Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Калининградский государственный технический университет»

О.М. Топоркова

ИНФОРМАТИКА

Допущено (рекомендовано) ученым советом ФГБОУ ВПО «КГТУ»

в качестве учебного пособия для студентов высших учебных заведений,

обучающихся в бакалавриате по направлениям подготовки:

«Информатика и вычислительная техника»;

«Прикладная информатика»

Калининград

Издательство ФГБОУ ВПО «КГТУ»

2014

УДК 004(075)

Рецензенты:

доцент, кандидат технических наук В.В. Капустин

доцент, кандидат педагогических наук Н.Б. Розен

Топоркова О.М.

Информатика: учебное пособие/ О.М. Топоркова. - Калининград: Издательство ФГБОУ ВПО «КГТУ», 2014. – 91 с.

Изложены основные понятия информатики, существенные для ее начального изучения: системы счисления и правила выполнения простейших арифметических операций в них, правила перевода чисел между различными системами счисления, кодирование дискретного сигнала, измерение информации, структурные основные элементы компьютера и их взаимодействие в процессе функционирования.

Учебное пособие предназначено для студентов вузов, обучающихся по направлениямподготовки: Информатика и вычислительная техника; Прикладная информатика.

Ил. 15, табл. 37, список лит. - 6 наименований

Кафедра систем управления и вычислительной техники

Учебное пособие рекомендовано к изданию методической комиссией факультета автоматизации производства и управления 17.02.2014, протокол № 2.

УДК 004(075)

|  |  |
| --- | --- |
|  | © Федеральное государственное бюджетное образовательное учреждение  высшего профессионального образования  «Калининградский государственный технический университет» |

Оглавление

[ВВЕДЕНИЕ 5](#_Toc384451094)

[1. Основные понятия информатики и информации 6](#_Toc384451095)

[1.1. Информатизация общества 6](#_Toc384451096)

[1.2. Понятие информатики 7](#_Toc384451097)

[1.3. Понятие и характерные черты информации 8](#_Toc384451098)

[1.4. Классификация информации 10](#_Toc384451099)

[1.5. Свойства информации 11](#_Toc384451100)

[2. Кодирование информации 13](#_Toc384451101)

[2.1. Виды сигнала как материального носителя информации 13](#_Toc384451102)

[2.2. Преобразования сигнала 14](#_Toc384451103)

[2.3. Системы счисления 15](#_Toc384451104)

[2.4. Правила перевода чисел 17](#_Toc384451105)

[2.4.1. Правила перевода целых чисел 17](#_Toc384451106)

[2.4.2. Правила перевода правильных дробей 19](#_Toc384451107)

[2.4.3. Правило перевода неправильных дробей 22](#_Toc384451108)

[2.5. Правила выполнения простейших арифметических действий 22](#_Toc384451109)

[2.6. Кодирование дискретного сигнала 23](#_Toc384451110)

[2.7. Кодирование по образцу 24](#_Toc384451111)

[2.7.1. Прямые коды 25](#_Toc384451112)

[2.7.2. ASCII-коды 26](#_Toc384451113)

[2.7.3. Коды, учитывающие частоту информационных элементов 28](#_Toc384451114)

[2.7.4. Коды Грея 30](#_Toc384451115)

[2.8. Криптографическое кодирование 31](#_Toc384451116)

[2.8.1. Метод простой подстановки 31](#_Toc384451117)

[2.8.2. Метод Виженера 33](#_Toc384451118)

[2.9. Эффективное кодирование 36](#_Toc384451119)

[2.9.1. Универсальные методы 36](#_Toc384451120)

[2.9.1.1. Метод Шеннона-Фано 36](#_Toc384451121)

[2.9.1.2. Метод Хаффмена 39](#_Toc384451122)

[2.9.1.3. Повышение эффективности кодирования 40](#_Toc384451123)

[2.9.1.4. Декодирование универсальных эффективных кодов 41](#_Toc384451124)

[2.9.2. Специальные методы эффективного кодирования 42](#_Toc384451125)

[2.9.2.1. Методы эффективного кодирования числовых последовательностей 42](#_Toc384451126)

[2.9.2.2. Методы эффективного кодирования словарей 44](#_Toc384451127)

[2.9.2.3. Методы эффективного кодирования естественно-языковых текстов 45](#_Toc384451128)

[2.10. Помехозащитное кодирование 46](#_Toc384451129)

[2.10.1. Искажение кодовых комбинаций 46](#_Toc384451130)

[2.10.2. Кодовое расстояние и корректирующая способность кода 47](#_Toc384451131)

[2.10.3. Коды, исправляющие ошибки 50](#_Toc384451132)

[3. Измерение информации 54](#_Toc384451133)

[3.1. Структурный подход к измерению информации 54](#_Toc384451134)

[3.1.1. Геометрическая мера 54](#_Toc384451135)

[3.1.2. Комбинаторная мера 55](#_Toc384451136)

[3.1.3. Аддитивная мера 56](#_Toc384451137)

[3.2. Статистический подход к измерению информации 57](#_Toc384451138)

[3.3. Семантический подход к измерению информации 59](#_Toc384451139)

[3.3.1. Целесообразность информации 59](#_Toc384451140)

[3.3.2. Полезность информации 59](#_Toc384451141)

[3.3.3. Истинность информации 60](#_Toc384451142)

[3.4. Качество информации 61](#_Toc384451143)

[4. Технические средства информатики 63](#_Toc384451144)

[4.1. Структура компьютера и принципы его функционирования 63](#_Toc384451145)

[4.2. Виды современных компьютеров 65](#_Toc384451146)

[4.3. Структурные элементы компьютера 66](#_Toc384451147)

[4.3.1. Память 68](#_Toc384451148)

[4.3.1.1. Внутренняя память 69](#_Toc384451149)

[4.3.1.2. Внешняя память 71](#_Toc384451150)

[4.3.2. Устройство управления 77](#_Toc384451151)

[4.3.3. Арифметико-логическое устройство 78](#_Toc384451152)

[4.3.3.1. Формы представления целых чисел 79](#_Toc384451153)

[4.3.3.2. Формы представления вещественных чисел 79](#_Toc384451154)

[4.3.3.3. Коды представления числовых данных 81](#_Toc384451155)

[4.3.3.4. Принципы выполнения арифметической операции сложения 82](#_Toc384451156)

[ЛИТЕРАТУРА 89](#_Toc384451157)

[ПРИЛОЖЕНИЕ 1. Положения комбинаторики, используемые в измерении информации 90](#_Toc384451158)

# ВВЕДЕНИЕ

Понятие информатики является относительно новым в лексиконе современного человека. Несмотря на повсеместное употребление, его содержание остается не проясненным до конца в силу своей новизны. Интуитивно ясно, что оно связано с информацией, а также с ее обработкой на компьютерах. Это подтверждается существующей легендой о происхождении данного слова: считается, что оно составлено из двух слов - **ИНФОР**мация и авто**МАТИКА** (как наука, изучающая автоматы, в частности, для преобразования информации).

Вследствие широкого распространения компьютеров и информационного бума, который переживает человечество, с азами информатики должен быть знаком всякий грамотный современный человек; вот почему ее преподавание включено в курс средней школы и продолжается в высшей школе.

Данное учебное пособие раскрывает понятие информатики через основополагающие компоненты - информацию и компьютер - и состоит из четырех частей.

В первой части определяются понятия информатики, информационного общества, информации.

Во второй части вводится понятие сигнала как материального носителя информации, описываются виды и преобразования сигнала. Рассматриваются способы кодирования дискретного сигнала, принятые в информатике для решения различных прикладных задач. Поскольку при кодировании используются различные системы счисления, приводятся те их них, которые применимы в современной информатике, даются правила перевода из одной системы счисления в другую, разбираются правила сложения в двоичной системе счисления.

В третьей части рассматриваются различные подходы к измерению информации и к оценке ее качества.

В четвертой части приводится структура компьютера и принципы функционирования ЭВМ, заложенные в проекте Принстонской машины, рассматриваются элементы структурной схемы компьютера и принципы их функционирования.

# 1. Основные понятия информатики и информации

## 1.1. Информатизация общества

Современное общество характеризуется резким ростом объемов информации, циркулирующей во всех сферах человеческой деятельности. Это привело к информатизации общества.

Под **информатизацией общества** понимают организованный социально-экономический и научно-технический процесс создания оптимальных условий для удовлетворения информационных потребностей и реализации прав физических и юридических лиц на основе формирования и использования **информационных ресурсов** – данных и знаний в различной форме представления.

Целью информатизации является создание **информационного общества**, когда большинство людей занято производством, хранением, переработкой и реализацией информации. Для решения этой задачи возникают новые направления в научной и практической деятельности членов общества. Так возникли информатика и информационные технологии.

Характерные черты информационного общества:

1. решена проблема информационного кризиса, когда устранено противоречие между информационной лавиной и информационным голодом;
2. обеспечен приоритет информации перед другими ресурсами;
3. главной формой развития общества является информационная экономика;
4. в основу общества закладывается автоматизированная генерация, хранение, обработка и использование знаний с помощью новейшей информационной техники и технологии;
5. информационные технологии приобретают глобальный характер, охватывая все сферы социальной деятельности человека;
6. формируется информационное единство всей человеческой цивилизации;
7. с помощью средств информатики реализован свободный доступ каждого человека к информационным ресурсам всей цивилизации;
8. реализованы гуманистические принципы управления обществом и воздействия на окружающую среду.

Возможны и негативные тенденции, сопровождающие процесс информатизации общества:

1. все большее влияние приобретают средства массовой информации;
2. информационные технологии могут разрушить частную жизнь человека;
3. существенное значение приобретает проблема качественного отбора достоверной информации;
4. некоторые люди испытывают сложности адаптации к информационному обществу.

В настоящий момент ближе всех к информационному обществу находятся США, Япония, Англия, страны Западной Европы.

## 1.2. Понятие информатики

Следует отметить, что определений информатики в современной литературе множество. Это происходит оттого, что данная область знаний относительно новая и соответствующий понятийный аппарат не совсем устоялся. Анализ определений позволил выделить их существенную часть и сформулировать то определение, которое приведено ниже.

**Информатик****а** – область человеческой деятельности, связанная с процессами преобразования информации с помощью компьютеров и других средств вычислительной техники. Более конкретно с информатикой связывают одно из следующих понятий или их комбинацию:

1. информатика - это совокупность средств (технических, программных, интеллектуальных) преобразования информации:

* в состав технических (аппаратных) средств(hardware) входят компьютеры и связанные с ними периферийные устройства (мониторы, клавиатуры, принтеры и плоттеры, модемы и т.д.), линии связи, средства оргтехники и т.п., т.е. те материальные ресурсы, которые обеспечивают преобразование информации, причем главенствующую роль в этом списке играет компьютер. По своей специфике компьютер нацелен на решение очень широкого круга задач по преобразованию информации, при этом выбор конкретной задачи при использовании компьютера определяется программным средством, под управлением которого функционирует компьютер;
* к программным средствам(продуктам) (software) относятся операционные системы, интегрированные операционные оболочки, системы программирования и проектирования программных продуктов, различные прикладные пакеты, такие, как текстовые и графические редакторы, бухгалтерские и издательские системы и т.д. Конкретное применение каждого программного продукта специфично и служит для решения определенного круга задач прикладного или системного характера;
* интеллектуальные средства информатики(brainware) соответствуют математическим методам, моделям и типовым алгоритмам, которыеявляются базисом, положенным в основу проектирования и изготовления любого программного или технического средства в силу их исключительной сложности и, как следствие, невозможности умозрительного подхода к созданию.

Перечисленные выше три ресурсных компонента информатики играют разную роль в процессе информатизации общества. Так, совокупность программных и технических средств, имеющихся в том или ином обществе, позволяет сделать его информационным, когда каждый член общества имеет возможность получить практически любую (исключая, естественно, секретную) интересующую его информацию (такие потребители информации называются *конечными пользователями*). В то же время, сложность технических и программных систем заставляет использовать имеющиеся технические и программные продукты, а также нужные методы, модели и алгоритмы для проектирования и производства новых и совершенствования старых технических и программных систем. В этом случае можно сказать, что средства преобразования информации используются для производства себе подобных. Тогда их пользователем является специалист в области информатики и информационных технологий, а не конечный пользователь;

1. информатика - это отрасли производства для создания технических и программных средств. Наиболее известными фирмами для изготовления технических средств информатики являются IBM, Apple, Intel, Hewlett Packard, для производства программных продуктов - Microsoft, Lotus, Borland;
2. информатика - это фундаментальная наука, которая занимается разработкой методов, моделей и алгоритмов преобразования информации, т.е. созданием brainware. Ее прерогативой является исследование процессов преобразования информации и на основе этих исследований разработка соответствующих теорий, моделей, методов и алгоритмов, которые затем применяются на практике;
3. информатика - это прикладная наука, позволяющая путем объединения технических, программных и интеллектуальных средств автоматизировать конкретные информационные процессы в той или иной предметной области, т.е. создавать информационные технологии.

Таким образом, главная функция информатики состоит в разработке методов и средств преобразования информации с использованием компьютера и в применении их при организации технологического процесса преобразования информации.

Выполняя свою функцию, информатика решает следующие задачи:

* исследует информационные процессы в различных системах;
* разрабатывает информационную технику и создает новейшие технологии преобразования информации на основе результатов, полученных в ходе исследования информационных процессов;
* решает научные и инженерные проблемы создания, внедрения и обеспечения эффективного использования компьютерной техники и технологии во всех сферах человеческой деятельности.

## 1.3. Понятие и характерные черты информации

Как видно из определения информатики, ее функций и задач, приведенных выше, одним из ключевых понятий информатики является информация.

**Информа****ция** – это сведения об окружающем мире (объекте, процессе, явлении, событии и т.д.), которые являются объектом преобразования (включая хранение, передачу и т.д.) и используются для принятия решения.

Мы постоянно пользуемся информацией. Например, выходя утром из дома, мы слушаем по радио прогноз погоды на день и принимаем решение о том, понадобится ли нам зонт. В этом случае информация, содержащаяся в прогнозе, влияет на наше решение, которое рождается в недрах собственного сознания, например, в форме фразы: «возьму-ка я зонтик, поскольку давление понижается». Так же и преподаватель использует ответ студента на экзамене в качестве информации, которая позволяет принять решение об оценке.

Характерные чертыинформации:

1. Это наиболее важный ресурс современного производства: он снижает потребность в земле, труде, капитале, уменьшает расход сырья и энергии. Примером является применяемая в сельском хозяйстве агротехника выращивания овощных культур без привлечения земельных ресурсов - гидропоника, которая использует вместо земли растворы, насыщенные полезными для растений веществами. Здесь потребность в земле снижается до нуля. Таким образом, владея подобной технологией, т.е. обладая соответствующей информацией, можно выращивать овощные культуры.
2. Информация вызывает к жизни новые производства. Так, изобретенный в первой половине 20-го века лазерный луч (в 1964 году ученые Ч. Таунс, Н.Г. Басов и А.М. Прохоров получили Нобелевскую премию по физике за это изобретение) послужил началу развития лазерной связи.
3. Информация является товаром, который остается у продавца после продажи, что делает информацию очень выгодным товаром, поскольку одну и ту же информацию можно многократно продавать.
4. Информация придает дополнительную ценность другим ресурсам. Действительно, работник с высшим образованием ценится больше, чем со средним. Автомобиль последней модели, в котором воплощены научно-технические разработки ученых и инженеров, т.е. новая информация, более дорогой, чем его предшественники.
5. Информация может накапливаться. Речь идет не о простом количественном пополнении объемов информации, а о формировании новых знаний, что позволяет развивать общество, поскольку владея определенными знаниями, человек может генерировать новую информацию. Именно таким образом учеными совершаются открытия в области науки, инженерами создаются новые технические изобретения. Чем больше знает человек и чем качественнее эти знания, тем большее количество новой информации он может сформировать.

Как следует из определения, с информацией связаны три понятия (их взаимосвязь показана на рис. 1.1):

* **источник информации** – элемент окружающего мира, сведения о котором являются объектом преобразования;
* **потребитель информации** – элемент окружающего мира, который использует информацию для принятия решения;
* **сиг****нал** – материальный носитель, который фиксирует информацию для переноса ее от источника к потребителю.

2.1.bmp

Рис. 1.1. Схема взаимосвязи основных понятий информации

Так, источником информации, которую в данный момент получает читатель настоящего учебного материала, является информатика как сфера человеческой деятельности; потребителем – сам читатель, а сигналом – текст на электронном (или бумажном) носителе. Будучи прочитанной и запомненной студентом, данная информация приобретет еще один носитель – биологический, когда она «записывается» в память обучаемого. Очевидно, что источник и потребитель в этом случае не меняются.

Уже отмечалось, что информационный ресурс существует в виде данных и знаний: **данные** – это полученные путем наблюдения и зафиксированные факты, дискретно, т.е. через отдельные свойства, описывающие источник информации (говорят – предметную область (ПО)); **знания** – это закономерности источника информации (понятия, сведения, принципы, связи, законы), приобретенные в результате практической деятельности; они позволяют специалистам ставить и решать задачи в этой предметной области.

Примером *данных* могут служить сведения о результатах сдачи сессии студентами и о последующих решениях деканата - это некоторый набор фактов. Например, студент Х имеет двойку по дисциплине Y; студента Х отчисляют из вуза.

Если на основании множества фактов делается обобщение следующего вида: каждый студент, не сдавший экзамен, т.е. имеющий двойку, отчисляется из вуза, то это уже выявленная закономерность, т.е. *знание*.

## 1.4. Классификация информации

**Классификацию** информации выполняют по нескольким основаниям (эта классификация не является строгой и может меняться):

1. По времени возникновения:

а) априорная – известна потребителю заранее, до получения сигнала;

б) апостериорная – становится известной потребителю после получения сигнала.

Получаемая сейчас читателем информация является априорной, т.е. уже известной, если он освоил азы информатики в школе, и апостериорной, т.е. новой, в противном случае.

1. По стабильности:

а) переменная – отражает фактические характеристики источника информации. Может меняться;

б) постоянная – неизменная и многократно используемая в течение длительного периода времени. Строго говоря, и эта информация может меняться, но с гораздо меньшей частотой, которой можно пренебречь.

В настоящем учебном пособии используются оба вида информации. Упомянутые выше фирмы-производители технических и программных средств относятся к первому виду. В самом деле, может быть, что к моменту прочтения данного текста эти фирмы перестанут существовать на рынке производителей. В то же время весь понятийный аппарат, излагаемый по тексту, относится к постоянной информации и является тем базисом, который позволяет информатикам говорить на одном профессиональном языке.

1. По способу использования:

а) открытая – ее использование ничем не ограничено;

б) закрытая – ее использование возможно с согласия определенных физических или юридических лиц;

в) коммерческая – является объектом купли-продажи.

Отметим в качестве комментария, что излагаемая в учебном пособии информация не является ни закрытой, ни коммерческой – она находится в открытом доступе для всех желающих.

## 

## 1.5. Свойства информации

Будучи объектом преобразования и использования, информация, зафиксированная в сигнале, характеризуется следующими **свойствами**:

* **синта****ксис** – свойство, определяющее способ представления информации на носителе (в сигнале). Данная информация представлена на носителе с помощью определенного шрифта. Здесь же можно рассматривать такие параметры представления информации, как стиль и цвет шрифта, его размеры, формат бумаги и ее качество и т.д. Выделение нужных параметров как синтаксических свойств, очевидно, определяется предполагаемым способом преобразования. Например, для плохо видящего человека существенным является размер и цвет шрифта;
* **сема****нтика** – свойство, определяющее смысл информации как соответствие сигнала реальному миру. Семантика сигнала «информатика» заключается в данном ранее определении. Семантика может рассматриваться как некоторое соглашение, известное потребителю информации, о том, что означает каждый сигнал (так называемое правило интерпретации). Например, именно семантику сигналов изучает начинающий автомобилист, штудирующий правила дорожного движения, познавая дорожные знаки (в этом случае сигналами выступают сами знаки). Семантику слов (сигналов) познаёт обучаемый какому-либо иностранному языку. Можно сказать, что смысл настоящего обучения информатике заключается в изучении семантики различных сигналов – суть ключевых понятий этой дисциплины;
* **прагм****атика** – свойство, определяющее влияние информации на поведение потребителя. Прагматика информации, получаемой читателем настоящего учебного пособия, заключается, по меньшей мере, в успешной сдаче экзамена по информатике. Хочется верить, что этим прагматика данного труда не ограничится, и он послужит для дальнейшего обучения и профессиональной деятельности читателя.

Проецируя приведенные свойства информации на элементы схемы рис. 1.1, можно схему видоизменить следующим образом (рис. 1.2):

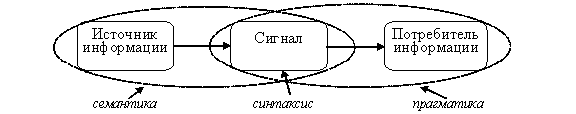


Рис. 1.2. Соотношение свойств информации

Синтаксис определяет свойства сигнала как такового, семантика выражает связь сигнала с источником информации, а прагматика представляет влияние сигнала на потребителя информации.

Следует отметить, что различные по синтаксису сигналы могут иметь одинаковую семантику. Например, сигналы «ЭВМ» и «компьютер» означают электронное устройство для преобразования информации. В этом случае обычно говорят о *синонимии* сигналов.

С другой стороны, один сигнал (т.е. информация с одним синтаксическим свойством) может иметь разную прагматику для потребителей. Дорожный знак, известный под названием «кирпич» и имеющий вполне определенную семантику («въезд запрещен»), означает для автомобилиста запрет на въезд, а на пешехода никак не влияет.

В то же время, сигнал «ключ» может иметь разную семантику: скрипичный ключ, родниковый ключ, ключ для открытия замка или гаечный ключ (в этом случае говорят об *омонимии* сигнала).

# 2. Кодирование информации

## 2.1. Виды сигнала как материального носителя информации

Сигнал, который фиксирует информацию, может быть дискретным и непрерывным (аналоговым).

**Дискретный с****игнал** слагается из счетного множества (т.е. такого множества, элементы которого можно пересчитать) элементов (говорят – информационных элементов). Например, дискретным является сигнал «кирпич» (см. выше). Он состоит из следующих двух элементов (это синтаксическая характеристика данного сигнала): красного круга и белого прямоугольника внутри.

Именно в виде дискретного сигнала представлена та информация, которую сейчас осваивает читатель. Можно выделить следующие ее элементы: разделы (например, «Информатизация общества»), подразделы (например, «Виды сигнала»), абзацы, предложения, отдельные фразы, слова и отдельные знаки (буквы, цифры, знаки препинания и т.д.).

Последний пример показывает, что в зависимости от прагматики информации можно выделять разные информационные элементы. В самом деле, для лица, изучающего информатику по данному конспекту, важны более крупные информационные элементы, такие как разделы, подразделы, отдельные абзацы. Они позволяют ему легче ориентироваться в структуре материала, лучше его усваивать и готовиться к экзамену. Для того, кто готовил данный материал, помимо указанных информационных элементов, важны также и более мелкие, например, отдельные предложения, с помощью которых излагается та или иная мысль и которые реализуют тот или иной способ доступности материала. Для издателя важны еще более мелкие детали, например, такие характеристики отдельных символов, как их размер, стиль и т.д., поскольку существуют определенные правила подготовки печатного текста.

Набор самых «мелких» элементов дискретного сигнала называется алфавитом, а сам дискретный сигнал называют также **соо****бщением**.

Если в качестве дискретного сигнала рассматривать некоторый текст, то его алфавитом можно считать буквы, знаки препинания и т.д. Если дискретный сигнал – это некоторый набор целых десятичных чисел, то алфавит – это десятичные цифры.

**Непреры****вный сигнал** отражается некоторой физической величиной, изменяющейся в заданном интервале времени. Примерами такого сигнала в окружающем мире могут служить звуки, световые волны.

В виде непрерывного сигнала представлена настоящая информация для тех студентов–потребителей, которые посещают лекции по информатике и через звуковые волны, переносящие голос лектора и носящие непрерывный (между паузами) характер, воспринимают материал.

Современные технические средства информатики связаны, в основном, с преобразованиями дискретного сигнала. По этой причине возникает задача прямого и обратного преобразования аналогового сигнала в дискретный.

## 2.2. Преобразования сигнала

Для преобразования аналогового сигнала в дискретный используется процедура, которая называется **кван****тованием**. Она включает два последовательных этапа: квантование по времени и квантование по уровню (дискретизацию).

**Квантование по времени** – замена непрерывной (по времени и по уровню) функции x(t) (рис. 2.1а) некоторым множеством непрерывных (по уровню) функций x(ti) (на рис. 2.1б i={1,2,3,4}).

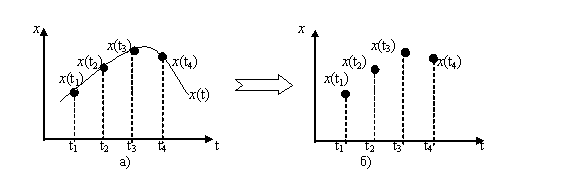


Рис.2.1. Иллюстрация к квантованию по времени:

а) аналоговый сигнал *x*(t) до квантования;

б) дискретный (по времени) сигнал *x*(t) – результат квантования.

Таким образом, на рис. 2.1 сигнал x(t) преобразован в множество сигналов {x(t1), x(t2), x(t3), x(t4)}: x(t)→{x(t1), x(t2), x(t3), x(t4)}.

Очевидно, квантование по времени связано с потерей информации. В самом деле, дискретный по времени сигнал на рис. 2.1б не показывает, как ведет себя исходный сигнал в моменты времени, например, между t3 и t4. Иначе говоря, данный шаг квантования связан с некоторой погрешностью ε, которая зависит от шага квантования Δt=ti–ti-1: при малых значениях шага Δt число точек замера высоко, и теряется мало информации; очевидно, картина обратная при больших шагах Δt.

Квантование по времени различается по регулярности отсчетов:

* равномерное, когда Δt постоянно;
* неравномерное, когда Δt переменно, причем этот вид, в свою очередь, делится на подвиды:
* адаптивное, когда Δt меняется автоматически в зависимости от текущего изменения сигнала. Это позволяет увеличивать шаг Δt, когда изменения сигнала x(t) незначительны, и уменьшать – в противном случае;
* программируемое, когда Δt изменяется оператором или в соответствии с заранее выставленными условиями, например, в фиксированные моменты времени.

**Квантование по уровню** - преобразование непрерывных (по уровню) сигналов x(ti) в дискретные (рис. 2.2). Шаг квантования Δx определяется по формуле: Δx=xj–xj-1. Можно сказать, что квантование по уровню – это измерение сигнала. Для этого введем на оси ординат мерную шкалу и спроецируем на нее сигналы x(ti) (рис. 2.2б).

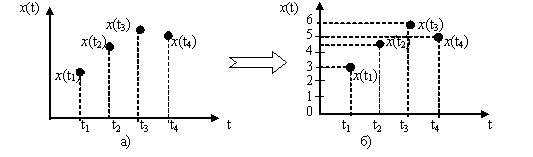


Рис. 2.2. Иллюстрация к квантованию по уровню:

а) аналоговые по уровню (но дискретные по времени) сигналы x(ti) до квантования;

б) квантованные по уровню (измеренные) сигналы x(ti).

Видно, что сигнал x(t1) составляет три уровня, сигнал x(t3) – 6, а x(t4) – пять уровней квантования. В то же время сигнал x(t2) попадает в промежуток между четырьмя и пятью и не может быть однозначно измерен. В таком случае поступают одним из следующих способов:

1. x(ti) отождествляют с ближайшим значением (в нашем примере – с 4). Тогда в результате процедуры квантования по уровню множество сигналов {x(t1),x(t2),x(t3),x(t4)}, полученное в первом шаге квантования, преобразуется в множество целых чисел {3,4,6,5}: {x(t1),x(t2),x(t3), x(t4)}→{3,4,6,5};
2. x(ti) отождествляют с ближайшим большим значением. Тогда сигнал x(t2) отождествится с 5 независимо от того, насколько близко он к этому уровню квантования находится. Сформируется множество {3,5,6,5}: {x(t1),x(t2),x(t3), x(t4)}→{3,5,6,5};
3. x(ti) отождествляют с ближайшим меньшим значением. Тогда сигнал x(t2) отождествится с 4. Сформируется множество {3,4,6,5}: {x(t1),x(t2),x(t3),x(t4)}→{3,4,6,5}.

Таким образом, в результате проведения двух последовательных шагов квантования непрерывный сигнал x(t) преобразуется в дискретный сигнал {3,4,6,5} (для первого способа приближения при измерении), который является одномерным массивом для любого языка программирования и с легкостью может быть обработан компьютером.

Очевидно, и при квантовании по уровню возникает погрешность квантования. Она тем меньше, чем меньше шаг квантования.

Виды квантования по уровню:

1. равномерное, когда диапазон изменения сигнала разбивается на m одинаковых частей,
2. неравномерное, когда диапазон изменения сигнала разбивается на m различных частей.

## 2.3. Системы счисления

Для удобства последующего преобразования дискретный сигнал подвергается **кодированию**. Большинство кодов основано на системах счисления, причем использующих позиционный принцип образования числа, при котором значение каждой цифры зависит от ее положения в числе.

Примером позиционной формы записи чисел является та, которой мы пользуемся (так называемая арабская форма чисел). В числах 123 и 321 значения цифры 3, например, определяются ее положением в числе: в первом случае она обозначает три единицы (т.е. просто три), а во втором – три сотни (т.е. триста).

Тогда полное значение числа получается по формуле (2.1):

(2.1)



где *l –* количество разрядов числа,

*i* – порядковый номер разряда,

*m* – основание системы счисления,

*ai*– множитель, принимающий любые целочисленные значения от 0 до *m*-1 и соответствующий цифре в *i*-й позиции числа.

Например, для десятичного (*m*=10) числа 345 его полное значение рассчитывается следующим образом: 3\*102+4\*101+5\*100=345.

Римские числа являются примером полупозиционной системы образования числа: в числах IX и XI знак I обозначает в обоих случаях единицу (признак непозиционной системы), но будучи расположенным слева от знака X (обозначающего десять), вычитается из десяти, а при расположении справа – прибавляется к десяти. В первом случае полное значение числа равно 9, во втором – 11.

В современной информатике используются в основном три системы счисления (все – позиционные): двоичная, шестнадцатеричная и десятичная.

**Двоичная система счисления** используется для кодирования дискретного сигнала, потребителем которого является вычислительная техника. Такое положение дел сложилось исторически, поскольку двоичный сигнал проще представлять на аппаратном уровне. В этой системе счисления для представления числа применяются два знака – 0 и 1.

**Шестнадцатеричная система счисления** используется для кодирования дискретного сигнала, потребителем которого является хорошо подготовленный пользователь – специалист в области информатики. В такой форме представляется содержимое любого файла, затребованное через интегрированные оболочки, например, средствами Far. Используемые знаки для представления числа – десятичные цифры от 0 до 9 и буквы латинского алфавита – A, B, C, D, E, F.

**Десятичная система счисления** используется для кодирования дискретного сигнала, потребителем которого является так называемый конечный пользователь – неспециалист в области информатики (очевидно, что и любой человек может выступать в роли такого потребителя). Используемые знаки для представления числа – цифры от 0 до 9.

Соответствие между первыми несколькими натуральными числами всех трех систем счисления представлено в табл. 2.1.

Таблица 2.1.

Соответствие некоторых целых чисел

|  |  |  |
| --- | --- | --- |
| Десятичная  система | Двоичная  система | Шестнадцатеричная  система |
| 0 | 0 | 0 |
| 1 | 1 | 1 |
| 2 | 10 | 2 |
| 3 | 11 | 3 |
| 4 | 100 | 4 |
| 5 | 101 | 5 |
| 6 | 110 | 6 |
| 7 | 111 | 7 |
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |
| 16 | 10000 | 10 |

Для различения систем счисления, в которых представлены числа, в обозначение двоичных и шестнадцатеричных чисел вводят дополнительные реквизиты:

* для двоичных чисел – нижний индекс справа от числа в виде цифры 2 или букв В или b (binary – двоичный), либо знак B или b справа от числа. Например, 1010002 = 101000b = 101000B = 101000B = 101000b;
* для шестнадцатеричных чисел - нижний индекс справа от числа в виде числа 16 или букв H или h (hexadecimal – шестнадцатеричный), либо знак H или h справа от числа. Например, 3AB16 = 3ABH = 3ABh = 3ABH = 3ABh.

Для перевода чисел из одной системы счисления в другую существуют определенные правила.

## 2.4. Правила перевода чисел

Правила перевода различаются в зависимости от формата числа – целое или правильная дробь. Для вещественных чисел используется комбинация правил перевода для целого числа и правильной дроби.

### 2.4.1. Правила перевода целых чисел

Результатом перевода целых чисел **всегда** является целое число.

Разобьем эти правила на три группы:

1. **Из десятичной системы счисления** – в двоичную или шестнадцатеричную:

а) исходное целое число делится на основание системы счисления, в которую переводится (2 или 16); получается частное и остаток;

б) если полученное частное меньше основания системы счисления, в которую переводится число, процесс умножения прекращается, переходят к шагу в). Иначе над частным выполняют действия, начиная с шага а);

в) все полученные остатки и последнее частное преобразуются в соответствии с табл. 2.1 в цифры той системы счисления, в которую выполняется перевод;

г) формируется результирующее число: его старший разряд – полученное последнее частное, каждый последующий младший разряд образуется из полученных остатков от деления, начиная с последнего и кончая первым. Таким образом, младший разряд полученного числа – первый остаток от деления, а старший – последнее частное.

**Пример 2.1**. Выполнить перевод числа 19 в двоичную систему счисления:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| -19 | 2 |  |  |  |
| 18 | -9 | 2 |  |  |
| **1** | 8 | -4 | 2 | последнее частное от деления, поскольку 1<2. Это старший разряд результирующего двоичного числа. |
|  | **1** | 4 | -2 | 2 |
|  |  | **0** | 2 | **1** |
|  |  |  | **0** |  |
|  |  |  |  |  |
|  |  |  |  |  |

1 0 0 1 1 – результат.

Таким образом, 19=100112.

**Пример 2.2**. Выполнить перевод числа 19 в шестнадцатеричную систему счисления:

|  |  |
| --- | --- |
| -19 | 16 |
| 16 | **1** |
| **3** |  |

1 3 – результирующее число.

Таким образом, 19=1316.

**Пример 2.3.** Выполнить перевод числа 123 в шестнадцатеричную систему счисления:

|  |  |
| --- | --- |
| -123 | 16 |
| 112 | **7** |
| **11** |  |

7 В – результирующее число.

Таким образом, 123=7В16.

2. **Из двоичной или шестнадцатеричной систем счисления** – в десятичную. В этом случае рассчитывается полное значение числа по формуле (2.1).

**Пример 2.4.** Выполнить перевод числа 1316 в десятичную систему счисления. Имеем:

1316 = 1\*161 + 3\*160 = 16 + 3 = 19.

Таким образом, 1316=19.

**Пример 2.5.** Выполнить перевод числа 100112 в десятичную систему счисления. Имеем:

100112 = 1\*24 + 0\*23 + 0\*22 + 1\*21 + 1\*20 = 16+0+0+2+1 = 19.

Таким образом, 100112=19.

3. **Из двоичной системы счисления** в шестнадцатеричную:

а) исходное число разбивается на тетрады (наборы из четырёх цифр), начиная с младших разрядов. Если количество цифр исходного двоичного числа не кратно 4, оно дополняется *слева* незначащими нулями до достижения кратности 4;

б) каждая тетрада заменяется соответствующей шестнадцатеричной цифрой в соответствии с табл. 2.1.

**Пример 2.6.** Выполнить перевод числа 100112 в шестнадцатеричную систему счисления.

Поскольку в исходном двоичном числе количество цифр не кратно 4, дополняем его слева незначащими нулями до достижения кратности 4 числа цифр. Имеем:

100112 = 000100112

первая тетрада – младшая цифра числа

вторая тетрада – старшая цифра числа

В соответствии с табл. 2.1 00112=112=316 и 00012=12=116. Тогда 100112= 1316.

4. **Из шестнадцатеричной системы счисления** в двоичную:

а) каждая цифра исходного числа заменяется тетрадой двоичных цифр в соответствии с табл. 2.1. Если в таблице двоичное число имеет менее 4 цифр, оно дополняется слева незначащими нулями до тетрады;

б) незначащие нули в результирующем числе отбрасываются.

**Пример 2.7.**  Выполнить перевод числа 1316 в двоичную систему счисления.

По табл. 2.1 имеем: 116=12 и после дополнения незначащими нулями 12=00012; 316=112 и после дополнения незначащими нулями 112=00112. Тогда 1316=000100112. После удаления незначащих нулей имеем 1316=100112.

Таким образом, 1316=100112.

### 2.4.2. Правила перевода правильных дробей

Результатом перевода правильной дроби всегда является правильная дробь. Выделим также три группы правил:

1. **Из десятичной системы счисления** – в двоичную или шестнадцатеричную:

а) исходная дробь умножается на основание системы счисления, в которую переводится (2 или 16);

б) в полученном произведении целая часть преобразуется в соответствии с табл. 2.1 в цифру нужной системы счисления и отбрасывается – она является старшей цифрой получаемой дроби;

в) оставшаяся дробная часть вновь умножается на нужное основание системы счисления с последующей обработкой полученного произведения в соответствии с шагом б);

г) процедура умножения продолжается до тех пор, пока не будет получен нулевой результат в дробной части произведения или не будет достигнуто требуемое количество цифр в результате;

д) формируется результат: последовательно отброшенные в шаге б) цифры составляют дробную часть результата, причем в порядке уменьшения старшинства.

**Пример 2.8**. Выполнить перевод числа 0,847 в двоичную систему счисления. Перевод выполнить до четырех значащих цифр после запятой.

Имеем:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| \*0,847 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| **1**,694 |  | \*0,694 |  |  |  |  |
|  |  | 2 |  |  |  |  |
|  |  | **1**,388 |  | \*0,388 |  |  |
|  |  |  |  | 2 |  |  |
|  |  |  |  | **0**,776 |  | \*0,776 |
|  |  |  |  |  |  | 2 |
|  |  |  |  |  |  | **1**,552 и т.д. |

0,1101 – результирующее число.

В данном примере процедура перевода прервана на четвертом шаге, поскольку получено требуемое число разрядов результата. Очевидно, это привело к потере ряда цифр.

Таким образом, 0,847=0,11012.

**Пример 2.9**. Выполнить перевод числа 0,847 в шестнадцатеричную систему счисления. Перевод выполнить до трех значащих цифр.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| \*0,847 |  |  |  |  |  |
| 16 |  |  |  |  |  |
| **13**,552 |  | \*0,552 |  |  |  |
|  |  | 16 |  |  |  |
|  |  | **8**,832 |  | \*0,832 |  |
|  |  |  |  | 16 |  |
|  |  |  |  | **13**,312 и т.д. |  |
|  |  |  |  |  |  |

0,D8D – результирующее число.

В данном примере также процедура перевода прервана.

Таким образом, 0,847=0,D8D16.

2. **Из двоичной или шестнадцатеричной систем счисления** – в десятичную. В этом случае рассчитывается полное значение числа по формуле (2.1), причем коэффициенты *ai* принимают десятичное значение в соответствии с табл. 2.1.

**Пример 2.10**. Выполнить перевод из двоичной системы счисления в десятичную числа 0,11012. Имеем:

0,11012 = 1\*2-1 + 1\*2-2 + 0\*2-3 +1\*2-4 = 0,5 + 0,25 + 0 + 0,0625 = 0,8125.

Расхождение полученного результата с исходным для получения двоичной дробичислом (см. пример 2.8) вызвано тем, что процедура перевода в двоичную дробь была прервана.

Таким образом, 0,11012=0,8125.

**Пример 2.11**. Выполнить перевод из шестнадцатеричной системы счисления в десятичную числа 0,D8D16. Имеем:

0,D8D16 = 13\*16-1 + 8\*16-2 + 13\*16-3 = 13\*0,0625 + 8\*0,003906 + 13\* 0,000244 = 0,84692.

Расхождение полученного результата с исходным для получения двоичной дроби числом (см. пример 2.9) вызвано тем, что процедура перевода в шестнадцатеричную дробь была прервана.

Таким образом, 0,D8D16=0,84692.

3. **Из двоичной системы счисления** в шестнадцатеричную:

а) исходная дробь делится на тетрады, начиная с позиции десятичной точки вправо. Если количество цифр дробной части исходного двоичного числа не кратно 4, оно дополняется справа незначащими нулями до достижения кратности 4;

б) каждая тетрада заменяется шестнадцатеричной цифрой в соответствии с табл. 2.1.

**Пример 2.12**. Выполнить перевод из двоичной системы счисления в шестнадцатеричную числа 0,11012. Имеем:

0,11012 = 0,11012

первая (и единственная) тетрада

В соответствии с табл. 2.1 11012=D16.

Таким образом, 0,11012=0,D16.

**Пример 2.13**. Выполнить перевод из двоичной системы счисления в шестнадцатеричную числа 0,00101012.

Поскольку количество цифр дробной части не кратно 4, добавим справа незначащий ноль:

0,00101012 = 0,001010102.

вторая тетрада

первая тетрада

В соответствии с табл. 2.1 00102=102=216 и 10102=A16. Тогда имеем 0,00101012=0,2A16.

4. **Из шестнадцатеричной системы счисления** в двоичную:

а) каждая цифра исходной дроби заменяется тетрадой двоичных цифр в соответствии с табл. 2.1;

б) незначащие нули отбрасываются.

**Пример 2.14**. Выполнить перевод из шестнадцатеричной системы счисления в двоичную числа 0,2А16.

По табл. 2.1 имеем 216=00102 и А16=10102.

Тогда 0,2А16=0,001010102.

Отбросим в результате незначащий ноль и получим окончательный результат.

Таким образом, 0,2А16=0,00101012.

### 2.4.3. Правило перевода неправильных дробей

В результате перевода неправильной дроби всегда получается неправильная дробь.

Отдельно переводится целая часть числа, отдельно – дробная. Результаты складываются.

**Пример 2.15**. Выполнить перевод из десятичной системы счисления в шестнадцатеричную числа 19,847. Перевод выполнять до трех значащих цифр после запятой.

Представим исходное число как сумму целого числа и правильной дроби: 19,847 = 19 + 0,847.

Как следует из примера 2.2, 19 = 1316; а в соответствии с примером 2.9 0,847=0,D8D16.

Тогда имеем: 19 + 0,847 = 1316 + 0,D8D16 = 13,D8D16.

Таким образом, 19,847=13,D8D16.

## 2.5. Правила выполнения простейших арифметических действий

Арифметические операции для двоичных и шестнадцатеричных чисел выполняются по тем же правилам, что и для десятичных чисел, которые хорошо знакомы читателю. Рассмотрим выполнение важнейших для информатики операций арифметического сложения и сложения по модулю 2 для двоичных чисел.

Правила арифметического сложения двоичных чисел представлены ниже:

02 + 02 = 02; 02 + 12 = 12; 12 + 02 = 12; 12 + 12 = 102.

**Пример 2.16**. Сложить числа 11012 и 110112.

Запишем слагаемые в столбик и пронумеруем разряды, присвоив младшему разряду номер 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| номера разрядов | **5** | **4** | **3** | **2** | **1** |
|  | + | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 | 1 |

Процесс образования результата по разрядам описан ниже:

а) разряд 1 формируется следующим образом: 12+12=102; 0 остается в разряде 1, 1 переносится во второй разряд;

б) разряд 2 формируется следующим образом: 02+12+12=102, где вторая 12 – единица переноса; 0 остается в разряде 2, 1 переносится в третий разряд;

в) третий разряд формируется следующим образом: 12+02+12=102, где вторая 12 – единица переноса; 0 остается в разряде 3, 1 переносится в разряд 4;

г) четвертый разряд формируется следующим образом: 12+12+12=112, где третья 12 – единица переноса; 1остается в разряде 4, 1 переносится в пятый разряд;

д) пятый разряд формируется следующим образом: 12+12=102; где вторая 12 – единица переноса; 0 остается в разряде 5, 1 переносится в шестой разряд.

Таким образом:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| номера разрядов | **6** | **5** | **4** | **3** | **2** | **1** |
|  |  | + | 1 | 1 | 0 | 1 |
|  |  | 1 | 1 | 0 | 1 | 1 |
|  | 1 | 0 | 1 | 0 | 0 | 0 |

Проверим результат. Для этого определим полные значения слагаемых и результата:

11012 = 1\*23 +1\*22 + 0\*21 + 1\*20 = 8 + 4 + 1 = 13;

110112 = 1\*24 + 1\*23 + 0\*22 + 1\*21 + 1\*20 = 16 + 8 + 2 + 1 = 27;

1010002 = 1\*25 + 0\*24 + 1\*23 + 0\*22 + 0\*21 + 0\*20 = 32 + 8 = 40.

Поскольку 13+27=40, двоичное сложение выполнено верно.

Сложение по модулю – это арифметическое сложение, при котором оговаривается, превышение какого числа (модуля) должно отражаться при сложении, и, кроме того, при таком сложении отсутствует единица переноса в старший разряд. Обозначается операция знаком . Как правило, такому сложению подвергаются числа с одинаковым количеством разрядов.

Правила сложения по модулю 2 приведены ниже:

02⊕02=02; 02⊕12=12; 12⊕02=12; 12⊕12=02.

Анализ результатов показывает, что эту операцию можно рассматривать как операцию сравнения двух двоичных цифр: если они равны – результат нулевой, если различаются, результат равен 1.

**Пример 2.17**. Сложить по модулю 2 числа 1012 и 0112.

Запишем алгебраические слагаемые в столбик:

|  |  |  |  |
| --- | --- | --- | --- |
| ⊕ | 1 | 0 | 1 |
|  | 0 | 1 | 1 |
|  | 1 | 1 | 0 |

## 2.6. Кодирование дискретного сигнала

**Кодирование** – это представление сигнала в определенной форме, удобной или пригодной для последующего использования сигнала. Говоря строже, это правило, описывающее отображение одного набора знаков в другой набор знаков. Тогда отображаемый набор знаков называется **исходным алфавитом**, а набор знаков, который используется для отображения, - **кодовым алфавитом,** или алфавитом для кодирования. При этом кодированию подлежат как отдельные символы исходного алфавита, так и их комбинации. Аналогично для построения кода используются как отдельные символы кодового алфавита, так и их комбинации.

Совокупность символов кодового алфавита, применяемых для кодирования одного символа (или одной комбинации символов) исходного алфавита, называется **кодовой комбинацией**, или, короче, **кодом символа**. При этом кодовая комбинация может содержать один символ кодового алфавита.

Символ (или комбинация символов) исходного алфавита, которому соответствует кодовая комбинация, называется **исходным символом**.

Совокупность кодовых комбинаций называется **кодом**.

Взаимосвязь символов (или комбинаций символов, если кодируются не отдельные символы исходного алфавита) исходного алфавита с их кодовыми комбинациями составляет **таблицу соответствия** (или таблицу кодов).

Следует отметить, что понятие «код» омонимично: оно может употребляться и в смысле кодовой комбинации, и в приведенном выше смысле. Аналогично понятие «кодовая комбинация» синонимично понятию «код».

Кодирование дискретного сигнала выполняется в два этапа:

1. вначале разрабатывается код для заданного исходного алфавита. Этот этап выполняется единожды, и разработанный код применяется многократно для второго этапа,
2. произвольный, предъявленный к кодированию дискретный сигнал собственно кодируется с помощью кодовой таблицы, созданной на этапе 1.

Обратная процедура получения исходных символов по кодам символов называется **декодированием**. Очевидно, для выполнения правильного декодирования код должен быть однозначным, т.е. одному исходному знаку должна соответствовать точно одна кодовая комбинация и наоборот.

В зависимости от целей кодирования, различают следующие его виды:

1. **кодирование по образцу** - используется, в частности, всякий раз для ввода информации в компьютер для ее внутреннего представления, а потому является обязательным видом кодирования;
2. **криптографическое** кодирование, или шифрование, – позволяет защитить информацию от несанкционированного доступа;
3. **эффективное,** или оптимальное, кодирование – устраняет избыточность данных путем снижения среднего числа символов кодового алфавита для представления одного исходного символа и применяется в архиваторах;
4. **помехозащитное,** или помехоустойчивое, кодирование – обеспечивает заданную достоверность в случае, когда на сигнал накладывается помеха, иначе говоря, позволяет обнаруживать и/или исправлять ошибки в кодовых комбинациях.

## 2.7. Кодирование по образцу

Большинство кодов, используемых в информатике для кодирования по образцу, имеют одинаковую длину и используют двоичную систему для представления кода (и, возможно, шестнадцатеричную как средство промежуточного представления). Рассмотрим несколько видов таких кодов: прямые; ASCII-коды; коды, учитывающие частоту информационных элементов; коды Грея.

### 2.7.1. Прямые коды

Применяются для представления в ЭВМ числовых данных и используют двоичную систему счисления.

Выполним построение прямого кода для десятичных цифр с помощью табл. 2.1. Вначале выпишем десятичные цифры и их двоичные эквиваленты (табл. 2.2):

Таблица 2.2.

Связь десятичных и двоичных чисел

| Десятичная цифра | Двоичный эквивалент |
| --- | --- |
| 0 | 0 |
| 1 | 1 |
| 2 | 10 |
| 3 | 11 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |
| 8 | 1000 |
| 9 | 1001 |

Эти коды имеют переменную длину, что неудобно для их обработки. С целью получения кодов постоянной длины кодовые комбинации дополняются незначащими нулями. Тогда прямые коды постоянной длины десятичных цифр представлены в табл. 2.3.

Таблица 2.3.

Прямые коды десятичных цифр

| Десятичные цифры | Прямые коды |
| --- | --- |
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |

Закодируем построенными кодами дискретное сообщение 1743. Для этого каждый символ сообщения заменим двоичной комбинацией из табл. 2.3. Получим: 0001011101000011.

Закодируем построенными кодами дискретное сообщение 2012 тем же способом. Получим: 0010000000010010.

Для декодирования, например, последнего сигнала он просматривается (сканируется) в направлении слева направо (т.е. в направлении его составления). Поскольку известно, что длина кодовой комбинации – 4 двоичных символа, в строке выделяются последовательно четырехсимвольные комбинации и каждая сопоставляется с кодовой таблицей (в нашем примере это табл. 2.2). В кодовой таблице ищется подходящая строка и выполняется замена исходным символом. Таким образом, процедура декодирования примера представляется макетом:

|  |  |  |  |
| --- | --- | --- | --- |
| 0010 | 0000 | 0001 | 0010 |
|  |  |  |  |
| 2 | 0 | 1 | 2 |

Прямые коды могут использоваться для кодирования и нечисловых данных.

**Пример 2.18**. Построить двоичные коды для символов *a, b, c, d.*

Пронумеруем исходные символы, начиная с нуля, и по табл. 2.1 сформируем двоичные коды для номеров символов. Тогда двоичные коды исходных символов примут вид:

|  |  |  |
| --- | --- | --- |
| *Исходные символы* | *Номер* | *Двоичный код* |
| *a* | 0 | 0 |
| *b* | 1 | 1 |
| *c* | 2 | 10 |
| *d* | 3 | 11 |

Для получения двоичного кода постоянной длины добавим незначащие нули к кодовым комбинациям для *a* и *b.* Получим табл. 2.4:

Таблица 2.4.

Прямые коды символов

|  |  |
| --- | --- |
| Исходные символы | Двоичные коды |
| *a* | 00 |
| *b* | 01 |
| *c* | 10 |
| *d* | 11 |

Закодируем полученными кодами дискретный сигнал *abba*. Для этого выполним замену каждого символа сигнала на соответствующую кодовую комбинацию из табл. 2.4. Получим: 00010100.

Декодирование выполняется аналогично рассмотренному выше.

### 2.7.2. ASCII-коды

Наиболее распространенным кодом по образцу является код ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange), который используется для внутреннего представления символьной информации в операционной системе MS DOS, в Блокноте операционной системы Windows’xx, а также для кодирования текстовых файлов в Интернете. Структура кода представлена ниже:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **A** | **B** | **C** | **D** | **E** | **F** |
| **0** | … | … |  | 0 | @ | P | ` | p | А | Р | а | … | … | … | р | Ё |
| **1** | … | … | ! | 1 | A | Q | a | q | Б | С | б | … | … | … | с | ё |
| **2** | … | … | “ | 2 | B | R | b | r | В | Т | в | … | … | … | т | / |
| **3** | … | … | # | 3 | C | S | c | s | Г | У | г | … | … | … | у | \ |
| **4** | … | … | $ | 4 | D | T | d | t | Д | Ф | д | … | … | … | ф | / |
| **5** | … | … | % | 5 | E | U | e | u | Е | Х | е | … | … | … | х | \ |
| **6** | … | … | & | 6 | F | V | f | v | Ж | Ц | ж | … | … | … | ц | → |
| **7** | … | … | ‘ | 7 | G | W | g | w | З | Ч | з | … | … | … | ч | ← |
| **8** | … | … | ( | 8 | H | X | h | x | И | Ш | и | … | … | … | ш | ↓ |
| **9** | … | … | ) | 9 | I | Y | i | y | Й | Щ | й | … | … | … | щ | ↑ |
| **A** | … | … | \* | : | J | Z | j | z | К | Ъ | к | … | … | … | ъ | ÷ |
| **B** | … | … | + | ; | K | [ | k | { | Л | Ы | л | … | … | … | ы | ± |
| **C** | … | … | , | < | L | \ | l | | | М | Ь | м | … | … | … | ь | № |
| **D** | … | … | - | = | M | ] | m | } | Н | Э | н | … | … | … | э | ¤ |
| **E** | … | … | . | > | N | ^ | n | ~ | О | Ю | о | … | … | … | ю | • |
| **F** | … | … | / | ? | O | \_ | o | ¤ | П | Я | п | … | … | … | я |  |

Таблица кодов содержит 16 столбцов и 16 строк; каждая строка и столбец пронумерованы в шестнадцатеричной системе счисления цифрами от 0 до F. Шестнадцатеричное представление ASCII-кода складывается из номера столбца и номера строки, в которых располагается символ. Так, например, ASCII-код символа 1 есть число 3116, что по правилам перевода означает 1100012. В двоичной системе код представляется восемью разрядами, т.е. двоичный ASCII-код символа 1 есть 001100012.

Данная таблица делится на две части: столбцы с номерами от 0 до 7 составляют стандарт кода – неизменяемую часть; столбцы с номерами от 8 до F являются расширением кода и используются, в частности, для кодирования символов национальных алфавитов. В столбцах с номерами 0 и 1 находятся управляющие символы, которые применяются, например, для управления принтером. Столбцы с номерами от 2 до 7 содержат знаки препинания, арифметических действий, некоторые служебные символы, а также заглавные и строчные буквы латинского алфавита. Расширение кода включает символы псевдографики, буквы национальных алфавитов и другие символы.

В приведенной таблице ASCII-кода в качестве национального выбран русский алфавит. Пустые ячейки означают, что они не используются, а ячейки с многоточием содержат символы, которые умышленно не показаны.

Закодируем ASCII-кодом название одной из операционных систем - WINDOWS (поскольку код уже готов, остается только выполнить второй этап). Для этого каждый символ исходного дискретного сигнала заменим соответствующим ASCII–кодом из таблицы (используем для краткости шестнадцатеричный формат кода, а сами кодовые комбинации отделим друг от друга для простоты восприятия):

WINDOWS

57 49 4E 44 4F 57 53

Если удалить пропуски между кодовыми комбинациями, получим дискретный сигнал, построенный из символов шестнадцатеричного алфавита: 57494E444F5753.

Закодируем теперь ASCII-кодом тот же дискретный сигнал, но «написанный» несколько по-другому – Windows. Получим: 57696Е646F7773.

Анализ показывает, что первое шестнадцатеричное число имеет меньшее значение, чем второе, т.е. 57494E444F5753<57696Е646F7773. Это позволяет компьютеру выполнять упорядочение символьных данных, используя для этого внутреннее представление любых типов данных в виде чисел. Таким образом, WINDOWS<Windows.

### 2.7.3. Коды, учитывающие частоту информационных элементов

В некоторых системах кодирования значение кода определяется частотой встречаемости кодируемого символа. Как правило, такие частоты известны для букв алфавитов естественных языков, например, английского или русского, и применяются уже давно при размещении символов на клавишах клавиатуры: наиболее часто используемые буквы располагаются на клавишах в середине клавиатуры, наиболее редко используемые – на периферии, что создает удобство работы для человека.

Учет частоты символов позволяет строить «экономные» для техники коды постоянной длины. Например, для первых ламповых компьютеров двоичная единица технически реализовалась включенной лампочкой накаливания, а двоичный ноль – выключенной лампочкой. Поэтому использовали коды с учетом частоты символов: чем больше частота символа, тем меньше в соответствующем коде единиц, т.е. тем меньше включенных лампочек применяется для представления символа в компьютере, а значит, меньше тратится электроэнергии.

Пусть известны частоты букв и русского алфавита, и служебных символов (табл. 2.5):

Таблица 2.5.

Частоты букв русского алфавита и служебных символов

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Буква | Частота |  | Буква | Частота |  | Буква | Частота |
| о | 0,090 |  | м | 0,026 |  | й | 0,010 |
| е (ё) | 0,072 |  | д | 0,025 |  | х | 0,009 |
| а | 0,062 |  | п | 0,023 |  | ж | 0,007 |
| и | 0,062 |  | у | 0,021 |  | ю | 0,006 |
| т | 0,053 |  | я | 0,018 |  | ш | 0,006 |
| н | 0,053 |  | ы | 0,016 |  | ц | 0,004 |
| с | 0,045 |  | з | 0,016 |  | щ | 0,003 |
| р | 0,040 |  | ь,ъ | 0,014 |  | э | 0,003 |
| в | 0,038 |  | б | 0,014 |  | ф | 0,001 |
| л | 0,035 |  | г | 0,013 |  | пробелы  и знаки препинания | 0,175 |
| к | 0,028 |  | ч | 0,012 |  |  |  |

Построим коды, учитывающие частоту, для символов кириллицы (заданные символы русского алфавита уже упорядочены в соответствии с их частотой по невозрастанию).

Можно утверждать, что для кодирования одного символа в нашей задаче требуется 6 двоичных разрядов (подробнее об этом см. в разделе измерения информации). Будем с помощью перебора так формировать кодовые комбинации, чтобы число единиц в соответствующих кодовых комбинациях было минимальным (табл. 2.6):

Таблица 2.6.

Коды, учитывающие частоту

| Буква | Частота | Код |  | Буква | Частота | Код |  | Буква | Частота | Код |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| о | 0,090 | 000000 |  | к | 0,028 | 100001 |  | г | 0,013 | 000111 |
| е | 0,072 | 000001 |  | м | 0,026 | 000110 |  | ч | 0,012 | 001101 |
| ё | 0.072 | 000010 |  | д | 0,025 | 001010 |  | й | 0,010 | 011001 |
| а | 0,062 | 000100 |  | п | 0,023 | 010010 |  | х | 0,009 | 110001 |
| и | 0,062 | 001000 |  | у | 0,021 | 100010 |  | ж | 0,007 | 001011 |
| т | 0,053 | 010000 |  | я | 0,018 | 001100 |  | ю | 0,006 | 010011 |
| н | 0,053 | 100000 |  | ы | 0,016 | 010100 |  | ш | 0,006 | 100011 |
| с | 0,045 | 000011 |  | з | 0,016 | 100100 |  | ц | 0,004 | 001110 |
| р | 0,040 | 000101 |  | ь | 0.014 | 011000 |  | щ | 0,003 | 011100 |
| в | 0,038 | 001001 |  | ъ | 0,014 | 101000 |  | э | 0,003 | 111000 |
| л | 0,035 | 010001 |  | б | 0,014 | 110000 |  | ф | 0,001 | 101010 |

Как видно из табл. 2.6, минимальное число единиц, равное нулю, у символа с максимальной частотой – буквы «о». У кодов букв от «е» до «н» использована только одна единица в разных позициях кодовых комбинаций. В кодах символов от «с» до «б» применены 2 единицы в разных комбинациях. В остальных случаях используются 3 единицы.

Закодируем, например, слово «окно» с помощью построенной нами кодовой таблицы: 000000100001100000000000. Как видно, здесь только три единицы (читай – включенные лампочки).

Закодируем теперь слово «шумы»: 10001110001000110010100. Здесь девять единиц, т.е. этот сигнал более энергозатратен.

### 2.7.4. Коды Грея

В качестве кодового алфавита используются двоичные символы.

Часто бывает необходимым, чтобы упорядоченные символы при двоичном кодировании различались минимальным количеством разрядов. Коды, удовлетворяющие этому условию, называются кодами Грея, или одношаговыми кодами.

Пусть надо построить код Грея для десятичных цифр. Для решения задачи можно использовать следующую последовательность действий:

1. код Грея для 0 и 1 равен 02 и 12, соответственно. Получаем два кода Грея;
2. полученные коды Грея позволяют построить матрицу такого же размера, т.е. 2х2, строки и столбцы которой поименованы построенными упорядоченными кодами Грея. Эта матрица позволяет получить коды Грея для 2х2=4 символов исходного алфавита, начиная с первого символа (обозначения строк и столбцов выделены серым фоном):

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | номера столбцов | |
|  |  | 0 | 1 |
| номера строк | 0 | **0** | **1** |
| 1 | **3** | **2** |

Как видно, в ячейках матрицы размещены кодируемые десятичные числа, включая и уже закодированные 0 и 1. Стрелки показывают, как заполняются ячейки матрицы символами исходного алфавита (выделены полужирно). Тогда код Грея для произвольного числа, размещенного в некоторой ячейке, формируется как номер строки и номер столбца для этой ячейки. Так, код Грея для числа 0 – это 002, а для 1 – это 012, для 2 – 112, для 3 – это 102.

1. получив коды Грея для четырех десятичных чисел, используем их в качестве номеров строк и столбцов, чтобы сформировать кодовые комбинации для всех (4х4=16>10) символов алфавита:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 | **0** | **1** | **2** | **3** |
| 01 | **7** | **6** | **5** | **4** |
| 11 | **8** | **9** |  |  |
| 10 |  |  |  |  |

Получаем табл. 2.7:

Таблица 2.7.

Код Грея для десятичных цифр

|  |  |
| --- | --- |
| Десятичная цифра | Код Грея |
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0011 |
| 3 | 0010 |
| 4 | 0110 |
| 5 | 0111 |
| 6 | 0101 |
| 7 | 0100 |
| 8 | 1100 |
| 9 | 1101 |

Как видно, коды упорядоченных чисел 1 и 2 различаются одним двоичным разрядом, а прямые коды этих же цифр – двумя разрядами. Аналогичную картину можно наблюдать в случае пар цифр 3 и 4; 5 и 6; 7 и 8.

Используем построенный код для кодирования числа 2012: 0011000000010011.

## 2.8. Криптографическое кодирование

В качестве символов кодирования могут использоваться как символы произвольного алфавита, так и двоичные коды. Это наиболее древний вид кодирования, поэтому современная информатика знает довольно много таких кодов. Рассмотрим только два метода: простой подстановки и Виженера.

### 2.8.1. Метод простой подстановки

Первый этап кодирования – построение кодовой таблицы – сводится к следующему: каждому символу исходного алфавита ставится в соответствие произвольный символ кодирования из какого-либо другого алфавита или из исходного – получается таблица соответствия. Само кодирование дискретного сообщения (второй этап) заключается в замене символов в сообщении в соответствии с полученной таблицей.

**Пример 2.19**. Пусть исходным является русский алфавит. Составим упомянутую таблицу соответствия, используя различные символы из таблицы ASCII-кодов в качестве кодового алфавита (табл. 2.8):

Таблица 2.8.

Таблица соответствия символов из примера 2.19

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ь | Ы | Ъ | Э | Ю | Я |
| ! | № | $ | % | ? | \* | ( | ) | { | } | [ | ] | - | | | = | « | ‘ | ~ | \ | : | \_ | < | > | ^ | , | . | 4 | 7 | в | 0 | е | ? | @ |

Тогда сообщение ИНФОРМАТИКА будет закодировано следующим образом: }=<«~|!:}]!.

Данный метод кодирования является ненадежным, так как при достаточно большой выборке закодированных сообщений при известных частотах символов исходного алфавита можно с определенной долей погрешности выполнить декодирование. В самом деле, пусть есть представительная выборка закодированных русскоязычных сообщений, общее число букв в которых равно *M*, и известны частоты букв русского алфавита (табл. 2.9):

Таблица 2.9.

Частоты букв кириллицы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Буква | Частота | Буква | Частота | Буква | Частота |
| о | 0,090 | м | 0,026 | й | 0,010 |
| е (ё) | 0,072 | д | 0,025 | х | 0,009 |
| а | 0,062 | п | 0,023 | ж | 0,007 |
| и | 0,062 | у | 0,021 | ю | 0,006 |
| т | 0,053 | я | 0,018 | ш | 0,006 |
| н | 0,053 | ы | 0,016 | ц | 0,004 |
| с | 0,045 | з | 0,016 | щ | 0,003 |
| р | 0,040 | ь,ъ | 0,014 | э | 0,003 |
| в | 0,038 | б | 0,014 | ф | 0,001 |
| л | 0,035 | г | 0,013 | пробелы и знаки препинания | 0,175 |
| к | 0,028 | ч | 0,012 |  |  |

Можно рассчитать частоту *fS* каждого s-го символа по формуле (2.2):

(2.2)



где *mS* – количество s-х символов в сообщениях (иначе – абсолютная частота s-того символа).

Тогда получив частоты и сопоставив их с приведенной выше таблицей частот, можно определить исходный текст.

**Пример 2.20**. Пусть есть закодированное сообщение из примера 2.19: }=<»~|!:}]!

Известно, что до кодирования оно было составлено из букв русского алфавита. Требуется декодировать его, используя в качестве представительной выборки закодированных русскоязычных текстов настоящий конспект, предварительно выполнив все замены русских букв символами из таблицы соответствия примера 2.19.

Воспользуемся встроенными средствами текстового процессора WINWORD для определения требуемых статистических данных.

Определим, что общее число символов *М* в конспекте на момент подготовки данного примера составляет 275979 символов.

Определяем, сколько раз встречаются интересующие нас символы из закодированного сообщения - *ms* (табл. 2.10, строка *ms*). Это позволяет рассчитать частоты символов *fs* по формуле (2.2) и заполнить строку *fs* табл. 2.10.

Сопоставим теперь полученные данные с таблицей частот. Наиболее близкие по значению символы для полученных частот сведены в табл. 2.10 в последнюю строку:

Таблица 2.10.

Таблица близости символов по частоте

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| s-й символ | **}** | **=** | **<** | **«** | **~** | **|** | **!** | **:** | **]** |
| *ms* | 18716 | 14396 | 1436 | 22027 | 12058 | 8503 | 16835 | 13426 | 6592 |
| *fs* | 0,068 | 0,052 | 0,005 | 0,078 | 0,044 | 0,031 | 0,061 | 0,049 | 0,024 |
| подходящий символ из таблицы частот | е,а,и | т,н | ю,ш,ц | е,о | с,р | л,к | а,и | т,н,с | д,п |

Таким образом, кодовые символы из закодированного сообщения могут быть заменены символами из соответствующего множества:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| } | = | < | « | ~ | | | ! | : | } | ] | ! | исходные символы |
|  |  |  |  |  |  |  |  |  |  |  |  |
| е,а,и | т,н | ю,ш,ц | е,о | с,р | л,к | а,и | т,н,с | е,а,и | д,п | а,и | символы для замены |

Если построить все возможные сочетания символов из указанных множеств, там будет, в частности, и сочетание вида и н \* о р \* а т и \* а, где знак \* означает любой символ из соответствующего определенного выше множества исходных символов (в случае \* декодирование, очевидно, выполнено неверно). Если предъявить полученную строку человеку или автомату, способному распознать русское слово, зашифрованное сообщение можно считать декодированным.

Очевидно, декодирование также возможно при известной таблице соответствия.

### 2.8.2. Метод Виженера

Разрушить статистические зависимости в закодированных сообщениях и тем самым повысить надежность кодирования можно с помощью метода Виженера.

Первый этап кодирования (построение кодовой таблицы) сводится к тому, что символы исходного алфавита нумеруются, начиная с нуля, например, для кириллицы (табл. 2.11):

Таблица 2.11.

Таблица соответствия для метода Виженера

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ь | Ы | Ъ | Э | Ю | Я |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |

Затем задаются ключом кодирования – словом в исходном алфавите, например, АСУ.

Собственно кодирование (второй этап) выполняется следующим образом: выписывают дискретный сигнал, подлежащий кодированию, например, пусть это будет сообщение ИНФОРМАТИКА, и выполняют следующие шаги:

а) под каждым его символом записывают порядковый номер из таблицы соответствия:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| И | Н | Ф | О | Р | М | А | Т | И | К | А |
| 9 | 14 | 21 | 15 | 17 | 13 | 0 | 19 | 9 | 11 | 0 |

б) под сообщением многократно выписывают ключевое слово:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| И | Н | Ф | О | Р | М | А | Т | И | К | А |
| 9 | 14 | 21 | 15 | 17 | 13 | 0 | 19 | 9 | 11 | 0 |
| А | С | У | А | С | У | А | С | У | А | С |

в) под символами ключа выписывают их порядковые номера из таблицы соответствия:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| И | Н | Ф | О | Р | М | А | Т | И | К | А |
| 9 | 14 | 21 | 15 | 17 | 13 | 0 | 19 | 9 | 11 | 0 |
| А | С | У | А | С | У | А | С | У | А | С |
| 0 | 18 | 20 | 0 | 18 | 20 | 0 | 18 | 20 | 0 | 18 |

г) порядковые номера символов складываются по модулю, равному числу символов исходного алфавита (в нашем случае – 33):

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| И | Н | Ф | О | Р | М | А | Т | И | К | А |
| 9 | 14 | 21 | 15 | 17 | 13 | 0 | 19 | 9 | 11 | 0 |
| А | С | У | А | С | У | А | С | У | А | С |
| 0 | 18 | 20 | 0 | 18 | 20 | 0 | 18 | 20 | 0 | 18 |
| 9 | 32 | 8 | 15 | 2 | 0 | 0 | 4 | 29 | 11 | 18 |

Напомним, что сложение по модулю (обозначается ⊕) выполняется без перемещения единицы переноса в старший разряд. Так мы получили, например, при сложении по модулю 33 чисел 21 и 20 (сумма равна 41, что на 8 превышает модуль 33) значение 8,

д) полученный числовой ряд преобразуется в символы исходного алфавита по таблице соответствия (табл. 2.11):

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| И | Н | Ф | О | Р | М | А | Т | И | К | А |
| 9 | 14 | 21 | 15 | 17 | 13 | 0 | 19 | 9 | 11 | 0 |
| А | С | У | А | С | У | А | С | У | А | С |
| 0 | 18 | 20 | 0 | 18 | 20 | 0 | 18 | 20 | 0 | 18 |
| 9 | 32 | 8 | 15 | 2 | 0 | 0 | 4 | 29 | 11 | 18 |
| И | Я | З | О | В | А | А | Д | Ъ | К | С |

Таким образом, вместо дискретного сигнала ИНФОРМАТИКА имеем сообщение ИЯЗОВААДЪКС.

Очевидно, что статистика не поможет декодировать это сообщение, поскольку повторяются совсем не те символы, что в исходном сообщении.

Для декодирования подобных сообщений требуется таблица соответствия и ключ. Тогда выполняют описанные выше процедуры кодирования в обратном порядке. Сложность может представлять только операция вычитания с учетом модуля. При этом следует помнить, что не должны получаться отрицательные значения. Если такое происходит, нужно занять число, соответствующее модулю.

**Пример 2.21**. Декодировать сообщение ИЯЗОВААДЪКС, задавшись ключом АСУ и зная таблицу соответствия:

а) выписываем под закодированным сообщением порядковые номера символов из таблицы соответствия:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| И | Я | З | О | В | А | А | Д | Ъ | К | С |
| 9 | 32 | 8 | 15 | 2 | 0 | 0 | 4 | 29 | 11 | 18 |

б) выписываем под сообщением ключ с порядковыми номерами символов:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| И | Я | З | О | В | А | А | Д | Ъ | К | С |
| 9 | 32 | 8 | 15 | 2 | 0 | 0 | 4 | 29 | 11 | 18 |
| А | С | У | А | С | У | А | С | У | А | С |
| 0 | 18 | 20 | 0 | 18 | 20 | 0 | 18 | 20 | 0 | 18 |

в) вычитаем с учетом модуля 33 из чисел в закодированном сообщении числа для ключа:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| И | Я | З | О | В | А | А | Д | Ъ | К | С |
| 9 | 32 | 8 | 15 | 2 | 0 | 0 | 4 | 29 | 11 | 18 |
| А | С | У | А | С | У | А | С | У | А | С |
| 0 | 18 | 20 | 0 | 18 | 20 | 0 | 18 | 20 | 0 | 18 |
| 9 | 14 | 21 | 15 | 17 | 13 | 0 | 19 | 9 | 11 | 0 |

г) преобразуем числа в символы по таблице соответствия:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| И | Я | З | О | В | А | А | Д | Ъ | К | С |
| 9 | 32 | 8 | 15 | 2 | 0 | 0 | 4 | 29 | 11 | 18 |
| А | С | У | А | С | У | А | С | У | А | С |
| 0 | 18 | 20 | 0 | 18 | 20 | 0 | 18 | 20 | 0 | 18 |
| 9 | 14 | 21 | 15 | 17 | 13 | 0 | 19 | 9 | 11 | 0 |
| И | Н | Ф | О | Р | М | А | Т | И | К | А |

При декодировании возникла сложность в получении кодов символов Ф, Р, М, Т. В самом деле, при вычитании из 8 числа 20 получалось –12. Тогда к 8 прибавили модуль 33, получили 41 и уже из 41 вычли 20. Получили 21 – порядковый номер символа Ф. Аналогично поступили и с остальными проблемными символами.

## 2.9. Эффективное кодирование

Методы эффективного кодирования делятся на две группы - универсальные, применяемые к любым дискретным сигналам, и специальные, ориентированные на дискретные сигналы определенного типа.

### 2.9.1. Универсальные методы

Для кодирования символов исходного алфавита необходимо знание частот *fs* символов исходного алфавита. Причем должно выполняться условие (2.3), показывающее, что при построении кода использован полный алфавит:

(2.3)



где *k* – число символов исходного алфавита.

Строятся двоичные коды переменной длины по принципу: чем больше частота символа, тем короче его код. Эффективность кода определяется средним числом двоичных разрядов для кодирования одного символа – *l*ср по формуле (2.4):

(2.4)



где *ns* – число двоичных разрядов для кодирования s-го символа;

*fs –* частота s-го символа.

Существуют два универсальных метода эффективного кодирования: Шеннона-Фано и Хаффмена. Входными данными для обоих методов является множество исходных символов с частотами; результат - эффективные коды.

#### 2.9.1.1. Метод Шеннона-Фано

Для построения кодов требуется упорядочение исходного множества символов по невозрастанию их частот. Затем выполняются следующие шаги:

а) список символов делится на две части (назовем их первой и второй частями) так, чтобы суммы частот обеих частей (назовем их Σ1 и Σ2) были точно или примерно равны. В случае когда точного равенства достичь не удается разница между суммами должна быть минимальна;

б) кодовым комбинациям первой части приписывается 1, кодовым комбинациям второй части приписывается 0;

в) анализируют первую часть: если она содержит только один символ, работа с ней заканчивается, – считается, что код для ее символов построен, и выполняется переход к шагу г) для построения кода второй части. Если символов больше одного, переходят к шагу а) и процедура повторяется с первой частью как с самостоятельным упорядоченным списком;

г) анализируют вторую часть: если она содержит только один символ, работа с ней заканчивается и выполняется обращение к оставшемуся списку (шаг д). Если символов больше одного, переходят к шагу а), и процедура повторяется со второй частью как с самостоятельным списком;

д) анализируется оставшийся список: если он пуст – код построен, работа заканчивается. Если нет – выполняется шаг а).

**Пример 2.22**. Даны символы *a, b, c, d* с частотами *fa=*0,5*; fb=*0,25*; fc=*0,125*; fd*=0,125. Построить эффективный код методом Шеннона-Фано.

Сведем все построение в таблицу (табл. 2.12), где разместим исходные данные, упорядочив их, как требует метод.

Первая линия деления проходит под символом *a*: соответствующие суммы Σ1 и Σ2 равны между собой и равны 0,5. Тогда формируемым кодовым комбинациям дописывается 1 для верхней (первой) части и 0 для нижней (второй) части. Поскольку это первый шаг формирования кода, двоичные цифры не дописываются, а только начинают формировать код. В силу того, что верхняя часть списка содержит только один элемент (символ *а*), работа с ней заканчивается, а эффективный код для этого символа считается сформированным.

Второе деление выполняется под символом *b*: суммы частот Σ1 и Σ2 вновь равны между собой и равны 0,25. Тогда кодовой комбинации символов верхней части дописывается 1, а нижней части – 0. Таким образом, к полученным на первом шаге фрагментам кода, равным 0, добавляются новые символы. Поскольку верхняя часть нового списка содержит только один символ (*b*), формирование кода для него закончено.

Третье деление проходит между символами *c* и *d*: к кодовой комбинации символа *c* приписывается 1, коду символа *d* приписывается 0.

Таким образом, получили коды:

*a* - 1,

*b* - 01,

*c* - 001,

*d* - 000.

Определим эффективность построенного кода:

*lср* = 0,5\*1 + 0,25\*01 + 0,125\*3 + 0,125\*3 = 1,75.

Поскольку при кодировании четырех символов кодом постоянной длины требуется два двоичных разряда (см. примеры ранее), сэкономлено 0,25 двоичного разряда в среднем на один символ.

Закодируем дискретный сигнал *abba* построенным кодом: 101011.

Таблица 2.12.

Построение эффективного кода методом Шеннона-Фано

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Исход-  ные  сим-  волы | Час-  тоты символов | Этапы построения кода | | | Формируемый код | | |
| первое деление | второе деление | третье деление | первое  деление | второе  деление | третье  деление |
| *a*  линия деления | 0,5 | Σ1=0,5 | код для символа *a* сформирован | | 1 | - | - |
| *b*  линия деления | 0,25 |  | Σ1 = 0,25 | код для символа *b* сформирован | 0 | 1 | - |
| *c*  линия деления | 0,125 | Σ2=0,25+0,125+0,125=0,5 | Σ2=0,125+0,125=0,25 | Σ1 = 0,125  Σ2 = 0,125 | 0 | 0 | 1 |
| *d* | 0,125 |  | 0 | 0 | 0 |

#### 2.9.1.2. Метод Хаффмена

Этот метод имеет два преимущества по сравнению с методом Шеннона-Фано: он устраняет неоднозначность кодирования, возникающую из-за примерного равенства сумм частот при разделении списка на две части (линия деления проводится неоднозначно), и имеет, в общем случае, большую эффективность кода.

Для построения кодовой таблицы исходное множество символов упорядочивается по невозрастанию частоты и выполняются следующие шаги:

1) объединение частот:

* две последние частоты складываются, а соответствующие символы исключаются из списка;
* оставшийся после исключения символов список пополняется полученной в предыдущем пункте суммой частот и вновь упорядочивается;
* предыдущие шаги повторяются до тех пор, пока не получится единица в результате суммирования и список не уменьшится до одного символа;

2) построение кодового дерева:

* строится двоичное кодовое дерево: корнем его является вершина, полученная в результате объединения частот, равная 1; листьями – исходные вершины; остальные вершины соответствуют либо суммарным, либо исходным частотам, причем для каждой вершины левая подчиненная вершина соответствует большему слагаемому, а правая – меньшему; ребра дерева связывают вершины-суммы с вершинами-слагаемыми. Структура дерева показывает, как происходило объединение частот;
* ребра дерева кодируются: каждое левое кодируется единицей, каждое правое – нулем;

3) формирование кода: для получения кодов листьев (исходных кодируемых символов) продвигаются от корня к нужной вершине и «собирают» коды проходимых ребер.

**Пример 2.23**. Даны символы *a, b, c, d* с частотами *fa=*0,5*; fb=*0,25*; fc=*0,125*; fd*=0,125. Построить эффективный код методом Хаффмена.

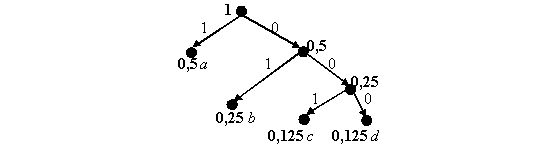
1. объединение частот (табл. 2.13):

Таблица 2.13.

Объединение частот

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исходные символы *s* | Частоты *fs* | Этапы объединения | | |
| первый | второй | третий |
| *a* | 0,5 | 0,5 | 0,5 | 1 |
| *b* | 0,25 | 0,25 | 0,5 |  |
| *c* | 0,125 | 0,25 |  |  |
| *d* | 0,125 |  |  |  |

1. построение кодового дерева:



1. формирование кода:

*a -* 1;

*b -* 01;

*c -* 001;

*d -* 000.

Как видно, полученные коды совпадают с теми, что были сформированы методом Шеннона-Фано, следовательно, они имеют одинаковую эффективность.

Закодируем дискретное сообщение *abba* построенным кодом: 101011.

#### 2.9.1.3. Повышение эффективности кодирования

Повысить эффективность кодирования можно, строя код не для символа, а для блоков из n символов. Рассмотрим этот тезис на примере.

**Пример 2.24**. Даны символы *a* и *b* с частотами, соответственно, 0,9 и 0,1. Построить эффективный код методом Шеннона-Фано для блоков из двух символов (n=2).

Сформируем список возможных блоков и их частот. При этом частоту блока будем рассчитывать как произведение частот символов, входящих в блок. Тогда имеем:

***Блоки исходных Частоты блоков***

***символов f блока = f1 \* f2***

*aa* 0,81

*ab*  0,09

*ba* 0,09

*bb* 0,01

Построение кода сведем в табл. 2.14.

Таблица 2.14.

Метод Шеннона-Фано для блоков символов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Блоки исходных символов | Частоты блоков | Этапы построения кода | | |
| первый | второй | третий |
| *aa* | 0,81 | 1 | код построен | |
| *ab* | 0,09 | 0 | 1 | код построен |
| *ba* | 0,09 | 0 | 0 | 1 |
| *bb* | 0,01 | 0 | 0 | 0 |

Таким образом, получены коды:

*aa -* 1;

*ab -*  01;

*ba* - 001;

*bb -* 000.

Определим эффективность построенного кода. Для этого рассчитаем сначала показатель эффективности для блока символов:

*lср блока =* 0,81\*1 + 0,09\*2 + 0,09\*3 + 0,01\*3 = 1,28.

Поскольку в блоке два символа (n=2), для одного символа

При посимвольном кодировании для эффективного кода потребуется по одному двоичному разряду. В самом деле, применение метода Шеннона-Фано дает результат, представленный в табл. 2.15.

Таблица 2.15.

Метод Шеннона-Фано для отдельного символа

|  |  |  |
| --- | --- | --- |
| Исходные символы | Частоты символов | Построение кода |
| *a* | 0,9 | 1 |
| *b* | 0,1 | 0 |

Таким образом, при блочном кодировании выигрыш составил 1-0,64=0,36 двоичных разрядов на один кодируемый символ в среднем.

Эффективность блочного кодирования тем выше, чем больше символов включается в блок.

Закодируем построенным кодом дискретный сигнал *abba*: 01001.

#### 2.9.1.4. Декодирование универсальных эффективных кодов

Особенностью эффективных кодов является переменное число двоичных разрядов в получаемых кодовых комбинациях. Это затрудняет процесс декодирования.

Тем не менее подобные закодированные сообщения могут декодироваться благодаря свойству **префиксности** эффективных кодов: ни одна более короткая кодовая комбинация не является началом ни одной другой более длинной кодовой комбинации.

Для раскрытия данного тезиса вернемся к кодам из примера 2.23. Там самым коротким кодом был код для символа *a* со значением 1. Как видно, ни один другой код (более длинный) не имеет в своем начале символ 1. Второй по длине код для символа *b* имеет значение 01 и, как показывает анализ, не является началом ни для кода 001, ни для кода 000. Таким образом, код из примера 2.23 является префиксным.

Свойство префиксности позволяет декодировать сообщения, закодированные эффективными кодами.

Пусть получено сообщение 1010010001, составленное из кодов, построенных в примере 2.23:

*a -* 1;

*b -* 01;

*c -* 001;

*d -* 000.

Выполним его декодирование.

В сообщении слева направо выделяется по одному двоичному символу и делается попытка декодирования в соответствии с таблицей кодов. Если попытка успешна, двоичный символ (или символы) исключается из исходной цепочки и заменяется соответствующим исходным символом. Если попытка не удается, во входной цепочке выделяется следующий двоичный символ и уже с двумя двоичными символами делается попытка их декодирования по таблице кодов. Если попытка и тогда неудачна, выделяют следующий третий и т.д.

Итак, имеем:

направление просмотра цепочки

1 0 1 0 0 1 0 0 0 1

*a* -  -  - *a*

*b*  - -

*с d*

Здесь знак «-» означает, что попытка декодирования не удалась.

Таким образом, при декодировании получили строку *abcda.*

Отметим, что методы Шеннона-Фано и Хаффмена строят префиксные коды.

### 2.9.2. Специальные методы эффективного кодирования

В зависимости от типа исходного сообщения эти методы делятся на методы эффективного кодирования числовых последовательностей, словарей, текстов. Отличительная черта этих методов – отсутствие необходимости построения кодовой таблицы.

#### 2.9.2.1. Методы эффективного кодирования числовых последовательностей

Различают два метода – разностное кодирование и кодирование повторений.

Суть **разностного кодирования** заключается в хранении вмес­то абсолютных значений либо разностей двух смежных чисел, либо отк­лонения чисел от их среднего значения. Например, для последо­вательности чисел: 2 14 18 27 34 первый способ даст после­довательность: 2 12 4 9 7.

Здесь первое число представлено в исходном виде, а все последующие – как отклонение от предыдущего числа в исходной последовательности.

Этот метод эффективен для монотонно возрастающих или монотонно убывающих после­довательностей. Его недостаток состоит в том, что для получения значения n-го члена последовательности надо декодировать все предыдущие (n-1) членов последовательности.

Второй способ порождает последо­вательность: -17 -5 -1 8 15, поскольку среднее значение для исходной последовательности - 19.

Этот способ эффективен, когда максимальное отклонение от сред­него значительно меньше абсолютного значения среднего. Достоинство данного подхода заключается в независимости декодирования любого n-го члена числовой последовательности от декодирования остальных ее составляющих: для этого нужно знать только значение среднего арифметического данной последовательности, что вынуждает хранить это число в некоторой фиксированной позиции вместе с самой закодированной последовательностью.

Таким образом, реальная закодированная последовательность из примера имеет несколько другой вид, что снижает эффективность: -17 -5 -1 8 15 19. Здесь среднее арифметическое записано крайним справа.

Оба метода могут использоваться не только для эффективного кодирования прикладных массивов данных (тех, которые создает пользователь компьютера), но и для сжатия любой информации во внутреннем представлении. В самом деле, внутреннее представление символьной информации выполнено с использованием одной из систем кодирования по образцу, например ASCII-кода, который представляет собой двузначные шестнадцатеричные числа для каждого кодируемого символа. Иными словами, внутреннее представление любой информации – массив двузначных шестнадцатеричных чисел, к которому может быть применен один из указанных выше методов.

**Кодирование повторений** заключается в замене цепочки оди­наковых цифровых символов самим символом и числом повторений (возможно включение разделителей). Напри­мер, для последовательности: 55556666888888 применение этого способа даст последовательность: 5(4)6(4)8(6), где круглые скобки играют роль разделителей.

Данный метод может быть использован для эффективного кодирования растровых форматов изображений. Растровыми называются форматы изображений, которые получаются во время ввода изображения путем кодирования каждой точки – пиксела (pixel – **PI**сture **EL**ement) – двумерного пространства, на котором расположено исходное изображение, даже если эта точка не содержит самого изображения. Очевидно, в общем случае, изображение занимает не все пространство. Тем не менее кодированию подлежат и «пустоты», при этом те точки, которые содержат изображение, в простейшем случае (для монохромных изображений) кодируются двоичной 1, точки без изображения кодируются двоичным 0. В результате получаются числовые последовательности, подобные следующей:

00000000000000000000000000000000000000000000000010000000000000000000.

Переведем эту двоичную последовательность в набор шестнадцатеричных цифр, используя тетрады. Получим последовательность шестнадцатеричных цифр: 00000000000080000.

Очевидно, к таким последовательностям можно применить метод кодирования повторений. В результате для нашего случая получим (круглые скобки используем как разделители): 0(С)8000, что означает: 0 повторяется 12 раз (С16 = 12), для остальных символов число повторений не вводится.

Поскольку результирующая последовательность должна также быть шестнадцатеричной, полученное выражение преобразуем следующим образом: заменим круглые скобки соответствующими ASCII-кодами. Тогда открывающей скобке соответствует код 2816, закрывающей – 2916. Получим: 028С2980000.

Длина результата меньше исходной последовательности (11 символов против 17), поэтому получен эффект в 6 символов.

#### 2.9.2.2. Методы эффективного кодирования словарей

Словари очень часто применяются в современных прикладных программных продуктах, например, при проверке орфографии в текстовом процессоре WinWord. Для их рационального хранения применяются рассмотренные ниже методы.

Обычно в словарях слова упорядочены по алфавиту, поэтому для эффективного кодирования словарей можно применить метод, аналогичный разностному кодированию для числовых последовательностей: у каждого n-го слова отбрасываются начальные буквы, совпадающие с начальными буквами предыдущего (n-1)-го слова, и заменяются на количество отброшенных букв.

Пусть есть фрагмент словаря со словами: вычислитель, вычислительный, вычислять.

Очевидно, у двух подряд расположенных слов есть общие буквы в начале. Тогда можно выполнить следующую замену: вычислитель, 11ный , 6ять , где числа означают, сколько букв из предыдущего слова надо взять, чтобы восстановить данное слово. Здесь слово «вычислитель» можно рассматривать как некоторое базовое слово, которое не подвергается кодированию.

Таким образом, имеем:

***исходный словарь закодированный словарь***

вычислитель вычислитель

вычислительный 11ный

вычислять 6ять

Недостаток данного подхода заключается в необходимости декодировать все (n-1) слов, начиная с базового слова, для декодирования n-го слова словаря.

Второй подход состоит в том, что формируется вспомогательный словарь, в который включаются наиболее часто повторяющиеся части слов (в нашем случае, например, это фрагменты слов «вычисл» и «итель»). Каждому такому фрагменту назначается код, например, его порядковый номер во вспомогательном словаре. Затем в основном словаре фрагменты слов заменяются их кодами из вспомогательного словаря.

Для нашего примера будут сформированы два словаря:

**основной вспомогательный**

12 1 - вычисл

12ный 2 - итель

1ять

Таким образом, при заданном вспомогательном словаре выполняется преобразование исходного словаря в закодированный:

***исходный словарь закодированный словарь***

вычислитель 12

вычислительный 12ный

вычислять 1ять

#### 2.9.2.3. Методы эффективного кодирования естественно-языковых текстов

Наиболее распространенным и эффективным является адаптивный алгоритм, который в литературе называется также алгоритмом Зива (по имени его разработчика) или алгоритмом с указателями назад (или вперед).

В соответствии с этим алгоритмом в исходном тексте ищутся повторяющиеся фрагменты, каждый последующий из которых заменяется указателем на такой же фрагмент, который встречался ранее в начале (указатель назад) или в конце (указатель вперед) текста. В первом случае текст просматривается от начала к концу, во втором – от конца к началу.

Например, пусть задан исходный текст:

«**информа**тика изучает способы обработки **информа**ции».

В этом тексте дважды повторяются фрагменты «информа» длиной 7 символов (они выделены полужирно). Построим эффективно закодированный текст:

а) с указателями назад. Текст просматривается от начала к концу, и специальный механизм определяет, что повторно встречается указанный выше фрагмент. Алгоритм строит адресный указатель взамен этого фрагмента со структурой: количество символов, которые надо отсчитать в обратном направлении к началу первого вхождения фрагмента, и длина фрагмента. Таким образом, для нашего примера имеем:

«**информа**тика изучает способы обработки 38/7ции»,

38 символов

где знак / играет роль разделителя.

б) с указателями вперед. Текст просматривается от конца к началу, и специальный механизм определяет, что повторно встречается указанный выше фрагмент. Алгоритм строит адресный указатель взамен этого фрагмента со структурой: количество символов, которые надо отсчитать в обратном направлении к началу первого вхождения фрагмента, и длина фрагмента.

Таким образом, для нашего примера имеем:

«31/7тика изучает способы обработки **информа**ции».

31 символ

Очевидно, в обоих случаях полученный результат короче исходного текста.

## 2.10. Помехозащитное кодирование

В качестве базового кода, который подвергается помехозащитному кодированию, используется двоичный код постоянной длины. Такой исходный (базовый) код называется **первичным**, поскольку подвергается модификации.

### 2.10.1. Искажение кодовых комбинаций

Для понимания сущности вопроса рассмотрим сначала, как происходит искажение кодовых комбинаций при наличии помех в каналах связи.

При передаче по каналу связи на передаваемый код накладывается помеха, которая формально представляется вектором ошибки - кодовой комбинацией с числом разрядов, равным числу разрядов передаваемого кода, причем эта кодовая комбинация содержит 1 в искажаемых разрядах. С помехой связано понятие ее кратности *q* – это число искажаемых разрядов, т.е. число единиц в векторе ошибки.

Искажение рассматривается как сложение по модулю 2 исходной кодовой комбинации *a1a2…ak* и вектора ошибки *b1b2…bk*: *a1a2…ak* ⊕ *b1b2…bk* = *c1c2…ck ,*где *c1c2…ck* – искаженная кодовая комбинация.

Пусть имеется таблица кодов (табл. 2.16):

Таблица 2.16.

Прямые коды

|  |  |
| --- | --- |
| Исходные символы | Прямые коды |
| *a* | 00 |
| *b* | 01 |
| *c* | 10 |
| *d* | 11 |

Пусть на передаваемые кодовые комбинации накладывается помеха с кратностью ошибки 1, т.е. соответствующие ошибке кодовые комбинации – элементы множества {01, 10}.

Передается кодовая комбинация 10 (код символа *c*). Тогда возможное искажение представлено в табл. 2.17.

Таблица 2.17.

Результат искажения

|  |  |  |  |
| --- | --- | --- | --- |
| Передаваемая  кодовая  комбинация | Вектор  ошибки | Принимаемая  кодовая  комбинация | Результат  декодирования |
| 10 | 01 | 11 | *d* |
| 10 | 10 | 00 | *a* |

Таким образом, в результате ошибки принимающая сторона вместо символа *c* примет символ *d* или *a* и ошибка даже не будет обнаружена.

### 2.10.2. Кодовое расстояние и корректирующая способность кода

Под **корректирующей способностью кода** понимается его свойство обнаруживать и/или исправлять ошибку максимальной кратности *q*. Корректирующая способность кода связана с его кодовым расстоянием.

**Расстоянием** *dij* **между** кодовыми комбинациями i и j называется число различных разрядов в кодовых комбинациях i и j. Например, если есть коды 01 и 10, расстояние между ними равно 2: они различаются в двух разрядах.

**Кодовым расстоянием** *d* для кода, содержащего *m* кодовых комбинаций, является минимальное расстояние между всеми парами кодовых комбинаций, т.е. *d=*min{*dij*}.

Определим кодовое расстояние для кода из табл. 2.16:

*dab =* 1; *dad* = 2; *dbd* = 1; *dac* = 1; *dbc* = 2; *dcd* = 1.

Тогда *d* = min{1,2,1,1,2,1} = 1. Это означает, что всякая ошибка кратности 1 (и более) переводит исходную кодовую комбинацию в другую кодовую комбинацию, которая также принадлежит коду.

Увеличить кодовое расстояние можно, введя в кодовые комбинации дополнительный разряд (или разряды). Тогда начальные разряды называют **информационными**, а дополнительный (или дополнительные) – **проверочным** (проверочными).

Значение одного проверочного разряда в простейшем случае определяется как результат суммирования по модулю 2 информационных разрядов.

Вернемся к нашей таблице кодов, введем дополнительный разряд и сформируем его значение (табл. 2.18):

Таблица 2.18.

Помехозащитный код

|  |  |  |  |
| --- | --- | --- | --- |
| Исходные  символы | Информационные  разряды кода | Проверочный  разряд кода | Результирующий  код |
| *a* | 00 | 0 | 000 |
| *b* | 01 | 1 | 011 |
| *c* | 10 | 1 | 101 |
| *d* | 11 | 0 | 110 |

Таким образом, полученный код является трехразрядным.

Определим кодовое расстояние полученного кода:

*dab =* 2; *dad* = 2; *dbd* = 2; *dac* = 2; *dbc* = 2; *dcd* = 2.

Тогда *d* = min{2,2,2,2,2,2} = 2.

Пусть передается кодовая комбинация, соответствующая символу *c*, – 101. Пусть на нее накладывается ошибка кратности 1. Возможные результаты искажения приведены в табл. 2.19.

Таблица 2.19.

Результаты искажения

|  |  |  |  |
| --- | --- | --- | --- |
| Передаваемая  кодовая  комбинация | Вектор  ошибки | Принимаемая  кодовая  комбинация | Результат  декодирования |
| 101 | 001 | 100 | Невозможно декодировать |
| 101 | 010 | 111 | То же |
| 101 | 100 | 001 | “-“ |

В результате данной ошибки получаемые кодовые комбинации невозможно декодировать, так как они отсутствуют в таблице.

Последний пример дает возможность ввести понятия разрешенных и запрещенных кодовых комбинаций.

**Разрешенными кодовыми комбинациями** называются те, которые соответствуют символам исходного алфавита. Их количество равно числу исходных символов (*m*). **Запрещенные кодовые комбинации** – это те, которые отсутствуют в исходной кодовой таблице. Их количество определяется по формуле: 2*r*– *m*, где *r* – общее число двоичных разрядов (информационные плюс проверочные) в коде.

Сформируем все разрешенные и запрещенные кодовые комбинации для нашего кода, при этом используем схему формирования кода Грея (обозначения строк – исходные коды, обозначения столбцов – значения проверочных разрядов):

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 00 | *a* |  |
| 01 |  | *b* |
| 11 | *d* |  |
| 10 |  | *c* |

Здесь пустые ячейки означают запрещенные кодовые комбинации. Как видно, отстояние кодовых комбинаций для исходных символов *a, b, c, d* равно двум разрядам:

* символы, находящиеся в одном столбце (*a* и *d*, *b* и *c*), имеют одинаковый проверочный разряд, но находятся в несмежных строках, которые различаются двумя разрядами;
* символы, находящиеся в смежных строках (*a* и *b*, *b* и *d*, *d* и *c*), которые различаются одним разрядом, расположены попарно в разных столбцах, имеющих различное обозначение.

Поэтому при наличии ошибки кратности 1 кодовая комбинация переходит в соседнюю запрещенную комбинацию.

До введения проверочного разряда формирование исходного кода можно было представить схемой, показанной ниже:

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | *a* | *b* |
| 1 | *c* | *d* |

Поскольку символы расположены «плотно» в схеме, всякое искажение кода приводило к попаданию в другую ячейку с кодом.

Существует связь между кодовым расстоянием *d* и минимальной кратностью ошибки *q*, которую код может обнаруживать: *d* ≥ *q* + 1.

**Пример 2.25**. На базе кода табл. 2.16 построить код, обнаруживающий ошибки кратности 2.

Воспользуемся схемой формирования кода Грея с некоторыми модификациями.

Поскольку код для обнаружения ошибки кратностью 1 построен, используем его для обозначения строк схемы, причем с каждой строкой свяжем символ, который соответствует данной кодовой комбинации: так с первой строкой свяжем символ *a*, со второй – *b* и т.д. Очевидно, кодовые комбинации в обозначении строк схемы различаются двумя разрядами.

Поскольку в ячейках этой схемы следует расположить символы, расстояние между кодовыми комбинациями которых должно быть не меньше 3, они должны быть расположены в соседних столбцах, чтобы обеспечивать различимость кодовых комбинаций еще как минимум в одном разряде (строки расположения символов обговорены выше).

С учетом сделанных замечаний схема имеет четыре столбца и четыре строки и представлена ниже:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 000 | *a* |  |  |  |
| 011 |  | *b* |  |  |
| 110 |  |  | *d* |  |
| 101 |  |  |  | *c* |

Таким образом, построен следующий код:

00000 → *a*, 01101 → *b*, 11011 → *d*, 10110 → *c.*

Определим кодовое расстояние *d* построенного кода:

*dab =* 3; *dad* = 4; *dbd* = 3; *dac* = 3; *dbc* = 4; *dcd* = 3.

Тогда *d =* min{3,4,3,3,4,3} = 3.

Проверим, обнаруживает ли построенный код ошибку кратности 2. Для этого зададимся произвольной кодовой комбинацией, например, 01101 (символ *b*). Результат проверок приведен в табл. 2.20:

Таблица 2.20.

Результат декодирования

| Передаваемая  кодовая  комбинация | Вектор  ошибки | Принимаемая  кодовая  комбинация | Результат  декодирования |
| --- | --- | --- | --- |
| 01101 | 00011 | 01110 | Невозможно декодировать |
| 01101 | 00101 | 01000 | То же |
| 01101 | 00110 | 01011 | “-“ |
| 01101 | 01001 | 00100 | “-“ |
| 01101 | 01010 | 00111 | “-“ |
| 01101 | 01100 | 00001 | “-“ |
| 01101 | 10001 | 11100 | “-“ |
| 01101 | 10010 | 11111 | “-“ |
| 01101 | 10100 | 11001 | “-“ |
| 01101 | 11000 | 10101 | “-“ |

Таким образом, задача решена.

### 2.10.3. Коды, исправляющие ошибки

Особое значение имеют помехозащитные коды, которые могут исправлять ошибки определенной кратности. Соотношение между максимальной кратностью исправляемой ошибки *q* и кодовым расстоянием *d* определяется по формуле (2.5):

*d* ≥2*q* + 1. (2.5)

В основу исправления ошибок положена следующая идея: определяется множество кодовых комбинаций, включающее все разрешенные и те запрещенные, которые получены при искажении ошибкой кратности не более *q*. Это множество разбивается на *m* подмножеств, где *m* – число исходных кодируемых символов. В каждое подмножество входят: одна разрешенная кодовая комбинация и ближайшие к ней запрещенные, которые отстоят от разрешенной на расстояние не больше *q*. Тогда при декодировании определяется, в какое подмножество входит принятая кодовая комбинация. Если она является разрешенной, то сразу декодируется; если она запрещенная, то исправляется на разрешенную, с которой находится в одном подмножестве, а затем декодируется.

**Пример 2.26**. Построить помехозащитный код, исправляющий ошибку кратности 1, для передачи двух символов: *a* и *b.*

Построим первичный код. Поскольку для кодирования двух символов достаточно одного двоичного разряда, первичный код может иметь следующий вид:

*a* 0,

*b* 1.

Поскольку по заданию *q* = 1, для исправления ошибки кратности 1 кодовое расстояние должно быть равно, по меньшей мере, 3.

Поскольку в первичном коде обеспечено расстояние между кодовыми комбинациями, равное 1, для выполнения условия *d* = 3 необходимо, чтобы проверочные разряды обеспечивали расстояние между кодовыми комбинациями, по меньшей мере, равным 2. Очевидно, для этого число проверочных разрядов должно быть не меньше 2. Тогда разрешенные кодовые комбинации могут иметь вид:

*исходный информационный проверочные результирующий*

*символ разряд разряды код*

*a* 0 00 000

*b* 1 11 111

Очевидно, кодовое расстояние равно 3, а построенные кодовые комбинации являются разрешенными.

Определим общее число всевозможных комбинаций, если число разрядов кода равно 3:

000 - разрешенная кодовая комбинация,

001

010

011

- запрещенные кодовые комбинации,

100

101

110

111 - разрешенная кодовая комбинация.

Определим подмножества кодовых комбинаций, которые отстояли бы от каждой разрешенной на минимальное расстояние, равное 1:

**для 000 для 111**

001 011

010 101

100 110

Пусть передается кодовая комбинация 000 (символ *a*) и на нее накладывается ошибка кратности 1. В табл. 2.21 показаны полученные кодовые комбинации и их декодирование:

Таблица 2.21.

Результат декодирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Передаваемая  кодовая  комбинация | Вектор  ошибки | Принимаемая  кодовая  комбинация | Результат  исправления | Результат  декодирования |
| 000 | 100 | 100 | 000 | *a* |
| 000 | 010 | 010 | 000 | *a* |
| 000 | 001 | 001 | 000 | *a* |

Таким образом, построенный код позволяет исправлять ошибки кратности 1.

**Пример 2.27**. Построить помехозащитный код, исправляющий ошибку кратности 1, для передачи символов: *a,* *b* и *c.*

Построим первичный код: *a* – 00;  *b –* 01; *c –* 10.

Для решения поставленной задачи необходимо обеспечить *d* = 3.

Воспользуемся схемой формирования кода Грея из рассмотренного ранее примера:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 00 | 01 | 11 |
| 000 | *a* |  |  |
| 011 |  | *b* |  |
| 101 |  |  | *c* |

Таким образом, получены коды: *a - 00000, b - 01101, c- 10111.*

Кодовое расстояние *d* = min{*dab, dac, dbc*} = min{3,4,3} = 3 обеспечивает исправление ошибки кратности *q* = 1.

Рассмотрим, как исправляются ошибки в данном случае. Все множество кодовых комбинаций пятиразрядного двоичного кода равно 25 = 32. Из них три кодовые комбинации – разрешенные, остальные – запрещенные. Разобьем кодовые комбинации на три подмножества, в каждое из которых будут входить: одна разрешенная и те запрещенные, которые отстоят от разрешенной на расстояние в 1. Имеем:

**для 00000 для 01101 для 10111**

00001 01100 10110

00010 01111 10101

00100 01001 10011

01000 00101 11111

10000 11101 00111

Очевидно, общее число кодовых комбинаций, включенных в построенные подмножества, равно 24. Оставшиеся 8 кодовых комбинаций являются следствием ошибки кратности больше 1 и в сформированные подмножества не включены – такие ошибки данный код исправить не может.

Проверим, как выполняется исправление ошибки кратности 1. Пусть передается кодовая комбинация 01101 (символ *b*) и на нее накладывается ошибка кратности 1. В табл. 2.22 показаны полученные кодовые комбинации и их декодирование:

Таблица 2.22.

Результат декодирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Передаваемая  кодовая  комбинация | Вектор  ошибки | Принимаемая  кодовая комбинация | Результат  исправления | Результат  декодирования |
| 01101 | 10000 | 11101 | 01101 | *b* |
| 01101 | 01000 | 00101 | 01101 | *b* |
| 01101 | 00100 | 01001 | 01101 | *b* |
| 01101 | 00010 | 01111 | 01101 | *b* |
| 01101 | 00001 | 01100 | 01101 | *b* |

Пусть на ту же кодовую комбинацию накладывается ошибка кратности 2 (табл. 2.23). Результирующие кодовые комбинации либо невозможно декодировать, либо декодирование неверно.

Таблица 2.23.

Результат декодирования

|  |  |  |  |
| --- | --- | --- | --- |
| Передаваемая  кодовая  комбинация | Вектор  ошибки | Принимаемая  кодовая  комбинация | Результат декодирования |
| 01101 | 10001 | 11100 | *Невозможно декодировать* |
| 01101 | 01001 | 00100 | *То же* |
| 01101 | 00101 | 01000 | *a* |
| 01101 | 00011 | 01110 | *Невозможно декодировать* |
| 01101 | 10010 | 11111 | *с* |
| 01101 | 01010 | 00111 | *То же* |
| 01101 | 00110 | 01011 | *Невозможно декодировать* |
| 01101 | 10100 | 11001 | *То же* |
| 01101 | 01100 | 00001 | *а* |
| 01101 | 11000 | 10101 | *с* |

В заключение отметим, что для обнаружения ошибки кратности *q1* и исправления ошибки кратности *q2* при *q1* ≥ *q2* минимальное кодовое расстояние должно удовлетворять следующему соотношению (2.6):

*d ≥ q1 + q2 + 1.* (2.6)

# 3. Измерение информации

В информатике измерению подвергается информация, представленная как дискретным, так и аналоговым сигналами. При этом различают следующие подходы:

1. **структурный** подход. Измеряет количество информации простым подсчетом информационных элементов, составляющих сообщение, или подсчетом числа сообщений. Данный метод применяется *только* для дискретных сигналов,
2. **статистический** подход. Учитывает вероятность появления сигналов: более информативным считается тот сигнал, который менее вероятен, т.е. менее всего ожидаем. Применяется для обоих видов сигналов,
3. **семантический** подход. Учитывает целесообразность и полезность информации. Применяется для обоих видов сигналов.

## 3.1. Структурный подход к измерению информации

В рамках структурного подхода выделяют три меры информации:

1. геометрическую;
2. комбинаторную;
3. аддитивную, или меру Хартли.

### 3.1.1. Геометрическая мера

Измеряет информацию подсчетом информационных элементов в дискретном сигнале. Единица измерения – информационный элемент. Мера может быть использована, например, для определения информационной емкости памяти компьютера. В этом случае в качестве информационного элемента выступает минимальная единица хранения – бит. Список самых распространенных более крупных единиц и соотношение между ними приведено ниже:

8 бит = 1 байт (сокращенно б или Б),

1024 Б = 1 килобайт (сокращенно Кб или К),

1024 К = 1 мегабайт (сокращенно Мб или М),

1024 М = 1 гигабайт (сокращенно Гб или Г).

Тогда, например, объем винчестера – 3 гигабайта; объем основной памяти компьютера – 32 мегабайта и т.д.

**Пример 3.1**. Пусть сообщение 5555 6666 888888 закодировано одним из специальных методов эффективного кодирования - кодированием повторений - и имеет вид: 5(4) 6(4) 8(6) .

Измерить информацию I в исходном - Iисх и закодированном – Iкод сообщениях геометрической мерой и оценить эффективность кодирования.

В качестве информационного элемента зададимся символом сообщения. Тогда:

Iисх = ***l***исх = 14 символов;

Iкод = ***l***код = 12 символов,

где ***l***исх, ***l***код – число символов (длина) соответствующего сообщения.

Эффект кодирования определяется как разница между IисхиIкоди составляет 2 символа.

Очевидно, геометрическая мера не учитывает, какими символами заполнено сообщение. Так, одинаковыми по количеству информации, измеренной геометрической мерой, являются, например, сообщения «компьютер» и «программа», также 346 и 101, тест и test.

### 3.1.2. Комбинаторная мера

Оценивает возможность представления информации при помощи различных комбинаций информационных элементов в заданном объеме. Использует типы комбинаций элементов и соответствующие математические соотношения, которые приводятся в одном из разделов дискретной математики – комбинаторике (необходимые положения приведены в приложении 1).

Комбинаторная мера может использоваться для оценки информационных возможностей некоторого автомата, который способен генерировать дискретные сигналы (сообщения) в соответствии с определенным правилом комбинаторики (см. приложение 1).

Пусть, например, есть автомат, формирующий двузначные десятичные целые положительные числа (исходный алфавит – десятичные цифры 0,1,2,3,4,5,6,7,8,9). В соответствии с положениями комбинаторики данный автомат генерирует числа со следующими характеристиками:

1. числа формируются по правилу размещения, поскольку различаются числа, например, 34 и 43,
2. числа формируются из 10 элементов, поскольку используются 10 цифр,
3. в число включаются по 2 символа, поскольку по условию задачи формируются двузначные числа,
4. возможны повторения цифр в числе - очевидно, возможны числа, состоящие из одинаковых цифр, например, 33.

Тогда можно оценить, сколько различных сообщений (двузначных чисел) может сформировать автомат, иначе говоря, можно оценить информационную емкость данного устройства: .

Комбинаторная мера используется для определения возможностей кодирующих систем, которые широко используются в информационной технике.

**Пример 3.2**. Определить емкость ASCII-кода, представленного в двоичной или шестнадцатеричной системе счисления.

ASCII-код позволяет формировать сообщения по правилу размещений с повторениями:

* для двоичного представления – из символов алфавита {0, 1}, сообщение длиной (объемом) 8 символов;
* для шестнадцатеричного представления – из символов алфавита {0,1,2,….,А,В, С,D,E,F}, сообщение длиной (объемом) 2 символа.

Тогда: *,*

где *I*двоичное, *I*шестнадцатеричное – количества информации, соответственно, для двоичного и шестнадцатеричного представления ASCII-кода.

Таким образом, емкость ASCII-кода для двоичного и шестнадцатеричного представления одинакова и равна 256.

Следует отметить, что все коды постоянной длины формируются по правилам комбинаторики или их комбинациям.

В случае когда сообщения формируются как размещения с повторениями из элементов алфавита мощности *h* и известно количество сообщений *М*, можно определить требуемый объем сообщения (т.е. его длину *l*) для того, чтобы в этом объеме представить все сообщения: *,* где скобки [] означают округление в большую сторону.

Например, есть 4 сообщения – *a, b, c, d.* Выполняется двоичное кодирование этих сообщений кодом постоянной длины. Для этого требуются два двоичных разряда. В самом деле: *.*

Очевидно, комбинаторная мера является развитием геометрической меры, так как помимо длины сообщения учитывает объем исходного алфавита и правила, по которым из его символов строятся сообщения.

Особенностью комбинаторной меры является то, что ею измеряется информация не конкретного сообщения, а всего множества сообщений, которые могут быть получены.

Единицей измерения информации в комбинаторной мере является число комбинаций информационных элементов.

### 3.1.3. Аддитивная мера

Эта мера предложена в 1928 году американским ученым Хартли, поэтому имеет второе название – мера Хартли. Он впервые ввел специальное обозначение для количества информации – *I* и предложил следующую зависимость для расчета количества информации: , где *I* – количество информации, содержащейся в сообщении; *l* – длина сообщения; *h* – мощность исходного алфавита.

При исходном алфавите {0,1}; *l*=1; *h*=2 имеем *I*=1\*log22=1. Данная формула даёт аналитическое определение бита (BIT - **BI**nary digi**T**) по Хартли: это количество информации, которое содержится в двоичной цифре.

Единицей измерения информации в аддитивной мере является бит.

**Пример 3.3**. Рассчитать количество информации, которое содержится в шестнадцатеричном и двоичном представлении ASCII-кода для числа 1.

В соответствии с таблицей ASCII-кодов имеем:

* шестнадцатеричное представление числа 1 – 31,*I*=2\*log216=8 бит,
* двоичное представление числа 1 – 00110001, *I*=8\*log22=8 бит.

Таким образом, разные представления ASCII-кода для одного символа содержат одинаковое количество информации, измеренной аддитивной мерой.

В то же время, если те же два сообщения – 31 и 00110001 – измерить геометрической мерой, объем информации в них различен: 2 и 8 символов соответственно.

**Пример 3.4**. Рассчитать количество информации в сообщениях: тест и test.

Поскольку мощности кириллицы и латиницы равны, соответственно, 33 и 26, имеем: 33=20,178 бит, =18,8 бит. Очевидно, будучи измерены геометрической мерой, данные сообщения содержат одинаковое количество информации.

## 3.2. Статистический подход к измерению информации

В 30-х годах ХХ века американский ученый Клод Шеннон предложил связать количество информации, которое несет в себе некоторый сигнал, с вероятностью получения этого сигнала.

**Вероятность** *p* – количественная априорная (т.е. известная до проведения опыта) характеристика одного из исходов некоторого опыта. Измеряется в пределах от 0 до 1. Если заранее известны все исходы опыта, сумма их вероятностей равна 1, а сами исходы составляют **полную группу событий**. Если все исходы могут свершиться с одинаковой вероятностью, они называются **равновероятными**.

Например, пусть опыт состоит в сдаче студентом экзамена по информатике. Очевидно, у этого опыта всего 4 исхода (по количеству возможных оценок, которые студент может получить на экзамене). Тогда эти исходы составляют полную группу событий, т.е. сумма их вероятностей равна 1. Если студент учился хорошо в течение семестра, значения вероятностей всех исходов могут быть, например, такими: *p*(5) = 0,5; *p*(4) = 0,3; *p*(3) = 0,1; *p*(2) = 0,1. Здесь запись *p*(j) означает вероятность получения оценки j (j = {2,3,4,5}).

Если студент учился плохо, можно заранее оценить возможные исходы сдачи экзамена, т.е. задать вероятности исходов, например, следующим образом: *p*(5) = 0,1; *p*(4) = 0,2; *p*(3) = 0,4; *p*(2) = 0,3.

В обоих случаях сумма вероятностей равна 1.

Пусть можно получить *n* сигналов по результатам некоторого опыта (т.е. у опыта есть *n* исходов), причем известны вероятности получения каждого сигнала (исхода) - *p*i. Тогда, в соответствии с идеей Шеннона, количество информации *I* в i-м сигнале определяется по формуле (3.1):

, (3.1)

где *p*i – вероятность i-го сигнала.

**Пример 3.5**. Определить количество информации, содержащейся в сигнале о результате сдачи экзамена для студента-хорошиста, для которого *p*(5)=0,5; *p*(4)=0,3; *p*(3)=0,1; *p*(2)=0,1.

Пусть *I*(j) – количество информации в сигнале о получении оценки j. Тогда имеем:

,

,

3,32,

3,32.

**Пример 3.6**. Определить количество информации, содержащейся в сигнале о результате сдачи экзамена для плохо успевающего студента, для которого *p*(5)=0,1; *p*(4)=0,2; *p*(3)=0,4; *p*(2)=0,3:

,

,

,

.

Таким образом, количество получаемой с сигналом информации тем больше, чем неожиданнее данный сигнал.

Этот тезис использован при эффективном кодировании кодами переменной длины (т.е. имеющими разную геометрическую меру): исходные символы, имеющие большую частоту (или вероятность), имеют код меньшей длины, т.е. несут меньше информации в геометрической мере, и наоборот.

Соотношение (3.1) позволяет определять также размер двоичного эффективного кода, требуемого для представления того или иного сообщения, имеющего определенную вероятность появления. Поскольку размер кодовой комбинации – целое число, модифицируем формулу (3.1): , где *l* – число разрядов кода, скобки [] означают округление в сторону ближайшего большего целого числа.

**Пример 3.7**. Есть 4 сообщения: *a, b, c, d* с вероятностями, соответственно, р(*a*)=0,5;р(*b*)=0,25;р(*c*)=0,125;р(*d*)=0,125. Определить число двоичных разрядов, требуемых для кодирования каждого их четырех сообщений:

,

2,

,

3.

Помимо информационной оценки одного сигнала, Шеннон предложил количественную информационную оценку всех сигналов, которые можно получить по результатам проведения некоторого опыта. Так, среднее количество информации *I*ср, получаемой со всеми *n* сигналами, определяется по формуле (3.2):

(3.2)



где *pi –* вероятность i-го сигнала.

**Пример 3.8**. Определить среднее количество информации, получаемое студентом-хорошистом, по всем результатам сдачи экзамена:

*I*ср=-(0,5\*log20,5+0,3\*log20,3+0,1\*log20,1+0,1\*log20,1)=1,67.

**Пример 3.9**. Определить среднее количество информации, получаемое плохо успевающим студентом, по всем результатам сдачи экзамена:

*I*ср=- (0,1\*log20,1+0,2\*log20,2+0,4\*log20,4+0,3\*log20,3)=1,73.

Большее количество информации, получаемое во втором случае, объясняется большей непредсказуемостью результатов: в самом деле, у отличника два исхода равновероятны.

Пусть у опыта два равновероятных исхода, составляющих полную группу событий, т.е. *p*1=*p*2=0,5. Тогда имеем: *I*ср=-(0,5\*log20,5+0,5\*log20,5)=1. Данная формула есть аналитическое определение бита по Шеннону: это среднее количество информации, которое содержится в двух равновероятных исходах некоторого опыта, составляющих полную группу событий.

Единица измерения информации при статистическом подходе – бит.

На практике часто вместо вероятностей используются частоты исходов. Это возможно, если опыты проводились ранее и существует определенная статистика их исходов. Так, строго говоря, в построении эффективных кодов участвуют не частоты символов, а их вероятности.

## 3.3. Семантический подход к измерению информации

В рамках этого подхода рассмотрим такие меры, как целесообразность, полезность (учитывают прагматику информации) и истинность информации (учитывает семантику информации).

### 3.3.1. Целесообразность информации

Количество *I* получаемой информации с позиций ее целесообразности определяется по формуле (3.3):

(3.3)



где *p*1, *p*2 – вероятности достижения цели после и до получения информации соответственно.

**Пример 3.10**. Пусть вероятность сдачи экзамена по информатике до получения некоторого сообщения (подсказки от соседа) оценивается студентом как 0,2, т.е. *p*2=0,2. После того, как ему удалось получить подсказку, вероятность сдачи увеличилась: *p*1=0,8. Определить количество информации, содержащейся в подсказке, с точки зрения ее целесообразности:

*I*=log2(0,8/0,2)=log24=2.

**Пример 3.11.** Пусть положение студента до получения подсказки оценивается аналогично примеру 3.10. После получения подсказки, вопреки ожиданиям, вероятность сдачи еще уменьшилась, поскольку подсказка содержала неверную информацию: *p*1=0,1. Определить количество информации, содержащейся в подсказке, с точки зрения ее целесообразности:

*I*=log2(0,1/0,2)=log20,5=-1.

Таким образом, полученная информация является дезинформацией, поскольку имеет отрицательный знак при измерении.

### 3.3.2. Полезность информации

Количество усваиваемой потребителем информации *I*усв тесно связано с теми знаниями, которые имеет потребитель к моменту получения информации – с **тезаурусом** (ТЗ) потребителя. Этим определяется полезность информации. В самом деле, для усвоения тех знаний, которые получаются в университете, требуется среднее образование - иначе студент ничего не поймет. С другой стороны, любая учебная дисциплина ориентируется на знания, которые учащийся должен приобрести в предыдущих курсах. Этим объясняется последовательность учебных дисциплин по годам обучения.

Зависимость усваиваемой потребителем информации от его тезауруса выражается графически кривой, показанной на рис. 3.1.

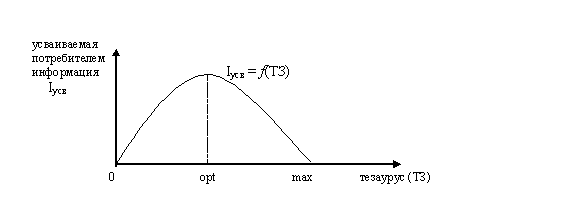


Рис. 3.1. Иллюстрация полезности информации

Как видно из графика, при тезаурусе, равном нулю и максимальному значению в точке max, информация не усваивается: в первом случае потребителю непонятна принимаемая информация, во втором – она ему уже известна. Максимально усваивается информация (т.е. она наиболее полезна) в точке opt, когда потребитель обладает достаточным (но не максимально возможным) тезаурусом для понимания получаемой информации.

При значении тезауруса i-го потребителя ТЗi количество усваиваемой им информации определяется как *I*усв = *f*(ТЗi). Сам тезаурус ТЗi может быть практически определен как результат интеллектуального тестирования, которое проводится, например, в некоторых западных странах. При таком тестировании человеку выставляется некоторый балл, который и может расцениваться как его тезаурус ТЗi.

### 3.3.3. Истинность информации

Пусть *r(mess)* – функция, оценивающая истинность информации, получаемой с сигналом *mess,* как соответствие её реальному положению вещей: 0 ≤ *r(mess)* ≤ 1, причем при *r(mess)* = 1 информация истинна, а при *r(mess)* = 0 информация ложна.

Например,

r(«данное пособие посвящено информатике») = 1,

r(«данное пособие имеет объем 5 страниц») = 0,

r(«компьютер с большим быстродействием») = 0,3, поскольку понятие большого быстродействия компьютера относительно.

Оценить истинность сложного сообщения можно, разбив его на простые. Например, сообщение *mess*: «данный конспект посвящен информатике и имеет объем 5 страниц» можно представить как два простых сообщения *mess*1 и *mess*2: «данный конспект посвящен информатике» и «данный конспект имеет объем 5 страниц».

Тогда можно предложить рассчитывать истинность информации в сигнале, фиксирующем совокупности некоторых фактов, как среднее арифметическое значение истинностей простых сигналов, его составляющих (что называют «истинно лишь наполовину»). В таком случае имеем:

*r*(*mess*) = ½ (*r*(*mess*1) + *r*(*mess*2)) = ½ (1 + 0) = 0,5.

Количество информации *I*, получаемое с сигналом *mess*, можно рассчитать как *I* = 2\**r*(*mess*) –1, где *r(mess*) – истинность сигнала *mess*.

## 3.4. Качество информации

Качество информации является одним из важнейших параметров для потребителя информации. Оно определяется следующими характеристиками:

1. **репрезентативность** – правильность отбора информации в целях адекватного отражения источника информации. Например, в целях большей репрезентативности данных о себе студенты при назначении стипендии стремятся представить в деканат как можно больше справок, подтверждающих их активную общественную и научную деятельность в вузе. В то же время нет необходимости, например, предоставлять медицинские справки о своих заболеваниях – эта информация не используется при назначении стипендии;
2. **содержательность** – семантическая емкость информации. Рассчитывается как отношение количества семантической информации к ее количеству в геометрической мере. Это характеристика сообщения, про которое говорят, что «мыслям в нем тесно, а словам просторно». В целях увеличения содержательности сигнала, например, можно для характеристики успеваемости студента использовать не полный перечень его оценок в сессию, а средний балл;
3. **достаточность** (полнота) – минимальный, но достаточный состав данных для достижения целей, которые преследует потребитель информации. Эта характеристика похожа на репрезентативность, однако разница состоит в том, что в данном случае учитывается минимальный состав информации, который не мешает принятию решения. Например, при назначении повышенной стипендии часть справок может не учитываться, поскольку уже предоставленных студентом документов достаточно для принятия решения по нему;
4. **доступность** – простота или возможность выполнения процедур получения и преобразования информации. Эта характеристика применима не ко всей информации, а лишь к той, которая не является закрытой. Для обеспечения доступности бумажных документов используются различные средства оргтехники для их хранения, а для облегчения их обработки используются средства вычислительной техники;
5. **актуальность** – зависит от динамики изменения характеристик информации и определяется сохранением ценности информации для пользователя в момент ее использования. Очевидно, что касается информации, которая используется при зачислении в университет, она актуальна, так как само обучение в среднем учебном заведении уже закончилось и его результаты изменены быть не могут, а значит, остаются актуальными;
6. **своевременность** – поступление информации не позже заранее назначенного срока. Этот параметр также очевиден недавним абитуриентам: опоздание с представлением позитивной информации о себе при поступлении может быть чревато незачислением;
7. **точность** – степень близости информации к реальному состоянию источника информации. Например, неточной информацией является медицинская справка, в которой отсутствуют данные о перенесенных абитуриентом заболеваниях;
8. **достоверность** – свойство информации отражать источник информации с необходимой точностью. Эта характеристика вторична относительно точности. В предыдущем примере получаемая информация недостоверна;
9. **устойчивость** – способность информации реагировать на изменения исходных данных без нарушения необходимой точности.

# Технические средства информатики

## 4.1. Структура компьютера и принципы его функционирования

В 1946 – 1948 гг. в Принстонском университете (США) коллективом исследователей под руководством Джона фон Неймана был разработан проект ЭВМ, идеи которого используются и по сей день. Этот проект получил название машины фон Неймана или Принстонской машины. В его состав входила схема (рассматривается далее) и принципы функционирования вычислительной машины:

1) **принцип программного управления**: работа ЭВМ регламентируется программой, что позволяет, вводя разные программы, решать разные задачи. Команды, из которых состоит программа, интерпретируются специально введенным в схему устройством – устройством управления. Структура отдельной команды имеет вид: <код операции> <операнды>, где <код операции> определяет, какая операция должна выполняться, <операнды> - список (возможно, одноэлементный) тех констант, адресов или имен переменных, над которыми выполняется данная операция.

В зависимости от числа операндов различают одно-, двух- и трехадресные машинные команды. Каждая команда имеет определенный объем, измеряемый байтами.

Этот принцип был самым прогрессивным среди включенных в проект, поскольку обеспечивал универсальность ЭВМ. В соответствии с принципом программного управления любая ЭВМ – это совокупность аппаратной (технической) и программной частей;

2) **принцип условного перехода**: команды из программы не всегда выполняются одна за другой. Возможно присутствие в программе команд условного перехода, которые меняют последовательное выполнение команд в зависимости от значений данных;

3) **принцип размещения программы в памяти**: программа, требуемая для работы ЭВМ, предварительно размещается в памяти компьютера, а не вводится команда за командой;

4) **принцип иерархии памяти**: память ЭВМ не однородна. Для часто используемых данных выделяется память меньшего объема, но большего быстродействия; для редко используемых данных выделяется память большего объема, но меньшего быстродействия;

5) **принцип двоичной системы счисления**: для внутреннего представления данных и программ в памяти ЭВМ применяется двоичная система счисления, которую можно проще реализовать технически.

Структура Принстонской машины представлена на рис. 4.1. Рассмотрим назначение отдельных элементов этой схемы и их взаимосвязь в процессе функционирования ЭВМ.

**Устройство ввода (УВв)** преобразуетинформацию в форму, понятную компьютеру, после чего он может ее обрабатывать и запоминать. Через УВв в **память (П)** первоначальновводится программа – набор команд, предписывающих ЭВМ выполнять требуемые действия (на схеме связь 1), а затем и данные для обработки программой.

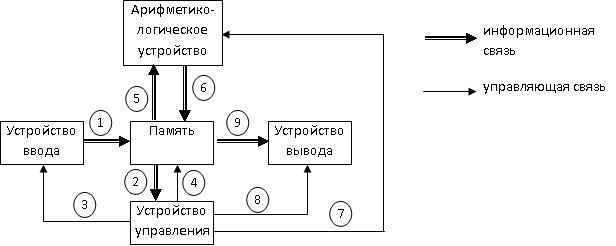


Рис. 4.1. Схема Принстонской машины

После размещения программы в памяти **устройство управления (УУ)** выбирает последовательно команду за командой из памяти (связь 2) и интерпретирует ее по следующим правилам:

* если выбранная команда является командой ввода данных, УУ посылает управляющий сигнал (связь 3) в УВв для начала ввода данных. Данные также вводятся по связи 1 и размещаются в памяти П;
* если выбранная команда связана с выполнением арифметических или логических операций, то в память П из УУ посылается сигнал (связь 4) на выборку указанных в команде данных с последующей их пересылкой в **арифметико-логическое устройство (АЛУ)** (связь 5), а в само АЛУ передается сигнал с кодом нужной операции (связь 7). АЛУ выполняет арифметические и логические действия над переданными операндами. После выполнения требуемых действий АЛУ возвращает результат в память П (связь 6);
* если выбранная команда является командой вывода, УУ генерирует управляющий сигнал **устройству вывода (УВыв)** (связь 8) на начало операции по выводу данных. Сами данные выбираются из памяти П по связи 9.

**УВыв** переводит информацию из внутреннего представления компьютера в образы, понятные человеку**,** т.е. УВыввыводит информацию из ЭВМ.

В соответствии с принципом иерархии памяти блок Память на рис. 4.1 делится на два блока – **внешнюю** и **внутреннюю память.** Внешняя память традиционно отводится для долговременного хранения данных и программ, а сама оперативная обработка данных в соответствии с программой, как это было рассмотрено выше, выполняется во внутренней памяти.

В современных компьютерах блоки УУ и АЛУ объединены в блок, называемый **процессором**. В состав процессора, кроме указанных блоков, входят также несколько регистров – специальных небольших областей памяти, куда процессор помещает промежуточные результаты и некоторую другую информацию, необходимую ему в ближайшие такты работы.

## 4.2. Виды современных компьютеров

По **назначению** выделяют следующие виды компьютеров:

а) **универсальные -** предназначены для решения различных задач, типы которые не оговариваются. Эти ЭВМ характеризуются:

* разнообразием форм обрабатываемых данных (числовых, символьных и т.д.) при большом диапазоне их изменения и высокой точности представления;
* большой емкостью внутренней памяти;
* развитой системой организации ввода-вывода информации, обеспечивающей подключение разнообразных устройств ввода-вывода.

б) **проблемно-ориентированные -** служат для решения более узкого круга задач, связанных, как правило, с управлением технологическими объектами, регистрацией, накоплением и обработкой небольших объемов данных, выполнением расчетов по несложным правилам. Они обладают ограниченным набором аппаратных и программных средств.

в) **специализированные -** применяются для решения очень узкого круга задач, что позволяет специализировать их структуру, снизить стоимость и сложность при сохранении высокой производительности и надежности. К этому классу ЭВМ относятся компьютеры, управляющие работой устройств ввода-вывода и внешней памятью в современных компьютерах. Такие устройства называются адаптерами, или контроллерами.

По **размерам и функциональным возможностям** различают виды компьютеров: суперЭВМ, большие, малые и микроЭВМ.

**СуперЭВМ** являются мощными многопроцессорными компьютерами с огромным быстродействием. Многопроцессорность позволяет распараллеливать решение задач и увеличивает объемы памяти, что значительно убыстряет процесс решения. Они часто используются для решения экспериментальных задач, например, для проведения шахматных турниров с человеком.

**Большие** ЭВМ (их называют мэйнфреймами от англ. mainframe) характеризуются многопользовательским режимом (до 1000 пользователей одновременно могут решать свои задачи). Основное направление – решение научно-технических задач, работа с большими объемами данных, управление компьютерными сетями и их ресурсами.

**Малые** ЭВМ используются как управляющие компьютеры для контроля над технологическими процессами. Применяются также для вычислений в системах: многопользовательских, автоматизации проектирования, моделирования несложных объектов, искусственного интеллекта.

**Микро-ЭВМ** делятся по признакам:

* *по назначению* - могут быть универсальными и специализированными,
* *по числу пользователей*, одновременно работающих за компьютером, – много- и однопользовательские.

*Специализированные многопользовательские* микроЭВМ (серверы - от англ. server) являются мощными компьютерами, используемыми в компьютерных сетях для обработки запросов всех компьютеров сети.

*Специализированные однопользовательские* микроЭВМ (рабочие станции – от англ. workstation, англ.) эксплуатируются в компьютерных сетях для выполнения прикладных задач.

*Универсальные многопользовательские* микроЭВМ являются мощными компьютерами, оборудованными несколькими терминалами.

*Универсальные однопользовательские* микроЭВМ общедоступны. К их числу относятся персональные компьютеры – ПК. Наиболее популярным представителем ПК в нашей стране является компьютер класса IBM PC (**I**nternational **B**usiness **M**achines – **P**ersonal **C**omputer). По конструктивным особенностям ПК делятся на стационарные (настольные – тип DeskTop) и портативные. В свою очередь, портативные компьютеры включают следующие виды:

* переносные (portable) - имеют небольшую массу и габариты и поддаются транспортировке одним человеком;
* наколенные (laptop) - выполнены в виде дипломата;
* блокнотные (notebook) – имеют габариты большого блокнота;
* карманные (pocket) - помещаются в карман.

## 4.3. Структурные элементы компьютера

Рассмотренная выше структура компьютера является его логической структурой, которая определяет функциональные основные элементы компьютера и их взаимосвязь. Рассмотрим также и физическую структуру компьютера как универсальной технической системы, состав устройств (конфигурацию) которой можно менять по мере необходимости. В эту техническую систему входят *системный блок* и *периферийные устройства* (периферия).

***Системный блок*** –основная часть компьютера. Состоит из корпуса, в котором располагаются основные компоненты компьютера (с ним соединены кабелями периферийные устройства). Основные элементы:

- *материнская плата* – основа системного блока – самостоятельный элемент, который управляет внутренними связями и с помощью системы прерываний взаимодействует с внешними устройствами. На ней размещаются:

* процессор (микропроцессор) – основная микросхема, выполняющая вычисления и управляющая работой всех компонентов компьютера;
* микропроцессорный комплект – набор микросхем, управляющих работой внутренних устройств компьютера и определяющих функциональные основные возможности материнской платы;
* системная шина – осуществляет информационную связь между устройствами компьютера. Это кабель, состоящий из множества проводников. По одной группе проводников **(***шине данных*) передаётся обрабатываемая информация, по другой (*шине адреса*) передаются адреса памяти или внешних устройств, к которым обращается процессор. Третья часть магистрали (*шина управления*) передает управляющие сигналы (сигнал готовности устройства к работе, сигнал к началу работы устройства и т.п.). Системная шина характеризуется тактовой частотой и разрядностью.Количество одновременно передаваемых по ней бит называется *разрядностью шины,* измеряется в битах. *Тактовая частота* характеризует число элементарных операций по передаче данных в 1 секунду, измеряется в мегагерцах. Всякая информация, передаваемая от процессора к другим устройствам по шине данных, сопровождаетсяадресом, передаваемым по адресной шине. Это может быть адрес ячейки памяти или адрес периферийного устройства. Необходимо, чтобы разрядность шины позволила передать адрес ячейки памяти. Таким образом, разрядность шины ограничивает объем оперативной памяти ЭВМ, он не может быть больше чем 2n, где n – разрядность шины. Важно, чтобы производительности всех подсоединённых к шине устройств были согласованы. Неразумно иметь быстрый процессор и медленную память или быстрый процессор и память, но медленный винчестер;
* оперативная память, или оперативное запоминающее устройство (ОЗУ), - набор микросхем, предназначенных для временного хранения данных, когда компьютер включен (описана далее);
* постоянное запоминающее устройство (ПЗУ) – микросхема, предназначенная для длительного хранения данных, в том числе и когда компьютер выключен (описано далее);
* память CMOS (Complementary Metal-Oxide Semiconductor) – хранит информацию о составе оборудования компьютера;
* слоты – разъемы для подключения дополнительных устройств - контроллеров (другие названия контроллера - адаптер, плата, карта), через которые, в свою очередь, аппаратно подключаются периферийные устройства к системной шине;
* счетчик времени;
* системы индикации и защиты;

- *жесткий диск* для долговременного хранения больших объемов данных и программ;

- *дисковод компакт-дисков*;

- *видеоадаптер* (видеокарта);

- *звуковая карта*;

- *блок питания*, преобразующий электропитание сети в постоянный ток низкого напряжения, подаваемый на электронные схемы компьютера;

- *вентиляторы* для охлаждения греющихся элементов.

***Периферийные (внешние) устройства*** включают:

* *устройства ввода данных* - клавиатура, манипуляторы, сканеры, цифровые фотокамеры;
* *устройства вывода данных* – мониторы, принтеры, графопостроители (плоттеры);
* *устройства хранения данных* - диски, стримеры (накопители на магнитных лентах), флэш[[1]](#footnote-2)-карты,
* *устройства обмена данными* - модемы, адаптеры, каналы связи (проводные, кабельные, оптоволоконные, радиочастотные). В зависимости от типа канала связи используют радиомодемы, кабельные модемы и пр.

Программное управление работой периферийного устройства производится через программу - драйвер, которая является компонентой операционной системы. Так как существует огромное количество разнообразных устройств, которые могут быть установлены в компьютер, то обычно к каждому устройству поставляется драйвер, взаимодействующий непосредственно с этим устройством. Связь компьютера с периферийными устройствами осуществляется через порты – специальные разъёмы на задней панели системного блока. Различают последовательные и параллельные порты. Последовательные (COM – порты) служат для подключения манипуляторов курсора, модема и передают небольшие объёмы информации на большие расстояния. Параллельные (LPT - порты) служат для подключения принтеров, сканеров и передают большие объёмы информации на небольшие расстояния. В последнее время широкое распространение получили последовательные универсальные порты (USB), к которым можно подключать различные устройства.

Для компьютера существует понятие *базовой* конфигурации, которая включает системный блок, монитор, клавиатуру, мышь.

Рассмотрим структурные элементы компьютера более подробно.

### 4.3.1. Память

Памятью компьютера называется совокупность устройств для хранения программ, вводимой информации, промежуточных результатов и выходных данных.

Классификация памяти выполняется по следующим основаниям:

1. *по способу достижения* нужного участка памяти:

* прямого (произвольного) доступа – нужный участок достигается непосредственно по его адресу. Это наиболее быстрый вид доступа,
* последовательного доступа – для достижения нужного участка сначала просматриваются предыдущие участки с меньшими адресами;

1. *по способу хранения* информации (по носителю):

* электронная, носитель в виде микросхем,
* магнитная, носитель – магнитный материал,
* оптическая, носитель хранит информацию с помощью неоднородностей, от которых по-разному отражается луч света;

1. *по срокам действия*:

* оперативная (энергозависимая) – хранит информацию только на время работы с ней (теряет информацию при выключении компьютера),
* постоянная (энергонезависимая) – хранит данные постоянно (не теряет информацию при выключении компьютера);

1. *по отношению к компьютеру*:

* внутренняя - предназначена для хранения относительно ***небольших*** объемов информации при ее обработке процессором.
* внешняя - предназначена для ***длительного*** хранения ***больших*** объемов информации независимо от того, включен или выключен компьютер. Различаются эти виды памяти также по следующему критерию: если при удалении элемента памяти компьютер продолжает оставаться работоспособным, это – внешняя память; если он перестает работать – это внутренняя память.

Характеристика видов памяти представлена в табл. 4.1.

Таблица 4.1.

Характеристика видов памяти компьютера

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название памяти | Доступ | Носитель | Сроки  действия | По отношению  к компьютеру |
| Магнитный диск | прямой | магнитный | постоянная | внешняя |
| Стример | последовательный | магнитный | постоянная | внешняя |
| Оптический диск | прямой | оптический | постоянная | внешняя |
| Флэш-карта | прямой | электронный | постоянная | внешняя |
| Озу | прямой | электронный | оперативная | внутренняя |
| Кэш[[2]](#footnote-3) | прямой | электронный | оперативная | внутренняя |
| ПЗУ | прямой | электронный | постоянная | внутренняя |
| CMOS | прямой | электронный | постоянная | внутренняя |

#### 4.3.1.1. Внутренняя память

Как отмечалось, внутренняя память компьютера предназначена для хранения оперативно обрабатываемых данных. Она является более быстрой, чем внешняя память, что соответствует принципу иерархии памяти, выдвинутому в проекте Принстонской машины. Следуя этому принципу, можно выделить уровни иерархии и во внутренней памяти.

Как видно из табл. 4.1, внутренняя память включает следующие компоненты: ПЗУ, CMOS, ОЗУ, кэш.

Содержимое **ПЗУ** устанавливается на заводе-изготовителе и в дальнейшем не меняется (второе название этой памяти – ROM – Read Only Memory). Эта память составлена из микросхем, как правило, небольшого объема. В ней хранятся:

* программы, обеспечивающие минимальный базовый набор функций управления устройствами компьютера – этот комплект программ образует базовую систему ввода-вывода (BIOS[[3]](#footnote-4)), предназначенную для проверки состава и работоспособности компьютера и обеспечения взаимодействия с клавиатурой, монитором, винчестером, дисководами;
* программы начала загрузки операционной системы. При включении компьютера первоначально управление передается программе из ПЗУ, которая тестирует компоненты компьютера и запускает программу - загрузчик операционной системы;
* программа настройки конфигурации компьютера – Setup. Позволяет установить характеристики: типы видеоконтроллера, жестких дисков, режимы работы с RAM, запрос пароля при загрузке и т.п.

Микросхема **CMOS** постоянно подпитывается от небольшой батарейки, расположенной на материнской плате, от ПЗУ отличается тем, что данные в нее можно заносить и изменять самостоятельно, в соответствии с тем, какое оборудование входит в состав системы (таким образом программы, записанные в ПЗУ, считывают данные о составе оборудования компьютера из микросхемы CMOS).

В **ОЗУ** в двоичном виде запоминается обрабатываемая информация, программа ее обработки, промежуточные данные и результаты работы. ОЗУ обеспечивает режимы записи, считывания и хранения информации, причём в любой момент времени возможен доступ к любой произвольно выбранной ячейке памяти, что отражено в англоязычном названии ОЗУ – RAM (Random Access Memory – память с произвольным доступом). Часть ОЗУ занимает **видеопамять**, которая отводится для хранения изображений, получаемых на экране монитора. Чем больше видеопамять, тем более сложные и качественные картинки может выводить компьютер.

**Кэш** является самой быстрой и меньшей по объему из внутренней памяти. Она служит буфером между памятью RAM и процессором и позволяет увеличить скорость выполнения операций, так как является сверхбыстродействующей. В нее помещаются данные, которые процессор получил и будет использовать в ближайшие такты своей работы. Эта память хранит копии наиболее часто используемых участков RAM. При обращении процессора к памяти сначала ищутся данные в кэш-памяти, а затем, при необходимости, в RAM. Существует два вида кэш-памяти: внутренняя, размещаемая внутри процессора, и внешняя, размещаемая на материнской плате.

Память дискретна. Элементарной (минимальной) единицей хранения является бит. Он может содержать 02 или 12. Однако компьютер при работе с памятью для размещения или выборки данных из нее оперирует не битами, а байтами и более крупными единицами – словами и двойными словами. Для обращения к элементам памяти они снабжаются адресами, начиная с нуля. Максимальный адрес ОЗУ – MAX - определяется функциональными возможностями того или иного компьютера.

Структура внутренней памяти для компьютеров класса IBC PC с указанием начальных адресов отдельных областей представлена в табл. 4.2.

Таблица 4.2. Структура внутренней памяти

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Непосредственно адресуемая память операционной системы | | | | Расширенная память XMA[[4]](#footnote-5) | |
| стандартная память СМА[[5]](#footnote-6) (640К) | | верхняя память UMA[[6]](#footnote-7) (384К) | | Высокая память HMA[[7]](#footnote-8) (64К) | Остальная память |
| ОЗУ | | | ПЗУ |
| Область служебных программ и данных операционной системы (64К) | Область программ и данных пользователя (576К) | Область видеопамяти и служебных программ (256К) | Область программы начальной загрузки операционной системы и других программ BIOS (128К) |
| 0 64К 640К 896К 1024К 1088К МАХ | | | | | |

В силу особенностей операционной системы непосредственно адресуются только первые 1024К памяти, доступ к остальным адресам осуществляется драйверами. Их подключение выполняется в файле конфигурации config.sys в предложении device.

#### 4.3.1.2. Внешняя память

Внешняя память представлена в основном магнитными и оптическими носителями.

**Магнитные носители** делятся на магнитные ленты (стримеры), которые используются для хранения архивов и нашли неширокое применение, и магнитные диски. **Оптические накопители** включают следующие виды: CD, DVD, магнитооптические.

##### 4.3.1.2.1. Магнитные диски

Магнитные диски являются элементами устройств чтения-записи информации – дисководов. Сам магнитный диск – это алюминиевые или керамические круги (их также называют дисками) с магниточувствительным покрытием. Все круги в центре посажены на один стержень. Во время работы диск раскручивается. Схема дисковода (для простоты показан только один круг) представлена на рис. 4.2.

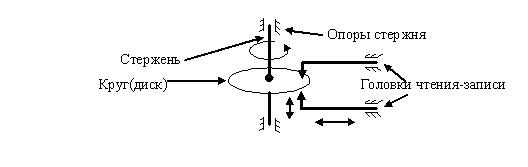


Рис. 4.2. Схема дисковода

Головки чтения-записи могут синхронно перемещаться в горизонтальном и вертикальном направлении (это показано стрелками), что позволяет им приблизиться к любой точке поверхности диска, которая рассматривается как отдельный бит внешней памяти.

Так же как и основная память, поверхность диска имеет структуру – физическую и логическую. Элементы физической структуры следующие:

1. **дорожка** – концентрическая окружность, по которой движутся головки чтения-записи при размещении или поиске информации. Дорожки нумеруются, начиная с нуля. Нулевой номер имеет самая внешняя дорожка на диске;
2. **секторы** – блоки, в которых размещаются данные на дорожке при записи. Нумеруются, начиная с единицы. Помимо пользовательской информации, сектора содержат служебную информацию, например, собственный номер. Сектора являются минимальными адресуемыми элементами *для диска*;
3. **стороны диска**. Нумеруются, начиная с нуля. Для жесткого диска, расположенного вертикально, нулевой номер имеет самая верхняя сторона;
4. **цилиндр** – совокупность дорожек с одинаковыми номерами на разных сторонах диска. Номера цилиндров совпадают с номерами дорожек;
5. **кластер** – совокупность секторов, имеющих смежные номера. Является минимальным адресуемым элементом для *операционной системы*. Кластеры используются операционной системой для добавления информации к файлу: добавление очередной «порции» к файлу выполняется в объеме кластера независимо от того, что реальный объем меньше объема кластера. Это приводит к нерациональному расходованию внешней памяти. Поэтому не рекомендуется хранить на диске большое количество маленьких файлов: они имеют много пустых «хвостов».

Разбивка непрерывного пространства поверхности диска на указанные элементы выполняется при его форматировании. При этом также формируется маркер начала и конца дорожки, места расположения секторов, в сектора записывается служебная информация.

Рассмотренная физическая структура диска соответствует, в свою очередь, определенной логической структуре, которая частично может меняться в зависимости от типа операционной системы. Ниже приведена основная логическая структура, характерная для любой операционной системы (она описана в порядке возрастания физических адресов):

1. загрузочная запись операционной системы BR (**B**oot **R**ecord). Содержит следующую информацию: программу загрузки операционной системы, размер кластера, количество копий FAT, количество файлов в корневом каталоге Root, размер FAT и некоторую другую информацию;
2. таблица размещения файлов FAT (**F**ile **A**llocation **T**able) и ее копии. Содержит полную карту принадлежности кластеров файлам и используется операционными системами для хранения сведений о размещении файлов на диске и о «плохих» (bad) кластерах. В силу важности FAT она дублируется несколько раз;
3. корневой каталог Root. Это таблица, в которой каждая запись соответствует файлу или подкаталогу[[8]](#footnote-9), подчиненному корневому каталогу диска, и имеет структуру:

* имя файла или подкаталога;
* тип файла,
* атрибуты, в которых определяются следующие параметры файла или подкаталога: предназначенность только для чтения, скрытость, системность, маркер принадлежности данной записи метке тома, признак принадлежности данной записи подкаталогу, а не файлу, архивность;
* время создания;
* дата создания;
* номер начального кластера файла или подкаталога;
* размер файла;

1. область размещения файлов FA (**F**ile **A**rea). Содержит файлы и подкаталоги, которые описаны в Root.

Взаимосвязь областей Root, FAT, FA при поиске файла или каталога по его имени, указанному, например, в команде MS DOS, показана ниже.

Пусть файл с именем ABC.TXT расположен в области FA в тех кластерах, которые выделены серым цветом (их номера показаны ниже):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

10 11 12 13 14 15 16 17

Остальные кластеры свободны. Тогда в FAT в записях с номерами 10 – 17 появятся данные, показанные в таблице:

| номера записей FAT | Содержимое записей FAT |
| --- | --- |
| 10 | 12 |
| 11 |  |
| 12 | 15 |
| 13 |  |
| 14 |  |
| 15 | 17 |
| 16 |  |
| 17 | EOF[[9]](#footnote-10) |

Каждая запись FAT, как видно из таблицы, соответствует одному кластеру и либо содержит номер следующей записи в FAT, соответствующей кластеру с продолжением файла, либо пуста (для незанятых кластеров). Строго говоря, для свободных или «плохих» кластеров записи в FAT заполняются специальными кодами.

В области Root имеется запись с описанием файла ABC.TXT следующей структуры:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя файла | Тип файла | … | Номер первого кластера | … |
| ABC | TXT |  | 10 |  |

Тогда при упоминании имени файла в команде операционная система выполняет следующие действия:

* обращается в область Root, адрес которой фиксирован и известен, и находит там нужную запись (напомним, что операционные системы требуют уникальности имен файлов и подкаталогов, содержащихся в одной директории). Выбирает из этой записи номер первого кластера;
* обращается к FAT, адрес которой фиксирован и известен, и по номеру первого кластера находит соответствующую запись FAT;
* в найденной записи FAT выбирает содержимое записи – это номер следующей записи FAT с описанием продолжения (или окончания) файла;
* если очередная запись FAT содержит код EOF, “сборка” файла закончена; иначе повторяется предыдущий шаг.

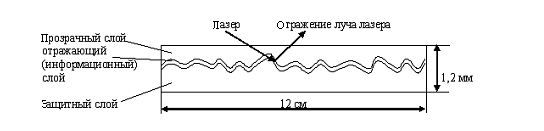
##### 4.3.1.2.2. Оптические накопители

Эта память содержит следующие виды: компакт диски CD, DVD-диски, магнитооптические диски.

Оптические накопители типа СD (**C**ompact **D**isk) включают следующие разновидности:

* *только для чтения* - CD-ROM (Read Only Memory),
* *для однократной записи* - CD-R (Recordable), или CD-WORM (Write Once Read Many),
* *для многократной перезаписи* - CD-RW (ReWritable).

В CD используется лазерный луч с длиной волны 0,78 мкм. Прожигаемая лазером цифровая информация сохраняется в виде пит – «чёрточек» шириной 0,6-0,8 мкм и длиной 0,9-3,3 мкм. Длина пит зависит от количества последовательных двоичных цифр записанного цифрового сигнала. Ямка соответствует двоичному нулю – луч лазера, попадая в нее, поглощается. Гладкая поверхность диска воспринимается как двоичная единица, так как падающий на нее луч лазера отражается, преломляется через призму и фиксируется фотодетектором. Схема CD показана на рис. 4.3.



О

Рис. 4.3. Схема диска типа CD

Линейная скорость вращения современных устройств считывания компакт-дисков постоянна и определяется значением (коэффициентом), определяющим отношение к скорости первых образцов, т.е. указанное на диске число, например, х52, нужно умножить на 150 (это скорость первых компакт-дисков), получится число в Кб/с. Особенности размещения информации:

1. есть только одна спиральная дорожка, «закрученная» против часовой стрелки со стороны поверхности с записью, причем расстояние между витками одинаково,
2. запись на CD начинается от центра диска,
3. линейная скорость вращения постоянна, угловая – переменна.

DVD (1997 г.) – цифровой универсальный диск (Digital Versatile Disk) - применяется для накопления видеоизображений и больших объемов любой компьютерной информации. Как и CD, они делятся на три типа. В них используется луч красного спектра с длинами волн 650 и 535 нм (нанометров) в зависимости от толщины диска.

Конструктивно эти диски аналогичны CD, но имеют бóльший объем за счет бóльшего числа информационных слоев.

Виды DVD и их схемы показаны ниже:

1. односторонний однослойный (SSSL – Single Sided Single Layer) – имеет конструкцию CD (рис. 4.3) – односторонний диск с одним информационным слоем, читается с одной стороны;
2. двусторонний однослойный (флиппер-диск) – DSSL (Double Sided Single Layer) – читается с двух сторон (рис. 4.4);

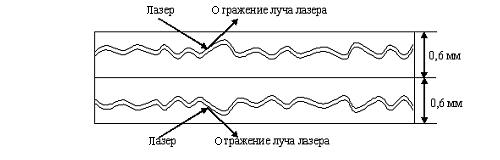


Рис. 4.4. Схема DVD диска типа DSSL

1. односторонний двухслойный (SSDL - Single Sided Double Layer) – двусторонний с одним информационным слоем с каждой стороны, но читается только с одной стороны (рис. 4.5). Верхний информационный и защитные слои прозрачные, нижний информационный слой непрозрачный.

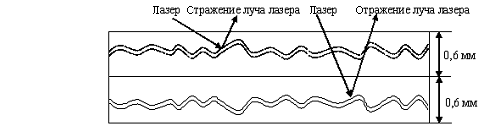


Рис. 4.5. Схема DVD диска типа SSDL

1. двусторонний двухслойный (DSDL – Double Sided Double Layer) - с каждой стороны имеются по два информационных слоя, причем один из них прозрачный. Читается с двух сторон (рис. 4.6).

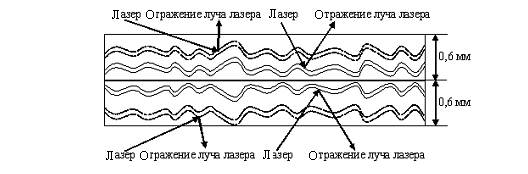


Рис. 4.6. Схема DVD диска типа DSDL

В магнитооптических дисках (1988 г.) запись данных осуществляется лазером с температурой 200оС в магнитооптическом слое одновременно с изменением магнитного поля (рис. 4.7).

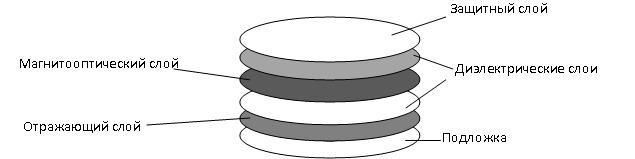


Рис. 4.7. Схема магнитооптического диска

Под воздействием температуры в месте нагрева в магнитооптическом слое уменьшается сопротивляемость изменению полярности, и магнитное поле изменяет полярность в нагретой точке на полярность, соответствующую двоичной единице. По окончании нагрева сопротивляемость увеличивается, но установленная полярность сохраняется.

Стирание создает в магнитном поле одинаковую полярность, соответствующую двоичным нулям, при этом лазерный луч последовательно нагревает стираемый участок.

Считывание данных производится лазером с меньшей интенсивностью, не приводящей к нагреву считываемого участка. При этом в отличие от компакт-дисков поверхность диска не деформируется. Гарантируют сохранность информации - 50 лет.

### 4.3.2. Устройство управления

Структура устройства управления (УУ) определяется важнейшей характеристикой процессора – адресностью машинных команд (см. выше). Рассмотрим структуру УУ для двухадресных команд (рис. 4.8) и взаимодействие его элементов в процессе функционирования.

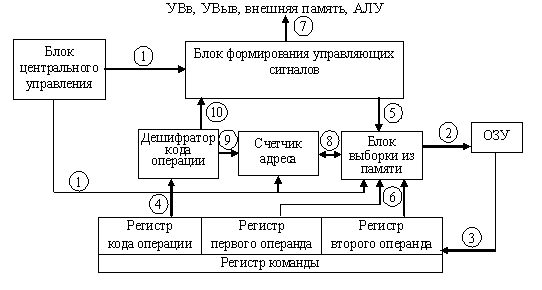


Рис. 4.8. Структура устройства управления

**Блок центрального управления** генерирует сигнал о начале выполнения очередной команды (связь 1). Ее адрес А находится в **счетчике адреса команд**.

**Блок выборки из памяти** по сигналу считывает из **ОЗУ** по адресу А, который выбирается из счетчика адреса команд (связь 8), очередную команду (связь 2) и помещает ее на временное хранение в **регистр команд** (связь 3).

**Дешифратор кода операции** выбирает код (связь 4) и расшифровывает его. Затем передает информацию **блоку формирования управляющих сигналов** (связь 10):

* если операция арифметическая, от **блока формирования управляющих сигналов** поступает сигнал в **блок выборки из памяти** (связь 5) с командой считать из ОЗУ операнды, расположенные по адресам, указанным в **регистрах первого и второго операндов** (связь 6), и поместить их в соответствующие регистры АЛУ (о структуре АЛУ см. в п. 9.3). Затем формируется сигнал в **АЛУ** на выполнение нужной операции (связь 7). Счетчик адреса команд увеличивается на объем команды (связь 9);
* если операция ввода-вывода, **блок формирования управляющих сигналов** формирует сигнал УВв и УВыв (связь 7). Счетчик адреса увеличивается на объем команды (по связи 9);
* если операция условного перехода, **блок центрального управления** анализирует результат предыдущей операции, находящийся в АЛУ. Если знак результата отрицателен, в **счетчик адреса команд** записывается адрес из **регистра первого операнда**. Если знак положителен, в **счетчик адреса команд** записывается адрес из **регистра второго операнда.** Если результат равен 0, в **счетчик адреса команд** добавляется 1 (эти связи не показаны)**.** Так реализуется принцип условного перехода;
* если операция безусловного перехода, в **счетчик адреса команд** пересылается содержимое **регистра первого операнда** (связь не показана).

### 4.3.3. Арифметико-логическое устройство

Структура АЛУ представлена на рис. 4.9.

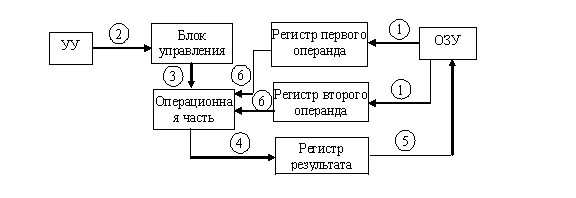


Рис. 4.9. Структура арифметико-логического устройства

Исходные данные (операнды) по командам УУ (см выше) считываются из ОЗУ в **регистры первого** и **второго операндов** (связь 1).

Из УУ в **блок управления АЛУ** поступает команда на выполнение той или иной операции (связь 2), которая передается им в **операционную часть** (связь 3).

В соответствии с этой командой **операционная часть** выполняет нужное действие с данными, которые выбираются из **регистров первого** и **второго операндов** (связь 6). Результат заносится в **регистр результата** (связь 4), откуда – в ОЗУ (связь 5).

Структура регистров АЛУ, куда помещаются исходные и результирующие данные, а также размер регистров (число двоичных разрядов t) формируют понятие **разрядной сетки** (далее – сетки), которое используется ниже.

Для представления данных числовых типов существуют специальные формы, используемые для унификации выполнения арифметических операций. Различаются формы представления для целых и вещественных чисел.

#### 4.3.3.1. Формы представления целых чисел

Пусть сетка имеет в составе t двоичных разрядов. Один из разрядов (например, самый старший) резервируется для знака числа. Сам знак кодируется следующим образом: «-» соответствует 1, «+» соответствует 0. Само число представляется в двоичной системе счисления и заносится в разрядную сетку так, что его самый младший разряд помещается в самый младший разряд сетки, а последующие разряды – следом. Если число имеет меньшее количество разрядов, чем может вместить разрядная сетка, оставшиеся незаполненными разряды заполняются нулями. Таким образом, (t-1) двоичных разрядов занимаются самим числом и, возможно, дополнительными нулями и называются числовыми (в отличие от знакового разряда).

**Пример 4.1**. Пусть разрядная сетка имеет 8 двоичных разрядов. Разместить в ней двоичное число –101112. Результат показан ниже:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| номера разрядов: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

Здесь разряд 7 – знаковый, имеет значение 1, поскольку исходное число отрицательно. В разрядах 0 – 4 размещено само исходное число, разряды 5 и 6 заполнены дополнительными нулями.

**Пример 4.2**. Пусть разрядная сетка имеет 8 двоичных разрядов. Разместить в ней двоичное число 101112. Результат показан ниже:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| номера разрядов: | 7 | 5 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

Различие с предыдущим примером в значении седьмого разряда: поскольку исходное число положительно, там записан 0.

**Пример 4.3**. Пусть разрядная сетка имеет 8 двоичных разрядов. Разместить в ней двоичное число 111101112.

Поскольку размещаемое число содержит разрядов больше, чем количество числовых разрядов разрядной сетки (оно равно 7), задача не имеет решения. Возникает ситуация переполнения (overflow), которая для целых чисел приводит к логической ошибке при выполнении операции.

#### 4.3.3.2. Формы представления вещественных чисел

Для представления вещественных чисел используется логарифмическое представление, или форма с плавающей точкой, или экспоненциальная форма. Она была введена в обиход в 1937 году немецким ученым Конрадом Цузе.

Формальная запись такой формы имеет вид: *x = m× be,* где *x* – вещественное число, *m* – мантисса числа, *b* – основание системы счисления, *e* – порядок (целое).

При обозначении основания *b* и порядка *e* используется, как правило, десятичная система счисления. При обозначении мантиссы *m* применяется, как правило, та система счисления, в которой представлено само число *x*.

Данная форма позволяет перемещать десятичную запятую в вещественном числе вправо и влево, не меняя истинного значения числа.

**Пример 4.4**. Выполнить представление в логарифмической форме десятичного числа 34,28, превратив его в правильную дробь.

Для решения этой задачи надо десятичную запятую в числе сместить как минимум на два разряда влево, т.е. в таком случае уменьшить число на два порядка. Для сохранения первоначального значения числа введем в его запись порядок, равный +2. Имеем: 34,28 = 0,3428×10+2. Здесь 0,3428 – мантисса числа, 10 – основание системы счисления, +2 (можно просто 2) – порядок.

**Пример 4.5**. Выполнить представление в логарифмической форме двоичного числа 0,1011012, превратив его в целое число.

Для решения задачи необходимо сдвинуть десятичную запятую на шесть разрядов вправо, т.е. увеличить число на шесть порядков. Для сохранения первоначального значения числа в его запись введем порядок, равный -6. Имеем: 0,1011012 = 1011012×2-6.

Для простоты обозначения числа в логарифмической форме используют специальный разделитель – букву Е (от слова exponential, англ., - экспоненциальный). Тогда результаты из предыдущих примеров приобретут другой вид:

0,3428 ×102 ⇒ 0,3428Е2,

1011012 × 2-6 ⇒ 1011012Е-6.

В этой записи основание системы счисления *b* подразумевается равным основанию системы счисления для мантиссы *m*.

Разновидностью экспоненциальной формы является ее **нормализованный вид**. Нормализованное вещественное число в экспоненциальной форме имеет мантиссу в виде правильной дроби, у которой старший дробный разряд отличен от 0. Например, 0,2345; 0,10112; 0,ADC2316.

Разрядная сетка для вещественного числа состоит из двух частей: одна предназначена для размещения порядка, другая – для мантиссы. По одному разряду в обеих частях отводится для знака - порядка и мантиссы. Перед размещением в разрядной сетке вещественное число в обязательном порядке должно быть нормализовано.

Размещение порядка выполняется аналогично размещению целого числа. При размещении мантиссы используется только ее дробная часть: старший числовой разряд размещается в самом левом числовом разряде отведенной под нее разрядной сетки. Если число разрядов мантиссы меньше, чем число разрядов сетки, оставшиеся разряды заполняются нулями. Если разрядов сетки не хватает, неразмещенные двоичные цифры отбрасываются.

**Пример 4.6**. Пусть разрядная сетка имеет четырнадцать двоичных разрядов, из них пять разрядов отводятся под порядок, девять – под мантиссу. Пусть под знак отводятся самые левые разряды в соответствующих частях разрядной сетки. Разместить в сетке двоичное число 0,111012Е-4:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| номера разрядов: | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |

Здесь разряды 9 – 13 отведены под порядок, 0 – 8 – под мантиссу. Знаковые разряды, соответственно, имеют номера 13 и 8: поскольку порядок отрицателен, разряд 13 содержит 1; мантисса положительна, поэтому разряд 8 равен 0. В разрядах 9 – 11 размещено значение порядка в двоичной системе счисления. Оставшийся «лишним» разряд 12 заполнен 0. В разрядах 7 – 3 помещено значение мантиссы. Оставшиеся незанятыми разряды 0 – 2 заполнены 0.

**Пример 4.7**. Пусть разрядная сетка имеет четырнадцать двоичных разрядов, из них пять разрядов отводятся под порядок, девять – под мантиссу. Пусть под знак отводятся самые левые разряды в соответствующих частях разрядной сетки. Разместить в сетке двоичное отрицательное число -0,11101101112Е4:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| номера разрядов: | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |

Здесь знаковые разряды имеют значения, противоположные предыдущему примеру. Поскольку число разрядов мантиссы превышает вместимость разрядной сетки, часть из них потеряна: так, вместо мантиссы 0,1110110111 размещена мантисса 0,11101101. Таким образом, при размещении вещественных чисел возможна потеря информации, что приводит к неточным вычислениям на компьютере. Напомним, что часть информации теряется при переводе чисел в двоичную систему счисления. Для повышения точности вычислений рекомендуется использовать типы данных с повышенной разрядностью, например, для Турбо-Паскаля тип long integer вместо integer.

#### 4.3.3.3. Коды представления числовых данных

Для представления данных числовых типов существуют специальные коды, которые призваны облегчить выполнение арифметических операций:

1. **прямой код** есть двоичный эквивалент числа;
2. **обратный код** строится только для отрицательного числа -*х*. Пусть двоичное значение числа –*х* расположено в n двоичных разрядах. Тогда значение обратного кода для *х* (обозначим ⎤*х*) определяется по формуле:

. (4.1)

Иными словами, это преобразованный прямой код, в котором все числовые разряды инвертируются, т.е. 1 преобразуются в 0, 0 превращается в 1;

1. **дополнительный код** строится только для отрицательного числа –*х*. Пусть двоичное значение числа –*х* расположено в n двоичных разрядах. Тогда значение дополнительного кода для *х* (обозначим ⎦*х*) определяется по формуле (4.2):

. (4.2)

Это обратный код, к которому прибавляется 1.

Очевидно следующее: *х* + ⎤*х =* 2n –1.

**Пример 4.8.** Пусть разрядная сетка имеет 4 двоичных разряда, т.е. n=4. Надо построить обратный и дополнительный коды числа -2.

Разместим прямой код числа -2 в разрядной сетке:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 1 | 0 |

Для получения *обратного кода* рассчитаем его значение по формуле (4.1): ⎤*x*=24-2-1=16-3=13. Разместим полученное число 13 (13=11012) в заданной разрядной сетке:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 0 | 1 |

Сравнение обеих разрядных сеток позволяет утверждать, что обратный код можно получить, инвертировав числовые разряды исходного отрицательного числа.

Для получения *дополнительного кода* рассчитаем его значение по формуле (4.2): ⎦*x*=24-2=16-2=14. Разместим полученное число 14 (14=11102) в заданной разрядной сетке:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 0 |

Сравнение разрядных сеток (последней и для обратного кода) позволяет утверждать, что дополнительный код можно получить, увеличив на 1 обратный код числа.

#### 4.3.3.4. Принципы выполнения арифметической операции сложения

В основе выполнения всех простейших арифметических действий лежит операция сложения. Рассмотрим ее выполнение для целых и вещественных чисел.

***При сложении целых чисел последовательность шагов следующая:***

1. слагаемые размещаются в разрядных сетках в прямых кодах;
2. отрицательное слагаемое (или слагаемые) преобразуется в обратный или дополнительный код (в зависимости от того, в какой форме выполняет операции АЛУ);
3. слагаемые складываются по правилам сложения двоичных чисел. При этом знаковые разряды участвуют в вычислениях наряду с числовыми;
4. единица переноса из знакового разряда (если таковая возникнет) отбрасывается при сложении в дополнительном коде или прибавляется к младшему числовому разряду при сложении в обратном коде;
5. если результат положителен – он представлен в прямом коде и не требует никаких преобразований. Если результат отрицателен, то он представлен в обратном или дополнительном коде в зависимости от того, в каком коде происходило сложение. Результат в таком случае преобразуется в прямой код.

**Пример 4.9.** Сложить в обратном коде числа –34 и +15. Разрядная сетка – восемь бит.

1. преобразуем слагаемые в прямые коды и разместим их в разрядных сетках:

-34 = -1000102

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

+15 = +11112

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

1. преобразуем отрицательное слагаемое в обратный код:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |

1. складываем слагаемые:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| слагаемые | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| результат | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |

1. единица переноса не образована;
2. судя по знаку, результат отрицателен, значит, представлен в обратном коде (поскольку сложение выполнялось в этом коде) и требует перевода в прямой код:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |

Таким образом, получено число –100112. Для проверки правильности результата представим его в десятичной системе счисления. Имеем: -100112 = -19, что соответствует правильному результату.

**Пример 4.10**. Сложить в обратном коде числа –34 и -15. Разрядная сетка – восемь бит.

1. преобразуем слагаемые в прямые коды и разместим их в разрядных сетках:

-34 = -1000102

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

-15 = -11112

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

1. преобразуем отрицательные слагаемые в обратный код:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

1. складываем слагаемые:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| слагаемые |  | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| результат | **1** | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| единица переноса из знакового разряда |  |  |  |  |  |  |  |  |  |

1. образовалась единица переноса из знакового разряда. В соответствии с правилами сложения в обратном коде она прибавляется к младшему числовому разряду:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| результат | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| единица переноса | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| окончательный результат | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |

1. судя по знаку, результат отрицателен, значит, представлен в обратном коде (поскольку сложение выполнялось в этом коде) и требует перевода в прямой код. Имеем:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

Таким образом, получено число –1100012. Для проверки правильности результата представим его в десятичной системе счисления. Имеем: -1100012 = -49, что соответствует правильному результату.

**Пример 4.11**. Сложить в дополнительном коде числа –34 и -15. Разрядная сетка – восемь бит.

Первый этап совпадает с предыдущим примером.

Преобразуем слагаемые в дополнительный код. Для этого воспользуемся обратными кодами из предыдущих примеров:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

Складываем эти слагаемые и имеем результат:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| единица переноса из знакового разряда |  |  |  |  |  |  |  |  |

Поскольку сложение выполняется в дополнительном коде, единица переноса из знакового разряда теряется. Таким образом, полученный результат в дополнительном коде. Поскольку он отрицателен, преобразуем его в прямой код:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| результат | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| обратный код | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| прямой код результата | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

При сложении в обратном или дополнительном коде возникают ситуации переполнения.

**Пример 4.12**. Сложить в обратном коде числа –64 и -67. Разрядная сетка – восемь бит.

После перевода чисел в двоичную систему счисления и размещения в разрядной сетке полученных прямых кодов имеем:

для -64

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

для -67

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

После перевода обоих слагаемых в обратный код и сложения имеем:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| слагаемые |  | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| результат | **1** | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| единица переноса из знакового разряда |  |  |  |  |  |  |  |  |  |

После сложения единицы переноса имеем окончательный результат:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

Анализ показывает, что результат положительный, что противоречит исходным данным: складывались два отрицательных числа. Это свидетельствует о переполнении (overflow) разрядной сетки.

Таким образом, формальным признаком переполнения разрядной сетки при выполнении операции сложения является то, что знак результата отличается от знаков слагаемых. Такая ситуация может возникнуть только при сложении чисел с одинаковыми знаками. С подобными ситуациями при сложении целых чисел самостоятельно компьютер не справляется, требуется вмешательство программиста.

Порядок действий при ***сложении вещественных чисел*** гораздо сложнее, нежели для целых чисел. Он приводится ниже:

1. слагаемые размещаются в разрядных сетках в прямых кодах;
2. определяется, порядок какого слагаемого меньше и насколько меньше. Для этого из любого из двух порядков вычитается оставшийся порядок и анализируется результат: если результат положителен, то большим порядком является уменьшаемое, если отрицателен, – то вычитаемое. Значение результата используется в следующих двух шагах. При вычитании, очевидно, выполняется сложение в обратном или дополнительном коде, причем в случае работы с порядками, которые являются целыми числами, эта операция идентична рассмотренной ранее;
3. порядок слагаемого, имеющий меньшее значение, заменяется на второй порядок;
4. числовые разряды мантиссы слагаемого с меньшим порядком сдвигаются вправо на столько разрядов, на сколько был увеличен меньший порядок. Часть разрядов при этом теряется, а оставшиеся свободными разряды заполняются нулями. Шаги 2 – 4 позволяют выровнять порядки исходных слагаемых, что создает необходимые условия для алгебраического сложения;
5. мантиссы складываются. При этом отрицательные мантиссы переводятся в обратный или дополнительный код и складываются по правилам, рассмотренным ранее. Результату приписывается общий (выровненный в предыдущих шагах) порядок;
6. в случае необходимости мантисса результата нормализуется. Необходимость нормализации определяется старшим числовым разрядом мантиссы: если он равен 0, нормализация нужна. Для этого мантисса сдвигается влево на нужное количество разрядов, а порядок уменьшается на соответствующее число. При этом, очевидно, уменьшение порядка выполняется как сложение с константой в обратном или дополнительном коде. Оставшиеся незаполненными младшие разряды мантиссы заполняются нулями.

**Пример 4.13**. Сложить в дополнительном коде 0,10112Е-2 и -0,10012Е-3. Разрядная сетка из девяти разрядов, из них четыре разряда – под порядок, пять разрядов – под мантиссу.

Решение показано ниже:

1. разместим слагаемые в разрядных сетках:

0,10112Е-2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

-0,10012Е-3

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |

1. вычитаем из порядка –2 порядок –3. Имеем: -2 – (-3) = -2 + 3. Таким образом, надо сложить числа –2 и +3 в дополнительном коде, располагая разрядной сеткой в 4 разряда (мы на время «забудем» о мантиссе). При выполнении сложения используются правила сложения в дополнительном коде для целых чисел:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Прямые коды | | | |  |  |  | Обратные  коды | | | |  |  |  | Дополнительные коды | | | |
| 1 | 0 | 1 | 0 |  |  |  | 1 | 1 | 0 | 1 |  |  |  | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 |  |  |  | 0 | 0 | 1 | 1 |  |  |  | 0 | 0 | 1 | 1 |
|  |  |  |  |  |  |  |  | Результат сложения: | | | | | | 0 | 0 | 0 | 1 |

Поскольку результат положителен, он представлен в прямом коде.

Таким образом, большим порядком обладает первое слагаемое (действительно, -2 > -3). Порядок больше на 1.

1. меньший порядок, принадлежащий числу –0,10012Е-3, заменяется на второй порядок:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 1 | 0 |

Таким образом, результат увеличения меньшего порядка равен –102 = -2.

1. сдвиг числовых разрядов мантиссы на один разряд вправо (на время «забываем» о порядке):

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 1 |  | 1 | 0 | 1 | 0 | 0 |

1. сложение мантисс. Поскольку второе слагаемое имеет отрицательный знак, сложение выполняется в дополнительном коде:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Прямые коды | | | | |  |  | Обратные коды | | | | |  |  | Дополнительные коды | | | | |
| 0 | 1 | 0 | 1 | 1 |  |  | 0 | 1 | 0 | 1 | 1 |  |  | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |  |  | 1 | 1 | 0 | 1 | 1 |  |  | 1 | 1 | 1 | 0 | 0 |
|  |  |  |  |  |  |  |  | Результат сложения: | | | | | | 0 | 0 | 1 | 1 | 1 |

Поскольку результат положителен, он представлен в прямом коде. После приписывания результату общего, выровненного порядка имеем:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |

1. поскольку старший числовой разряд мантиссы равен 0, она требует нормализации. Для этого:

* сдвигаем числовые разряды мантиссы на один разряд влево. Освободившийся младший разряд заполняем нулем:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 0 |

* поскольку мантисса увеличилась при сдвиге на один порядок, уменьшаем порядок на 1. Имеем: -102 –12 (или –2 –1). Для сложения отрицательных чисел переведем их в дополнительный код. Тогда:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Прямые коды | | | | |  |  | Обратные коды | | | | |  |  | Дополнительные коды | | | | |
| 1 | 0 | 1 | 0 |  |  |  | 1 | 1 | 0 | 1 |  |  |  | 1 | 1 | 1 | 0 |  |
| 1 | 0 | 0 | 1 |  |  |  | 1 | 1 | 1 | 0 |  |  |  | 1 | 1 | 1 | 1 |  |
|  |  |  |  |  |  |  |  | Результат сложения: | | | | | | 1 | 1 | 0 | 1 |  |

Поскольку результат отрицателен, он представлен в дополнительном коде. Переведем его сначала в обратный код, затем – в прямой код:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 |  | 1 | 0 | 1 | 1 |

После преобразования результат имеет вид:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

Поскольку старший числовой разряд мантиссы равен единице, она нормализована.

Таким образом, получен результат 0,1112 Е-3.

При сложении вещественных чисел также возникают ситуации переполнения, с которыми компьютер справляется самостоятельно. Рассмотрим еще один пример.

**Пример 4.14**. Сложить в дополнительном коде -0,11002Е4 и –0,10002Е4. Разрядная сетка из девяти разрядов, из них четыре разряда – под порядок, пять разрядов – под мантиссу.

Решение:

1. разместим слагаемые в разрядных сетках:

-0,11002 Е4

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

-0,10002 Е4

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

1. вычитаем из порядка 4 порядок 4. Имеем: 4 – 4. Для сложения чисел в дополнительном коде выполним сначала необходимый перевод, а затем - сложение:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Прямые коды | | | |  |  |  | Обратные  коды | | | |  |  |  | Дополнительные коды | | | |
| 0 | 1 | 0 | 0 |  |  |  | 0 | 1 | 0 | 0 |  |  |  | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |  |  |  | 1 | 0 | 1 | 1 |  |  |  | 1 | 1 | 0 | 0 |
|  |  |  |  |  |  |  |  | Результат сложения: | | | | | | 0 | 0 | 0 | 0 |

Поскольку результат равен нулю, порядки слагаемых равны и никаких преобразований по выравниванию порядков (шаги 2 – 4 из приведенной выше последовательности действий) не требуется. Поэтому сразу складываем мантиссы. Поскольку оба слагаемых отрицательны, переведем их сначала в обратный, затем в дополнительный коды:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Прямые коды | | | | |  |  | Обратные коды | | | | |  |  | Дополнительные коды | | | | |
| 1 | 1 | 1 | 0 | 0 |  |  | 1 | 0 | 0 | 1 | 1 |  |  | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |  |  | 1 | 0 | 1 | 1 | 1 |  |  | 1 | 1 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  | Результат сложения: | | | | | | 0 | 1 | 1 | 0 | 0 |

Полученный результат положителен, несмотря на то, что складывались отрицательные слагаемые. Тот факт, что при сложении слагаемых с одинаковыми знаками результат имеет противоположный знак, является формальным признаком переполнения разрядной сетки. В этом случае выполняются следующие действия:

1. мантисса результата вместе со знаком сдвигается вправо на один разряд. Таким образом, старший числовой разряд мантиссы становится равным ее знаковому разряду. При этом, очевидно, теряется младший числовой разряд;
2. знаковый разряд мантиссы результата приравнивается знаковому разряду слагаемых;
3. порядок увеличивается на единицу.

Для нашего примера имеем следующие действия:

1. сдвиг мантиссы («забываем» о порядке):

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 0 | 0 |  |  |  | 0 | 1 | 1 | 0 |

1. формирование знакового разряда дает результат:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 1 | 0 |

1. формирование порядка: к порядку +4 прибавляется 1. Оба слагаемых положительны, поэтому никаких преобразований не требуется:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| слагаемые | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| результат | 0 | 1 | 0 | 1 |

Таким образом, результат примера 4.14 имеет вид:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |

Поскольку мантисса результата отрицательна, она представлена в дополнительном коде и требует перевода в прямой код. Для этого выполним соответствующие преобразования:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| дополнительный код результата | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| обратный код результата | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| прямой код результата | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |

Очевидно, не всегда компьютер в состоянии справиться с переполнением при сложении вещественных чисел. Ограничением является размер разрядной сетки порядка.

# ЛИТЕРАТУРА

1. Гаврилов, М.В. Информатика и информационные технологии. – М.: Юрайт, 2011. – 350 с.
2. Информатика. Базовый курс: учеб./ под ред. С.В. Симоновича. - СПб.: Питер, 2005. – 347 с.
3. Информатика: учеб. / Л.А. Матвеев [и др.]. – М.: Финансы и статистика, 2006. – 496 с.
4. Информатика: учеб. пособ. / А.П. Курносов [и др.]. – М.: КолосС, 2005. – 561 с.
5. Информатика: учеб./ под ред. В.В. Трофимова. – М.: Юрайт, 2011. – 911 с.
6. Мациевский, С.В. Теоретическая информатика. – Калининград: изд-во РГУ. – 2007. – 501 с.

# ПРИЛОЖЕНИЕ 1. Положения комбинаторики, используемые в измерении информации

Комбинаторика – раздел дискретной математики, изучающий способы формирования подмножеств из элементов исходных множеств.

В соответствии с положениями комбинаторики, из конечного счетного множества элементов мощности h можно сформировать следующие простейшие виды комбинаций элементов:

1. **сочетания С**, когда элементы исходного множества группируются в подмножества одинаковой мощности *l,* такие, что элементы в них различаются составом, а порядок элементов безразличен.

Например, пусть исходное множество содержит некоторые символы латинского алфавита и имеет вид - {a,b,c} (h=3). Тогда можно сформировать следующие подмножества мощности 2 по правилу сочетаний: {a,b}, {a,c}, {b,c}. В соответствии с определением сочетания множества {a, b} и {b, a} являются идентичными и не формируются.

1. **перестановки П**, когда элементы исходного множества группируются в подмножества одинаковой мощности *l* (*l* = h) такие, что элементы в них различаются только порядком.

Например, из приведенного выше исходного множества можно сформировать следующие подмножества по правилу перестановок: {a,b,c}, {b,c,a}, {a,c,b}, {b,a,c}, {c,a,b}, {c,b,a}.

1. **размещения Р**, когда элементы исходного множества группируются в подмножества одинаковой мощности *l*, такие, что элементы в них различаются и составом, и порядком.

Например, из приведенного выше исходного множества можно сформировать следующие подмножества по правилу размещения: {a,b}, {b,a}, {a,c}, {c,a}, {b,c}, {c,b}.

Помимо указанных способов, возможны их модификации, когда элементы в результирующих подмножествах могут повторяться (тогда указанные соотношения между *l и* h не выполняются). В этом случае говорят о группировании элементов с повторениями, причем для перестановки указывается, сколько раз повторяется в результирующем подмножестве каждый элемент. Так получаем следующие результаты для примера исходного множества:

1. сочетания по 2 элемента с повторениями (**С**п): {a,b}, {a,c}, {b,c}, {a,a}, {b,b}, {c,c};
2. перестановки с повторениями (**П**п) (число повторений задано: ra=2, rb=1, rc=1, где ri – число повторений элемента i): {a,a,b,c}, {a,a,c,b}, {a,b,a,c}, {a,b,c,a}, {a,c,a,b}, {a,c,b,a}, {b,c,a,a}, {b,a,c,a}, {b,a,a,c}, {c,a,a,b}, {c,a,b,a}, {c,b,a,a};
3. размещения по 2 элемента с повторениями (**Р**п): {a,b}, {b,a}, {a,c}, {c,a}, {b,c}, {c,b}, {a,a}, {b,b}, {c,c}.

Комбинаторика позволяет для каждого из шести указанных способов группирования элементов рассчитывать число получаемых подмножеств:

1. число сочетаний из h элементов по *l* без повторений :
2. число сочетаний из h элементов по *l* с повторениями :
3. число перестановок из h элементов без повторений :
4. число перестановок из h элементов с повторениями ri, где i – номер символа из исходного множества, :
5. число размещений из h элементов по *l* без повторений :
6. число размещений из h элементов по *l* c повторениями :

Рассчитаем число получаемых подмножеств элементов для приведенного выше примера. Имеем:

1. число сочетаний из 3 элементов по 2 без повторений :
2. число сочетаний из 3 элементов по 2 с повторениями :
3. число перестановок из 3 элементов без повторений П(3):
4. число перестановок из 3 элементов с повторениями Пп(3), причем ra=2, rb=1, rc=1**:**
5. число размещений из 3 элементов по 2 без повторений :
6. число размещений из 3 элементов по 2 с повторениями :

1. Флэш - flash (англ.) - вспышка [↑](#footnote-ref-2)
2. Кэш - cash (англ.) - наличные [↑](#footnote-ref-3)
3. BIOS – Basic Input-Output System [↑](#footnote-ref-4)
4. XMA – eXtended Memory Area [↑](#footnote-ref-5)
5. CMA – Conventional Memory Area [↑](#footnote-ref-6)
6. UMA - Upper Memory Area [↑](#footnote-ref-7)
7. HMA – High Memory Area [↑](#footnote-ref-8)
8. Записи для файлов и подкаталогов идентичны за исключением двух характеристик: для подкаталога в поле атрибутов выставлен соответствующий признак и в поле размеров выставлен ноль. [↑](#footnote-ref-9)
9. End Of File (англ.) – конец файла [↑](#footnote-ref-10)