

Учреждение образования  
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

П. П. Урбанович, Д. В. Шиман

# ЗАЩИТА ИНФОРМАЦИИ И НАДЕЖНОСТЬ ИНФОРМАЦИОННЫХ СИСТЕМ

*Рекомендовано  
учебно-методическим объединением  
по образованию в области информатики  
и радиоэлектроники в качестве пособия  
для студентов учреждений высшего образования  
по направлению специальности 1-40 05 01-03  
«Информационные системы и технологии  
(издательско-полиграфический комплекс)»*

Минск 2014

УДК 004.056(075.8)  
ББК 32.97я7  
У69

**Р е ц е н з е н т ы :**

кафедра информационных технологий автоматизированных  
систем УО «Белорусский государственный университет  
информатики и радиоэлектроники»  
(кандидат технических наук, доцент,  
заведующий кафедрой *А. А. Навроцкий*;  
кандидат технических наук, доцент кафедры *О. В. Герман*);  
доктор технических наук, профессор, директор учреждения  
«Главный информационно-аналитический центр Министерства  
образования Республики Беларусь» *Н. И. Листопад*

*Все права на данное издание защищены. Воспроизведение всей книги или ее части не может быть осуществлено без разрешения учреждения образования «Белорусский государственный технологический университет».*

**Урбанович, П. П.**

У69      Защита информации и надежность информационных  
систем : пособие для студентов направления специальности  
1-40 05 01-03 «Информационные системы и технологии» /  
П. П. Урбанович, Д. В. Шиман. – Минск : БГТУ, 2014. – 90 с.  
ISBN 978-985-530-367-2.

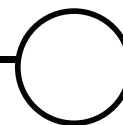
Цель издания – помочь студентам в овладении теоретическими знаниями о методах и средствах повышения информационной безопасности и надежности систем хранения, преобразования и передачи информации, ее защиты в информационно-вычислительных системах, в приобретении практических навыков по созданию и использованию методов и средств повышения информационной безопасности и надежности систем.

Пособие также может быть полезно аспирантам, изучающим избранные аспекты вышеуказанной проблемы. Издание предназначено для выполнения заданий на занятиях по курсу «Информационная безопасность и надежность систем».

**УДК 004.056(075.8)  
ББК 32.97я7**

**ISBN 978-985-530-367-2**

© Урбанович П. П., Шиман Д. В., 2014  
© УО «Белорусский государственный  
технологический университет», 2014



Настоящее пособие предназначено в помощь студентам заочной формы обучения специальности «Информационные системы и технологии» при изучении ими дисциплины «Защита информации и надежность информационных систем».

В соответствии с учебным планом дисциплины аудиторные занятия проводятся в форме лекций – 18 ч (по 6 ч в 8, 9 и 10-м семестрах) и лабораторных занятий – 18 ч (10 ч – в 9-м семестре, 8 ч – в 10-м). Промежуточный контроль знаний студентов осуществляется по результатам выполнения индивидуальных заданий в форме тестов на компьютере (по одному заданию в 9-м и 10-м семестрах).

Изучению дисциплины должно предшествовать усвоение базовых курсов высшей математики, физики, микропроцессорных и вычислительных устройств, базовых языков программирования, основ построения и функционирования реляционных баз данных.

В процессе изучения настоящей дисциплины студент должен освоить основы создания защищенных информационно-вычислительных систем, включающие анализ угроз, перечень атак, методов и средств защиты информации, методологию оценки надежности и информационной безопасности.

В результате изучения дисциплины студент должен знать и уметь реализовать на практике:

- 1) особенности информационных (информационно-вычислительных) систем (ИС (ИВС)) как объекта защиты;
- 2) правовые методы защиты ИС;
- 3) организационные методы защиты информации в ИС;
- 4) программно-технические средства преобразования и защиты информации в ИС;
- 5) методы криптографической защиты информации;
- 6) методы и средства повышения функциональной надежности программных, аппаратных и аппаратно-программных средств ИС (ИВС).

Весь учебный материал, в соответствии с учебным планом дисциплины, структурирован в виде шести разделов, разделенных на 18 тем. Настоящее пособие призвано помочь студентам в изучении тем с первой по седьмую. Заголовки в пособии полностью повторяют названия тем из учебного плана дисциплины.

# ФУНДАМЕНТАЛЬНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ ИЗ ОБЛАСТИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ И НАДЕЖНОСТИ СИСТЕМ

1

## Основные понятия и определения

Информация – сведения (данные) о внутреннем и окружающем нас мире, событиях, процессах, явлениях и т. д., воспринимаемые и передаваемые людьми или техническими устройствами.

Информационная (информационно-вычислительная) система – организационно упорядоченная совокупность документов, технических средств и информационных технологий, реализующая информационные (информационно-вычислительные) процессы.

Информационные процессы – процессы сбора, накопления, хранения, обработки (переработки), передачи и использования информации.

Информационные ресурсы – отдельные документы или массивы документов в информационных системах.

Доступ – специальный тип взаимодействия между объектом и субъектом, в результате которого создается поток информации от одного к другому.

Несанкционированный доступ (НСД) – доступ к информации, устройствам ее хранения и обработки, а также к каналам передачи, реализуемый без ведома (санкции) владельца и нарушающий тем самым установленные правила доступа.

Объект – пассивный компонент системы, хранящий, перерабатывающий, передающий или принимающий информацию; примеры объектов: страницы, файлы, папки, директории, компьютерные программы, устройства (мониторы, диски, принтеры и т. д.).

Субъект – активный компонент системы, который может инициировать поток информации; примеры субъектов: пользователь, процесс либо устройство.

Безопасность ИВС – свойство системы, выражающееся в способности системы противодействовать попыткам несанкционированного доступа или нанесения ущерба владельцам и пользователям системы при различных умышленных и неумышленных воздействиях на нее.

Защита информации – организационные, правовые, программно-технические и иные меры по предотвращению угроз информационной безопасности и устранению их последствий.

Атака – попытка несанкционированного преодоления защиты системы.

Информационная безопасность системы – свойство информационной системы или реализуемого в ней процесса, характеризующее способность обеспечить необходимый уровень своей защиты.

Информационная безопасность – все аспекты, связанные с определением, достижением и поддержанием конфиденциальности, целостности, доступности информации или **средств ее обработки**:

- конфиденциальность (confidentiality) – состояние информации, при котором доступ к ней осуществляют только субъекты, имеющие на нее право;
- целостность (integrity) – избежание несанкционированной модификации информации;
- доступность (availability) – избежание временного или постоянного сокрытия информации от пользователей, получивших права доступа.

Идентификация – процесс распознавания определенных компонентов системы (объектов или субъектов) с помощью уникальных идентификаторов.

Аутентификация – проверка идентификации пользователя или иного компонента ИС для принятия решения о разрешении доступа к ресурсам системы.

Надежность системы – характеристика способности программного, аппаратного, аппаратно-программного средства выполнить при определенных условиях требуемые функции в течение определенного периода времени.

Достоверность работы системы (устройства) – свойство, характеризующее истинность конечного (выходного) результата работы (выполнения программы), определяемое способностью средств контроля фиксировать правильность или ошибочность работы.

Ошибка устройства – неправильное значение сигнала (бита – в цифровом устройстве) на внешних выходах устройства или отдельного его узла, вызванное технической неисправностью, или воздействующими на него помехами (преднамеренными либо непреднамеренными), или иным способом.

Ошибка программы – проявляется в не соответствующем реальному (требуемому) промежуточном или конечном значении (результату) вследствие неправильно запрограммированного алгоритма или неправильно составленной программы.

Основные определения из предметной области можно найти также в п. 1.1, 1.2, 1.5 пособия [1].

## **Краткая историческая информация**

I этап – примерно до 1816 года – характеризуется использованием естественно возникавших средств информационных коммуникаций. Основная задача информационной безопасности – защита сведений о событиях, фактах, имуществе и т. д.

II этап – начиная с 1816 года – связан с **началом использования технических средств электро- и радиосвязи**. Характеризуется **применением помехоустойчивого кодирования сообщения (сигнала) с последующим декодированием принятого сообщения (сигнала)**.

III этап – начиная с 1935 года – связан с появлением **радиолокационных и гидроакустических средств**. Обеспечение информационной безопасности основывалось на сочетании организационных и технических мер, направленных на **повышение защищенности радиолокационных средств от воздействия на их приемные устройства активных и пассивных помех**.

IV этап – начиная с 1946 года – связан с **изобретением и внедрением в практическую деятельность электронно-вычислительных машин (компьютеров)**. Эру появления компьютерной техники связывают с разработкой в **Пенсильванском университете (США) ЭВМ EN IAC (Electronic Numerical Integrator And Computer (Calculator))**. Задачи информационной безопасности **решались в основном методами и способами ограничения физического доступа к оборудованию средств сбора, переработки и передачи информации**.

V этап – начиная с 1964 года – обусловлен **созданием и развитием локальных информационно-коммуникационных сетей**. Задачи безопасности решались в основном методами и **способами физической защиты средств, путем администрирования и управления доступом к сетевым ресурсам**.

**VI этап** – начиная с 1973 года – связан с использованием мобильных коммуникационных устройств с широким спектром задач. В этот период созданы известные сейчас во всем мире фирмы Microsoft (Билл Гейтс и Пол Аллен) и Apple (Стив Джобс и Стивен Возняк).

Образовались сообщества людей – *хакеров*, ставящих своей целью нанесение ущерба информационной безопасности отдельных пользователей, организаций и целых стран. Формируется *информационное право* – новая отрасль международной правовой системы.

**VII этап** – начиная с 1985 года – связан с созданием и развитием глобальных информационно-коммуникационных сетей с использованием космических средств обеспечения.

Предусматривает комплексное использование мер и средств защиты.

**VIII этап** – с конца XX – начала XXI вв. – связан с повсеместным использованием сверхмобильных коммуникационных устройств с широким спектром задач и глобальным охватом в пространстве и времени, обеспечиваемым космическими информационно-коммуникационными системами. Характеризуется «*широким переходом на цифру*». Предусматривает комплексное использование мер и средств защиты.

## **Общая характеристика факторов, влияющих на безопасность и надежность ИВС**

Фактор, воздействующий на ИВС, – это явление, действие или процесс, результатом которых может быть утечка, искажение, уничтожение данных, блокировка доступа к ним, повреждение или уничтожение системы защиты.

Все многообразие дестабилизирующих факторов можно разделить на два класса: внутренние и внешние.

Внутренние дестабилизирующие факторы влияют:

**1) на программные средства (ПС):**

- а) некорректный исходный алгоритм;
- б) неправильно запрограммированный исходный алгоритм (первичные ошибки);

**2) на аппаратные средства (АС):**

- а) системные ошибки при постановке задачи проектирования;
- б) отклонения от технологии изготовления комплектующих изделий и АС в целом;

в) нарушение режима эксплуатации, вызванное внутренним состоянием АС.

Внешние дестабилизирующие факторы влияют:

1) на программные средства:

а) неквалифицированные пользователи;  
б) несанкционированный доступ к ПС с целью модификации кода;

2) на аппаратные средства:

а) внешние климатические условия;  
б) электромагнитные и ионизирующие помехи;  
в) перебои в электроснабжении;  
г) недостаточная квалификация обслуживающего персонала;  
д) несанкционированный (в том числе удаленный) доступ с целью нарушения работоспособности АС.



### **Вопросы для контроля и самоконтроля**

1. Дать определение основных понятий и терминов, относящихся к области защиты информации и надежности информационных систем.
2. Чем отличается идентификация от авторизации?
3. Охарактеризовать основные этапы развития информационных технологий с точки зрения их безопасности.
4. Привести классификацию основных факторов, влияющих на ИВС.
5. К каким последствиям приводит влияние дестабилизирующих факторов на ИС или ИВС?
6. Дать характеристику внутренних факторов, дестабилизирующих работу ИС или ИВС.
7. Охарактеризовать внешние факторы, дестабилизирующие работу ИС или ИВС.
8. Как Вы понимаете «некорректный исходный алгоритм»?
9. Что такое «первичная ошибка» в программе?
10. Как влияют климатические условия на надежность аппаратных средств?



# ПОТЕНЦИАЛЬНЫЕ УГРОЗЫ БЕЗОПАСНОСТИ ИНФОРМАЦИИ В ИВС. ОБЪЕКТЫ И МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ

2

## Естественные и искусственные помехи

Одним из важнейших дестабилизирующих работу ИВС факторов считают электромагнитные и ионизирующие излучения. Источниками первых являются практически все средства, функционирование которых основано на использовании электроэнергии, и в особенности такие АС, которые целенаправленно излучают электромагнитные волны (к ним относятся, например, приемо-передающие и иные подобные радиоэлектронные устройства). По большому счету, любой проводник с током является источником электромагнитных помех. Такие источники относятся к классу искусственных или промышленно-бытовых. В свою очередь их можно подразделить на непреднамеренные и преднамеренные. Последние имеют место в ситуациях, похожих на эпизод в фильме «Операция “Ы” и другие приключения Шурика»: профессор на экзамене включил генератор помех, который «забил» канал.

Ионизирующие излучения также могут иметь естественную (солнечная радиация) и искусственную (изотопы урана и тория излучают даже пластмассы) природу.

Основным последствием влияния помех на АС являются ошибки в хранящейся, передаваемой или обрабатываемой информации. Другими словами – помехи снижают функциональную надежность АС.

Для лучшего понимания сути и особенностей угроз со стороны деструктивных программных средств, а также физических лиц, использующих такие средства для организации НСД (хакеры и кракеры), на рис. 2.1 приведен пример периметра современной ИС, а на рис. 2.2 схематически показаны наиболее уязвимые места локальной сети.

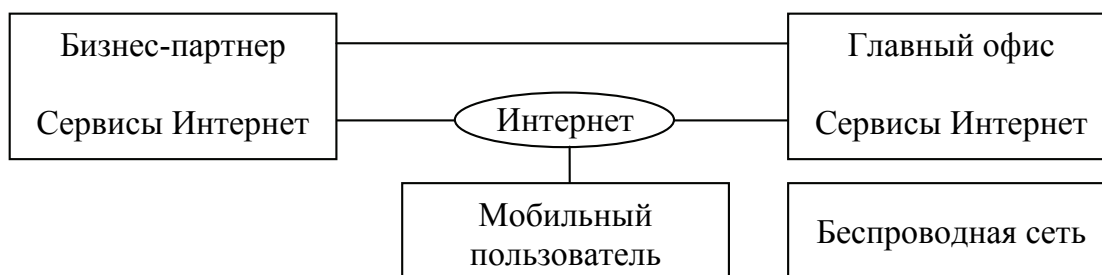


Рис. 2.1. Пример периметра современной ИС

Основные факторы (угрозы):

- 1) действия злоумышленника;
- 2) наблюдение за источниками информации;
- 3) подслушивание конфиденциальных разговоров и акустических сигналов работающих механизмов;
- 4) перехват электрических, магнитных и электромагнитных полей, электрических сигналов и радиоактивных излучений;
- 5) несанкционированное распространение материальных носителей за пределами организации;
- 6) разглашение информации компетентными людьми;
- 7) утеря носителей информации;
- 8) несанкционированное распространение информации через поля и электрические сигналы, случайно возникшие в аппаратуре;
- 9) воздействие стихийных сил (наводнения, пожары и т. п.);
- 10) сбои и отказы в аппаратуре сбора, обработки и передачи информации;
- 11) отказы системы электроснабжения;
- 12) воздействие мощных электромагнитных и электрических помех (промышленных и природных).

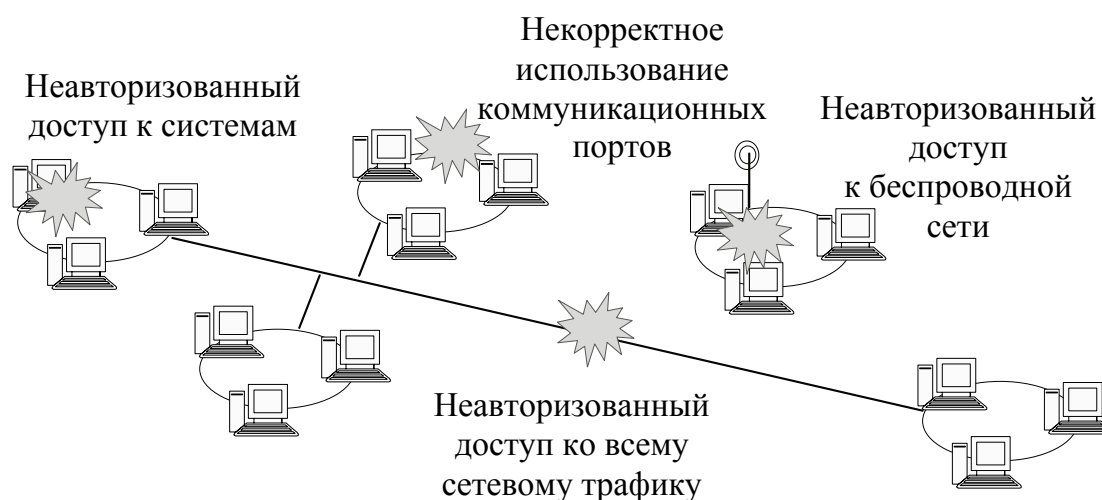


Рис. 2.2. Наиболее уязвимые места локальной сети

Несанкционированный доступ с помощью деструктивных программных средств осуществляется, как правило, через компьютерные сети.

Классификацию вредоносного ПО можно представить следующим образом:

- **вирусы (viruses)** – это саморазмножающиеся программы путем дописывания собственных кодов к исполняемым файлам; вирусы могут содержать или не содержать деструктивные функции;

- **черви (worms)** – это программы, которые самостоятельно размножаются по сети и, в отличие от вирусов, не дописывают себя (как правило) к исполняемым файлам; все черви «съедают» ресурсы компьютера, «нагоняют» интернет-трафик и могут привести к утечке данных с вашего компьютера;

- **анализаторы клавиатуры, или кейлоггеры (keyloggers)**, – программы, которые регистрируют нажатия клавиш, делают снимки рабочего стола, отслеживают действия пользователя во время работы за компьютером и сохраняют эти данные в скрытый файл на диске, затем этот файл попадает к злоумышленнику;

- **трояны (trojans), или троянские кони**, – собирают конфиденциальную информацию с компьютера пользователя (пароли, базы данных и пр.) и тайно по сети высылают их злоумышленнику (своему хозяину);

- **боты (bots)** – распространенный в наше время вид зловредного ПО, который устанавливается на компьютерах пользователей (*seti botnet*) и используется для атак на другие компьютеры;

- **снифферы (sniffers)** – это анализаторы сетевого трафика; могут использоваться в составе зловредного ПО, скрытно устанавливаться на компьютере пользователя и отслеживать данные, которые отправляет или получает пользователь по сети;

- **руткиты (rootkits)** – сами по себе не являются зловредным ПО; назначение – скрывать работу других зловредных программ (кейлоггеров, троянов, червей и т. д.) как от пользователя, так и от программ и средств обеспечения безопасности (антивирусов, файерволов (firewalls), систем обнаружения атак и пр.).

Важно отметить, что вирусы, трояны и иные деструктивные программы активизируются только после загрузки инфицированного файла в оперативную память компьютера (RAM).

Более подробно свойства и особенности деструктивных программ, как и методы борьбы с ними, будут проанализированы во второй части курса.

## Основные методы повышения безопасности и надежности ИС и ИВС

В контексте сформулированной цели здесь кратко проанализируем методы и средства повышения информационной безопасности систем. Вопросы надежности будут рассмотрены во второй части курса (при подготовке ко второму тесту).

Политика информационной безопасности (ИБ) систем должна строиться на основе системного подхода, предусматривающего всесторонний анализ причин и угроз безопасности, оценки их последствий, необходимости, экономической или иной целесообразности и адекватности принимаемых противодействий.

Все многообразие используемых методов и средств защиты можно разделить на три класса:

- законодательная, нормативно-правовая и научная база;
- организационно-технические и режимные меры и методы (политика информационной безопасности);
- аппаратные, программно-аппаратные и программные способы и средства обеспечения ИБ.

К первому классу относятся следующие:

### 1. Акты национального законодательства:

- а) международные договоры Республики Беларусь;
- б) Конституция Республики Беларусь;
- в) законы Республики Беларусь, например *Закон Республики Беларусь от 10 ноября 2008 г. № 455-3 «Об информации, информатизации и защите информации»;*
- г) указы Президента Республики Беларусь, например *Указ № 515 Президента Республики Беларусь от 30 сентября 2010 г. «О некоторых мерах по развитию сети передачи данных в Республике Беларусь», Указ № 60 Президента Республики Беларусь от 1 февраля 2010 г. «О мерах по совершенствованию использования национального сегмента сети Интернет»;*
- д) постановления Правительства РБ, например *постановление Совета Министров Республики Беларусь от 11 февраля 2006 г. № 192 «Об утверждении Положения о сопровождении интернет-сайтов республиканских органов государственного управления, иных государственных организаций, подчиненных Правительству Республики Беларусь»;* *постановление Совета Министров Республики Беларусь от 11 августа 2011 г. № 1084 «О внесении изменений и дополнений*

в постановление Совета Министров Республики Беларусь от 29 апреля 2010 г. № 644», *постановление Совета Министров Республики Беларусь от 26 мая 2009 г. № 675 «О некоторых вопросах защиты информации»*, *постановление Совета Министров Республики Беларусь от 26 мая 2009 г. № 673 «О некоторых мерах по реализации Закона Республики Беларусь “Об информации, информатизации и защите информации” и о признании утратившими силу некоторых постановлений Совета Министров Республики Беларусь»*;

е) нормативные правовые акты министерств и ведомств, например *постановление № 4/11 Оперативно-аналитического центра при Президенте Республики Беларусь и Министерства связи и информатизации Республики Беларусь от 29 июня 2010 г. «Об утверждении положения о порядке ограничения доступа пользователей интернет-услуг к информации, запрещенной к распространению в соответствии с законодательными актами»*, *приказ № 60 Оперативно-аналитического центра при Президенте Республики Беларусь от 2 августа 2010 г. «Об утверждении положения о порядке определения поставщиков интернет-услуг, уполномоченных оказывать интернет-услуги государственным органам и организациям, использующим в своей деятельности сведения, составляющие государственные секреты»*;

ж) нормативные правовые акты субъектов, органов местного самоуправления и т. д.

## 2. Международные стандарты, например:

а) BS 7799-1:2005 – Британский стандарт BS 7799 Part 1 – *Code of Practice for Information Security Management* (Практические правила управления информационной безопасностью) описывает 127 механизмов контроля, необходимых для построения системы управления информационной безопасностью организации, определенных на основе лучших примеров мирового опыта в данной области. Этот документ служит практическим руководством по созданию системы управления информационной безопасностью (СУИБ);

б) BS 7799-2:2005 – Британский стандарт BS 7799 Part 2 – *Information Security management* (Specification for information security management systems – спецификация системы управления информационной безопасностью) определяет спецификацию СУИБ. Вторая часть стандарта используется в качестве

критериев при проведении официальной процедуры сертификации СУИБ организации;

в) ISO/IEC 17799:2005 – «Информационные технологии – Технологии безопасности – Практические правила менеджмента информационной безопасности». Международный стандарт, базирующийся на BS 7799-1:2005;

г) ISO/IEC 27001:2005 – «Информационные технологии – Методы обеспечения безопасности – Системы управления информационной безопасностью – Требования». Международный стандарт, базирующийся на BS 7799-2:2005;

д) ISO/IEC 27002 – сейчас: ISO/IEC 17799:2005 – «Информационные технологии – Технологии безопасности – Практические правила менеджмента информационной безопасности». Дата выхода – 2007 год.

е) ISO/IEC 27005 – сейчас: BS 7799-3:2006 – Руководство по менеджменту рисков ИБ.

## **Организационно-технические и режимные меры и методы**

Для построения политики ИБ рассматривают следующие направления защиты ИС:

- защита объектов ИС;
- защита процессов, процедур и программ обработки информации;

- защита каналов связи;
- подавление побочных электромагнитных излучений;
- управление системой защиты.

Организационная защита обеспечивает:

- организацию охраны, режима, работу с кадрами, с документами;

- использование технических средств безопасности (например, простейших дверных замков, магнитных или иных карт и др.), информационно-аналитическую деятельность по выявлению внутренних и внешних угроз.

Аппаратные, программно-аппаратные и программные способы и средства обеспечения ИБ условно можно классифицировать следующим образом:

1) средства защиты от несанкционированного доступа:

- а) средства авторизации;
- б) аудит;



2) системы мониторинга сетей:

- а) системы мониторинга сетей;
- б) анализаторы протоколов;

3) антивирусные средства:

- а) антивирусные программы;
- б) программные и иные антиспамовые средства;
- в) межсетевые экраны;

4) криптографические средства:

- а) шифрование данных;
- б) электронная цифровая подпись;

5) системы бесперебойного питания;

6) системы аутентификации:

- а) пароль;
- б) ключ доступа (физический или электронный);
- в) биометрия (анализаторы отпечатков пальцев, анализаторы сетчатки глаза, анализаторы голоса, анализаторы геометрии ладони и др.).

Вопросы систематизации методологии информационной безопасности достаточно подробно описаны в [2].

## **Вопросы для контроля и самоконтроля**

1. К чему приводят электромагнитные или ионизирующие излучения?
2. Привести пример источников электромагнитных или ионизирующих излучений.
3. Что такое «преднамеренная помеха», «непреднамеренная помеха»?
4. Из каких основных частей состоит периметр современной ИС?
5. Как можно осуществить неавторизованный доступ к сетевому трафику?
6. К каким последствиям могут привести отказы системы электроснабжения ИС?
7. Что такое «компьютерный вирус»? Чем он отличается от остальных деструктивных программ? Привести примеры известных вирусов.
8. Охарактеризовать известные Вам деструктивные программные средства.
9. В чем сущность системного подхода при проектировании политики безопасности?

10. Дать классификацию методов и средств защиты информации.

11. Перечислить основные направления реализации политики информационной безопасности.

12. В чем назначение организационных методов защиты информации?

13. В чем назначение правовых методов защиты информации?

14. В чем назначение режимных методов защиты информации?

15. Какие национальные правовые акты, регламентирующие доступ к информационным ресурсам, Вы знаете?

16. Какие международные правовые акты, регламентирующие доступ к информационным ресурсам, Вы знаете?

17. В чем заключается назначение организационно-технических методов защиты информации?

18. Как можно защитить каналы связи?

19. Как можно защититься от мешающих электромагнитных излучений?

20. Перечислить известные методы и средства авторизации.

21. Назначение межсетевых экранов.

22. Назначение источников бесперебойного питания.

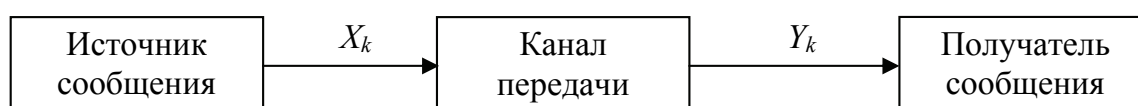
23. На чем основаны биометрические средства идентификации?



# ОБЩАЯ ХАРАКТЕРИСТИКА, СТРУКТУРА И МАТЕМАТИЧЕСКОЕ ОПИСАНИЕ КАНАЛОВ ПЕРЕДАЧИ И ХРАНЕНИЯ ИНФОРМАЦИИ

3

Передача информации (данных) осуществляется между двумя абонентами, называемыми *источником сообщения (ИсС)* и *получателем сообщения (ПС)*. Источником и получателем могут быть люди либо технические средства. ИсС и ПС обмениваются информацией посредством канала передачи. Отметим также, что и в системах хранения информации всегда можно выделить ИсС и ПС. В данном случае каналом передачи выступает *устройство хранения информации* (память). Например, при записи данных в ОЗУ (оперативное запоминающее устройство) компьютера в качестве ИсС и ПС может выступать процессор (соответственно при записи и чтении данных). Таким образом, простейшая информационная система состоит из трех перечисленных элементов. Ее обобщенная структурная схема приведена на рисунке (здесь параметр  $k$  означает число символов в сообщении).



Обобщенная структурная схема информационной системы  
(системы передачи информации)

Под определение ИС (ИВС) подпадает любая система обработки информации. Далее будем рассматривать ИС как совокупность аппаратно-программных средств, задействованных для решения некоторой прикладной задачи.

Эффективность функционирования ИС во многом зависит от ее архитектуры. В настоящее время наиболее распространенной является архитектура клиент – сервер.

Математической основой описания и анализа процессов в ИС в широком смысле является теория информации. Возникновение

теории информации связывают обычно с появлением фундаментальной работы американского ученого К. Шеннона [3].

Для перенесения информации в пространстве и времени она представляется в форме сообщений. *Сообщение*, вне зависимости от его содержания, всегда отображается в виде сигнала.

*Сигнал* – физический процесс, отображающий передаваемое сообщение.

Отображение сообщения обеспечивается изменением какой-либо физической величины, характеризующей процесс (например, амплитуда, частота, фаза). Эта величина является *информационным параметром сигнала*.

Сигналы, как и сообщения, могут быть *непрерывными* и *дискретными*. Информационный параметр непрерывного сигнала с течением времени может принимать любые мгновенные значения в определенных пределах. Непрерывный сигнал часто называют *аналоговым*. Дискретный сигнал характеризуется конечным числом значений информационного параметра. Часто этот параметр принимает всего два значения (0 или 1).

Сообщение или канал его передачи на основе этих двух значений сигнала называют *двоичным* или *бинарным*.

Построение сигнала по определенным правилам, обеспечивающим соответствие между сообщением и сигналом, называют *кодированием*.

Кодирование в широком смысле – *преобразование сообщения в сигнал*.

Кодирование в узком смысле – *представление исходных знаков*, называемых символами, в другом алфавите с меньшим числом знаков. Оно осуществляется с целью повышения надежности и преобразования сигналов к виду, удобному для передачи по каналам связи. Об этом речь пойдет ниже.

Здесь же отметим, что последний тип кодирования относится к так называемой *прикладной теории кодирования информации*, занимающейся поиском и реализацией методов и средств обнаружения несоответствий (*ошибок*) между переданным  $X_k$  и принятым  $Y_k$  сообщениями.

Двоичный канал передачи информации, в котором вероятность искажения переданного – 0 (принятого – 1) и переданного – 1 (принятого – 0), называют *двоичным симметричным каналом (ДСК)*.



## **Вопросы для контроля и самоконтроля**

1. В каком наиболее общем виде можно представить структуру ИС?
2. Что такое «источник сообщения»? Привести примеры.
3. Что такое «получатель сообщения»? Привести примеры.
4. Что такое «канал передачи данных»? Привести примеры.
5. Привести примеры ИС, ИВС.
6. Что такое «двоичный канал передачи данных»?
7. Почему «двоичный симметричный канал» относится к двоичным?
8. Почему «двоичный симметричный канал» относится к симметричным?
9. Представить схематично «двоичный симметричный канал».

# ПОНЯТИЕ ИНФОРМАЦИИ. ЭНТРОПИЯ ИСТОЧНИКА СООБЩЕНИЙ

4

## Основы теории информации К. Шеннона. Понятие алфавита источника сообщения. Энтропия Шеннона и Хартли.

ИсС и ПС обмениваются информацией в технических системах в виде сигналов, сформированных на основе определенного *алфавита*. Характеристикой алфавита является его *мощность*,  $N$  – количество символов, на основе которых формируется сообщение. Например, мощность английского алфавита – 26 символов, русского – 33 символа, мощность алфавита, на основе которого функционируют и взаимодействуют между собой компьютеры, составляет 2 символа (0 и 1).

Далее будем рассматривать только ИсС и ПС дискретных сообщений. Наибольший интерес представляет собой цифровой сигнал на основе алфавита  $A\{0,1\}$ .

В произвольном сообщении символы алфавита могут появляться с различной вероятностью. Если длина сообщения достаточно велика, то статистический анализ этого сообщения позволит получить вероятностные характеристики данного алфавита. Очевидно, что отличные символы в произвольном сообщении (особенно при  $N > 2$ ) появляются с различной вероятностью, т. е. существуют символы с минимальной и максимальной вероятностью появления. Например, подсчитано, что наиболее часто (в 13% случаев) в документах на английском языке ( $N = 26$ ) появляется буква «е», а наиболее редко – буквы «х», «q» и «z». В таком случае вероятность того, что произвольный символ  $\xi$  произвольного документа (текст, база данных, текст программы) будет буквой «е» (или другой из указанных букв), записывается так:

$$\begin{aligned}P(\xi = e) &= 0,13, P(\xi = x) = 0,0015, \\P(\xi = y) &= 0,0010, P(\xi = z) = 0,0007.\end{aligned}$$

Информационной характеристикой алфавита (источника сообщений на основе этого алфавита) является *энтропия*.

Этот термин применительно к техническим системам был введен К. Шенноном и Р. Хартли.

Энтропию алфавита  $A\{a_i\}$  по Шеннону рассчитывают по следующей формуле:

$$H_S(A) = -\sum_{i=1}^N P(a_i) \cdot \log_2 P(a_i), \quad (4.1)$$

где  $P(a_i)$  – вероятность  $P(\xi = a_i)$ ;  $a_i$  – элемент алфавита,  $i = \overline{1, N}$ .

Заметим, что  $\sum_{i=1}^N P(a_i) = 1$ .

С физической точки зрения *энтропия показывает, какое количество информации приходится в среднем на один символ алфавита*.

Частным случаем энтропии Шеннона является энтропия Хартли. Дополнительным условием при этом является то, что все вероятности одинаковы и постоянны для всех символов алфавита. С учетом этого формулу (4.1) можно преобразовать к виду:

$$H_H(A) = \log_2 N.$$

Например, энтропия Хартли для латинского (английского) алфавита составляет 4,7 бит.

Если подсчитать энтропию Шеннона и энтропию Хартли для одного и того же алфавита, то они окажутся неравными. Это несоответствие указывает на избыточность любого алфавита (при  $N > 2$ ).

Сообщение  $M$ , которое состоит из  $n$  символов, должно характеризоваться определенным *количеством информации*  $I(M)$ :

$$I(M) = H(A) \cdot n. \quad (4.2)$$

Нетрудно предположить и просто убедиться, что количество информации в сообщении, подсчитанное по Шеннону, не равно количеству информации, подсчитанному по Хартли. На основе этого парадокса строятся и функционируют все современные системы сжатия (компрессии) информации.

## **Двоичный канал передачи информации**

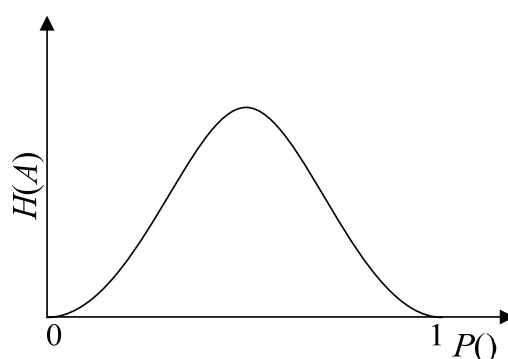
Как было подчеркнуто выше, двоичный канал передачи информации является дискретным – он основан на алфавите, состоящем из двух символов: 0 и 1 –  $A\{0,1\}$ . Используя (4.1), вычислим энтропию этого алфавита:

$$H(A_2) = -P(0) \cdot \log_2(P(0)) - P(1) \cdot \log_2(P(1)). \quad (4.3)$$

К примеру, полагая, что сообщение  $M$  состоит только из единиц ( $M = 11\dots 1$ ) и имеет длину  $n$ :  $M = \underbrace{111\dots 11}_n$ , вероятность того, что произвольный символ равен единице, составляет единицу ( $P(1) = 1$ ); другая вероятность –  $P(0) = 0$  для  $i = \overline{1, N}$ . Здесь имеет место использование моноалфавита: алфавита, состоящего из одного символа.

Если в этом случае подставить в (1.3) соответствующие значения, то получим, что энтропия моноалфавита равна 0 бит, количество информации в сообщении из единиц (либо из нулей) также составляет 0 бит. Этот практический результат поясняет физический смысл понятия информации в теории Шеннона: *информацией является лишь такое сообщение, которое снимает некоторую неопределенность, т. е. содержит новые для получателя данные*. Если известно, что сообщение будет состоять из набора одинаковых символов, то для получателя сообщения оно никакой неопределенности не содержит.

Если для бинарного алфавита вероятность появления в произвольном сообщении одного из этих символов стремится к нулю (или равна ему), то энтропия такого алфавита также будет стремиться к (или равняться) нулю (рисунок).



Качественная характеристика энтропии  
бинарного алфавита при различных значениях  
вероятностей появления символов в сообщении

Между этими точками значение функции (энтропии) должно пройти через максимум (так как энтропия не может быть отрицательной). Для нахождения этого максимального значения надо найти производную:

$$\frac{\partial H(A)}{\partial P(1) \cdot \partial P(0)} \Rightarrow \max H(A_2) = \begin{cases} P(0) = \frac{1}{2}; \\ P(1) = \frac{1}{2} \end{cases}$$

$$\max H(A_2) = -\frac{1}{2} \cdot \log_2 \frac{1}{2} - \frac{1}{2} \cdot \log_2 \frac{1}{2} = 1 \text{ бит.}$$

Таким образом, энтропия бинарного алфавита принимает максимальное значение, равное 1 бит, при условии равновероятного появления каждого символа алфавита в сообщении.

Интересным представляется оценка количества информации в сообщении, выполненная на основе различных подходов.

**! Пример.** Анализируем сообщение  $M = \text{'We are happy'}$  (пробелы не учитываем). Если принять, что энтропия английского алфавита, вычисленная по Шеннону, составляет 4,2 бит (примерно), то в анализируемом сообщении содержится 42 бита информации:  $I(M) = 4,2 \cdot 10 = 42$  бита.

С другой стороны, предполагая, что сообщение переведено в ASCII коды (один символ алфавита заменяется соответствующим байтом двоичных символов) и  $P(0) = P(1) = 0,5$ , получаем  $I(M_{\text{ASCII}}) = 1 \cdot 80 = 80$  бит.

Данный пример является свидетельством и подтверждением избыточности не только алфавита, но и сообщений, сформированных и обрабатываемых в компьютерных системах, т. е. любые сообщения характеризуются информационной избыточностью, что позволяет сжимать их без потери информации.

## **? Вопросы для контроля и самоконтроля**

1. Что такое «алфавит источника сообщения»?
2. Что такое «мощность алфавита источника сообщения»?
3. Какова мощность алфавита белорусского языка?
4. Какова мощность алфавита русского языка?
5. Какова мощность алфавита «компьютерного» языка?
6. Что такое «энтропия алфавита»?
7. От чего зависит энтропия алфавита?
8. Записать формулу для вычисления энтропии.
9. Что нужно знать для вычисления энтропии алфавита?
10. Как рассчитываются энтропия Шеннона и энтропия Хартли? В чем принципиальное различие между этими характеристиками? Дайте толкование физического смысла энтропии.
11. Что такое избыточность алфавита и избыточность сообщений, сформированных в компьютерных системах? Принцип действия каких систем основан на существовании данной избыточности?

12. Расположить в порядке возрастания энтропии известные Вам алфавиты.

13. Вычислить энтропию алфавита белорусского (русского) языка.

14. Вычислить энтропию Шеннона бинарного алфавита, если вероятность появления в произвольном документе на основе этого алфавита одного из символов составляет 0,25, другого – 0,75; либо 0 и 1; либо 0,5 и 0,5.

15. Чем отличается энтропия Шеннона от энтропии Хартли?

16. Чему равна энтропия алфавита по Хартли, если мощность этого алфавита: а) 1 символ; б) 2 символа; в) 8 символов?



# КОЛИЧЕСТВО ИНФОРМАЦИИ. ЭНТРОПИЙНАЯ ОЦЕНКА ПОТЕРЬ ПРИ ПЕРЕДАЧЕ ИНФОРМАЦИИ

5

## Количество информации в сообщении. Информационная избыточность сообщений.

Эти вопросы рассмотрены в [1].

## Потери информации в зашумленных каналах. Условная энтропия и ее использование для оценки потерь информации в двоичных каналах передачи

Пусть в ИсС сообщение  $X_k = x_1, x_2, \dots, x_i, \dots, x_k$  на входе канала формируется на основе алфавита  $A = \{a_i\}, i = 1 \dots N$ , где  $N$  – мощность алфавита.

Сообщение на выходе канала ( $Y_k = y_1, y_2, \dots, y_j, \dots, y_k$ ) формируется на основе того же алфавита.

Если при передаче сообщений в двоичном канале с одинаковой вероятностью ( $p$ ) появляются ошибки типа  $0 \rightarrow 1$  (передан ноль, получена 1) либо  $1 \rightarrow 0$ , то такой канал, как мы определили выше, называют *двоичным симметричным*.

Здесь нужно вспомнить из теории вероятностей: запись  $P(A | B)$  – *вероятность гипотезы  $A$  при наступлении события  $B$  (апостериорная вероятность)*. В первом из вышеуказанных случаев появления ошибки можем формально записать:  $p = P(x_i = 0 | y_j = 1)$ , или более кратко:  $p = P(0 | 1)$ ; во втором случае:  $p = P(x_i = 1 | y_j = 0)$ , или более кратко:  $p = P(1 | 0)$ . Обозначим символом  $q$  вероятность правильной передачи двоичного символа:  $q = P(0 | 0) = P(1 | 1)$ . Понятно, что  $p + q = 1$ .

**Задача:** определить количественно потери информации, вызванные несовершенством ИС (канала), т. е. при  $p > 0$ . Задача относится к области *проверки гипотез и принятия статистических решений*. Математической основой ее решения является *теорема Байеса*: совместная вероятность случайных событий  $A$  и  $B$ :

$$P(A, B) = P(A | B) P(B) = P(B | A) P(A) \quad (5.1)$$

или

$$P(A | B) = P(B | A) P(A) / P(B). \quad (5.2)$$

В соответствии с (5.2) для ДСК можно записать (используя дискретную форму теоремы Байеса):

$$P(x_i | y_j) = P(y_j | x_i) P(x_i) / P(y_j), \quad (5.3)$$

где

$$P(y_j) = \sum P(y_j | x_i) P(x_i). \quad (5.4)$$

В общем случае  $i$  и  $j$  могут принимать различные значения в пределах от 1 до  $N$ . В соответствии с (5.3) и (5.4) для ДСК можем записать:

$$P(x_i = 0 | y_j = 0) = [P(y_j = 0 | x_i = 0) \cdot P(x_i = 0)] / [P(y_j = 0 | x_i = 0) \times \\ \times P(x_i = 0) + P(y_j = 0 | x_i = 1) \cdot P(x_i = 1)];$$

$$P(x_i = 1 | y_j = 0) = [P(y_j = 0 | x_i = 1) \cdot P(x_i = 1)] / [P(y_j = 0 | x_i = 0) \times \\ \times P(x_i = 0) + P(y_j = 1 | x_i = 1) \cdot P(x_i = 1)];$$

$$P(x_i = 0 | y_j = 1) = [P(y_j = 1 | x_i = 0) \cdot P(x_i = 0)] / [P(y_j = 1 | x_i = 0) \times \\ \times P(x_i = 0) + P(y_j = 1 | x_i = 1) \cdot P(x_i = 1)];$$

$$P(x_i = 1 | y_j = 1) = [P(y_j = 1 | x_i = 1) \cdot P(x_i = 1)] / [P(y_j = 1 | x_i = 0) \times \\ \times P(x_i = 0) + P(y_j = 1 | x_i = 1) \cdot P(x_i = 1)].$$

Если  $p > 0$ , то это можно трактовать как *неоднозначность* (по Шеннону – equivocation) между переданным и принятым сообщениями. Эта неоднозначность определяется как **условная энтропия** сообщения ( $X$ ), обусловленная полученным сообщением ( $y_j$ ):

$$H(X | y_j) = - \sum_{i=1} P(x_i | y_j) \cdot \log P(x_i | y_j). \quad (5.5)$$

В (5.5) и везде ниже логарифмирование ведется по основанию 2.

В соответствии с (5.5) можно определить, какому количеству информации соответствует один символ сообщения  $X$ , если на выходе канала получен 0:

$$H(X | y_j = 0) = - P(x_i = 0 | y_j = 0) \cdot \log P(x_i = 0 | y_j = 0) - \\ - P(x_i = 1 | y_j = 0) \cdot \log P(x_i = 1 | y_j = 0) = - q \cdot \log q - p \cdot \log p.$$

То же, если на выходе получена 1:

$$\begin{aligned} H(X | y_j = 1) &= -P(x_i = 0 | y_j = 1) \cdot \log P(x_i = 0 | y_j = 1) - \\ &- P(x_i = 1 | y_j = 1) \cdot \log P(x_i = 1 | y_j = 1) = -p \cdot \log p - q \cdot \log q. \end{aligned}$$

Условной энтропией *источника дискретного сообщения*  $X$  называется величина

$$\begin{aligned} H(X | Y) &= P(y_j = 0) \cdot H(X | y_j = 0) + P(y_j = 1) \cdot H(X | y_j = 1) = \\ &= -p \log p - q \log q. \end{aligned} \quad (5.6)$$

$H(X | Y)$  означает среднее количество информации для входного символа относительно полученного сообщения  $Y$  или *потерю информации на каждый символ переданного сообщения*.

**! Пример.** Пусть известно, что  $P(x = 0) = P(x = 1) = 0,5$  и  $p = 0,01$ . Нужно определить потерю информации на каждый символ переданного сообщения.

Из (5.6) определим:

$$\begin{aligned} H(X | Y) &= -p \cdot \log p - q \cdot \log q = \\ &= 0,01 \cdot \log 0,01 - 0,99 \cdot \log 0,99 = 0,081 \text{ бит.} \end{aligned}$$

Это означает, что при указанных параметрах канала и источника сообщения при передаче каждого двоичного символа будет потеряно 0,081 бит информации.

Вспомним, что основанием логарифма является число 2.

К. Шеннон показал, что *эффективная информация* на выходе канала относительно входной в расчете на 1 символ (эффективная энтропия алфавита) составляет:

$$H_e = H(X) - H(X | Y). \quad (5.7)$$

Для случая из примера  $H_e = 0,919$  бит.

Если вероятность ошибки  $p = 0$ , то  $H(X | Y) = 0$  и *потерь информации при передаче нет*.

### **? Вопросы для контроля и самоконтроля**

1. Что такое «количество информации»? В каких единицах оно выражается?
2. Записать формулу для подсчета количества информации.
3. Вычислить количество информации в сообщении, состоящем из Ваших фамилии и имени.

4. Какое количество информации содержится в сообщении «information», если принять, что энтропия алфавита составляет 4,7 бит?
5. В чем сущность «информационной избыточности» сообщений?
6. Чем вызваны потери информации в каналах передачи?
7. Что характеризует «условная энтропия»?
8. Записать формулу для вычисления условной энтропии сообщения  $X_k$ , обусловленной полученным сообщением  $Y_k$ .
9. Известно, что  $P(x = 0) = 0,2$ ,  $P(x = 1) = 0,8$  и вероятность ошибки при передаче ( $p$ ) составляет 0,01. Определить потерю информации на каждый символ переданного сообщения.
10. Известно, что  $P(x = 0) = 0,2$ ,  $P(x = 1) = 0,8$  и вероятность безошибочной передаче ( $q$ ) составляет 0,01. Определить потерю информации на каждый символ переданного сообщения.
11. Для предыдущего условия определить количественно потерю информации в двоичном сообщении из 100 символов.
12. Какое количество информации будет передано по каналу связи за 1 час при скорости передачи 1 Мбит/с, если вероятность ошибки равна 0,5?
13. Что такое «эффективная информация», «эффективная энтропия»?
14. Определить эффективную энтропию алфавита для задач 9 и 10.

# МЕТОДЫ СТРУКТУРНОЙ, ИНФОРМАЦИОННОЙ И ВРЕМЕННОЙ ИЗБЫТОЧНОСТИ В ИВС

6

## 6.1. Общая характеристика ИС с избыточностью

Известно, что любое сообщение характеризуется информационной избыточностью. Для ее удаления из сообщения применяются различные методы сжатия (об этом речь пойдет в теме 7).

Однако при проектировании ИС, как правило, создатели систем сознательно идут по противоположному пути (хотя зачастую используют и то, и другое): «вводят» в систему избыточность искусственно. Основная цель такого шага – повышение *надежности* (функциональной надежности) ИС.

Вопросам обеспечения надежности уделяется внимание на всех этапах *жизненного цикла* (проектирование, изготовление, эксплуатация) устройств и каналов передачи информации. Мировой опыт показывает, что значительный эффект при решении задач обеспечения качества и надежности дает системная организация работ на основе внедрения международных стандартов серии ISO 9000 или TQM (*Total Quality Management*), разработанных Международной организацией по стандартизации (*International Standard Organizatin, ISO*).

Методы *структурной, информационной и временной* избыточности на необходимом для понимания уровне изложены в [1]. К числу этих методов относится и помехоустойчивое кодирование, разработка и практическая реализация которых относятся к *прикладной теории кодирования*, о которой мы упоминали в теме 3.

Для лучшего понимания материала приведем общую классификацию кодов.

*Блочные коды* – каждому сообщению из  $k$  ( $X_k$ ) символов (бит) сопоставляется блок нового сообщения (кодového слова) из  $n$  символов (кодový вектор  $X_n$  длиной  $n = k + r$ ), где  $k$  и  $r$  – длина соответственно информационного и проверочного слов.

*Непрерывные (рекуррентные, цепные, сверточные) коды* – непрерывная последовательность символов, не разделяемая

на блоки. Передаваемая последовательность образуется путем размещения в определенном порядке проверочных символов между информационными символами исходной последовательности.

*Систематические коды* характеризуются тем, что сумма по модулю 2 двух разрешенных кодовых комбинаций кодов снова дает разрешенную кодовую комбинацию.

*Несистематические коды* не обладают отмеченными выше свойствами (к ним относятся *итеративные коды*).

*Линейные коды* – проверочные (избыточные) символы вычисляются как *линейная комбинация информационных символов*; для кодов принимается обозначение  $[n, k]$ -код.

*Циклические коды* – относятся к линейным систематическим.

*Нелинейные коды* являются противоположностью линейным.

*Линейные блочные коды* – это класс кодов с контролем четности, которые можно описать парой чисел  $(n, k)$ .

Первое из чисел определяет длину кодового слова ( $X_n$ ), второе – длину информационного слова ( $X_k$ ). Отношение числа бит данных к общему числу бит данных  $k/n$  именуется *степенью кодирования* (code rate) – доля кода, которая приходится на полезную информацию.

Для формирования проверочных символов (кодирования) используется порождающая матрица. Совокупность базисных векторов будем далее записывать в виде матрицы  $G$  размерностью  $k \times n$  с единичной подматрицей ( $I$ ) в первых  $k$  строках и столбцах:

$$G = [P | I]. \quad (6.1)$$

Матрица  $G$  называется *порождающей* матрицей линейного корректирующего кода в приведенно-ступенчатой форме. Кодовые слова являются линейными комбинациями строк матрицы  $G$  (кроме слова, состоящего из нулевых символов). Кодирование, результатом которого является кодовое слово  $X_n$ , заключается в умножении вектора сообщения длиной  $k$  ( $X_k$ ) на порождающую матрицу по правилам матричного умножения (все операции выполняются по модулю 2). Очевидно, что при этом первые  $k$  символы кодового слова равны соответствующим символам сообщения, а последние  $r$  символов ( $X_r$ ) образуются как *линейные комбинации* первых.

Для всякой порождающей матрицы  $G$  существует матрица  $H$  размерности  $r \times n$ , задающая базис нулевого пространства кода и удовлетворяющая равенству

$$G \cdot H^T = 0. \quad (6.2)$$

Матрица  $H$ , называемая *проверочной*, может быть представлена так:

$$H = \left[ -P^T \mid I \right]. \quad (6.3)$$

В последнем выражении  $I$  – единичная матрица порядка  $r$ .

Кодовое слово  $X_n$  может быть получено на основе следующего тождества:

$$H \cdot (X_n)^T = 0. \quad (6.4)$$

Результат умножения сообщения ( $Y_n$ ) на *транспонированную* проверочную матрицу ( $H$ ) называется *синдромом* (вектором ошибки)  $S$ :

$$S = (Y_n)^T \cdot H, \quad (6.5)$$

где  $Y_n = y_1, y_2, \dots, y_n$ . Слово  $Y_n$  обычно представляют в следующем виде:

$$Y_n = X_n \oplus E, \quad (6.6)$$

где  $E = e_1, e_2, \dots, e_n$  – вектор ошибки.

Если все  $r$  символов синдрома нулевые ( $S = 0$ ), то принимается решение об отсутствии ошибок в принятом сообщении, в противном случае – об их наличии.

*Минимальным кодовым расстоянием*  $d_{\min}$  между двумя кодовыми словами  $X$  и  $Y$  является число, соответствующее числу позиций, в которых эти слова отличаются.

**⚠ Пример.** Если  $X = 1011$  и  $Y = 1001$ , то  $d_{\min}(X, Y) = 2$ , так как 2-й и 3-й разряды в этих словах разные.

В общем случае код, характеризующийся *минимальным кодовым расстоянием*  $d_{\min}$  между двумя произвольными кодовыми словами, позволяет обнаруживать  $t_0$  ошибок, где  $t_0 = \frac{d}{2}$ , если  $d$  – четно, и  $t_0 = \frac{d-1}{2}$ , если  $d$  – нечетно.

Количество исправляемых кодом ошибок ( $t_u$ ) определяется следующим образом:

$$t_u = \begin{cases} \frac{d-1}{2}, & d - \text{нечетное}, \\ \frac{d-2}{2}, & d - \text{четное}. \end{cases} \quad (6.7)$$

**Избыточный код простой четности.** Простейший избыточный код основан на контроле четности (либо нечетности) единичных символов в сообщении. Количество избыточных символов  $r$  всегда равно 1 и не зависит от  $k$ . Значение этого символа будет нулевым, если сумма всех символов кодового слова по модулю 2 равна нулю.

Назначение  $X_r$  в данном алгоритме – обнаружение ошибки. Код простой четности позволяет обнаруживать все нечетные ошибки (нечетное число ошибок), но не позволяет их исправить. Нетрудно убедиться, что данный код характеризуется минимальным кодовым расстоянием, равным 2.

**! Пример.** Пусть информационная последовательность будет  $X_k = 10101$ , тогда  $X_r = \sum_{i=1}^n X_i = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 1$ .

Проверочная матрица для данного случая будет состоять из одной строки и шести столбцов и примет следующий вид:

$$H = 111111.$$

Кодовое слово  $X_n$ , вычисленное в соответствии с (6.4), будет равно  $101011$ . Как видим,  $w(X_n)$  имеет четное значение.

Слово  $X_n$  будет передаваться от ИсС к ПС. Пусть на приемной стороне имеем  $Y_n = 1\underline{1}1011$ :  $Y_k = 1\underline{1}101$  и  $Y_r = 1$  (ошибочный символ подчеркнут).

Для определения синдрома ошибки в соответствии с (6.5) достаточно выполнить следующие простые действия:

а) вычислить дополнительное слово (в данном случае – символ)  $Y'_r$ , которое является сверткой по модулю 2 слова  $Y_k$ :  $Y'_r = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$ ;

б) найти синдром  $S = Y_r \oplus Y'_r = 1 \oplus 0 = 1$ . Неравенство синдрома нулю означает, что получено сообщение с ошибкой (или с ошибками).

**Код Хэмминга.** Данный код характеризуется минимальным кодовым расстоянием  $d_{\min} = 3$ . При его использовании кодирование сообщения также должно удовлетворять соотношению (6.4). Причем вес столбцов подматрицы  $A$  должен быть больше либо равен 2. Второй особенностью данного кода является то, что используется расширенный контроль четности групп символов информационного слова, т. е.  $r > 1$ . Для упрощенного вычисления  $r$  можно воспользоваться следующим простым соотношением:

$$r = \log_2 k + 1. \quad (6.8)$$



В сравнении с предыдущим кодом данный позволяет не только обнаруживать, но и исправлять одиночную ошибку в кодовом слове (см. (3.7)).

В нашем случае подматрицу  $A$  можно определить как

$$A = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ h_{r1} & \cdots & \cdots & h_{rk} \end{bmatrix}. \quad (6.9)$$

Элемент этой подматрицы (0 или 1)  $h_{ij}$  относится к  $i$ -й строке и  $j$ -му столбцу ( $i = 1, r$   $j = 1, k$ ).

Вычислим проверочные символы в соответствии с (6.4):

$$x_{ri} = \sum_{ij}^{rk} h_{ij} \cdot x_k \mod 2. \quad (6.10)$$

Определим синдром:

$$y_{ri} = \sum_{ij}^{rk} h_{ij} \cdot y_k \mod 2, \quad S = y_{ri} \oplus y'_{ri}. \quad (6.11)$$

**⚠ Пример.** Имеется информационное слово  $X_k = 1001$ . Проанализируем использование рассматриваемого кода.

Для начала отмечаем, что  $k = 4$ . В соответствии с (6.8) подсчитываем длину избыточного слова:  $r \geq \log_2(4 + 1) = 3$ , тогда  $n = k + r = 7$ .

Создаем проверочную матрицу  $H_{7,4}$ :

$$\underbrace{H_{7,4}}_{n \cdot k} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \underbrace{0 & 1 & 1 & 1}_A & \underbrace{1 & 0 & 0}_I \\ \underbrace{1 & 0 & 1 & 1}_A & \underbrace{0 & 1 & 0}_I \\ \underbrace{1 & 1 & 0 & 1}_A & \underbrace{0 & 0 & 1}_I \end{bmatrix}.$$

Вычисляем проверочные символы, используя (6.10).

В соответствии с этим первый проверочный символ  $x_{r1}$  будет равен 1, остальные – нулю:

$$x_{r1} = h_{11} \cdot x_1 \oplus h_{12} \cdot x_2 \oplus \dots \oplus h_{14} \cdot x_4 = 0 \cdot 1 \oplus 1 \cdot 0 \oplus 1 \cdot 0 \oplus 1 \cdot 1 = 1;$$

$$x_{r2} = h_{21} \cdot x_1 \oplus h_{22} \cdot x_2 \oplus \dots \oplus h_{24} \cdot x_4 = 1 \cdot 1 \oplus 0 \cdot 0 \oplus 1 \cdot 0 \oplus 1 \cdot 1 = 0;$$

$$x_{r3} = h_{31} \cdot x_1 \oplus h_{32} \cdot x_2 \oplus \dots \oplus h_{34} \cdot x_4 = 1 \cdot 1 \oplus 1 \cdot 0 \oplus 0 \cdot 0 \oplus 1 \cdot 1 = 0.$$

Таким образом, избыточное слово будет таким:  $X_r = 100$ , а кодовое слово –  $X_n = 1001\ 100$ .

Рассмотрим ситуацию, когда ошибок в переданной информации нет,  $t = 0$ , т. е.  $X_n = Y_n = 1001\ 100$ .

Вычислим новый набор проверочных символов в соответствии с (6.11) и синдром:

$$Y'_r = 100;$$

$$S = Y_r \oplus Y'_r = 100 \oplus 100 = 000 \equiv 0.$$

Нулевой синдром означает безошибочную передачу (или прием) информации.

Рассмотрим ситуацию, когда возникает одиночная ошибка,  $t = 1$ .

Пусть ошибка произошла в служебных символах  $Y_n = 1001\underline{1}00$  (ошибочный символ подчеркнут).

Синдром вычисляем по методике, приведенной для случая отсутствия ошибок. Получаем  $S = 100$ . Вес синдрома равен 1 и это означает, что произошла ошибка. Местоположение ошибки выявляется анализом (декодированием) синдрома. Декодирование опирается на вышеприведенное соотношение (6.5), в соответствии с которым, принимая во внимание (6.6), можем записать:

$$H \cdot (Y_n)^T = H \cdot (X_n \oplus E)^T = H \cdot (X_n)^T \oplus H \cdot (E)^T = 0 \oplus h_5, \quad (6.12)$$

где  $h_5$  – пятый столбец матрицы, номер которого соответствует номеру ошибочного символа в принятом кодовом слове. Действительно,  $h_5 = S = 100$ .

В результате декодирования синдрома получается вектор ошибки (*унарный вектор*, имеющий единичный вес):  $E = 0000100$ . Исправление ошибочного бита достигается простым сложением по модулю 2 вектора  $E$  и кодового слова  $Y_n$ :

$$Y_n = E \oplus Y_n = 0000100 \oplus 1001000 = 1001100.$$

Пусть ошибка произошла в бите информационного слова  $Y_n = \underline{0}001100$ .

Вычислим дополнительные проверочные символы и синдром:

$$Y'_r = 111;$$

$$S = Y_r \oplus Y'_r = 011.$$

Убеждаемся, что синдром соответствует первому столбцу используемой проверочной матрицы. Это означает, что декоди-

рование синдрома однозначно укажет на местоположение ошибочного бита:  $E = 1000000$  и  $Y'_n = E \oplus Y_n = 0001100 \oplus 1000000 = 1001100 \equiv X_n$ .

При возникновении количества ошибок, кратного двум (например, на позициях  $l$  и  $m$ ), данный код не позволяет однозначно идентифицировать ошибки, поскольку с учетом (6.12) имеем

$$S = h_l \oplus h_m. \quad (6.13)$$

Таким образом, код Хемминга с  $d_{\min} = 3$  *гарантированно обнаруживает и исправляет* одиночную ошибку в любом разряде кодового слова.

Порядок следования вектор-столбцов в матрице  $A$  не имеет значения, однако важно, чтобы на передающей и на принимающей сторонах используемые матрицы были бы абсолютно идентичны.

**Модифицированный код Хемминга.** Этот код характеризуется минимальным кодовым расстоянием, равным 4, и позволяет *обнаруживать две ошибки и исправлять одну*.

Алгоритм использования кода такой же, как и для  $d_{\min} = 3$ . Принципиальное отличие состоит в виде проверочной матрицы, которая при неизменной длине информационного слова имеет одну дополнительную строку и один дополнительный столбец (в подматрице  $I$ ):

$$H' = \left| \begin{array}{cccc|c} & & & & 0 \\ & & & & 0 \\ & & & & \vdots \\ & & & & 0 \\ \hline 1 & 1 & \dots & 1 & 1 \end{array} \right|. \quad (6.14)$$

Запись матрицы в таком виде не считается канонической, так как подматрица  $I$  не является единичной диагональной матрицей. Для преобразования такой записи к каноническому виду воспользуемся свойствами линейного кода: в дополнительную строку необходимо записать сумму по модулю 2 соответствующих символов матрицы кода с  $d_{\min} = 3$ .

**⚠ Пример.** Пусть  $k = 6$  для случая  $d_{\min} = 3$  и  $r = \log_2 k + 1 = 4$ ,  $n = 10$ .

При этих параметрах матрица  $H_{10,6}$  может иметь следующий вид:

$$H_{10,6} = \begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{vmatrix}.$$

Стоит задача преобразования такой матрицы для случая  $d_{\min} = 4$ . Дополним ее строкой, состоящей из единиц, и одним дополнительным столбцом:

$$H_{11,6} = \begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{vmatrix}.$$

Для преобразования такой матрицы к каноническому виду, используя свойство линейности кода, надо сложить по модулю 2 символы соответствующих столбцов последней матрицы и записать полученный результат вместо последней строки. Тогда имеем:

$$H_{11,6} = \begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{vmatrix}.$$

В общем случае любая проверочная матрица кода Хемминга с  $d_{\min} = 4$  имеет нечетный вес столбцов, т. е. вес любого из столбцов подматрицы  $A'$  может быть равен 3, 5, 7, ... .

Как и в предыдущем случае (при  $d_{\min} = 3$ ), равенство нулю синдрома означает отсутствие ошибок. Если же синдром не равен нулю и имеет нечетный вес, то это говорит о том, что произошла одиночная ошибка. Если же синдром не равен нулю и его вес четный, то произошла двойная ошибка, так как вес суммы любых двух столбцов всегда четный (см. (6.12)).

Вспомним, что повышение надежности достигается аппаратно-временной и информационной избыточностью, вводимой в исходную систему или устройство для обнаружения и исправления (либо только для обнаружения) ошибок, возникающих при передаче или

хранении данных. Определение рассматриваемых методов как «помехоустойчивые» означает, прежде всего, что они являются противодействием помехам, действующим на систему и приводящим к ошибкам в данных.

Информационная система, показанная на рисунке в разделе 3, в последнем случае может быть представлена в виде структурной схемы (рис. 6.1).

Суть метода избыточного (помехоустойчивого) кодирования состоит в преобразовании исходного информационного сообщения  $X_k$  ( $k$  – длина сообщения), называемого также *информационным словом*. К слову  $X_k$  дополнительно присоединяют (наиболее часто – по принципу конкатенации) избыточные символы длиной  $r$  бит, составляющие *избыточное слово*  $X_r$ . Таким образом формируют *кодированное слово*  $X_n$  длиной  $n = k + r$  двоичных символов:  $X_n = X_k X_r$ . Информацию содержит только информационное слово. Назначение избыточности  $X_r$  – обнаружение и исправление ошибок.

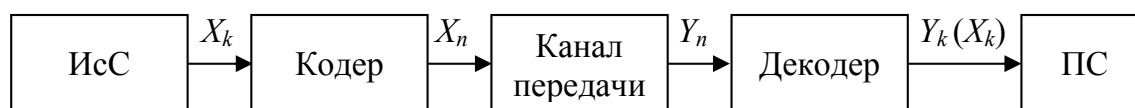


Рис. 6.1. Обобщенная структурная схема информационной системы (системы передачи информации), содержащей средства обнаружения и коррекции ошибок

В зависимости от принципа вычисления дополнительных символов и их числа реализуются различные алгоритмы помехоустойчивого кодирования. Общим является то, что избыточное слово  $X_r$  генерируется на передающей стороне и используется принимающей для обнаружения и/или исправления ошибок.

Декодер работает на принципе сравнения принятого дополнительного символа(-ов) и вычисленного на принимающей стороне. Если в принятом сообщении нет ошибок, то слово  $Y_n$  точно соответствует слову  $X_n$ , однако если произошла ошибка, то  $Y_n$  обязательно отличается от  $X_n$ . Задачей декодера является обнаружение факта появления ошибки в принятом сообщении либо обнаружение и исправление коррекции.

Ниже рассмотрим другой класс известных кодов – циклических.

Основные свойства *циклических кодов* (ЦК):

- относятся к классу линейных, систематических;
- сумма по модулю 2 двух разрешенных кодовых комбинаций дает также разрешенную кодовую комбинацию;

- каждый вектор (кодированное слово), получаемый из исходного кодированного вектора путем циклической перестановки его символов, также является разрешенным кодированным вектором;

- при циклической перестановке символы кодированного слова перемещаются слева направо на одну позицию.

**! Пример.** Если кодированное слово имеет следующий вид: 1101100, то разрешенной кодированной комбинацией будет и такая: 0110110.

Принято описывать ЦК при помощи порождающих полиномов  $G(X)$  степени  $r = n - k$ , где  $r$  – число проверочных символов в кодированном слове.

**! Пример.** Переведем кодированное слово  $X_n = 101100$  в полиномиальный вид:

$$B_i(X) = 1 \cdot X^5 + 0 \cdot X^4 + 1 \cdot X^3 + 1 \cdot X^2 + 0 \cdot X^1 + 0 \cdot X^0 = X^5 + X^3 + X^2.$$

Операции кодирования и декодирования ЦК сводятся к известным процедурам умножения и деления полиномов. Действия с кодированными словами в виде полиномов производятся по правилам арифметики по модулю 2 (*вычитание равносильно сложению*).

Из равенства  $X^n - 1 = 0$  получаем  $X^n = 1$ . Прибавив к левой и правой частям по единице, имеем  $X^n + 1 = 1 + 1 = 0$ . Таким образом, вместо двучлена  $X^n - 1$  можно ввести бином  $X^n + 1$  или  $1 + X^n$ , из чего следует, что  $X^n + X^n = X^n (1 + 1) = 0$ .

Приведем далее порядок суммирования (вычитания), умножения и деления полиномов (по модулю 2). В примерах используем вышеприведенные кодированные комбинации:  $A_1(X) = X^5 + X^3 + X^2$  (101100) и  $A_2(X) = X^4 + X^2 + X$  (10110).

*Суммирование (вычитание):*

$$\begin{aligned} A_1(X) + A_2(X) &= A_1(X) - A_2(X) = X^5 + X^4 + X^3 + \underline{X^2} + \underline{X^2} + X = \\ &= X^5 + X^4 + X^3 + X \end{aligned}$$

$$101100$$

или  $\oplus$  10110

$$111010 = X^5 + X^4 + X^3 + X.$$

Помним, что  $X^2 + X^2 = 0$ .

*Умножение:*

$$\begin{aligned} A_1(X) \cdot A_2(X) &= (X^5 + X^3 + X^2) \cdot (X^4 + X^2 + X) = X^9 + \underline{X^7} + X^6 + \underline{X^7} + X^5 + \\ &+ \underline{X^4} + X^6 + \underline{X^4} + X^3 = X^9 + X^5 + X^3 = 1000101000. \end{aligned}$$

*Деление:*

$$\begin{array}{r} X^5 + X^3 + X^2 \quad | \quad X^4 + X^2 + X \\ \underline{X^5 + X^3 + X^2} \quad | \quad X \\ 0 \quad 0 \quad 0 \end{array}$$

остаток при делении – 000 или просто 0 ( $R(X) = 0$ ).

**Следует запомнить:** при циклическом сдвиге вправо на один разряд необходимо исходную кодовую комбинацию поделить на  $X$ , а умножение на  $X$  эквивалентно сдвигу влево на один символ.

**Порождающие полиномы циклических кодов.** Формирование разрешенных кодовых комбинаций ЦК  $B_j(X)$  основано на предварительном выборе порождающего (образующего) полинома  $G(X)$ , который обладает важным отличительным признаком: все комбинации  $B_j(X)$  делятся на порождающий полином  $G(X)$  без остатка:

$$B_j(X) / G(X) = A_j(X), \quad (6.15)$$

здесь  $B_j(X) = X_n$  – кодовое слово,  $A_j(X) = X_k$  – информационное слово.

Степень порождающего полинома определяет число проверочных символов:  $r = n - k$ . Из этого свойства следует простой способ формирования разрешенных кодовых слов ЦК – умножение информационного слова на порождающий полином  $G(X)$ :

$$B(X) = A(X)G(X). \quad (6.16)$$

Порождающими могут быть только такие полиномы, которые являются делителями двучлена (бинома)  $X^n + 1$ :

$$(X^n + 1)/G(X) = H(X) \quad (6.17)$$

при нулевом остатке:  $R(X) = 0$ .

С увеличением максимальной степени порождающих полиномов  $r$  резко увеличивается их количество: при  $r = 3$  имеется всего два полинома, а при  $r = 10$  их уже несколько десятков. В таблице приведены некоторые коды.

Два варианта порождающих полиномов кода Хемминга (7, 4), с записью по модулю 2 в виде 1101 и 1011, представляют собой так называемые двойственные многочлены (полиномы): весовые коэффициенты одного полинома, зачитываемые слева направо, становятся весовыми коэффициентами двойственного полинома при считывании их справа налево.

### Некоторые из известных кодов, описываемые с помощью полиномов

Степень полинома, $r$	Полином $G(X)$	Двоичное представление полинома	$n$	$k$	Примечание
1	$X + 1$	11	3	2	Код с проверкой на четность
2	$X^2 + X + 1$	111	3	1	Код с повторением
3	$X^3 + X^2 + 1$ $X^3 + X + 1$	1101 1011	7	4	Классический код Хемминга (7, 4)
4	$X^4 + X^3 + 1$	11001	15	11	Классический код Хемминга (15, 11)
	$X^4 + X + 1$	10011	15	11	Классический код Хемминга (15, 11)
	$X^4 + X^2 + X + 1$	10111	7	3	Коды Файра – Абрамсона
	$X^4 + X^3 + X^2 + 1$	11101	7	3	Коды Файра – Абрамсона
5	$X^5 + X^2 + 1$ $X^5 + X^3 + 1$	100101 101001	31	26	Классический код Хемминга (31, 26)

Порождающие полиномы кода Хемминга (7, 4) являются не только двойственными, но и *неприводимыми*.

*Неприводимые полиномы* не делятся ни на какой другой полином степени меньше  $r$ , поэтому их называют еще *неразложимыми*, *простыми* и *примитивными*. Например, порождающий полином  $G(X) = X^7 + 1$  раскладывается на три неприводимых полинома:

$$X^7 + 1 = (X + 1)(X^3 + X^2 + 1)(X^3 + X + 1) = G_1(X) \cdot G_2(X) \cdot G_3(X),$$

каждый из которых является порождающим для следующих кодов:

$G_1(X) = X + 1$  – код с проверкой на четность, КПЧ (7, 6);

$G_2(X) = X^3 + X^2 + 1$  – первый вариант кода Хемминга (7, 4);

$G_3(X) = X^3 + X + 1$  – двойственный  $G_2(X)$ , второй вариант кода Хемминга.

Различные вариации произведений  $G_{1,2,3}(X)$  дают возможность получить остальные порождающие полиномы:

код Абрамсона (7, 3):  $G_4(X) = G_1(X)G_2(X) = (X + 1)(X^3 + X^2 + 1) = X^4 + X^2 + X + 1$ ;

двойственный  $G_4(X)$ :  $G_5(X) = G_1(X)G_3(X) = (X + 1)(X^3 + X + 1) = X^4 + X^3 + X^2 + 1$ ;

код с повторением (7, 1):  $G_6(X) = G_2(X)G_3(X) = (X^3 + X^2 + 1)(X^3 + X + 1) = X^6 + X^5 + X^4 + X^3 + X^2 + X + 1$ .

*Порождающая матрица*  $G$  циклического кода имеет в качестве строк векторы  $G(x), xG(x), \dots, x^{k-1}G(x)$ :



$$G = \begin{matrix} G(x) & & g_0 g_1 g_2 \dots g_r 0 \dots 0 \\ xG(x) & = & 0 g_0 g_1 g_2 \dots g_r 0 \dots 0 \\ \dots & & \dots \\ x^{k-1}G(x) & & 0 \dots 0 g_0 g_1 g_2 \dots g_r, \end{matrix}$$

где  $g_0, \dots, g_r$  – коэффициенты генераторного полинома.

Проверочная матрица  $H$  кода строится на основе полинома (см. (6.3)):

$$H(X) = (X^n + 1) / G(X) \quad (6.18)$$

или

$$H = \begin{matrix} H(x) & & 0 \dots 0 h_k \dots h_2 h_1 h_0 \\ x H(x) & = & 0 \dots h_k h_{k-1} \dots h_1 h_0 0 \\ \dots & & \dots \\ X^{r-1} H(x) & & h_k \dots h_1 h_0 0 \dots 0, \end{matrix}$$

где  $h_j$  – коэффициенты полинома  $H(X)$ .

Справедливо

$$G \cdot H^T = 0 \text{ или } H \cdot G^T = 0, \quad (6.19)$$

здесь индекс « $T$ » означает транспонирование матрицы.

**⚠ Пример.** Задан ЦК (7, 4) дуальными порождающими полиномами  $G(7, 4) = X^3 + X + 1$  и  $\underline{G}(7, 4) = X^3 + X^2 + 1$ . Составить порождающие матрицы кодов.

Первой строкой в матрице записывается порождающий полином (в двоичном представлении) с умножением его на оператор сдвига  $X_r$  для резервирования места под запись трех ( $r = 3$ ) проверочных символов. Следующие  $(k - 1)$  строк матриц получаются путем последовательного циклического сдвига базового кодового слова матриц  $G$  и  $\underline{G}$  на одну позицию вправо:

$$\begin{matrix} & 1011000 & & 1101000 \\ G(7, 4) = & 0101100 & \underline{G}(7, 4) = & 0110100 \\ & 0010110 & & 0011010 \\ & 0001011 & & 0001101 \end{matrix}$$

Для построения порождающей матрицы, формирующей разделимый блочный код, необходимо матрицу преобразовать к каноническому виду путем линейных операций над строками.

Каноническая матрица должна в левой части порождающей ЦК матрицы содержать единичную диагональную квадратную подматрицу порядка  $k$  для получения в итоге блочного ЦК.

❗ **Пример.** Привести к каноническому виду матрицы из предыдущего примера.

С этой целью для получения первой строки канонической матрицы  $Gk(7, 4)$  необходимо сложить по модулю 2 строки с номерами 1, 3 и 4 матрицы  $G(7, 4)$ , а для матрицы  $\underline{Gk}(7, 4)$  – строки с номерами 1, 2 и 3 матрицы  $\underline{G}(7, 4)$ , оставшиеся строки – без изменений. В итоге имеем следующий вид первой из *дуальных канонических матриц*:

$$\begin{array}{rcl} & 1000 & 101 \quad (1 + 3 + 4) \\ Gk(7, 4) = & 0100 & 111 \quad (2 + 4) \\ & 0010 & 110 \quad (3 = 3) \\ & 0001 & 011 \quad (4 = 4) \end{array}$$

Запись  $(1 + 3 + 4)$  означает, что данная строка матрицы получена в результате суммы по модулю 2 первой, третьей и четвертой строк матрицы  $G(7, 4)$ .

Вторую матрицу построить самостоятельно.

Проверочная матрица  $H_{7,4}$  размерностью  $n \times r$  может быть получена из порождающей матрицы канонического вида путем дополнения проверочной подматрицы единичной матрицей размерности  $r \times r$ :

$$\begin{array}{c} 101 \\ 111 \\ H_{7,4} = 110 \\ \underline{011} \\ 100 \\ 010 \\ 001, \end{array}$$

или в каноническом виде:

$$\begin{array}{rcl} & 1110 & 100 \\ H_{7,4} = & 0111 & 010 \\ & 1101 & 001 . \end{array}$$

*Вычисление проверочных символов ( $X_r$ ) кодового слова ( $X_n$ ) чаще всего основывается на методе деления полиномов. Метод позволяет представить разрешенные к передаче кодовые комбинации в виде разделенных информационных  $X_k$  и проверочных  $X_r$  символов, т. е. получить *блочный код*.*

Поскольку число проверочных символов равно  $r$ , то для компактной их записи в последние младшие разряды кодового слова

надо предварительно к  $X_k$  (соответствует  $A_j(X)$  в формуле (5.8)) справа приписать  $r$  «нулей», что эквивалентно умножению  $X_k$  на оператор сдвига  $X_r$ . При этом имеется возможность представить кодовую комбинацию в виде последовательности информационных и проверочных символов:

$$X_n = X_k \cdot X_r \parallel R(X), \quad (6.20)$$

где  $R(X)$  – остаток от деления  $X_k \cdot X_r / G(X)$  (см. (5.9)).

В алгоритме на основе (6.20) можно выделить три этапа формирования разрешенных кодовых комбинаций в кодере:

1) к комбинации слова  $X_k$  дописывается справа  $r$  нулей, что эквивалентно умножению  $X_k$  на  $X_r$ ;

2) произведение  $X_k \cdot X_r$  делится на соответствующий порождающий полином  $G(X)$  и определяется остаток  $R(X)$ , степень которого не превышает  $(r - 1)$ , этот остаток и дает группу проверочных символов ( $X_r$ );

3) вычисленный остаток присоединяется справа к  $X_k$ .

**! Пример.** Рассмотрим процедуру кодирования для при  $X_k = 1001$ , т. е. сформируем кодовое слово циклического кода  $(7, 4)$ .

В заданном ЦК  $n = 7$ ,  $k = 4$ ,  $r = 3$ , выберем порождающий полином  $G(X) = X^3 + X + 1$  (код Хемминга).

$X_k = 1001 \sim X^3 + 1$ , (знак « $\sim$ » – тильда, означает соответствие).

1.  $X_k \cdot X_r = (X^3 + 1) \cdot X^3 = X^6 + X^3 \sim 1001000$ , ( $n = 7$ ).

$$\begin{array}{r} 2. X_k \cdot X_r / G(X) = X^6 + X^3 \quad \begin{array}{l} \underline{\phantom{X^6 + X^4 + X^3} \phantom{X^3 + X}} \\ X^3 + X + 1 \end{array} \\ \hline X^6 + X^4 + X^3 \quad \begin{array}{l} \phantom{X^6 + X^4 + X^3} \phantom{X^3 + X} \\ \phantom{X^6 + X^4 + X^3} \phantom{X^3 + X} \end{array} \\ \hline X^4 \\ \hline X^4 + X^2 + X \\ \hline X^2 + X \end{array}$$

$X^2 + X$  – остаток;  $R(X) = X^2 + X \sim 110$ .

3.  $X_n = X_k \cdot X_r \parallel R(X) = 1001110$  – итоговая комбинация ЦК (кодое слово).

**Синдромный метод декодирования ЦК.** Основная операция: принятое кодовое слово ( $X_n$ ) нужно поделить на порождающий полином. Если  $X_n$  принадлежит коду, т. е. не искажено помехами, то остаток от деления (синдром) будет нулевым. Ненулевой остаток свидетельствует о наличии ошибок в принятой кодовой комбинации. Для исправления ошибки нужно определить вектор (полином) ошибки (обозначим его  $E_n$ ).

После передачи по каналу с помехами принимается кодовое слово

$$Y_n = X_n + E_n, \quad (6.21)$$

здесь также сумма по модулю два.

При декодировании принятое кодовое слово делится на  $G(X)$ :

$$(Y_n) / (G(X)) = U, S_r, \quad (6.22)$$

$S_r$  – остаток от деления  $(Y_n) / (G(X))$  – синдром, а  $U$  – результат деления.

Всякому ненулевому синдрому соответствует определенное расположение (конфигурация) ошибок: *синдром для ЦК имеет те же свойства, что и для кода Хемминга* (используются при декодировании синдрома).

**! Пример.** Рассмотрим процедуру декодирования сообщения, сформированного в предыдущем примере. Пусть  $Y_n = 10\underline{1}1110$  (ошибочным является третий бит – подчеркнут).

Вспомним, что порождающая матрица имеет вид, показанный в примерах 3 и 4:

$$H_{7,4} = \begin{array}{cc} 1110 & 100 \\ 0111 & 010 \\ 1101 & 001. \end{array}$$

Для решения задачи последовательно выполняем следующие операции:

1. Выполняем деление:

$$\begin{array}{r} Y_n / G(X) = X^6 + X^4 + X^3 + X^2 + X \Big| \frac{X^3 + X + 1}{X^3} \\ \underline{X^6 + X^4 + X^3} \phantom{+ X^2 + X} \\ X^2 + X; \end{array}$$

$X^2 + X$  – остаток, т. е. синдром  $S_r = X^2 + X \sim 110$ .

2. Декодирование синдрома – позволяет определить местоположение ошибки: по полученному синдрому 110 в анализаторе синдрома (дешифраторе синдрома) определяем вид вектора  $E_n = 0010000$ .

Здесь определим внимание на важнейшую деталь: *синдром равен третьему вектор-столбцу в матрице  $H_{7,4}$ , поэтому единичный символ будет в третьем разряде вектора  $E_n$ .*

3. Исправление ошибки:  $Y_n + E_n = 10\underline{1}1110 + 0010000 = 10\underline{0}1110$ .

После исправления (исправленный бит подчеркнут двойной линией) получили такое же слово, которое было сформировано в источнике сообщения (см. п. 3 из предыдущего примера).



## **Вопросы для контроля и самоконтроля**

1. Что такое и для чего используются «структурная избыточность», «информационная избыточность» и «временная избыточность»?

2. В чем особенность перечисленных ниже кодов и в чем состоит отличие между кодами: а) блочным и непрерывным; б) систематическим и несистематическим; в) линейным и нелинейным?

3. К какому классу относятся код Хемминга; циклический код?

4. Что такое «расстояние Хемминга», «вес Хемминга», «информационное слово», «кодированное слово», «избыточное слово», «информационный символ», «проверочный символ»?

5. Какое минимальное или максимальное число избыточных символов может содержать кодированное слово, принадлежащее коду?

6. Вычислите число и значение проверочных символов для кода простой четности, если информационное слово имеет вид: 1100; 1110; 101010; 11110000.

7. Какое число проверочных символов имеет кодированное слово кода Хемминга с минимальным кодовым расстоянием 3, если информационное слово имеет вид: 1100; 1110; 101010; 11110000; 1010111; 1111111100?

8. Построить проверочную матрицу кода Хемминга с минимальным кодовым расстоянием 3 (4) для вычисления проверочных символов, если информационное слово имеет вид из задания 7.

9. Вычислить проверочные символы кодированного слова кода Хемминга с минимальным кодовым расстоянием 3 (4), если проверочная матрица  $H$  имеет вид:

$$H = \begin{pmatrix} 0111 & 1 \\ 1011 & 1 \\ 1101 & 1 \end{pmatrix},$$

а информационное слово имеет вид: 1100, 1101, 0000, 1111.

10. Вычислить синдром ошибки для случая из задания 9, если ошибка при передаче кодированного слова произошла: а) в первом бите этого слова; б) во втором; в) в пятом; г) в первом и в седьмом.

11. Как вычислить синдром ошибки?

12. Как определить вектор ошибки?

13. Какая конечная операция используется для исправления ошибки?

14. Чем характеризуется итеративный код?

15. Каково минимальное кодовое расстояние итеративного кода?

16. Какое число ошибок может исправить код с минимальным кодовым расстоянием: а) 3; б) 4; в) 7?

17. Вычислите минимальное количество и значения проверочных символов кодового слова итеративного кода, если информационное слово имеет вид: 1100, 0101, 101010101, 0000111100001111.

18. Если кодовое слово ЦК имеет следующий вид: 11011010, то будут ли разрешенными кодовыми комбинациями такие: 1101110, 01101011, 11011011?

19. Запишите в полиномиальном виде кодовое слово ЦК: 11011011, 1010101010, 0000000, 0000000001.

20. Чему равен результат операции по модулю 2 над полиномами  $(X^6 + X^3 + X^2)$  и  $(X^4 + X^2 + X)$ , если такой операцией будет: сложение, вычитание, деление, умножение? Записать результат операции в двоичной форме.

21. Определить число и вычислить значения проверочных символов ЦК, если код задается полиномом  $X^3 + X^2 + 1$ , а информационное слово имеет вид: 1010, 1100.

22. Чему будет равен синдром ошибки, возникшей при передаче сообщения для условий из задачи 21, если ошибка произошла: а) в первом бите кодового слова; б) во втором; в) в пятом; г) в первом и в седьмом?

## **6.2. Методы и средства перемежения данных. Использование перемежителей/деперемежителей в системах передачи данных**

*Перемежителем* называют такое устройство (реализовано аппаратно) или программное средство, которое определенным образом перемешивает (меняет местами) символы передаваемого сообщения (или кодового слова).

Основная причина разработки и использования перемежителей – желание разнести расположенные рядом (сгруппированные) ошибки в сообщении («размазать» ошибки по сообщению) с целью упрощения и сокращения во времени процедуры исправления таких ошибок сравнительно простыми кодами.

На рис. 6.2 показана структурная схема ИС, в которой наряду с помехоустойчивым кодом используется перемежение/деперемежение символов передаваемого сообщения.

Выполнение операции декодирования в два этапа позволяет почти полностью избавиться от влияния помех. На первом этапе производится преобразование групп (пакетов) ошибок в группу случайных (обычно одиночных) ошибок. На втором этапе сигнал обрабатывается с помощью классических методов борьбы со случайными ошибками (линейные итеративные коды, сверточные коды, турбо-коды), что должно приводить к полной коррекции ошибок.



Рис. 6.2. Структурная схема ИС с перемежителем/деперемежителем

Процедура перемежения/деперемежения состоит в перестановке символов кодового слова и восстановлении исходной последовательности после передачи ее по каналу. При перемежении обеспечивается преобразование бит входной последовательности в выходную последовательность без изменения ее длины. Однако чем больше *глубина перемежения* (минимальное расстояние – в битах, на которое разносятся соседние символы входной последовательности), тем больше задержка.

В общем случае выбор глубины перемежения зависит от двух факторов. С одной стороны, чем больше расстояние между соседними символами, тем большей длины пакет ошибок может быть исправлен. С другой стороны, чем больше глубина перемежения, тем сложнее аппаратно-программная реализация оборудования и больше задержка сигнала.

В литературных источниках для сравнения перемежителей используют следующие характеристики: глубина перемежения; время перемежения; *рандомизация* бит (местоположение любого бита выходной последовательности должно отличаться от его

местоположения в исходной последовательности). Однако основной характеристикой считают глубину перемежения.

Наиболее простым является метод блочного перемежения (рис. 6.3), основанный на принципе формирования прямоугольной матрицы (рис. 6.4), состоящей из  $n$ -разрядных строк ( $n$  – длина кодовой комбинации), с которой осуществляется поразрядное считывание каждого столбца через  $d$  разрядов.

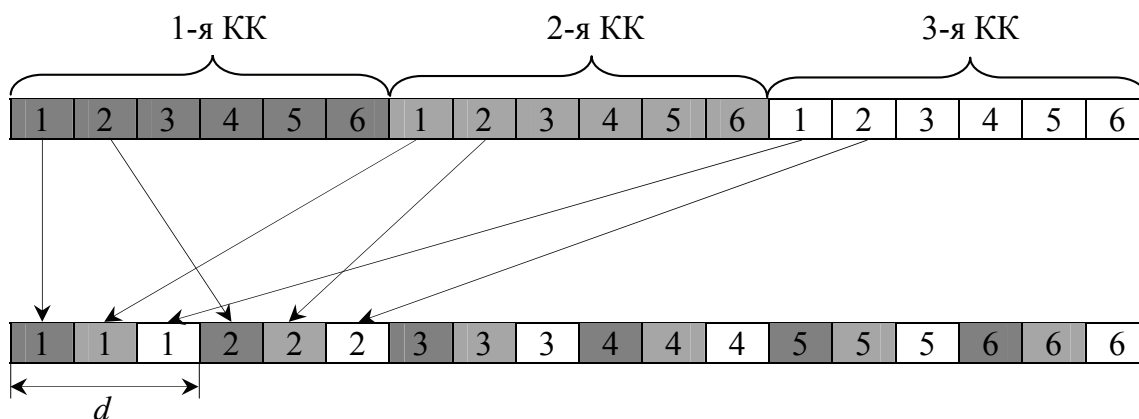


Рис. 6.3. Пояснение к алгоритму блочного перемежения ( $d$  – интервал декорреляции, КК – кодовая комбинация)

1	0	1	0	1	1	0	1	0	1
1	0	0	1	1	1	0	0	1	1
0	0	1	1	0	0	0	1	1	0
1	0	0	0	1	1	0	0	0	1
0	0	0	0	1	0	0	0	0	1

Рис. 6.4. Пояснение принципа блочного перемежения

**! Пример.** Рассмотрим процесс передачи информации с использованием кода Хемминга и блочного перемежителя. Информационный поток на входе кодера Хемминга (используется код 7, 4) имеет вид, как показано на рис. 6.5.

На выходе кодера сообщение будет иметь вид, представленный на рис. 6.6.

Инф. комб. 1	Инф. комб. 2	Инф. комб. 3	Инф. комб. 4	Инф. комб. 5	Инф. комб. 6	Инф. комб. 7	...	Инф. комб. $N$
1 0 0 1	1 1 0 0	0 0 1 0	0 1 0 1	0 1 1 1	1 0 1 0	1 1 1 0	...	0 0 0 0

Рис. 6.5. Структура информационного потока на входе кодера



Кодовая комб. 1							Кодовая комб. 2							Кодовая комб. 3						
1	0	0	1	1	1	0	1	1	0	0	0	1	0	0	0	1	0	1	1	0
Кодовая комб. 4							Кодовая комб. 5							Кодовая комб. 6						
0	1	0	1	1	0	0	0	1	1	1	0	1	0	1	0	1	0	0	1	1
Кодовая комб. 7							...							Кодовая комб. N						
1	1	1	0	1	0	0	...							0	0	0	0	0	0	0

Рис. 6.6. Последовательность символов сообщения на выходе кодера

Матрица перемежения размером  $7 \times 7$  будет иметь вид, как показано на рис. 6.7.

1	0	0	1	1	1	0
1	1	0	0	0	1	0
0	0	1	0	1	1	0
0	1	0	1	1	0	0
0	1	1	1	0	1	0
1	0	1	0	0	1	1
1	1	1	0	1	0	0

Рис. 6.7. Вид и содержание матрицы перемежения

После перемежения сообщение соответствует последовательности, изображенной на рис. 6.8.

Кодовая комбинация 1–7 после перемежения																				
1	1	0	0	0	1	1	0	1	0	1	1	0	1	0	0	1	0	1	1	1
Кодовая комбинация 1–7 после перемежения																				
1	0	0	1	1	0	0	1	0	1	1	0	0	1	1	1	1	0	1	1	0
Кодовая комбинация 1–7 после перемежения							Кодовая комбинация 8–14 после перемежения													
0	0	0	0	0	1	0	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Рис. 6.8. Бинарная последовательность после перемежения

Предположим, что в процессе передачи информации по каналу возник пакет ошибок  $P$  (выделено черным) длиной 7 бит (рис. 6.9).

Кодовая комбинация 1–7 после перемежения																		
1	1	0	0	0	1	1	0	1	0	1	1	0	1	0	1	0	1	0
Кодовая комбинация 1–7 после перемежения																		
0	0	0	1	1	0	0	1	0	1	1	0	0	1	1	1	1	0	1
Кодовая комбинация 1–7 после перемежения							Кодовая комбинация 8–14 после перемежения											
0	0	0	0	0	1	0	...	...	...	...	...	...	...	...	...	...	...	...

Рис. 6.9. Кодовое слово, содержащее группу ошибок

Сообщение на выходе канала связи записывается по столбцам в матрицу деперемежения (рис. 6.10).

1	0	0	0	1	1	0
1	1	1	0	0	1	0
0	0	0	0	1	1	0
0	1	1	1	1	0	0
0	1	0	1	0	1	0
1	0	0	0	0	1	1
1	1	0	0	1	0	0

Рис. 6.10. Вид и содержание матрицы деперемежения

Из матрицы деперемежения двоичные символы сообщения считываются по строкам и поступают на декодер кода Хемминга (см. рис. 6.11).

Кодовая комб. 1							Кодовая комб. 2							Кодовая комб. 3						
1	0	0	0	1	1	0	1	1	1	0	0	1	0	0	0	0	0	1	1	0
Кодовая комб. 4							Кодовая комб. 5							Кодовая комб. 6						
0	1	1	1	1	0	0	0	1	0	1	0	1	0	1	0	0	0	0	1	1
Кодовая комб. 7							...							Кодовая комб. N						
1	1	0	0	1	0	0	...	...	0	0	0	0	0	0	0	0	0	0	0	0

Рис. 6.11. Ошибки, разнесенные по всему сообщению

После деперемежения пакет ошибок преобразован в независимые ошибки кратности 1 для каждой из кодовых комбинаций кода Хемминга. Как помним, такие ошибки код в состоянии

исправить. Информационный поток на выходе декодера кода Хемминга имеет вид в соответствии с рис. 6.12.

Инф. комб. 1	Инф. комб. 2	Инф. комб. 3	Инф. комб. 4	Инф. комб. 5	Инф. комб. 6	Инф. комб. 7	...	Инф. комб. $N$
1 0 0 1	1 1 0 0	0 0 1 0	0 1 0 1	0 1 1 1	1 0 1 0	1 1 1 0	...	0 0 0 0

Рис. 6.12. Информационный поток на выходе декодера кода Хемминга

Рассмотренный метод блочного перемежения применяется в GSM. К числу других используемых на практике относятся следующие методы перемежения/деперемежения: псевдослучайный,  $S$ -типа (применяется в турбо-кодировании, CDMA и др.), циклически-сдвиговой, случайный, диагональный, многошаговый.

Кратко охарактеризуем *метод  $S$ -случайного перемежения*.

Работа  $S$ -случайного перемежителя построена следующим образом. Имеется информационная последовательность длиной  $K$  символов. Предварительно устанавливается размер минимального расстояния разнесения группирующихся ошибок  $S$ . Данное значение показывает, что каждый символ в перемеженной последовательности должен иметь с каждым из предыдущих  $S$  символов разницу во входной последовательности минимум  $S$  позиций. При формировании перемеженной последовательности следующая позиция символа выбирается случайно из входной последовательности. Если данная позиция не находится в пределах  $\pm S$  с предыдущими  $S$  символами выходной последовательности, то она добавляется в выходную последовательность и больше не участвует в выборе. В противном случае выбор позиции символа происходит еще раз. Процесс продолжается до тех пор, пока все  $K$  символов не будут выбраны.

Достоинство: минимальное расстояние составляет  $S$ .

Недостатки: время поиска алгоритма увеличивается с увеличением  $S$ , кроме того, нет гарантии сходимости алгоритма.

**! Пример.** Пусть имеется кодовая последовательность, равная  $K = 16$  бит. Установим значение параметра  $S$  равным 3. Сгенерируем позицию символа. Пусть она будет равна 0. Так как предыдущих позиций символов нет, то ее и записываем. Генерируем новую позицию. Пусть она будет равна 4. Сравниваем эту позицию с предыдущими тремя позициями новой последовательности на условии разницы каждой с генерированной. Если разница хотя бы

с одной позицией меньше 3, то выбранная позиция на данном этапе не учитывается и выбирается новая позиция из числа оставшихся. В противном случае данную позицию записываем в новую последовательность. Пусть в итоге получим сгенерированную последовательность позиций 0, 4, 8, 12, 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15. Перемежение будет выглядеть, как показано на рис. 6.13.

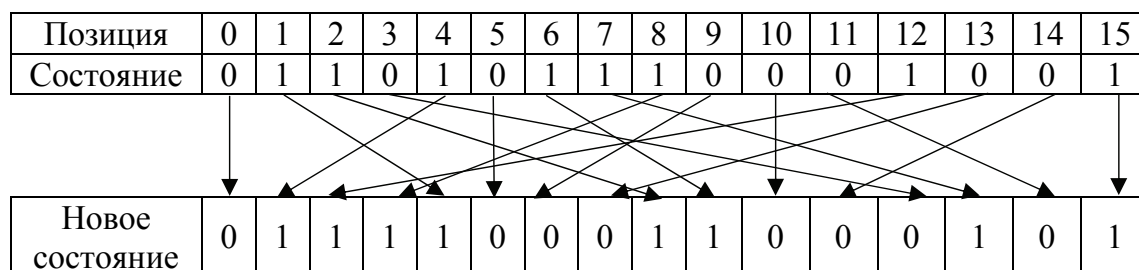


Рис. 6.13. Метод  $S$ -случайного перемежения

Рассмотрим *метод циклически-сдвигового перемежения*.

Циклически-сдвиговое перемежение описывается выражением:

$$\pi(i) = (pi + s) \bmod K,$$

где  $s$  – размер сдвига;  $p$  – размер шага. Значение  $p$  выбирается взаимно простым к  $K$ .

Достоинство: небольшое время перемежения битовых символов. Недостатки: небольшое расстояние разнесения бит.

**! Пример.** Пусть имеется кодовая последовательность  $K$ , равная 16 бит.

Позиция	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Состояние	0	1	1	0	1	0	1	1	1	0	0	0	1	0	0	1

Установим размер сдвига  $s$  равным 2, а размер шага  $p$  равным 5. Определим новое местоположение каждого символа:

$$\begin{aligned} \pi(0) &= (5 \cdot 0 + 2) \bmod 16 = 2 \bmod 16 = 2; \\ \pi(1) &= (5 \cdot 1 + 2) \bmod 16 = 7 \bmod 16 = 7; \\ \pi(2) &= (5 \cdot 2 + 2) \bmod 16 = 12 \bmod 16 = 12; \\ \pi(3) &= (5 \cdot 3 + 2) \bmod 16 = 17 \bmod 16 = 1; \\ \pi(4) &= (5 \cdot 4 + 2) \bmod 16 = 22 \bmod 16 = 6; \\ \pi(5) &= (5 \cdot 5 + 2) \bmod 16 = 27 \bmod 16 = 11; \\ \pi(6) &= (5 \cdot 6 + 2) \bmod 16 = 32 \bmod 16 = 0; \end{aligned}$$

$$\begin{aligned}\pi(7) &= (5 \cdot 7 + 2) \bmod 16 = 37 \bmod 16 = 5; \\ \pi(8) &= (5 \cdot 8 + 2) \bmod 16 = 42 \bmod 16 = 10; \\ \pi(9) &= (5 \cdot 9 + 2) \bmod 16 = 47 \bmod 16 = 15; \\ \pi(10) &= (5 \cdot 10 + 2) \bmod 16 = 52 \bmod 16 = 4; \\ \pi(11) &= (5 \cdot 11 + 2) \bmod 16 = 57 \bmod 16 = 9; \\ \pi(12) &= (5 \cdot 12 + 2) \bmod 16 = 62 \bmod 16 = 14; \\ \pi(13) &= (5 \cdot 13 + 2) \bmod 16 = 67 \bmod 16 = 3; \\ \pi(14) &= (5 \cdot 14 + 2) \bmod 16 = 72 \bmod 16 = 8; \\ \pi(15) &= (5 \cdot 15 + 2) \bmod 16 = 77 \bmod 16 = 13.\end{aligned}$$

Переменная последовательность будет иметь вид в соответствии с нижеследующей таблицей.

Позиция	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Состояние	1	1	1	1	1	0	0	0	0	1	1	0	0	0	1	0



### **Вопросы для контроля и самоконтроля**

1. Назовите основное назначение устройств перемежения.
2. Поясните принцип работы устройств перемежения.
3. Перечислите основные характеристики методов перемежения.
4. Поясните принцип работы блочного метода перемежения.
5. Какая последовательность получится на выходе блочного перемежителя, если входная последовательность имеет вид 101110100?
6. Поясните принцип работы S-случайного метода перемежения.
7. Поясните принцип работы циклически-сдвигового метода перемежения.

# СЖАТИЕ ИНФОРМАЦИИ КАК МЕТОД ПОВЫШЕНИЯ ЕЕ БЕЗОПАСНОСТИ И ЦЕЛОСТНОСТИ

7

## 7.1. Цели использования и классификация методов сжатия сообщений

Сжатие является одним из основных (как и помехоустойчивое кодирование) методов преобразования информации.

ИС, в которой *используется* единственный метод преобразования – сжатие – может быть представлена в виде, схожем со структурной схемой, показанной на рис. 6.1 (с. 37). Только в нашем случае вместо кодера будет использоваться *архиватор* (по сути также являющийся кодером), осуществляющий *сжатие* сообщения, а вместо декодера – распаковщик (осуществляет обратное преобразование).

В отличие от методов помехоустойчивого кодирования, для которых всегда  $k < n$ , методы сжатия подразумевают, что для них характерным является противоположное:  $k > n$ . Таким образом, *основной целью сжатия* информации является уменьшение физического объема сообщения (файла). Это актуально не только при хранении архивов (в том числе и при хранении документов на сайтах), но и при передаче трафика. Следует вспомнить, что создание архивных копий документов и приложений – один из важных элементов защиты информации.

Принято считать, что все существующие методы сжатия разделяются на 2 класса: сжатие без потерь информации и сжатие с частичной потерей информации. В первом случае обратное преобразование (распаковка) всегда позволяет получить исходный оригинал ( $X_k$ ), во втором – нет.

Предел максимального сжатия сообщения без потерь определяет *теорема Шеннона об источнике шифрования*, которая показывает, что когда в потоке независимо и одинаково распределенных случайных переменных данные стремятся к бесконечности, то невозможно сжать данные настолько, что оценка кода сжатого сообщения (среднее число бит на символ сообщения) будет меньше, чем энтропия Шеннона исходных данных.

**❗ Пример.** Если принять, что энтропия английского языка с учетом пробелов составляет 4,7 бит, то сообщение  $X_k = \text{'We are happy'}$  в сжатом виде не может быть меньше, чем 57 бит ( $4,7 \cdot 12 = 56,4$ ).

Следует отметить, что если это сообщение представить в кодах ASCII (одному символу текста соответствует 1 байт), то его длина составит 96 ( $8 \cdot 12$ ) бит.

Если посмотреть на проблему *сжатия без потерь*, то не сложно установить, что это не возможно по определению. Действительно, для того чтобы не было потерь информации, любому сжатому файлу ( $k > n$ ) однозначно должен соответствовать только строго определенный исходный, а соответственно, и распакованный файл. Однако предположим, что имеются все возможные комбинации исходных файлов одинаковой длины, например  $x$ . Общее число таких файлов будет  $2^x$ . Теперь предположим, что каждый исходный файл после сжатия уменьшился на 1 бит, т. е. длина сжатого файла будет  $(x - 1)$  бит. Общее возможное число таких сжатых файлов будет  $2^{x-1}$ . Это означает, что двум или большему числу разных файлов длины  $x$  будет соответствовать один и тот же сжатый файл длины  $(x - 1)$ . А это, в свою очередь, означает, что после распаковки некоторого файла ему в соответствие будет поставлен не тот исходный файл – потери информации неизбежны.

Если абстрагироваться от этих рассуждений, то следует согласиться с тем, что сжатие с потерей части информации недопустимо, если сжимаются текстовые документы, файлы баз данных или коды программ. Частичная потеря практически не сказывается на качестве графических, видео- или аудио-файлов.

Другим, не менее важным классификационным признаком методов сжатия является используемый алгоритм для прямого и обратного преобразований. Именно алгоритм определяет эффективность и скорость прямого (сжатие) и обратного (распаковка) преобразований.

В различных источниках можно встретить отличающиеся в деталях классы методов (в основном – по названиям). Здесь будут рассматриваться следующие классы методов:

- 1) символориентированные и словарные;
- 2) вероятностные или статистические;
- 3) арифметические.

Современные архиваторы, как правило, строятся на базе сочетания различных методов.

Основным критерием эффективности того или иного метода сжатия является значение *коэффициента сжатия*.

*Коэффициент сжатия информации  $R$*  есть отношение объема сжатого сообщения  $V_{\text{сж}}$  (файла) к объему исходного сообщения  $V_{\text{ис}}$  (файла):

$$R = V_{\text{сж}} / V_{\text{ис}}, \quad (7.1)$$

либо отношение разницы упомянутых величин к объему исходного сообщения:


$$R' = (V_{\text{ис}} - V_{\text{сж}}) / V_{\text{ис}} = 1 - R. \quad (7.2)$$

### **Вопросы для контроля и самоконтроля**

1. Пояснить основные цели сжатия информации.
2. Привести общую классификацию методов сжатия информации.
3. Что означает с физической точки зрения «сжатие информации без потерь»? Привести примеры практического использования.
4. Чем определяется нижняя граница сжатия без потерь?
5. Что означает с физической точки зрения «сжатие информации с потерями»? Привести примеры практического использования.
6. Пояснить физический смысл теоремы Шеннона об источнике шифрования.
7. Что является основным критерием эффективности методов сжатия информации?
8. Перечислить и охарактеризовать известные Вам компьютерные архиваторы данных.

## **7.2. Символориентированные и словарные методы сжатия**

**Метод кодирования длин серий (Run-length encoding, RLE) или кодирования повторов.** При сжатии (кодировании) *строка одинаковых символов, составляющих серию, заменяется строкой, которая содержит сам повторяющийся символ и количество его повторов.*

 **Пример.** Черный текст (черный пиксел –  $B$ ) на белом фоне (белый пиксел –  $W$ ).



Вход:

$X_k = \text{'WWWWWWWWWBBBWWWWWWWWWWWWWWBBBWW'}$ ,

$k = 67$ ,

Выход:

$X_n = \text{'10W3B12W3B24W1B14W'}$ ,

$n = 18$ .

Нетрудно подсчитать, что  $R = 18 / 67 \approx 0,27$  (27%). Это означает, что сжатое сообщение занимает 27% от объема исходного.

С другой стороны,  $R' = (67 - 18) / 67 \approx 0,73$  (73%). Это означает, что размер исходного сообщения уменьшился (сжался) на 73%.

**Метод Бэрроуза – Уилера (Burrows – Wheeler Transform, BWT).** Впервые опубликован в 1994 г. Авторы – Дэвид Уилер и Майк Бэрроуз. Сам по себе алгоритм Б-У не является классическим алгоритмом сжатия, однако выходная последовательность гораздо удобнее для сжатия, нежели исходная. Объем сжатого сообщения всегда (!) будет превышать исходный объем.

Преобразование Б-У применяется в алгоритмах сжатия качественных данных. Является компромиссным между быстрыми словарными алгоритмами и статистическими алгоритмами. Обычно применяется совместно со статистическими алгоритмами.

Сущность метода и алгоритм его реализации следующие. Алгоритм оперирует целым блоком данных длиной  $k$  символов.

Для выполнения прямого преобразования необходимо:

1) выделить блок данных  $X_k$  (длиной  $k$ ) из непрерывного потока входных данных;

2) из полученного блока данных создать матрицу  $W_1$  всех возможных его циклических перестановок; первой строкой матрицы будет исходная последовательность  $X_k$ , второй строкой – она же, сдвинутая на один символ влево, и т. д.;

3) отсортировать все строки  $W_1$  в соответствии с лексикографическим порядком символов; получим матрицу  $W_2$ .

Результат преобразования: символы последнего столбца  $w_k$  матрицы  $W_2$  и номер ( $z$ ,  $1 \leq z \leq k$ ) исходной строки среди отсортированных. Именно на значение числа  $z$  увеличивается объем исходного сообщения после прямого преобразования.

Важнейшее свойство алгоритма – *рекуррентность* (*рекурсивная функция* (от лат. *recursio* – возвращение) – это числовая функция  $f(n)$  числового аргумента, которая в своей записи содержит

себя же; такая запись позволяет вычислять значения  $f(n)$  на основе значений  $f(n-1)$ ,  $f(n-2)$  ..., подобно рассуждению по индукции).

Это свойство алгоритма используется при обратном преобразовании. Для обратного преобразования необходимо воспроизвести матрицу  $W_2$  и по значению  $z$  определить исходное сообщение.

Для выполнения обратного преобразования необходимо выполнить  $k$  двухшаговых операций: а) в крайний справа (из числа свободных) столбцов воссоздаваемой матрицы вписать значение столбца  $w_k$ ; б) выполнить сортировку строк этой матрицы (исходя из заполненных значений) по возрастанию. На выходе обратного преобразования получим значение  $X_k$ , зная число  $z$ .

**❗ Пример.** Исходное сообщение  $X_k = \text{'столб'}$ .

*Прямое преобразование.*

$$W_1 = \begin{vmatrix} c & t & o & l & b \\ t & o & l & b & c \\ o & l & b & c & t \\ l & b & c & t & o \\ b & c & t & o & l \end{vmatrix} \Rightarrow W_2 = \begin{vmatrix} b & c & t & o & l \\ l & b & c & t & o \\ o & l & b & c & t \\ c & t & o & l & b \\ t & o & l & b & c \end{vmatrix}$$

Таким образом, имеем:  $w_k = \text{'лотбс'}$  ( $k = 5$ ),  $z = 4$  (исходное сообщение – это 4-я строка матрицы  $W_2$ ).

*Обратное преобразование.*

$$W = \begin{vmatrix} & & l \\ & o \\ & t \\ & b \\ & c \end{vmatrix} \Rightarrow \begin{vmatrix} & l & b \\ & o & l \\ & t & o \\ & b & c \\ & c & t \end{vmatrix} \Rightarrow \begin{vmatrix} & l & b & c \\ & o & l & b \\ & t & o & l \\ & b & c & t \\ & c & t & o \end{vmatrix} \Rightarrow$$

$$\Rightarrow \begin{vmatrix} l & b & c & t \\ o & l & b & c \\ t & o & l & b \\ b & c & t & o \\ c & t & o & l \end{vmatrix} \Rightarrow \begin{vmatrix} l & b & c & t & o \\ o & l & b & c & t \\ t & o & l & b & c \\ b & c & t & o & l \\ c & t & o & l & b \end{vmatrix} \Rightarrow W = \begin{vmatrix} b & c & t & o & l \\ l & b & c & t & o \\ o & l & b & c & t \\ c & t & o & l & b \\ t & o & l & b & c \end{vmatrix}$$

Как видно, на первом шаге в крайний справа незаполненный столбец (пятый) воссоздаваемой матрицы ( $W$ ) записывается значение слова  $w_k = \text{'лотбс'}$ , на втором шаге матрица сортируется (реально сортируются строки, содержащие только символы последнего столбца): получим в последнем столбце 'блост'. Первая операция закончена. Вторая операция: в крайний справа (теперь четвертый) столбец опять записывается входное сообщение: 'лотбс'. Производится сортировка матрицы по значениям 2 последних столбцов, и т. д. После выполнения указанных операций фактически будет воссоздана (если не возникли ошибки) матрица  $W_2$ . Зная  $z = 4$ , получим исходное сообщение: 'столб'.

❗ **Пример.**  $X_k = 101010$ ,  $k = 6$ .

Матрица  $W_1$ :  
 101010  
 010101  
 101010  
 010101  
 101010  
 010101,

Матрица  $W_2$ :  
 010101  
 010101  
 010101  
 101010  
 101010  
 101010.

Результат преобразования:  $w_k = 111000$ ,  $z = 4$  (либо 5, либо 6). Как видно, двоичное сообщение может на выходе создавать бинарные последовательности с блоками повторяющихся символов. Такие последовательности легко преобразуются в другие гораздо меньшей длины (с помощью, например, метода кодирования длин серий).

Обратное преобразование: рассмотрим ход преобразования, описывая каждый шаг.

$W$ :	1	0	10	01	101	010	1010	0101	10101	01010	101010	010101
	1	0	10	01	101	010	1010	0101	10101	01010	101010	010101
	1	0	10	01	101	010	1010	0101	10101	01010	101010	010101
	0	1	01	10	010	101	0101	1010	01010	10101	010101	101010
	0	1	01	10	010	101	0101	1010	01010	10101	010101	101010
	0	1	01	10	010	101	0101	1010	01010	10101	010101	101010

Поясним значение информации в каждом графическом столбце последней матрицы.

Первый шаг: содержание первого свободного справа столбца восстанавливаемой матрицы (вписано слово  $w_k = 111000$ ).

Второй шаг: выполнена сортировка матрицы (на самом деле это только правый столбец): 000111.

Новая операция. Первый шаг: в крайний справа свободный столбец (пятый) восстанавливаемой матрицы вновь записывается слово  $w_k = 111000$ . Теперь эта восстанавливаемая матрица содержит уже два крайних столбца.

Второй шаг: выполняется сортировка, и т. д.

После выполнения 6-й операции будет восстановлена матрица, повторяющая матрицу  $W_2$ . По значению  $z = 4$  имеем исходное значение: 101010.

Особенности алгоритма Б-У:

- главной проблемой в реализации алгоритма Б-У является выбор быстрого алгоритма сортировки данных с большим  $k$ ;
- чем меньше мощность алфавита, тем эффективнее метод сжатия по Б-У;
- метод Б-У используется в архиваторах класса zip совместно с другими методами.

**Метод Лемпеля – Зива** (Lempel-Ziv Method). В 1977 г. два исследователя из Израиля предложили новый подход к проблеме сжатия. Авраам Лемпель и Якоб Зив выдвинули идею формирования «словаря» общих последовательностей данных. При этом сжатие данных осуществляется за счет замены записей соответствующими кодами из словаря. Классический алгоритм Лемпеля – Зива – LZ77, названный так по году своего опубликования, предельно прост. Он формулируется следующим образом: *«если в проанализированном (сжатом) ранее выходном потоке уже встречалась подобная последовательность байт, причем запись о ее длине и смещении от текущей позиции короче, чем сама эта последовательность, то в выходной файл записывается ссылка (смещение, длина), а не сама последовательность»*.

Рассмотренный выше метод сжатия RLE, который заключается в записи вместо последовательности одинаковых символов одного символа и их количества, является подклассом алгоритма LZ77.

Суть метода LZ77 (как и последующих его модификаций) состоит в следующем: упаковщик постоянно хранит некоторое количество последних обработанных символов в *буфере*. По мере

обработки входного потока вновь поступившие символы попадают в конец буфера, сдвигая предшествующие символы и вытесняя самые старые. Размеры этого буфера, называемого также *скользящим словарем* (sliding dictionary), варьируются в разных реализациях систем сжатия. Скользящее окно имеет длину  $n$ , т. е. в него помещается  $n$  символов, и состоит из двух частей:

- последовательности длиной  $n_1 = n - n_2$  уже закодированных символов (словарь);
- упреждающего буфера (буфера предварительного просмотра, *lookahead*) длиной  $n_2$  – буфер кодирования.

Формальное представление алгоритма следующее. Пусть к текущему моменту времени закодировано  $t$  символов:  $S_1, S_2, \dots, S_t$ . Тогда словарем будут являться  $n_1$  предшествующих символов:  $S_{t - (n_1 - 1)}, S_{t - (n_1 - 1) + 1}, \dots, S_t$ .

В буфере находятся ожидающие кодирования (сжатия) символы  $S_{t+1}, S_{t+2}, \dots, S_{t+n_2}$ . Если  $n_2 \geq t$ , то словарем будет являться вся уже обработанная часть входной последовательности.

*Нужно найти самое длинное совпадение между строкой буфера кодирования, начинающейся с символа  $S_{t+1}$ , и всеми фразами словаря.*

Эти фразы могут начинаться с любого символа  $S_{t - (n_1 - 1)}, S_{t - (n_1 - 1) + 1}, \dots, S_t$ , выходить за пределы словаря, вторгаясь в область буфера, но должны лежать в окне. Буфер не может сравниваться сам с собой. Длина совпадения не должна превышать размера буфера. Полученная в результате поиска фраза  $S_{t - (p - 1)}, S_{t - (p - 1) + 1}, \dots, S_{t - (p - 1) + (q - 1)}$  кодируется с помощью двух чисел:

- 1) смещения (*offset*) от начала буфера  $p$ ;
- 2) длины соответствия, или совпадения (*match length*),  $q$ .

Ссылки ( $p$ - и  $q$ -указатели) однозначно определяют фразу. Дополнительно в выходной поток записывается символ  $s$ , следующий за совпавшей строкой буфера.

Длина кодовой комбинации (триады –  $p, q, s$ ) на каждом шаге определяется соотношением

$$l(C_i) = \log_N n_1 + \log_N n_2 + 1,$$

где  $N$  – мощность алфавита.

После каждого шага окно смещается на  $(q + 1)$  символов вправо и осуществляется переход к новому циклу кодирования. Величина сдвига объясняется тем, что мы реально закодировали именно  $(q + 1)$  символов:  $q$  – с помощью указателя и 1 – с помощью тривиального копирования.

Передача одного символа в явном виде ( $s$ ) позволяет разрешить проблему обработки еще ни разу не встречавшихся символов, но существенно увеличивает размер сжатого блока.

**! Пример.** Используется алфавит  $A = \{0,1,2,3\}$ ,  $N = 4$ , длина словаря  $n_1 = 15$ , длина буфера данных (кодирования)  $n_2 = 13$ .

Для обозначения  $p$  и  $q$  используется четверичная система счисления. Длина кодовой комбинации на каждом шаге:

$$l(C_i) = \log_N n_1 + \log_N n_2 + 1 = 2 + 2 + 1 = 5. \quad (7.3)$$

$S = 2000302013020130313031303133333333$ .

Вспомним соответствие между числами в десятичной и четверичной системах счисления:

$0_{10} = 00_4$ ,  $1_{10} = 01_4$ ,  $2_{10} = 02_4$ ,  $3_{10} = 03_4$ ,  $4_{10} = 10_4$ ,  $5_{10} = 11_4$  и т. д.

*Прямое преобразование.* На первом шаге: состояние буфера отображает рис. 7.1.

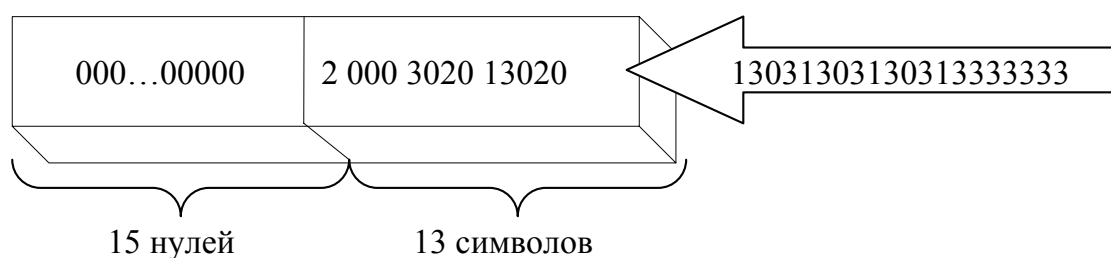


Рис. 7.1. Состояние буфера на первом шаге сжатия сообщения  $S$

Анализируем 1-й символ в буфере кодирования на предмет соответствия (наличия) такого же символа или нескольких символов в словаре. Таких символов нет, следовательно,  $p_1 = 0$ . Длина повторения  $q_1 = 0$ . Таким образом, имеем следующую триаду:

$$(p_1, q_1, s_1) = (00 \ 00 \ 2)_4.$$

Здесь нижний индекс справа от знака равенства означает основание системы счисления.

На втором шаге: состояние буфера отображает рис. 7.2.

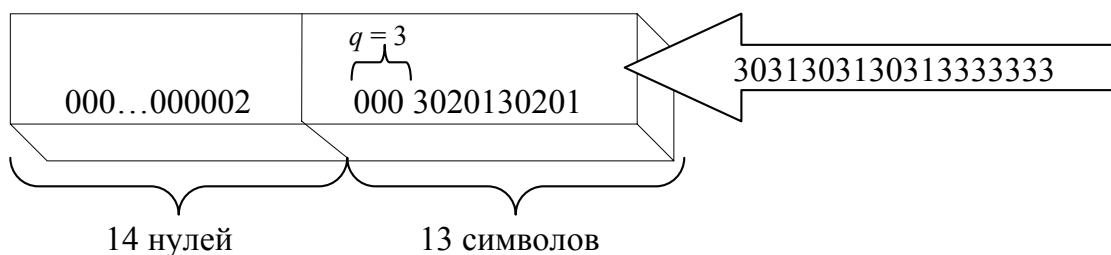


Рис. 7.2. Состояние буфера на втором шаге сжатия сообщения  $S$

Находим повторение (000), длина этого повторения  $q_2 = 3$ . Поскольку в словаре нулевые символы записываются с 1-й по 14-ю позицию (для упрощения индексация ведется слева направо), то индексом  $p_2$  (началом повторения) может быть выбрано любое число от 1 до 12:

$1 \leq p_2 \leq 12$ ; выбираем  $p_2 = 6$ .

Итак, получим следующую триаду:

$$(p_2, q_2, s_2) = (6, 3, 3) = (12 \ 03 \ 3)_4.$$

Передвигаем сообщение в окне на  $q_2 + 1 = 3 + 1 = 4$  позиции.

На третьем шаге: состояние буфера отображает рис. 7.3.

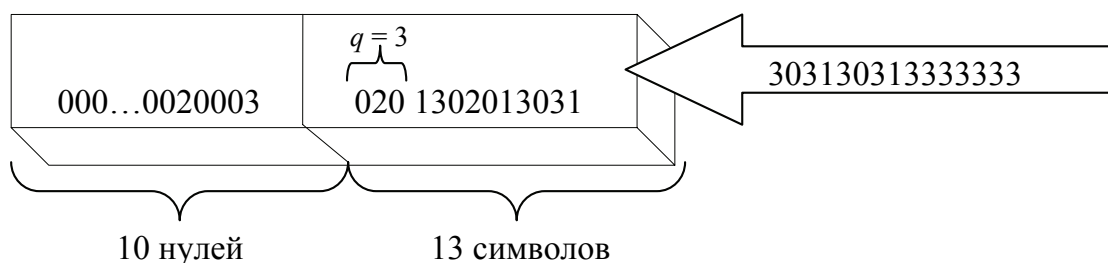


Рис. 7.3. Состояние буфера на третьем шаге сжатия сообщения  $S$

В этой ситуации наиболее длинный повтор – 020,  $q_3 = 3$ ,  $p_3 = 10$ . Передвигаем сообщение в окне на  $q_3 + 1 = 4$  позиции и получаем триаду:

$$(p_3, q_3, s_3) = (10, 3, 1) = (22 \ 03 \ 1)_4.$$

На четвертом шаге состояние буфера отображает рис. 7.4.

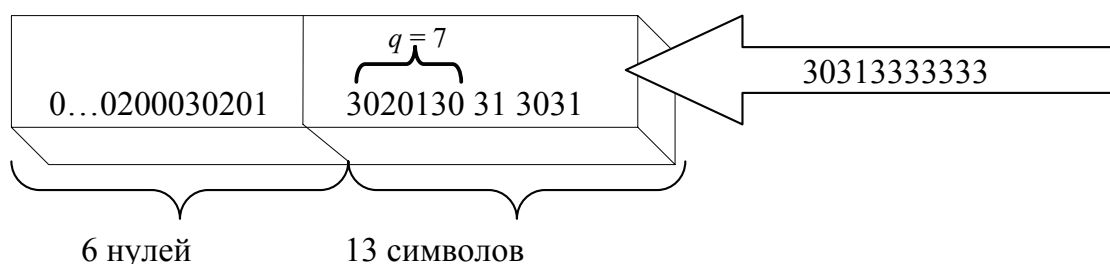


Рис. 7.4. Состояние буфера на четвертом шаге сжатия сообщения  $S$

В этой ситуации наиболее длинный повтор – 3020130. Значит,  $q_4 = 7$ ,  $p_4 = 11$ . Передвигаемся на  $q_4 + 1 = 8$  и получаем триаду:

$$(p_4, q_4, s_4) = (11, 7, 3) = (23 \ 13 \ 3)_4 \text{ и т. д.}$$

*Обратное преобразование.* Имеем в 4-й системе счисления запакованное сообщение: 00002 12033 22031 23133 30301 02013 32103... .

В исходном состоянии в окне (используется одно скользящее окно) записывают 15 нулей (такой выбрана длина окна). Результат анализа – символы исходного сообщения на основе выбранного алфавита.

*Первый шаг:* анализируем первые 5 символов (вспомним,  $l(C_i) = 5$ ): 00002, следовательно  $q_1 = 0$  и  $p_1 = 0$ . В результате в младший (крайний справа) разряд окна записывается лишь символ 3 ( $s_1 = 3$ ).

*Второй шаг:* анализируем следующие 5 символов: 12033, т. е.  $p_2 = 6$ ,  $q_2 = 3$ . Поскольку в окне содержатся только нули, за исключением последнего разряда (15, индексация ведется как и в случае прямого преобразования), то мы можем сделать вывод, что наш повтор – это следующие три символа: 000. К ним дописывается последний символ из анализируемой триады ( $s_2 = 3$ ). Таким образом, после двух шагов начальные символы распакованного сообщения будут такими: 20003. В остальных (начальных: с первого по десятый) разрядах окна будут нули.

*Третий шаг:* анализируем третью триаду сжатого сообщения: 22031. Следовательно,  $p_3 = 10$ ,  $q_3 = 3$ . Разряды с десятого ( $p_3 = 10$ ) по двенадцатый ( $q_3 = 3$ ) окна содержат символы 020. Эти разряды допишутся справа к существующим символам окна и дополнительно к ним допишется символ 1 ( $s_3 = 1$ ). Таким образом, в окне после этого будет записано: 000000200030201.

И так далее.

**!** *Пример.*  $S = \text{«КРАСНАЯ КРАСКА»}$ . Длина словаря  $n_1 = 8$ , длина буфера данных (кодирования)  $n_2 = 8$ .

*Прямое преобразование.* Словарь изначально заполнен нулями.

*Первый шаг.*

Словарь	Буфер данных	Сообщение	Код
«00000000»	«КРАСНАЯ »	«КРАСКА»	(0,0,K)

Из буфера данных берем первый слева символ («К») и осуществляем поиск данного символа в словаре. Символ в словаре отсутствует. Формируем триаду с начальным значением повторяющейся последовательности ( $p_1$ ), равным 0, длиной повторяющихся символов ( $q_1$ ), равной 0, и следующим за повторяющейся последовательностью символом ( $c_1$ ): «К». Сдвигаем окна на  $q_1 + 1$  позицию влево. Состояние буфера показано ниже.



Второй шаг.

Словарь	Буфер данных	Сообщение	Код
«0000000К»	«РАСНАЯ К»	«РАСКА»	(0,0,Р)

Отсутствие в словаре символа «Р». Триада формируется с началом повторяющей последовательности ( $p_2$ ), равной 0, длиной повторяющихся символов ( $q_2$ ), равной 0, и следующим за повторяющейся последовательностью символом ( $c_2$ ): «Р». Сдвигаем окна на  $q_2 + 1$  позицию влево.

Третий шаг.

Словарь	Буфер данных	Сообщение	Код
«000000КР»	«АСНАЯ КР»	«АСКА»	(0,0,А)

Нетрудно установить, что  $p_3 = 0$ ,  $q_3 = 0$ ,  $c_3 = «А»$ . Сдвигаем окна на  $q_3 + 1$  позицию влево.

Четвертый шаг.

Словарь	Буфер данных	Сообщение	Код
«00000КРА»	«СНАЯ КРА»	«СКА»	(0,0,С)

Новая триада:  $p_4 = 0$ ,  $q_4 = 0$ ,  $c_4 = «С»$ . Сдвигаем окна на  $q_4 + 1$  позицию влево.

Пятый шаг.

Словарь	Буфер данных	Сообщение	Код
«0000КРАС»	«НАЯ КРАС»	«КА»	(0,0,Н)

Как и на предыдущих шагах, повторов не найдено. Триада:  $p_5 = 0$ ,  $q_5 = 0$ ,  $c_5 = «Н»$ . Сдвигаем окна на  $q_5 + 1$  позицию влево.

Шестой шаг.

Словарь	Буфер данных	Сообщение	Код
«000КРАСН»	«АЯ КРАСК»	«А»	(6,1,Я)

В словаре есть повторяющийся символ «А». Формируем триаду:  $p_6 = 6$ ,  $q_6 = 1$  и  $c_6 = «Я»$ . Осуществляем сдвиг в окнах на  $q_6 + 1$  позицию влево.

Седьмой шаг.

Словарь	Буфер данных	Сообщение	Код
«0КРАСНАЯ»	« _ КРАСКА»	«»	(0,0, _)

Повторений не найдено. Сформированная триада:  $p_7 = 0$ ,  $q_7 = 0$  и  $c_7 = «_»$ . Сдвиг в окнах произойдет на  $q_7 + 1$  позицию влево.

Восьмой шаг.

Словарь	Буфер данных	Сообщение	Код
«КРАСНАЯ_»	«КРАСКА»	«»	(1,4,K)

Легко понять, что в этом случае значение совпадающих подстрок соответствует «КРАС». В результате имеем:  $p_8 = 1$ ,  $q_8 = 4$ ,  $c_8 = \text{«К»}$ . Сдвигаем окна на 5 позиций влево.

Девятый шаг.

Словарь	Буфер данных	Сообщение	Код
«АЯ КРАСК»	«А»	«»	(1,1,)

Так как символ «А» является последним, то при формировании триады параметр  $s$  останется пустым.

Результатом сжатия исходного сообщения будет набор полученных триад (кодов): 00K00P00A00C00H61Я00\_14K11.

*Обратное преобразование.* В процессе восстановления используются только набор триад (кодов) и один буфер с нулевыми начальными значениями. Размер окна такой же, как и при сжатии.

Исходное состояние.

Восстановленное сообщение	Буфер	Код
«»	«00000000»	(0,0,K)

Первый шаг: анализируется триада (0,0,K). Начало повторяющейся последовательности ( $p_1$ ) и ее длина ( $q_1$ ) равны 0 – повторений нет. Добавляем в буфер символ «К» и сдвигаем окно на  $q_1 + 1$  позиций влево.

Дальнейшие шаги и результаты выполненных операций понятны из последующих таблиц.

Так как до шестого шага повторений нет ( $p_2, p_3, p_4, p_5$  и  $q_2, q_3, q_4, q_5$  равны 0), то добавляем в буфер на каждом шаге соответствующий символ и сдвигаем на 1 позицию влево.

Второй шаг.

Восстановленное сообщение	Буфер	Код
«К»	«00000000K»	(0,0,P)

Третий шаг.

Восстановленное сообщение	Буфер	Код
«КР»	«00000000КР»	(0,0,A)

Четвертый шаг.

Восстановленное сообщение	Буфер	Код
«КРА»	«00000КРА»	(0,0,С)

Пятый шаг.

Восстановленное сообщение	Буфер	Код
«КРАС»	«0000КРАС»	(0,0,Н)

Шестой шаг.

Восстановленное сообщение	Буфер	Код
«КРАСН»	«000КРАСН»	(6,1,Я)

Из буфера берется символ «А» ( $p_6 = 6$  и  $q_6 = 1$ ) и дописывается в конец буфера. Кроме этого, добавляется символ «Я» ( $c_6 = \text{«Я»}$ ) и происходит сдвиг на 2 ( $q_6 = 1$ ) позиции влево.

Седьмой шаг.

Восстановленное сообщение	Буфер	Код
«КРАСНАЯ»	«0КРАСНАЯ»	(0,0, _)

Рассматриваем триаду:  $p_7 = 0$ ;  $q_7 = 0$ . Добавляем в конец буфера символ «\_» и сдвигаем на  $q_7 + 1$  позицию влево.

Восьмой шаг.

Восстановленное сообщение	Буфер	Код
«КРАСНАЯ _»	«КРАСНАЯ _»	(1,4,К)

Повторяющуюся последовательность «КРАС» ( $p_8 = 1$  и  $q_8 = 4$ ) дописываем в буфер вместе с символом «К» ( $c_8 = \text{«К»}$ ). Сдвигаем на 5 позиций влево.

Девятый шаг.

Восстановленное сообщение	Буфер	Код
«КРАСНАЯ КРАСК»	«АЯ КРАСК»	(1,1,)

Осуществляем добавление последней буквы «А» ( $p_9 = 1$  и  $q_9 = 1$ ).

Результатом декомпрессии будет сообщение «КРАСНАЯ КРАСКА».

Восстановленное сообщение	Буфер	Код
«КРАСНАЯ КРАСКА»	«Я КРАСКА»	

**!** *Пример.*

$S = \text{«1001001110110010101100110»}$ . Длина словаря  $n_1 = 16$ , длина буфера данных (кодирования)  $n_2 = 16$ .

*Прямое преобразование.* Словарь изначально заполнен нулями. Буфер данных содержит первые 16 символов исходного сообщения. Так как размер буфера словаря и буфера данных имеет двухзначное число, то при формировании триады необходимо для записи значений  $p$  и  $q$  отводить 2 знака.

Первый шаг.

Словарь	Буфер данных	Сообщение	Код
«0000000000000000»	«1001001110110010»	«101100110»	(00,00,1)

Из буфера данных берем первый слева символ «1» и осуществляем поиск данного символа в буфере словаря. Символ в словаре отсутствует. Формируем триаду с началом повторяющейся последовательности  $p_1$ , равной 0, длиной повторяющихся символов  $q_1$ , равной 0, и следующим за повторяющейся последовательностью символом  $c_1 = \text{«1»}$ . Сдвигаем окна на  $q_1 + 1$  позицию влево.

Второй шаг.

Словарь	Буфер данных	Сообщение	Код
«0000000000000001»	«0010011101100101»	«01100110»	(14,03,0)

Определяем максимальную повторяющуюся последовательность «001». Записываем триаду:  $p_2 = 14$ ;  $q_2 = 3$ ;  $c_2 = \text{«0»}$ . Производим сдвиг на  $q_2 + 1$  позицию влево.

Третий шаг.

Словарь	Буфер данных	Сообщение	Код
«00000000000010010»	«0111011001010110»	«0110»	(11,02,1)

Начало повторяющейся последовательности «01» в словаре начинается с 11-й позиции ( $p_3$ ). Ее длина равна 2 ( $q_3$ ). Следующий символ за повторением – «1» ( $c_3$ ). Сформируем триаду и сдвинем на 3 позиции влево окна.

Четвертый шаг.

Словарь	Буфер данных	Сообщение	Код
«0000000010010011»	«1011001010110011»	«0»	(09,02,1)

Следующая найденная комбинация «10» заменяется триадой с началом повторяющейся последовательности ( $p_4$ ), равной 9, длиной повторяющихся символов ( $q_4$ ), равной 2, и следующим за повторяющейся последовательностью символом ( $c_4$ ) «1». Производим сдвига окон на  $q_4 + 1$  позицию влево.

Пятый шаг.

Словарь	Буфер данных	Сообщение	Код
«00000 <b>1001001</b> 1101»	« <b>10010</b> 101100110»	«»	(06,05,1)

После замены набора символов «10010», встречающегося в словаре с шестой позиции, триадой ( $p_5 = 6, q_5 = 5, c_5 = \langle 1 \rangle$ ) сдвигаем на  $q_5 + 1$  позиций окно влево.

Шестой шаг.

Словарь	Буфер данных	Сообщение	Код
«001001110 <b>1100101</b> »	« <b>011001</b> 10»	«»	(09,06,1)

Последовательность «011001», состоящую из 6 символов, подвергнем замене триадой ( $p_6 = 9, q_6 = 6, c_6 = \langle 1 \rangle$ ). Сдвигаем окна на  $q_6 + 1$  позицию влево.

Седьмой шаг.

Словарь	Буфер данных	Сообщение	Код
«1011001010110011»	«0»	«»	(02,01,)

Для последнего символа формируем триаду. Так как такой символ присутствует в словаре, то  $c_7$  будет пустым. Сформированная триада получит вид ( $p_7 = 2, q_7 = 1, c_7 = \langle \rangle$ ).

Результатом сжатия будет набор полученных триад (кодов) – 0000114030110210902106051090610201.

*Обратное преобразование.* В процессе восстановления используются только набор триад (кодов) и буфер. Размер буфера и его начальное содержимое такое же, как и при сжатии.

Первый шаг.

Восстановленное сообщение	Буфер	Код
«»	«0000000000000000»	(00,00,1)

Анализируется триада (00,00,1). Начало повторяющейся последовательности ( $p_1$ ) и ее длина ( $q_1$ ) равны 0 – повторений нет. Добавляем к буферу символ  $c_1 = \langle 1 \rangle$  и сдвигаем окно буфера на  $q_1 + 1$  позицию влево.

Второй шаг.

Восстановленное сообщение	Буфер	Код
«1»	«0000000000000001»	(14,03,0)

Следующая триада (14,03,0). Дописываем к буферу повторяющуюся последовательность «001», так как  $p_2 = 14$ , а длина  $q_2 = 3$ , и символ «0» ( $c_2$ ). Двигаем окно буфера на  $q_2 + 1$  позицию влево.

Третий шаг.

Восстановленное сообщение	Буфер	Код
«10010»	«0000000000010010»	(11,02,1)

Добавляем к буферу последовательность «01» из этого же буфера согласно позиции начала  $p_3 = 11$  и длиной  $q_3 = 2$  вместе с символом  $c_3 = «1»$ . Производим соответствующий сдвиг.

Четвертый шаг.

Восстановленное сообщение	Буфер	Код
«10010011»	«0000000010010011»	(09,02,1)

Присоединяем повторяющуюся последовательность «01» ( $p_4 = 9$ ,  $q_4 = 2$ ) вместе с символом  $c_4 = «1»$  к буферу с последующим сдвигом на необходимое количество позиций влево.

Пятый шаг.

Восстановленное сообщение	Буфер	Код
«10010011101»	«0010011101100101»	(06,05,1)

Разбираем следующую триаду. Комбинация символов «10010» с началом в шестой позиции ( $p_5$ ) и длиной ( $q_5$ ), равной 5, добавляем к буферу. Дописываем символ «1» ( $c_5$ ). Сдвигаем на  $q_5 + 1$  позицию влево.

Шестой шаг.

Восстановленное сообщение	Буфер	Код
«10010011101100101»	«0010011101100101»	(09,06,1)

Повторяющаяся последовательность ( $p_6$ ) начинается с девятой позиции, ее длина ( $q_6$ ) равна 6. Добавляем к словарию повторяющуюся последовательность «011001» и символ «1» ( $c_6$ ). Сдвигаем буфер на  $q_6 + 1$  позицию влево.

Седьмой шаг.

Восстановленное сообщение	Буфер	Код
«100100111011001010110011»	«1011001010110011»	(02,01,)

Рассматриваем последнюю триаду. Добавляем символ «1» ( $p_7 = 2$ ,  $q_7 = 1$ ,  $c_7 = «»$ ). На этом процесс восстановления окончен.

Результат декомпрессии – сообщение «1001001110110010101100110».

Как видно из приведенных примеров, алгоритм LZ77 обладает следующими очевидными недостатками:

1) невозможность кодирования подстрок, отстоящих одна от другой на расстояние, большее длины словаря;

2) длина кодируемой подстроки ограничена размером буфера.

Однако если увеличивать размер словаря, то это приведет к увеличению времени преобразования и возрастанию длины кодовых последовательностей.

В 1978 г. создателями алгоритма LZ77 был разработан усовершенствованный алгоритм LZ78, лишенный названных недостатков. LZ78 не использует «скользящее» окно. Он хранит словарь из уже просмотренных подстрок. При старте алгоритма этот словарь содержит только одну пустую строку (строку длины нуль). Алгоритм считывает символы сообщения до тех пор, пока накапливаемая подстрока входит целиком в одну из фраз словаря. Как только эта подстрока перестанет соответствовать хотя бы одной фразе словаря, алгоритм генерирует код, состоящий из индекса строки в словаре, которая до последнего введенного символа содержит входную строку, и символа, нарушившего совпадение. Затем в словарь добавляется введенная подстрока. Если словарь уже заполнен, то из него предварительно удаляют наиболее редко используемую в сравнениях фразу.

Ключевым для размера получаемых кодов является размер словаря во фразах. Это связано с тем, что каждый код при кодировании по данному методу содержит номер фразы в словаре. А это означает, что указанные коды (соответствующие ранее упоминавшимся триадам) имеют постоянную длину.

Алгоритмы LZ-формата используются в тех случаях, когда требуется универсальное сжатие. Например, в стандарте модема V.42bis, протоколе передачи данных ZModem, форматах GIF, TIFF, ARC и других прикладных программах. Некоторые алгоритмы данного класса используются в дисковых утилитах сжатия типа DoubleSpace и Stacker, графических форматах типа PNG, а также в универсальных утилитах архивирования и сжатия, включая ZIP, GZIP и LHA.



### **Вопросы для контроля и самоконтроля**

1. Поясните сущность символоориентированных или словарных методов сжатия.
2. Охарактеризуйте сжатие данных методом «кодирования длин серий». Приведите примеры. Оцените эффективность сжатия.
3. Охарактеризуйте сжатие данных методом Берроуза – Уилера. Приведите примеры. Оцените эффективность сжатия.

4. Как вы понимаете рекуррентность (рекурсивность) преобразования по методу Берроуза – Уилера?

5. Приведите пример прямого и обратного преобразования методом Берроуза – Уилера сообщений:

- а) 101010;
- б) 1110011;
- в) «колобок»;
- г) «форма».

Оцените эффективность сжатия.

6. Приведите пример прямого и обратного преобразования методом Лемпеля – Зива сообщений:

- а) состоит из ваших имени и фамилии;
- б) 11100111101000011101110111;
- в) «охарактеризуйте сжатие данных»;
- г) «метод Лемпеля – Зива».

Обоснуйте выбор размера элементов буфера. Оцените эффективность сжатия сообщений.

### 7.3. Вероятностные (статистические) методы

До появления уже упоминавшихся работ К. Шеннона кодирование символов алфавита при передаче сообщения по каналам связи осуществлялось одинаковым количеством бит, получаемым по формуле Хартли (см. (4.2)). Позднее начали появляться способы, кодирующие символы разным числом бит в зависимости от вероятности появления их в тексте. Таким образом, за счет использования для каждого значения байта (символа алфавита) кода различной длины можно значительно уменьшить общий размер данных. Эта базовая идея лежит в основе алгоритмов сжатия Шеннона – Фано (Shannon – Fano) и Хаффмана (Huffman). Подобные алгоритмы позволяют создавать более короткие коды для часто встречающихся и более длинные – для редко встречающихся значений байта (символа). Частота (частота или вероятность) появления того или иного символа алфавита в произвольном сообщении, лежащая в основе алгоритмов, дала название этим алгоритмам и соответствующим методам.

Иногда эти методы называют также *префиксными*.

Если имеется некоторый код  $X_1 = A_1A_2$  и другой код  $X_2 = A_1$ , то говорят, что  $X_2$  является *префиксом*  $X_1$ .





**Пример 1.**  $X_1 = 100, X_2 = 1$ .

**! Пример 2.**  $X_1 = \text{«ававс»}, X_2 = \text{«ав»}$ .

**Метод Шеннона – Фано.** Метод реализует общую идею вероятностного кодирования (сжатия) с использованием префиксных множеств и работает следующим образом. Создается отсортированная (в порядке убывания или возрастания значения вероятности  $p(a_i)$ ;  $p(a_i)$  – вероятность появления в сжимаемом сообщении на произвольной позиции символа  $a_i$  алфавита) таблица символов алфавита, на основе которого генерируется сжимаемое сообщение. Далее эта таблица символов делится на две группы таким образом, чтобы каждая из групп имела приблизительно одинаковую суммарную частоту (вероятность). Очевидно, такая суммарная вероятность в каждой из групп должна быть максимально близка к 0,5. Первой группе устанавливается начало кода в «0», второй – в «1». Для вычисления следующих бит кодов символов данная процедура повторяется рекурсивно для каждой группы, в которой больше одного символа. Получим таблицу, в которой длина кодовых комбинаций меняется от минимального ( $l_{\min}$ ) до максимального ( $l_{\max}$ ) значений.

Если упомянутая таблица создается на основе анализа вероятностных свойств символов только сжимаемого сообщения, то такой метод кодирования называется *адаптивным* или *динамическим*, в противном случае метод относят к *статическим*.

**! Пример.** Имеем алфавит  $A\{a_i\}$ ,  $i = [1...10]$ ,  $N(A) = 10$  – мощность алфавита. При этом получены следующие вероятности для символов алфавита:

$$\begin{aligned} p(a_1) &= 0,05; & p(a_6) &= 0,12; \\ p(a_2) &= 0,10; & p(a_7) &= 0,05; \\ p(a_3) &= 0,03; & p(a_8) &= 0,10; \\ p(a_4) &= 0,20; & p(a_9) &= 0,15; \\ p(a_5) &= 0,15; & p(a_{10}) &= 0,05. \end{aligned}$$

Если такая таблица принимается исходной при преобразовании произвольного сообщения на основе этого алфавита (заранее известна и архиватору, и распаковщику), то метод преобразования следует классифицировать как статический.

*Алгоритм прямого преобразования.* Необходимо выполнить одну операцию: заменить символы входного сообщения соответствующими бинарными кодами.

*Алгоритм обратного преобразования.* На входе – сообщение в виде бинарной последовательности.

Анализируются  $l_{\min}$  начальных бинарных символов: осуществляется поиск в таблице соответствующего совпадения. Если такое будет найдено, то на выходе будет символ исходного алфавита с совпадающим кодом. После этого процедура повторяется, т. е. анализируются очередные  $l_{\min}$  символов. Если не найдено в таблице совпадения, переход к шагу 2.

Длина анализируемой последовательности увеличивается на 1 бит:  $l_{\min} + 1$ . Осуществляется поиск такой же бинарной комбинации в таблице. Если такая комбинация существует, на выходе распаковщика формируется соответствующий символ исходного алфавита, если нет – длина анализируемой последовательности увеличивается еще на один бит, и т. д.

По различным причинам при анализе очередной последовательности длиной  $l_{\max}$  совпадение в таблице может быть не найдено. Для нейтрализации подобных коллизий некоторые архиваторы содержат средства контроля ошибок, использующие корректирующие коды.

**❗ Пример.** Архиватор должен закодировать сообщение вида  $X = \langle a_1 a_1 a_9 a_9 a_9 a_5 a_1 a_5 a_5 a_1 \rangle$ . Легко заметить, что в этом сообщении встречаются только 3 различных символа, а соответствующие им вероятности будут равны:  $p(a_1) = 0,40$ ,  $p(a_5) = 0,30$ ,  $p(a_9) = 0,30$ .

В данном примере, по-существу, сформирована онлайн-таблица вероятностей. В подобных случаях процессы кодирования символов этой таблицы индивидуальны: каждый раз получаются новые бинарные последовательности, соответствующие символам исходного сообщения. Для однозначности обратного преобразования в таких случаях вместе со сжатым сообщением распаковщику следует переслать и такую таблицу бинарных кодов.

Далее рассмотрим более подробно процесс создания таблиц кодов. За исходный алфавит возьмем данные из примера на с. 73. Отсортируем таблицу символов в порядке убывания вероятностей. Разделим ее на две части (два подмножества), как показано ниже. Видно, что сумма вероятностей в обоих подмножествах одинакова и равна 0,5. Символам верхней части общей таблицы определим старший символ кода (1), нижней части – 0.

$p(a_4) = 0,20$	1
$p(a_5) = 0,15$	1
<u><math>p(a_9) = 0,15</math></u>	1
$p(a_6) = 0,12$	0
$p(a_2) = 0,10$	0
$p(a_8) = 0,10$	0
$p(a_1) = 0,05$	0
$p(a_7) = 0,05$	0
$p(a_{10}) = 0,05$	0
$p(a_3) = 0,03$	0

Далее в качестве исходного рассматриваем каждое из двух подмножеств. Выполняем рекурсивно одну и ту же процедуру. В частности, для первых трех символов таблицы (для первого подмножества) имеем после следующей итерации (ее деления на два меньших подмножества и определения следующих бинарных символов кода):

<u><math>p(a_4) = 0,20</math></u>	11
$p(a_5) = 0,15$	10
$p(a_9) = 0,15$	10

Последняя таблица в одном из подмножеств (первая строка) содержит только один элемент. Кодирование этого элемента закончено. Далее делим на два нижнюю часть последней таблицы и получим очередные символы соответствующих кодов:

$p(a_5) = 0,15$	101
$p(a_9) = 0,15$	100.

Далее переходим к кодированию символов после первого деления исходной таблицы (начиная с символа  $p(a_6)$ ).

Выполнив стандартные операции, получим таблицу бинарных кодов, как показано ниже:

$p(a_4) = 0,20$	11
$p(a_5) = 0,15$	101
$p(a_9) = 0,15$	100
$p(a_6) = 0,12$	011
$p(a_2) = 0,10$	010
$p(a_8) = 0,10$	0011
$p(a_1) = 0,05$	0010
$p(a_7) = 0,05$	0001
$p(a_{10}) = 0,05$	00000
$p(a_3) = 0,03$	00001

После расположения символов в порядке возрастания индексов символов алфавита получаем:

$$\begin{array}{ll} p(a_1) = 0010 & p(a_6) = 011 \\ p(a_2) = 010 & p(a_7) = 0001 \\ p(a_3) = 00001 & p(a_8) = 0011 \\ p(a_4) = 11 & p(a_9) = 100 \\ p(a_5) = 101 & p(a_{10}) = 00000 \end{array}$$

Как видим, действительно, символам с большими вероятностями соответствуют коды меньшей длины ( $l_{\min} = 2$ ) и наоборот:  $l_{\max} = 5$ . Замечаем, что выполнено основное требование: все кодовые комбинации разные; а также выполнено «требование префикса»: ни одна из кодовых комбинаций меньшей длины не является началом кодовой комбинации большей длины.

Именно последняя таблица используется в неизменном виде (речь о кодах) в процессах прямого и обратного преобразований.

*Прямое преобразование.* Итак, требуется сжать сообщение  $X = \langle a_1 a_1 a_9 a_9 a_9 a_5 a_1 a_5 a_5 a_1 \rangle$  (10 символов). Заменяя символы соответствующими им кодами из таблицы, на выходе получим  $X_n = 0010001010010010010100101011010010$ . Как видим, общая длина (объем после сжатия) сообщения составляет 34 бита. Если бы каждый символ сообщения  $X$  заменялся некоторым кодом, подобным ASCII, то длина его составила бы 80 бит ( $10 \times 8$ ).

*Обратное преобразование.* На входе имеем бинарную последовательность  $Y_n = 0010001010010010010100101011010010$ .

Анализируются начальные 2 символа ( $l_{\min} = 2$ ) этой последовательности: 00. Совпадающих комбинаций в таблице нет.

Длину анализируемой последовательности увеличиваем на один бит: 001. Совпадение не найдено.

Анализируем 4-битовую комбинацию: 0010. Этой комбинации в таблице соответствует символ исходного алфавита « $a_1$ ». На выходе распаковщика будет именно этот символ.

Анализируется очередные 2 символа ( $l_{\min} = 2$ ): 00, и т. д.

Теперь предположим, что во входном сообщении изменился только один символ (первый):  $Y_n = 1010001010010010010100101011010010$ . А распаковщик не содержит средств для ее обнаружения.

В таком случае первой совпадающей комбинацией будет 101: « $a_5$ ». Следующей – « $a_7$ », за ней « $a_2$ », и т. д.

Следуя логике рассуждений, нетрудно подсчитать коэффициент компрессии для нашего примера:

$$R = 34/80 \text{ либо } R' = (80 - 34)/80 = 46/80.$$

**Метод Хаффмана.** Отличается от метода Шеннона – Фано лишь в части кодирования символов исходного алфавита.

В данном случае бинарные коды создаются на основе дерева, ветви которого обозначаются бинарными символами.

*Бинарным кодом символа исходного алфавита будет последовательность обозначений ветвей дерева от корня до листа, соответствующего этому символу.*

Построение дерева начинается с сортирования символов исходного алфавита в порядке убывания. Далее выбираются два символа ( $a_i, a_j$ ) с наименьшими вероятностями ( $p(a_i), p(a_j)$ ) и объединяются в узел. Ветви этого узла обозначаются «1» и «0». Данный узел рассматривается далее как новый, виртуальный символ ( $a_{ij}$ ), которому будет соответствовать вероятность  $p(a_{ij}) = p(a_i) + p(a_j)$ . Такой виртуальный символ будет рассматриваться далее наравне с остальными символами исходного алфавита. Два его потомка из дальнейшего рассмотрения исключаются. Создаются новые узлы дерева по тому же принципу. Корень дерева образуют два символа с наибольшими вероятностями.

**! Пример.** Пусть имеется алфавит из пяти символов, который в отсортированном виде выглядит так:

$$p(a_4) = 0,35$$

$$p(a_2) = 0,20$$

$$p(a_1) = 0,20$$

$$p(a_5) = 0,15$$

$$p(a_3) = 0,10$$

Строим дерево. Два нижних символа создают первый узел этого дерева, который обозначаем  $a_{53}$ . Ему соответствует вероятность  $p(a_{53}) = 0,25$ .

$$\begin{array}{l} p(a_5) = 0,15 \\ p(a_3) = 0,10 \end{array} \begin{array}{c} \diagup \\ \diagdown \end{array} p(a_{53}) = 0,25$$

Верхней ветви этого узла будет соответствовать «1», нижней – «0».

Далее рассматриваем алфавит, состоящий из следующих символов:  $p(a_4) = 0,35, p(a_2) = 0,20, p(a_1) = 0,20, p(a_5) = 0,15, p(a_{53}) = 0,25$ .

Строго говоря, новая последовательность опять должна быть отсортирована. Далее снова нужно объединить два символа с наименьшими вероятностями, и т. д. С другой стороны, при построении дерева можно придерживаться следующего правила: на одном уровне иерархии дерева в узлы объединяются символы с примерно одинаковыми вероятностями; при обозначении ветвей двоичная единица присваивается символу с большей вероятностью.

Один из возможных вариантов дерева представлен на рис. 7.5. Обозначения узлов опущены.

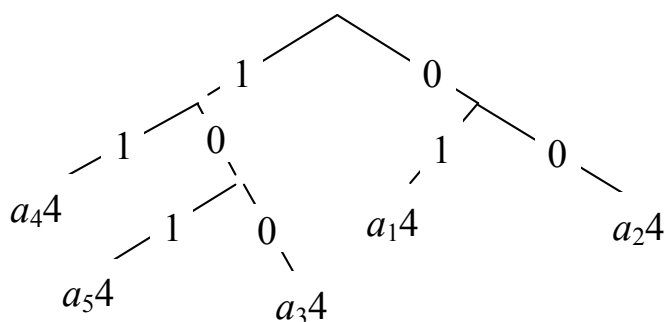


Рис. 7.5. Бинарное дерево, созданное по методу Хаффмана

Ниже представлена таблица соответствий между символами исходного алфавита и их бинарными кодами:

$a_1$	01
$a_2$	00
$a_3$	100
$a_4$	11
$a_5$	101

Как видно, для одного и того же алфавита по обоим рассмотренным статистическим методам могут быть созданы разные таблицы, т. е. одним и тем же символам могут соответствовать разные кодовые комбинации. Наилучшей является та таблица, которой соответствует минимальное значение интегрального коэффициента  $C$ :

$$C = \sum p(a_i) \cdot l_i, \quad (7.4)$$

где  $p(a_i)$  и  $l_i$  – соответственно вероятность появления символа и длина бинарного кода для этого символа.

Для данных из примера имеем:

$$C = 0,20 \cdot 2 + 0,20 \cdot 2 + 0,10 \cdot 3 + 0,35 \cdot 2 + 0,15 \cdot 3 = 2,1 \text{ бит.}$$

Это означает, что для выполненной кодировки одному символу исходного алфавита в среднем соответствует 2,1 бита.

Д. Хаффман составил дерево для алфавита английского языка, которое является оптимальным (наилучшим из возможных, так как ему соответствует наименьшее значение  $C$ ). В таблице представлены эти коды.

**Бинарные коды английского алфавита  
в соответствии с деревом Хаффмана**

Символ алфавита	Бинарный код	Символ алфавита	Бинарный код
E	100	M	00011
T	001	U	00010
A	1111	G	00001
O	1110	Y	00000
N	1100	P	110101
R	1011	W	011101
I	1010	B	011100
S	0110	V	1101001
H	0101	K	110100011
D	11011	X	110100001
L	01111	J	110100000
F	01001	Q	1101000101
C	01000	Z	1101000100

В текстах на английском языке наиболее часто встречается буква «е» (ей соответствует вероятность 0,13).

**Арифметический метод сжатия.** Рассмотренный выше метод Хаффмана эффективен, когда частоты появления символов пропорциональны  $1/2^i$  (где  $i$  – натуральное положительное число). Действительно, код Хаффмана для каждого символа всегда состоит из *целого числа бит*. В случае, когда частота появления символа равна 0,2, оптимальный код, соответствующий этому символу, должен иметь длину  $\log_2(0,2) = 2,3$  бита. Понятно, что префиксный код Хаффмана не может иметь такую длину, содержащую часть бита, т. е. в конечном итоге это приводит к ухудшению сжатия данных.

Арифметическое кодирование предназначено для того, чтобы решить эту проблему. *Основная идея арифметического метода*

сжатия заключается в том, чтобы присваивать коды не отдельным символам, а их последовательностям.

Таким образом, как и во всех энтропийных алгоритмах, исходной является информация о частоте встречаемости каждого символа алфавита. Алгоритмы прямого и обратного преобразования базируются на операциях с «рабочим отрезком».

*Рабочим отрезком* называется интервал  $[a; b]$  с расположенными на нем точками. Причем точки расположены таким образом, что длины образованных ими отрезков пропорциональны (или равны) частоте (вероятности) появления соответствующих символов.

**! Пример.** Дана информационная последовательность «молоко». Рассчитываем статистику:

$$p(\text{м}) = 0,1666$$

$$p(\text{л}) = 0,1666$$

$$p(\text{к}) = 0,1666$$

$$p(\text{о}) = 0,5.$$

Строим основной рабочий отрезок (на нулевом шаге (рис. 7.6)).

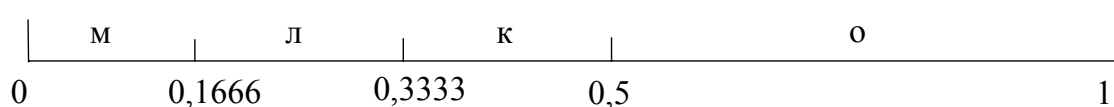


Рис. 7.6. Разметка и точки основного рабочего отрезка

Как показано на рис. 7.6, при инициализации алгоритма  $a = 0$   $b = 1$  ( $a_0 = 0$ ,  $b_0 = 1$ ).

*Прямое преобразование (сжатие).* Один шаг сжатия (кодирования) заключается в простой операции: берется кодируемый символ, для него ищется соответствующий участок на рабочем отрезке. Найденный участок становится новым рабочим отрезком. Его тоже необходимо разбить с помощью точек.

Это и последующие разбиения отрезка (на шаге  $i$ ) подразумевают определение новых значений верхней ( $H_i$ ) и нижней ( $L_i$ ) границ для всего участка и осуществляются по следующим правилам:

$$\left. \begin{aligned} H_i(\alpha_j) &= L_{i-1} + (H_{i-1} - L_{i-1}) \cdot H(\alpha_j)_0; \\ L_i(\alpha_j) &= L_{i-1} + (H_{i-1} - L_{i-1}) \cdot L(\alpha_j)_0; \end{aligned} \right\} \quad (7.5)$$

где  $\alpha_j$  –  $j$ -й символ сжимаемой последовательности;  $L_{i-1}$  и  $H_{i-1}$  – соответственно нижняя и верхняя границы рабочего отрезка на



$(i - 1)$ -м шаге;  $L(\alpha_j)_0$  и  $H(\alpha_j)_0$  – соответственно исходные нижняя и верхняя границы символа  $\alpha_j$ .

В нашем примере на первом шаге берется первый символ последовательности: «м» ( $\alpha_1 = \text{«м»}$ ) и ищется соответствующий участок на рабочем отрезке. Легко понять из рис. 7.6, что этот участок соответствует интервалу  $[0; 0,1666]$ . Он становится новым рабочим отрезком и опять разбивается согласно статистике и соотношениям (7.5):

$$H_1(\text{м}) = L_0 + (H_0 - L_0) \cdot H(\alpha_j = \text{м})_0 = 0 + (0,1666 - 0) \cdot 0,1666 = 0,0277;$$

$$L_1(\text{м}) = L_0 + (H_0 - L_0) \cdot L(\alpha_j = \text{м})_0 = 0 + (0,1666 - 0) \cdot 0 = 0;$$

$$H_1(\text{л}) = L_0 + (H_0 - L_0) \cdot H(\alpha_j = \text{л})_0 = 0 + (0,1666 - 0) \cdot 0,3333 = 0,055555;$$

$$L_1(\text{л}) = L_0 + (H_0 - L_0) \cdot L(\alpha_j = \text{л})_0 = 0 + (0,1666 - 0) \cdot 0,1666 = 0,0277$$

и т. д.

После выполнения всех вычислений на первом шаге получим разметку рабочего отрезка ( $a_1 = L_1 = 0$ ,  $b_1 = H_1 = 0,1666$ ) в соответствии с рис. 7.7.

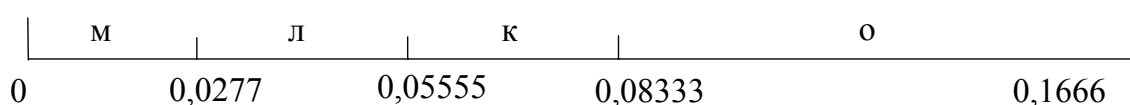


Рис. 7.7. Разметка и точки рабочего отрезка после анализа первого символа («м») последовательности (молоко)

Анализируем следующий символ (шаг 2). На новом рабочем отрезке очередному символу (о) соответствует интервал  $[a_2 = L_2 = 0,083333; b_2 = H_2 = 0,1666]$ . Для следующего шага он станет рабочим отрезком (рис. 7.8).

Строго говоря, на первом шаге не было необходимости рассчитывать новые границы для каждого символа. Как видим, это только увеличивает время анализа. Достаточно было определить новые верхнюю и нижнюю границы лишь для очередного (на шаге 2) символа: «о». Мы здесь привели расчеты только для того, чтобы помочь читателю понять процесс вычислений.

Поскольку очередным из анализируемых символов (на шаге 3) будет буква «л», достаточно в соответствии с (7.5) определить новые границы:

$$\begin{aligned} H_3(\text{л}) &= L_2 + (H_2 - L_2) \cdot H(\alpha_j = \text{л})_0 = \\ &= 0,08333 + (0,1666 - 0,08333) \cdot 0,3333 = 0,1111, \end{aligned}$$

$$L_3 (\text{л}) = L_2 + (H_2 - L_2) \cdot L(\alpha_j = \text{л})_0 = \\ = 0,08333 + (0,1666 - 0,08333) \cdot 0,1666 = 0,972.$$

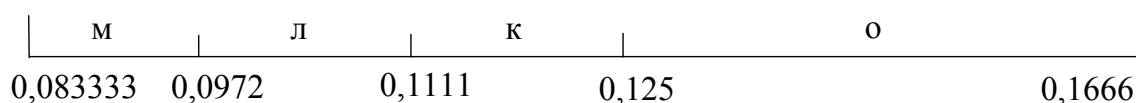


Рис. 7.8. Рабочий отрезок после анализа первых 2 символов

Новый рабочий отрезок определим как интервал  $[a_3 = L_3 = 0,0972; b_3 = H_3 = 0,1111]$ , так как он соответствует символу «л». Вид разметки рабочего отрезка представлен на рис. 7.9.

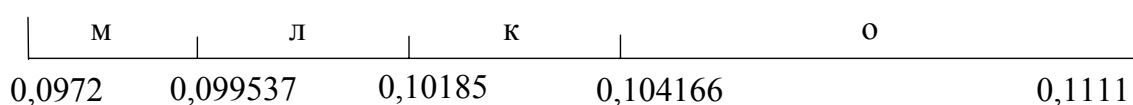


Рис. 7.9. Рабочий отрезок с разметками после анализа последовательности «мол»

Продолжаем анализировать символы сообщения. Находим на рабочем отрезке интервал  $[0,104166; 0,1111]$ , который соответствует очередному символу: «о». Производим разметку нового рабочего отрезка (рис. 7.10).

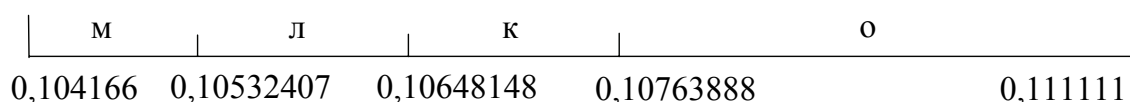


Рис. 7.10. Разметка рабочего отрезка на четвертом шаге

Следующий символ последовательности – «к». На рабочем отрезке данному символу принадлежит интервал  $[0,10648148; 0,10763888]$ . Новый рабочий отрезок имеет вид, как показано на рис. 7.11.

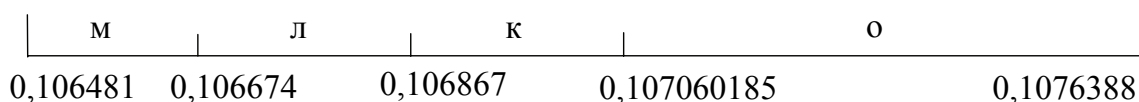


Рис. 7.11. Разметка рабочего отрезка на пятом шаге прямого преобразования

У нас остался последний символ: «о». Результатом кодирования цепочки символов является любое число с итогового рабочего отрезка  $[0,107060185; 0,10763888]$ . Обычно таким числом является

нижняя граница указанного отрезка. Поступим мы в соответствии с указанным принципом: возьмем число с меньшим количеством знаков после запятой.

Таким образом, итогом сжатия входной последовательности «молоко» будет число 0,107060185 (для упрощения дальнейших вычислительных операций округлим его до **0,1071**).

*Декомпрессия.* Для восстановления исходного сообщения необходима информация:

- о значении числа, являющегося итогом сжатия сообщения (в нашем случае 0,1071);
- о количестве символов в сжатом сообщении;
- о вероятностных параметрах всех символов исходного сообщения (таблица вероятностей).

Как и при сжатии, вначале необходимо начальный рабочий отрезок  $[0; 1)$  разбить на интервалы, длины которых равны вероятностям появления соответствующих символов.

Анализ будем проводить с использованием конкретных данных, взятых из последнего примера.

Итак, в качестве исходного участка для обратного преобразования принимается исходный участок для прямого преобразования с одинаковыми точками его разбиения (рис. 7.12). Ниже он повторяется.

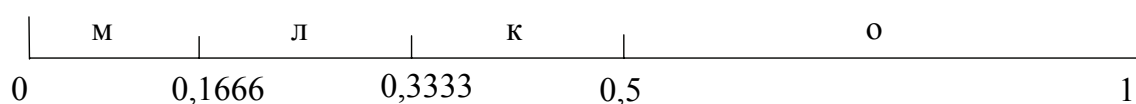


Рис. 7.12. Разметка рабочего отрезка на первом шаге обратного преобразования

На каждом шаге обратного преобразования выбираем отрезок, в который попадает текущее число (код). Символ, который соответствует данному отрезку, является очередным символом восстановленного (распакованного) сообщения.

В общем случае код символа, восстанавливаемого на шаге  $i$ , вычисляется соотношением

$$\text{код } i = [\text{код } (i - 1) - L(\alpha_{i-1})_0] / [H(\alpha_{i-1})_0 - L(\alpha_{i-1})_0], \quad (7.6)$$

где код  $(i - 1)$  – число, анализ которого производится на предыдущем шаге –  $(i - 1)$ -м;  $H(\alpha_{i-1})_0$  и  $L(\alpha_{i-1})_0$  – соответственно верхняя и нижняя исходные границы символа сообщения, восстановленного на предыдущем шаге.

Первый шаг: определяем интервал, в который попадает начальное число: 0,1071 (код 1 = 0,1071). Это интервал [0; 0,166666] и ему соответствует символ «м». Данный интервал становится новым рабочим отрезком, а первый символ восстановленного сообщения – «м».

Второй шаг: производим вычисление в соответствии с (7.6):

$$\begin{aligned}\text{код } 2 &= [\text{код } 1 - L(\alpha_1 = \text{м})_0] / [H(\alpha_1 = \text{м})_0 - L(\alpha_1 = \text{м})_0] = \\ &= (0,1071 - 0) / (0,1666 - 0) = 0,643.\end{aligned}$$

Вычисленный код соответствует символу «о».

Третий шаг: выполняем известное вычисление:

$$\begin{aligned}\text{код } 3 &= [\text{код } 2 - L(\alpha_2 = \text{о})_0] / [H(\alpha_2 = \text{о})_0 - L(\alpha_1 = \text{м})_0] = \\ &= (0,643 - 0,5) / (1 - 0,5) = 0,286.\end{aligned}$$

Полученное число соответствует символу «л».

Аналогичным образом производятся и последующие вычислительно-аналитические операции.

Другой вариант алгоритма распаковки «сжатого» сообщения основан на *свойстве рекуррентности прямого преобразования*. Производя на каждом шаге вычисление новых границ рабочего участка в соответствии с восстановленным на данном шаге символом (наподобие тех вычислений, которые мы выполняли при сжатии) и анализируя, на какой из отрезков этого участка попадает входное число (в нашем примере это 0,1071), восстанавливаем исходное сообщение.

Таким образом, на первом шаге это число соответствует букве «м», которая попадает в интервал [0; 0,1666]. Этот новый диапазон будет рабочим участком на шаге 2.

На втором шаге интервал [0; 0,1666] опять разбивается согласно статистике (рис. 7.13).

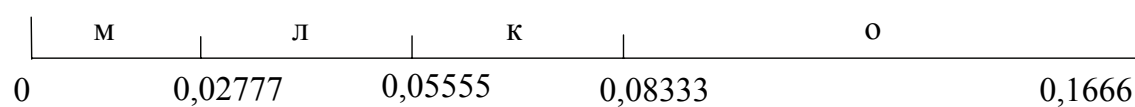


Рис. 7.13. Рабочий отрезок с разметкой после первого шага восстановления

Снова ищем интервал, в который попадает наше число 0,1071. Выбираем последний интервал [0,083333; 0,16666] в качестве рабочего отрезка (рис. 7.14) для последующих шагов, а символ «о» добавляем к восстановленному сообщению.

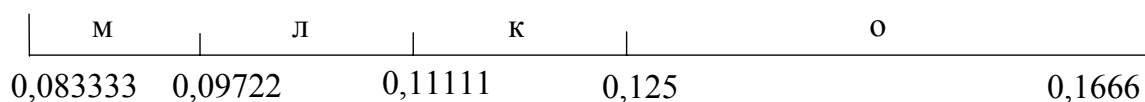


Рис. 7.14. Вид рабочего отрезка после 2 шагов

Определяем следующее соответствие. Число 0,1071 попадает в интервал  $[0,097222; 0,111111]$ , который становится очередным рабочим отрезком (рис. 7.15). К восстановленной части сообщения «мо» присоединяем символ интервала «л».

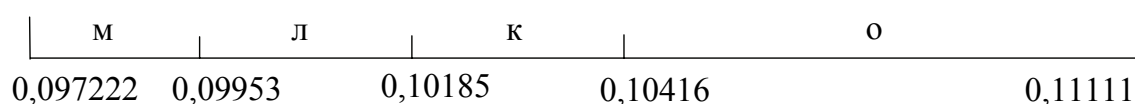


Рис. 7.15. Вид рабочего отрезка после 3 шагов  
обратного преобразования

Следующий интервал –  $[0,104166; 0,111111]$  – соответствует символу «о». Его деление показано на рис. 7.16.

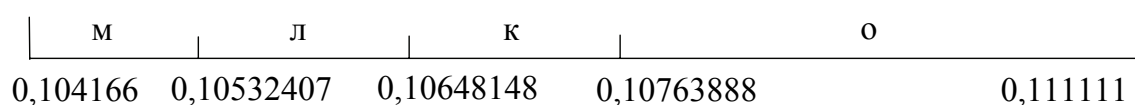


Рис. 7.16. Рабочий отрезок на 4 шаге

На пятом шаге наше число попадает в интервал символа «к». Данный символ присоединяем к тексту сообщения, а интервал  $[0,104166; 0,1076388]$  становится рабочим отрезком (рис. 7.17).

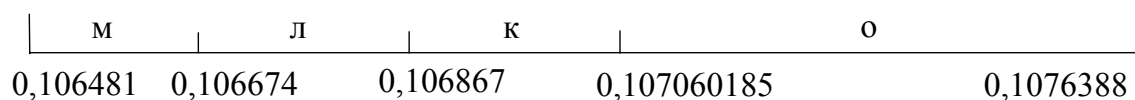


Рис. 7.17. Вид рабочего отрезка на последнем шаге обратного преобразования сообщения «молоко»

Так как нам известно, что символов в сообщении было 6, то на последнем шаге необходимо только определить недостающий символ. Таким символом становится символ «о». Полученное сообщение «молоко» и есть восстановленная («распакованная») последовательность.

**! Пример.** Дана информационная последовательность: «переговоры».

Подсчитываем частоту встречаемости каждого символа в сообщении. Разбиваем интервал  $[0; 1)$  на 7 отрезков. Длина каждого отрезка – вероятность встречаемости соответствующего символа в информационной последовательности. Далее необходимо осуществить ряд уже известных читателю операций.

*Прямое преобразование.* Первый шаг: первым выбираем интервал из рабочего отрезка, который соответствует первому символу в сообщении. Это символ «п». Границы интервалов, соответствующие отдельным символам, здесь и далее для компактности сведены в таблицы. Границы анализируемого на данном шаге символа выделены жирным шрифтом.

СИМВОЛ	<b>п</b>	е	р	г	о	в	ы
0,00000	<b>0,10000</b>	0,30000	0,50000	0,60000	0,80000	0,90000	1,00000

Таким образом, рабочий участок после первого шага составляет диапазон  $[0; 0,1]$ .

Второй шаг: разбиваем полученный отрезок  $[0; 0,1]$  на интервалы. Определяем отрезок, который соответствует второму символу: «е». Это отрезок  $[0,1; 0,3]$ .

СИМВОЛ	п	<b>е</b>	р	г	о	в	ы
0,00000	<b>0,01000</b>	<b>0,03000</b>	0,05000	0,06000	0,08000	0,09000	0,10000

Третий шаг: разделяем текущий рабочий интервал  $[0,01; 0,03]$  на отрезки согласно статистике появления символов. Находим отрезок, который соответствует символу «р». Этот отрезок принимаем за новый рабочий и делим его на 7 частей.

СИМВОЛ	п	е	<b>р</b>	г	о	в	ы
0,01000	0,01200	<b>0,01600</b>	<b>0,02000</b>	0,02200	0,02600	0,02800	0,03000

Четвертый шаг: из текущего интервала выбираем отрезок  $[0,0164; 0,0172]$ , который соответствует символу «е». Актуальная таблица представлена ниже.

СИМВОЛ	п	<b>е</b>	р	г	о	в	ы
0,01600	<b>0,01640</b>	<b>0,01720</b>	0,01800	0,01840	0,01920	0,01960	0,02000

Пятый шаг: продолжаем вычисление для следующего символа: «г». На рабочем отрезке выбираем диапазон  $[0,0168; 0,01688]$ .

СИМВОЛ	п	Е	р	<b>г</b>	о	в	ы
0,01640	0,01648	0,01664	<b>0,01680</b>	<b>0,01688</b>	0,01704	0,01712	0,01720

Шестой шаг: разбиваем полученный интервал  $[0,0168; 0,01688]$  на отрезки. Выбираем отрезок, который соответствует символу «о».

СИМВОЛ	п	е	р	г	о	в	ы
0,016800	0,016808	0,016824	0,016840	<b>0,016848</b>	<b>0,016864</b>	0,016872	0,016880

Седьмой шаг: определяем интервал для символа «в». В качестве рабочего берем отрезок [0,016861; 0,016862].

СИМВОЛ	п	е	р	г	о	в	ы
0,016848	0,016850	0,016853	0,016856	0,016858	<b>0,016861</b>	<b>0,016862</b>	0,016864

Восьмой шаг: выбранный интервал [0,0168618; 0,0168621], который соответствует символу «о», используем на следующем этапе.

СИМВОЛ	п	е	р	г	о	в	ы
0,0168608	0,0168610	0,0168613	0,0168616	<b>0,0168618</b>	<b>0,0168621</b>	0,0168622	0,0168624

Девятый шаг: для анализа символа «р» используем отрезок [0,01686186; 0,01686192] из рабочего интервала.

СИМВОЛ	п	е	р	г	о	в	ы
0,01686176	0,01686179	<b>0,01686186</b>	<b>0,01686192</b>	0,01686195	0,01686202	0,01686205	0,01686208

Десятый шаг: Определяем последний интервал. Последний символ – «ы». Разбиваем полученный интервал [0,01686186; 0,01686192] на отрезки. Выбираем отрезок, который соответствует данному символу.

СИМВОЛ	п	е	р	г	о	в	ы
0,01686186	0,01686186	0,01686188	0,01686189	0,01686189	0,01686191	<b>0,01686191</b>	<b>0,01686192</b>

Процесс преобразования (сжатия) закончен. Предположим, что конечным результатом будет верхняя граница последнего диапазона: 0,01686192. Это число и есть сжатое сообщение.

*Декомпрессия.* Для восстановления исходного сообщения необходимо:

- вещественное число – 0,01686192;
- количество символов в сообщении – 10;
- таблица вероятностей встречаемости символов в этом сообщении.

Как и при сжатии, необходимо исходный интервал [0; 1) разбить на отрезки, длины которых равны вероятностям встречаемости символов. Далее на каждом шаге выбираем отрезок, в который попадает наше число (0,01686192). Символ, который соответствует данному отрезку, и есть очередной символ восстанавливаемого сообщения.

Первый шаг: первым выбираем отрезок исходного рабочего участка, в который попадает число 0,01686192. Это интервал, которому соответствует символ «п», что видно из нижеследующей таблицы.

СИМВОЛ	п	е	р	г	о	в	ы
0,00000	0,10000	0,30000	0,50000	0,60000	0,80000	0,90000	1,00000

Второй шаг: разбиваем текущий интервал  $[0; 0,1]$  на отрезки. Определяем отрезок, в который попадает число 0,01686192. Символ «е» будет следующим символом на выходе распаковщика, а соответствующий интервал  $[0,1; 0,3]$  – рабочим отрезком.

СИМВОЛ	п	е	р	г	о	в	ы
0,00000	0,01000	0,03000	0,05000	0,06000	0,08000	0,09000	0,10000

Третий шаг: число 0,01686192 попадает в интервал  $[0,016; 0,02]$ . На выходе – символ «р».

СИМВОЛ	п	е	р	г	о	в	ы
0,01000	0,01200	0,01600	0,02000	0,02200	0,02600	0,02800	0,03000

Процесс продолжается до полного восстановления информации.



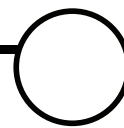
### **Вопросы для контроля и самоконтроля**

1. В чем заключается суть арифметического метода сжатия?
2. Поясните принцип определения границ отрезков на каждом шаге (на каждом рабочем интервале) прямого и обратного преобразования.
3. Осуществить сжатие и распаковку следующих сообщений арифметическим методом:
  - а) «мама мыла раму»;
  - б) «насос»;
  - в) «университет»;
  - г) «шалаш»;
  - д) «информатика»;
  - е) «246824328»;
  - ж) «101101001011010»;
  - з) ваша фамилия.



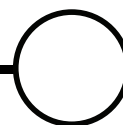
## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

---



1. Урбанович, П. П. Информационная безопасность и надежность систем: учеб.-метод. пособие для студентов специальности 1-40 01 02-03 «Информационные системы и технологии» / П. П. Урбанович, Д. М. Романенко, Е. В. Романцевич. – Минск: БГТУ, 2007. – 90 с.
2. Леонов, А. П. Безопасность автоматизированных банковских и офисных систем / А. П. Леонов, К. А. Леонов, Г. В. Фролов. – Минск: НКП Беларуси, 1996. – 280 с.
3. Шеннон, К. Работы по теории информации и кибернетике / К. Шеннон. – М.: Изд. иностр. лит., 1963. – 830 с.
4. Скляр, Б. Цифровая связь. Теоретические основы и практическое применение / Б. Скляр; пер. с англ. – 2-е изд. – М.: Вильямс, 2003. – 1104 с.
5. Системы мобильной связи: учеб. пособие / В. П. Ипатов [и др.] // Сайт Ташкентского университета информационных технологий [Электронный ресурс]. – Режим доступа: [http://www.uftuit.uzpak.uz/Tatulib/book/sistemi\\_mob\\_svyazi/sistemi\\_mobilnoj\\_svyazi.htm](http://www.uftuit.uzpak.uz/Tatulib/book/sistemi_mob_svyazi/sistemi_mobilnoj_svyazi.htm). – Дата доступа: 05.02.12.
6. Гуров, И. П. Основы теории информации и передачи сигналов / И. П. Гуров. – СПб.: ВНУ-Санкт-Петербург, 2000. – 97 с.

# ОГЛАВЛЕНИЕ



Предисловие .....	3
1. Фундаментальные понятия и определения из области информационной безопасности и надежности систем .....	4
2. Потенциальные угрозы безопасности информации в ИВС. Объекты и методы защиты информации .....	9
3. Общая характеристика, структура и математическое описание каналов передачи и хранения информации .....	17
4. Понятие информации. Энтропия источника сообщений .....	20
5. Качество информации. Энтропийная оценка потерь при передаче информации .....	25
6. Методы структурной, информационной и временной избыточности в ИВС .....	29
6.1. Общая характеристика ИС с избыточностью .....	29
6.2. Методы и средства перемежения данных. Использование перемежителей/деперемежителей в системах передачи данных .....	46
7. Сжатие информации как метод повышения ее безопасности и целостности .....	54
7.1. Цели использования и классификация методов сжатия сообщений .....	54
7.2. Символориентированные и словарные методы сжатия .....	56
7.3. Вероятностные (статистические) методы .....	72
Список использованных источников .....	89

Учебное издание

**Урбанович Павел Павлович**  
**Шиман Дмитрий Васильевич**

**ЗАЩИТА ИНФОРМАЦИИ  
И НАДЕЖНОСТЬ  
ИНФОРМАЦИОННЫХ  
СИСТЕМ**

Пособие

Редактор *О. П. Приходько*  
Компьютерная верстка *Е. В. Ильченко*  
Корректор *О. П. Приходько*

Подписано в печать 06.07.2014. Формат 60×84<sup>1</sup>/<sub>16</sub>.  
Бумага офсетная. Гарнитура Таймс. Печать офсетная.  
Усл. печ. л. 5,3. Уч.-изд. л. 5,5.  
Тираж 150 экз. Заказ .

Издатель и полиграфическое исполнение:  
УО «Белорусский государственный технологический университет».  
Свидетельство о государственной регистрации издателя,  
изготовителя, распространителя печатных изданий  
№ 1/227 от 20.03.2014.  
ЛП № 02330/12 от 30.12.2013.  
Ул. Свердлова, 13а, 220006, г. Минск.