ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО РЫБОЛОВСТВУ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«МУРМАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

С.Б. Луковкин

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ

Допущено Учёным советом университета
в качестве учебного пособия по дисциплине «Информатика»
для студентов технических специальностей.

Мурманск
2008

УДК 004 (075.8)

ББК 32.81 я 73

Л 84

Луковкин, С.Б. Теоретические основы информатики: учеб. пособие / С.Б. Луковкин – Мурманск: Изд-во МГТУ, 2008. - 125 с.

Учебное пособие написано на основе курса лекций по дисциплине «Информатика», читаемого автором для студентов 1 курса МГТУ различных специальностей. Учебное пособие содержит теоретические основы информатики и полностью соответствует Государственному образовательному стандарту. Рассмотрены основные понятия информатики, вопросы измерения количества информации, история развития вычислительной техники, основы формальной логики, теории алгоритмов и базовые понятия теории кодирования. Изложение материала сопровождается примерами и задачами.

The manual is based on a lectures on discipline the "Computer science", readable by the author for students of first rate of MGTU of various specialties. The manual contains theoretical bases of computer science and completely corresponds to the State educational standard. The basic concepts of computer science, questions of measurement of quantity of the information, history of development of computer facilities, a basis of formal logic, the theory of algorithms and base concepts of the theory of coding are considered. The statement of a material is accompanied by examples and problems.

Ил. 9, список лит. – 14 названий.

Резензенты: В.Б. Илюшин, зам. директора по учебно-методической работе Егорьевского технологического института филиала ГОУ ВПО МГТУ «Станкин»;

В.И. Батищев, д.т.н., профессор, заведующий кафедрой информационных технологий Самарского гос. технического университета.

Луковкин Сергей Борисович
Теоретические основы информатики
Редактор
Корректор
Электронная вёрстка

ОГЛАВЛЕНИЕ

Введение. 4

1. Данные и информация. Информатика. 10

2. Количество информации. Формула Хартли. 22

3. Количество информации. Формула Шеннона. 30

4. Условная энтропия. 38

5. Кибернетика. 42

6. Алгоритмический, семантический и ценностный подходы к определению информации. 53

7. История развития вычислительной техники. 60

8. Основы формальной логики. 76

9. Системы счисления. 86

10 . Алгоритмы. Машина Тьюринга. 103

11. Основы теории кодирования. 118

Заключение. 124

Список литературы. 125

Введение.

Информатика не более наука о компьютерах,
чем астрономия — наука о телескопах.
Э́дсгер Ви́бе Де́йкстра (1930-2002)[1].

Информатика, как новая научная дисциплина, возникла во второй половине XX века, приблизительно в 60-е годы. Р.М. Юсупова и В.П. Котенко утверждают, что впервые термин информатика появился в названии статьи Ф.Е. Темникова в 1963 г. в журнале «Известия вузов», другие авторы считают, что термин имеет французское происхождение. Именно в эти годы в работах философов и социологов появляется термин «информационное общество», которым обозначили качественные изменения в социуме, вызванные появлением в обществе первых ЭВМ, разработкой информационных технологий и их использованием в процессах производства.

Всё больший вклад в экономику развитых стран со второй половины XX века приходится на деятельность, связанную с обработкой информации и оказанием услуг. В обществе постоянно нарастают потоки информации, циркулирующие по различным каналам связи. С. Лем сравнил лавинообразное распространение информации в современном мире со взрывом информационной «мегабитовой» бомбы. Процесс информатизации охватил практически все страны мира. При этом темпы модернизации в сфере информации только нарастают, зачастую вызывая у людей то, что Э. Тоффлер назвал «шок от будущего».

---

[1] выдающийся нидерландский учёный, идеи которого оказали огромное влияние на развитие компьютерной индустрии; автор концепции структурного программирования, разработчик языка Алгол, лауреат премии Тьюринга.

Современное общество столкнулось с ранее неизвестной в таких масштабах проблемой обработки, хранения, передачи и использования информации. Информационная сфера становится своеобразным социальным аттрактором, притягивая к себе всё большее количество людей. Информатизация охватила практически все важнейшие направления практической жизни человека: транспорт, связь, медицину, экономику, образование – везде мы обнаруживаем влияние информатизации. И, конечно, Интернет – современная система связи, некоторое подобие нервной системы человека для ноосферы, охватившая весь мир.

Современному специалисту в любой сфере деятельности неизбежно придётся взаимодействовать с различными и часто меняющимися средствами информационных технологий. Информатика - единственная дисциплина, изучение которой хотя бы в какой-то мере способствует подготовке студентов к дальнейшей жизни в информационном обществе. Причём, такая подготовка нужна студентам самых различных специальностей. В связи с этим возникает важный вопрос: «Чему и как надо учить студентов в процессе освоения курса информатики?».

Ещё 8-10 лет назад в нашей стране практически отсутствовали учебники по данному предмету. Сложность в создании учебной литературы по информатике состоит в том, что мы имеем дело с самой молодой наукой среди других естественных и технических наук. Для сравнения можно привести время становления дифференциального исчисления во времена Ньютона и Лейбница – тогда, наверное, было также трудно написать учебник по дифференциальному и интегральному исчислению как сейчас по информатике. Другой пример: при изучении курса математики студенты осваивают результаты, полученные учёными 100 – 200 лет назад, а при изучении вопросов прикладной и технической

информатики осваиваются достижения, полученные либо совсем недавно, либо не позже чем 20 - 30 лет назад.

Содержание курса по информатике постоянно изменяется на всём протяжении своего недолгого существования. Охватить все существующие направления в информатике невозможно. Сама информатика на данном этапе предстаёт как дисциплина, в которой можно выделить самостоятельные подразделы: фундаментальная (теоретическая), прикладная и техническая информатика.

К технической информатике можно отнести задачи разработки сетевого коммуникационного оборудования, вопросы передачи и кодирование информации, проблемы организации вычислительных сетей, устранения помех при передаче данных и т.п. К прикладному направлению можно отнести разработку компьютеров, создание программ, разработку и использование языков программирования. Фундаментальная (теоретическая) информатика изучает вопросы возникновения информации, её эволюцию, свойства информации, те основы информатики, которые уже не подвергаются столь быстрым изменениям, как её прикладной и технический разделы.

Например, существует важное практическое направление в информатике, связанное с освоением уже существующего ПО: средства обработки текстов, изображений, электронных таблиц, всё то, что в первую очередь интересует конечных пользователей. Очевидно, что это наиболее быстро меняющийся аспект информатики. Такое же замечание можно сделать и о том направлении в информатике, которое изучает аппаратное устройство вычислительной техники, устройства памяти и печати, архитектуру процессоров, средства связи, внешние устройства компьютеров и т.п. Скорость модификации уже существующих и возникновения новых устройств в этой области такова, что оперативно отразить эти изменения в рамках учебного процесса не представляется

возможным. Однако и в этой области существуют уже устоявшиеся результаты, мало подверженные изменениям и ставшие уже «классическими»: принципы Дж. фон Неймана организации ЭВМ, принцип работы устройства управления процессора.

Наряду с уже упомянутыми направлениями в информатике появляются новые, ранее неизвестные прикладные проблемы, связанные с информационными процессами в обществе: защита информации от несанкционированного доступа, условия свободного обращения информации в обществе, развитие электронного правительства, контроль за контентом ресурсов в Интернете и т.п. Уже почти устоялись термины социальная информатика и биоинформатика, а в философии активно разрабатываются проблемы виртуальной реальности и искусственного интеллекта.

Изложенный в учебном пособии материал призван сформировать у студентов понимание того, что появление ЭВМ в Пенсильванском университет США было неслучайным. Что и это событие, и дальнейшее развитие информатики и вычислительной техники имеют в качестве своего основания результаты работы многих поколений учёных прошлого, таких как Блез Паскаль, Готфрид Лейбниц, Чарльз Бэббидж, Ада Лавлейс, Джордж Буль, Конрад Цузе.

Огромное влияние на развитие информатики оказывала и продолжает оказывать математика. Одними из первых программистов в нашей стране были выпускники факультетов математики, физики, автоматики, электроники. Следует отметить наличие корреляции между важными историческими этапами в развитии цивилизации и появлением новых направлений в математике. В истории принято выделять аграрный период, - до XVII в., индустриальный с XVII в. по XX в., и информационный, отсчёт которого ведётся с середины XX в. Этим периодам развития соответствуют свои технологии, своя картина мира и своя «математика».

Аграрному периоду соответствовали сельскохозяйственное производство, материальная картина мира, элементарная математика (арифметика), которая включала элементы алгебры, планиметрию, стереометрию, тригонометрию. Основным понятием математики того времени было понятие числа.

Индустриальный период связан с изучением динамических систем: в это время изучалось движение небесных тел и движение тел на Земле, развивалось мореплавание; технологии того времени базировались на преобразовании энергии из одного вида в другой: например, энергия пара и ветра преобразовывалась в механическую энергию движения. Развитие математики того времени определялось работами Декарта, Ньютона, Лейбница. Это был период развития высшей математики: анализа бесконечно малых, теории функций действительного переменного, аналитической и дифференциальной геометрии. Доминирующее понятие математики индустриального периода - понятие функции.

Для информационного периода характерна информационная картина мира; вместо динамических систем стали доминировать их модели, в центр внимания становятся информационные системы: сначала кибернетические, а затем и интеллектуальные системы, например, экспертные системы. Классической математики уже недостаточно для моделирования информационных и кибернетических систем, т.к. её методы - дифференциальное и интегральное исчисление, были предназначены для изучения систем в пространственно – временной среде. Математика информационного периода – это дискретная математика; она предназначена для описания поведения систем в языковой среде. Здесь используются алгебраические и топологические методы. На первый план выходят множества и отношения, алгебра, топология, математическая логика. Дискретная математика не отрицает классическую высшую математику, но включает её.

Информатика в вузе – общеобразовательный предмет, на изучение которого отводится всего два семестра. Содержание образования по информатике практически одинаково для всех специальностей, за исключением тех, которые непосредственно связаны с изучением программирования, прикладной математики и вычислительной техники. В данном учебном пособии рассматриваются теоретические аспекты информатики, которые уже стали в некотором смысле «классическими»: они излагаются для студентов любых специальностей. К этим аспектам мы относим определение основных понятий информатики и её предмета, подходы Хартли и Шеннона в определении количества информации, связь между кибернетикой и информатикой, историю развития вычислительной техники, основы формальной логики, системы счисления, основы теории алгоритмов и теории кодирования, с изложением классических результатов, полученных Хэммингом.

В основу данного учебного пособия положен курс лекций по информатике, читаемый автором для студентов биологического, естественно-технического и политехнического факультетов МГТУ на протяжении последних восьми лет.

1. Данные и информация. Информатика.

В повседневной жизни мы часто встречаемся с такими понятиями как информация, информатика, информационные технологии и т.п. Этими понятиями пользуются учёные, дикторы ТВ, журналисты и политики. Однако, до настоящего времени не существует общепринятого определения понятия «информация»: многочисленные исследователи предлагают самые различные определения. Составители словарно-энциклопедических изданий фактически были вынуждены признать неразрешимость данной проблемы, и поэтому отказались от попыток дать единое определение информации. В одной словарной статье можно найти перечень сразу нескольких понятий информации.

Попытки связать информацию с привычными понятиями материя или энергия успехом не увенчались. Известно знаменитое отрицательное определение Винера: «информация есть информация, а не материя и не энергия». Из этого определения следует лишь один вывод: по своей значимости понятие информации не уступает таким основным физическим понятиям как материя или энергия.

Использование понятия «информация» в повседневной практике не вызывает у нас особых затруднений. Говоря об информации, мы обычно подразумеваем разъяснение, сообщение, изложение, какие-либо сведения, данные, объявление. В обычном, «житейском» смысле информация - это сумма сведений, которую получает некоторый субъект, человек, группа людей или животных об окружающем мире, о самом себе, о другом субъекте или изучаемом явлении. Используя эти сведения, человек может прогнозировать результаты своих действий, выбирать различные способы для достижения поставленных целей.

В СЭС приводится следующее определение информации: 1) информация – это сведения, передаваемые людьми устным, письменным

или каким-либо другим способом (с помощью условных знаков, сигналов, технических средств и т.п.); 2) с середины XX века информация – это обмен сведениями между людьми, человеком и автоматом, автоматом и автоматом, обмен сигналами в живом и растительном мире, передача признаков от клетки к клетке, от организма к организму.

Известно ещё одно распространённое определение: информация – это сведения, уменьшающие неопределённость нашего знания об окружающем нас мире, которые являются объектом хранения, преобразования, передачи и использования.

В своей книге «Синергетика и информация» Д.С. Чернавский приводит обширную коллекцию неудовлетворительных, по его мнению, тавтологических определений информации. Большое число похожих и непохожих друг на друга определений понятия «информация» означает, что общепринятого определения информации ещё нет. Более того, отмечает Д.С. Чернавский, нет даже четкого понимания сути этого явления, хотя потребность в нем давно уже назрела.

Рассуждая об информации, Н.Н. Моисеев приходит к выводу, что являясь центральным понятием в информатике, оно до сих пор не имеет чёткого определения. Н.Н. Моисеев утверждает, что информация не является всеобщим свойством материи и считает, что необходимость понятия информации возникает лишь при изучении систем, обладающих целеполаганием.

Существует подход, в котором вводится понятие информации как отраженного разнообразия. Источником разнообразия, по мнению В.М. Глушкова, является неоднородность распределения материи и энергии в пространстве и во времени. Отсюда и определение, данное В.М. Глушковым: информация – это мера неоднородности распределения материи и энергии в пространстве и во времени, показатель изменений, которыми сопровождаются все происходящие в мире процессы.

Трудность в построении общего определения информации состоит в том, что существуют разные типы информации. Например, социальная информация, информация биологическая, информация в экономике, научная информация. В самом простом случае мы говорим об информации, которая введена в компьютер для решения задачи, или об информации, передаваемой по проводам и радиоканалам. В этом случае можно определить количество информации, указать носитель информации, память, оценить качество информации. Отметим, что здесь мы имеем дело скорее с данными, чем с информацией. В общем случае, когда мы говорим об информации при изучении окружающего мира, возникают только вопросы, на которые пока нет ответов.

Другое важное понятие информатики - **данные**. Этот термин встречается не менее часто, чем информация и также является основным в информатике, но не вызывает таких затруднений при определении. Есть несколько различных по форме, но эквивалентных по сути определений того, что такое «данные». Наиболее часто встречаются следующие определения:

1. Данные - это зарегистрированные сигналы.
2. Данные - это информация, представленная в виде, позволяющем запоминать, хранить, передавать или обрабатывать её с помощью технических средств.
3. Данные – это информация об объекте или отношениях объектов, выраженная в знаковой форме.

Первое определение, на наш взгляд, наиболее удачное и наиболее общее. Под сигналом здесь понимается условный знак, физический процесс, явление, несущие сообщение о каком-либо событии, состоянии объекта и режиме его работы или передающие команды управления, оповещения. Сигнал – это изменяющийся во времени физический процесс. К регистрации сигналов можно отнести: запись музыки на магнитофон,

запись лекции в тетрадь, запись наблюдений в ходе эксперимента в виде чисел, графиков, фотографирование каких-либо объектов, запоминание учеником материала на уроке, нарисованный план, запись данных в память компьютера, на жёсткий диск и т.д.

Второе и третье определения понятия «данные» являются неудачными, так как в них делается попытка определить данные через информацию. Получается circulus vitiosus - порочный круг. Второе определение сужает общность понятия «данные» до уровня данных, используемых в технике. Третье определение также носит прикладной характер и имеет отношение к базам данных.

Понятия «данные» и «информация» близки, но не тождественны. Эти понятия часто смешивают и, как отмечалось выше, делаются попытки определить одно через другое. Данные и информация взаимосвязаны. Информация не может существовать без данных, без какого-либо носителя: она как-то должна быть представлена с помощью данных. Именно этот факт и пытались подчеркнуть авторы второго определения данных. С другой стороны, любые данные всегда несут в себе какую-то информацию.

Пример: мы слушаем иностранную речь по телефону, но не понимаем её. Идёт регистрация данных (запоминание сигнала), но нет процесса получения информации. Если мы запишем это сообщение на магнитофон (зарегистрируем сигнал) и передадим запись переводчику, то он сможет передать нам содержание телефонного сообщения, и мы получим информацию, которая в нём содержалась. С другой стороны, даже не понимая иностранную речь, мы можем получить информацию о том, кто нам звонил – мужчина или женщина, в каком состоянии находился звонивший человек, а в некоторых случаях мы сможем определить язык, на котором он говорил.

Любой процесс передачи данных (информации) может быть описан с помощью следующей схемы:

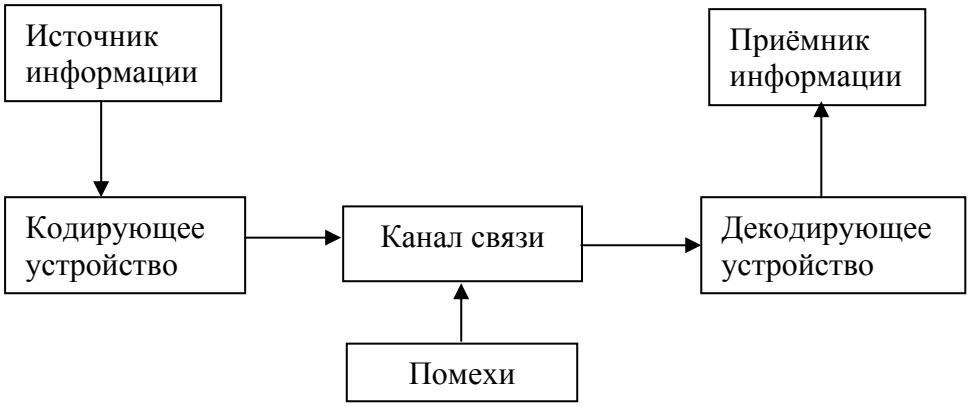


Рис. 1 Общая схема передачи информации (данных).

В качестве примера реализации данной схемы можно привести процесс обмена сообщениями по телефону. Источник информации – один из абонентов. Кодирующее устройство – микрофон. Он преобразует звуковые колебания в электрические сигналы, которые затем передаются по телефонному кабелю. Кабель в данном случае это и есть канал связи.

Иногда канал связи называют средой передачи данных. Во время передачи сигналов по каналу связи могут возникать различного рода помехи, которые искажают передаваемый сигнал. В этом случае мы слышим в телефонной трубке посторонние шумы, потрескивания и т.п. Декодирующее устройство – динамик телефонной трубки. Он совершает обратное преобразование электрических сигналов, полученных по кабелю, в акустические колебания, которые слышит другой абонент – приёмник информации.

Другой пример: преподаватель читает лекцию в аудитории для студентов, во время которой происходит передача информации. Каналами связи в этом случае являются воздух и доска, на которой во время лекции преподаватель мелом делает пояснительные записи. Помеха - шум в аудитории, события, отвлекающие внимание слушателей, плохое качество

доски или мела. Источник информации - преподаватель, его знания; кодирующие устройства – голосовые связки, язык, мел. Информационные сигналы по каналам связи поступают в органы зрения и слуха, где воспринимаются и фиксируются слушателями, декодируются и запоминаются ими.

Данные могут восприниматься человеком или техническим устройством; их можно переводить из одной знаковой системы в другую без потери содержащейся в них информации. Для выделения информации из данных нужно применить к ним методы обработки, «адекватные» этим данным. Такое получение информации из данных, после применения адекватных методов обработки, называется «информационными технологиями».

Технология – совокупность методов обработки, изготовления, изменения состояния, свойств, формы сырья, материала или полуфабриката в процессе материального производства. Или: технология - алгоритм целенаправленного воздействия на сырьё, материалы или полуфабрикаты соответствующими орудиями производства. Например, технология в металлургии, строительстве, производстве одежды и т.п. Технологический процесс в сфере материального производства можно представить в виде схемы:

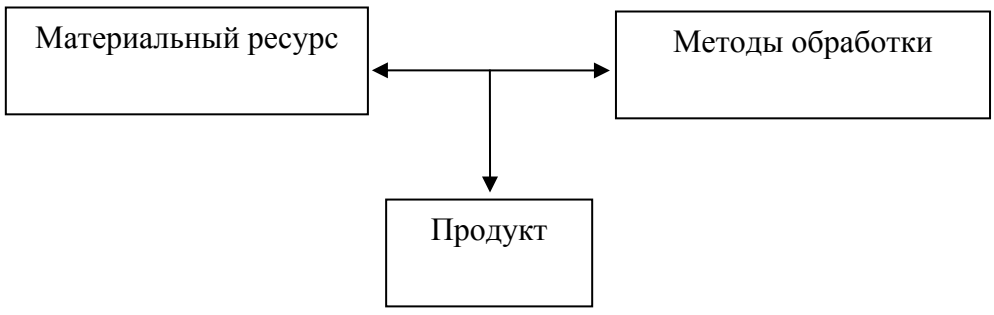


Рис. 2. Общая схема технологии материального производства.

Отличие информационных технологий от материальных заключается в том, что роль материального ресурса играют данные, а технологический процесс сводится к выбору адекватных методов обработки этих данных с применением, как правило, средств вычислительной техники. В результате мы получаем информацию, представленную в свою очередь в виде некоторых, уже новых данных, к которым опять могут быть применены другие адекватные методы обработки, получена новая информация и т.д.

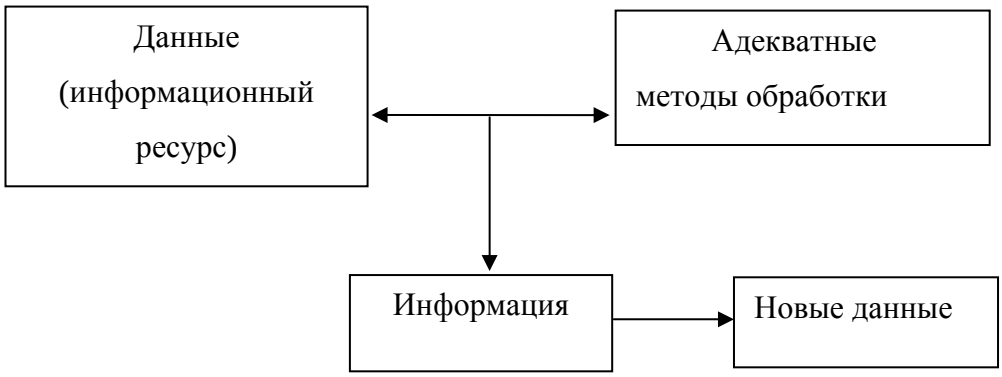


Рис. 3. Общая схема информационных технологий.

В качестве примера ИТ можно привести процесс обработки на компьютере данных сейсмической разведки. Результатом такой обработки является, например: информация о наличии нефтяных залежей, их расположении и размерах; данные о строении Земли.

Другие примеры ИТ: в результате многолетних метеорологических наблюдений получены данные о температуре воздуха. Эти данные представлены в виде таблиц чисел. Обработка таких данных позволяет сделать прогноз о возможном изменении температуры, климата и т.п. Обработка результатов ЕГЭ, проведённого в школах страны, позволяет получить информацию об уровне знаний и преподавания в отдельных регионах и в стране в целом. Существенным компонентом ИТ является компьютер (ЭВМ).

Итак, **информационные технологии** – машинизированные способы обработки, хранения, передачи и использования информации. Два основных элемента ИТ – это человек и ЭВМ.

Основное отличие ИТ от обычных технологий состоит в том, что применение материальных технологий изменяет окружающий нас материальный мир. А результатом применения ИТ является информация, которая оказывает воздействие на сознание людей и побуждает их к действию. Это свойство ИТ активно используется в средствах массовой информации. Следует отметить, что воздействуя на сознание людей, ИТ опосредованно влияют и на окружающий нас мир.

Термин **информатика** появился в середине 60-х годов XX века практически одновременно во Франции и в нашей стране. Он использовался для обозначения самой молодой науки среди других естественных и технических наук. Как отмечено в книге Р.М. Юсупова и В.П. Котенко, в 1963 г. в журнале «Известия вузов» была опубликована статья Ф.Е. Темникова «Информатика». В ней была представлена наука об информации как совокупность трёх разделов: теории информационных элементов, теории информационных систем и теории информационных процессов. Однако, эта статья осталась незамеченной, и более популярным оказалось французское толкование термина «informatique», которым обозначили науку об электронно-вычислительных машинах (ЭВМ) и их применении. В США вместо термина информатика используют термин «computer science».

Попытки дать определение информатики не прекращаются до настоящего времени. Как и в случае с понятием информация, существуют десятки различных определений того, что такое информатика. Вот некоторые из наиболее известных определений.

Информатика – наука, изучающая структуру и общие свойства информации, а также вопросы, связанные с **её** сбором, хранением,

поиском, преобразованием, распространением и использованием в различных сферах человеческой деятельности.

В своём учебнике «Информатика» С.В. Симонович приводит такое определение: «**Информатика** – техническая наука, систематизирующая приёмы создания, хранения, воспроизведения, обработки и передачи данных средствами вычислительной техники, а также принципы функционирования этих средств и методы управления ими». В этом определении акцент делается на понятии «данные», при этом понятие информация вообще отсутствует. Здесь происходит сужение информатики до уровня прикладной, технической науки. Это то, что называют в США computer scince.

Французская академия наук предлагает следующее определение: «**Информатика** – это наука об осуществляемой преимущественно с помощью автоматических средств целесообразной обработке информации, рассматриваемой как представление знаний и сообщений в технических, экономических и социальных областях».

Определение А.П. Ершова: «**Информатика** – это фундаментальная естественная наука, изучающая процессы передачи и обработки информации».

Д.С. Чернавский даёт следующее определение информатики: «**Информатика** - наука о процессах передачи, возникновения, рецепции, хранения и обработки информации». Он предлагает выделять в информатике три направления: техническое, прикладное и фундаментальное. В техническом аспекте информатика включает всебя передачу, кодирование и приём информации. В прикладном аспекте информатика занимается разработкой компьютеров, созданием программ (computer science). Фундаментальный аспект информатики включает в себя изучение процессов возникновения, эволюции, извлечения и реализации ценной информации.

Ценность информации связана с целеполаганием и зависит от того, насколько данная информация способствует достижению поставленной цели. Подход Д.С. Чернавского будет рассмотрен подробнее в отдельном параграфе.

Анализируя предлагаемые определения информатики можно сделать вывод, что информации присущи четыре основных вида «движения»: восприятие, хранение, передача и обработка.

В вопросе о том, является ли информация всеобщим атрибутом материи или нет, ученые делятся на две основные группы: атрибутивистов и функционалистов. Я придерживаюсь позиции функционалистов, т.е. полагаю, что информация присуща только живой природе. Так как она возникает там, где есть момент целеполагания - в системах способных к самоуправлению и самоорганизации. Основным аргументом в пользу функционалистов является то, что в неживой природе невозможен свободный выбор одного из нескольких равноправных состояний, при котором в системе и происходит генерация информации. Кроме того, в неживой природе отсутствует процесс обработки информации.

Информация обладает рядом свойств. Основными свойствами являются следующие:

1. Информация невоспроизводима.
2. Информация эмерджентна (от английского “emergency”).
3. Информация операциональна: информация побуждает к действию.
4. Объективность (зависит от методов получения информации).
5. Полнота информации зависит от достаточности данных для принятия решения или создания новых данных на основе уже имеющихся (это, скорее, свойство данных).
6. Достоверность (зависит от уровня шумов в регистрируемых сигналах и от точности, с которой происходит регистрация сигналов датчиками).

7. Адекватность – а) способность информации однозначно соответствовать отображаемому объекту; б) степень соответствия реальному, объективному состоянию дел.
8. Доступность - возможность получить нужную информацию. Степень доступности зависит как от доступности данных, так и от доступности адекватных методов их интерпретации.
9. Актуальность - соответствие информации данному моменту времени. Нередко с актуальностью информации связывают коммерческую ценность информации.
10. Коммерческая ценность – возможность получения дополнительной прибыли или возможность уклонения от убытков, благодаря использованию информации.

Первые три свойства редко упоминаются в учебниках по информатике, хотя они являются наиболее характерными для феномена информации. Невоспроизводимость означает, что при повторном приёме сообщения, несущего информацию, вы не получаете дополнительной информации. Например, вы ожидаете отправления своего поезда на вокзале и слышите объявление, в котором сообщается, что отправление задерживается на два часа. Если вы повторно прослушаете это объявление через некоторое время, вы не получите никакой новой информации относительно отправления поезда. Или вы ждёте объявления результата матча по футболу между двумя командами. Информация о победе вашей команды невоспроизводима в том смысле, что от повтора такого сообщения вы ничего нового уже не узнаете.

Свойство эмерджентности (emergency) информации означает, что информация обладает свойством неожиданности, внезапности. Сообщение о некотором событии несёт тем больше информации, чем меньше возможность наступления этого события. Например, сообщение о том, что 12 июля в г. Мурманске ожидается температура -10° несёт больше

информации, чем сообщение о том, что температура будет $+10^{\circ}$. Сообщение об отправке поезда по расписанию, несёт меньше информации, чем сообщение о задержке отправления (при условии, что обычно поезда отправляются по расписанию). Если некоторое сообщение лишено свойства эмерджентности, то такое сообщение не является информацией.

Свойство операциональности информации состоит в том, что информация побуждает нас к действию. Сообщение о задержке отправления поезда приводит к тому, что вы начинаете действовать незапланированно: идёте в привокзальный киоск, делаете звонки по телефону, сообщая о задержке, и т.п. Сообщение о падении курса акций на бирже приводит к попытке их продажи. Известие о возможном дожде заставляет вас взять зонт при выходе из дома.

Ещё одно важное свойство информации заключается в том, что когда вы делитесь с кем-то информацией, то у вас количество информации не уменьшается. Можно сказать так: при передаче информация увеличивается в объёме, её становится больше. В этом её отличие от других объектов окружающего мира. Например, если я отдам кому-то 100 рублей, то они появятся у другого, а у меня их уже не будет; если я поделился с кем-то знанием, информацией, то у меня останется столько же информации, но она появится ещё и у того, кому я её передал. При копировании информации с помощью средств ВТ мы получаем копию, которая ничем не отличается от оригинала, причём затраты на изготовление копии практически равны нулю. Информация начинает стремительно распространяться там, где в ней есть потребность. Она стремится занять максимально возможный объём в окружающем мире. Поэтому так нерезультативна борьба с так называемым «пиратством». Эта борьба по своей эффективности напоминает попытки преградить путь воде во время разлива рек.

2. Количество информации. Формула Хартли.

При изучении различных явлений и объектов окружающего мира люди стремились связать с этими объектами число, ввести их количественную меру. Люди научились измерять расстояния, взвешивать различные предметы, вычислять площади фигур и объёмы тел. Научившись измерять время, его длительность, мы до сих пор пытаемся понять его природу. Термометр был придуман за много лет до того, как учёные поняли, что он измеряет: с момента появления первого термометра до создания термодинамики прошло примерно три столетия. Количественное изучение некоторого явления, объекта может опережать его качественное изучение, процесс формирования соответствующего понятия может следовать за количественным изучением.

Похожая ситуация сложилась и в отношении информации. Р. Хартли в 1928, а затем К. Шеннон в 1948 предложили формулы для вычисления количества информации, однако на вопрос о том, что такое информация, они так и не ответили. В теории связи информация выступает в виде различных сообщений: например, букв или цифр, как в телеграфии, или в виде непрерывной функции времени, как при телефонии или радиовещании. В любом из указанных примеров, в конечном итоге, задача состоит в передаче смыслового содержания человеческой речи. В свою очередь, человеческая речь может быть представлена в звуковых колебаниях или в письменном изложении. Это ещё одно из свойств этого вида информации: способность представлять одно и то же смысловое содержание в различном физическом виде. Впервые на это обратил особое внимание У. Эшби. Представление информации в различном физическом виде называется кодированием. Для того, чтобы общаться с другими людьми, человеку приходится постоянно заниматься кодированием, перекодированием и декодированием. Очевидно, что по каналам связи

информация может передаваться в самых различных системах кодирования.

Р. Хартли первым ввел в теорию передачи информации методологию «измерения количества информации». При этом Р. Хартли считал, что информация, которую он собирался измерять, это «… группа физических символов – слов, точек, тире и т. п., имеющих по общему соглашению известный смысл для корреспондирующих сторон». Таким образом, Хартли ставил перед собой задачу ввести какую-то меру для измерения кодированной информации.

Пусть передаётся последовательность из n символов $a_1a_2a_3…a_n$, каждый из которых принадлежит алфавиту A_m, содержащему m символов. Чему равно число K различных вариантов таких последовательностей? Если n = 1 (передаётся один символ), то K = m; если n=2 (передаётся последовательность из 2-х символов), то $K = m*m = m^2$; в общем случае для последовательности из n символов получим

$$K = \underbrace{m * m * m ... * m}_{n-раз} = m^n$$

Количество информации, содержащееся в такой последовательности, Хартли предложил вычислять как логарифм числа K по основанию 2:

$$I = \mathrm{Log}_2 K, \qquad (2.1)$$

где $K = m^n$.

То есть, количество информации, содержащееся в последовательности из n символов из алфавита A_m, в соответствии с формулой Хартли равно

$$I = \mathrm{Log}_2(m^n) = n\,\mathrm{Log}_2 m\ . \qquad (2.2)$$

Замечание 1. Хартли предполагал, что все символы алфавита A_m могут с равной вероятностью (частотой) встретиться в любом месте сообщения. Это условие нарушается для алфавитов естественных языков: например, не все буквы русского алфавита встречаются в тексте с одинаковой частотой.

Замечание 2. Любое сообщение длины n в алфавите A_m будет содержать одинаковое количество информации. Например, в алфавите {0; 1} сообщения 00111, 11001 и 10101 содержат одинаковое количество информации. Это означает, что при вычислении количества информации, содержащегося в сообщении, мы отвлекаемся от его смыслового содержания. «Осмысленное» сообщение и сообщение, полученное из него произвольной перестановкой символов, будут содержать одинаковое количество информации.

Пример. В телеграфном сообщении используются два символа – точка (.) и тире (-), т.е. алфавит состоит из m = 2 символов. Тогда при передаче одного символа (n =1) количество информации $I = Log_2 2 = 1$. Это количество было принято за единицу измерения количества информации и называется 1 бит (от английского **binary unit** = **bit**) . Если телеграфное сообщение в алфавите {. ; -} содержит n символов, то количество информации $I = n\ Log_2 2 = n$ (бит).

С помощью символов 0 и 1 кодируется информация в компьютере и при передаче в вычислительных сетях, т.е. алфавит состоит из двух символов {0 ; 1}; один символ и в этом случае содержит $I = Log_2 2 = 1$ бит информации, поэтому сообщение длиной n символов в алфавите {0 ; 1} в соответствии с формулой Хартли (2.2) будет содержать n бит информации.

Если рассматривать передачу сообщений в алфавите русского языка, состоящего из 33 букв, то количество информации, содержащееся в сообщении из n символов, вычисленное по формуле Хартли, равно $I = n*Log_2 33 \approx n*\ 5.0444$ бит. Английский алфавит содержит 26 букв, один символ содержит $Log_2\ 26 \approx 4.7$ бит, поэтому сообщение из n символов, вычисленное по формуле Хартли, содержит $n*\ Log_2\ 26 \approx 4.7\ *n$ бит информации. Однако, этот результат не является правильным, так как не все буквы встречаются в тексте с одинаковой частотой. Кроме того, к

буквам алфавита надо добавить разделительные знаки: пробел, точку, запятую и др.

Формула (2.1) внешне напоминает формулу Больцмана для вычисления энтропии системы с N равновероятными микросостояниями:

$$S= - k*Ln(W), \quad (2.3)$$

где k - постоянная Больцмана = $1,38*10^{-23}$, а W- вероятность спонтанного принятия одного из микросостояний системы в единицу времени $t = 10^{-13}$ сек., W = 1/N, т.е.

$$S= -k*Ln(1/N) = k*Ln(N), \quad (2.4)$$

что полностью согласуется с формулой (2.1) за исключением множителя k и основания логарифма. Из-за этого внешнего сходства величину Log_2K в теории информации также называют энтропией и обозначают символом H. **Информационная энтропия** – это мера неопределённости состояния некоторой случайной величины (физической системы) с конечным или счётным числом состояний. **Случайная величина(**с.в.**)** – это величина, которая в результате эксперимента или наблюдения принимает числовое значение, заранее неизвестно какое.

Итак, пусть X – случайная величина, которая может принимать N различных значений $x_1, x_2, \ldots x_N$; если все значения с.в. X равновероятны, то энтропия (мера неопределённости) величины X равна:

$$H(X) = Log_2 N. \quad (2.5)$$

Замечание. Если случайная величина (система) может находиться только в одном состоянии (N=1), то её энтропия равна 0. Фактически это уже не случайная величина. Неопределённость системы тем выше, чем больше число её возможных равновероятных состояний.

Энтропия и количество информации измеряются в одних и тех же единицах – в битах.

Определение. 1 бит – это энтропия системы с двумя равновероятными состояниями.

Пусть система X может находиться в двух состояниях x1 и x2 с равной вероятностью, т.е. N = 2; тогда её энтропия H(X) = Log_2 2 = 1 бит. Пример такой системы даёт нам монета, при подбрасывании которой выпадает либо орёл (x1), либо решка (x2). Если монета «правильная», то вероятность выпадения орла или решки одинаковая и равна 1/2.

Дадим ещё одно определение единицы измерения информации.

Определение. Ответ на вопрос любой природы (любого характера) содержит 1 бит информации, если он с равной вероятностью может быть «да» или «нет».

Пример. Игра в «пусто-густо». Вы прячете мелкий предмет в одной руке и предлагаете партнёру угадать, в какой руке вы его спрятали. Он спрашивает вас « в левой руке?» (или просто выбирает руку: левую или правую). Вы отвечаете «да», если он угадал, или «нет», в противном случае. При любом варианте ответа партнёр получает 1 бит информации, а неопределённость ситуации полностью снимается.

Формулу Хартли можно использовать при решении задач на определение выделенного элемента некоторого заданного множества. Этот результат можно сформулировать в виде следующего правила.

Если в заданном множестве M, состоящем из N элементов, выделен некоторый элемент x, о котором ничего более неизвестно, то для определения этого элемента необходимо получить Log_2N бит информации.

Рассмотрим несколько задач на применение формулы Хартли.

Задача 1. Некто задумал натуральное число в диапазоне от 1 до 32. Какое минимальное число вопросов надо задать, чтобы **гарантированно** угадать задуманное (выделенное) число. Ответы могут быть только «да» или «нет».

Комментарий. Можно попытаться угадать задуманное число простым перебором. Если повезёт, то придётся задать только один вопрос, а при самом неудачном варианте перебора придётся задать 31 вопрос. В

предложенной задаче нужно определить минимальное число вопросов, с помощью которых вы гарантированно определяете задуманное число.

Решение. По формуле Хартли можно вычислить количество информации, которое необходимо получить для определения выделенного элемента x из множества целых чисел {1,2,3 ……, 32}. Для этого необходимо получить $H = \mathrm{Log}_2\ 32 = 5$ бит информации. Вопросы надо задавать так, чтобы ответы на них были равновероятны. Тогда ответ на каждый такой вопрос будет приносить 1 бит информации. Например, можно разбить числа на две равные группы от 1 до 16 и от 17 до 32 и спросить, в какой группе находится задуманное число. Далее, аналогично следует поступить с выделенной группой, которая содержит уже лишь 16 чисел, и т.д. Пусть, например, задумано число 7.

Вопрос №1: Задуманное число принадлежит множеству {17 .. 32}? Ответ «нет» приносит вам 1 бит информации. Мы теперь знаем, что число принадлежит множеству {1 .. 16}.

Вопрос №2: Задуманное число принадлежит множеству {1 .. 8}? Ответ «да» приносит вам ещё 1 бит информации. Мы теперь знаем, что число принадлежит множеству {1 .. 8}.

Вопрос №3: Задуманное число принадлежит множеству {1 .. 4}? Ответ «нет» приносит вам ещё 1 бит информации. Мы теперь знаем, что число принадлежит множеству {5 .. 8}.

Вопрос №4: Задуманное число принадлежит множеству {7 ; 8}? Ответ «да» приносит вам ещё 1 бит информации. Мы теперь знаем, что число принадлежит множеству {7 ; 8}.

Вопрос №5: Задуманное число равно 8? Ответ «нет» приносит вам ещё 1 бит информации. Мы теперь знаем, что задуманное число равно 7. Задача решена. Было задано пять вопросов, в ответ получено 5 бит информации и определено задуманное число. ■

Задача 2. (Задача о фальшивой монете). Имеется 27 монет, из которых 26 настоящих и одна фальшивая. Каково минимальное число взвешиваний на рычажных весах, за которое можно гарантированно определить одну фальшивую монету из 27, используя то, что фальшивая монета легче настоящей.

Рычажные весы имеют две чашки и с их помощью можно лишь установить, одинаково ли по весу содержимое чашек, и если нет, то содержимое какой из чашек тяжелее.

Решение. Это задача на определение одного выделенного элемента из 27. По формуле Хартли мы сразу можем определить количество информации, которое нужно получить для определения фальшивой монеты: оно равно $I = \mathrm{Log}_2 27 = \mathrm{Log}_2(3^3) = 3\,\mathrm{Log}_2 3$ бит. Отметим, что ещё не зная стратегии взвешивания, можно сказать, сколько информации мы должны получить для решения задачи.

Если положить на чашки весов равное количество монет, то возможны три равновероятных исхода:

1. левая чашка тяжелее правой (Л > П);
2. левая чашка легче правой (Л < П);
3. левая чашка находится в равновесии с правой (Л = П);

Система «рычажные весы» может находиться в трёх равновероятных состояниях, поэтому одно взвешивание даёт $\mathrm{Log}_2 3$ бит информации. Всего для решения задачи надо получить $I = 3\,\mathrm{Log}_2 3$ бит информации, значит надо сделать три взвешивания для определения фальшивой монеты. Мы уже знаем минимальное число взвешиваний, но ещё не знаем, как их следует проводить. Стратегия должна быть такой, чтобы каждое взвешивание давало максимальное количество информации. Разделим все монеты на три равные кучки A, B и C по 9 штук в каждой. Фальшивая монета, обозначим её буквой f, может с равной вероятность находиться в

любой из трёх кучек. Выберем любые две из них, например A и B, и взвесим их. Возможны три исхода:

1) A тяжелее B (A > B); значит $f \in B$;

2) A легче B (A < B); значит $f \in A$;

3) A находится в равновесии с B (A = B); значит $f \in C$.

При любом исходе мы определим в какой кучке находится фальшивая монета f, но в этой кучке будет уже только 9 монет. Разобъём её на три равные кучки A1, B1, C1 по 3 монеты в каждой. Выберем любые две и взвесим их. Как и на предыдущем шаге, мы определим ту кучку монет, в которой находится фальшивая монета, но теперь кучка состоит только из трёх монет. Выберем любые две монеты и взвесим их. Это будет последнее, третье взвешивание, после которого мы найдём фальшивую монету. ■

Задача 3. Не используя калькулятор, оцените с точность до одного бита энтропию системы, которая может с равной вероятностью находится в 50 состояниях.

Решение. По формуле Хартли $H = \text{Log}_2 50$. Оценим данное выражение.

Очевидно, $32 < 50 < 64$; логарифмируем это неравенство → $\text{Log}_2 32 < \text{Log}_2 50 < \text{Log}_2 64$ → $5 < \text{Log}_2 50 < 6$. Энтропия системы с точностью до 1 бита $5 < H < 6$. ■

Задача 4. Известно, что энтропия системы составляет 7 бит. Определите число состояний этой системы, если известно, что все они равновероятны.

Решение. Обозначим через N число состояний системы. Так как все состояния равновероятны, то $H = \text{Log}_2 N$ → $N = 2^H$, т.е. $N = 2^7 = 128$. ■

3. Количество информации. Формула Шеннона.

Своё дальнейшее развитие теория информации получила в работах Клода Шеннона, американского инженера и математика (1916 – 2001). Шеннон является одним из создателей математической теории информации. Его основные труды посвящены теории релейно-контактных схем, математической теории связи, кибернетике. К. Шеннон изучал вопросы передачи информации в телеграфии, телефонии или радиовещании в виде сигналов электромагнитных колебаний. Одна из задач, которую ставил перед собой К. Шеннон, заключалась в том, чтобы определить систему кодирования, позволяющую оптимизировать скорость и достоверность передачи информации. Так как в годы войны он служил в шифровальном отделе, где занимался разработкой криптографических систем, то это позже помогло ему открыть методы кодирования с коррекцией ошибок. В своих работах 1948-1949 годов К. Шеннон определил количество информации через энтропию - величину, известную в термодинамике и статистической физике как мера разупорядоченности системы, а за единицу количества информации принял то, что впоследствии назвали битом (bit).

Для дальнейшего изложения необходимо использовать некоторые понятия теории вероятности: случайное событие, опыт, вероятность события, случайная величина. В окружающем нас мире происходят различные события, причём мы можем интуитивно, основываясь на опыте, оценивать одни из них как более возможные, чем другие. Случайным называют событие, которое может наступить или не наступить в результате некоторого испытания, опыта или эксперимента. Будем обозначать события заглавными буквами A, B, C и т.д. Количественная мера возможности наступления некоторого события A называется его вероятностью и обозначается как p(A), p – от английского probability. Чем более возможно наступление случайного события, тем больше его

вероятность: если А более возможно чем В, то p(A) > p(B). Вводится понятие достоверного события – событие, которое обязательно наступит. Это событие обозначают Ω и полагают, что его вероятность p(Ω) = 1. Невозможным называют событие, которое никогда не произойдёт. Его обозначают ∅ и полагают, что его вероятность p(∅) = 0. Для вероятностей всех остальных событий А выполняется неравенство p(∅) < p(A) < p(Ω), или 0 < p(A) < 1.

Для событий вводится понятие суммы и произведения. Сумма событий A+B – это событие, которое состоит в наступлении события А или В. Произведение событий A*B состоит в одновременном наступлении события А и В. События А и В **несовместны**, если они не могут наступить вместе в результате одного испытания. Вероятность суммы несовместных событий равна сумме их вероятностей. Если А и В несовместные события, то p(A+B) = p(A) + p(B).

События A1, A2, A3, …An образуют **полную группу**, если в результате опыта обязательно наступит хотя бы одно из них. Если события A1, A2, A3, …An попарно несовместны и образуют полную группу, то сумма их вероятностей p1+p2+p3+ …. pn =1. Если они при этом ещё и равновероятны, то вероятность каждого равна p = 1/n , где n – число событий. **Вероятность** события определяется как отношение числа благоприятных событию исходов опыта к общему числу исходов. **Частота** события – эмпирическое приближение его вероятности. Она вычисляется в результате проведения серии опытов как отношение числа опытов, в которых событие наступило к общему числу опытов. При большом числе опытов (испытаний) частота события стремится к его вероятности.

К. Шеннон, используя подход Р. Хартли, обратил внимание на то, что при передаче словесных сообщений частота (вероятность) использования различных букв алфавита не одинакова: некоторые буквы используются очень часто, другие - редко.

Рассмотрим алфавит A_m состоящий из m символов. Обозначим через p_i вероятность (частоту) появления i-ого символа в любой позиции передаваемого сообщения, состоящего из n символов. Один i – ый символ алфавита несёт количество информации равное $-Log_2(p_i)$. Перед логарифмом стоит «минус» потому, что количество информации величина неотрицательная, а $Log_2(x) < 0$ при $0<x<1$.

На месте каждого символа в сообщении может стоять любой символ алфавита A_m; количество информации, приходящееся на один символ сообщения, равно среднему значению информации по всем символам алфавита A_m:

$$-\sum_{i=1}^{m} p_i Log_2(p_i) \qquad (3.1)$$

Общее количество информации, содержащееся в сообщении из n символов равно:

$$I = -n * \sum_{i=1}^{m} p_i Log_2(p_i) \qquad (3.2)$$

Если все символы алфавита A_m появляются с равной вероятностью, то все $p_i = p$. Так как $\Sigma p_i = 1$, то $p = 1/m$.

$$-\sum_{i=1}^{m} p_i Log_2(p_i) = -\sum_{i=1}^{m} \frac{1}{m} Log_2(\frac{1}{m}) = -Log_2(\frac{1}{m}) = Log_2(m).$$

Формула (3.2) в случае, когда все символы алфавита равновероятны, принимает вид

$$I = n * Log_2(m).$$

Вывод: формула Шеннона (3.2) в случае, когда все символы алфавита равновероятны, переходит в формулу Хартли (2.2).

В общем случае количество энтропии H произвольной системы X (случайной величины), которая может находиться в m различных состояниях x_1, x_2, … x_m с вероятностями p_1, p_2, … p_m , вычисленное по формуле Шеннона, равно

$$H(X) = -\sum_{i=1}^{m} p_i Log_2(p_i). \qquad (3.3)$$

Напомним, что $p_1 + p_2 + \ldots + p_m = 1$. Если все p_i одинаковы, то все состояния системы X равновероятны; в этом случае $p_i = 1/m$, и формула (3.3) переходит в формулу Хартли (2.5): $H(X) = Log_2(m)$.

Замечание. Количество энтропии системы (случайной величины) X не зависит от того, в каких конкретно состояниях $x_1, x_2, \ldots x_m$ может находиться система, но зависит от числа m этих состояний и от вероятностей $p_1, p_2, \ldots p_m$, с которыми система может находиться в этих состояниях. Это означает, что две системы, у которых число состояний одинаково, а вероятности этих состояний $p_1, p_2, \ldots p_m$ равны (с точностью до порядка перечисления), имеют равные энтропии.

Теорема. Максимум энтропии H(X) достигается в том случае, когда все состояния системы равновероятны. Это означает, что

$$-\sum_{i=1}^{m} p_i Log_2(p_i) \le Log_2(m). \qquad (3.4)$$

Если система X может находиться только в одном состоянии (m=1), то её энтропия равна нулю. Рассмотрим систему, которая может принимать только два состояния x1 и x2 с вероятностями p1 и p2:

X1	X2
P1	P2

Среди всех таких систем с двумя состояниями наибольшая энтропия будет у системы с равновероятными состояниями, т.е. когда p1=p2 =1/2.

X1	X2
1/2	1/2

Количество энтропии такой системы равно

$$H(X) = -(1/2 * Log_2(1/2) + 1/2 * Log_2(1/2)) = -Log_2(1/2) = Log_2(2) = 1$$

Это количество принимается за единицу измерения энтропии (информации) и называется 1 бит (1 bit).

Рассмотрим функцию

$$h(x) = -(x*\log_2(x) + (1-x)*\log_2(1-x)). \qquad (3.5)$$

Область её определения – интервал (0 ;1), Lim h(x) = 0 при x → 0 или 1. График этой функции представлен на рисунке:

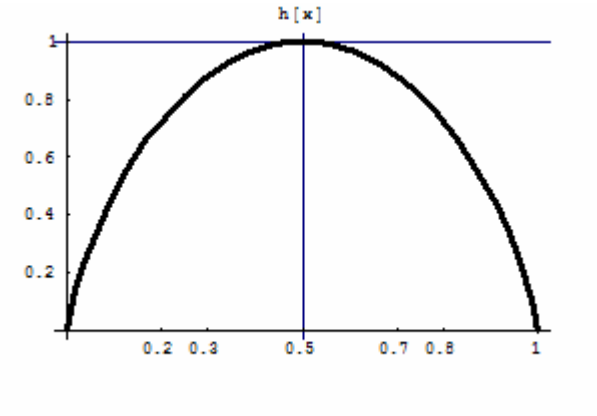


Рис. 4. График функции (3.5)

Если обозначить x через p_1, а (1-x) через p_2, то $p_1 + p_2 = 1$; $p_1, p_2 \in (0;1)$, $h(x) = H(p_1, p_2) = -(p_1*\log_2(p_1) + (p_2)*\log_2(p_2))$ – энтропия системы с двумя состояниями; максимум H достигается при $p_1 = p_2 = 0.5$.

График h(x) можно использовать при решении следующих задач:

Задача 1. Заданы три случайных величины X, Y, Z, каждая из которых принимает по два значения с вероятностями:

1. P(X = x1) = 0.5; P(X = x2) = 0.5;
2. P(Y = y1) = 0.2; P(Y = y2) = 0.8;
3. P(Z = z1) = 0.3; P(Z = z2) = 0.7 .

Запись P(X = x1) = 0.5 означает, что случайная величина X принимает значение x1 с вероятностью 0.5. Требуется расположить энтропии этих систем в порядке возрастания.

Решение. Энтропия H(X) равна 1 и будет наибольшей; энтропия H(Y) равна значению функции h(x), см. (3.5), при x = 0.2, т.е. H(Y)=h(0.2);

энтропия H(Z) = h(0.3). По графику h(x) можно определить, что h(0.2) < h(0.3). Следовательно, H(Y) < H(Z) < H(X). ■

Замечание 1. Энтропия системы тем больше, чем менее отличаются вероятности её состояний друг от друга. На основании этого можно сделать вывод, что H(Y) < H(Z). Например, если для систем X и Y с тремя состояниями заданы вероятности: для X {0.4; 0.3; 0.3}, для Y {0.1; 0.1; 0.8}, то очевидно, что неопределённость системы X больше, чем неопределённость системы Y: у последней, скорее всего, будет реализовано состояние, вероятность которого равна 0.8 .

Энтропия H(X) характеризует степень неопределённости системы. Чем больше объём полученных о системе сведений, тем больше будет информации о системе, и тем менее неопределённым будет её состояние для получателя информации.

Если энтропия системы после получения информации становится равной нулю, это означает, что неопределённость исчезла, вся энтропия «перешла» в информацию. В этом случае говорят, что была получена полная информацию о системе X. **Количество информации, приобретаемое при полном выяснении состояния физической системы, равно энтропии этой системы.**

Если после получения некоторого сообщения неопределённость системы X стала меньше, но не исчезла совсем, то количество информации, содержащееся в сообщении, равно приращению энтропии:

$$I = H1(X) - H2(X), \qquad (3.6)$$

где H1(X) и H2(X) - энтропия системы до и после сообщения, соответственно. Если H2(X) = 0, то мера неопределённости системы равна нулю и была получена полная информация о системе.

Пример. Вы хотите угадать количество очков, которое выпадет на игральном кубике. Вы получили сообщение, что выпало чётное число очков. Какое количество информации содержит это сообщение?

Решение. Энтропия системы «игральный кубик» H1 равна $Log_2 6$, т.к. кубик может случайным образом принять шесть **равновозможных** состояний {1, 2, 3, 4, 5, 6}. Полученное сообщение уменьшает число возможных состояний до трёх: {2, 4, 6}, т.е. энтропия системы теперь равна H2= $Log_2 3$. Приращение энтропии равно количеству полученной информации I = H1 – H2 = $Log_2 6$ - $Log_2 3$ = $Log_2 2$ = 1 bit. ■

На примере разобранной задачи можно пояснить одно из распространённых определений единицы измерения – 1 бит: **1 бит - количество информации, которое уменьшает неопределённость состояния системы в два раза.** Неопределённость дискретной системы зависит от числа её состояний N. Энтропия до получения информации H1= Log_2 N. Если после получения информации неопределённость уменьшилась в два раза, то это означает, что число состояний стало равным N/2, а энтропия H2 = Log_2 N/2. Количество полученной информации I= H1 -H2 = Log_2 N - Log_2 N/2 = Log_2 2 = 1 бит.

Рассмотрим несколько задач на применение формулы Шеннона и Хартли.

Задача 2. Может ли энтропия системы, которая принимает случайным образом одно из 4-х состояний, равняться: а) 3; б) 2.1 в) 1.9 г) 1; д) 0.3? Ответ объяснить.

Решение. Максимально возможное значение энтропия системы с 4-мя состояниями достигает в случае, когда все состояния равновероятны. Это значение по формуле Хартли равно $Log_2 4$ = 2 бита. Во всех других случаях энтропия системы с 4-мя состояниями будет меньше 2. Следовательно, возможными значениями энтропии из перечисленных выше, могут быть значения 1.9, 1, 0.3. ■

Задача 3. Задана функция H(x)= -x*Log_2(x) - (1-x)*Log_2(1-x). Расположите в порядке возрастания следующие значения: H(0.9), H(0.85), H(0.45), H(0.2), H(0.15).

Решение. Используем график функции (3.5). Наибольшим значением будет H(0.45), наименьшим значением – H(0.9), затем по возрастанию идут значения H(0.15) и H(0.85) = H(0.15); H(0.2). Ответ: H(0.9) < H(0.15)=H(0.85)< H(0.2) < H(0.45). ■

Задача 4. По линии связи переданы сообщения: a) «начало_в_10»; b) «лоанча_1_в0». Сравните количество информации в первом и втором сообщении.

Решение. Первое и второе сообщение состоят из одних и тех же символов: второе получено из первого в результате перестановки этих символов. В соответствии с формулой Шеннона эти сообщения содержат одинаковое количество информации. При этом первое сообщение несёт содержательную информацию, а второе – простой набор символов. Однако, в этом случае можно сказать, что второе сообщение является «зашифрованным» вариантом первого, и поэтому количество информации в обоих сообщениях одинаковое.■

Задача 5. Получены три различных сообщения A, B, C:

A= «прибытие в десять часов»; B= «прибытие в десять часов ноль минут»; C= «прибытие ровно в десять часов». Используя энтропийный подход Шеннона, сравните количество информации, содержащееся в этих сообщениях.

Решение. Обозначим количество информации в сообщениях A, B, C через I(A), I(B), I(C) соответственно. В смысле «содержания» эти сообщения совершенно одинаковы, но одинаковое содержание выражено с помощью разного количества символов. При этом все символы сообщения A содержатся в сообщении B и C, сообщение C = A + «ровно», B = A + «ноль минут»; в соответствии с подходом Шеннона получаем: I(A) < I(C) < I(B). ■

4. Условная энтропия.

Рассматривая формулу Шеннона (3.3) для вычисления энтропии случайной величины и количества информации, мы предполагали, что информация о случайной величине (X) поступает непосредственно к наблюдателю. Однако, как правило, мы получаем информацию не о той случайной величине (X), которая нас интересует, а о некоторой другой (Y), которая связана с X стохастическим образом. Такая связь случайных величин отличается от функциональной связи, при которой каждому значению одной величины соответствует единственное, вполне определённое значение другой величины. Стохастическая (вероятностная) связь двух случайных величин X и Y означает, что изменение одной из них влияет на значение другой, но таким образом, что зная значение X нельзя точно указать значение, которое примет величина Y. Можно лишь указать тенденцию изменения величины Y.

Пусть B – случайное событие; p(B) – вероятность его наступления; обозначим через X случайную величину, которая принимает N различных значений $\{x_1, x_2, \ldots x_N\}$, а через A_k событие, состоящее в том, что случайная величина X примет значение x_k:

$$A_k = \{ X = x_k\}, \; k=1,2, \ldots N ;$$

Вероятность события A_k обозначим через $p(A_k)$. Вероятность наступления некоторых событий может меняться в зависимости от того, наступило или нет некоторое другое событие. Вероятность $p_B(A_k)$ события A_k, вычисленная в предположении, что наступило событие B, называется условной вероятностью события A_k , при этом:

$$p_B(A_k) = \frac{p(A_k B)}{p(B)} \qquad (4.1)$$

События A_k и B называются независимыми, если вероятность наступления события A_k не зависит от того, наступило или нет событие B. Это означает, что условная вероятность события $p_B(A_k)$ равна «обычной» вероятности $p(A_k)$.

Определение. Условной энтропией случайной величины X при условии B называется величина

$$H_B(X) = -\sum_{k=1}^{N} p_B(A_k) Log_2(p_B(A_k)) \qquad (4.2)$$

Отличие от формулы Шеннона (3.3) заключается в том, что вместо вероятностей $p(A_k)$ используются условные вероятности $p_B(A_k)$.

Пусть теперь Y – другая случайная величина, принимающая значения $\{y_1, y_2, \ldots y_M\}$. Обозначим через B_j событие, состоящее в том, что случайная величина Y примет значение y_j:

$$B_j = \{ Y = y_j\}, \quad j=1, 2,\ldots M.$$

Вероятность события B_j обозначим через $p(B_j)$.

Определение. Условной энтропией случайной величины X при заданном значении случайной величины Y называется величина $H_Y(X)$

$$H_Y(X) = \sum_{j=1}^{M} p(B_j) H_{B_j}(X) \qquad (4.3)$$

Выполним преобразование формулы (4.3):

$$H_Y(X) = \sum_{j=1}^{M} p(B_j) H_{B_j}(X) = -\sum_{j=1}^{M} p(B_j) \sum_{k=1}^{N} p_{B_j}(A_k) Log_2(p_{B_j}(A_k) =$$

$$= -\sum_{j=1}^{M}\sum_{k=1}^{N} p(B_j) p_{B_j}(A_k) Log_2(p_{B_j}(A_k)) = -\sum_{j=1}^{M}\sum_{k=1}^{N} p(A_k B_j) Log_2(\frac{p(A_k B_j)}{p(B_j)})$$

Формула (4.3) принимает вид:

$$H_Y(X) = -\sum_{j=1}^{M}\sum_{k=1}^{N} p(A_k B_j) Log_2\left(\frac{p(A_k B_j)}{p(B_j)}\right) \quad (4.4)$$

Вычислим количество информации о случайной величине X, полученное при наблюдении за случайной величиной Y. Это количество информации I(X,Y) равно убыли энтропии случайной величины X при наблюдении за случайной величиной Y:

$$I(X,Y) = H(X) - H_Y(X) \quad (4.5)$$

Подставим в (15) выражения для H(X) и H_Y(X):

$$I(X,Y) = -\sum_{k=1}^{N} p(A_k) Log_2(p(A_k)) + \sum_{j=1}^{M}\sum_{k=1}^{N} p(A_k B_j) Log_2\left(\frac{p(A_k B_j)}{p(B_j)}\right)$$

Заменим в первой сумме $p(A_k)=p(A_kB_1)+ p(A_kB_2)+ p(A_kB_3)\ldots+ p(A_kB_M)$. Это равенство действительно имеет место, т.к. события A_kB_1, A_kB_2, … A_kB_M – попарно несовместные, при этом одно из них наступит, если наступит A_k . Наоборот, если наступит одно из B_j , то наступит и A_k . Продолжая преобразования, получим:

$$I(X,Y) = -\sum_{k=1}^{N}\sum_{j=1}^{M} p(A_k B_j) Log_2(p(A_k)) + \sum_{j=1}^{M}\sum_{k=1}^{N} p(A_k B_j) Log_2\left(\frac{p(A_k B_j)}{p(B_j)}\right) =$$

$$= \sum_{j=1}^{M}\sum_{k=1}^{N} p(A_k B_j)\left[Log_2\left(\frac{p(A_k B_j)}{p(B_j)}\right) - Log_2(p(A_k))\right]$$

$$= \sum_{j=1}^{M}\sum_{k=1}^{N} p(A_k B_j) Log_2\left(\frac{p(A_k B_j)}{p(A_k)p(B_j)}\right)$$

Итак, мы получили формулу для вычисления количества информации о случайной величине X при наблюдении за другой случайной величиной Y:

$$I(X,Y) = \sum_{j=1}^{M}\sum_{k=1}^{N} p(A_k B_j) Log_2\left(\frac{p(A_k B_j)}{p(A_k)p(B_j)}\right) \quad (4.6)$$

Если случайные величины (или события) независимы, то для них выполняется соотношение $p(A_kB_j) = p(A_k)p(B_j)$ – вероятность совместного наступления двух событий равна произведению вероятностей этих событий.

Относительно величины I(X,Y) справедливы следующие утверждения.

Для независимых случайных величин получим

$$I(X,Y) = 0.$$

Это означает, что наблюдение за случайной величиной Y не даст никакого преимущества в получении информации о случайной величине X.

В других случаях I(X,Y) >0, при этом выполняется неравенство:

$$I(X,Y) \leq H(X).$$

Равенство достигается в случае наличия функциональной связи Y=F(X). В этом случае наблюдение за Y даёт полную информацию о X. Если Y=X, то I(X,X) = H(X).

Величина I(X,Y) симметрична: I(X,Y) = I(Y,X). Это означает, что наблюдение случайной величины Y даёт такое же количество информации о случайной величине X, какое наблюдение случайной величины X даёт относительно случайной величины Y. Если мы рассматриваем две случайные величины, которые находятся в стохастической зависимости, то средствами теории информации нельзя установить какая из них является причиной, а какая следствием.

5. Кибернетика.

Официальная история кибернетики началась в 1948 г., когда вышла в свет знаменитая книга “Кибернетика, или управление и связь в животном и машине”, автором которой был Норберт Винер, профессор математики Массачусетского технологического института.

В предисловии к этой книге, издание которой состоялось в СССР в 1968 г., редактор перевода Г.Н. Поваров приводит следующие биографические данные: «Норберт Винер родился 26 ноября 1894 г. в городе Колумбия, штат Миссури, в семье иммигранта. Его отец, Лео Винер (1862–1939), уроженец Белостока, тогда принадлежавшего России, в молодости учился в Германии, а затем переселился за океан, в Соединенные Штаты». Винер значительно облегчил задачу своих биографов, написав на склоне лет две книги воспоминаний: одна из них посвящена детству и годам учения - «Бывший вундеркинд»; другая – профессиональной карьере и творчеству «Я – математик».

Основной тезис книги Винера «Кибернетика» – подобие процессов управления и связи в машинах, живых организмах и обществах, будь то общества животных (муравейник) или человеческие сообщества. Прежде всего - это процессы передачи, хранения и переработки информации, т.е. различных сигналов, сообщений, сведений. Любой сигнал, любую информацию, независимо от ее конкретного содержания и назначения, можно рассматривать как некоторый выбор между двумя или более значениями, наделенными известными вероятностями, и это позволяет подойти ко всем процессам с единой меркой, с единым статистическим аппаратом.

Количество информации – количество выбора, которое отождествляется Винером с отрицательной энтропией и становится, подобно количеству вещества или энергии, одной из фундаментальных

характеристик явлений природы. Отсюда толкование кибернетики как теории организации, как теории борьбы с мировым хаосом, с роковым возрастанием энтропии. Книга Винера не содержит последовательного курса кибернетики. В 1948 г. это был только проект. Винер не раз отмечает в книге ее предварительный, вводный характер - до подробного, систематического построения новой науки было еще далеко.

Термин кибернетика происходит от греческого «kybernetike», что означает искусство управления рулём. По определению Норберта Винера : «Кибернетика - наука о управлении и связи в живом организме и машине». В современной литературе встречается похожее определение: «Кибернетика - наука об общих закономерностях процессов управления и передачи информации в машинах, живых организмах и обществе».

Следует отметить, что создатель кибернетики работал в непосредственном контакте и с Клодом Шенноном, и с Джоном фон Нейманом, и с другими учёными, занимавшимися разработкой первых ЭВМ, с создателями теории информации и кибернетики. Огромное значение для понимания сущности информации и развития кибернетики имели выдающиеся работы английского биолога У. Эшби. Первоначально «задуманная» как наука об управлении, в частности об управлении государством, кибернетика, «благодаря» излишней, может быть, математизации превратилась в науку об обработке информации с помощью вычислительной техники. Такому понижению статуса кибернетики способствовали работы Шеннона, Бриллюэна, Шредингера, в которых излишне акцентировались моменты аналогии со статистической энтропией в ущерб более физическому подходу в изучении информации.

Рассмотрим некоторые общие понятия, которые будут использоваться в дальнейшем изложении: система, структура и связанные с ними понятия. **Системой** называется совокупность элементов, взаимосвязь и взаимодействие которых приводит к возникновению новых интегративных

свойств этой совокупности, не сводимых к свойству составляющих её элементов. В отличие от системы, а**грегат** - простая совокупность элементов, механическая смесь или соединение в одно целое разнородных или однородных частей. Агрегат не является системой. **Множество** – это родовое понятие для системы. Связь между элементами в множестве отсутствует.

Выделяют малые, сложные, саморегулирующиеся и саморазвивающиеся системы (по количеству элементов и количеству взаимосвязей между ними). Малую систему (в отличие от большой) можно разобрать и собрать (например, механические часы, велосипед и т.п.). Любой организм – саморазвивающаяся система, не подлежит разборке на составные части. Под **строением системы** подразумевают элементы, из которых она состоит и из которых могут быть образованы отдельные её части – **подсистемы**. Подсистемы обычно встречаются в иерархически организованных системах. Примеры таких систем – социальные и живые системы. Человеческий организм – иерархическая система, состоящая из нервной системы, сердечнососудистой, дыхательной, пищеварительной и других подсистем. В этих подсистемах можно выделить органы, состоящие из ткани, ткань – из клеток, и т.д.

С**труктура системы** - связи и взаимодействия между её элементами, благодаря которым возникают новые интегративные свойства системы, отличные от свойств её элементов. Характер взаимодействия элементов определяет тип систем: химические, физические, биологические, социальные.

К семидесятым годам XX века кибернетика сложилась как физико-математическая наука со своим собственным предметом исследования – кибернетическими системами. **Кибернетическая система** – множество взаимосвязанных объектов (элементов), способных воспринимать, хранить, перерабатывать и использовать информацию для управления и

регулирования системой. Кибернетические системы рассматриваются независимо от их материальной природы: пчелиный рой, автоматические регуляторы, ЭВМ, человеческий мозг, государства, например СССР, США, и т.п.

Пример пчелиного роя показывает, что элементы кибернетической системы могут быть устроены сложнее самой системы. Пчелиный рой, представляя из себя нечто единое, состоит из отдельный пчёл, каждая из которых сама является сложно устроенным организмом. Пчёлы образуют рой благодаря существующим между ними информационным связям: они могут воспринимать, хранить, перерабатывать и использовать информацию для управления и регулирования системой – пчелиным роем. Всякое государство даёт пример кибернетической системы; наиболее характерным является пример такого государства, которое само состоит из отдельных государственных образований – республик (штатов): СССР, Россия, США. Разрушение информационных связей между элементами кибернетической системы приводит к её разрушению, распаду. Поэтому для государства так важно существование общего информационного пространства, единой системы образования, общих культурных и религиозных традиций и т.п.

В 1959 г. акад. А.Н. Колмогоров в предисловии к книге английского кибернетика д-ра У.Р. Эшби писал: «Сейчас уже поздно спорить о степени удачи Винера, когда он в своей известной книге в 1948 году выбрал для новой науки название “кибернетика”. Это название достаточно установилось и воспринимается как новый термин, мало связанный со своей греческой этимологией. Кибернетика занимается изучением систем любой природы, способных воспринимать, хранить и перерабатывать информацию, а также использовать ее для управления и регулирования. При этом кибернетика широко пользуется математическими методами и стремится к получению конкретных специальных результатов,

позволяющих как анализировать такого рода системы (восстанавливать их устройство на основании опыта обращения с ними), так и синтезировать их (рассчитывать схемы систем, способных осуществлять заданные действия)».

Приведём определение кибернетики, данное А.Н. Колмогоровым (на основе понятия информации): «Кибернетика изучает машины, живые организмы и их объединения исключительно с точки зрения их способности:

1. воспринимать определённую «информацию»;
2. сохранять эту информацию в «памяти»;
3. передавать её по «каналам связи»;
4. перерабатывать её в «сигналы», направляющие их деятельность в соответствующую сторону.

Процессы восприятия информации, её хранения и передачи называются в кибернетике связью, а переработка воспринятой информации в сигналы, направляющие деятельность машин и организмов, - управлением».

Математическая формализация определения Колмогорова приводит к следующему определению кибернетической системы. Пусть кибернетическая система состоит только из одного элемента A. В абстрактном плане элемент A состоит из набора пяти объектов A = { x, y, z, f, g }, где

- x(t) – входной сигнал элемента A;
- y(t) – выходной сигнал (реакция) элемента A;
- z(t) – внутреннее состояние элемента A;
- текущее значение z(t) зависит от входного сигнала, момента времени и предыдущего внутреннего состояния. Зависимость определяется функцией f: $z(t) = f(t, x, z(t_{пред}))$, где $z(t_{пред})$ - состояние системы в предыдущий момент времени;

- реакция (выход) системы $y(t) = g(t, x, z(t_{пред}))$ зависит от входного сигнала, момента времени и предыдущего внутреннего состояния.

К перечисленным выше пяти объектам элемента А надо добавить состояние кибернетической системы в начальный момент t_0: z_0 – начальное состояние системы и y_0 – реакция в начальный момент времени. Как правило, при описании кибернетической системы время t - это дискретные равноотстоящие моменты $t_n = t(n)$; тогда $z(_{пред})=z(t_{n-1})$ и $z(t_n) =f(t_n, x, z(t_{n-1}))$.

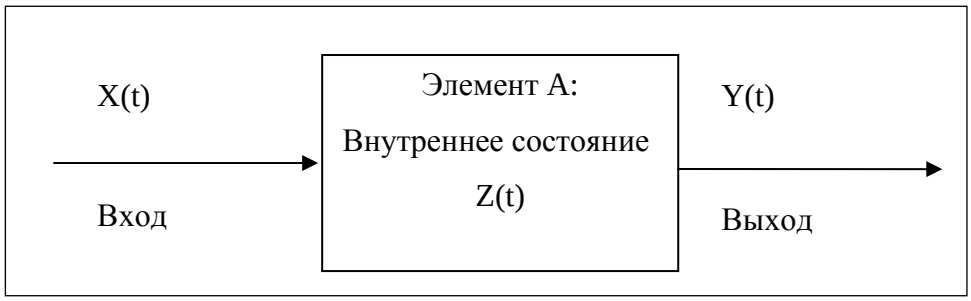
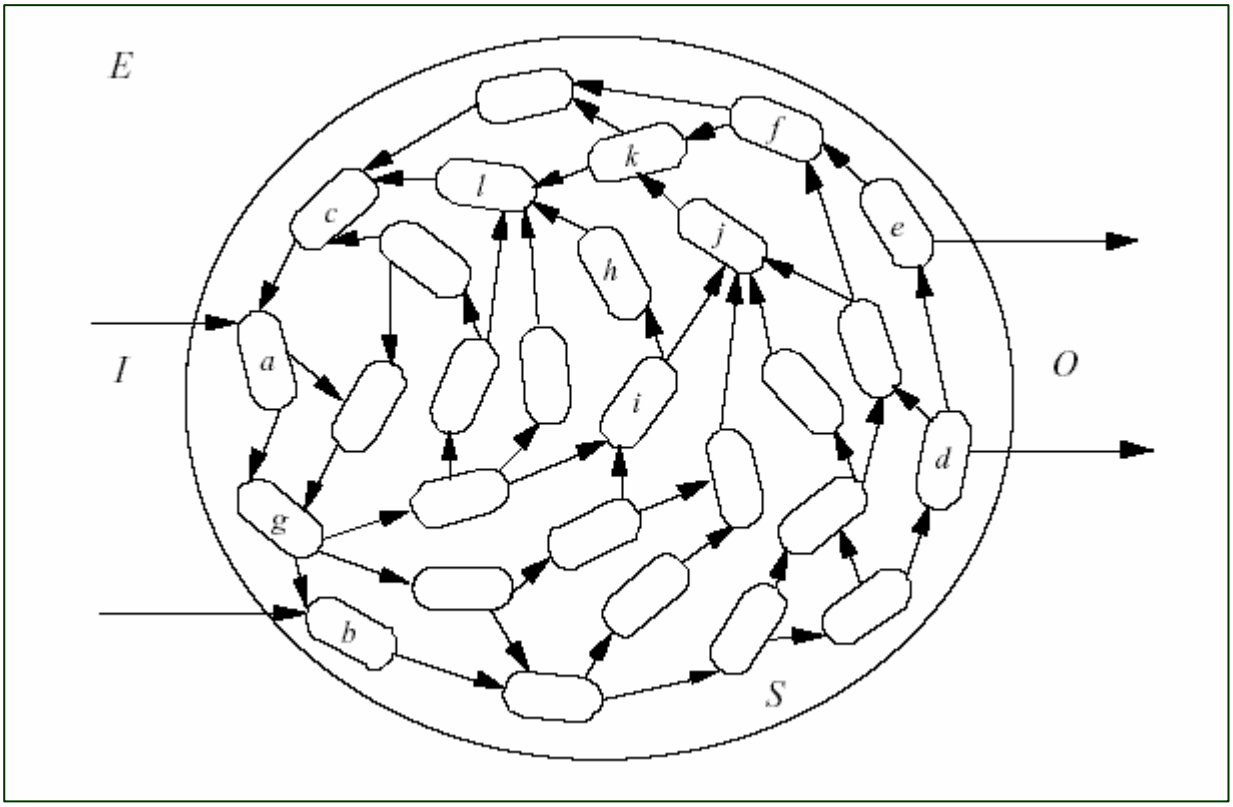


Рис. 5. Одноэлементная кибернетическая система.

Многоэлементная кибернетическая система S строится из набора элементов {a, b, c, d, i, h, …} путем отождествления выхода одних элементов с входами других. Свободные входы и выходы образуют вход I (input) и выход O (output) всей кибернетической системы, см. рис. 6. Система S состоит из сети компонентов или подсистем {a, b, c, d,...}, связанных друг с другом через входы и выходы, они постоянно воспроизводят собственную организацию. Например, вход элемента *l* связан с выходами k и h (товары, услуги, информация,), а выход *l* связан с входом c. В целом, сеть замкнута (пути, соединяющие компоненты, находятся внутри системы), но она связана с окружающей средой E через вход I и выход O. В сети существует ряд избыточных или "параллельных" путей, которые начинаются из одного и того же компонента (например, i)

и заканчиваются на одном компоненте (например, *l*). В этом особом случае компонент h выполняет ту же самую функцию для *l* как j и k , к тому же *l* может предпочесть обходной путь i → j → k →*l* более короткому пути i → h → *l*.

Рис. 6. Многоэлементная кибернетическая система.



В середине XX века появился новый метод исследования – машинный эксперимент. До этого в науке использовались два классических метода: дедуктивный и индуктивный. **Дедукция** - вывод по правилам логики, цепь умозаключений от аксиом к следствиям. Дедуктивные методы применяются в математике – это среда их обитания. Классический пример применения дедуктивного метода даёт геометрия Евклида. **Индукция** – переход от единичных фактов к общему утверждению. Все естественные науки используют индукцию: физика, геология, химия и т.д. Делая наблюдения и анализируя результаты экспериментов, учёные

формулируют утверждения всеобщего характера. Например, «все тела при нагревании расширяются», закон Архимеда, и т.п.

Машинный эксперимент занимает промежуточное положение между классическими дедуктивным и индуктивным методами научного исследования. Его становление было связана с появлением ЭВМ и новой науки – кибернетики. Для описания элементов кибернетической системы и связей между ними используется математический аппарат, который включает в себя: алгебраические соотношения, дифференциальные уравнения, уравнения в частных производных, разностные уравнения. Строится математическая модель изучаемой системы - приближённое описание какого-либо класса явлений, использующее математический язык (математическую символику). Затем разрабатывается алгоритм решения поставленной на языке математики задачи, выбирается язык программирования, на котором составляется программа решения задачи, программа вводится в ЭВМ. Все эти этапы машинного эксперимента можно отнести к дедукции. В результате получаем модель изучаемого объекта в памяти ЭВМ. Теперь мы можем имитировать поведение объекта, запуская программу при различных начальных условиях. Эта часть машинного эксперимента относится к индукции. Изучая поведение модели в ходе работы ЭВМ, можно делать выводы о том, как поведёт себя реальный объект при аналогичных начальных условиях, проверять гипотезы относительно свойств объекта. Значение математического моделирования определяется тем, что на ЭВМ можно проводить такие эксперименты, которые невозможно осуществить с реальным объектом. Это может быть связано либо с большой стоимостью, либо с опасными последствиями реальных экспериментов. Примеры машинного эксперимента: моделирование геологических процессов (тектоника плит), решение экологических задач, исследование последствий экономических преобразований, всевозможные эмуляторы и тренажёры. Одна из наиболее

известных задач, решённых с применением ЭВМ – моделирование последствий ядерной войны. Соответствующая модель и программа были разработаны под руководством русского учёного Н.Н. Моисеева ещё в период холодной войны. Было показано, что результатом применения ядерного оружия будет установление на всей планете «ядерной зимы». В дальнейшем американские учёные повторили расчёты Н.Н. Моисеева и согласились с его выводами. Ясно, что настоящий, реальный эксперимент в данном случае был невозможен.

В ходе проведения машинного эксперимента можно выделить следующие основные этапы:

1) постановка задачи;
2) построение математической модели изучаемой системы;
3) выбор или разработка алгоритма решения задачи;
4) написание программы на основе предложенного алгоритма;
5) анализ полученных результатов, сравнение модели и реального объекта;
6) корректировка модели, алгоритма или программы.

Существует четыре основных типа задач, возникающих в процессе изучения кибернетических систем методом математического моделирования. Обозначим кибернетическую систему L_a , где a – параметры системы, X(t) –вход системы (внешнее воздействие), Y(t) – выход (реакция системы). Можно записать $Y(t)=L_a(X(t))$; соответствующая структурная схема:

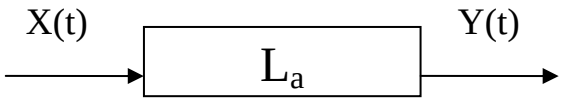


Рис.7 Структурная схема системы в терминах «вход-выход».

Основные задачи математического моделирования (перечисляются в порядке возрастания их сложности):

1. **Прямая задача:** при заданных значениях внешнего воздействия X(t), заданной системы L и её параметров «a» найти реакцию системы Y(t).
2. **Обратная задача:** Задана система L и её параметры «a», известна реакция системы Y(t). Требуется определить входное воздействие X(t), которое вызвало заданную реакцию Y(t).
3. **Задача идентификация параметров**: задано описание системы L, вход системы X(t) и реакция системы Y(t). Требуется уточнить значения параметров системы «a».
4. **«Чёрный ящик»:** Известна реакция системы Y(t) на воздействие X(t). Требуется воссоздать описание системы L_a так, чтобы для заданных воздействий получать заданные реакции.

Решение любой задачи методом машинного эксперимента начинается с решения прямой задачи. Она помогает понять, как поведёт себя реальная система при определённых внешних воздействиях. Научившись решать прямую задачу, можно переходить к решению обратной, т.е. попытаться ответить на вопрос «какие воздействия X могли привести к данной реакции Y?». Дальнейшие исследования связаны, как правило, с уточнением параметров моделируемой системы. Для этого проводят минимизацию рассогласования модельных (Y_M) и реальных (Y_R) значений реакции системы при известных значениях входных воздействий X путём варьирования параметров «a»:

$$\| Y_M - Y_R \| \underset{a}{\rightarrow} \min$$

Самая сложная задача – «чёрный ящик»: нужно определить (понять) устройство системы, если известны её реакции на заданные воздействия. Считается, что задача решена, если на заданные воздействия система

реагирует заданным образом. Пример такой задачи – изучение устройства и законов функционирования головного мозга человека.

Среди всех систем можно выделить весьма важный класс линейных систем. Линейные системы – это системы, для которых выполняются следующие свойства:

1) если y реакция системы на x: y = L(x), то на воздействие α*x реакция будет равна α*y, т.е. L(α*x) = α*L(x) = α*y;
2) если y1 реакция системы на воздействие x1, а y2 реакция системы на воздействие x2, то на сумму воздействий x1+x2 реакция равна сумме реакций y1+y2: L(x1+x2) = L(x1)+L(x2) = y1+y2.

Примеры линейных систем (моделей): 1) упругая среда, для которой справедлив закон Гука: деформация пропорциональна приложенной силе.

На протяжении трёх веков – с XVIII по XX, учёные исследовали линейные системы, а всякий нелинейный случай пытались свести к линейному. В это время господствовал принцип редукционизма, в соответствии с которым любую систему можно представить в виде суперпозиции нескольких линейных систем. Принцип редукционизма оказался весьма плодотворным и применяется в науке до настоящего времени. Однако, во второй половине XX в. – начале XXI в. было показано, что далеко не все нелинейные системы можно представить в виде суперпозиции линейных систем. Более того, основное внимание в новых научных направлениях, таких как синергетика, теория динамического хаоса, теория катастроф, уделяется именно нелинейным системам.

6. Алгоритмический, семантический и ценностный подходы к определению информации.

Классическая теория информации Шеннона не может охватить всего многообразия понятия информации и, в первую очередь, ее содержательного аспекта. Теория информации К. Шеннона также не занимается определением ценности информации, так как её интересуют лишь проблемы передачи данных оптимальным образом.

Наряду с энтропийным (синтаксическим) подходом Шеннона к определению информации существуют и другие, среди которых следует отметить алгоритмический, семантический и прагматический (ценностный) подходы.

Идея алгоритмического измерения количества информации была выдвинута в 1965 г. А.Н. Колмогоровым. Суть ее заключается в том, что количество информации определяется как минимальная длина программы, позволяющей преобразовать один объект (множество) в другой. Чем больше различаются два объекта между собой, тем сложнее (длиннее) программа перехода от одного объекта к другому. Так, воспроизвести последовательность букв **a, a,...,a** можно при помощи очень простой программы. Несколько большей окажется длина программы, восстанавливающей последовательность **a, в, c, a, в, c,...** Длина программы при этом измеряется количеством команд (операций), позволяющих воспроизвести заданную последовательность. Этот подход, не базирующийся на понятии вероятности, позволяет, например, определить прирост количества информации, содержащейся в результатах расчета, по сравнению с исходными данными. Вероятностный подход Шеннона в теории информации не может дать ответа на подобные вопросы.

Попытки оценить не только количественную, но и содержательную сторону информации дали толчок к дразвитию семантической (смысловой)

теории информации. Исследования в этой области теснее всего связаны с семиотикой – теорией знаковых систем. Семиотика исследует знаки как особый вид носителей информации. При этом знаком является условное изображение элемента сообщения, словом – совокупность знаков, имеющих смысловое значение, языком – словарь и правила пользования им. Таким образом, рассуждая о количестве, содержании и ценности информации, заключённой в сообщении, можно исходить из возможностей анализа знаковых структур.

В качестве знаковых систем используются естественные и искусственные языки, в том числе и языки программирования, различные системы сигнализации, логические, математические и химические символы. Сразу же заметим, что методы точного количественного определения смыслового содержания информации до настоящего времени еще не разработаны.

Рассматривая знаковые системы, выделяют три основных аспекта их изучения: синтактику, семантику и прагматику. **Синтактика** изучает синтаксис знаковых структур, т.е. способы сочетаний знаков, правила образования этих сочетаний и их преобразований безотносительно к их значениям. Отметим, что рассмотренные ранее способы определения количества информации можно отнести к синтаксическим способам.

Семантика изучает знаковые системы как средства выражения смысла, определенного содержания, т.е. правила интерпретации знаков и их сочетаний, смысловую сторону языка.

Прагматика рассматривает соотношение между знаковыми системами и их пользователями, или приемниками-интерпретаторами сообщений. Иными словами, к прагматике относится изучение практической полезности знаков, слов и, следовательно, сообщений, т.е. потребительский аспект изучения языка.

Ценностный подход к определению количества информации представлен в работах М.М. Бонгарда, А.А. Харкевича, В.И. Корогодина, Д.С. Чернавского.

Д.С. Чернавский придерживается следующего определения информации: «Информация - это запомненный выбор одного варианта из нескольких возможных и равноправных». Это определение является развитием подхода, предложенного Г. Кастлером.

Комментарий. Определение Кастлера начинается словами «**случайный и** запомненный выбор». Однако, как отмечает Д.С. Чернавский, выбор может быть сделан и неслучайным образом. Кроме этого, выбор должен быть обязательно запомнен. Запоминание означает, что сделанный выбор сохраняется в течение времени, которое больше, чем характерное время использования информации. Равноправие вариантов означает, что априорные различия между ними невелики. Равноправие выборов не означает их равновероятности, но означает, что вероятности вариантов это величины одного порядка.

Если выбор сделан случайным образом - говорят о **генерации** информации, если выбор неслучаен, продиктован «сверху» – это **рецепция** информации. Рецепция – выбор навязанный системе. Это означает перевод системы из одного состояния в другое, определённое состояние. Рецепция часто встречается в технике, когда некоторую динамическую систему переводят в нужное состояние путём воздействия на неё электрическим или световым импульсом. В этом случае говорят о силовом переключении. Существует параметрическое переключение, когда перевод системы в нужное состояние осуществляется путём изменения её параметров. Такой способ переключения чаще встречается в биологических системах. Процесс обучения «учитель – ученик» основан на рецепции информации: обучаемый воспринимает некоторую информацию, которую передаёт ему

учитель. Генерация информации более «творческий» процесс – его результат заранее непредсказуем.

Результат рецепции или генерации зависит от информации, которая уже воспринята и содержится в генераторе или рецепторе, т.е. новые выборы можно сделать только на основе выборов, сделанных ранее. В связи с этим можно ввести понятие тезауруса. Тезаурус - информация, содержащаяся в системе на данном уровне, необходимая для рецепции или генерации информации на следующем уровне. Без тезауруса нет множества, из которого нужно сделать выбор. Каждый раз, когда мы воспринимаем или генерируем информацию, нам необходимы некоторые знания (язык, математика, история, биология и т.п.), которые и составляют наш тезаурус. Поэтому студенты, слушающие лекцию в одной аудитории, воспринимают разную информацию, а кто-то вообще её не воспринимает, в силу разного тезауруса, которым они обладают на данный момент времени.

Запоминание сделанного выбора играет важную роль. Запомненный выбор называют макроинформацией. Если выбор не запоминается, то фиксации информации не происходит – такой выбор называется микроинформацией. Микроинформация встречается в физических системах с большим числом состояний; при этом спонтанный переход большого числа частиц из одного состояния в другое совершается в течение малого отрезка времени $t = 10^{-13}$ с. Именно поэтому, термодинамическая трактовка информации для таких систем некорректна – термодинамическая система не запоминает «сделанный выбор». Здесь мы имеем дело с микроинформацией. Свойство запоминания присуще лишь макросистемам, которые могут сохранять выбранное состояние достаточно долгий период времени. Таким простейшим запоминающим устройством является триггер – система с двумя устойчивыми состояниями. Переход из одного состояния в другое происходит не

спонтанно, а в результате внешнего воздействия – происходит рецепция информации. Вывод: нельзя путать Больцмановскую энтропию с информационной энтропией.

На начальном этапе своего развития информатика занималась преимущественно проблемами передачи и хранения информации. Вопросы, связанные с генерацией и ценностью информации, стали ставиться сравнительно недавно. Ценность информации зависит от той цели, к которой стремиться воспринимающая информацию система (объект). Чем в большей мере информация способствует достижению цели, тем более ценной она считается. Здесь возможны два случая.

1) Цель наверняка достижима, причём разными способами. Ценность информации можно определить по тому, насколько эта информация помогает уменьшить затраты (материальные или временные) для достижения цели.
2) Цель достижима с некоторой вероятностью. В этом случае мерой ценности информации может служить величина

$$V = Log_2 \frac{P_1}{P_0} \ , \qquad (6.1)$$

где P_0 – вероятность достижения цели до получения информации (априорная вероятность), P_1 - вероятность достижения цели после получения информации (апостериорная вероятность)[1]. Если до получения информации все варианты достижения цели равновероятны, то $P_0 = 1/N$, где N – число вариантов. Если варианты имеют разную вероятность, то можно воспользоваться формулой Шеннона и вычислить полную вероятность системы I и определить априорную вероятность из уравнения $I = Log_2(P_0)$; т.е. $P_0 = 2^{-I}$.

Апостериорная вероятность P_1 может быть: 1) больше P_0 : $P_1 > P_0$;

[1] От латинского «a priory» – знание полученное до опыта, «a posteriory» – после опыта.

2) равна P_0 : $P_1 = P_0$; 3) меньше P_0 : $P_1 < P_0$. В первом случае величина V, вычисленная по формуле (16), будет положительной. Это означает, что получена информация, которая будет способствовать достижению цели. Во втором случае V = 0, т.е. мы не получили никакой новой информации. В третьем случае V < 0 – это означает, что полученная информация будет мешать достижению цели. В этом случае мы имеем дело с дезинформацией. Значение V, найденное по формуле (6.1), изменяется в пределах от ($-\infty$; Vmax), где $Vmax = Log_2(1/P_0)$.

Другой способ измерения ценности V информации предложен В.И. Корогодиным:

$$V = \frac{P_1 - P_0}{1 - P_0} \qquad (6.2)$$

Смысл обозначений такой же, как и в (6.1): P_0 и P_1 – априорная и апостериорная вероятности достижения (осуществления) цели Z. Значения V принадлежат интервалу (0 ; 1) при условии $P_1 > P_0$. Если $P_1 = P_0$, то V = 0; если $P_1 = 1$, то V = 1.

Значения V не могут принимать отрицательные значения, т.к. ситуация, при которой $P_1 < P_0$, может возникнуть лишь в двух случаях. Первый, когда объект, поставляющий или использующий информацию, стремится уменьшить вероятность осуществления некоторого события. Тогда цель для него - неосуществление Z, вероятность чего $P' = 1- P_0$, и в этом случае P_1, которое меньше, чем P_0, будет превышать значение P', и, следовательно, требование V > 0 будет соблюдено. Второй случай – это ошибочное использование неподходящей информации, что требует коррекции, а не логического анализа. Ситуация с “сознательным обманом” целиком включается в первый случай.

Можно сделать вывод, что помимо количества информации, измерять и выражать в цифрах можно и такое ее свойство, как ценность. В основе определения ценности информации лежат такие ее свойства, как

операциональность и эмерджентность, а также предложенный А. А. Харкевичем способ исчисления ценности через приращение вероятности достижения той цели, для чего данная информация используется.

Можно утверждать, что в отличие от количества, ценность информации невозможно задать одним единственным числом. Ценность каждой информации имеет определенное значение лишь по отношению к некоторой данной ситуации и данной цели: по отношению к разным парам “ситуация-цель” ценность любой информации может варьировать в некоторых пределах, например, от 0 до 1. Следовательно, мы никогда не сможем иметь исчерпывающие сведения о ценности какой-либо информации – сколь бы ни представлялась она ничтожной, всегда остается надежда, что могут существовать такие ситуации и цели, где эта ценность близка к максимальной, т.е. к единице.

Поэтому ценность информации, сообщаемой во время лекции преподавателем, неодинакова для каждого из судентов из-за существующих различий в тезаурусе и целях этих студентов. Здесь можно провести аналогию с вычислением работы силы по перемещению в заданном направлении: «вектор информации» и «вектор личных целей» могут быть сонаправлены, перпендикулярны или составлять острый угол.

7. История развития вычислительной техники.

В практике повседневной жизни человеку приходится решать различные проблемы или задачи, многие из которых возникают в неизменном виде и достаточно регулярно. Появляется потребность в разработке типовых подходов и правил решения, часто повторяющихся проблем и задач. Набор правил, направленный на решение задачи и состоящий в выполнении некоторых простых, типизированных действий, называется алгоритмом. Однако для успешного решения задачи кроме алгоритма необходим ещё и его исполнитель. Достаточно давно возникла идея поручить выполнение алгоритма, если это возможно, машине. Нам, живущим в XXI веке, такие машины хорошо известны: всевозможные бытовые устройства (стиральные машины, кухонные комбайны), устройства связи, машины (роботы) промышленного производства, работающие на конвейере и т.п. Однако, исторически первыми появились устройства для выполнения вычислительных алгоритмов, и это случилось достаточно давно.

Одним из первых известных устройств, предназначенных для проведения вычислений, является абак, что означает «счётная доска». Предполагают, что абак впервые появился в Древнем Вавилоне около 3 тыс. до н. э. Первоначально он представлял собой доску, разграфлённую на полосы или со сделанными углублениями. Счётные марки (камешки, косточки) передвигались по линиям или углублениям. В 5 в. до н. э. в Египте вместо линий и углублений стали использовать палочки и проволоку с нанизанными камешками. На этом этапе абак использовался скорее для запоминания промежуточных результатов в цепочке вычислений. Начиная с IV в. до н.э., абак использовался для выполнения арифметических вычислений в древнегреческой и древнеримской

цивилизациях. В России аналогом абака явились «русские счёты». Они появились в XVI веке и применяются до настоящего времени.

Следующий этап развития характеризуется созданием вычислительных устройств на механической основе с применением шестерней. Среди разработчиков и создателей таких машин следует отметить Блеза Паскаля, Готфрида Лейбница, Чарльза Бэббиджа. Каждый из них внес в процесс развития вычислительной техники свои оригинальные идеи, которые используются и в современных ЭВМ.

Первую суммирующую 8–ми разрядную машину построил Блез Паскаль (1641-1645). Он наладил производство таких машин. Каждой цифре от 0 до 9 соответствовал угол поворота счётного колеса. Всего было восемь таких колес. Операция умножения заменялась многократным сложением. Вклад Паскаля в развитие вычислительной техники, не утративший своего значения, состоял в том, что он первым догадался заменить операцию вычитания сложением с дополнением вычитаемого. Этот способ выполнения вычитания и сейчас используется в современных процессорах.

Лейбниц (около 1673 г.) создал первый арифмометр, который выполнял все четыре арифметических действия. Он первым предложил выполнять вычисления в двоичной системе счисления (на уровне проекта). Авторство в создании двоичной системы также приписывается Лейбницу. Двоичное представление данных и двоичная арифметика лежат в основе работы современных компьютеров. Арифмометр Лейбница был более «продвинутым» устройством по сравнению с машиной Паскаля. Вклад Лейбница в развитие ВТ высоко оценил Норберт Винер, один из идейных разработчиков первой ЭВМ.

Следующий значительный шаг в деле создания вычислительных машин был сделан Чарльзом Бэббиджем в начале XIX века. Конструктивно

машина Бэббиджа аналогична современным ЭВМ. Она содержала следующие элементы:

- «Склад» для хранения чисел (устройство хранения данных в современных ЭВМ).
- «Фабрика» – вычислительное устройство (ВУ), выполняющее операции над числами (в современных ЭВМ ему соответствует процессор).
- Устройство управления (УУ) - также присутствует в современных ЭВМ.
- Устройство ввода-вывода (УВВ) данных – на печать и на перфокарты.

Перфокарта была изобретена Жозефом Жаккардом в 1801 г. и применялась для управления работой ткацкого станка. Позднее Герман Холлерит применил перфокарту для обработки данных по переписи населения в 1890 г. Эти работы привели в дальнейшем к созданию корпорации IBM. Перфокарты использовались в практике программирования для ввода программ и данных в ЭВМ вплоть до 80-х годов XX века.

Полностью реализовать свои идеи по созданию вычислительной машины Бэббиджу не позволил технологический уровень того времени. Передовыми достижением для того времени явились способ ввода алгоритма в машину с помощью перфокарт и сама возможность изменять алгоритм работы машины. Тогда же впервые возникла проблема составления программ и впервые возникла идея создания библиотеки программ для вычислительной машины. Рядом с Бэббиджем появляется ещё один исторически важный персонаж – леди Ада Лавлейс (1815–1852) , дочь Байрона. Она занималась разработкой алгоритмов и программ для машины Бэббиджа и считается первым в мире программистом. Ей приписывают создание алгоритма вычисления чисел Бернулли и изобретение команды для разветвления вычислительного процесса. В 1840 г. Бэббидж ездил по приглашению итальянских математиков в

Турин, где читал лекции о своей машине. Был издан конспект этих лекций на французском языке. Позже Ада Лавлейс перевела эти лекции на английский язык, дополнив их комментариями, которые по своему объёму превосходили исходный текст. В комментариях Ада сделала описание машины Бэббиджа и инструкции по программированию к ней. Это были первые в мире программы, поэтому Аду Лавлейс справедливо считают первым программистом. В восьмидесятых годах XX века был разработан язык программирования, который назвали «Ада», в честь Ады Лавлейс.

Следующий этап в истории создания ЭВМ связан с именем Конрада Цузе (1910 - 1995). Он считается создателем первой работающей программируемой ЭВМ и первого языка программирования высокого уровня.

К. Цузе проектировал самолёты в компании Henschel Aircraft. Ему приходилось выполнять огромные объёмы вычислений. Цузе решил автоматизировать процесс вычислений. В 1934 г. Цузе придумал модель автоматического калькулятора, которая состояла из УУ, ВУ, памяти и полностью совпадала с архитектурой современных компьютеров. Он сформулировал шесть принципов работы компьютеров:

1. должна использоваться двоичная система счисления;
2. должны использоваться устройства, работающие по принципу да/нет;
3. должен быть полностью автоматизирован процесс работы ВУ;
4. процесс вычислений должен управляться программно;
5. необходима поддержка арифметики с плавающей запятой, а не только с фиксированной;
6. следует использовать память большой ёмкости.

В период с 1938 по 1944 г. Цузе создал три модели вычислительных машин Z1, Z2, Z3. Модель Z1представляла собой двоичное механическое вычислительное устройство с электрическим приводом и возможностью программирования при помощи клавиатуры. Результат вычислений

отображался на ламповой панели. Это была экспериментальная модель. Машина Z2 считывала инструкции с перфорированной 35-миллиметровой киноплёнки. Модель Z3 сегодня многие считают первым, реально действовавшим программируемым компьютером. Порядок вычислений теперь можно было определять заранее, однако условные переходы и циклы отсутствовали. В сентябре 1950 года Цузе сконструировал машину Z4. В то время Z4 был единственным работающим компьютером в Европе и первым компьютером в мире, который был продан. Цузе первым разработал язык программирования, не привязанный к архитектуре ЭВМ (1966 г).

Важнейшей вехой в развитии вычислительной техники явилось создание в Пенсильванском университете первой ЭВМ под руководством Дж. Маучли и Преспера Эккерта. Проект стартовал в 1943 г. при поддержке Лаборатории баллистических исследований для расчётов таблиц стрельбы армии США, а уже в 1946 (1945) была продемонстрирована ЭВМ ENIAC (от Electronic Numerical Integrator and Automatic Calculator). Это был первый широкомасштабный, электронный, цифровой компьютер, способный быть перепрограммированным для решения целого диапазона задач. Его отдельные характеристики: потребляемая мощность — 150 кВт., вычислительная мощность — 300 операций умножения или 5000 операций сложения в секунду, вес - 27 тонн. Вычисления производились в десятичной системе.

Разработка второй ЭВМ началось ещё до окончательного запуска ENIAC. В группу разработчиков был включён Дж. фон Нейман. ЭВМ известна под аббревиатурой EDVAC (Electronic Discrete Variable Automatic Computer). В отличие от ENIAC, это был первый компьютер с хранимой в памяти программой, который работал в двоичной, а не десятичной системе счисления. Приведём основные технические характеристики EDVAC.

Компьютер располагал встроенными операциями сложения, вычитания и умножения, а также программной реализацией деления; объём памяти составлял 5,5 килобайт в современной терминологии. Основные конструктивные компоненты EDVAC:

- устройство чтения/записи с магнитной ленты;
- контролирующее устройство с осциллографом;
- устройство-диспетчер, принимающее инструкции от контролирующего устройства и из памяти и направляющее их в другие устройства;
- вычислительное устройство, выполняющее за раз одну арифметическую операцию над парой чисел и посылающее результат в память;
- таймер;
- три временных регистра, в каждом из которых хранилось одно слово.

Время выполнения операции сложения — 864 микросекунды, умножения — 2900 микросекунд. Компьютер состоял из почти 6000 электровакуумных ламп, и 12000 диодов, и потреблял 56 кВт энергии. Занимаемая площадь — 45,5 м², масса — 7850 кг. Полный состав обслуживающего персонала — 30 человек на каждую 8-часовую смену.

У истоков создания первой ЭВМ стояли многие учёные. В частности, Н. Винер в своей книге «Кибернетика» перечисляет следующие принципы конструирования ЭВМ.

1) Центральные суммирующие и множительные устройства должны быть цифровыми, как в обычном арифмометре, а не основываться на измерении (как в дифференциальном анализаторе Буша).

2) Эти устройства, являющиеся по существу переключателями, должны состоять из электронных ламп, а не из зубчатых передач или электромеханических реле. Это необходимо, чтобы обеспечить достаточное быстродействие.

3) В соответствии с принципами, принятыми для ряда существующих машин Белловских телефонных лабораторий, должна использоваться более экономичная двоичная, а не десятичная система счисления.

4) Последовательность действий должна планироваться самой машиной так, чтобы человек не вмешивался в процесс решения задачи с момента введения исходных данных до снятия окончательных результатов. Все логические операции, необходимые для этого, должна выполнять сама машина.

5) Машина должна содержать устройство для запасания данных. Это устройство должно быстро их записывать, надежно хранить до стирания, быстро считывать, быстро стирать их и немедленно подготавливаться к запасанию нового материала.

Как пишет Н. Винер: «Все эти рекомендации представляют собой идеи, положенные в основу современной сверхбыстрой вычислительной машины. Эти мысли почти носились тогда в воздухе, и я не хочу в данный момент заявлять какие-либо претензии на исключительный приоритет в их формулировке. Все же указанные рекомендации оказались полезными, и я надеюсь, что они имели некоторое влияние на популяризацию этого круга идей среди инженеров».

Другой известный учёный, Дж. фон Нейман, при конструировании ЭВМ EDVAC сформулировал ряд требований, которым должна удовлетворять ЭВМ. С тех пор (1945 г.) эти требования известны как принципы Джона фон Неймана, положенные в основу архитектуры современных ЭВМ.

1. Основные блоки ЭВМ: устройство управления (УУ), арифметико-логическое устройство (АЛУ), оперативная память (ОП или ОЗУ), внешнее запоминающее устройство (ВЗУ), устройство ввода-вывода данных (УВВ).

2. УУ и АЛУ объединяются в единое устройство, называемое процессором.
3. Алгоритм решения задачи (программа) представлен в виде последовательности управляющих слов – команд, которые определяют смысл выполняемой операции. Последовательность (совокупность) команд образует программу.
4. Команда – совокупность сведений, необходимых процессору для выполнения определённого действия. Адресный принцип состоит в том, что в команде указываются не сами числа, над которыми надо выполнить действия, а их адреса в ОП.
5. Структура команды (первые ЭВМ были 3-х адресными) имеет вид:

КОП	А1	А2	А3

Здесь КОП – код выполняемой операции (инструкция для процессора); А1, А2, А3 – адреса операндов. Например, команда могла содержать инструкцию: сложить числа, хранящиеся по адресам А1, А2, результат записать по адресу А3.

6. Данные и программа кодируются в двоичной системе счисления и хранятся в оперативной памяти (ОП) ЭВМ. Процессор определяет действия, подлежащие выполнению путём считывания команд из ОП. Порядок команд определяется программой.
7. После ввода программы и данных машина работает сама, без вмешательства человека. ЭВМ запоминает адрес выполняемой команды. Каждая команда содержит адрес следующей команды. Возможные варианты: переход к следующей, переход по заданному адресу (команда безусловного перехода), условный переход.

Все современные компьютеры по своей структуре являются Неймановскими машинами. Принцип «невмешательства» человека в процесс вычислений в современной практике часто нарушается – человек

может управлять ходом вычислительных процессов, менять параметры выполняемых алгоритмов и т.п.

В СССР первая ЭВМ была запущена в регулярную эксплуатацию в 1951 г. под руководством С.М. Лебедева. Эта машина известна под названием МЭСМ – малая электронно-счетная машина. В 1953 г. С.М Лебедевым была запущена самая производительная на тот момент в Европе ЭВМ – БЭСМ (большая электронно-счётная машина).

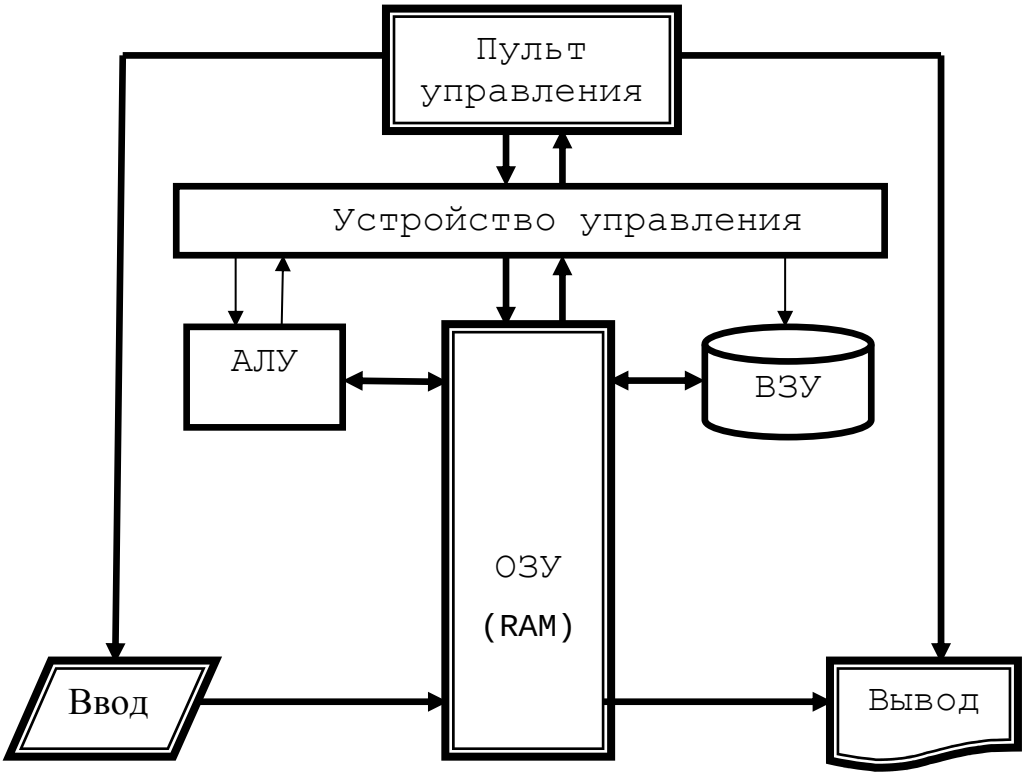


Рис. 8. Принципиальная схема ЭВМ Дж. фон Неймана.

Принцип работы ЭВМ (с шинной организацией) становится ясным после рассмотрения алгоритма и схемы работы УУ. Номер пункта соответствует номеру в кружочке на рисунке 9.

1. В память ЭВМ загружена программа. Счётчик адреса команд (САК) содержит адрес первой команды.

2. Процессор (ЦП) считывает команду из оперативной памяти (ОП), используя адрес из САК. Команда поступает в регистр команд.
3. Из прочитанной команды выделяется код операции (КОП) и адреса ОП, по которым хранятся операнды команды. КОП поступает в блок управления операциями ЭВМ.
4. Продолжается чтение команды (данных), если это необходимо. Длина команды прибавляется к содержимому САК. Теперь САК содержит адрес команды, которая будет выполняться на следующем шаге.
5. По адресам операндов, выделенных из текущей команды, считываются данные из ОЗУ и поступают в АЛУ.
6. КОП передаётся в АЛУ, где производятся вычисления.
7. Полученный результат записывается в память ЭВМ. Длина команды прибавляется к содержимому САК. Теперь САК содержит адрес команды, которая будет выполняться на следующем шаге.

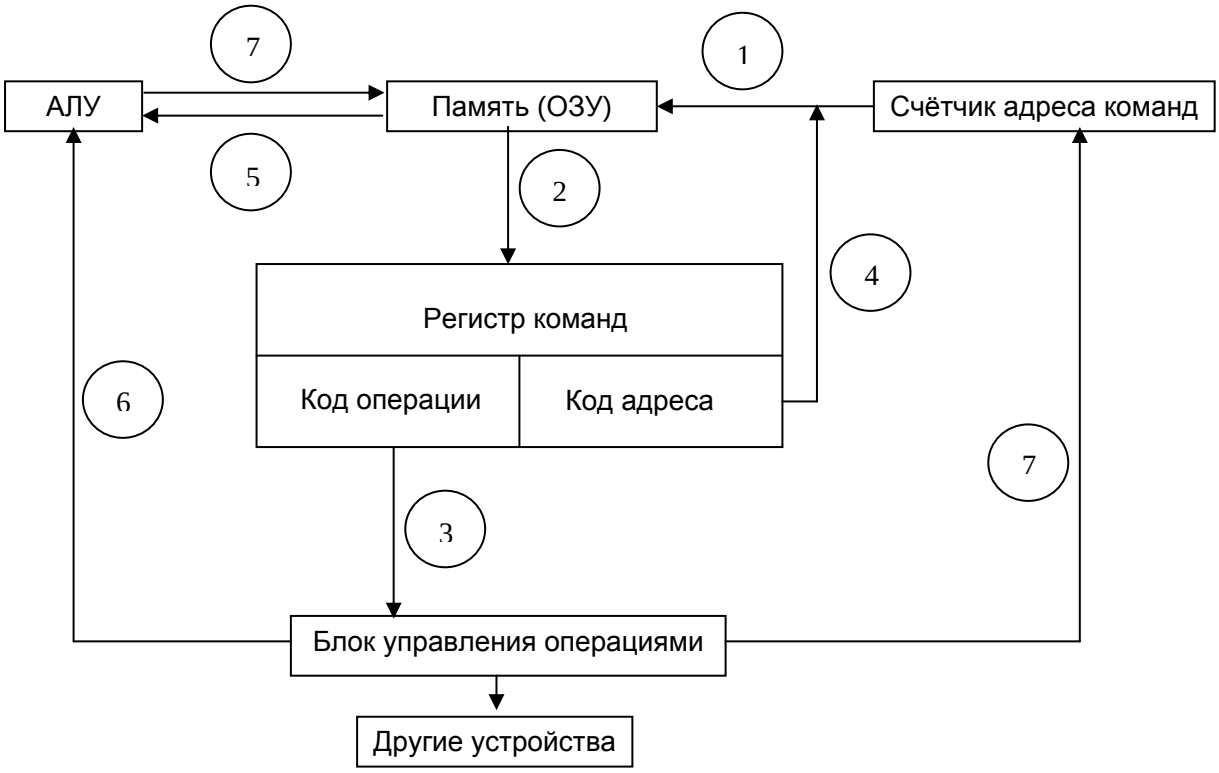


Рис. 9. Схема работы УУ.

Историю развития вычислительной техники с момента запуска первой ЭВМ и до настоящего времени принято описывать в терминах «поколений» ЭВМ. Деление компьютерной техники на поколения — весьма условная, нестрогая классификация вычислительных систем по степени развития аппаратных и программных средств, а также способам общения с компьютером.

Идея делить машины на поколения вызвана к жизни тем, что за время короткой истории своего развития компьютерная техника проделала большую эволюцию как в смысле элементной базы (лампы, транзисторы, микросхемы и др.), так и в смысле изменения её структуры. Стали появляться новые возможности, расширялись области применения, изменился характера использования ЭВМ.

Принято выделять до пяти поколений. Это деление достаточно условно и оправдывает себя в основном в отношении ЭВМ в промежутке времени от 1946 – 1990 г. XX века. В настоящее время скорость модернизации компьютеров столь высока, что любая попытка провести классификацию в современных условиях устаревает прежде своего осуществления. Приведём краткие характеристики каждого поколения.

Первое поколение ЭВМ. К первому поколению обычно относят машины, созданные на рубеже 50-х годов. В их схемах использовались электронные лампы. Эти компьютеры были огромными, неудобными и слишком дорогими машинами, которые могли приобрести только крупные корпорации и правительства. Лампы потребляли огромное количество электроэнергии и выделяли много тепла. Набор команд был небольшой, схема арифметико-логического устройства и устройства управления достаточно проста, программное обеспечение практически отсутствовало. Для ввода-вывода использовались перфоленты, перфокарты, магнитные ленты и печатающие устройства. Быстродействие порядка 10-20 тысяч операций в секунду. Ёмкость памяти от 2 до 8 килобайт. Но это только

техническая сторона. Очень важна и другая — способы использования компьютеров, стиль программирования, особенности математического обеспечения. Программы для этих машин писались на языке конкретной машины. Математик, составивший программу, садился за пульт управления машины, вводил и отлаживал программы, производил по ним вычисления. Процесс отладки был наиболее длительным по времени. Несмотря на ограниченные возможности, эти машины позволили выполнить сложнейшие расчёты, необходимые для прогнозирования погоды, решения задач атомной энергетики и др. Опыт использования машин первого поколения показал, что существует огромный разрыв между временем, затрачиваемым на разработку программ и временем счета. Эти проблемы начали преодолевать путем интенсивной разработки средств автоматизации программирования, создания систем обслуживающих программ, упрощающих работу на машине и увеличивающих эффективность её использования. Это, в свою очередь, потребовало значительных изменений в структуре компьютеров. Отечественные машины первого поколения: МЭСМ (малая электронная счётная машина), БЭСМ, Стрела, Урал, М-20.

Второе поколение ЭВМ — машины, сконструированные примерно в 1955-1965 г. Характеризуются использованием в них как электронных ламп, так и дискретных транзисторных логических элементов. Их оперативная память была построена на магнитных сердечниках. В это время стал расширяться диапазон применяемого оборудования ввода-вывода, появились высокопроизводительные устройства для работы с магнитными лентами, магнитные барабаны и первые магнитные диски. Быстродействие — до сотен тысяч операций в секунду, ёмкость памяти — до 100 Кбайт.

Совершился переход от программирования в машинных кодах к программированию на алгоритмических языках. Появились языки

высокого уровня (Fortran, Algol), средства которых допускают описание всей необходимой последовательности вычислительных действий в наглядном, легко воспринимаемом виде. Программа, написанная на алгоритмическом языке, непонятна компьютеру, воспринимающему только язык своих собственных команд. Поэтому специальные программы, которые называются трансляторами, переводят программу с языка высокого уровня на машинный язык. Появился широкий набор библиотечных программ для решения разнообразных математических задач.

Появились мониторные системы, управляющие режимом трансляции и исполнения программ. Из мониторных систем в дальнейшем выросли современные операционные системы. Операционная система — важнейшая часть программного обеспечения компьютера, предназначенная для автоматизации планирования и организации процесса обработки программ, ввода-вывода и управления данными, распределения ресурсов, подготовки и отладки программ, других вспомогательных операций обслуживания. Таким образом, операционная система является программным расширением устройства управления компьютера. Для некоторых машин второго поколения уже были созданы операционные системы с ограниченными возможностями.

Машинам второго поколения была свойственна программная несовместимость, которая затрудняла организацию крупных информационных систем. Поэтому в середине 60-х годов наметился переход к созданию компьютеров, программно совместимых и построенных на микроэлектронной технологической базе. К середине 1960-х годов мировой парк машин второго поколения оценивался в 30 000 штук.

Машины третьего поколения созданы после 60-х годов. Поскольку процесс создания компьютерной техники шел непрерывно, и в нём

участвовало множество людей из разных стран, имеющих дело с решением различных проблем, трудно и бесполезно пытаться установить, когда «поколение» начиналось и заканчивалось. Возможно, наиболее важным критерием различия машин второго и третьего поколений является критерий, основанный на понятии архитектуры. Машины третьего поколения — это семейства машин с единой архитектурой, т.е. программно совместимых. Другой важный признак: в качестве элементной базы в ЭВМ третьего поколения используются интегральные схемы, которые также называются микросхемами. Отличием интегральных схем от транзисторных с дискретными компонентами является объёмное или общее поверхностное формирование транзисторов, диодов, сопротивлений.

Машины третьего поколения имеют развитые операционные системы. Они обладают возможностями мультипрограммирования, т.е. одновременного выполнения нескольких программ. Многие задачи управления памятью, устройствами и ресурсами стала брать на себя операционная система или же непосредственно сама машина. Примеры машин третьего поколения — семейства IBM-360, IBM-370, ЕС ЭВМ (Единая система ЭВМ), СМ ЭВМ (Семейство малых ЭВМ). Быстродействие машин внутри семейства изменяется от нескольких десятков тысяч до нескольких миллионов операций в секунду. Ёмкость оперативной памяти достигает нескольких мегабайт.

Четвёртое поколение ЭВМ — это поколение компьютерной техники, разработанное после 1970 года. При проектировании машин четвёртого поколения использовались большие интегральные схемы - БИС. В аппаратурном отношении для них характерно широкое использование интегральных схем в качестве элементной базы, а также наличие быстродействующих запоминающих устройств с произвольной выборкой ёмкостью в десятки мегабайт. Наиболее важный в концептуальном

отношении критерий, по которому эти компьютеры можно отделить от машин третьего поколения, состоит в том, что машины четвёртого поколения проектировались в расчете на эффективное использование современных высокоуровневых языков программирования и упрощение процесса программирования для конечного пользователя.

С точки зрения структуры машины этого поколения представляют собой многопроцессорные и многомашинные комплексы, работающие на общую память и общее поле внешних устройств. Быстродействие составляет до нескольких десятков миллионов операций в секунду, ёмкость оперативной памяти порядка 1 - 64 Мбайт.

Для этого периода развития ВТ характерны: применение персональных компьютеров; телекоммуникационная обработка данных; компьютерные сети; широкое применение систем управления базами данных; появление отдельных элементов интеллектуального поведения систем обработки данных и устройств.

Пятое поколение ЭВМ – это машины, создание которых началось в конце 1980-х годов. Их разработка производится на основе больших интегральных схем повышенной степени интеграции, использования оптоэлектронных принципов (лазеры, голография).

Развитие идет также по пути «интеллектуализации» компьютеров, устранения барьера между человеком и компьютером. Компьютеры этого поколения способны воспринимать информацию с рукописного или печатного текста, с бланков, с человеческого голоса, узнавать пользователя по голосу, осуществлять перевод с одного языка на другой. В компьютерах пятого поколения произошёл качественный переход от обработки данных к обработке знаний. Решается проблема децентрализации вычислений с помощью компьютерных сетей, как больших, находящихся на значительном расстоянии друг от друга, так и миниатюрных компьютеров, размещённых на одном кристалле полупроводника. Некоторые

направления развития ВТ на современном этапе кажутся экзотическими, но предсказать их дальнейшую судьбу сложно. К таким направлениям следует отнести, например, разработку биокомпьютеров, оптических процессоров.

Вычислительная техника играет в становлении информатики основополагающую роль благодаря двум факторам. Во-первых, ВТ стала универсальным средством хранения, представления и обработки информации. Ушли в прошлое времена, когда ЭВМ использовалась только для выполнения вычислений для нужд учёных, промышленности или военных. При помощи компьютера мы слушаем музыку, смотрим фильмы, храним фотографии, читаем книги в электронном виде, делаем диагностику в медицине, получаем нужную информацию из Интернета. Во-вторых, теоретические основы ВТ и программирования стали общепризнанными разделами информатики и способствовали её дальнейшему развитию.

8. Основы формальной логики.

Формальная логика (математическая логика) является важнейшей основой функционирования современных компьютеров. Мы используем законы формальной логики, когда делаем запрос на поиск информации в Интернете или базе данных или когда анализируем правильность математического вывода. Многие элементы компьютеров проектируются и работают на основе законов формальной логики. Её общепризнанным создателем является английский математик Джордж Буль (1815 – 1864). До работ Буля логика всегда считалась одним из разделов философии. Основы логики, как науки о законах и формах мышления, были заложены ещё в работах Аристотеля в 384 г. до н.э. Аристотель ввёл понятие силлогизма, сделав важный шаг в разработке логической дедукции и формализации логических рассуждений. Когда мы говорим о формальной логике, то имеем в виду, что анализируется правильность формы высказываний и умозаключений, а не их конкретное содержание.

Достижения Аристотеля в области логики не претерпели существенных изменений вплоть до XVII века. Впервые идеи обоснования логики на основе вычислений, подобно тому как мы оперируем символами в алгебре, были высказаны ещё в XVII веке Готфридом Лейбницем. Идеи Лейбница реализовал в своих работах Дж. Буль. В 1847 г. он опубликовал работу "Математический анализ логики", в которой высказал идею, что логика более близка к математике, чем к философии. Эта работа была чрезвычайно высоко оценена английским математиком Августом Де Морганом, который преподавал математику для Ады Лавлейс. В 1854 году Буль опубликовал работу "Исследование законов мышления, базирующихся на математической логике и теории вероятностей". Эти исследования Буля заложили основы алгебры логики или булевой алгебры. Ученый первым показал, что существует аналогия между

алгебраическими и логическими действиями. Буль придумал систему обозначений и правил, пользуясь которыми, можно было закодировать любые высказывания, а затем манипулировать ими как обычными алгебраическими выражениями. Именно поэтому логику Буля часто называют математической логикой или Булевой алгеброй.

Рассмотрим основные понятия логики: суждение, понятие, простые и сложные высказывания. С помощью **понятий** мы раскрываем значение естественных или искусственных знаков, указываем классы, к которым принадлежат или не принадлежат мыслимые нами вещи. Умственное развитие – это способность переосмысливать старые и конструировать новые понятия. Только понятия делают нашу речь осмысленной. Мы имеем понятие о некоторой вещи, если знаем и можем словесно выразить, какие условия **необходимы и достаточны** для её однозначного определения.

Условие, определяющие некоторый класс вещей, называется необходимым, если все вещи из этого класса и, возможно, некоторые вещи из его дополнения удовлетворяют этому условию. Условие, определяющее некоторый класс вещей, называется достаточным, если некоторые (может быть и все) вещи из этого класса удовлетворяют этому условию, но ни одна вещь из дополнения класса не удовлетворяет ему.

В терминах свойств можно определить необходимость некоторого условия следующим образом: если некоторая вещь не может существовать без данного свойства, то это свойство необходимо для её существования. Если же из существования некоторого свойства можно сделать вывод о существовании некоторой вещи, то это свойство достаточно для этой вещи.

Суждения позволяют нам выразить разнообразные отношения между мыслимыми вещами. Мы имеем суждение о некоторой вещи, если можем выразить словесно её отношение к другой вещи или к себе самой.

Основная языковая форма суждения – повествовательное предложение. Суждение может быть истинным, ложным или неопределённым. Суждение (высказывание) является простым, если ни одна его часть не может рассматриваться как суждение. Простые суждения принято обозначать буквами: A, B, C, D

Любое простое суждение состоит из 4-х функционально различимых частей:

1) субъекта суждения (S) – класс вещей, о котором нечто утверждается;
2) предиката суждения (P) – класс вещей, который утверждается относительно субъекта; предикат выражает то, что утверждается относительно S;
3) утвердительной или отрицательной связки «есть» или «не есть», которая ставится между S и P;
4) слов «все», «некоторые», «ни один», которые ставятся перед субъектом.

$$\left\{\begin{matrix} все \\ некоторые \\ ни\ один \end{matrix}\right\} S \quad \left\{\begin{matrix} есть \\ не\ есть \end{matrix}\right\} \quad P \qquad (8.1)$$

Если простое суждение имеет форму, отличную от (1), то его можно преобразовать к этой форме.

Все простые суждения классифицируются как утвердительные и отрицательные, которые, в свою очередь, делятся на общие и частные: « Все студенты ходят на лекции», «Некоторые преподаватели читают детективы», «Ни один человек не хочет быть несчастливым», «Некоторые фрукты не растут в России».

Суждение истинно, если в нём утверждается связь между объектом и признаком, имеющая место в действительности, или отрицается связь, не имеющая места в действительности. **Суждение ложно**, если в нём утверждается связь между объектом и признаком, не имеющая места в действительности, или отрицается связь, которая имеет место в

действительности. Истину и ложь обозначают по–разному: 1 или 0, Т или F (от английского True и False), И или Л. Мы будем пользоваться обозначением 1, если суждение истинно: А =1. Ложное суждение обозначим нулём: В =0.

Сложные суждения состоят из нескольких простых, соединённых различными логическими союзами: «неверно, что А», « В и С», «А или D», « если В, то С », «или А или В». Например, «сегодня тихо и пасмурно». Сложные суждения можно выразить через простые, но не наоборот.

Связка «не А» или «неверно, что А» называется отрицанием. Отрицание суждения А является истинным, если А ложно, и ложным, если А истинно. Обозначают отрицание суждения А как «не А» или $\overline{A}$, или not(A). Правила вычисления логической операции «отрицание» можно задать с помощью таблицы:

А	не А
1	0
0	1

Связка «и» называется **конъюнкцией** высказываний А и В и принимает значений истина, только когда оба высказывания истинны, в других случаях конъюнкция принимает значение ложь. Обозначают конъюнкцию суждений А и В как A&B, A and B (в программировании), A ∧ B (в учебниках по логике). Правила вычисления конъюнкции зададим с помощью таблицы истинности:

А	В	А ∧ В
1	1	1
1	0	0
0	1	0
0	0	0

Конъюнкцию иногда называют логическим умножением: результат в третьем столбце формально можно получить как произведение чисел из 1-ого и 2-ого столбцов.

Логически связка «или» называется дизъюнкцией. **Дизъюнкция** двух высказываний A и B принимает значение «истина», если хотя бы одно из высказываний истинно, и значение «ложь», если оба высказывания ложны. Для дизъюнкции используют обозначение A ∨ B, в программировании используют обозначение A or B. Таблица истинности для дизъюнкции имеет вид:

A	B	A ∨ B
1	1	1
1	0	1
0	1	1
0	0	0

Дизъюнкцию иногда называют логическим сложением. Связка «или» в дизъюнкции не имеет исключающего характера. В логике используется так называемая **сильная дизъюнкция** (исключающее или), которая на русском языке выражается с помощью связки «или A, или B». Она носит исключающий характер, т.к. принимает значение истина, когда операнды имеют разное логическое значение. Обозначается в программировании как A xor B; таблица истинности имеет вид:

A	B	A xor B
1	1	0
1	0	1
0	1	1
0	0	0

Очевидно, что конъюнкция, дизъюнкция и сильная дизъюнкция являются коммутативными операциями, т.е.

$$A \wedge B = B \wedge A; \quad A \vee B = B \vee A; \quad A \text{ xor } B = B \text{ xor } A.$$

Справедливость этих формул следует из таблиц истинности: при перестановке первых двух столбцов таблицы мы получим в третьем столбце значения, совпадающие со значениями третьего столбца исходной таблицы. Сложные суждения эквивалентны, если они принимают одинаковые логические значения при одинаковых значения простых высказываний, входящих в них.

Логическая связка «если…, то ….» называется **импликацией.** Импликация высказываний «если А, то В» принимает значение ложь только в одном случае, когда А истинно, а В ложно. Импликация обозначается как $A \rightarrow B$. Суждение А называется посылкой, В следствием. Иногда употребляются термины антецедент для посылки А и консеквент – для заключения В. Таблица истинности для импликации имеет вид:

A	B	$A \rightarrow B$
1	1	1
1	0	0
0	1	1
0	0	1

Импликация не обладает свойством коммутативности. Это следует из таблицы истинности: если переставить столбцы 1 и 2, то значения в третьем столбце изменятся. Многие теоремы в математике имеют форму импликации. При доказательстве теорем вида $A \rightarrow B$ мы доказываем, что ситуация, в которой из верной посылки А можно вывести ложное заключение В, невозможна. Например, «Если числовой ряд сходится, то

его n–ый член стремится к нулю». Если теорема A →B имеет место, то говорят, что B является логическим следствием A.

Эквиваленция – это логическая связка, которая выражается словами «A тогда и только тогда, когда B », «для A необходимо и достаточно B». Эквиваленция обозначается как A↔B и выражается через импликацию и конъюнкцию:

$$A \leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A)$$

Эквиваленция принимает значение истина в случае, когда оба высказывания имеют одинаковые значения. Таблица истинности для эквиваленции имеет вид:

A	B	A ↔ B
1	1	1
1	0	0
0	1	0
0	0	1

Многие теоремы в математике имеют форму эквиваленции. Такие теоремы называются критериям. Например, «Скалярное произведение ненулевых векторов P и Q равно нулю тогда и только тогда, когда эти векторы перпендикулярны».

Из нескольких простых высказываний с помощью логических операций можно составить более сложные высказывания. Для указания порядка выполнения логических действий можно использовать круглые скобки. Для однозначного прочтения логических выражений принят следующий приоритет выполнения операций (перечислены в порядке убывания приоритета): отрицание, конъюнкция, дизъюнкция, сильная дизъюнкция, импликация, эквиваленция. Отрицание – самая «сильная» операция. Например,

$$A \wedge B \vee C = (A \wedge B) \vee C; \qquad \bar{A} \vee B \rightarrow C = ((\bar{A}) \vee B) \rightarrow C;$$

С помощью знака = (равно) будем обозначать **равносильные высказывания** – высказывания, которые принимают одинаковые логические значения при одинаковых значения простых высказываний, входящих в них. Логическое значение сложного высказывания определяется логическими значениями входящих в него простых высказываний. Например, требуется вычислить логическое значение сложного высказывания «не (A ∧ B) ∨ (не C)» в случае, если A = 1, B = 1, C=0. Подставим на место простых высказываний их значения. Тогда A ∧ B = 1, не (A ∧ B) = 0, (не C) = 1, дизъюнкция 0 ∨ 1 = 1. Заданное высказывание истинно при заданных значениях A, B, C.

Для определения всех возможных значений сложного высказывания, в зависимости от всевозможных значений входящих в него элементарных высказываний, можно построить таблицу истинности. В этой таблице для каждого простого высказывания, входящего в заданное сложное высказывание, надо создать отдельный столбец. Затем нужно заполнить строки таблицы для простых высказываний всевозможными комбинациями их логических значений. Если число простых высказываний равно n, то таких комбинаций будет 2^n . Затем надо представить сложное высказывание в виде комбинации более простых, но также сложных высказываний и завести для каждого из них свой столбец. Один столбец (обычно последний) заводим для заданного высказывания. Заполняем все строки полученной таблицы. Например, пусть задано высказывание (формула)

не A ∨ B → A ∧ не B.

Требуется составить для данного высказывания таблицу истинности. Запишем данную формулу с применением скобок: ((не A) ∨ B) → (A ∧ (не B)). В таблице будет четыре строки, т.к. простых высказываний два: A и B.

A	B	не A	не B	не A ∨ B	A ∧ не B	(не A ∨ B) → (A ∧ не B)
1	1	0	0	1	0	0
1	0	0	1	0	1	1
0	1	1	0	1	0	0
0	0	1	1	1	0	0

Отметим, что заданная формула эквивалентна формуле «не (A → B)» (см. таблицу истинности для импликации), так как принимает одинаковые с ней логические значения при одинаковых значениях простых суждений, входящих в эти формулы.

Перечислим основные правила преобразования логических выражений. Эти правила используются для упрощения заданных формул с целью получения из них более простых и эквивалентных им выражений.

1) $\overline{\overline{A}} = A$ (закон двойного отрицания).

2) $A \vee \overline{A} = 1$ (закон исключённого третьего)

3) $A \wedge A = A$

4) $A \wedge 1 = A$

5) $A \wedge 0 = 0$

6) $A \vee A = A$

7) $A \vee 1 = 1$

8) $A \vee 0 = A$

9) $A \wedge \overline{A} = 0$

10) $A \wedge (B \vee A) = A$

11) $A \vee (B \wedge A) = A$

Правила выражения одних логических операций через другие:

1) $A \to B = \overline{A} \vee B$

2) $\overline{A \wedge B} = \overline{A} \vee \overline{B}$ (закон де Моргана)

3) $\overline{A \vee B} = \overline{A} \wedge \overline{B}$ (закон де Моргана)

4) $A \leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A)$

В логике доказывается теорема, в которой утверждается, что все логические операции можно выразить через отрицание, конъюнкцию и дизъюнкцию. Говорят, что эти операции образуют полную систему логических операций.

Для операций конъюнкции и дизъюнкции имеют место свойства коммутативности, ассоциативности и дистрибутивности:

1) $A \wedge B = B \wedge A$; - коммутативность.
2) $A \vee B = B \vee A$; - коммутативность.
3) $A \wedge (B \wedge C) = (A \wedge B) \wedge C$;- ассоциативность.
4) $A \vee (B \vee C) = (A \vee B) \vee C$; - ассоциативность.
5) $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$; - дистрибутивность.
6) $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$; - дистрибутивность.

Рассмотрим пример равносильных преобразований. Упростить формулу, используя перечисленные выше свойства и правила преобразования логических выражений:

$$(\overline{A \vee B} \rightarrow A \vee B) \wedge B$$

Выполним цепочку равносильных преобразований:

$$(\overline{\overline{A \vee B}} \vee A \vee B) \wedge B = (A \vee B \vee A \vee B) \wedge B = (A \vee B) \wedge B = B .$$

В XX веке в математической логике произошли важные изменения: впервые со времен своего возникновения логика стала многозначной. В многозначной логике высказывания могут иметь более двух истинностных значений. В 1920 г. Ян Лукасевич разработал трёхзначную логику. В ней высказывания могут принимать три значения: «истина», «ложь» и «может быть» или «неопределено». В такой логике не действует закон исключенного третьего. В 1921 г. Э. Пост выдвинул идею многозначной логики. В k – значной логике высказывания могут принимать значения от 0 до к-1, где k=3,4, 5… и т.д.

9. Системы счисления.

Система счисления (СС) – совокупность приёмов и правил для изображения чисел с помощью символов, имеющих определённые количественные значения. В любой системе счисления выделяют некоторые числа, которые называют узловыми; другие числа – алгоритмические, получаются в результате выполнения каких-либо операций над узловыми. Например, в римской системе счисления узловыми являются числа I(1), V(5), X(10), L(50), C(100), D(500), M(1000). Алгоритмическими числами являются, например, числа II, III, IV, VI, VII, VIII, IX, LX(60) , CXXI(121). Если в римской СС меньшее узловое число стоит слева от другого узлового числа, то оно вычитается, а если справа – то прибавляется к соседнему. Например, XII = 10+2; XIX= X + IX = 10+9; XL = 50 - 10 = 40 и т.д. Существуют алфавитные СС, в которых для записи чисел используются буквы. Например, в церковно-славянском языке для записи чисел используются буквы: А(1), В(2), Г(3), Д(4), Е(5), S(6), I (10), К(20). Тогда, например, ІД = 14, КЕ = 25.

Системы счисления делятся на **позиционные и непозиционные**. В непозиционных системах счисления количественное значение каждой цифры не зависит от её позиции в записи числа. Непозиционные СС возникли раньше позиционных систем. Римская СС является непозиционной: например, в записи чисел XV и VII значение символа V (5) не зависит от его позиции в записи чисел.

Если значение числового символа зависит от его расположения в записи числа, то такая СС называется позиционной. Её изобретение, приписываемое шумерам и вавилонянам, имело неоценимые последствия в истории человеческой цивилизации. Дальнейшим развитием позиционной системы занимались индусы. Позиционные системы более удобны для

вычислений, чем непозиционные. В настоящее время используются в основном позиционные системы счисления.

К числу таких систем относится современная десятичная система счисления, возникновение которой связано со счётом на пальцах. В средневековой Европе она появилась благодаря итальянским купцам, которые в свою очередь заимствовали её у мусульман. «Позиционность» десятичной системы поясним на примере. В записи числа 525.351 цифра 5 встречается три раза, и её значение зависит от позиции в записи. Первая позиция слева - это количество сотен (500), третья слева - это количество единиц (5), а вторая после десятичной точки - это количество сотых долей (5/100) или ($5*10^{-2}$). Любая позиционная СС характеризуется своим основанием – количеством знаков, используемых для изображения чисел.

Для нас привычной является десятичная система счисления с основанием 10. В ней для записи чисел используются десять различных цифр: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. Основание позиционной СС не обязательно равно 10, а может быть любым натуральным числом большим 1. Например, если СС имеет основание 3, то используются три цифры {0,1, 2}, а числа могут записываться в виде 211_3, 102.21_3 – нижний индекс указывает на основание СС, в которой сделана запись числа. Далее будем обозначать 2-ую, 8-ую и 16-ую СС как Bin, Oct и Hex соответственно. Десятичную систему счисления обозначим как Dec.

Приведём общее правило для изображения чисел в позиционной системе счисления. Пусть задано основание системы счисления – натуральное число B >1. Задан алфавит (цифры) системы счисления, содержащий B символов. Пусть M ={A1, A2, …, AB} - алфавит системы счисления; а R_k – знаки алфавита M, k = -n, -n +1, …. 0, 1, 2, … m, где m, n - неотрицательные целые числа. Здесь индекс k указывает на позицию в записи числа, в которой стоит знак R_k .

Позиционной записью числа X в системе счисления с основанием B называется его представление в виде:

$$X = R_m R_{m-1} R_{m-2} \ldots R_1 R_0 \bullet R_{-1} R_{-2} \ldots R_{-n} \qquad (9.1)$$

В формуле (9.1) символ «•» - разделитель целой и дробной части числа X; обычно это точка « . » или запятая « , ». Номер позиции цифры в записи числа определяется относительно разделителя: влево или вправо. Отсчёт влево идёт с нуля. В формуле (9.1) R_4 означает, что эта цифра находится на 5-ой позиции, если считать от разделителя влево (5, а не 4, так как отсчёт идет с нуля); R_{-3} означает, что эта цифра стоит на третьей позиции, если считать от разделителя вправо.

Запись (9.1) означает, что число X равно:

$X= R_m*B^m + R_{m-1}*B^{m-1} + R_{m-2}*B^{m-2} +\ldots R_1*B^1 +R_0* B^0+R_{-1}*B^{-1} +R_{-2}*B^{-2} +\ldots R_{-n} * B^{-n}$,

где $B^0=1$; $B^m = B*B*\ldots..B$, т.е. B умножаем на себя m - раз; $B^{-n} = 1/B^n$.

Более компактная запись формулы (9.1) :

$$X = \sum_{k=-n}^{m} R_k * B^k \ . \qquad (9.2)$$

Например, для «обычной» десятичной системы счисления с основанием B=10 и M={0,1,2,3,4,5,6,7,8,9}, запись (9.1) для X= 3269.721 означает, что $X=3*10^3 + 2*10^2 + 6*10^1 + 9*10^0 + 7*10^{-1} + 2*10^{-2} + 1*10^{-3}$.

Основание позиционной СС – это количество единиц младшего разряда, переходящее в единицу старшего соседнего разряда. Используя это правило, выпишем первые 16 натуральных чисел в двоичной системе счисления (Bin) , используя «таблицу сложения» 1+0 = 1; 1+1 = 10:

0+1 = 1	1	100+1 =101	5	1000+1=1001	9	1100+1=1101	13
1+1 = 10	2	101 +1 =110	6	1001+1=1010	10	1101+1=1110	14
10+1 = 11	3	110 +1 = 111	7	1010+1=1011	11	1110+1=1111	15
11+1 = 100	4	111+1 =1000	8	1011+1=1100	12	1111+1= 10000	16

В 8-ой СС (Oct) используются первые восемь цифр десятичной системы {0, 1, 2, 3, 4, 5, 6, 7}. Выполняются соотношения

7+1=10	8	13+1=14	12
10+1=11	9	14+1=15	13
11+1=12	10	15+1=16	14
12+1=13	11	16+1=17	15
		17+1=20	16

В Hex используются десять цифр 0, 1, … 9, а недостающие цифры изображают с помощью букв A, B, C, D, E, F: A= 9+1, B=A+1, C=B+1, D=C+1, E=D+1, F=E+1, $F+1=10_{16}$.

В сводной **таблице 1** представлены натуральные числа от 1 до 16 в 2- ой , 8-ой и 16-ой системах счисления. Обратите внимание на то, что в таблице для записи двоичного представления чисел всегда используются 4 бита, хотя для записи чисел от 1 до 7 достаточно 1, 2 или 3 бит. Такое 4-х битовое представление нам понадобится в дальнейшем при переводе чисел из Bin в Hex, и наоборот.

Таблица 1.

Dec	Bin	Oct	Hex
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5

6	0110	06	6
7	0111	07	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

Пример 1. Приведём пример позиционной записи (1) для чисел в *Bin*. Рассмотрим двоичное число:

11100101.101

Это дробное число; для его перевода в *Dec* надо использовать формулу (9.2):

$2^7 + 2^6 + 2^5 + 2^2 + 2^0 + 1/2 + 1/2^3 =$

128 + 64 + 32 + 4 + 1 + 0.5 +0.125 = 229. 625.

Таким образом, формула (2) даёт способ перевода чисел из *Bin* в *Dec*.

Пример 2. Перевести из *Bin* в *Dec* число 101.011. Запишем данное число в виде (9.2):

$101.011 = 1*2^2 + 0*2^1 + 1*2^0 + 0*2^{-1} + 1*2^{-2} + 1*2^{-3} = 4+0+1+1/2 +1/4=4.75$

Алгоритм перевод чисел из *Dec* в *Bin* поясним на примерах.

Пример 2. Перевести из Dec в Bin число **42.73**

Отдельно переведем целую и дробную части числа.

а) Для перевода целой части (= 42) проводим ряд последовательных делений десятичного числа 42 на 2. При каждом таком делении находим **частное (r)** и **остаток (q).** На каждом шаге полученное частное вновь

делим на 2 и т.д. Процесс останавливается, когда при очередном делении частное будет равно 0.

42 : 2 = 21 **(0)** r = 21, q = 0

21 : 2 = 10 **(1)** r = 10, q = 1

10 : 2 = 5 **(0)** r = 5, q = 0

5 : 2 = 2 **(1)** r = 2, q = 1

2 : 2 = 1 **(0)** r = 1, q = 0

1 : 2 = 0 **(1)** r = 0, q = 1 (Stop)

Полученные остатки q выписываем, начиная с самого последнего и до первого. Это и будет двоичный код числа 42: $42_{10} = 101010_2$.

b) Для перевода дробной части 0.73 заданного десятичного числа в *Bin* проводим ряд последовательных умножений на 2. В полученном произведении на каждом шаге отделяем целую часть (r) от дробной (q). Дробную часть вновь умножаем на 2 и т.д. Процесс заканчивается, если на очередном шаге дробная часть будет равна нулю. Здесь возможны случаи, при которых очередная дробная часть q уже встречалась на предыдущих шагах процесса, то есть мы получим периодическую двоичную дробь. Если такой момент не наступает, мы имеем дело с бесконечной двоичной дробью. Выписываем полученные целые части r в порядке их появления **от первой до последней** - это и будет искомая двоичная дробь.

0. 73 * 2 = 1. (46) **r =1**, q= 0.46

0. 46 * 2 = 0. (92) **r =0**, q= 0.92

0. 92 * 2 = 1. (84) **r =1**, q= 0.84

0. 84 * 2 = 1. (68) **r =1**, q= 0.68

0. 68 * 2 = 1. (36) **r =1**, q= 0.36

и т.д. Мы прервали перевод дроби 0.73 и получили её приближенное представление в виде дроби системы *Bin*: $0.73 \approx 0.10111\ldots$

Окончательно получаем: $42.73 \approx 101010.10111\ldots$

Пример 2.

Записать в *Bin* десятичное число 0.15 (в виде двоичной дроби).

0.15 * 2 = 0. (3) r = 0, q = 0.3

0.3 * 2 = 0. (6) r = 0, q = 0.6

0.6 * 2 = 1. (2) r = 1, q = 0.2

0.2 * 2 = 0. (4) r = 0, q = 0.4

0.4 * 2 = 0. (8) r = 0, q = 0.8

0.8 * 2 = 1. (6) r = 1, q = 0.6

0.6 * 2 = 1. (2) r = 1, q = 0.2 – повторение: см. строку 3.

Получили представление числа 0.15 в виде двоичной периодической дроби:

0.15 = 0.00100110011001…= 0.00(1001).

Пример 3. Если знаменатель дроби является степенью двойки, то можно использовать следующий приём для перевода дроби из *Dec* в *Bin*. Покажем, как можно записать в двоичной системе дроби 3/4, 5/8, 13/16:

$3 = 11_2$, $4 = 100_2 \rightarrow 3/4 = (11/100)_2 = 0.11$;

$5 = 101_2$, $8 = 1000_2 \rightarrow 5/8 = (101/1000)_2 = 0.101$;

$13 = 1101_2$, $16 = 10000_2 \rightarrow 13/16 = (1101/10000)_2 = 0.1101$

Техника перевода чисел из *Hex* в *Dec* такая же, как и в случае перевода из *Bin* в *Dec*.

Пример 4. Задано дробное число 3FA.B5 в *Hex*. Записать это число в *Dec*. Используем формулу (9.2):

$3FA.B5 = 3*16^2 + F*16^1 + A*16^0 + B*16^{-1} + 5*16^{-2}$

$= 3*256 + 15*16 + 10 + 11/16 + 5/256$

$= 768+240+10+(11*16+5)/256 = 1018 + 181/256 \approx 1018.70703125…$

Пример 5. Перевести десятичное число 332 в *Hex*.

Как и при переводе из Dec в Bin делим заданное целое на 16 и при каждом таком делении находим **частное (r)** и **остаток (q).** Полученное частное вновь делим на 2 и т.д. Если остаток $10 \le q \le 15$, то заменяем его

на соответствующие буквенные изображения цифр *Hex*, см таблицу 1. Затем выписываем остатки по порядку «снизу – вверх»:

332 : 16 = 20 **(12), 12 это C_{16}**

20 : 16 = 1 **(4)**

1 : 16 = 0 **(1)** (Stop)

$332_{10} = 14C_{16}$.

Дробные числа из *Dec* переводятся в *Hex*, как и в случае перевода из *Dec* в *Bin*. Например, переведём дробь 0.13 в *Hex*:

0.13*16 = (2). 08

0.08*16 = (1). 28

0.28*16 = (4). 48

0.48*16 = (7). 68

0.68*16 = (A). 88, $A_{16}=10$

0.88*16 = (E). 08, E = 14, остаток 08 уже встречался, см. 2 строку.

Т.е. при переводе числа 0.13 в Hex мы получили периодическую дробь: $0.13 = 0.2147AE147AE\ldots = 0.2(147AE)_{16}$.

При переводе дробей, знаменатель которых является степенью 16, можно использовать более простой способ. Например, переведём дробь 11/16 в Hex: $11=B_{16}$, $16=10_{16}$; тогда $(11/16) = (B/10)_{16} = 0.B$; переведём дробь 167/256 в Hex: $167 = A7_{16}$; $256 = 100_{16}$; значит (167/256) = (A7/100) = 0.A7.

Для перевода чисел из *Bin* в *Hex* применяется следующий приём: двоичную запись числа надо разбить на группы цифр по 4 (на тетрады), начиная от разделителя влево и вправо (если число дробное). Самую левую и самую правую тетрады дополняем, если надо, нулями. Каждую двоичную тетраду нужно заменить на соответствующую цифру в Hex, используя таблицу 1. Пример:

111011001.110111 = 1 1101 1001 . 1101 11 =

0001 1101 1001 . 1101 11**00 = 1D9. DC .**

Перевод из *Hex* в *Bin* осуществляется ещё проще: каждую цифру числа, записанного в *Hex*, нужно заменить на соответствующую тетраду из таблицы 1. Например, A7C.2F = 1010 0111 1100. 0010 1111;

Значение систем *Bin* и *Hex* для информатики определяется тем, что все данные и программы представлены в компьютере в двоичной системе. Для большей «читабельности» двоичных кодов их выводят на экран в *Hex*; адреса памяти записываются в *Hex*; дампы памяти представляют в *Hex*.

Для записи чисел в компьютере используются два основных способа: **с фиксированной запятой и с плавающей точкой.**

Систему представления чисел с фиксированной запятой обозначают как P(b, t, f), где b – основание системы счисления, t – количество разрядов для записи числа, f - количество разрядов для записи дробной части.

Рассмотрим пример системы с фиксированной запятой P(10, 4, 1); здесь b=10, t=4, f=1. Т.е. это Dec, общее количество разрядов для записи числа равно 4, под дробную часть отводится 1 знак. Минимальное число, которое можно записать в этой системе, равно -999.9, максимальное число равно +999.9, всего положительных чисел с нулём 10000, всего отрицательных чисел 9 999. Любое число из [-1000; 1000] представимо в системе P(10, 4, 1) с ошибкой ≤ 0.05

Пример. Пусть задано число x=865.54. Его представление в P(10, 4, 1) обозначим как p(x) = 865.5. Ошибка равна | x - p(x)| = 0.04. Относительная ошибка представления этого числа E = | x - p(x)| /x = 0.04 / 865.54 = 0.000046213, что составляет ≈ 0.004%.

Если x = 0.86554, то p(x) = 0.9 (произвели округление при записи числа в системе P(10, 4, 1)), относительная ошибка E = 0.03446/0.86554 = 0.039813…, что составляет ≈3.98%.

Вывод: система P(b, t, f) образует равномерную сетку для представления вещественных чисел, но относительная ошибка при записи вещественных чисел в этой системе не является равномерной.

Система P(b, t, f) используется в современных ЭВМ **только для записи целых чисел в двоичной системе** (со знаком и без знака), то есть при b = 2, f = 0. Для записи целых чисел в современных компьютерах может отводиться, например, 8, 16 или 32 бита, т.е. t=8, 16, 32. Выделенные для записи чисел биты нумеруются справа налево, начиная с нуля. Если целое число со знаком, то старший бит хранит знак числа: 0 соответствует знаку плюс, 1 соответствует знаку минус.

Пример. Целые числа со знаком в системе P(2, 8, 0). Для их записи отводится 8 бит. Старший (седьмой) бит содержит знак числа: 0 или 1. Вот как будет записано число $+75 = 2^6+2^3+2^1+2^0$ в 8-битовом коде:

0100 1011.

Степени двойки в записи числа +75 указывают на то, что бит с соответствующим номером содержит 1.

7 бит (знак)	6 бит	5 бит	4 бит	3 бит	2 бит	1 бит	0 бит
0	**1**	**0**	**0**	**1**	**0**	**1**	**1**

Для записи отрицательных целых чисел используется **дополнительный код**. Правило для записи отрицательного числа (-N) в дополнительном коде:

1) записать двоичный код соответствующего положительного целого N и дополнить его слева до нужного числа битов нулями (до 8, или до 16, или до 32 битов).

2) инвертировать полученный код (операция инверсии двоичного кода заключается в замене всех 0 на 1 и всех 1 на 0);

3) прибавить к младшему разряду инвертированного кода 1. Полученный код и будет дополнительным кодом отрицательного числа (-N).

Пример 1. Записать дополнительный 8 битовый код числа -95. Решение: запишем 95 в двоичном коде: 95 = 64+16+8 +4+2+1 = 101 1111. Дополним этот код слева нулём (чтобы получилось 8 бит): 0101 1111. Инвертируем полученный код: 1010 0000. Прибавим 1 к младшему разряду данного кода: 1010 0001. Это и будет дополнительный двоичный код отрицательного числа -95. Для проверки сложим полученные коды:

+ 95 : 0 1 0 1 1 1 1 1

-95 : 1 0 1 0 0 0 0 1

сумма : (1) 0 0 0 0 0 0 0 0

Сумма единиц в 8-ом бите перешла в 9-ый бит, который отсутствует в записи числа – он теряется (говорят, что произошел перенос из знакового бита), и сумма становится равной 0.

Максимальное положительное 8-ми битовое число 0111 1111 = +127. Восьми битовое число 1000 0001 = -127. Минимальное отрицательное 8-ми битовое равно 1000 0000 = -128. Т.е. диапазон однобайтовых целых чисел со знаком от -128 до +127. Всего с помощью 8-ми бит можно представить 256 чисел, т.е. 2^8.

Для перевода отрицательных чисел из дополнительного кода поступаем также: инвертируем код отрицательного числа, прибавляем 1 к младшему разряду и получаем двоичный код соответствующего положительного числа.

Пример. Какое целое число со знаком записано в 8-ми битовом коде 1010 0011? Решение: Старший бит равен 1, значит это отрицательное число. Инвертируем данный код: 0101 1100; Прибавляем 1 к младшему разряду: 0101 1101. Переводим это число в *Dec*: $2^6+2^4+2^3+2^2+2^0$ = 64+16+8+4+1 = 93. Т.е. заданный дополнительный код соответствует числу -93.

В языке программирования Turbo Pascal используются, например, такие целые типы данных:

Тип данных	Размер памяти	Количество чисел	Диапазон
Byte	1 байт	$2^8 = 256$	0 ..255
ShortInt	1 байт	$2^8 = 256$	-128 .. 127
Word	2 байта	$2^{16} = 65\,536$	0 .. 65 535
Integer	2 байта	$2^{16} = 65\,536$	-32 768 .. 32 767

Пояснения к таблице. Типы ShortInt и Integer используются для записи целых со знаком, т.е. старший бит хранит знак числа; типы Byte и Word используются для записи целых без знака (неотрицательных); 1 байт = 8 бит; используя n бит можно записать 2^n различных чисел (доказывается по индукции). Покажем, что максимальное число, которое можно представить, используя n бит равно $(2^n – 1)$. Действительно, если имеется n бит для записи числа, то максимальное число получится в случае, когда все разряды (биты) равны 1:

$$x = \underbrace{111.....1}_{n\ раз}$$

Если к этому числу прибавить 1, получим число:

$$x+1 = 1\underbrace{000.....0}_{n\ раз} = 2^n,$$

т.е. $x = 2^n - 1$.

Для правильной интерпретации содержимого памяти нужно знать, какой тип данных в ней храниться, т.к. по «внешнему виду» отличить один тип от другого нельзя. Пусть, например, 1 байт памяти содержит следующие данные: 1001 1101. Если это число типа Byte, то оно равно $2^7+2^4+2^3+2^2+2^0= 128+16+8+4+1=157$. Если это целое число со знаком типа ShortInt, то для его определения надо инвертировать код: 0110 0010, затем прибавить 1: получим 0110 0011, т.е. $2^6+2^5+2^1+2^0= 64+32+2+1 = 99$, значит, данный байт содержит число - 99.

Система представления чисел с плавающей точкой **F(b, t, L, U)**. Такая система используется для записи вещественных чисел в современных ЭВМ. Здесь b – основание системы (для ЭВМ b равно 2), t - количество разрядов мантиссы, L , U – пределы изменений значений показателей порядка чисел в этой системе. Пример числа, записанного в виде числа с плавающей точкой: $0.31562781*10^5$; здесь 0.31562781 – мантисса числа, 5 – его порядок. Это число в «обычной», позиционной записи выглядит как 31562.781. Такой способ представления чисел позволяет компактно записывать числа из широкого диапазона значений, например:
$0.4671*10^{-15}$, или $0.8965*10^{12}$.

Числа системы F(b, t, L, U) имеют вид: $x = \pm M*b^k$, где M – мантисса числа x, b – основание системы счисления, k – порядок числа x. Развёрнутая запись числа x:

$$x = \pm \left(\frac{d_1}{b} + \frac{d_2}{b^2} + \frac{d_3}{b^3} + \ldots + \frac{d_t}{b^t} \right) * b^k , \qquad (9.3)$$

или, более компактно:

$$x = \pm 0.d_1d_2d_3.... d_t * b^k. \qquad (9.4)$$

Здесь d_1 , d_2 , d_3 , …. d_t - цифры системы счисления; они составляют мантиссу числа; t – разрядность мантиссы (количество знаков в мантиссе).

Должны выполняться следующие условия:

1) $1 \le d_1 < b$ (первая цифра мантиссы не должна равняться нулю);

2) $0 \le di < b$, i =2,3,…t ;

3) $L \le k \le U$.

Если мантисса удовлетворяет первому условию, то говорят, что число представлено в нормализованном виде (с нормализованной мантиссой).

Запись нуля в системе F(b, t, L, U): $0.000…0* b^L$.

Утверждение 1. Система F(b, t, L, U) содержит

$$2*(b-1)*b^{t-1}*(U-L+1) +1$$ различных чисел.

Утверждение 2. Числа системы F(b, t, L, U) образуют неравномерную сетку, но относительная плотность этой сетки равномерна (всюду одинакова).

Пример 1. Система F(10, 4, -2, 3). Здесь b=10, t=4, L= -2, U=3. Число x=865.54 в этой системе записывается в виде $F(x) = 0.8655*10^3$, а число y=0.86554 имеет вид $F(y) = 0.8655*10^0$. В обоих случаях относительная ошибка ≈ 0.005%. Действительно,

$$| F(x) – x| /x =0.04/865.54 =4/86554;$$

$$| F(y) –y | /y = 0.00004/0.86554 = 4/86554.$$

В отличие от системы с фиксированной точкой, здесь относительная ошибка представления чисел одинакова для всех чисел из заданного диапазона.

Пример 2. Системы F(2, 3, -1, 2). Здесь b=2 (двоичная система), t=3, L= -1, U=2. Выпишем все положительные числа этой системы. Сначала заполним столбец числами порядка 2^0; числа из соседних столбцов получаются умножением (делением) на 2.

числа порядка 2^{-1}	числа порядка 2^0	числа порядка 2^1	числа порядка 2^2
0.100 ~ 4/16 = ¼	0.100 ~ 4/8 =1/2	0.100 ~ 4/4 =1	0.100 ~ 4/2 = 2
0.101 ~ 5/16	0.101 ~ 5/8	0.101 ~ 5/4	0.101 ~ 5/2 = 2.5
0.110 ~ 6/16	0.110 ~ 6/8	0.110 ~ 6/4	0.110 ~ 6/2 = 3
0.111 ~ 7/16	0.111 ~ 7/8	0.111 ~ 7/4	0.111 ~ 7/2 = 3.5

Систему счисления с плавающей точкой **F(b, t, L, U)** можно описать с помощью чисел ε, σ, λ:

1) ε (машинное ε) – это минимальное положительное число, которое будучи сложенным с 1, даёт результат ≠ 1. Вычислим его. Т.к. ближайшее справа к 1 число это 0.1000 ..1*b, значит ε равно

разности между этим числом и 1, т.е. $\varepsilon = 0.10\ldots1*b - 0.1*b = b^{1-t}$, для приведённого выше примера $\varepsilon = ¼$.

2) σ - минимальное положительное число системы **F(b, t, L, U).** Оно равно расстоянию между нулём и ближайшим к нему числом системы F(b, t, L, U). Это число представимо в виде: (минимальная мантисса) * b^L : $\boldsymbol{\sigma = 0.1*b^L = b^{L-1}}$**,** для приведённого выше примера $\sigma = ¼$;
3) λ – максимальное число системы **F(b, t, L, U).** Это число представимо в виде: λ = (максимальная мантисса) * b^U, т.е. $\boldsymbol{\lambda = 0.11\ldots1*b^U = b^U*(1 - 0.00\ldots1) = b^U*(1-b^{-t})}$**.** Для приведённого выше примера λ= 3.5.

Система чисел с плавающей точкой используется для представления дробных чисел в памяти ЭВМ. В самом общем виде структура представления вещественных чисел в ЭВМ имеет следующий вид:

Знак мантиссы	Знак порядка	Порядок числа	Мантисса

Для хранения знака мантиссы и знака порядка числа отводится по 1 биту; количество бит для хранения порядка и мантиссы зависит от выбранного типа представления данных. Например, если для хранения дробного числа отведено 32 бита, то их назначение может быть распределено следующим образом (в первой строке стоят номера битов, старший бит под номером 31 - знак числа, мантисса хранится в битах с номерами от 0 до 22, порядок числа в битах с номерами от 23 до 30):

31	30	29	…..	23	22	21	…..	1	0
Знак числа	Порядок числа				Мантисса				

Так как в нормализованной мантиссе двоичного числа в первой позиции всегда стоит единица, то хранят мантиссу, начиная со второго

знака. Например, число $0.1101*2^5$ хранится как $1.101*2^4$, при этом единица целой части числа не хранится. Корректировка происходит при преобразовании числа или в ходе вычислений. При этом порядок дробного двоичного числа при хранении занимает 1 байт и изменяется в диапазоне от -127 до +128, а не от -128 до +127, как было бы в случае нормализованного представления чисел. Например, если надо записать число $0.11*2^{-127}$, то представляем его как $1.1*2^{-128}$; число $0.11011*2^{+128}$ $=1.1011*2^{+127}$. Далее, чтобы порядок числа всегда был положительным к нему прибавляют 127, т.е. если порядок числа -10, то хранится +117. Порядок 100 хранится как 227. Такой способ записи порядка называют смещённым.

Примеры. 1. Как будет представлено в памяти компьютера число -0.0625, если для его размещения выделено 32 бита памяти?

Решение: $0.0625 = 1/16 = 0.0001_2$, т.е. $-0.0625 = -0.1*2^{-3} = -1.0*2^{-4}$;

смещенный порядок равен (-4+127) = 123, или в Bin $123 = 1111011_2$; мантисса = 0 (все 23 бита). Старший бит равен 1, т.к. число отрицательное. Окончательно получаем:

1 **0111 1011** 000 0000 0000 0000 0000 0000.

2. Запишите число 25 как число с плавающей запятой, используя 32 бита. Решение: $25 = 11001_2 = 1.1001*2^4$. Смещённый порядок равен 4+127 = 131 или 10000011_2 . Мантисса = 1001, окончательно получаем:

0 1000 0011 100 1000 0000 0000 0000 0000.

В языке TurboPascal существуют следующие типы данных для хранения вещественных чисел:

Тип данных	Размер памяти	Разрядн. мантиссы (бит)	Порядок величин
Real	6 байт	40	10^{38}
Single	4 байт	24	10^{38}
Double	8 байт	54	10^{308}
Extended	10 байт	67	10^{4932}

Очевидно, что множество вещественных чисел R значительно отличается от множества рациональных чисел, с которыми мы имеем дело в ЭВМ. Для чисел, представленных в ЭВМ можно указать наибольшее по модулю число A, такое что все остальные числа будут меньше, чем A. Кроме того, не выполняется свойство «всюду плотности» множества вещественных чисел, суть которого состоит в том, что для любых x, y ($x < y$) из множества R, можно указать значение z, такое, что $x < z < y$. Должна быть разработана теория «машинных» действительных чисел. Неточный перевод действительных чисел в двоичную систему и конечность разрядов, выделяемых для хранения чисел, служат источниками систематических ошибок при реализации многочисленных вычислительных методов на ЭВМ.

10 . Алгоритмы. Машина Тьюринга.

В первой половине IX века узбекский математик Абу-Джафар Мухаммед ибн Муса (787 - 850 г.) аль-Хорезми (из Хорезма) опубликовал свой трактат «Китаб аль-джебр вальмукабала», в котором были разработаны правила выполнения четырёх действий арифметики. Трактат был переведен на латинский язык в XII в. и оказал значительное влияние на развитие математики в Европе. Название трактата (аль-джебр) и место, где проживал его автор (аль-Хорезми), послужили источниками названий таких понятий как алгебра и алгоритм. Хотя алгоритмические процедуры были известны ещё со времён Евклида, наиболее точная формулировка понятия алгоритм, связанная с понятием машины Тьюринга, появилась только в XX веке.

Наиболее известный пример алгоритма даёт нам правило отыскания наибольшего общего делителя (НОД)[1] двух натуральных чисел, предложенное Евклидом в 3 в. до Р.Х.

Для любых натуральных чисел N и K определена операция деления с остатком: $N = p*K + q$; p – частное, q – остаток ($0 \le q < K$). Если $q = 0$, то говорят, что N делится на K нацело. Для вычисления остатка от деления целых чисел в математике (и программировании) применяется обозначение N mod K (читается «N по модулю K»).

Пример. 17 делим на 3: $17 = 5*3+2$. Здесь частное равно 5, остаток равен 2, то есть $17 \bmod 3 = 2$.

Итак, пусть заданы два натуральных числа N и K. Требуется найти их НОД.

Алгоритм Евклида решения задачи.

1) Вычислим $R = N \bmod K$;

---

[1] НОД двух чисел это наибольшее натуральное число, на которое делится каждое из этих чисел нацело.

2) присвоим N:=K,

3) присвоим K: = R;

4) если K ≠ 0, то вернуться к выполнению 1 шага.

5) нашли НОД = N.

6) Закончить алгоритм (Stop).

Продемонстрируем работу алгоритма на примере чисел N=112 и K=24.

1) R = 112 mod 24 → R = 16;

2) N = 24;

3) K = 16;

4) K≠ 0 ; опять вычисляем R = N mod K;

5) R = 24 mod 16 → R = 8;

6) N=16;

7) K = 8;

8) K≠ 0 → опять вычисляем R = N mod K;

9) R = 16 mod 8 → R = 0;

10) N = 8;

11) K = 0 ;

12) т.к. K = 0 → найден НОД: N = 8;

13) Stop.

Приведём интуитивное (неформальное) определение алгоритма из Математической энциклопедии, том 1.

Определение 1. Алгоритм - точное предписание, которое задаёт вычислительный (алгоритмический) процесс, начинающийся с произвольного исходного данного (из некоторой совокупности возможных

для данного алгоритма данных) и направленный на получение полностью определяемого этим исходным данным результата.

Здесь алгоритмический процесс - процесс последовательного преобразования так называемых конструктивных объектов (слов, чисел, пар слов …), происходящий дискретными шагами.

Определение 2. Алгоритм — это точное предписание, определяющее порядок действий исполнителя, направленное на решение поставленной задачи.

Определение 3 (определение А.И. Мальцева). Алгоритм - процесс последовательного построения величин, идущий в дискретном времени таким образом, что в начальный момент задаётся исходная система величин, а в каждый следующий момент система величин получается по определённому закону из системы величин, полученных в предыдущие моменты времени.

Приведённые определения алгоритма можно представить в виде схемы:

Исходные данные → Алгоритм → Результат.

Во всех определениях алгоритма неявно предполагается, что есть некто (или нечто), кто будет выполнять заданное предписание или процесс преобразование данных - исполнитель алгоритма. Исполнитель должен понимать команды алгоритма и уметь их выполнять. Исполнителем может являться человек, ЭВМ, техническое устройство и т.д.

Понятие алгоритма является нестрогим понятием. Однако, в конечном счёте, все-таки имеется общее понимание того, что является алгоритмом, а что не является. Одним из основных отличительных свойств алгоритма является то, что это предписание. Примерами алгоритмов могут служить правила сложения и умножения чисел, кулинарные рецепты, последовательность действий, направленных на достижение некоторого результата.

В качестве данных для алгоритма могут выступать так называемые конструктивные объекты. В качестве примера конструктивного объекта можно привести символ, логическое значение, числа (целые и вещественные). Конструктивный объект – элемент какого-либо конечного множества или объект, вычисленный некоторым алгоритмом.

Основные свойства алгоритма:

1) Дискретность шагов алгоритма: каждый шаг отделён от другого ненулевым отрезком времени.
2) Элементарность шагов: на каждом шаге алгоритма надо выполнить простые, понятные для исполнителя действия, причём за конечный промежуток времени.
3) Определённость: каждый шаг заканчивается определённым результатом, зависящим от исходных данных для этого шага; последовательность шагов алгоритма строго определена.
4) Конечность: для получения результата надо выполнить, быть может, большое, но конечное число шагов.
5) Массовость: входные данные алгоритма принадлежат некоторому множеству значений.

Для записи алгоритмов существуют различные способы. Одни из них ориентированы на исполнителя-человека, другие – на исполнение техническими устройствами, в частности компьютерами, роботами и т.д. То есть алгоритм задается в той форме, которая наиболее понятна исполнителю. Существуют следующие способы представления алгоритмов:

- словесный (описательный);
- формульный;
- графический;
- табличный или схематичный, в виде графа;
- в виде блок-схемы;

- в виде программы.

В обыденной жизни наиболее распространенным способом задания алгоритма являются описательный (словесный и словесно-пошаговый) способ. Когда объясняют, как добраться до нужного места, или описывают внешность незнакомого человека, с которым необходимо встретиться, то строится словесное описание алгоритмов поиска и «опознания». Хорошим примером такого задания алгоритма является кулинарный рецепт приготовления какого-либо блюда. Если в словесном описании четко выделены и пронумерованы элементарные шаги, то такое описание называется словесно-пошаговым. Например, алгоритм быстрого получения большого количества денег, который лиса Алиса и кот Базилио предложили Буратино: «В Стране Дураков есть волшебное поле, - называется Поле Чудес... На этом поле выкопай ямку, скажи три раза: "Крекс, фекс, пекс", положи в ямку золотой, засыпь землей, сверху посыпь солью, полей хорошенько и иди спать. Наутро из ямки вырастет небольшое деревце, на нем вместо листьев будут висеть золотые монеты. Понятно?». Этот алгоритм состоит из семи шагов, исходные данные – Поле Чудес и золотая монета.

Некоторые из способов записи алгоритмов предназначены только для исполнителей, обладающих определенными знаниями и умениями, исполнителей-специалистов. Например, для определения площади какой-либо сложной фигуры математик воспользуется алгоритмом вычисления определенного интеграла — это формульный способ записи алгоритма. Химик сможет получить нужное ему вещество, если известна формула соответствующей реакции. По этой же формуле он определит, какие исходные вещества и в каком количестве ему необходимы – это будут исходные данные алгоритма.

В информатике для записи алгоритмов широко используются блок-схемы. Блок-схема представляет собой систему связанных геометрических

фигур (блоков), каждая из которых обозначает один элементарный шаг алгоритма. Порядок выполнения шагов указывается стрелками, соединяющими блоки, которые стараются размещать сверху вниз, в порядке их выполнения. Для наглядности операции разного типа изображаются на схеме различными геометрическими фигурами, имеющими стандартный смысл: овалом обозначается начало и конец алгоритма, прямоугольником — присваивание значений переменным, ромбом — проверка условий, параллелограммом — операции ввода и вывода данных, кружочком или специальной стрелкой — операция перехода на другой лист или блок. Важной особенностью описания алгоритмов в виде блок-схем является «наполнение» каждого блока некоторыми формулами или пояснительным текстом.

Следует отметить, что нет однозначного соответствия между поставленной задачей и блок-схемой алгоритма ее решения. С одной стороны, так как все мы думаем и решаем задачи по-разному, то для одной задачи может быть составлено много алгоритмов (а, следовательно, и много блок-схем) ее решения. Если алгоритм разрабатывается для выполнения его с помощью компьютера, то он должен быть записан (представлен) на языке программирования. Алгоритм, представленный в таком виде, называется программой.

Программы строятся по строгим формальным правилам, в блок-схемах же допустимы обозначения, введенные самим пользователем. В этом состоит одно из отличий программы от блок-схемы. Блок-схема создаётся в предположении, что она будет восприниматься человеком, программа предназначена для «восприятия» её ЭВМ. Блок-схема не является обязательным этапом при переходе от алгоритма к программе. Однако, наличие блок-схемы, как правило, облегчает написание программы. Известны три типа алгоритмов – линейный, ветвящийся и

циклический. Тип алгоритма определяется характером решаемой в соответствии с его командами задачи.

Линейный тип алгоритма - алгоритм, в котором команды выполняются в порядке их естественного следования друг за другом, переход от одной команды к другой происходит только после выполнения предыдущей команды и независимо от каких–либо условий.

Таким будет, например, алгоритм вычислений по самым простейшим формулам, не имеющим ограничений на значения входящих в них переменных. Например, заданы координаты 2-х точек на плоскости. Надо вычислить расстояние между ними.

Ветвящийся тип алгоритма.

В том случае, когда условие задачи предусматривает в ходе ее решения возможность выбора в зависимости от выполнения некоторых условий, алгоритм решения называется разветвляющимся (ветвящимся). Например, решить уравнение a*x = b при заданных значениях коэффициентов a и b. Алгоритм решения такой задачи будет ветвящимся.

Циклический тип алгоритма.

Алгоритм, составленный с использованием многократных повторений одних и тех же действий, называется циклическим. Все циклические конструкции делятся на три основных типа: цикл с предусловием, цикл с постусловием, цикл со счётчиком. Например: заданы N точек на плоскости (x_i , y_i), i = 1,2,.. N. Найти длину соответствующей ломаной. Для реализации алгоритма решения такой задачи потребуется использовать цикл со счётчиком.

Понятие алгоритма, которое приводилось выше, имеет интуитивный, нестрогий характер. Интерес к алгоритмам возрос в связи с проблемами Давида Гильберта, которые он сформулировал в 1900 г. на Парижском конгрессе математиков. Одна из этих проблем заключалась в

отыскании универсальной процедуры для решения математических задач или в ответе на вопрос о принципиальной возможности такой процедуры.

Пусть А - посылка, В - следствие. Для любых А, В требуется узнать, существует ли дедукция, с помощью которой можно вывести В из А? Решением этой проблемы занимались многие математики и логики. Алан Тьюринг был одним из них. В 1928 г. Гильберт поставил задачу построения математики на незыблемом фундаменте из аксиом и правил вывода (логики), установленных раз и навсегда.

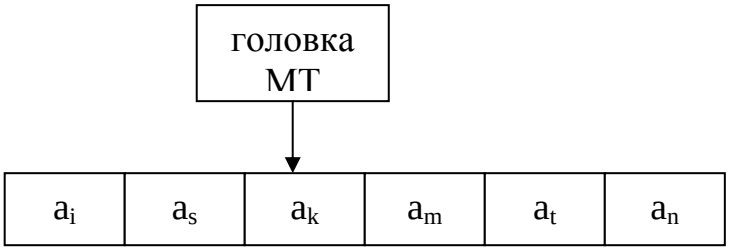
Сразу отметим, что все усилия, направленные на решение этой задачи были перечёркнуты великим открытием, которое сделал Курт Гёдель в 1931 г. Он доказал, что для любой полной и непротиворечивой аксиоматической системы можно сформулировать такие утверждения, которые не выводимы в ней (их нельзя ни доказать, ни опровергнуть в рамках таких систем). И даже проблема непротиворечивости системы аксиом, будучи сформулированной в виде теоремы, сама является недоказуемой в рамках такой системы. Примером такой аксиоматической теории является геометрия Евклида и проблема пятого постулата[1] о параллельных прямых. Эта проблема была снята после создания Н.И. Лобачевским в 1826 г. непротиворечивой геометрии, в которой отрицался пятый постулат.

Концепция универсального вычислительного устройства, известного как машина Тьюринга (МТ), была предложена в 1935-1936 годах математиком Аланом Тьюрингом как средство алгоритмического решения проблемы Д. Гильберта: существует ли универсальная механическая процедура позволяющая, в принципе, решить все математические задачи? Работая над этой проблемой, Тьюринг создал некоторую идеализацию

[1] Через точку Р вне прямой АВ в плоскости, проходящей через Р и АВ можно провести лишь одну прямую, не пересекающую АВ.

машины, выполняющей «механические процедуры». Следует подчеркнуть, что реально МТ не существует - это лишь идеальный (математический) объект.

МТ - это устройство, которое может принимать конечное, хотя, может быть, и очень большое число внутренних состояний, которое способно работать со сколь угодно большим объёмом данных. Эти данные располагаются на внешнем носителе, в качестве которого Тьюринг рассматривал **бесконечную** ленту, разбитую на отдельные «ячейки», в каждой из которых может храниться один символ. МТ имеет устройство (считывающую головку), которое может считывать символ из отдельной ячейки ленты и продвигать её влево и вправо относительно считывающей головки.



Обозначим через $A=\{a_0, a_1, a_2, a_3, \ldots a_n\}$ – алфавит МТ, $Q=\{ q_0, q_1, q_2, q_3, \ldots q_n \}$ – конечное множество состояний МТ. Работа МТ описывается следующим образом: если в некоторый момент времени машина находится в состоянии q_n и головка МТ обозревает символ a_m, то затем МТ записывает в текущую «ячейку» символ a_S, затем сдвигается на одну позицию влево (L) или вправо (R) или остаётся на месте (S), после чего переходит в состояние q_k. Описание работы МТ можно представить в виде: $q_n\ a_m \rightarrow a_s$ (L или R или S) q_k; q_1 – начальное состояние машины, q_0 – конечное состояние (останов МТ), a_0- символ пробела (пустая клетка).

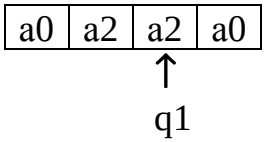
Набор таких правил (программу) для МТ обычно представляют в виде таблицы, например:

	a_0	a_1	a_2
q_1	a_2 L q_3	a_1 R q_2	a_2 L q_1
q_2	a_0 S q_2	a_2 S q_1	a_1 S q_2
q_3	a_0 R q_0	a_1 R q_4	a_2 S q_1
q_4	a_1 S q_3	a_0 R q_4	a_2 R q_4

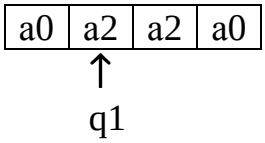
Эта машина может находиться в 5-ти состояниях – q0, q1, q2, q3, q4, а её алфавит состоит из трёх символов a0, a1 и a2. Заметим, что хотя бы одна ячейка таблицы должна содержать символ q0 – иначе МТ никогда не остановится.

Покажем, как работает эта МТ, если на входе у неё задана последовательность символов **a0, a2, a2, a0,** причём МТ в начальный момент обозревает второй символ справа – a2 и находится в состоянии q1.

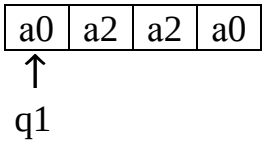
1 шаг (исходная позиция)



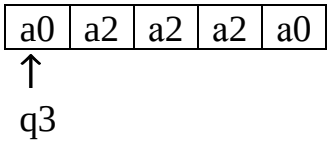
2 шаг



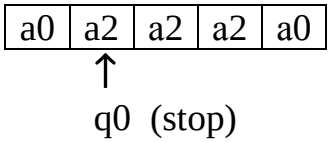
3 шаг



4 шаг

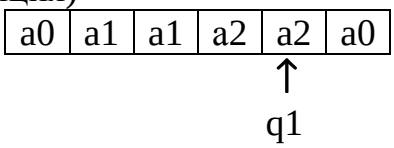


5 шаг

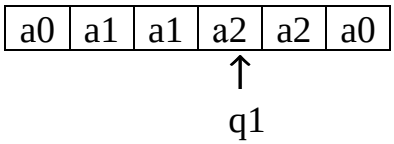


Другой пример: пусть той же МТ предъявлена на входе последовательность символов **a0, a1, a1, a2, a2, a0** и МТ обозревает второй справа символ a2 и находится в состоянии q1.

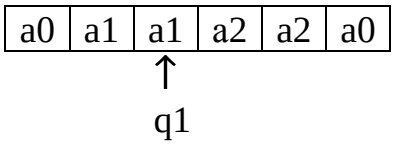
1 шаг (исходная позиция)



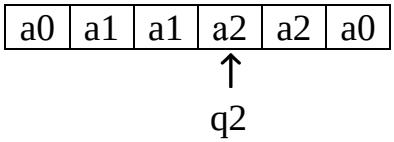
2 шаг



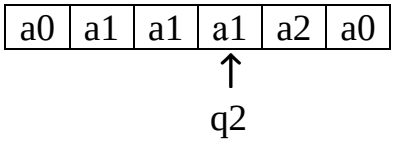
3 шаг



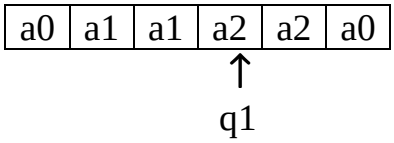
4 шаг



5 шаг



6 шаг



На 6 шаге полностью повторилась конфигурация 2 шага, т.е. произошло «зацикливание» работы МТ. Вывод: для такого исходного набора данных МТ никогда не остановится.

Устройство и работа машины Тьюринга по заданной программе удовлетворяют всем требованиям, предъявляемым к работе по заданному алгоритму. Единственное замечание – возможность «зацикливания»

конкретной МТ. Именно это сходство работы МТ и выполнения некоторого алгоритма позволило сформулировать знаменитый тезис Чёрча – Тьюринга, суть которого можно выразить следующим образом: «Для любого алгоритма можно построить машину Тьюринга, которая его реализует». Этот тезис нельзя доказать, поскольку нет формального, строгого определения алгоритма. Но его можно опровергнуть, если удастся привести пример алгоритма, для которого нельзя задать реализующую его МТ. До настоящего времени таких примеров построить не удалось. Наряду с машиной Тьюринга существует аналогичное «устройство» - машина Поста. Эти машины эквивалентны в том смысле, что алгоритм, который реализуем на МТ, может быть реализован и на машине Поста. Более того, в соответствии с тезисом Чёрча-Тьюринга, работа любого компьютера может быть воспроизведена на соответствующей МТ.

В настоящее время существует три основных направления в теории алгоритмов. Первое направление представлено такими учёными как А. Чёрч, Л. Гёдель, С. Клини и связано с уточнением понятия эффективно вычислимой функции. Функция F(x1,x2,x3, …. ,xn) от целочисленных аргументов x1,x2,x3, …. ,xn называется эффективно вычислимой, если существует алгоритм, позволяющий вычислять её значения. Это понятие также является интуитивным, так как опирается на нестрогое понятие алгоритма. Однако опираясь на строгое математическое определение частично рекурсивных функций и тезис А.Чёрча о том, что «Класс интуитивно вычислимых функций совпадает с классом частично рекурсивных функций», можно получить строго обоснованную теорию алгоритмов.

Второе направление связано с рассмотрением процессов, осуществляемых в вычислительной машине, и базируется на идеях, связанных с машиной Тьюринга. Третье направление связано с понятием

нормальных алгоритмов, введённым и разработанным российским математиком А.А. Марковым.

На современном этапе развития в теории алгоритмов возникли новые проблемы. В связи с появлением в программировании новых направлений: логического, функционального и объектно-ориентированного, - в теории алгоритмов, наряду с классическим «императивным» понятием алгоритма, были введены объектно-ориентированные и распределённые алгоритмы.

ООП «уравнивает» в правах алгоритмы и данные: алгоритм теперь – часть структуры данных. Развитию идеи распределённых алгоритмов способствовало тотальное расространение Интернета (WWW). Проблема данных, их представления и типа становится более серьёзной, чем проблема алгоритмов благодаря появлению суперструктуры всех файлов – WWW.

С понятием алгоритма тесно связана проблема «искусственного интеллекта». Проблемы искусственного интеллекта были поставлены задолго до появления первых ЭВМ. Однако прогресс в вычислительной технике только усилил интерес к этим проблемам. В самой общей постановке вопроса под искусственным интеллектом понимается умение имитировать различные аспекты деятельности человеческого разума. В отдельных направлениях разработки ИИ достигли впечатляющих результатов: в роботехнике, в экспертных системах, в игре в шахматы и т.д. Однако до настоящего времени полностью имитировать работу человеческого разума пока не удалось.

Алан Тьюринг предложил тест, с помощью которого можно было бы определить, что ИИ создан. Суть теста заключается в следующем: пусть имеется непроницаемый экран, за которым находится человек и компьютер. По другую сторону располагается другой человек (испытатель), который задаёт вопросы, вступает в контакт с расположенными по другую сторону экрана человеком или компьютером.

Человек пытается честно убедить испытателя, что он и есть живое существо, то же самое делает и компьютер. Испытатель не знает, кто отвечает ему – человек или компьютер. Он должен «разоблачить» машину. Естественно, что ответы должны выдаваться в обезличенной форме, например, на принтере. Если в ходе такого диалога испытателю не удастся определить, кто ему отвечал - человек или компьютер, то можно утверждать, что искусственный интеллект создан.

В вопросе о принципиальной возможности имитации человеческого разума с помощью машин есть как сторонники, так и противники. Сторонники идеи создания искусственного интеллекта (ИИ) опираются на понятие алгоритма и на тезис Чёрча-Тьюринга. Они утверждают, что любая деятельность человеческого мозга осуществляется в соответствии с некоторым алгоритмом. Следовательно, свойства разума присущи логическим действиям любого вычислительного устройства, т.к. умственная деятельность – это просто выполнение некоторой хорошо определённой последовательности операций, т.е. некоторого алгоритма.

Разница между работой мозга и более простого устройства (термостата) состоит лишь в сложности алгоритма. Раз существует алгоритм, по которому работает мозг, то можно создать соответствующую программу и запустить её на компьютере, т.е. получить «думающую» ЭВМ. Последовательное развитие идеи ИИ приводит к признанию возможности телепортации: закодировав с помощью программы работу сознания можно затем записать её на любой носитель, передать с помощью радиосигналов на удалённое расстояние, принять и воспроизвести на подходящем устройстве (компьютере).

Одним из известных противников теории сильного ИИ является Джон Сёрль. Ему принадлежит идея мысленного эксперимента под названием «Китайская комната», в котором критикуется возможность моделирования человеческого понимания естественного языка и, в

частности, критикуется тест Тьюринга. Сёрль считал, что при наличии исчерпывающего руководства по переходу от вопроса на китайском языке к ответу на этом же языке, человек способен поддерживать диалог на китайском языке, не понимая при этом по-китайски ни слова. Суть предлагаемого Сёрлем эксперимента состоит в следующем.

«Предположим, что меня поместили в комнату, в которой расставлены корзинки, полные китайских иероглифов и дали учебник на английском языке, в котором приводятся правила сочетания символов китайского языка, причём правила эти можно применять, зная лишь форму символов, понимать значение символов совсем необязательно. Представим себе, что находящиеся за дверью комнаты люди, понимающие китайский язык, передают в комнату наборы символов, и что в ответ я манипулирую символами согласно правилам и передаю обратно другие наборы символов. В данном случае книга правил есть не что иное, как «компьютерная программа». Люди, написавшие её, — «программисты», а я играю роль «компьютера». Корзинки, наполненные символами, — это «база данных»; наборы символов, передаваемых в комнату, это «вопросы»; а наборы, выходящие из комнаты, это «ответы».

Предположим далее, что книга правил написана так, что мои «ответы» на «вопросы» не отличаются от ответов человека, свободно владеющего китайским языком. Таким образом, я выдержу тест Тьюринга на понимание китайского языка. Но все же, на самом деле, я не понимаю ни слова по-китайски. Подобно компьютеру, я манипулирую символами, но не могу придать им какого бы то ни было смысла». Суть возражений Сёрля состоит в том, что правильное манипулирование символами с помощью компьютера ещё не означает, что компьютер понимает суть и смысл производимых манипуляций. В заключение отметим, что до сегодняшнего дня создать что-то, достойное называться подлинным интеллектом, пока никому не удалось.

11. Основы теории кодирования.

Проблема надёжной передачи данных по каналам связи - одна из основных проблем информатики. Любой процесс передачи данных (сигналов) описывается общей схемой, см. рис.1 на стр. 14. При передаче по каналам связи сигналы подвергаются искажениям, в связи с чем ставится задача обнаружения и исправления ошибок, возникших при передаче. Одним из важнейших разделов теории информации, в рамках которого решаются указанные проблемы, является теория кодирования, основы которой были заложены Ричардом Хэммингом (1915-1998). Не менее важный вклад в теорию кодирования сделал К. Шеннон. Ещё в годы Второй мировой войны он работал в шифровальном отделе и занимался проблемами надёжной передачи данных и выделения нужной информации из зашифрованного текста.

Постановка задачи надёжной передачи сообщений состоит в следующем. Пусть по каналу связи требуется передавать слова в некотором алфавите A. На вход канала подаётся слово α, а на выходе принимается искажённое слово ω. Требуется по слову ω восстановить слово α.

Основная идея решения этой задачи состоит в том, что вместо слова α передаётся его **код**: слово $\beta = K(\alpha)$. Здесь через $K(\alpha)$ обозначена функция преобразования слова α в его код β. Преобразование $K(\alpha)$ называется кодированием. Очевидно, должно существовать и обратное преобразование $\alpha = K^{-1}(\beta)$, которое называется декодированием.

Код должен позволять обнаруживать и, по возможности, исправлять ошибки, сделанные при передаче. Поэтому коды принято делить на два класса:

1) коды с обнаружением ошибок;

2) коды с исправлением ошибок.

Приведём пример простого кода с обнаружением одиночной ошибки. Рассмотрим сообщение α длины m над алфавитом B={0,1}:

$$\alpha = a_1\, a_2\, a_3 \ldots a_m, \text{ где } a_i \in B.$$

Код слова α :

$$\beta = K(\alpha) = b_1\, b_2\, b_3 \ldots b_m\, b_{m+1}, \text{ где } b_i = a_i, \text{ для } i = 1,2,\ldots m;$$

последний символ b_{m+1} равен сумме всех предыдущих по mod 2. Последнее означает, что $b_{m+1} = 0$, если сумма предыдущих символов чётная, и $b_{m+1} = 1$, если сумма предыдущих символов нечётная, так как для любого натурального k значение функции k mod 2 = 0 (для чётных k); либо =1 (для нечётных k)[1].

Значение символа b_{m+1}, вычисленное по данному правилу, называется контрольной суммой. Теперь для обнаружения ошибки в принятом слове длины m+1 надо вычислить контрольную сумму и сравнить её со значением b_{m+1}: если совпадения нет, то при передаче была допущена ошибка, в противном случае считается, что процесс передачи кодового слова β прошёл без ошибки. Ясно, что такое кодирование позволяет обнаружить в процессе передачи только нечётное число ошибок. Пусть надо передать слово a = 110011, (m=6), тогда b = 1100110 (контрольная сумма = 0). Если после передачи вместо кода b будет получено слово 1000110, то будет обнаружено наличие ошибки. Если будет получено, например, слово 0011110, то ошибка обнаружена не будет.

Рассмотрим пример кодирования с исправлением ошибки. Пусть требуется передать сообщение α. Будем кодировать сообщение α по правилу b=K(α) заменяя в α каждый символ его n-кратным повторением. После приёма кода b нужно выделить в полученном слове блоки, содержащие по n символов. Если в каком-либо блоке все символы

---

[1] Для вычисления значения функции k mod n (читается «ка по модулю эн») нужно вычислить остаток от деления натурального числа k на натуральное n: например, 5 mod 3 = 2, 17 mod 4 =1, 3 mod 7 = 3.

одинаковые, то этот блок не содержит ошибок. Если есть различия – блок содержит ошибку, которую можно исправить методом голосования: заменить все символы на тот, который встречается в чаще остальных – он и считается правильным.

Пример 1. Пусть требуется передать слово a = 01011; выберем кратность повторения n = 3; тогда кодовое слово для передачи имеет вид b = 000 111 000 111 111; пусть в результате передачи по каналам связи получено слово c=010011000110101; разобьём его на блоки по три символа: c = 010 011 000 110 101 и исправим ошибки методом голосования: получим d = 000 111 000 111 111 , т.е. получили слово b, от которого можно перейти к требуемому слову a.

Возникает вопрос, при каких условиях некоторый способ кодирования позволяет обнаружить ошибку, и когда может её исправить? Над решением этой проблемы работал Р. Хемминг. Он сформулировал ее решение в 1950 в своей единственной научной статье, посвящённой кодам для коррекции ошибок. Статья содержала конструкцию блочного кода, корректирующего одиночные ошибки, которые возникают при передаче сообщений. Следует отметить, что коды, способные корректировать ошибки при обработке сигналов, были предложены Хэммингом ещё до 1948, когда была опубликована знаменитая статья К. Шеннона «Математическая теория связи».

Р. Хемминг ввёл понятие расстояния между двоичными словами и пришел к выводу, что все зависит от того, насколько разнесены кодовые слова, и сколько ошибок может появиться в передаваемом слове. Нужно согласовывать число возможных ошибок и минимальное расстояние между кодовыми словами.

Определение. Пусть x и y – двоичные слова длины m в алфавите B={0,1}. Введем **расстояние Хемминга** d между x и y следующим

образом: d(x,y) равно числу несовпадений в соответствующих позициях слов x и y.

Пример 2. Пусть x = 110101, y = 011001, z = 000110 → d(x,y) = 3, d(y,z) = 5, d(x,z) = 4.

Введённая функция удовлетворяет всем аксиомам расстояния:

1. $d(x,y) \geq 0$, причём $d(x,y) = 0$ только при $x = y$;
2. $d(x,y) = d(y,x)$;
3. $d(x,z) \leq d(x,y) + d(y,z)$ – неравенство треугольника.

Теорема об обнаруживающем коде.

Пусть при передаче кодовых слов происходит не более k ошибок.

Для того чтобы код являлся обнаруживающим, необходимо и достаточно чтобы наименьшее расстояние между кодовыми словами удовлетворяло неравенству:

$$d(b1, b2) \geq k+1.$$

Теорема об исправляющем коде.

Пусть при передаче кодовых слов происходит не более k ошибок.

Для того чтобы код являлся исправляющим все ошибки, число которых не более k, необходимо и достаточно чтобы наименьшее расстояние между кодовыми словами удовлетворяло неравенству:

$$d(b1, b2) \geq 2k+1.$$

Доказательство.

Достаточность. Пусть для любых двух слов b1 и b2 выполняется неравенство $d(b1, b2) \geq 2k+1$. Пусть при передаче слова b1 было принято слово c1, причём произошло не более k ошибок, т.е. $d(b1, c1) \leq k$. Из неравенства треугольника следует: $d(b1, c1) + d(c1, b2) \geq d(b1, b2) \geq 2k+1$. Отсюда следует, что $d(c1, b2) \geq 2k+1 - d(b1, c1) \geq k + 1$, т.е. расстояние от c1 до любого другого кодового слова b2 больше чем k. Благодаря этому можно однозначно определить ближайшее к слову c1 кодовое слово b1, после чего декодировать его, т.е. исправить ошибку.

Необходимость. Доказательство от противного. Пусть минимальное расстояние между кодовыми словами меньше, чем 2k+1, т.е. найдутся два таких слова b1 и b2, что $d(b1, b2) \le 2k$. Пусть при передаче слова b1 получено слово c1, содержащее ровно k ошибок и расположеное на отрезке между b1 и b2.

Тогда $d(b1, c1) = k$, при этом $d(b2, c1) = d(b1, b2) - d(b1, c1) \le k$. Это означает, что по слову c1 нельзя однозначно восстановить, какое было передано кодовое слово - b1 или b2. Пришли к противоречию с условием, что код является исправляющим при условии, что имеется не более k ошибок при передаче.

Определение. Если для слов длины m в алфавите B={0, 1} используются кодовые слова длины n, то такие коды называются (m, n) – коды.

Для задания всех кодовых слов длины n можно их просто перечислить для каждого исходного передаваемого слова. Таких слов всего 2^m. Однако существует более удобный способ задания кодовых слов – матричный.

При матричном способе задания кодовых слов задаётся порождающая матрица $G=\| g_{ij} \|$, порядка $m \times n$, каждый элемент которой принадлежит алфавиту B= {0, 1}. Кодовые слова $\beta = b_1\ b_2\ b_3 \ldots\ b_m\ b_{m+1}$ определяются для каждого заданного слова $\alpha = a_1\ a_2\ a_3 \ldots\ a_m$ умножением слова α слева, как вектора, на порождающую матрицу G:

$$\beta = \alpha G,$$

причём $b_j = (a_1 g_{1j} + a_2 g_{2j} + \ldots a_m g_{mj}) \bmod 2$.

Предыдущая формула означает, что суммирование $a_1 g_{1j} + a_2 g_{2j} + \ldots a_m g_{mj}$ ведётся по модулю 2, т.е. сначала вычисляется сумма в скобках, а затем вычисляется остаток от деления этой суммы на 2.

Пример 3. Рассмотрим порождающую матрицу

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

С помощью этой матрицы можно задавать (3, 4) - коды. Четвёртый символ кодовых слов будет контрольным: он будет равен 0, если исходное слово содержит чётное число единиц, и будет равен 1 в противном случае. Пусть задано слово α =101; его кодом будет b = αG = b_1 b_2 b_3 b_4 , где

b_1 = (1*1+0*0+1*0) mod 2 = 1;

b_2 = (1*0+0*1+1*0) mod 2 = 0;

b_3 = (1*0+0*0+1*1) mod 2 = 1;

b_4 = (1*1+0*1+1*1) mod 2 = 0;

т.е. кодовое слово b = 1010.

Для слова α =110 кодовым словом является слово b = 1100.

Пример 4. Рассмотрим порождающую матрицу G= (1, 1, 1, 1, 1). Её размерность 1 × 5. Поэтому она задаёт (1, 5) – код. Для слова a1 = 0 получим кодовое слово b1 = 00000; для слова a2 =1 получим b2 = 11111. Здесь минимальное расстояние между b1 и b2 равно 5, т.е. этот код может обнаружить четырёхкратную ошибку (5=k+1 при k=4) и исправить двукратную (5 = 2*k+1 при k=2).

Пример 5. Порождающая матрица

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

задаёт (2, 5) – код: a1=00 → b1 = 00000; a2=01 → b2 = 01011; a3=10 → b3 = 10101; a4=11 → b4 = 11110.

Кодовые слова, полученные методом матричного кодирования, образуют коммутативную группу по сложению. Это означает, что если b1 = a1*G и b2 = a2*G, то b1+ b2 = a1*G +a2*G = (a1+a2)*G.

Заключение.

Информация, информационные системы и информационные технологии играют в современном обществе всё возрастающую роль. Знание теоретических основ информатики, её связей с другими фундаментальными дисциплинами будет способствовать лучшему пониманию информационных процессов, происходящих в окружающем нас мире. Для более полного изучения теоретических основ информатики я отсылаю читателей к списку литературы, в котором указаны книги, раскрывающие фундаментальные основы информатики и ставшие общепризнанной классикой в этой области. Некоторые параграфы данного учебного пособия могут показаться сложными при первом прочтении, например, тема «Условная энтропия», так как при их изложении привлекаются достаточно сложные для студентов первого курса математические понятия. Здесь уместно привести совет одного из преподавателей математики, который мне довелось услышать во время обучения в университете: «Читайте - и понимание придёт».

Возвращаясь к эпиграфу, помещённому во введении, следует отметить, что вопросы практического освоения работы за компьютером и изучение его устройства я считаю не менее важными для студентов, чем знание теоретических основ информатики. Однако, проблемы практической и технической информатики не нашли отражения в данном пособии по причине стремительных изменений в этой области: любое учебное пособие на эту тему рискует потерять актуальность уже в момент своего «выхода в свет». Известное высказывание: «Кто владеет информацией, тот владеет миром» находит всё большее подтверждение в повседневной жизни. Я надеюсь, что данное учебное пособие поможет студентам в изучении теоретических основ информатики и расширит их информационный кругозор.

Список литературы.

1. Арбиб, М. Мозг, машина и математика / М. Арбиб. - М. : Наука, 1968. – 224 с.
2. Винер, Н. Кибернетика, или Управление и связь в животном и машине / Н. Винер. - 2-е изд. - М. : Советское радио, 1968. -328 с.
3. История информатики и философия информационной реальности : учеб. пособие для вузов / под ред. Р. М. Юсупова, В. П. Котенко. – М. : Академический Проект, 2007. - 429 с.
4. Лихтарников, Л. М., Математическая логика / Л. М. Лихтарников, Т. Г. Сукачёва. – СПб. : Лань, 1998. – 228 с.
5. Математическая энциклопедия. В 5 т. Т. 2 / гл. редактор И. М. Виноградов. – М. : «Советская Энциклопедия», 1979. – 1104 стб.
6. Моисеев, Н. Н. Расставание с простотой / Н. Н. Моисеев. - М. : Аграф, 1998. - 480 с.
7. Пенроуз, Р. Новый ум короля / Р. Пенроуз. - М. : УРСС, 2003. - 384 с.
8. Реньи, А. Трилогия о математике / А. Реньи. – М. : Мир, 1980. -376 с.
9. Соболева, Т. С. Дискретная математика / Т. С. Соболева, А. В. Чечкин. – М.: Издательский центр «Академия», 2006. – 256 с.
10. Столл, Р. Множества. Логика. Аксиоматические теории / Р. Столл ; под. ред. Ю. А. Шихановича. – М. : Просвещение, 1968. – 231 с.
11. Урсул, А. Д. Природа информации / А. Д. Урсул. - М. : Политиздат, 1968. - 288 с.
12. Уэбстер, Ф. Теории информационного общества / Ф. Уэбстер. - М. : Аспект Пресс, 2004. – 400 с.
13. Чернавский, Д. С. Синергетика и информация / Д. С. Чернавский. - М. : УРСС, 2004. – 288 с.
14. Эшби, У. Введение в кибернетику : пер. с англ. / У. Эшби; под ред. В. А. Успенского. - Изд. 3-е, стереотип. – М. : КомКнига, 2006. – 432 с.