответствующие им значения функции.

В алгебре логики логическая переменная может принимать только одно из двух возможных значений – 0 (заменяет словесное обозначение "лжи") или 1 (синоним "истины"). *Логическая функция*, аналогом которой можно считать составное высказывание, также принимает только значения 0 или 1, причём последние "вычисляются" в результате выполнения логических операций, входящих в соответствующую логическую формулу, на основе таблиц истинности. Логические функции также называют булевыми функциями.

Итак, в формулах алгебры логики переменные являются логическими, булевыми или двоичными. Каждая формула задает логическую функцию от логических переменных, причем аргументы и функции алгебры логики также могут принимать только значения 1 или 0. Любую такую булеву функцию $f_i(x_1, \ldots, x_n)$ можно задать своей таблицей истинности, в левой части которой занесены все наборы значений ее аргументов, а правая часть представляет собой результирующий столбец значений функции, соответствующих этим наборам, и число строк такой таблицы равно 2^n. Для этого на каждом наборе аргументов последовательно выполняются действия, содержащиеся в

логической формуле, с учетом «старшинства» операций, как это было проделано в рассмотренном примере. С помощью указанных таблиц можно оценивать (определять истинность или ложность) любые сложные высказывания, другими словами, от функции, заданной в виде формулы, всегда можно перейти к ее табличному заданию. Решение обратной задачи – получение формулы логической функции по ее таблице истинности – мы сейчас и реализуем.

3.5. Получение логической формулы по таблице истинности

Итак, *задать* булеву функцию $f_i(x_1, \ldots, x_n)$ от логических переменных можно двумя способами: во-первых, можно записать формулу для вычисления значений логической функции, во-вторых, можно составить таблицу истинности, в которой содержатся все возможные комбинации аргументов, то есть наборы нулей и единиц, и соответствующие им значения функции (также нули и единицы). Оба эти способа эквивалентны. Удобство формулы – в её компактности, преимущество таблицы истинности заключается в ее наглядности.

Мы рассмотрели, как *составить* таблицу истинности, имея логическую формулу булевой функции, при оценке сложных высказываний. Теперь решим обратную задачу: как *получить* формулу функции по её таблице. Для её решения применим такой *алгоритм*: для каждого набора аргументов, на котором функция равна 1, записываем логическое произведение переменных, причём, если какой-то аргумент в этом наборе равен 0, берется его отрицание, затем все полученные произведения логически складываются.

Пусть задана следующая таблица истинности (таблица 3.7), надо определить, какой логической функции она соответствует.

Т а б л и ц а 3.7

Таблица истинности логической функции ***F***

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
F	0	0	1	1	0	1	0	0

Эта таблица по внешнему виду несколько отличается от предыдущих: в ней исходные данные развернуты по горизонтали, а переменные обозначены как x_i. Итак, имеем:

$$f_2(0,1,0) = f_3(0,1,1) = f_5(1,0,1) = 1.$$

На основе заданного алгоритма можем записать:

$$F = \overline{x_1} \cdot x_2 \cdot \overline{x_3} \vee \overline{x_1} \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x_2} \cdot x_3.$$

В итоге мы получили выражение для логической функции, но его надо постараться упростить. Дело в том, что чем короче формула, тем меньше логических элементов потребуется для её технической реализации при конструировании различных компонентов компьютера. В логических выражениях, как и в алгебраических, можно приводить подобные члены, выносить общий множитель за скобки и т.п. Кроме того, полезно помнить, что в алгебре логики $x \vee \overline{x} = 1$, так как $0 \vee 1 = 1 \vee 0 = 1$. С учётом этого преобразуем полученное выражение, сгруппировав первые два слагаемых и вынеся общий множитель за скобки. Получим:

$$F = \overline{x_1} \cdot x_2 \cdot (\overline{x_3} \vee x_3) \vee x_1 \cdot \overline{x_2} \cdot x_3 = \overline{x_1} \cdot x_2 \vee x_1 \cdot \overline{x_2} \cdot x_3.$$

Таким образом, на основе одной таблицы истинности можно получить несколько логических формул, но надо выбирать из них самую "короткую". В нашем случае имеем

$$F = f_i(x_1,x_2,x_3) = \overline{x_1} \cdot x_2 \vee x_1 \cdot \overline{x_2} \cdot x_3.$$

Полученная формула называется совершенной дизъюнктивной нормальной формой записи функции алгебры логики, задача ее *минимизации*

решается на основе эквивалентных преобразований с помощью соответствующих аксиом, законов и свойств, к изучению которых мы и приступим.

3.6. Минимизация функций алгебры логики

Решение задачи минимизации логической функции достигается, как было отмечено, путем эквивалентных преобразований с использованием следующих аксиом, законов и свойств алгебры логики:

1. Ассоциативность (сочетательный закон):

$$x_1 \vee (x_2 \vee x_3) = (x_1 \vee x_2) \vee x_3 = x_1 \vee x_2 \vee x_3;\ x_1 \cdot (x_2 \cdot x_3) = (x_1 \cdot x_2) \cdot x_3 = x_1 \cdot x_2 \cdot x_3.$$

2. Дистрибутивность (распределительный закон):

$$x_1 \cdot (x_2 \vee x_3) = x_1 \cdot x_2 \vee x_1 \cdot x_3;\ x_1 \vee x_2 \cdot x_3 = (x_1 \vee x_2) \cdot (x_1 \vee x_3).$$

3. Коммутативность (переместительный закон):

$$x_1 \cdot x_2 = x_2 \cdot x_1;\ x_1 \vee x_2 = x_2 \vee x_1.$$

4. Идемпотентность (отсутствие степеней и коэффициентов):

$$x \cdot x = x;\ x \vee x = x.$$

5. Закон двойного отрицания: $\neg\neg x = x$.
6. Свойства констант 0 и 1:

$$x \vee 0 = x;\ x \cdot 0 = 0;\ x \vee 1 = 1;\ x \cdot 1 = x,\ \ \overline{0} = 1;\ \ \overline{1} = 0.$$

7. Законы де Моргана: $\neg (x_1 \cdot x_2) = \overline{x_1} \vee \overline{x_2};\ \neg (x_1 \vee x_2) = \overline{x_1} \cdot \overline{x_2}$.
8. Следствия из 5 и 7: $x_1 \cdot x_2 = \neg (\overline{x_1} \vee \overline{x_2});\ x_1 \vee x_2 = \neg (\overline{x_1} \cdot \overline{x_2})$.
9. Закон противоречия: $x \wedge \overline{x} = 0$.
10. Закон исключенного третьего: $x \vee \overline{x} = 1$.
11. Законы поглощения: $x_1 \vee x_1 \cdot x_2 = x_1;\ x_1 \cdot (x_1 \vee x_2) = x_1$.
12. Закон склеивания: $x_1 \cdot x_2 \vee x_1 \cdot \overline{x_2} = x_1$.

Некоторого пояснения требует дистрибутивность дизъюнкции относительно конъюнкции. Действительно:

$$(x_1 \vee x_2)\cdot(x_1 \vee x_3) = x_1\cdot x_1 \vee x_1\cdot x_2 \vee x_1\cdot x_3 \vee x_2\cdot x_3 = x_1 \vee x_1\cdot x_2 \vee x_1\cdot x_3 \vee x_2\cdot x_3 = =$$

$$x_1\cdot(1 \vee x_2 \vee x_3) \vee x_2\cdot x_3 = x_1 \vee x_2\cdot x_3.$$

Следует иметь в виду, что процесс минимизации логической функции может привести к неоднозначному результату. Рассмотрим такой пример:

$$F = \overline{x_1}\cdot\ \overline{x_2}\cdot\ \overline{x_3} \vee\ \overline{x_1}\cdot\ \overline{x_2}\cdot x_3 \vee\ \overline{x_1}\cdot x_2\cdot\ \overline{x_3} \vee x_1\cdot\ \overline{x_2}\cdot\ x_3 \vee x_1\cdot\ x_2\cdot\ \overline{x_3} \vee x_1\cdot$$

$$x_2\cdot x_3 = \overline{x_1}\cdot\ \overline{x_2}\cdot(\ \overline{x_3} \vee x_3) \vee x_2\cdot\ \overline{x_3}\cdot(\ \overline{x_1} \vee x_1) \vee x_1\cdot x_3\cdot(\ \overline{x_2} \vee x_2) = \overline{x_1}\cdot\ \overline{x_2} \vee x_2$$

$$\cdot\ \overline{x_3} \vee x_1\cdot x_3.$$

Мы сгруппировали первое слагаемое со вторым, третье – с пятым, четвертое – с шестым, вынесли общие множители за скобку и воспользовались законом исключенного третьего, а также свойством константы 1. Однако можно сгруппировать слагаемые по-другому: первое – с третьим, второе – с четвертым, пятое – с шестым. Тогда после аналогичных преобразований мы получим:

$$F = \overline{x_1}\cdot\ \overline{x_2}\cdot\ \overline{x_3} \vee\ \overline{x_1}\cdot\ \overline{x_2}\cdot x_3 \vee\ \overline{x_1}\cdot x_2\cdot\ \overline{x_3} \vee x_1\cdot\ \overline{x_2}\cdot\ x_3 \vee x_1\cdot\ x_2\cdot\ \overline{x_3} \vee x_1\cdot$$

$$x_2\cdot x_3 = \overline{x_1}\cdot\ \overline{x_3}\cdot(\ \overline{x_2} \vee x_2) \vee\ \overline{x_2}\cdot\ x_3\cdot(\ \overline{x_1} \vee x_1) \vee x_1\cdot x_2\cdot(\ \overline{x_3} \vee x_3) =\ \overline{x_1}\cdot\ \overline{x_3} \vee\ \overline{x_2}\cdot$$

$$x_3 \vee x_1\cdot x_2.$$

Итак, мы получили две эквивалентные минимальные дизъюнктивные формы, которые равноценны с точки зрения их последующей технической реализации.

3.7. Логические элементы, реализующие булевы функции

Преобразование информации в компьютере осуществляется электронными устройствами двух классов: комбинационными схемами и цифровыми автоматами.

В *комбинационных схемах* совокупность выходных сигналов y в каждый дискретный момент времени t_i однозначно определяется *комбинацией* входных сигналов x, поступивших на входы схемы в этот же момент времени. Соответствие между входом и выходом комбинационной схемы может быть задано табличным способом, а также в аналитической форме:

$$y_1 = f_1 (x_1, x_2, \ldots, x_n),$$
$$y_2 = f_2 (x_1, x_2, \ldots, x_n),$$
$$\ldots$$
$$y_m = f_m (x_1, x_2, \ldots, x_n).$$

Поскольку все x_i и y_i принимают только два значения: 0 или 1, функции f_i являются булевыми.

Цифровой автомат, в отличие от комбинационных схем, обязательно содержит память. Он имеет конечное число различных внутренних состояний, причем может переходить из одного из них в другое под воздействием входного слова с получением соответствующих выходных слов. Переход от заданных условий работы цифрового автомата к его функциональной схеме осуществляется с помощью аппарата алгебры логики.

Система элементарных логических функций $f_1, f_2, \ldots, f_m$, то есть таких, которые образуются путем использования однородных связей между двоичными переменными, называется *функционально полной* или *базисом*, если любую функцию алгебры логики можно записать в виде формулы или уравнения с помощью суперпозиции (совмещения) элементарных функций. Точно так же набор логических элементов, реализующих указанные функции,

обладает функциональной полнотой, если при помощи конечного числа этих элементов можно построить схему с любым законом функционирования. В алгебре логики доказано, что базисом является набор функций И, ИЛИ, НЕ; функционально полными системами являются также наборы, состоящие из одной функции – стрелка Пирса (ИЛИ-НЕ) или штрих Шеффера (И-НЕ).

На функциональных схемах логические элементы компьютера, реализующие указанные функции, обозначаются так, как показано на рисунке 3.1:

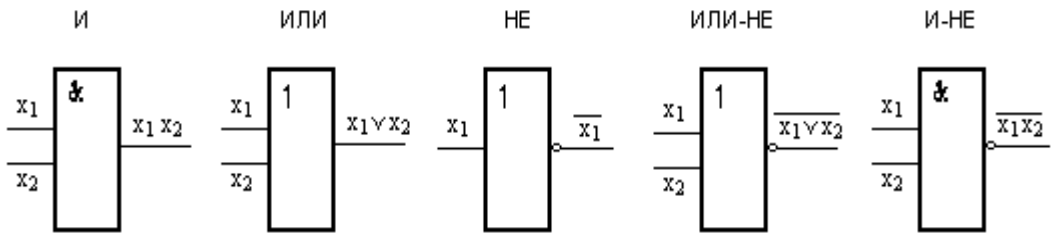
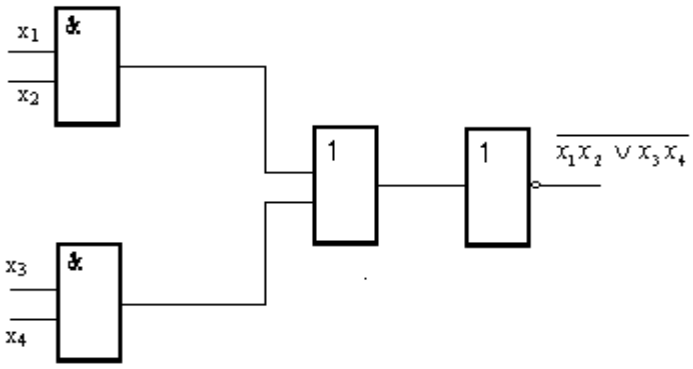


Рис. 3.1. Условные обозначения логических элементов на функциональных схемах

Рассмотренные логические функции являются элементарными, так как образуются путем использования однородных связей между двоичными переменными. Из них можно строить и более сложные функции (на рисунке 3.2 приведены два варианта представления одной и той же схемы), например:



Или более компактное представление:

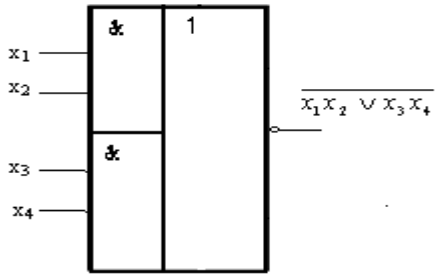


Рис. 3.2. Реализация функции 2И-ИЛИ-НЕ

Тот факт, что система элементов И-НЕ является функционально полной, иллюстрируется рисунком 3.3. Аналогично можно показать функциональную полноту системы элементов ИЛИ-НЕ.

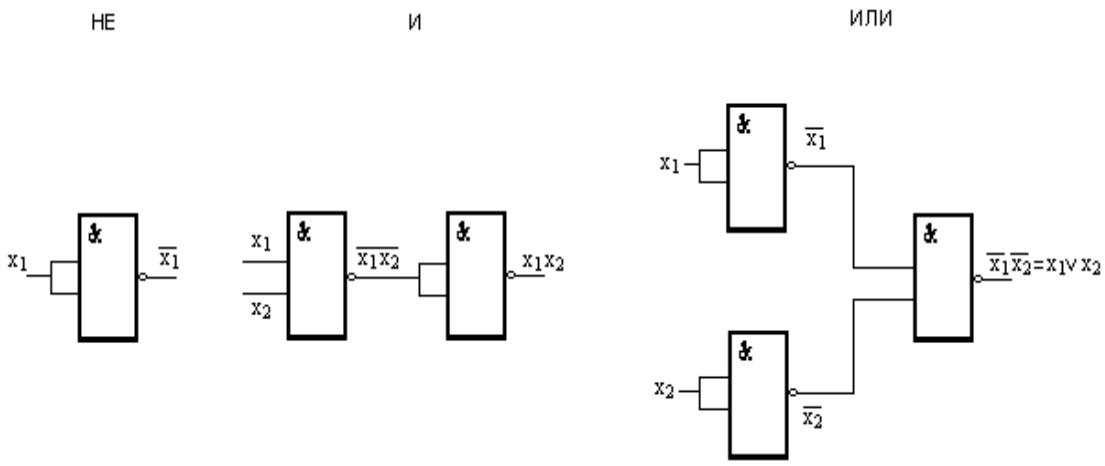


Рис. 3.3. Функциональная полнота системы элементов И-НЕ

Алгоритм функционирования различных логических узлов компьютера задается соответствующей таблицей истинности, которая связывает входные и выходные переменные и от которой можно перейти к аналитической форме описания работы логических устройств, например с помощью совершенной дизъюнктивной нормальной формы записи функции алгебры логики. Мы это проделали при решении задачи получения формулы логической функции по ее таблице истинности с последующим упрощением формулы.

Следует отметить, что в общем случае проблема минимизации логических функций сводится к проблеме выбора базиса, то есть системы

функций, являющейся функционально полной в некотором классе функций алгебры логики, и проблеме наиболее экономного представления функций в этом базисе. Под минимальной дизъюнктивной нормальной формой понимается такая форма, которая обладает минимальным количеством x_i по сравнению с другими дизъюнктивными нормальными формами, удовлетворяющими исходной функции f_i ($x_1, x_2, \ldots, x_n$).

Итак, проектирование логических схем компьютера осуществляется на основе аппарата алгебры логики. Анализ схемы, то есть выяснение того, какие двоичные сигналы появляются на ее выходе с приходом входных сигналов, сводится к выполнению логических вычислений, а синтез и оптимизация логических схем, то есть построение схем, реализующих заданные преобразования и содержащих минимальное число элементов, – к выполнению эквивалентных преобразований в том или ином базисе. Можно еще отметить, что практически любая компьютерная программа содержит команды условного перехода с логическими условиями, истинность или ложность которых определяется по правилам алгебры логики.

3.8. Функциональные схемы некоторых блоков компьютера

Сначала рассмотрим на уровне функциональных схем реализацию, пожалуй, основного блока компьютера – *сумматора*, который выполняет арифметическое суммирование кодов чисел.

При сложении двух чисел x и y в каждом разряде сумматора, начиная со второго, производится сложение трех цифр: двух цифр x_i и y_i данного разряда первого и второго слагаемых и цифры переноса P_i из соседнего младшего разряда. В результате получаются цифра суммы S_i для этого разряда и цифра

переноса P_{i+1} в следующий старший разряд. Работа сумматора иллюстрируется таблицей 3.8:

Т а б л и ц а 3.8

Входы и выходы сумматора

Входы			Выходы	
x_i	y_i	P_i	S_i	P_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

На основании этой таблицы можно составить булевы функции, описывающие функционирование одноразрядного сумматора:

$$S_i = \overline{x}_i \cdot \overline{y}_i \cdot P_i \vee \overline{x}_i \cdot y_i \cdot \overline{P}_i \vee x_i \cdot \overline{y}_i \cdot \overline{P}_i \vee x_i \cdot y_i \cdot P_i,$$

$$P_{i+1} = \overline{x}_i \cdot y_i \cdot P_i \vee x_i \cdot \overline{y}_i \cdot P_i \vee x_i \cdot y_i \cdot \overline{P}_i \vee x_i \cdot y_i \cdot P_i.$$

Выражение для P_{i+1} можно упростить:

$P_{i+1} = \overline{x}_i \cdot y_i \cdot P_i \vee x_i \cdot \overline{y}_i \cdot P_i \vee x_i \cdot y_i \cdot \overline{P}_i = x_i \cdot y_i \cdot \overline{P_i} \vee x_i \cdot y_i \cdot P_i \vee \overline{x_i} \cdot y_i \cdot P_i \vee x_i \cdot y_i \cdot P_i \vee x_i \cdot y_i \cdot P_i \vee x_i \cdot y_i \cdot P_i = y_i \cdot P_i \cdot (\overline{x_i} \vee x_i) \vee x_i \cdot P_i \cdot (\overline{y_i} \vee y_i) \vee x_i \cdot y_i \cdot (\overline{P_i} \vee P_i) = y_i \cdot P_i \vee x_i \cdot P_i \vee x_i \cdot y_i.$

Попробуйте определить, какие соотношения алгебры логики были использованы при проведении этих эквивалентных преобразований. Итак,

$$P_{i+1} = x_i \cdot y_i \vee x_i \cdot P_i \vee y_i \cdot P_i.$$

Функциональная схема одноразрядного *комбинационного сумматора*, реализующего выражения для S_i и P_{i+1}, и его условное обозначение показаны на рисунке 3.4:

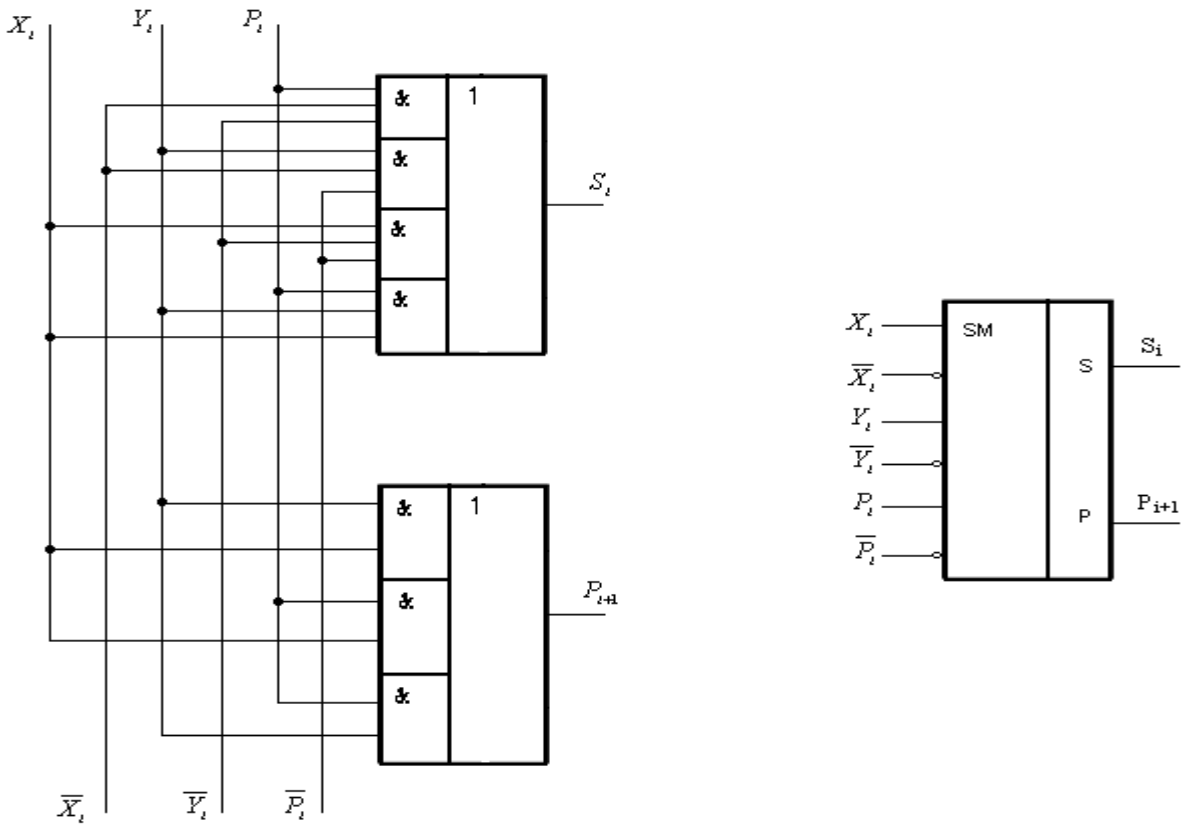


Рис. 3.4. Одноразрядный комбинационный сумматор

Такой сумматор вырабатывает выходные сигналы суммы S_i и переноса P_{i+1}, определяемые комбинацией цифр слагаемых и переноса из младшего разряда, *одновременно* поданных на входы, поскольку он не обладает памятью.

Многоразрядный комбинационный сумматор последовательного действия (рисунок 3.5) производит сложение двух двоичных чисел, представленных в последовательном коде.

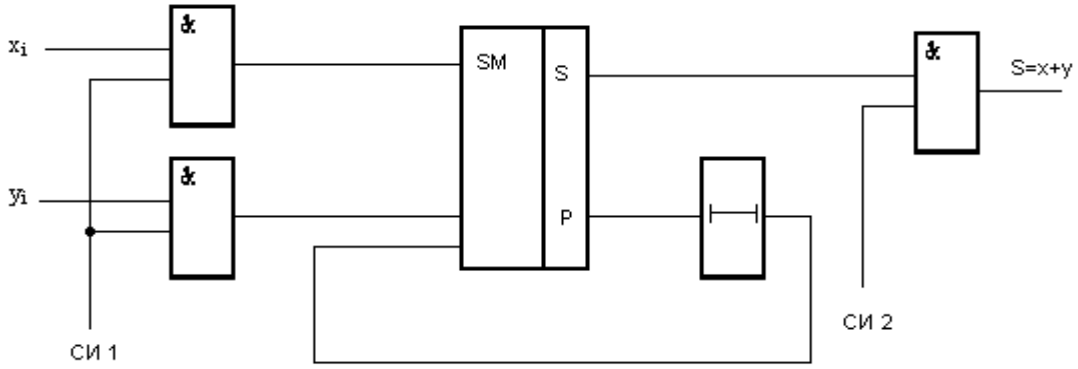


Рис. 3.5. Многоразрядный комбинационный сумматор последовательного действия

Параметры линии задержки (⫢) выбираются такими, чтобы обеспечить поступление сигнала переноса, образующегося при сложении цифр (i –1)-го разряда, одновременно с поступлением i-х разрядов чисел. Для синхронизации поступления входных сигналов используются схемы И, управляемые синхронизирующими импульсами СИ 1, синхроимпульсы СИ 2 обеспечивают выдачу результата.

Из одноразрядных сумматоров можно составить параллельный многоразрядный сумматор, схема которого приведена на рисунке 3.6. Параллельным он называется потому, что в нем входные и выходные сигналы, соответствующие цифрам слагаемых x_i и y_i и суммы S_i в пределах одного разряда, представляются в параллельных кодах.

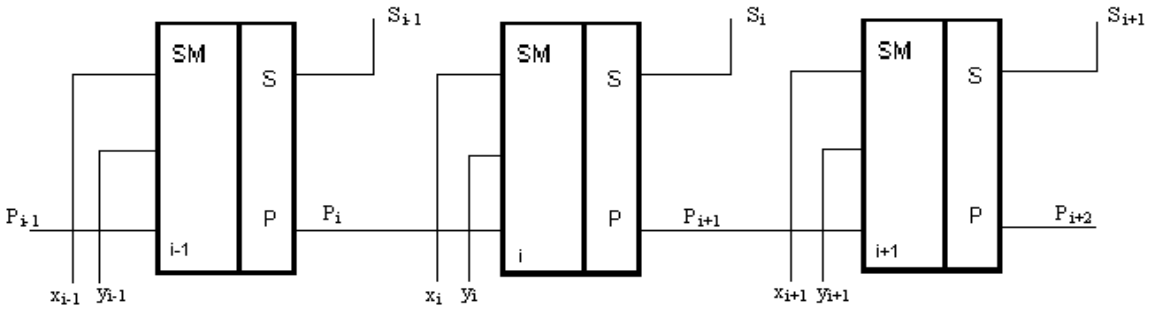


Рис. 3.6. Параллельный сумматор с последовательным переносом

По характеру распространения переноса различают также сумматоры с параллельным и групповым переносом, чтобы быстродействие работы этих схем было выше, но их рассмотрение выходит за рамки нашего курса.

До сих пор приводились примеры реализации сумматоров на основе комбинационных схем, не обладающих памятью. Однако, как уже отмечалось, обработка информации в компьютере осуществляется также цифровыми автоматами.

Простейшим таким автоматом, наиболее широко распространенным в компьютерах, является *триггер* – электронное устройство, обладающее двумя устойчивыми состояниями. Одному из них приписывается значение 1, другому – 0; состояние в котором в данный момент находится триггер, распознается по его выходному сигналу. Под воздействием входного сигнала триггер скачкообразно переходит из одного устойчивого состояния в другое, при этом так же скачкообразно изменяется уровень напряжения его выходного сигнала. Таким образом, триггер выполняет операцию запоминания кодов 0 и 1.

Существуют различные схемы триггеров, но мы рассмотрим только две из них. На рисунке 3.7, *а* приведено условное обозначение асинхронного однотактного RS-триггера, а на рисунке 3.7, *б* – двухтактного триггера со счетным входом.

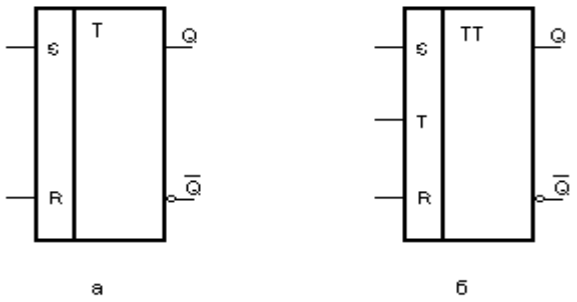


Рис. 3.7. Условные обозначения однотактного и двухтактного триггеров с установочными входами и со счетным входом (а и б)

Сигнал, соответствующий единице и поданный на R-вход RS-триггера, устанавливает его в нулевое состояние. Единичный сигнал, поданный на S-вход, устанавливает триггер в единичное состояние. Следует подчеркнуть, что если триггер находился в состоянии 0, то 1 на R-входе не изменит его состояния; аналогично 1 на S-входе не приведет к изменению единичного состояния триггера. Если на оба входа поступают сигналы, соответствующие 0, то состояние триггера, естественно, не меняется, и он находится в режиме хранения предыдущего состояния. Одновременная подача единичных сигналов на оба входа такого триггера запрещена. Приведенные рассуждения иллюстрируются таблицей 3.9:

Т а б л и ц а 3.9

Работа RS-триггера

t		*t*+1	Состояние
R	S	Q	
0	0	Q(*t*)	Хранение
0	1	1	Установка 1
1	0	0	Установка 0
1	1	-	Запрещено

Когда RS-триггер находится в единичном состоянии, на его прямом выходе Q – высокий потенциал, соответствующий коду 1; нулевое состояние характеризуется низким потенциалом кода 0 на выходе Q. Для инверсного выхода триггера – все наоборот: 1 на выходе для нулевого состояния и 0 – для единичного.

Отличие триггера со счетным входом от RS-триггера заключается в том, что он перебрасывается, то есть меняет свое состояние на противоположное, от каждого входного сигнала, соответствующего 1 и поступающего на его счетный вход.

Одноразрядным накапливающим сумматором, представленным на рисунке 3.8, называется схема, осуществляющая суммирование поочередно поступающих на вход цифр слагаемых и переноса с запоминанием результата сложения. Такой сумматор строится на основе триггера со счетным входом.

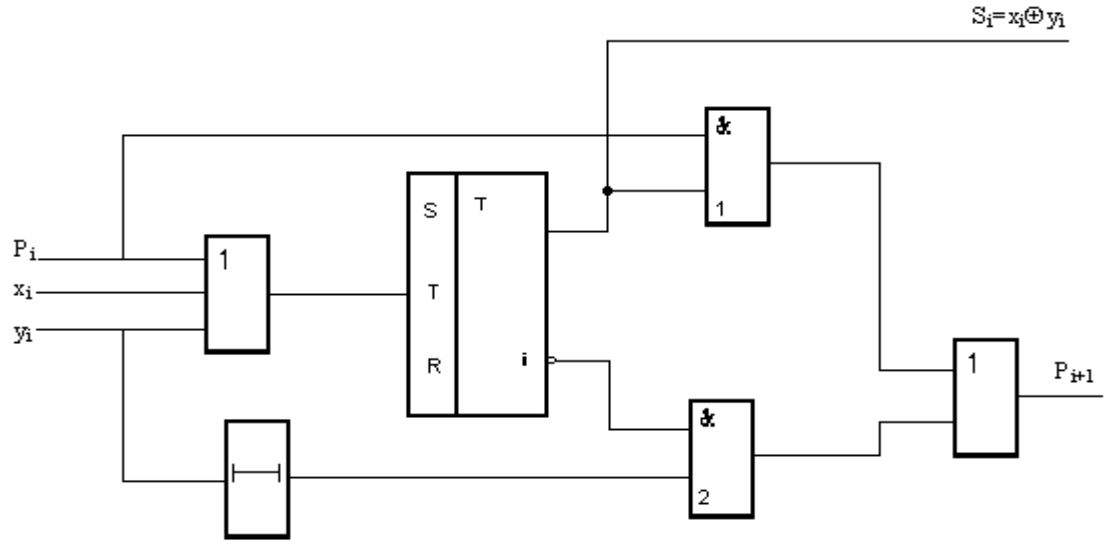


Рис. 3.8. Одноразрядный накапливающий сумматор

В момент t_i на вход поступает цифра первого слагаемого x_i и записывается в триггер, затем – код второго слагаемого y_i; в триггере образуется результат их сложения по модулю два. После окончания в триггере переходных процессов приходит сигнал переноса P_i, который суммируется с содержимым триггера и записывает в него цифру i-го разряда получаемой суммы S_i. Единица переноса P_{i+1} образуется в соответствии с выражением 3.2. Когда $x_i = y_i = 1$, срабатывает схема И2, в остальных случаях – схема И1. Линия задержки (⊨) предназначена для того, чтобы код y_i поступил на схему И2 после окончания в триггере переходных процессов.

Итак, на функциональном уровне мы рассмотрели несколько схем сумматоров, реализованных на базе комбинационных схем и цифровых автоматов. Теперь перейдем к изучению некоторых схем счетчиков.

Счетчик – это типовой узел компьютера, предназначенный для подсчета числа входных сигналов. Они классифицируются по коэффициентам пересчета, по выполняемым арифметическим операциям и по типам цепей переноса. Счетчики состоят из последовательно включенных триггеров, управляемых по счетному входу. Выходной сигнал триггера предыдущего разряда поступает на счетный вход триггера последующего разряда. Он называется сигналом переноса и *образуется на прямом выходе триггера при его переходе из состояния 1 в состояние 0*. Сигнал переноса на инверсном выходе триггера образуется, соответственно, при его переходе из 0 в 1.

Суммирующий двоичный счетчик с последовательным переносом и временная диаграмма его работы представлены на рисунке 3.9:

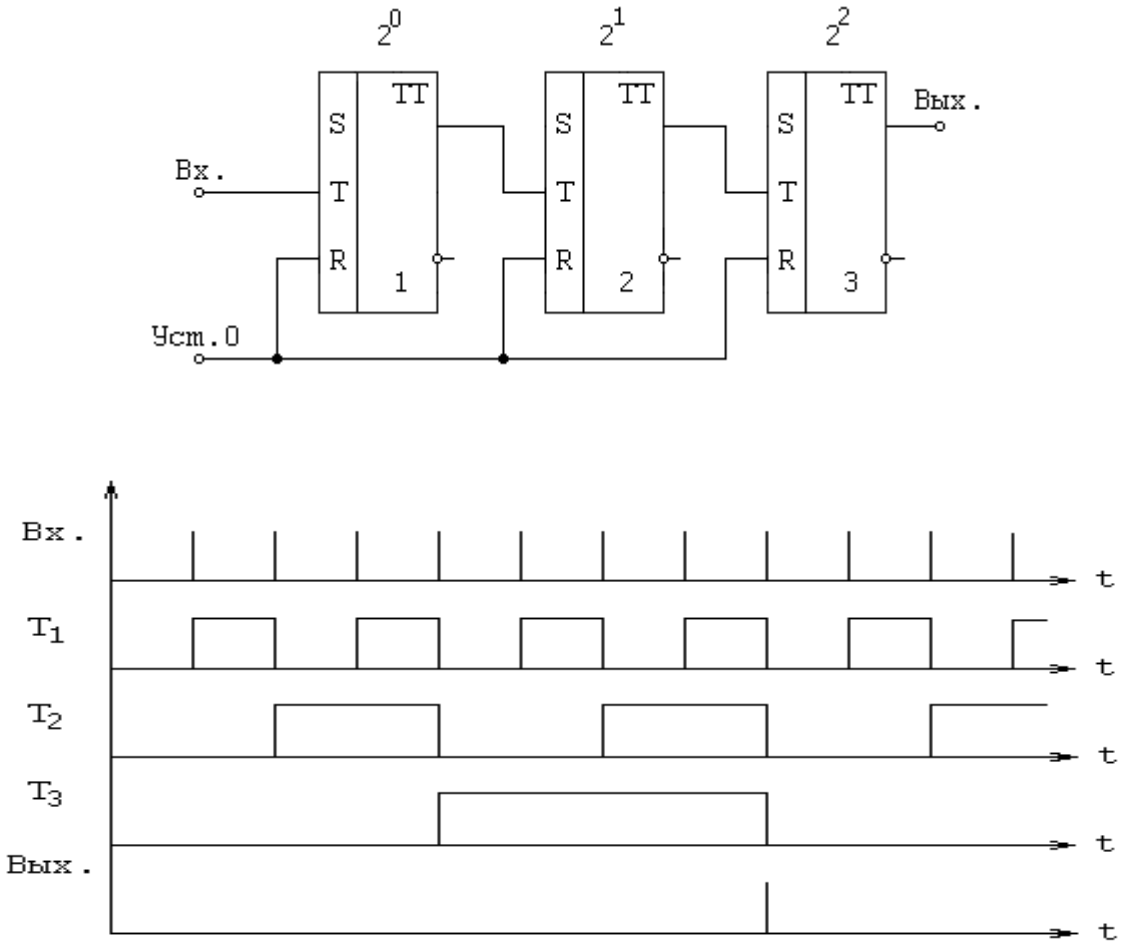


Рис. 3.9. Схема суммирующего счетчика
и временная диаграмма его работы

В исходное нулевое состояние такой счетчик устанавливается сигналом, поступающим на R-входы всех триггеров. Первый импульс, поступающий на вход схемы (на счетный вход триггера Т1), перебрасывает Т1 в состояние 1, при этом на его прямом выходе единицы переноса в следующий разряд не возникает, и триггер Т2 остается в нулевом состоянии. Второй входной импульс переключает Т1 в нулевое состояние, и сигнал переноса с его прямого выхода устанавливает Т2 в состояние 1. Третий входной импульс переведет Т1 в 1, после четвертого импульса Т1 и Т2 будут в нулевом, а Т3 – в единичном состоянии и т.д.

Таким образом, Т1 перебрасывается от каждого входного импульса, Т2 – от каждого второго, Т3 – от каждого четвертого импульса, и на выходе такого счетчика сигнал появится после прихода восьмого входного импульса.
Следовательно, рассмотренный счетчик имеет коэффициент пересчета, равный восьми. Состояния всех трех триггеров счетчика в зависимости от приходящих входных сигналов приведены в таблице 3.10:

Т а б л и ц а 3.10

Работа двоичного счетчика

Вход	Т1	Т2	Т3	Выход
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1

В общем случае на выходе *n*-разрядного счетчика сигнал будет после прохождения 2^n входных импульсов, а максимальное число, которое может хранить такой счетчик, равно $2^n - 1$. Отсюда ясно, как строятся счетчики с

коэффициентом пересчета, кратным 2^n: достаточно выбрать количество разрядов такого счетчика, равное показателю двойки. Например, счетчик с k = 16 должен содержать 4 разряда, с k = 32 – 5 разрядов и т.д. А как построить счетчик с любым коэффициентом пересчета, не кратным 2^n? Для этого надо руководствоваться следующим правилом.

Пусть необходимо реализовать схему счетчика с k = 10. Достаточно просто понять, что такая схема должна содержать 4 триггера. Далее замечаем, что $16 - 10 = 6_{10} = 110_2$. Полученное двоичное число показывает, что второй и третий триггеры искомой схемы охватываются обратными связями так, как это представлено на рисунке 3.10. Если бы мы строили счетчик, например, с k =21, то получили бы: $32 - 21 = 11_{10} = 1011_2$, то есть в пятиразрядном счетчике триггеры первого, второго и четвертого разрядов должны быть охвачены обратными связями.

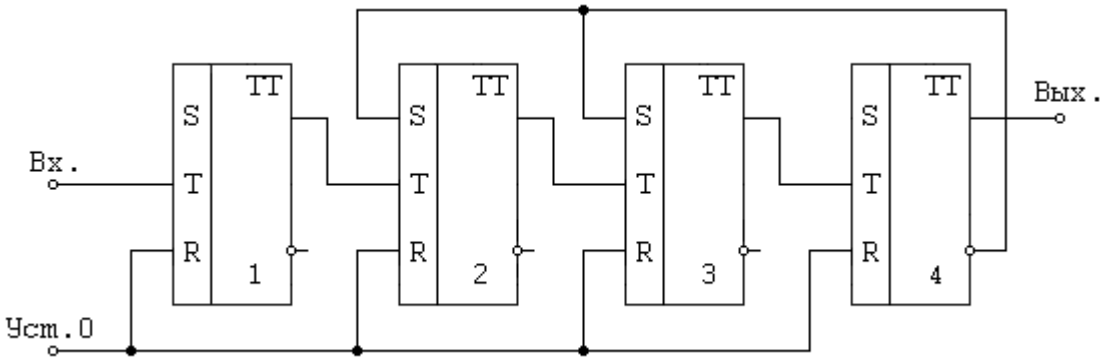


Рис. 3.10. Счетчик с k = 10

До седьмого входного импульса включительно такой счетчик работает в режиме обычного двоичного счетчика, то есть в нем будет записано число $111_2 = 7_{10}$, так как триггеры Т1, Т2 и Т3 будут в единичном состоянии, а Т4 – в нулевом. С приходом восьмого импульса Т1, Т2 и Т3 последовательно перебросятся в 0, а Т4 впервые перейдет из 0 в 1, поэтому единица переноса с его инверсного выхода затем перебросит Т2 и Т3 по их S-входам в единичное состояние, и в счетчике окажется число $1110_2 = 14_{10}$ (Т4 = Т3 = Т2 = 1, Т1= 0).

Девятый импульс переведет Т1 в 1, а после десятого импульса все триггеры будут в 0 с появлением сигнала на выходе схемы.

Мы также не будем рассматривать схемы счетчиков с различными цепями переноса, повышающими их быстродействие, так как это является предметом изучения в курсе «Схемотехника».

Контрольные вопросы

1. Что в алгебре логики понимается под высказыванием или утверждением?
2. Для чего используются логические связки?
3. Как устанавливается «старшинство» логических связок?
4. Что такое таблица истинности логической операции?
5. Сколько существует бинарных логических операций?
6. Каков алгоритм оценки истинности сложных высказываний?
7. Как можно задать любую булеву функцию?
8. Как получается формула логической функции по ее таблице истинности?
9. Как и для чего надо упрощать логические формулы?
10. В чем заключается неоднозначность результата минимизации логической функции?
11. Какие устройства двух типов преобразуют информацию в компьютере, в чем заключается различие между ними?
12. Что такое функционально полная система элементарных логических функций?
13. Что такое сумматор и как описывается его работа?
14. Что такое триггер и как он запоминает информацию?
15. Как образуются сигналы переноса между разрядами счетчика?
16. Как строятся счетчики с произвольным коэффициентом пересчета?

Словарь основных терминов

Автомат (греч. αὐτόματος – самодействующий) – абстрактная машина, определенная формальным способом, представляет собой математическую *модель* реально существующих или принципиально возможных систем, которые воспринимают, хранят и перерабатывают дискретную *информацию* в дискретном времени. В кибернетическом смысле автомат является *«черным ящиком»*, имеющим множества входов, выходов и внутренних состояний, в которые он скачкообразно переходит под воздействием входных сигналов.

Автоматов теория – раздел теоретической *кибернетики*, который изучает математические *модели* систем, осуществляющих преобразование *информации*. К ним относятся биологические системы, рассматриваемые с точки зрения восприятия и переработки информации, и технические – *компьютер* и его устройства, *программы*, системы управления и т.д.

Алгебра логики – раздел математической логики, в котором изучаются законы, выражаемые логическими формулами, построенными из логических переменных и символов логических связок, применяется также для формального описания *автоматов*.

Алгоритм (лат. algorithmi, от арабского имени узбекского математика IX века Аль-Хорезми) – система точно сформулированных правил, определяющая процесс преобразования за конечное число шагов входной *информации* в выходную за конечное число шагов.

Байт (англ. byte) – единица измерения *информации*, равная восьми *битам*, обычно соответствует одной литере. Используются также другие единицы измерения информации: 1 килобайт = 2^{10} байт; 1 мегабайт = 2^{20} байт; 1 гигабайт = 2^{30} байт; 1 терабайт = 2^{40} байт.

Бит (англ. bit – binary digit) – «двоичный разряд», принимающий одно из двух значений, нуль или единица; минимальное *количество информации*, содержащееся в одном двоичном разряде. Источник с двумя возможными сообщениями, *вероятность* каждого из которых равна ½, обладает *энтропией* в один бит.

Вероятностей теория – математическая дисциплина, изучающая закономерности случайных процессов. Ее центральным понятием является *вероятность*.

Вероятность – числовая характеристика степени возможности наступления при определенных условиях случайного события, т.е. отношение числа наступивших событий к общему числу всех возможных событий.

Вычислительная наука (англ. computer science) – аналог термина «информатика» в США и англоязычных странах.

Вычислительная техника – область техники, объединяющая средства автоматизации математических вычислений и обработки информации в различных областях человеческой деятельности, а также принципы и методы проектирования и функционирования этих средств.

Данные (англ. data) – представленные формализованным способом сведения или факты, допускающие их обработку вычислительными методами.

Дизъюнкция, логическое сложение, операция ИЛИ (лат. disjunctio – разобщение, обособление) – сложное высказывание, состоящее из двух простых, которое истинно, если, по крайней мере, одно из простых высказываний истинно.

Документалистика – направление в *кибернетике*, изучающее и оптимизирующее документальные системы.

Доступ (к информации) – возможность использования информации, хранящейся в ЭВМ или системе.

Запись – неоднородная упорядоченная статическая структура прямого доступа.

Импликация (лат. implicatio – сплетение) – сложное высказывание, состоящее из двух простых: посылки (условия) и заключения (следствия); импликация ложна только тогда, когда посылка истинна, а заключение ложно.

Информатизация общества – эволюционный процесс развития общества, при котором в результате внедрения новых компьютерных и телекоммуникационных технологий обеспечиваются оптимальные условия использования *информации* во всех сферах человеческой деятельности.

Информатика (франц. informatique = information + automatique) – наука о структуре и общих свойствах *информации*, методах ее сбора, хранения, поиска, преобразования, распространения и использования с помощью *компьютеров* в различных сферах человеческой деятельности.

Информации качество – совокупность потребительских показателей информации, определяющих возможность ее эффективного использования.

Информации количество – мера величины информации, содержащейся в одной случайной величине относительно другой; связана с понятием *энтропии сообщения*.

Информации обработка – операции сбора, хранения, поиска, преобразования, передачи и выдачи информации.

Информации передача – процесс ее распространения от источника к потребителю.

Информации передачи теория – часть *теории информации*, основы которой разработал К. Шеннон, решивший проблему нахождения максимально достижимой скорости *передачи информации* при сколь угодно малой *вероятности* ошибок, связав ее с *количеством информации*.

Информации теория – раздел *кибернетики*, занимающийся математическим описанием и оценкой методов передачи, хранения, получения и классификации информации, представляет собой совокупность

теорий, общими для которых являются методы теории вероятностей, что отражает присутствие в процессах случайных факторов, связанных с информацией. Ее составными частями являются *теория передачи информации,* теория кодирования и многие задачи математической статистики, в частности, статистики случайных процессов, распознавания образов, лингвистики и *информатики*.

Информационная модель – совокупность *информации*, описывающей свойства и состояния объекта, процесса или явления, а также их связь с внешним миром; различают знаковые и вербальные (лат. verbalis – устный) информационные модели. Часто словесная модель какого-либо процесса представляется в виде *алгоритма* с пронумерованными шагами, в котором четко выделены действия и объекты, над которыми они совершаются.

Информационная система – совокупность методов и средств ввода, обработки и вывода *информации*, а также персонала, используемых для принятия обоснованных решений в интересах достижения поставленной цели.

Информационная технология – совокупность методов и программно-технических средств, объединенных в технологическую цепочку, обеспечивающую сбор, хранение, обработку, передачу и отображение *информации*.

Информационно-поисковая система – совокупность алгоритмических, технических и языковых средств, предназначенных для ввода, хранения, поиска и выдачи *информации* по запросам пользователей.

Информация (лат. informatio – разъяснение, осведомление) – получаемые сведения об объектах и явлениях, которые уменьшают степень неполноты знаний о них.

Информация документальная – информация, закрепленная на каком-либо материальном носителе.

Искусственный интеллект (лат. intellectus – разум) – направление *информатики*, связанное с созданием аппаратно-программных средств и систем, пользователи-непрограммисты которых, общаясь с *компьютером* на ограниченном *естественном языке*, могут ставить и решать задачи творческого характера для разных предметных областей.

Кибернетика (греч. χυβερνητιχή τέχνη – искусство управления) – наука об управлении, получении, преобразовании и передаче *информации* в кибернетических системах, под которыми понимаются системы любой природы: административные, биологические, технические и др.

Кибернетика техническая – направление *кибернетики*, в котором изучаются технические системы управления. К нему относятся теория и практика автоматического регулирования и управления, а также решение задач комплексной автоматизации производства и различных сложных автоматизированных систем управления.

Коды машинные – прямой, обратный и дополнительный коды чисел, представленных в двоичной *системе счисления*; их использование позволяет свести выполнение четырех арифметических операций в *компьютере* к двум действиям – сложению и сдвигу.

Команда – элементарная инструкция, автоматически выполняемая *компьютером*. Набор команд образует *программу*.

Комбинационная схема – схема без *обратных связей*, построенная из элементов, являющихся *автоматами* без памяти,соединения между которыми выполнены по правилам, соответствующим в функциональном отношении операции суперпозиции (наложения) функций.

Компьютер (англ. computer – вычислитель, от лат. computo – считаю, вычисляю) – заимствованное из англоязычной литературы название *электронной вычислительной машины*, устройства алгоритмической обработки *информации*.

Компьютеров теория (англ. computer science – наука о компьютерах) – в англоязычных странах этот термин соответствует термину *информатика*.

Конъюнкция, логическое умножение, операция И (лат. conjunctio – связь) – сложное высказывание, состоящее из двух простых, которое истинно только тогда, когда оба простых высказывания истинны.

Лингвистика (франц. linguistique – языковедение, от лат. lingua – язык – языкознание, наука о *языке*.

Логическая схема – совокупность *логических элементов*, реализующая какую-либо булеву функцию.

Логический элемент – техническое устройство, осуществляющее элементарную логическую операцию над входными информационными сигналами.

Массив информационный (от франц. massif – плотный) – совокупность *информации*, хранящейся в определенном виде в запоминающем устройстве. Под массивом понимают набор однотипных по структуре и способу использования *данных*, *записей*, а также множество поисковых образов документов или записей фактов в *информационно-поисковой системе*.

Множество – одно из основных понятий математики, которое трактуется как неупорядоченная совокупность неповторяющихся объектов.

Моделирование – исследование объектов, процессов или явлений в разных предметных областях на основе построения и изучения их *моделей*.

Модель (франц. modèle – образец) – упрощенное представление о реальном объекте, процессе или явлении; искусственно созданный абстрактный или материальный аналог оригинала (прототипа).

Обратная связь – воздействие выходной величины некоторой системы на вход этой же системы или результатов функционирования системы на характер этого функционирования. Принцип обратной связи является одним из важнейших общих понятий *кибернетики* и теории автоматического управления.

Отрицание (инверсия, операция НЕ) – унарная операция, если простое высказывание ложно, то его отрицание истинно, и наоборот.

Очередь – однородная линейно упорядоченная динамическая структура *данных* последовательного доступа.

Поиск информационный – процесс отыскания в поисковом массиве таких записей, которые отвечают признакам, указанным в информационном запросе, и, следовательно, содержат искомую *информацию*.

Программа (греч. πρόγραμμα – объявление, распоряжение) – описание *алгоритма* решения задачи на языке *компьютера* в виде последовательности *команд*.

Разрядная сетка – совокупность двоичных разрядов, предназначенных для хранения и обработки чисел. В *компьютере* применяются две формы представления чисел: естественная – с фиксированной запятой (точкой) и нормальная или полулогарифмическая – с плавающей запятой (точкой).

Семиотика (от греч. σημεῖον – признак, знак) – наука о свойствах знаков и знаковых систем; первоначально этим термином обозначалась только медицинская наука о признаках болезней.

Система счисления – *язык* для представления чисел. Элементы, применяемые в *компьютере* для представления чисел, являются двухпозиционными, т.е. обладают двумя устойчивыми состояниями, одно из которых обозначается как 0, другое – как 1, поэтому числа в нем представляются в двоичной системе.

Сообщение – форма представления информации с помощью определенных знаков.

Список – упорядоченная структура данных, позволяющая представлять иерархическую связь.

Сумматор (от лат. summa – итог) – устройство, преобразующие информационные сигналы в сигнал суммы слагаемых.

Сумматор комбинационный – устройство, с помощью *комбинационных схем* преобразующее сигналы входных слагаемых в слово, числовое значение которого равно сумме числовых значений слагаемых.

Сумматор накапливающий – устройство, с помощью накапливающих схем преобразующее сигналы входных слагаемых в слово, числовое значение которого равно сумме числовых значений слагаемых; осуществляет не только суммирование, но и хранение результата.

Сумматор одноразрядный – устройство, обеспечивающее *сложение по модулю 2* цифр одного разряда слагаемых и цифры переноса из предыдущего младшего разряда, а также формирование единицы переноса в следующий старший разряд.

Счетчик – устройство, осуществляющее подсчет сигналов (импульсов), используется в *компьютере* для образования последовательности адресов *команд*, счета количества циклов выполнения операций и т.п.

Тезаурус (греч. Χησαυρός – клад, сокровищница) – совокупность накопленных человеком знаний, а также сведений, которыми располагает пользователь или система. В более узком смысле это словарь, отражающий смысловые связи между словами языка, и предназначенный для поиска слов по их смыслу.

Триггер (англ. trigger – защелка, пусковая схема) – *логическая схема* с *обратными связями*, обеспечивающими два ее устойчивых состояния, изменение которых происходит скачком под воздействием пускового импульса, подаваемого на один из входов.

Файл (англ. file – картотека, комплект) – то же, что и *массив информационный*.

«Черный ящик» - условное название системы, в которой внешнему наблюдателю доступны лишь входные и выходные величины, а ее внутреннее устройство и процессы, протекающие в ней, неизвестны. Это понятие

широко используется в *кибернетике* при решении задач идентификации и моделировании сложных объектов управления

Эвристика (от греч. εύρίσχω – обнаруживаю) – метод исследования, основанный на неформальных, интуитивных соображениях. Такой подход, основанный на догадках, вытекающих из общего опыта решения родственных задач, позволяет сокращать количество рассматриваемых вариантов, но не гарантирует получения наилучшего решения. В *кибернетике* широко применяются эвристические методы распознавания образов.

Экспертная система – интеллектуальная система, аккумулирующая формализованные опыт и знания экспертов в конкретных предметных областях для консультаций менее квалифицированных пользователей. Основными частями такой системы являются база знаний, ее редактор, блок логического вывода и интерфейс пользователя.

Электронная вычислительная машина, ЭВМ – то же, что и *компьютер*.

Энтропия (от греч. έν – в и τροπή – превращение) – количественная мера неопределенности ситуации вида $H = - \sum_{i=1}^{n} p_i \log_2 p_i$, где p_i – вероятности возможных событий, n – их количество.

Энтропия сообщения – функция, задающая минимально достижимое *количество информации*, содержащееся в передаваемом сигнале относительно случайной величины, в зависимости от заданной точности воспроизведения сигналом этой величины.

Энтропия языка – одна из основных статистических характеристик речи, измеряющая содержащуюся в ней *информацию*.

Язык естественный – язык общения, возникший естественным путем. Представляет собой совокупность алфавита, лексики и грамматики, последняя состоит из морфологии и синтаксиса.

Язык информационный – *искусственный язык*, предназначенный для записи семантической *информации* с целью ее последующего использования в *информационных системах*, обеспечивает однозначную запись информации и алгоритмическое распознавание (отождествление) по-разному записанных фактов с полнотой и точностью, которые отвечают требованиям, предъявляемым к системе.

Язык искусственный – специально созданная семиотическая система. Это понятие противопоставляется понятию «*язык естественный*», т.е. естественно возникший язык – русский, английский и т.д. Искусственные языки включают в себя универсальные языки, созданные для международного общения и являющиеся суррогатами естественных языков, и специализированные знаковые системы для записи *информации*, среди которых выделяются языки для автоматизированной переработки информации.

Литература

1. Акулов О. А., Медведев Н. В. Информатика: базовый курс. – М.: Омега-Л, 2004. – 552 с.
2. Глушков В. М. (ред.) Словарь по кибернетике. – Киев: Главная редакция УСЭ, 1979. – 624 с.: ил.
3. Информатика: Энциклопедический словарь для начинающих/Сост. Д. А. Поспелов. – М.: Педагогика-Пресс, 1994. – 352 с.: ил.
4. Информатика: Учебник/Под ред. Н. В. Макаровой. – 2-е изд. – М.: Финансы и статистика, 1998. – 768 с.: ил.
5 Таненбаум Э. Архитектура компьютера. – 4-е изд. – СПб.: Питер, 2002. – 704 с.: ил.
6. Хамахер К. и др. Организация ЭВМ. – 5-е изд. – СПб.: Питер; Киев: издательская группа BHV, 2003. – 848 с.: ил.
7. Яглом А. М., Яглом И. М. Вероятность и информация. – 3-е изд. – М.: Наука, 1973. – 511 с.