КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ПОСОБИЕ ДЛЯ ПОДГОТОВКИ К ЭКЗАМЕНУ ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ» РАЗДЕЛ «КОДИРОВАНИЕ ИНФОРМАЦИИ»

КАЗАНЬ 2012

УДК 519.72

Печатается по решению Редакционно-издательского совета ФГАОУВПО «Казанский (Приволжский) федеральный университет»

методической комиссии института вычислительной математики и информационных технологий
Протокол № 7 от 15 марта 2012 г.

заседания кафедры информатики и вычислительных технологий
Протокол № 6 от 16 февраля 2012 г.

Рецензенты:
канд. техн. наук, доц. КНИТУ Шлеймович М.И.
канд. физ.-мат. наук, доц. КГУ Широкова О.А.

Чепкунова Е.Г.

Название: Пособие к подготовке к экзамену по дисциплине «Теоретические остновы информатики». Раздел «Кодирование информации»: Учеб.пособие / Е.Г.Чепкунова. – Казань: Казанский университет, 2012. – 92 с.

Данное пособие предназначено студентам специальности «Информатика» для подготовки к экзамену по дисциплине «Теоретические основы информатики». Пособие содержит адаптированный курс лекций по темам раздела «Кодирование информации», методические рекомендации для подготовки ответа на экзаменационные вопросы, задания для самопроверки.

Оглавление

Введение

Данное пособие предназначено студентам специальности «Информатика» для подготовки к экзамену по дисциплине «Теоретические основы информатики». Пособие содержит теоретический материал раздела «Кодирование информации» соответствующий содержанию указанной дисциплины в Государственных образовательных стандартах.

Формулировка каждого параграфа данного пособия совпадает в вопросом экзамена. Параграф включает в себя план ответа на один вопрос итогового экзамена по указанной дисциплине, краткий теоретический материал, который можно использовать при ответе, и вопросы для самопроверки. Представленная информация не является исчерпывающей при подготовке к экзамену по курсу «Теоретические основы информатики» — она содержит лишь методические указания для подготовки к ответу и основное содержание ответа. При необходимости, студент может воспользоваться дополнительной информацией источников, указанных в разделе «Литература».

Теория кодирования: история возникновения.

При ответе на данный вопрос необходимо рассказать об истории возникновения различных способов кодирования и шифрования информации, сформулировать и пояснить на примерах требования к шифру Ф.Бекона.

Теория кодирования — это раздел теории информации, изучающий способы отображения дискретных сообщений сигналами в виде определенных сочетаний символов.

С глубокой древности люди искали эффективные способы передачи информации:

- Движение факелов использовал древнегреческий историк Полибий (II в. до н.э.};

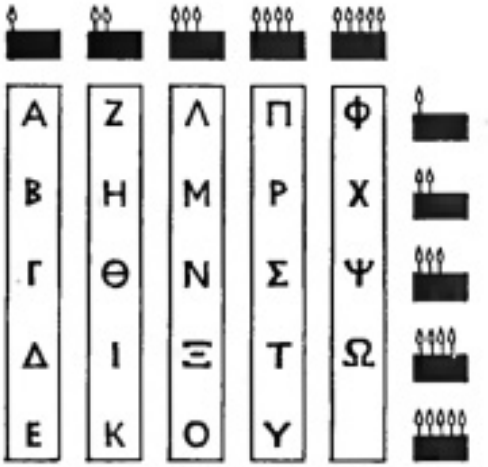


Рис.1 Схема кодирования букв греческого алфавита с помощью двух групп факелов.

- Оптический телеграф – семафор – впервые использовал

Клод Шапп в 1791 г.;

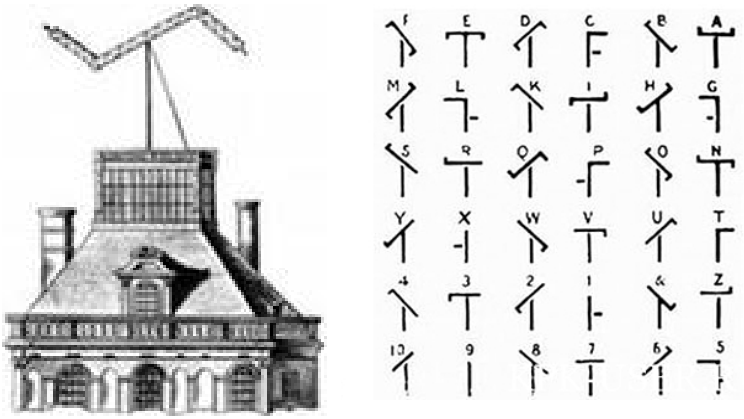


Рис.2 Оптический семафор К.Шаппа и его телеграфный алфавит.

- Движение электромагнитной стрелки в электромагнитных телеграфных аппаратах впервые применили русский физик П.Л. Шиллинг (1832) и профессора Гёттингенского университета Вебер и Гаусс (1833);

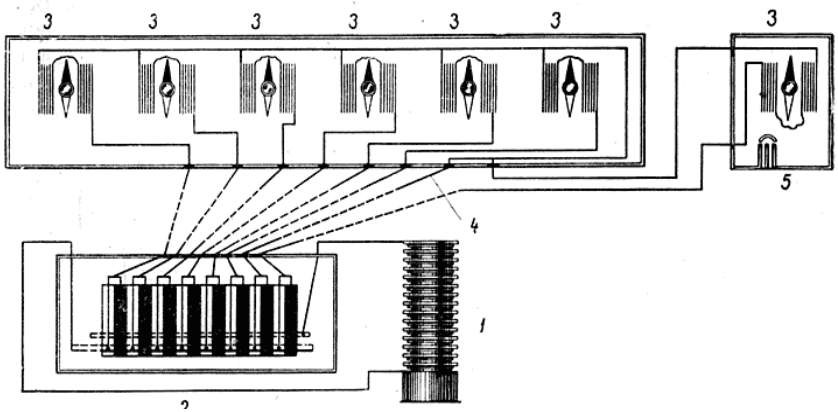


Рис.3 Схема электромагнитного телеграфа П.Л.Шиллинга (1 — источник тока, 2 — клавиатура, 3 — магнитные стрелки, 4 — провод обратной связи, 5 — вызывное устройство)

- Азбука и телеграфный аппарат Самюэла Морзе (1837);

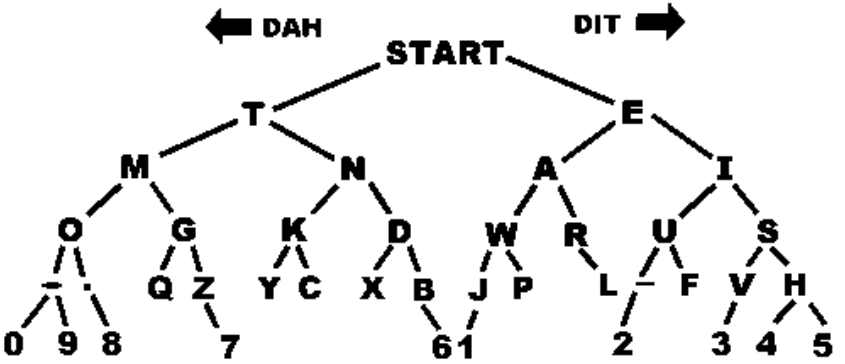


Рис. 4 Дерево кода Морзе - направо точка, налево тире.

- Международный флажковый код для передачи информации оптическими сигналами впервые ввел капитан Фредерик Марьят в 1861 г. на основе свода корабельных сигналов;

Рис.5 Морская азбука сигнальных флажков

- Беспроволочный телеграф (радиопередатчик) был изобретен А.С.Поповым в 1895 г. И Маркони в 1897 г. независимо друг от друга;

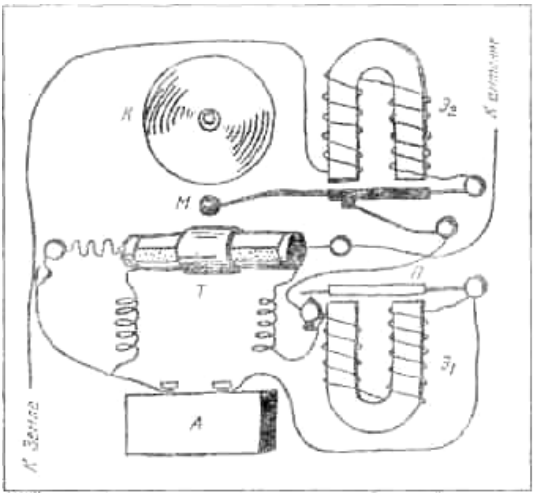


Рис.6а Схема прибора Попова: Т — трубочка с железными опилками; К — колокол звонка; М — молоточек; А — аккумулятор, подающий ток в трубочку с опилками; Э1 и Э2 — электромагниты; П — железная пластинка.

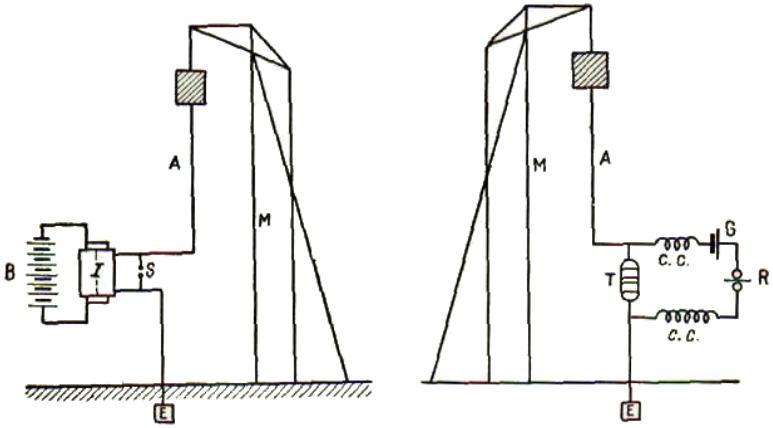


Рис. 6б Схема беспроволочного телеграфа Маркони. Слева

отправительная станция (передатчик): B — батарея аккумуляторов, I — индукционная катушка, S — вибратор, A — антенна с металлическим баком на конце, M — мачта, E — цинковыq лист, зарытый в землю. Справа — приемная станция: T — трубочка с никелевыми и серебряными опилками; G — батарея, подающая в трубочку ток; R -электрическийзвонок

- Беспроволочный телефон, телевидение (1935), затем и ЭВМ – новые средства связи, появившиеся в XX в., с которыми связана новая эпоха в информатизации общества.

Одновременно с потребностью передавать информацию люди искали способы скрыть смысл передаваемых сообщений от посторонних любопытных глаз. Императоры, торговцы, политики и шпионы искали способы шифрования своих посланий. Образцы тайнописи можно встретить еще у Геродота (V в. до н. э.). К тайнописи – криптографии прибегал Гай Юлий Цезарь, заменяя в своих тайных записях одни буквы другими. Использовали шифрование не только древнегреческие жрецы, но и ученые Средневековья: математики итальянец Джероламо Кардано и француз Франсуа Виет, нидерландский гуманист, историк, юрист Гроций, выдающийся английский философ Фрэнсис Бэкон. Отцом криптографии считается архитектор Леон Баттиста Альберти (1404-1472), который ввел шифрующие коды и многоалфавитные подстановки.

Сэр Фрэнсис Бэкон (1561 – 1626), автор двухлитерного кода, доказал в 1580 г., что для передачи информации достаточно двух знаков. Также Ф.Бэкон сформулировал требования к шифру:

1. Шифр должен быть **несложен**, прост в работе;

2. Шифр должен быть **надежен**, труден для дешифровки

посторонним;

3. Шифр должен быть **скрытен**, по возможности не должен вызывать подозрений.

Шифры Бэкона – сочетание шифрованного текста с дезинформацией в виде нулей. Таким образом, двузначные коды и шифры использовались задолго до появления ЭВМ.

Новый толчок развитию теории кодирования дало создание в 1948 году Клодом Эльвудом Шенноном (1916 — 2001) теории информации. Идеи, изложенные Шенноном в статье «Математическая теория связи», легли в основу современных теорий и техник обработки, передачи и хранения информации. Результаты его научных исследований способствовали развитию помехоустойчивого кодирования и простых методов декодирования сообщений.

Вопросы для самопроверки:

1. Перечислите основные способы передачи информации, расположив их в хронологическом порядке.

2. Расскажите об истории возникновения криптографии.

3. Сформулируйте основные требования к шифру Ф.Бекона.

Теория кодирования: цели, задачи и основные понятия.

При ответе на данный вопрос необходимо рассказать о целях теории кодирования, сформулировать основную задачу кодирования, определить понятия «код», «первичный алфавит», «вторичный алфавит», «кодирование», «декодирование».

На сегодняшний день основными **целями** теории кодирования являются:

- Разработка принципов наиболее экономного представления информации;

- Согласование параметров передаваемой информации с особенностями канала связи;

- Разработка приемов повышения надежности передачи информации.

Задача кодирования – это задача перевода дискретного сообщения из одного алфавита в другой. Причем такое преобразование не должно приводить к потере информации. Алфавит, с помощью которого представляется информация до преобразования называется **первичным**, а алфавит конечного представления – **вторичным**.

При определении понятия «код» используют два подхода. С одной стороны, **код — это правило**, описывающее соответствие знаков или их сочетаний первичного (исходного) алфавита знакам или их сочетаниям вторичного алфавита. Также кодом называют **набор знаков** вторичного алфавита, используемый для представления знаков или их сочетаний первичного алфавита.

Кодирование – это перевод информации, представленной символами первичного алфавита в последовательность кодов.

Декодирование – операция обратная кодированию — перевод последовательности кодов в соответствующий набор символов первичного алфавита.

Кодер – устройство, обеспечивающее выполнение операции кодирования.

Декодер – устройство, производящее декодирование.

Рассмотрим несколько примеров кодирования:

- перевод письменного текста с одного естественного языка на другой (в этом случае первичный алфавит — алфавит языка, на котором написан текст, вторичный алфавит — алфавит языка перевода);

- ввод и сохранение текста на компьютере (первичный алфавит — алфавит используемого естественного языка, вторичный алфавит — набор двоичных цифр {0; 1});

- флажковый семафор (первичный алфавит — алфавит используемого естественного языка, вторичный алфавит — совокупность различных положений рук (флажков) по отношению к туловищу сигнальщика)

Вопросы для самопроверки:

1. Подберите примеры для иллюстрации каждой из целей теории кодирования.

2. Поясните на конкретном примере двоиственность понятия «код».

3. Определите первичный и вторичный алфавит при кодировании сообщения, написанного на русском языке, посредством азбуки Морзе.

Код. Длина кода. Основание кода. Примеры.

При ответе на этот вопрос необходимо дать определение понятиям «код», «длина кода», «основание кода», привести примеры определения длины и основания кода для различных способов кодирования и различных кодовых признаков.

Правило или совокупность правил, в соответствии с которыми производится отображение дискретных сообщений сигналами в виде определенных сочетаний символов вторичного

алфавита, называют **кодом**.

Будем полагать, что источник выдает некоторое дискретное сообщение *a*, которое можно рассматривать как последовательность элементарных сообщений a_i (i = 1, 2, ..., l). Эти элементарные сообщения будем называть **символами сообщений**, а их совокупность $\{a_i\}$ - **алфавитом источника. Кодирование** заключается в том, что последовательность символов источника *a* заменяется последовательностью кодовых символов - кодовой комбинацией (кодовым словом).

Общее число символов, составляющих кодовую комбинацию, называется **значностью**, или **длиной кода** n. Количество значений кодовых признаков, используемых в кодовых комбинациях, называется **основанием кода m**.

Рассмотрим примеры:

На рисунке *7* приведен пример кодовой комбинации с длиной кода n = 5. В качестве импульсного признака здесь использована величина импульсов U_i. Основание кода (число значений импульсного признака) m = 3.

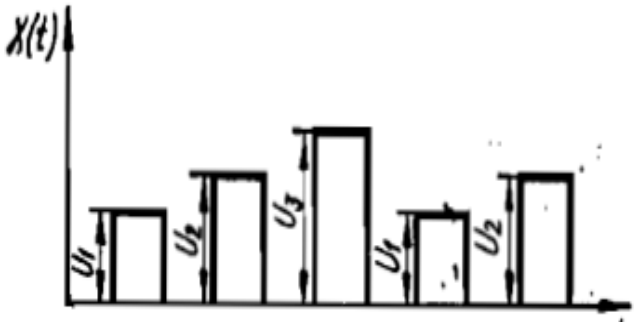


Рис.7 Пример кодовой комбинации с величиной импульсов U_i (i = 1..5)

На рисунке 8 приведена кодовая комбинация с n = 5 и m= 3,

в которой в качестве импульсного признака используется длительность импульсов t_i.

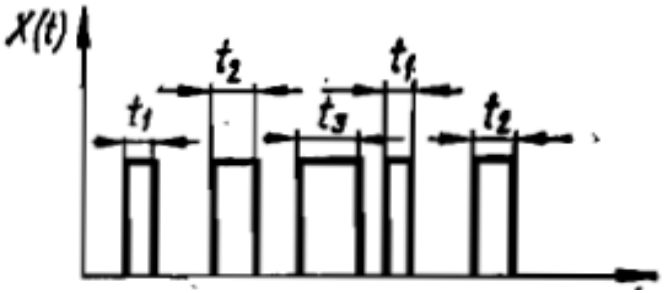


Рис.8 Пример кодовой комбинации с длительностью импульсов t_i

На ленте машины Поста для обозначения целых положительных чисел используется правило: число *n* задается (кодируется) *n+1* меткой. Тогда число 5, помещенное на ленте машины Поста, может быть интерпретировано как кодовая комбинация с длиной кода $n = 6$ и основанием кода $m = 1$ (число используемых меток — символов вторичного алфавита):

Рис.9 Число 5 на ленте машины Поста

Вопросы для самопроверки:

1. Что являлось символами сообщений при записи информации на магнитную ленту?

2. Определите значность и длину кода, если кодирование представляет собой перевод десятичного числа 20 в двоичную систему счисления.

*3. Определите значность и длину кода, если сообщение а — монохромное изображение размером 240*320 пикселей .*

Обратимое и необратимое кодирование. Условие обратимости кодирования.

При ответе на этот вопрос необходимо дать определение понятиям «кодирование» и «декодирование», привести примеры обратимого и необратимого кодирования, сформулировать условие обратимости кодирования и записать его математический вид.

Кодирование — перевод информации, представленной сообщением в первичном алфавите, в последовательность кодов.

Процесс обратного преобразования слова называется декодированием. Декодирование можно рассматривать как функцию F^{-1}– обратную функции F – кодированию.

Декодирование — операция, обратная кодированию, т.е. восстановление информации в первичном алфавите по полученной последовательности кодов.

Операции кодирования и декодирования называются **обратимыми,** если их последовательное применение не приводит к потере информации.

Примером обратимого кодирования является телеграф. Также к обратимому кодированию относят сжатие информации без потерь, помехоустойчивое кодирование. Необратимое кодирование происходит при переводе с одного естественного языка на другой, при сжатии с потерями, при аналого-цифровом преобразовании.

Необратимое кодирование можно подвергнуть более детальной классификации. Различают принципиально необратимое, обратимое с помощью дополнительной информации и безусловно обратимое кодирование.

Принципиально необратимое кодирование (хэширование) используется, например, в операционных системах для хранения паролей. При первоначальном вводе пароль преобразуется с помощью так называемых односторонних функций (хэш-функций), подобранных таким образом, чтобы из полученной на их выходе строки принципиально нельзя было получить первоначальное значение пароля. При дальнейшем использовании пароль каждый раз преобразуется такой же функцией и сравнивается с первоначальным хэшем. При их совпадении делается вывод о правильности ввода пароля. Объем полученной в итоге информации равен ровно одному биту: пароль совпал либо не совпал. В некоторых случаях этого может оказаться недостаточно, поэтому гораздо чаще используется кодирование, **обратимое с помощью дополнительной информации** (ключа шифрования). Входная информация преобразуется с помощью пароля таким образом, чтобы обратное преобразование также требовало знания пароля (простейший пример такого преобразования - операция исключающего ИЛИ между байтами исходного текста и байтами пароля). Именно этот способ обычно используется при шифровании архивов.

Наконец, последний случай - **безусловно обратимое кодирование**, в случае которого обратное преобразование не требует знания какой-то дополнительной информации. В подавляющем большинстве случаев для хранения паролей к внешним ресурсам используется именно этот способ. К примеру, почтовому клиенту для получения почты необходимо передать на POP3-сервер логин и пароль пользователя, который решил доверить их хранение клиенту, поставив соответствующую галочку. Пароль уходит на сервер в открытом виде, какие-то дополнительные пароли при его сохранении взять неоткуда

(бессмысленно требовать от пользователя какой-то дополнительный пароль, если он захотел избавиться от ввода основного пароля). В этом случае единственный вариант - использовать как можно более запутанные алгоритмы кодирования и декодирования, которые обеспечат относительную защиту данных.

Запишем условие обратимости кодирования в формализованном (математическом) виде. Введем некоторые обозначения:

- Пусть I_1 – это количество информации в исходном сообщении, состоящем из символов первичного алфавита А.

- Пусть I_2 – это количество информации в том же сообщении, записанном с помощью символов вторичного алфавита В, т.е. количество информации в сообщении после кодирования.

Тогда **условие обратимости кодирования** запишется в следующем виде:

$$I_1 \leq I_2$$

На практике при построении обратимого кода необходимо придерживаться следующих правил:

- разным символам первичного алфавита А должны быть сопоставлены разные кодовые комбинации;

- никакая кодовая комбинация не должна составлять начальную часть какой-нибудь другой кодовой комбинации.

Вопросы для самопроверки:

1. Является ли обратимым кодирование при помощи азбуки Морзе?

2. Классифицируйте кодирование информации на дорожных знаках с точки зрения его обратимости.

3. Определите возможный код символа «Г» в первичном алфавите {«А», «Б», «В», «Г»}, если в качестве вторичного алфавита используется двоичный алфавит и известны следующие коды: «А» - 00, «Б» - 010, «В» - 101.

Код. Средняя длина кода. Относительная избыточность кода.

При ответе на этот вопрос необходимо дать определение средней длины кода, рассказать об относительной избыточности кода и записать ее математическую формулу.

Условие обратимости кодирования имеет вид:

$$I_1 \le I_2 ,$$

Здесь I_1 – это количество информации в исходном сообщении, состоящем из n знаков первичного алфавита A, а I_2 – это количество информации в том же сообщении, записанном с помощью m знаков вторичного алфавита B.

Будем считать, что символы появляются на выходе источника независимо друг от друга. Тогда полученное соотношение можно переписать:

$$nI^A \le mI^B ,$$

Здесь I^A – информационный вес одного символа в алфавите A, а I^B – количество информации, приходящейся на один символ вторичного алфавита B.

Проведем дальнейшие преобразования условия

обратимости кодирования. Отношение $\frac{m}{n} \geq \frac{I^A}{I^B}$ характеризует среднее число знаков вторичного алфавита, которое приходится использовать для кодирования одного знака первичного алфавита. Частота каждого символа в любом сообщении бесконечной длины будет одинаковой и будет стремиться к вероятности появления этого символа на выходе источника.

Средняя длина кода для некоторого источника, первичного алфавита A, состоящего из N символов, вторичного алфавита B и кода φ определяется как сумма произведений длины кода символа первичного алфавита на соответствующую вероятность появления этого символа в сообщении:

$$K(\varphi, A, B) = \sum_{i=1}^{N} n_i p_i$$

Здесь n_i – это длина кодового слова для i-го символа первичного алфавита, p_i — вероятность появления i-го символа на выходе источника.

Если предположить постоянство поведения источника сообщений во времени, то предел отношения числа знаков вторичного алфавита к числу знаков первичного алфавита, кодирующих одно и то же сообщение, длина которого стремится к бесконечности, будет стремиться к средней длине кода:

$$\frac{\lim_{n\to\infty} m}{n} = \lim_{n\to\infty} \sum_{i=1}^{N} \frac{n_i l_i(n)}{n} =$$

$$= \sum_{i=1}^{N} n_i \frac{\lim_{n\to\infty} l_i(n)}{n} = \sum_{i=1}^{N} n_i p_i = K(\varphi, A, B)$$

Здесь $l_i(n)$ – количество i-х символов первичного алфавита в произвольном сообщении, длина n которого стремится к бесконечности. Длину кодового слова, кодирующего i-ый символ

первичного алфавита, мы обозначили через n_i.

Так как $nI^A \le mI^B$, **минимально возможное** значение **длины кода** будет:

$$K_{min}(\wp, A, B) = \frac{I^A}{I^B}$$

В качестве меры превышения длины кода будем использовать относительную избыточность кода.

Относительной избыточностью кода назовем величину

$$Q(\wp, A, B) = \frac{K(\wp, A, B) - K_{min}(\wp, A, B)}{K_{min}(\wp, A, B)}$$

Относительная избыточность кода показывает насколько операция кодирования увеличивает длину сообщения по отношению к первоначальной. Достаточно часто эту величину выражают в процентах. Так, если относительная избыточность некоторого кода равна 10%, то это означает, что при данном способе кодирования мы будем вынуждены передавать по каналу связи на 10% больше информации, чем содержится в исходном сообщении.

Вопросы для самопроверки:

1. Определите среднюю длину кода для символов первичного алфавита {«А», «Б», «В», «Г»}, если при кодировании получены следующие коды этих символов: «А»- 0, «Б»- 10, «В»- 110, «Г»- 111.

2. Определите минимально возможную длину кода при двоичном кодировании символов первичного алфавита {«А», «Б», «В», «Г»} с известными вероятностями появления символов { 0,45; 0,15; 0,05; 0,35}.

3. Определите относительную избыточность кода задания 1 с учетом вычисленной в задании 2 минимально возможной длины кода.

Асимптотически оптимальный код. Первая теорема Шеннона.

При ответе на этот вопрос необходимо дать определение асимптотически оптимального кода и рассказать о практических способах его построения, сформулировать основную теорему о кодировании при отсутствии шумов как для общего случая, так и для случая двоичного кодирования.

Для фиксированного первичного алфавита A и фиксированного вторичного алфавита B существует множество различных способов построения кода, ставящего символам из алфавита A символы или комбинации символов алфавита B. Относительная избыточность кода – характеристика кода, показывающая, во сколько раз требуется удлинить сообщение, чтобы обеспечить его надежную (безошибочную) передачу или хранение. Для разных вариантов построения относительная избыточность кода тоже может быть различной. Кроме того ресурсы (машинная память, время работы), необходимые для кодирования/декодирования разных кодов могут быть различны. Возникает вопрос: «Какой же из построенных кодов лучше? Какой код является более оптимальным?»

Для некоторого первичного алфавита A и вторичного алфавита B **асимптотически оптимальным кодом** будем называть такой способ кодирования, при котором избыточность кодирования стремится к нулю, если длина сообщений стремится к бесконечности.

Насколько хороший код можно построить? На этот вопрос

дает ответ **первая теорема Шеннона** (основная теорема о кодировании при отсутствии шумов):

При отсутствии шумов всегда возможен такой вариант кодирования сообщения, при котором относительная избыточность кода будет сколь угодно близка к нулю.

Теорема Шеннона не указывает конкретного способа кодирования, но из нее следует, что при выборе каждого символа кодовой комбинации необходимо стараться, чтобы он нес максимальную информацию. По определению $K_{min}(\wp, A, B) = \frac{I^A}{I^B}$. Известно, что сообщения, записанные символами первичного алфавита генерируются некоторым источником S, который характеризуется вероятностями появления отдельных символов алфавита А на выходе источника. Из этого следует, что уменьшить K_{min} можно либо уменьшив числитель I^A, либо увеличив знаменатель I^B. Уменьшения информационного веса символа алфавита А можно достичь, если учесть различие частот появления символов первичного алфавита. Увеличить информационный вес символов алфавита В можно используя такой способ кодирования, при котором появление знаков вторичного алфавита было бы равновероятным, то есть $I^B = log_2 | B |$.

Будем учитывать различные вероятности появления отличных символов первичного алфавита на выходе источника, считая при этом, что корреляций между символами, генерируемыми источником нет. То есть, источник не запоминает, какие символы он уже выдавал, и генерирует новые символы независимо от прошлого. Такие источники называют источниками без памяти. Тогда, минимальное значение средней длины кода можно записать следующим образом:

$$K_{min}(\wp, A, B) = \frac{I^A}{\log_2(B)}$$

На практике почти повсеместно в цифровой технике используется двоичное кодирование, то есть $|B| = 2$, а сам вторичный алфавит B состоит из нуля и единицы (B = {0, 1}). Такое кодирование проще всего реализовать. Например, информацию можно хранить как последовательность намагниченных или не намагниченных участков жесткого диска. Нетрудно видеть, что в этом случае имеем:

$$K_{min}(\wp, A, 2) = \frac{I^A}{\log_2(2)} = I^A$$

Таким образом, для двоичного кодирования первая теорема Шеннона может быть переформулирована.

Теорема Шеннона для двоичного кодирования: При отсутствии помех средняя длина двоичного кода может быть сколь угодно близкой к средней информации, приходящейся на один символ первичного алфавита.

Формула относительной избыточности кодирования в случае двоичного кода принимает вид:

$$Q(\wp, A, 2) = \frac{K(\wp, A, 2)}{I^A} - 1$$

Вопросы для самопроверки:

1. Сформулируйте основную теорему кодирования при отсутствии шумов.

2. Влияет ли наличие шумов в канале связи на значение относительной избыточности кода?

3. Определите в каком случае для символов первичного

алфавита {«А», «Б», «В», «Г»}, обладающих одинаковой частотой появления, будет получен более оптимальный код:

а) в качестве вторичного алфавита выбран двоичный алфавит;

б) в качестве вторичного алфавита выбран алфавит троичных цифр.

Классификация способов кодирования.

При ответе на этот вопрос необходимо произвести классификацию способов кодирования по различным критериям, привести примеры различных видов кода, дать сравнительную оценку различных способов кодирования в рамках одного критерия.

Существует множество способов классификации способов кодирования.

Критерий классификации: ***Условие построения кодовых комбинаций***

По условию построения кодовых комбинаций коды делят на равномерные и неравномерные. В **равномерных** кодах все сообщения передаются кодовыми группами с **одинаковым** числом элементов (длина кода $n = const$). Примером такого кода может служить телеграфный код Бодо, который является равномерным пятибитным кодом ($n=5$). Первоначально код, разработанный Эмилем Бодо (1845 -1903) в 1870 году для своего телеграфа, вводился прямо клавиатурой, состоящей из пяти клавиш, нажатие или ненажатие клавиши соответствовало передаче или непередаче одного бита в пятибитном коде. В 1901 году Дональд Мюррей (1866 - 1945) переработал код, изменил порядок знаков и добавил некоторые дополнительные знаки, адаптировав код Бодо к раскладке современной клавиатуры

QWERTY. Однако общие принципы — пятибитная кодировка и использование буквенного и цифрового регистров — остались неизменными (см. Приложение 1).

При использовании **неравномерных** кодов разные сообщения могут передаваться кодовыми группами, содержащими **неодинаковое** число элементов ($n = var$). Типичным представителем неравномерного кода является код Морзе, созданный в 1838 году Самюэлем Морзе и подвергавшийся впоследствии неоднократным изменениям. В настоящее время кодом Морзе называют способ знакового кодирования, в котором для представления букв алфавита, цифр, знаков препинания и других символов используется последовательность троичных сигналов, например, длинных и коротких: «тире» и «точек» (см. рис. 4). За единицу времени принимается длительность одной точки. Длительность тире равна трём точкам. Пауза между элементами одного знака — одна точка, между знаками в слове — три точки, между словами — семь точек.

Равномерный код обладает большими возможностями с точки зрения обеспечения помехозащищенности передачи, так как потеря элементов или возникновение новых элементов в кодовых комбинациях с $n = const$ могут быть легко обнаружены. Неравномерные коды могут обеспечить наибольшую экономичность построения кодов и наибольшее быстродействие передачи сообщений. Такие коды используются при так называемом статистическом кодировании. Вместе с тем неравномерные коды менее помехозащищенные, чем равномерные. Потеря или возникновение новых элементов в комбинации в результате действия помех могут привести к созданию новой ложной комбинации, воспринимаемой на приемной стороне как истинная. Неравномерные коды требуют

при передаче либо специальных разделительных символов, указывающих конец одной и начало другой кодовой комбинации (например, код Морзе требует наличия разделительного символа), либо же должны строиться так, чтобы никакая кодовая комбинация не явилась началом другой.

Критерий классификации:
Число различных символов в кодовых комбинациях (значность кода)

По числу уникальных символов, используемых в кодовых комбинациях различают единичные, двоичные и многопозиционные коды. В **единичном коде** используются одинаковые символы. Кодовые комбинации отличаются друг от друга лишь количеством символов (импульсов). Такие коды называют еще числоимпульсными. Единичные коды используются в машине Поста (для кодирования целых положительных чисел), машине Тьюринга, в цифровых электронных счетчиках, в которых измеряемая величина преобразуется в пропорциональное ей число импульсов.

Единичный код отличается своей простотой. Однако вследствие того, что он неравномерен, помехозащищенность его низкая. Кроме того, при передаче большого количества сообщений происходит изменение в широких пределах длины кода, что вызывает определенные неудобства.

В связи с этим единичный код практически не используется для передачи информации по каналу связи, а используется лишь при промежуточных преобразованиях сигналов на передающей и приемной сторонах.

Наибольшее распространение получили **двоичные коды**. Это обусловлено следующим. Формирование кодовых сигналов и их дешифрация производятся с помощью релейных устройств,

способных занимать ряд устойчивых состояний. Количество таких состояний определяется основанием кода. Очевидно, что простейшими релейными устройствами являются устройства с двумя состояниями. К такого типа устройствам принадлежит большинство электромагнитных, электронных, магнитных и других типов бесконтактных реле. Кроме того, следует также учитывать простоту хранения информации и выполнения арифметических и логических операций при двоичном кодировании.

Многопозиционные коды, алфавит которых состоит из большого числа символов, пока не нашли широкого применения в информационных системах.

Критерий классификации:
Форма представления в канале передачи данных

По форме представления сигнала в канале передачи данных различают последовательные и параллельные коды. В **последовательных кодах** элементарные сигналы, составляющие кодовую комбинацию, посылаются в канале передачи последовательно во времени. Как правило, они разделены между собой определенным временным интервалом.

В **параллельных кодах** элементарные сигналы посылаются одновременно по нескольким электрическим цепям, число которых соответствует количеству элементов кода.

Параллельная форма представления кода, хотя и связана с меньшей затратой времени для передачи сообщений, используется для передачи информации по каналу связи редко, так как требует значительных материальных затрат на многопроводные линии связи. Практически параллельная форма кода при передаче информации по однопроводной линии связи используется лишь в тех случаях, когда в качестве импульсного признака применяется частота и на приемной стороне элементы

кодовой комбинации можно легко разделить с помощью частотных фильтров.

Параллельная форма представления кода часто используется при преобразовании аналоговых величин в код и обратных преобразованиях, в устройствах памяти, регистрации, при логической и математической обработке информации, когда важную роль играет быстродействие.

Критерий классификации:
Возможность обнаружения и исправления ошибок

В зависимости от возможности обнаружения и исправления ошибок в полученных по каналу связи кодах различают простые и корректирующие коды. В **простых кодах** все возможные кодовые комбинации используются непосредственно для передачи информации и ошибка в приеме хотя бы одного элемента кодовой комбинации приводит к неправильной регистрации передаваемого сообщения. В простых равномерных кодах превращение одного символа комбинации в другой, например 1 в 0 или 0 в 1, приводит к появлению новой комбинации, т.е. к ошибке.

Корректирующие коды - это коды, позволяющие по имеющейся в кодовой комбинации избыточности обнаруживать и исправлять определённые ошибки, появление которых приводит к образованию ошибочных или запрещенных комбинаций. Применяются при передаче и обработке информации в вычислительной технике, телеграфии, телемеханике и технике связи, где возможны искажения сигнала в результате действия различного рода помех. Кодовые слова корректирующих кодов содержат информационные и проверочные разряды (символы). В процессе кодирования при передаче информации из информационных разрядов в

соответствии с определёнными для каждого корректирующего кода правилами формируются дополнительные символы — проверочные разряды. При декодировании из принятых кодовых слов по тем же правилам вновь формируют проверочные разряды и сравнивают их с принятыми; если они не совпадают, значит при передаче произошла ошибка. Существуют коды, обнаруживающие факт искажения сообщения, и коды, исправляющие ошибки, т.е. такие, с помощью которых можно восстановить первичную информацию.

Критерий классификации:
Число одновременно кодируемых символов первичного алфавита

Исходя из числа одновременно кодируемых символов первичного алфавита кодирование классифицируют на алфавитное и блочное. При **алфавитном кодировании** передаваемое сообщение представляет собой последовательность кодов отдельных знаков первичного алфавита. Однако возможны варианты кодирования, при которых кодовый знак относится сразу к нескольким буквам первичного алфавита (будем называть такую комбинацию блоком) или даже к целому слову первичного языка. Кодирование блоков понижает избыточность. Применение **блочного метода** кодирования имеет свои недостатки. Во-первых, необходимо хранить огромную кодовую таблицу и постоянно к ней обращаться при кодировании и декодировании, что замедлит работу и потребует значительных ресурсов памяти. Во-вторых, помимо основных слов разговорный язык содержит много производных от них, например, падежи существительных в русском языке или глагольные формы в английском; в данном способе кодирования им всем нужно присвоить свои коды, что

приведет к увеличению кодовой таблицы еще в несколько раз. В-третьих, возникает проблема согласования (стандартизации) этих громадных таблиц, что непросто. Наконец, в-четвертых, алфавитное кодирование имеет то преимущество, что буквами можно закодировать любое слово, а при кодировании слов – можно использовать только имеющийся словарный запас.

Вопросы для самопроверки:

1. Расшифруйте сообщение S = 04061501, если известно, что это равномерный код, в котором каждой букве кирилицы ставится в соответствие ее порядковый номер в алфавите минимально возможной длины.

2. Приведите пример многопозиционного кода.

3. Приведите пример блочного кодирования.

Неравномерный код с разделителем.

При ответе на этот вопрос необходимо определить основные свойства и правила построения неравномерного кода с разделителем, привести пример построения такого кода при заданном наборе правил построения.

Неравномерный код с разделителями должен обладать следующими свойствами:

- длина кода для различных символов первичного алфавита различна;
- существуют разделители символов, слов или предложений (сообщений).

Рассмотрим пример построения такого кода. Определим правила построения:

- будем применять алфавитное кодирование - каждая буква первичного алфавита будет иметь собственный уникальный код;

- введем разделители символов — разделителем отдельных кодов будет последовательность двух нулей «00»;

- введем разделители слов — это будет последовательность трех нулей «000» - пробел;

- код признака конца символа включим в код символа, так как код признака конца символа не используется сам по себе, отдельно от кода символа, следовательно все коды символов будут заканчиваться на «00»;

- коды символов не должны содержать двух (или более) нулей подряд нигде кроме как в конце кода. В противном случае такой код будет воспринят как два кода двух различных знаков первичного алфавита;

- код символа не должен начинаться с «0», в противном случае ноль в начале кода текущего символа и два нуля в конце кода предыдущего символа будут образовывать код пробела.

- т. к. разделителю слов «000» всегда предшествует признак конца знака (конце каждого слова реализуется последовательность «00000»), то для однозначного декодирования нужно потребовать, чтобы коды символов оканчивались не более чем четырьмя нулями.

В соответствии с перечисленными правилами можно построить таблицу кодов для кириллицы.

Код пробела уже определен - «000» (можно рассмотреть этот код как совокупность порядкового номера символа - «0» и

признака конца символа - «00»). Коды остальных символов будем определять в порядке их распределения в таблице частоты использования букв русского языка (см. Приложение 2).

При этом коды символов будем получать последовательным прибавлением единицы к коду предыдущего символа. При этом необходимо проверять получающиеся комбинации на соответствие выбранным правилам и отвергать кодовые последовательности не отвечающие этим правилам.

Как видно из таблицы, чаще всего в словах русского языка встречается буква «о». Определим ей код «100»: к порядковому номеру предыдущего символа - «пробела» (порядковый номер 0) добавим единицу, в конце кода допишем два нуля как признак конца символа. Полученный код не противоречит принятым правилам.

Поступая аналогично, определим коды букв «е, ё» - «1000» ($1_2+1_2=10_2$ — порядковый номер буквы в таблице, выраженный в двоичной системе счисления; два нуля в конце кода — признак конца символа), код буквы «а» - «1100», код буквы «и» - «10000», код буквы «т» - «10100», код буквы «н» - «11000», код буквы «с» - «11100». Все эти коды удовлетворяют установленным нами правилам построения кода. Применяя алгоритм построения кода к следующей букве таблицы (букве «р»), получим: номер буквы — $111_2+1_2 = 1000_2$, к нему нужно добавить «00» - признак конца символа, но тогда код буквы «р» будет «100000», что противоречит последнему условию формирования нашего кода (коды символов должны оканчиваться не более чем четырьмя нулями). Увеличим порядковый номер символа еще на единицу: $1000_2+1_2 = 1001_2$ — и, приписав к нему справа два нуля, сформируем возможный код символа «р». Получим «100100». Однако и этот код тоже не

может быть кодом символа в нашем неравномерном коде с разделителем — он противоречит следующему правилу: «коды символов не должны содержать двух (или более) нулей подряд нигде кроме как в конце кода». Продолжаем увеличение порядкового номера кода символа. Следующий возможный код - «101000». Он удовлетворяет всем правилам построения нашего кода и может быть выбран в качестве кода буквы «р».

Действуя аналогично можно определить коды для всех символов алфавита:

Буква	*пробел*	*о*	*е, ё*	*а*	*и*
Код	000	100	1000	1100	10000
Буква	*т*	*н*	*с*	*р*	*в*
Код	10100	11000	11100	101000	101100
Буква	*л*	*к*	*м*	*д*	*п*
Код	110000	110100	111000	111100	1010000
Буква	*у*	*я*	*ы*	*з*	*ъ, ь*
Код	1011000	1011100	1101000	1101100	1110000
Буква	*б*	*г*	*ч*	*й*	*х*
Код	1110100	1111000	1111100	10101000	10101100
Буква	*ж*	*ю*	*ш*	*ц*	*щ*
Код	10110000	10110100	10111000	10111100	11010000
Буква	*э*	*ф*			
Код	11010100	11011100			

В итоге имеем неравномерный код, в котором длина кода для разных символов изменяется от 3 до 8 символов. Найдем среднюю длину кода для данного способа кодирования:

$$K(\wp, Cyrilic, 2) = \sum_{i=0}^{32} P_i n_i \approx 5{,}5$$

Здесь P_i - частота использования i-й буквы, n_i - это длина кодового слова для i-й буквы.

Определим избыточность кодирования, учитывая, что средний информационный вес символа русского алфавита равен 4,72 бита:

$$Q(\wp, Cyrilic, 2) = \frac{K(\wp, Cyrilic, 2)}{I^A} - 1 = \frac{5{,}5}{4{,}72} - 1 \approx 0{,}17$$

Это значит, что при данном способе кодирования мы будем вынуждены передавать на 17% больше информации, чем содержится в исходном сообщении.

Вопросы для самопроверки:

1. Сформулируйте основный правила построения неравномерного кода с разделителем.

2. Может ли для выбранного в параграфе набора правил построения неравномерного кода существовать символ с кодом 11001000?

3. Постройте неравномерный двоичный код с разделителем для первичного алфавита {«+», «-», «», «/»}.*

Префиксные коды. Условие Фано. Примеры.

При ответе на этот вопрос необходимо объяснить понятие «префикс», дать определение префиксного кода, сформулировать условие Фано, привести примеры префиксных кодов.

В языковедении термин «префикс» означает «приставка».

Префиксный код в теории кодирования — код со словом переменной длины, удовлетворяющий условию Роберта Фано.

Условие Фано: неравномерный код может быть однозначно декодирован, если никакой из кодов не совпадает с началом (префиксом) какого-либо другого, более длинного кода.

Например, если один из символов первичного алфавита имеет код «110» то в качестве кодов других символов нельзя использовать последовательности «1», «11», но можно использовать «0», «10».

Декодирование для префиксных кодов однозначно и выполняется слева направо. При этом сопоставление с таблицей кодов всегда дает точное указание конец одного кода и начало другого. Это дает возможность не использовать разделители.

Заметим, что любой код со словом фиксированной длины также является префиксным. Примерами префиксных кодов служат: совокупность телефонных номеров в стационарных сетях, Юникод, код Хаффмана.

Рассмотрим пример собственного префиксного кода:. Пусть требуется составить префиксный код для первичного алфавита А={а, л, м, р, у, ы}. В качестве вторичного алфавита возьмем двоичный алфавит В={0; 1}.

Определим для символа «а» код «00». Для выполнения условия Фано коды других символов не должны начинаться с «00». Букве «л» определим код «01», «м» - код «100», «р» - код «101». Коды оставшихся букв должны начинаться с «11»: пусть код буквы «у» будет «110», а код буквы «ы» - «111».

$\mathbf{a_i}$	а	л	м	р	у	ы
Код	00	01	100	101	110	111

Теперь, используя построенную таблицу кодов, декодируем сообщение: 10000100001001110100101001001110

Начинать следует слева направо, последовательно вычеркивая обнаруженные коды, и записывая соответствующие им знаки первичного алфавита.

Символа с кодом «1» не существует. Добавляем к «1» следующую цифру кода - «0». Символа с кодом «10» в кодовой таблице тоже нет. Присоединяем следующую цифру кодовой последовательности. Получаем код «100». Это код буквы «м».

Собираем код следующей буквы. Символа с кодом «0» в таблице нет. Присоединение к коду следующей цифры кодовой последовательности дает нам код «00» - код буквы «а».

Поступая аналогично можно получить однозначный вариант расшифровки (декодирования) заданного сообщения:

100 00 100 00 100 111 01 00 101 00 100 110

м а м а м ы л а р а м у

Вопросы для самопроверки:

1. Сформулируйте условие Фано.

2. Приведите примеры существующих префиксных кодов?

3. Определите является ли префиксным следующий двоичный код первых 5 букв латинского алфавита:

a	b	c	d	e
000	110	01	001	10

Декодируйте сообщение 1100000100110, записанное с использованием приведенного кода.

Префиксный код Шеннона-Фано.

При ответе на данный вопрос необходимо привести пример построения префиксного кода Шеннона-Фано для заданного начального алфавита и известных частот использования символов этого алфавита с помощью таблицы и графа.

В 1948-1949 гг. Клод Шеннон (*Claude Elwood Shannon*) и Роберт Фано (*Robert Mario Fano*) независимо друг от друга предложили префиксный код, названный в последствие в их честь. Алгоритм Шеннона — Фано использует избыточность сообщения, заключённую в неоднородном распределении частот символов его первичного алфавита, то есть заменяет коды более **частых** символов **короткими** двоичными последовательностями, а коды более **редких** символов — более **длинными** двоичными последовательностями.

Рассмотрим алгоритм построения этого кода на примере. Пусть имеется первичный алфавит, состоящий из шести символов: {A; B; C; D; E; F}, также известны вероятности появления этих символов в сообщении, они равны соответственно 0,15; 0,2; 0,1; 0,3; 0,2; 0,05. Расположим эти символы в таблице в порядке убывания их вероятностей.

Первичный алфавит	D	B	E	A	C	F
Вероятность появления	0,3	0,2	0,2	0,15	0,1	0,05

Кодирование осуществляется следующим образом. Все символы делятся на две группы **с сохранением порядка следования** (по убыванию вероятностей появления), так чтобы суммы вероятностей в каждой группе были приблизительно

равны, а, следовательно, абсолютные разности суммарных вероятностей каждой группы близки к нулю. В нашем примере в первую группу попадают символы D и B (их суммарная вероятность использования равна 0,5), все остальные буквы (также имеющие суммарную вероятность появления 0,5) попадают во вторую группу. Поставим ноль в первый знак кодов для всех символов из первой группы, а первый знак кодов символов второй группы установим равным единице.

Продолжим деление каждой группы. В первой группе два элемента, и деление на подгруппы здесь однозначно: в первой подгруппе будет символ D, а во второй - символ B. Во второй группе теоретически возможны три способа деления на подгруппы: {E} и {A, C, F}, {E, A} и {C, F}, {E, A, C} и {F}. Но в первом случае абсолютная разность суммарных вероятностей будет |0,2 — (0,15 + 0,1 + 0,05)| = 0,1. Во втором и третьем варианте деления аналогичные величины будут 0,2 и 0,4 соответственно. Согласно алгоритму необходимо выбрать тот способ деления, при котором суммы вероятностей в каждой подгруппе были примерно одинаковыми, а, следовательно, вычисленная разность минимальна. Соответственно наилучшим способом деления будет следующий вариант: символ {E} остается в первой подгруппе, а символы {A, C, F} образуют вторую группу. Далее по имеющемуся алгоритму распределим нули и единицы в соответствующие знаки кода каждой подгруппы.

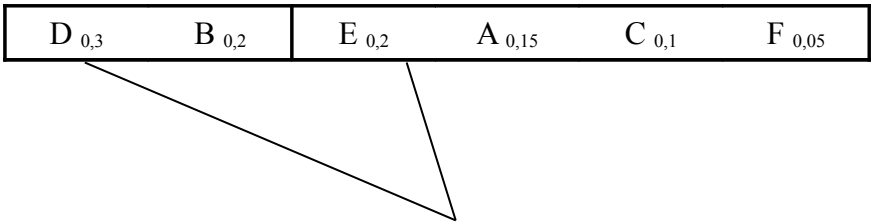
Осуществляем деление на подгруппы по той же схеме до тех пор, пока не получим группы, состоящие из одного элемента. Процедура деления изображена в нижеприведенной таблице (символ «X» означает, что данный знак кода отсутствует):

Первичный алфавит	Вероятности появления символов	Знаки кода символа				Код символа	Длина кода
		I	II	III	IV		
D	0,3	0	0	X	X	00	2
B	0,2	0	1	X	X	01	2
E	0,2	1	0	X	X	10	2
A	0,15	1	1	0	X	110	3
C	0,1	1	1	1	0	1110	4
F	0,05	1	1	1	1	1111	4

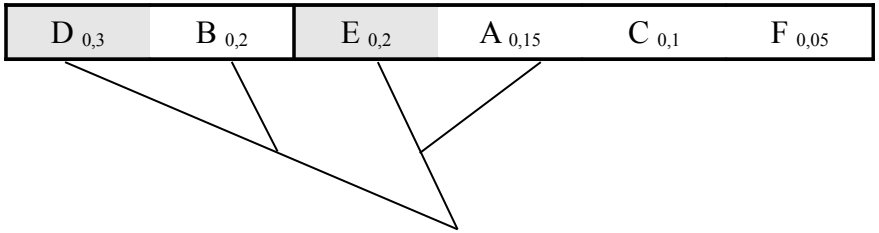
Данный код может быть построен и с помощью графа. Распределим символы алфавита в порядке убывания вероятностей — это будут концевые вершины (листья) будущего двоичного дерева (нижние индексы соответствуют вероятностям появления символов):

$D_{0,3}$ $B_{0,2}$ $E_{0,2}$ $A_{0,15}$ $C_{0,1}$ $F_{0,05}$

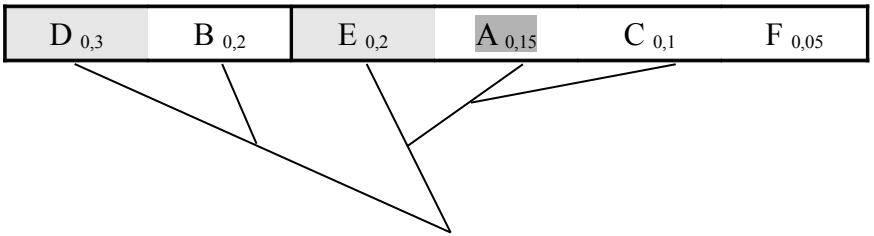
Согласно алгоритму построения кода Шеннона-Фано разобьем эти символы на две группы с приблизительно равными суммарными вероятностями появления. Порядок следования символов при этом изменять нельзя. Для нашего первичного алфавита в первую группу войдут символы D и B, а во вторую E, A, C и F. Соединим первые символы каждой группы с корнем дерева:



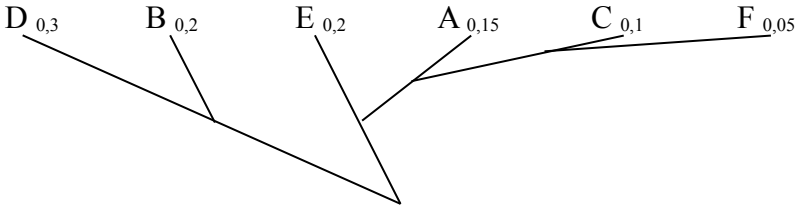
Продолжаем построение графа по приведенному алгоритму, соединяя первые символы получающихся подгрупп с узлами ветвления более высоких уровней. Таким образом, на следующем этапе построения получим:



Далее:

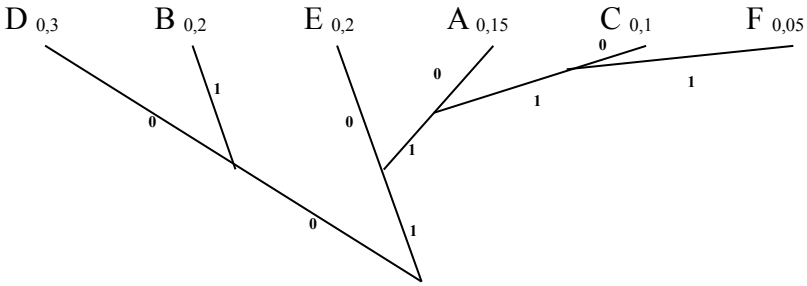


Окончательно имеем следующий граф:



Теперь для каждого узла ветвления обозначим каждую

левую исходящую дугу цифрой 0, а каждую правую исходящую дугу цифрой 1:



Для получения кода символа достаточно пройти по дугам полученного дерева от корня к соответствующей вершине и записать номера дуг, по которым осуществляется движение. Например, для символа А, двигаясь от корня дерева, проходим дуги с номерами 1, 1 и 0, следовательно код символа А — 110. Аналогично могут быть получены коды других символов.

Полученный код удовлетворяет условию Фано, следовательно он является префиксным. Средняя длина этого кода равна:

$$K(Шеннона-Фано, A, 2) = \sum_{i=1}^{6} n_i p_i = 0{,}3*2 + 0{,}2*2 +$$

$$+ 0{,}2*2 + 0{,}15*3 + 0{,}1*4 + 0{,}05*4 = 2{,}45 \text{ символов}$$

Среднее количество информации на один символ первичного алфавита равно:

$$I^A = -(0{,}3*\log_2 0{,}3 + 0{,}2*\log_2 0{,}2 + 0{,}2*\log_2 0{,}2 + 0{,}15*\log_2 0{,}15 + + 0{,}1*\log_2 0{,}1 + 0{,}05*\log_2 0{,}05) = 2{,}41 \text{ бит.}$$

Теперь по известной нам формуле найдем избыточность кода Шеннона –Фано:

$$Q(Шеннона\text{-}Фано, A, Binary) = 2{,}45/2{,}41 - 1 = 0{,}01659751.$$

То есть избыточность кода Шеннона-Фано для нашего шестибуквенного алфавита составляет всего около 1,7 %. Заметим, что для русского алфавита избыточность кодирования кодом Шеннона-Фано составила бы примерно 1,47%.

Вопросы для самопроверки:

1. Объясните алгоритм построения кода Шеннона-Фано.

2. Докажите, что построенный по алгоритму Шеннона-Фано код будет префиксным.

3. Постройте код Шеннона-Фано для символов первичного алфавита A={«+», «-», «», «/»} с известными вероятностями появления символов P={0,4; 0,2; 0,3; 0,1}:*

а) при помощи таблицы,

б) при помощи графа.

Префиксный код Хаффмана.

При ответе на данный вопрос необходимо привести пример построения префиксного кода Хаффмана для заданного начального алфавита и известных частот использования символов этого алфавита с помощью таблицы и графа.

В 1952 году Давид Хаффман показал, что предложенный им метод кодирования является оптимальным префиксным кодом для дискретных источников без памяти (заметим, что для таких источников все генерируемые сообщения независимы друг от друга).

Алгоритм кодирования методом Хаффмана состоит из двух этапов. На первом этапе исходный алфавит на каждом шаге сокращается на один символ и на следующем шаге рассматривается новый, сокращенный первичный алфавит.

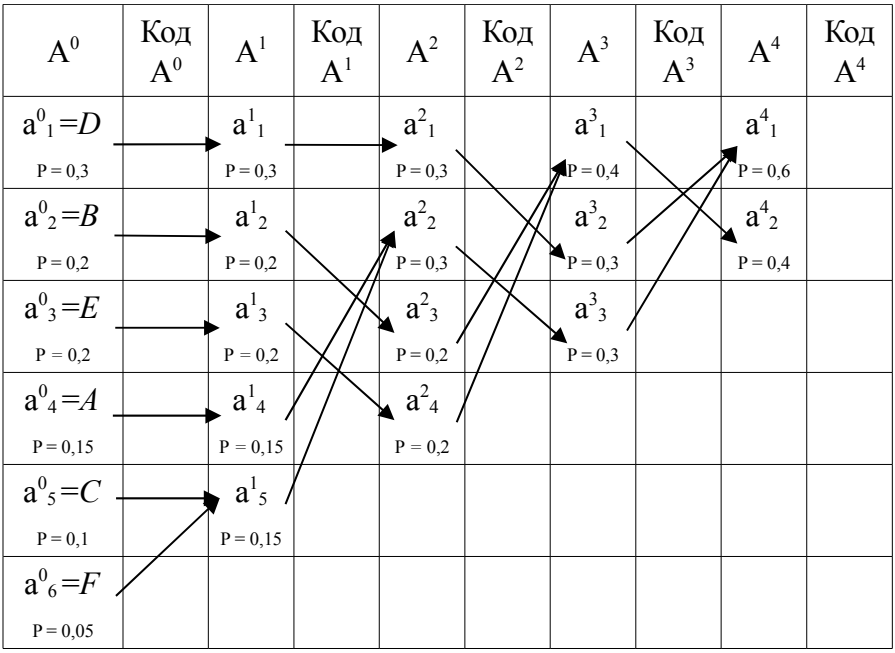
Число таких шагов будет на две единицы меньше первоначального числа символов. На втором этапе происходит пошаговое формирование кода символов, при этом заполнение кода осуществляется с символов последнего сокращенного первичного алфавита.

Рассмотрим алгоритм построения кода Хаффмана на примере. Пусть имеется первичный алфавит, состоящий из шести символов: {A; B; C; D; E; F}, также известны вероятности появления этих символов в сообщении соответственно {0,15; 0,2; 0,1; 0,3; 0,2; 0,05}. Расположим эти символы в таблице в порядке убывания их вероятностей.

Первичный алфавит	D	B	E	A	C	F
Вероятности появления символов	0,3	0,2	0,2	0,15	0,1	0,05

На первом шаге алгоритма два символа исходного алфавита, который мы назовем A^0, с наименьшими вероятностями объединяются в один новый символ. Вероятность нового символа есть сумма вероятностей тех символов, которые его образовали. Таким образом, получаем новый алфавит, который содержит на один символ меньше чем предыдущий. На следующем шаге алгоритма описанная процедура применяется к новому алфавиту. И так до тех пор, пока в очередном алфавите не остается только двух символов.

Процедура построения всех промежуточных алфавитов отражена в следующей таблице (в каждом алфавите символы упорядочены по убыванию вероятностей, верхний индекс символа совпадает с индексом промежуточного алфавита, которому он принадлежит, а нижний — показывает порядковый номер символа):



Теперь начинается второй этап алгоритма кодирования по Хаффману. Для формирования кода мы нумеруем символы всех промежуточных алфавитов, начиная с последнего. В нашем примере – с A^4.

В A^4 всего два символа. Они получают соответственно номера 0 и 1. В алфавите A^3 уже три символа. Причем, один из символов алфавита A^4, назовем этот символ «предок», был получен объединением двух символов алфавита A^3, назовем первый из этих символов «дочкой», а второй «сыном». Коды этих двух символов формируются следующим образом. К номеру «предка» приписываются **справа** 0, чтобы получить номер «дочки», и 1 – чтобы получить номер «сына». Следующая итерация алгоритма по той же схеме формирует коды символов алфавита A^2. В нем два первых символа будут иметь те же коды, что были у них в A^1, а два последних символа изменят свой код,

удлинив его на 1 символ («0» и «1» соответственно). Процесс останавливается при достижении первичного алфавита A^0 – коды для знаков первичного алфавита получены.

Отразим формирование кода в построенной таблице:

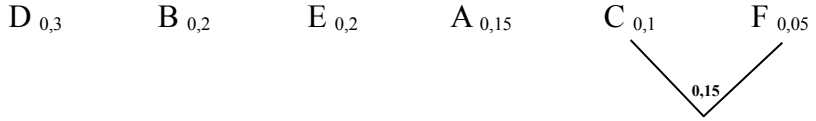
A^0	Код A^0	A^1	Код A^1	A^2	Код A^2	A^3	Код A^3	A^4	Код A^4
$a^0_1=D$ P = 0,3	00	a^1_1 P = 0,3	00	a^2_1 P = 0,3	00	a^3_1 P = 0,4	1	a^4_1 P = 0,6	0
$a^0_2=B$ P = 0,2	10	a^1_2 P = 0,2	10	a^2_2 P = 0,3	01	a^3_2 P = 0,3	00	a^4_2 P = 0,4	1
$a^0_3=E$ P = 0,2	11	a^1_3 P = 0,2	11	a^2_3 P = 0,2	10	a^3_3 P = 0,3	01		
$a^0_4=A$ P = 0,15	010	a^1_4 P = 0,15	010	a^2_4 P = 0,2	11				
$a^0_5=C$ P = 0,1	0110	a^1_5 P = 0,15	011						
$a^0_6=F$ P = 0,05	0111								

Данный алгоритм построения можно осуществить и с помощью графа. Расположим символы первичного алфавита в порядке убывания вероятностей их появления. Эти символы будут листьями будущего кодового дерева. Будем считать, что уровень этих концевых узлов равен N.

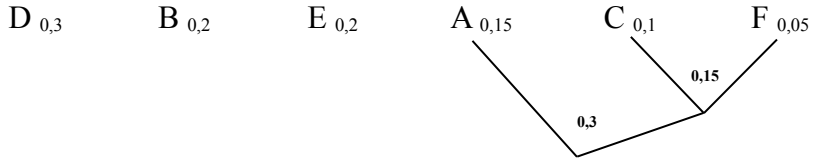
$D_{0,3}$ $B_{0,2}$ $E_{0,2}$ $A_{0,15}$ $C_{0,1}$ $F_{0,05}$

Теперь соединим два крайних правых символа (C и F), имеющих наименьшую вероятность появления, дугами, исходящими из одного и того же узла уровня N-1 (в основании узла указана суммарная вероятность использования

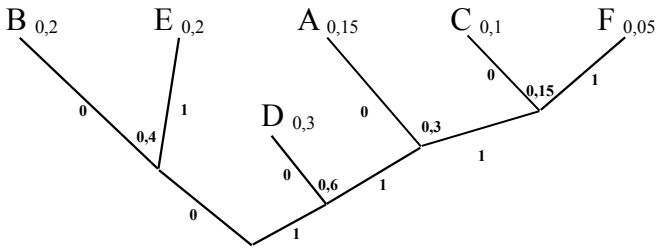
получившегося символа нового промежуточного алфавита):



В получившемся промежуточном алфавите вновь выбираем два символа с наименьшей частотой использования. Это символ A с вероятностью 0,15 и новый символ, получившийся в результате объединения символов C и F на предыдущем этапе и имеющий ту же вероятность использования. Соединяем эти символы дугами, исходящими из одного узла N-2 уровня:



Продолжая описанный алгоритм построения, получим следующее дерево:



Также как в методе Шеннона-Фано пронумеруем каждую левую исходящую дугу узла цифрой 0, а каждую правую дугу цифрой 1. Теперь для получения кода символа достаточно пройти от корня до нужного символа и записать номера дуг, по которым осуществляется прохождение.

Посчитаем среднюю длину кодового слова для кода Хаффмана и нашего первичного алфавита А.

$$K(Хаффман, A, Binary) = 0,3*2 + 0,2*2 + 0.2*2 + 0,15*3 + +0,1*4 + 0.05*4 = 2,45 \text{ символа}$$

Среднее количество информации на один символ первичного алфавита равно:

$$I^A = -(0,3*\log_2 0,3 + 0,2*\log_2 0,2 + 0,2*\log_2 0,2 +$$

$$+ 0,15*\log_2 0,15 + 0,1*\log_2 0,1 + 0,05*\log_2 0,05) = 2,41 \text{ бит.}$$

Относительная избыточность кода Хаффмана в нашем случае:

$$Q(Хаффмана, A, Binary) = 2,45/2,41 - 1 = 0,01659751.$$

Таким образом, для нашего примера код Шеннона-Фано и код Хаффмана обладают одинаковой избыточностью. Однако, в тех случаях когда вероятности символов первичного алфавита сильно разнятся, ситуация меняется. Код Хаффмана обладает существенно меньшей избыточностью. Например, для русского языка избыточность кодирования кодом Хаффмана оказывается равной примерно 0,0090.

Вопросы для самопроверки:

1. *Объясните алгоритм построения кода Хаффмана.*

2. *Докажите, что построенный по алгоритму Хаффмана код будет префиксным.*

3. *Постройте код Хаффмана для символов первичного алфавита А={«+», «-», «*», «/»} с известными вероятностями появления символов Р={0,4; 0,2; 0,3; 0,1}:*

   а) при помощи таблицы,

   б) при помощи графа.

Арифметическое кодирование.

При ответе на данный вопрос необходимо объяснить понятие «арифметическое кодирование», сравнить его с другими известными вам способами кодирования и рассказать об алгоритме построения арифметического кода некоторого сообщения.

Арифметическое кодирование — один из алгоритмов энтропийного сжатия. Алгоритм арифметического кодирования обеспечивает почти оптимальную степень сжатия с точки зрения энтропийной оценки кодирования Шеннона. На каждый символ требуется почти H бит, где H — информационная энтропия источника.

Арифметическое кодирование является методом, позволяющим упаковывать символы входного алфавита без потерь при условии, что известно распределение частот этих символов и является наиболее оптимальным, т.к. достигается теоретическая граница степени сжатия.

Последовательность символов, при сжатии методом арифметического кодирования **рассматривается** как некоторая **двоичная дробь** из интервала [0, 1). Результат сжатия представляется как последовательность двоичных цифр из записи этой дроби.

Опишем алгоритм построения этого кода. Пусть у нас есть некий первичный алфавит, состоящий из трех символов: A = {a, b, !}, а также данные о вероятности использования символов P = {0,3; 0,6; 0,1}. Последний символ рассматриваемого алфавита («!») не несет информационной нагрузки, а является признаком конца сообщения. Он необходим декодировщику для однозначного декодирования полученного сообщения. На

практике вероятность его использования берут заведомо малой. Данный символ можно не использовать только в том случае, когда декодировщику точно известна длина кодируемого сообщения.

Пусть требуется закодировать сообщение S = «bab!». Рассмотрим на координатной прямой отрезок от 0 до 1. Назовём этот отрезок **рабочим**. Расположим на нём точки таким образом, что длины образованных отрезков будут равны частоте использования символа, и каждый такой отрезок будет соответствовать одному символу.

Рабочий отрезок [0;1)		
Символ	Частота использования	Отрезок
a	0,3	[0; 0,3)
b	0,6	[0,3; 0,9)
!	0,1	[0,9; 0,1)

Первый символ сообщения S — символ «b». Ему соответствовала часть рабочего отрезка от 0,3 до 0,9. Теперь эта часть рабочего отрезка сама становится рабочим отрезком следующего этапа кодирования. Разобьём его таким образом чтобы соотношения длин получаемых отрезков соответствовало соотношению вероятностей использования символов — 3:6:1. Длина рассматриваемого рабочего отрезка d = 0,9 - 0,3 = 0,6. Разбиваем его на 10 равных частей (3 + 6 + 1 = 10) длиной 0,06 каждая. Три части (отрезок длиной 0,18) относим к рабочему отрезку буквы «a», шесть частей (отрезок длиной 0,36) — к рабочему отрезку буквы «b» и одну часть длиной 0,06 — к рабочему отрезку символа «!». Порядок следования рабочих отрезков должен соответствовать первоначальному. Получим следующую таблицу разбиения текущего рабочего отрезка:

Рабочий отрезок [0,3; 0,9)		
Символ	Частота использования	Отрезок
a	0,3	[0,3; 0,3 + 0,06*3) = **[0,3; 0,48)**
b	0,6	[0,48; 0,48 + 0,06*6) = **[0,48; 0,84)**
!	0,1	[0,84; 0,84 + 0,06*1) = **[0,84; 0,9)**

Далее в сообщении S расположен символ «a». Сейчас ему соответствует отрезок [0,3; 0,48). Выбираем этот отрезок в качестве рабочего и, согласно алгоритму, проведем его деление на части (общая длина отрезка 0,18, значит длина 1/10 части будет 0,018, для буквы «a» нужно три таких части, для «b» - шесть, для «!» - одна часть):

Рабочий отрезок [0,3;0,48)		
Символ	Частота использования	Отрезок
a	0,3	[0,3; 0,3 + 0,018*3) = **[0,3; 0,354)**
b	0,6	[0,354; 0,354 + 0,018*6) = **[0,354; 0,462)**
!	0,1	[0,462; 0,462 + 0,018*1) = **[0,462; 0,48)**

На следующем этапе рабочим отрезком будет [0,354; 0,462) — именно он соответствует новой букве в сообщении S, букве «b». Длина отрезка d = 0,462 — 0,354 = 0,108. Разделяем его на 10 частей длиной 0,0108. Букве «a» будет соответствовать часть

рассматриваемого рабочего отрезка длиной 3*0,0108 = 0,0324. Далее расположится часть рабочего отрезка буквы «b». Ее длина будет 6*0,0108 = 0,0648. Оставшаяся часть рабочего отрезка длиной 0,0108 отводится символу «!». Рассмотрим таблицу деления нового рабочего отрезка:

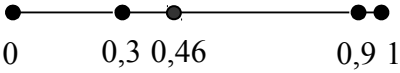
Рабочий отрезок [0,354; 0,462)		
Символ	Частота использования	Отрезок
a	0,3	[0,354; 0,3864)
b	0,6	[0,3864; 0,4512)
!	0,1	[0,4512; 0,462)

Заканчиваем сообщение символом «!». Его рабочий отрезок на данном этапе [0,4512; 0,462). Выберем любое число из рабочего отрезка, например число 0,46. Переведем его в двоичную систему счисления: $46_{10} = 0{,}0111011_2$. Биты этого числа вместе с длиной его битовой записи и есть результат арифметического кодирования использованных символов потока.

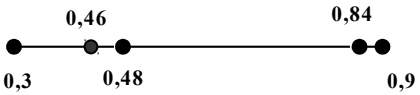
Видно, что после каждого обработанного символа текущий интервал становится все меньше, поэтому требуется все больше бит, чтобы выразить его, однако окончательным выходом алгоритма является единственное число, которое не является объединением индивидуальных кодов последовательности входных символов. Среднюю длину кода можно найти, разделив размер выхода (в битах) на размер входа (в символах).

Рассмотрим как происходит декодирование сообщения. Декодировщик получает код сообщения 0,46. Ему известен первичный алфавит и вероятности использования его символов. Он разбивает рабочий отрезок [0; 1) на отрезки согласно

вероятностям появления символов:



Полученный код принадлежит отрезку [0,3; 0,9) — это часть отрезка соответствующая букве «b». Значит первый символ закодированного сообщения это символ «b». Далее декодировщик рассматривает рабочий отрезок полученной буквы, делит его пропорционально известным вероятностям появления символов и определяет в какую часть нового рабочего отрезка попадает число 0,46 — код сообщения S.



Число 0,46 принадлежит той части рассматриваемого рабочего отрезка, которая соответствует букве «a». Это второй декодированный символ сообщения S.

Декодировщик продолжает указанное деление до тех пор, пока полученный код не попадет в рабочий отрезок символа «!», являющийся признаком конца сообщения.

Вопросы для самопроверки:

1. Объясните алгоритм построения арифметического кода.

2. Постройте арифметический код сообщения S = «2+2!» для символов первичного алфавита A={«+», «2», «!»} с известными вероятностями появления символов P={0,4; 0,5; 0,1} (символ «!» не несет смысловой нагрузки и является признаком конца сообщения).

3. Декодируйте сообщение S, составленное из символов

алфавита A={«1», «2», «3», «!»} с вероятностями P={0,4; 0,3; 0,2; 0,1}, если известен его арифметический код, записываемый в двоичной системе счисления как $0,1000101_2$

Адаптивное арифметическое кодирование.

При ответе на данный вопрос необходимо дать понятие «адаптивное арифметическое кодирование» и рассказать об алгоритме построения такого кода.

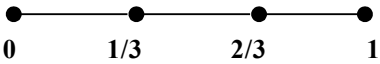
Для арифметического кодирования, позволяющего закодировать сообщение единственным числом в диапазоне [0; 1), существует адаптивный алгоритм. Такой алгоритм при каждом сопоставлении символу кода, изменяет внутренний ход вычислений так, что в следующий раз этому же символу может быть сопоставлен другой код, т.е. происходит “адаптация” алгоритма к поступающим для кодирования символам. При декодировании происходит аналогичный процесс. Необходимость применения адаптивного алгоритма возникает в том случае, если вероятностные оценки для символов сообщения неизвестны до начала работы алгоритма.

Рассмотрим на примере алгоритм адаптивного арифметического кодирования. Пусть требуется закодировать сообщение S = «bab!». Известен первичный алфавит символов {a, b, !}, но неизвестны вероятности их использования.

Сопоставим каждому символу первичного алфавита его вес. Первоначально вес всех символов равен 1. Все символы располагаются в естественном порядке, например по возрастанию. Вероятность каждого символа устанавливается равной его весу, деленному на суммарный вес всех символов. После получения очередного символа и постройки интервала для него, вес этого символа увеличивается на 1. Для того чтобы

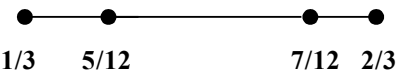
обеспечить остановку алгоритма распаковки вначале сжимаемого сообщения, надо поставить его длину или ввести дополнительный символ — признак конца сообщения (в нашем случае это символ «!»). Затем, аналогичным простому арифметическому кодированию методом, выбирается число, описывающее кодирование.

Итак, на первом этапе вес символов «a», «b» и «!» одинаков и равен 1. Соответственно, вероятности использования каждого символа также одинаковы и равны 1/3. Разделим рабочий отрезок [0; 1) на три равные части длиной 1/3:

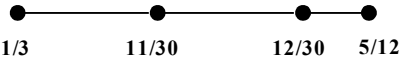


Первая часть соответствует букве «a», вторая букве «b», третья — символу «!». Первый символ сообщения S — буква «b». Ей соответствует отрезок [1/3; 2/3). Теперь он будет новым рабочим отрезком.

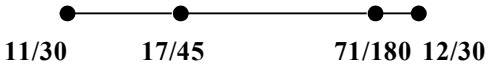
Вес буквы «b» увеличивается на единицу, веса остальных символов не изменяются. Суммарный вес всех символов меняется (сейчас он равен 1 + 2 + 1 = 4). Соответственно изменяются и вероятности использования символов: символ «a» имеет вероятность 1/4, символ «b» - вероятность 2/4 или 1/2, символ «!» - вероятность 1/4. Отношение вероятностей 1:2:1. Именно в этом отношении необходимо разделить новый рабочий отрезок (т. е. отрезок нужно разделить на 4 части длиной 1/12 каждая, по одной части отдать символам «a» и «!», две части отдать символу «b», сохранив при этом первоначальный порядок следования букв):



Следующая буква в сообщении S — буква «a». Ее рабочий отрезок [1/3; 5/12). Вес буквы «a» увеличивается на единицу. Суммарный вес всех символов теперь равен 2 + 2 + 1 = 5. Вероятности использования символов: P_a =2/5, P_b =2/5, $P_!$ =1/5. Отношение вероятностей 2:2:1. Отрезок [1/3; 5/12) делим следующим образом:



В сообщении S появляется новый символ: «b». Ей соответствовал отрезок [11/30; 12/30). Переходим к этому новому рабочему отрезку. Делим его на шесть равных частей, ведь суммарный вес символов увеличился на единицу, вследствие увеличения на единицу веса символа «b». Букве «a» соответствует две части из полученных шести, букве «b» - три части, символу «!» - одна часть:



Последний символ сообщения S — признак конца строки «!». Его рабочий отрезок на данном этапе кодирования [71/180; 12/30). Любое число из этого отрезка может быть выбрано в качестве кода сообщения S. Например 71/180 ≈ 0,394. Полученный код должен быть переведен в двоичную систему счисления и в качестве конечного результата кодирования необходимо взять его битовое представление и его длину.

Декодирование происходит следующим образом: на каждом шаге определяется интервал, содержащий данный код (выбранное число) – по этому интервалу однозначно задается исходный символ сообщения. Затем из текущего кода вычитается нижняя граница содержащего интервала, полученная

разность делится на длину этого же интервала. Полученное число считается новым текущим значением кода. Получение символа конца сообщения или заданного перед началом работы алгоритма числа символов означает окончание работы.

Вопросы для самопроверки:

1. Объясните алгоритм построения адаптивного арифметического кода.

2. Постройте адаптивный арифметический код сообщения S=«2+2=4!» для символов первичного алфавита A={«+», «=», «2», «4», «!»} (символ «!» не несет смысловой нагрузки и является признаком конца сообщения).

3. *Декодируйте сообщение S, составленное из символов алфавита A={«1», «2», «3», «!»}, если известен его адаптивный арифметический код, записываемый в двоичной системе счисления как* $0{,}1000101_2$

Хранение чисел в памяти компьютера. Дополнительный код целого положительного числа.

При ответе на данный вопрос необходимо рассказать о принципах хранения числа в памяти компьютера, а также сформулировать алгоритм получения дополнительного кода целого положительного числа без знака.

Вся информация, которую обрабатывает компьютер, должна быть представлена двоичным кодом с помощью двух цифр — 0 и 1. С помощью двух цифр 1 и 0 можно закодировать любое сообщение. С точки зрения технической реализации использование двоичной системы счисления для кодирования информации оказывается намного более простым, чем применение других способов. Действительно, удобно

кодировать информацию в виде последовательности нулей и единиц, если представить эти значения как два возможных устойчивых состояния электронного элемента:

- 0 — отсутствие электрического сигнала или сигнал имеет низкий уровень;
- 1 — наличие сигнала или сигнал имеет высокий уровень.

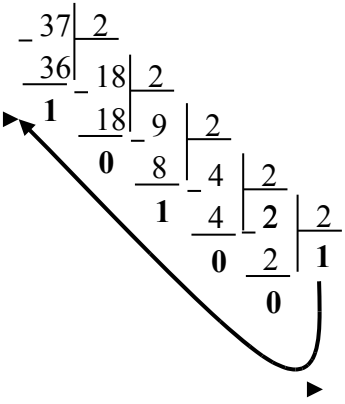
Для кодирования чисел в компьютере также используется двоичный код. При этом для представления целых чисел используется так называемый дополнительный код.

В случае представления величины со знаком самый левый (старший) разряд указывает на положительное число, если содержит нуль, и на отрицательное, если — единицу.

Известно, что диапазон значений величин зависит от количества бит памяти, отведенных для их хранения. Например, величины типа Integer языка Turbo Pascal лежат в диапазоне от -32768 (-2^{15}) до 32767 ($2^{15}-1$) и для их хранения отводится 2 байта; величины типа LongInt — в диапазоне от -2^{31} до $2^{31}-1$ и размещаются в 4 байтах; величины типа Word — в диапазоне от 0 до 65535 ($2^{16}-1$) (используется 2 байта) и т.д.

Дополнительный код положительного числа совпадает с его прямым кодом. **Прямой код** целого числа может быть получен следующим образом: модуль числа переводится в двоичную систему счисления, а затем его двоичную запись слева дополняют таким количеством незначащих нулей, сколько требует тип данных, к которому принадлежит число.

Например, необходимо получить дополнительный код числа 37_{10}. Это целое положительное число. Получим его прямой код. Для этого переведем число 37_{10} в двоичную систему счисления. Для этого последовательно поделим число на 2 и



запишем полученные остатки от деления в обратном порядке (справа налево).

Получим, что $37_{10} = 100101_2$. То есть двоичное представление числа содержит шесть цифр. Дополним его слева незначащими нулями. Как уже было сказано, их число зависит от типа числа. Если число объявлено величиной типа Integer (2 байта), то его прямым кодом будет 0000000000100101 (16 позиций), а если величиной типа LongInt, то его прямой код будет 00000000000000000000000000100101 (32 позиции). Дополнительный код рассматриваемого числа совпадает с его прямым кодом.

Для более компактной записи чаще используют шестнадцатеричный код. Полученные коды можно переписать соответственно как 0025_{16} и 00000025_{16}.

Решим обратную задачу. Запишем число, соответствующее дополнительному коду: 0000000000010111.

Поскольку в старшем разряде записан нуль, то результат будет положительным. Отбрасываем незначащие нули слева и переводим код 10111 из двоичной системы в десятичную.

$$1\cdot2^4 + 0\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 = 16 + 0 + 4 + 2 + 1 = 23_{10}$$

Следовательно заданный код - это код числа 23.

Вопросы для самопроверки:

1. Объясните алгоритм получения кода целого положительного числа.

2. *Получите шестнадцатиразрядный дополнительный код числа 65_{10}.*

3. *Определите число по его дополнительному коду:*

0001110011100001

Хранение чисел в памяти компьютера. Дополнительный код целого отрицательного числа.

При ответе на данный вопрос необходимо рассказать о принципах хранения числа в памяти компьютера, а также сформулировать алгоритм получения дополнительного кода целого отрицательного числа.

Все данные в компьютере хранятся в виде набора двоичных кодов — последовательности нулей и единиц. Числовые данные могут быть интерпретированы как числа со знаками, так и без знаков. В случае представления величины со знаком самый левый разряд указывает на положительное число, если содержит нуль, и на отрицательное, если — единицу. Разряды нумеруются справа налево, начиная с 0.

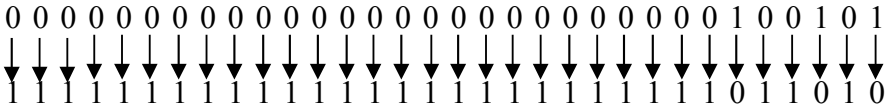
Дополнительный код целого отрицательного числа может быть получен по следующему **алгоритму**:

- записать прямой код модуля числа;
- инвертировать его (заменить единицы нулями, нули — единицами);
- прибавить к инверсному коду единицу.

Рассмотрим пример. Запишем дополнительный код числа $(-37)_{10}$, интерпретируя его как величину типа LongInt, для хранения которой отводится 4 байта.

Для получения прямого кода модуля числа $(-37)_{10}$ переведем его модуль - число 37_{10} - в двоичную систему счисления. Получим число 100101_2. Так как для хранения отводится 4 байта, т. е. 32 бита или 32 ячейки памяти, то прямой код числа 37_{10} есть 00000000000000000000000000100101.

Инвертируем полученный прямой код:



В результате получим инверсный код:

11111111111111111111111111011010.

Прибавим к инверсному коду единицу:

$$\begin{array}{r} 11111111111111111111111111011010 \\ +\ 1 \\ \hline 11111111111111111111111111011011 \end{array}$$

Значит, дополнительный код числа $(-37)_{10}$ будет 11111111111111111111111111011011 или $FFFFFFDB_{(16)}$.

Если необходимо получить **число по его дополнительному коду**, то прежде всего, определяем его знак по значению старшего бита. Если значение старшего бита будет равно единице, то число отрицательное. В этом случае необходимо выполнить следующие действия:

- вычесть из дополнительного кода числа единицу;
- инвертировать код;
- отбросить незначащие нули слева и перевести полученный код в десятичную систему счисления;
- полученное число записать со знаком минус.

Например, пусть известно, что некоторое число имеет дополнительный код 1111111111000000. Необходимо определить само число.

По значению старшего бита определяем знак числа. Число отрицательное. Вычитаем из его дополнительного кода единицу (действия производим в двоичной системе счисления):

$1111111111000000_2 - 1_2 = 1111111110111111_2$

Инвертируем полученный код: 0000000001000000

Отбрасываем незначащие нули и переводим полученное число в десятичную систему счисления:

$1000000_2 = 1*2^6 = 64_{10}$

Запишем найденное число со знаком «минус»: –64.

Вопросы для самопроверки:

1. Объясните алгоритм получения кода целого отрицательного числа.

2. *Получите четырехбайтный дополнительный код числа* $(-48)_{10}$.

3. Определите число по его дополнительному коду:

1001110011100001

Хранение чисел в памяти компьютера. Дополнительный код вещественного числа.

При ответе на данный вопрос необходимо рассказать о способе хранения в памяти компьютера вещественных чисел, привести пример нормализованного вида числа, мантиссы, порядка, объяснить алгоритм вычисления смещенного порядка вещественного числа.

Известно, что любое действительное число можно записать в стандартном виде $M \times 10^p$, где $1 \le M < 10$, p — целое. Например, $120100000 = 1{,}201 \times 10^8$. Поскольку каждая позиция десятичного числа отличается от соседней на степень числа 10, умножение на 10 эквивалентно сдвигу десятичной запятой на одну позицию вправо. Аналогично деление на 10 сдвигает десятичную запятую на позицию влево. Поэтому приведенный выше пример можно продолжить: $120100000 = 1{,}201 \times 10^8 = 0{,}1201 \times 10^9 = 12{,}01 \times 10^7$. Десятичная запятая «плавает» в числе и больше не помечает абсолютное место между целой и дробной частями.

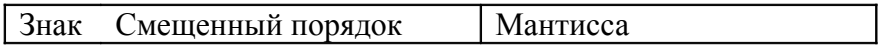
В приведенной выше записи M называют **мантиссой** числа, а p — его **порядком**. Числа в компьютере сохраняются в виде набора двоичных кодов. Для того чтобы сохранить максимальную точность, вычислительные машины почти всегда хранят мантиссу в нормализованном виде, что означает, что мантисса в данном случае есть число, лежащее между 1_{10} (в двоичной системе это 1_2) и 2_{10} (в двоичной системе - 10_2). Способ хранения мантиссы с плавающей точкой подразумевает, что двоичная запятая находится на фиксированном месте. Фактически подразумевается, что двоичная запятая следует после первой двоичной цифры, т.е. нормализация мантиссы делает единичным первый бит, помещая тем самым значение между десятичной единицей и двойкой. Место, отводимое для числа с плавающей точкой, делится на два поля. Одно поле содержит знак и значение мантиссы, а другое содержит знак и значение порядка.

Размер памяти отводимый под хранение числа определяется его типом. Так для хранения числа в самом распространенном вещественном формате real отводится шесть байт. Размеры памяти, отводимые для других вещественных форматов представлены в таблице:

Тип числа	Диапазон значений	Количество значащих цифр	Байты
Single	$1{,}5 \times 10^{-45} .. 3{,}4 \times 10^{38}$	7–8	4
Real	$2{,}9 \times 10^{-39} .. 1{,}7 \times 10^{38}$	11–12	6
Double	$5{,}0 \times 10^{-324} .. 1{,}7 \times 10^{308}$	15–16	8
Extended	$3{,}4 \times 10^{-4932} .. 1{,}1 \times 10^{4932}$	19–20	10

Покажем преобразование действительного числа для представления его в памяти компьютера на примере величины типа Real.

Как видно из таблицы, величина это типа занимает в памяти 6 байт или 48 бит. Как известно, биты нумеруются справа налево и нумерация начинается с нуля. Значит бит с номером 47 будет знаковым, следующие 11 бит отводятся на хранение смещенного порядка, далее ячейки памяти занимаются мантиссой. Ниже показано, как здесь представлены поля мантиссы и порядка:



Можно заметить, что старший бит, отведенный под мантиссу, имеет номер 35, т.е. мантисса занимает младшие 36 бит. Черта указывает здесь на положение двоичной запятой. Перед запятой должен стоять бит целой части мантиссы, но поскольку она всегда равна 1, здесь данный бит не требуется и соответствующий разряд отсутствует в памяти (но он подразумевается).

Для упрощения вычислений и сравнения действительных чисел значение порядка в памяти компьютера хранится в виде **смещенного числа**, т.е. к настоящему значению порядка перед записью его в память прибавляется смещение.

Смещение выбирается так, чтобы минимальному значению порядка соответствовал нуль. Например, для типа Real порядок занимает 11 бит. Самое большое число, которое можно разместить в 11 битах, это число с дополнительным кодом 01111111111, т.е. число 1023. Самое маленькое число, хранимое в одиннадцати битах (-1023). Поэтому диапазон порядка будет от –1023 до 1023. Для того чтобы минимальному значению порядка соответствовал нуль, необходимо, чтобы смещение было равно $1023_{10} = 1111111111_2$.

Сформулируем **алгоритм** для получения представления действительного числа в памяти компьютера:

- перевести модуль данного числа в двоичную систему счисления;
- нормализовать двоичное число, т.е. записать в виде $M \times 2^p$, где M — мантисса (ее целая часть равна 1_2) и p — порядок, записанный в десятичной системе счисления;
- прибавить к порядку смещение и перевести смещенный порядок в двоичную систему счисления;
- определить значение старшего бита по знаку числа (0 для положительного числа, 1 для отрицательного), выписать его представление в памяти компьютера.

Рассмотрим пример. Запишем код числа –312,3125 интерпретируя его как число типа Real.

Двоичная запись модуля этого числа имеет вид 100111000,0101. Нормализируем полученное двоичное представление. Имеем $100111000{,}0101 = 1{,}001110000101 \times 2^8$. Как уже было сказано, для типа Real смещение равно 1023. Прибавим его к порядку и получим смещенный порядок $8 + 1023 = 1031$. Далее переводим смещенный порядок в двоичную систему счисления. Имеем $1031_{10} = 10000000111_2$.

Окончательно получаем следующее представление числа -312,3125:

1	10000000111	00111000010100000000000000000000000000
знак	смещенный порядок	мантисса

Полученный код можно записать в шестнадцатиричной системе: $C07385000000_{16}$.

Рассмотрим пример обратной задачи: перейдем от кода действительного числа к самому числу. Пусть дан код $3FEC60000000_{16}$. Переведя его в двоичную систему счисления получим: 00111111111011000110000000000000000000000000000000

Прежде всего, замечаем, что это код положительного числа, поскольку в старшем разряде записан нуль. Следующие 11 бит отведены на смещенный порядок - 01111111110. Переведем это двоичное число в десятичную систему счисления: $01111111110_2 = 1022_{10}$. Вычитая смещения, получим порядок числа: $1022 – 1023 = –1$. Число имеет вид $1{,}1100011 \times 2^{-1}$ или 0,11100011. Переводом в десятичную систему счисления получаем 0,88671875.

Вопросы для самопроверки:

1. Объясните алгоритм получения кода вещественного числа.

2. *Переведите число $(-56{,}375)_{10}$ в двоичную систему счисления и запишите его в нормализированном виде.*

3. *Получите дополнительный код числа $65{,}025_{10}$, интерпритируя его как число типа Double.*

Цифровые коды. Двоично-десятичное кодирование.

При ответе на данный вопрос необходимо рассказать о

способе построения двоично-десятичного кода, о требованиях, предъявляемых к весам разрядов двоично-десятичных кодов, о принципах работы двоично-десятичных сумматоров.

В двоично-десятичном коде основной системой счисления является десятичная. Однако каждая значащая десятичная цифра в двоично-десятичном коде представляется четырьмя двоичными знаками и содержит десять значений сигнала от 0 до 9. Так, например, десятичное число 10 можно представить как 0001 0000, а десятичное число 99 можно представить как 1001 1001.

Так как при кодировании четырьмя двоичными знаками можно получить 16 кодовых значений, то приведенное двоично-десятичное представление не является единственным. Например, широко используют двоично-десятичные коды с весами 2-4-2-1 и 5-1-2-1. Покажем как представляются в этих кодах десятичные цифры:

Десятичный код	Двоично-десятичный код		
	8-4-2-1	2-4-2-1	5-1-2-1
0	0000 0000	0000 0000	0000 0000
1	0000 0001	0000 0001	0000 0001 или 0000 0100
2	0000 0010	0000 0010 или 0000 1000	0000 0010 или 0000 0101
3	0000 0011	0000 0011 или 0000 1001	0000 0011 или 0000 0110

Десятичный код	Двоично-десятичный код		
	8-4-2-1	2-4-2-1	5-1-2-1
4	0000 0100	0000 0100 или 0000 1010	0000 0111
5	0000 0101	0000 0101 или 0000 1011	0000 1000
6	0000 0110	0000 0110 или 0000 1100	0000 1001 или 0000 1100
7	0000 0111	0000 0111 или 0000 1101	0000 1101 или 0000 1010
8	0000 1000	0000 1110	0000 1110 или 0000 1011
9	0000 1001	0000 1111	0000 1111

Для формирования чисел мспользуем тот же принцип. Так, десятичное число 15 в двоично-десятичном коде с весами 8-4-2-1 будет представлено в следующем виде 0001 0101.

Как видно из таблицы, практически все двоично-десятичные коды не имеют однозначности в отображении. Исключением является код с весами 8-4-2-1.

При построении двоично-десятичного кода с весами q_4-q_3-q_2-q_1 необходимо учитывать следующие **условия**:

- Вес наименьшей значащей цифры (q_1) равен 1;
- Вес второй значащей цифры (q_2) равен 1 или 2;

• Вес, соответствующий двум оставшимся цифрам кода, должен быть не меньше семи ($q_3+q_4\geq 7$), если $q_2=1$, и не меньше шести ($q_3+q_4\geq 6$), если $q_2=2$;

• Совокупность весов должна удовлетворять соотношению $q_4-(q_1+q_2+q_3)\leq 1$.

Двоично-десятичные коды широко применяются в АЦП, предназначенных для различных цифровых измерительных приборов. Каждая значимая десятичная цифра в таком коде представляется четырьмя двоичными знаками и содержит десять значений сигнала от 0 до 9.

Для кода 8-4-2-1 представляется возможным производить арифметические операции на двоично-десятичных сумматорах, которые проектируют как обычные двоичные сумматоры, добавляя лишь устройства формирования дополнительных переносов, необходимых в тех случаях, когда сумма двух двоично-десятичных чисел S становится больше или равна 10. Причем, если $10\leq S\leq 15$, то после переноса в следующую четверть из суммы необходимо вычитать число 10 (1010), а если S = 16, то к сумме после переноса необходимо добавить 6 (0110). Например, при сложении двух двоично-десятичных чисел 0111 и 0100 получится число 1011, которое в двоично-десятичном изображении не предусмотрено. После переноса и коррекции суммы получим число 0001 0001.

Для упрощения двоично-десятичных счетчиков процедуру вычитания числа 1010 заменяют двумя процедурами: вычитания числа 16 и добавления 6, что сводится к добавлению к сумме двоично-десятичного числа 01010 и переносу единицы в следующую четверть без восстановления. Так, для рассмотренного примера получим 1011+ 0110 = 1 0001.

Описанный способ построения двоично-десятичных

счетчиков не исключает и возможности преобразования двоично-десятичного кода в натуральный двоичный код с последующим проведением арифметических операций.

Вопросы для самопроверки:

1. Сформулируйте условия, необходимые для построения двоично-десятичного кода.

2. *Получите двоично-десятичный код с весами 8-4-2-1 для числа 65_{10}.*

3. Расскажите о практическом применении двоично-десятичных кодов.

Цифровые коды. Код Грея.

При ответе на данный вопрос необходимо рассказать о характерных особенностях, алгоритме кодирования и правилах декодирования кода Грея, а также о способах его применения на практике.

Особое место среди позиционных двоичных кодов занимает циклический код, называемый кодом Грея. Характерной особенностью этого кода является изменение только одной позиции при переходе от одного кодовой комбинации к другой. Это свойство кода Грея широко используют как для построения некоторых типов АЦП, так и для повышения надежности преобразователей с помощью резервирования и самоконтроля. Используется в технике аналогово-цифровых преобразователях, где он позволяет свести к «1» младшего разряда погрешность неоднозначности при считывании.

Рассмотрим алгоритм построения кода Грея. Код Грея можно построить на основе натурального двоичного кода числа.

Для перехода от натурального двоичного кода к коду Грея существуют **правила**:

- если в предыдущем разряде двоичного кода стоит 0, то в данном разряде цифра сохраняется;
- если в предыдущем разряде двоичного кода стоит 1, то в данном разряде цифра меняется на противоположную (цифра 0 изменится на 1 и наоборот).

Рассмотрим пример. Возьмем целые десятичные числа от 0 до 15. Запишем их двоичное представление, выделив для хранения каждого числа полбайта:

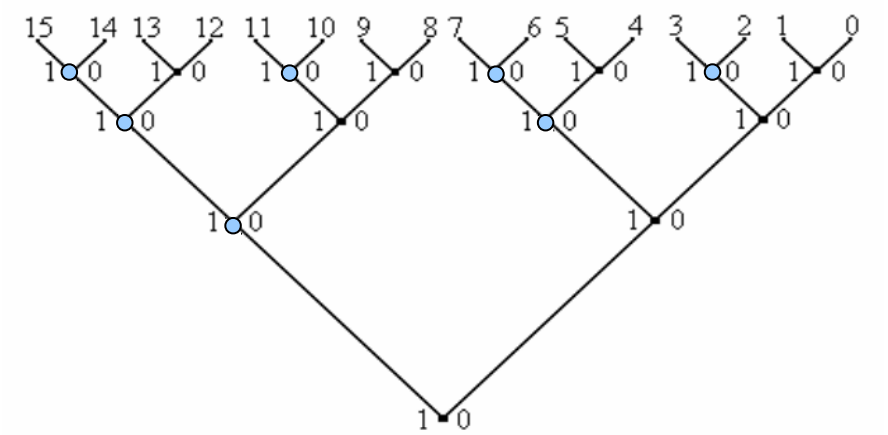
$\mathbf{A_{10}}$	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
$\mathbf{A_2}$	0000	0001	0010	0011	0100	0101	0110	0111
$\mathbf{A_{10}}$	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
$\mathbf{A_2}$	1000	1001	1010	1011	1100	1101	1110	1111

Перейдем к построению кода Грея. Натуральный двоичный код числа 0 (0000) не содержит ни одной единицы. Следовательно, согласно правилу построения ни одна его цифра в коде Грея не изменится. Код числа 1 (0001) содержит одну единицу только в самом последнем справа разряде. Разряда для которого она являлась бы предыдущей нет. Значит натуральный двоичный код числа 1 совпадает с кодом Грея числа 1. А вот двоичный код числа 2 (0010) содержит единицу во третьем слева разряде. Согласно правилу построения кода Грея, следующий за ней разряд должен изменить свою цифру на противоположную. Значит код Грея для числа 2 будет 0011. Аналогичным образом можно получить код Грея для оставшихся чисел. Результат такого преобразования представлен в таблице. Разряды двоичного кода, подвергнутые изменению подчеркнуты:

A_{10}	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
A_2	0000	0001	0010	0011	0100	0101	0110	0111
Код Грея	0000	0001	0011	0010	0110	0111	0101	0100
A_{10}	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
A_2	1000	1001	1010	1011	1100	1101	1110	1111
Код Грея	1100	1101	1111	1110	1010	1011	1001	1000

Код Грея можно построить и с помощью графа. Покажем алгоритм построения на рассмотренном примере.

Мы имеем числа от 0 до 15. Расположим их на концевых узлах будущего дерева. Объединим парами данные числа. При этом слева от каждого узла, поставим единицу, а справа нуль.



Для получения по этой схеме кода Грея нужно в каждой второй справа в своем ряду «вилке» (на рисунке эти узлы отмечены знаком ○) поменять местами 1 и 0. Теперь при прохождении по измененному графу от корня до концевых узлов мы сможем определять код Грея для выбранного числа.

Основными трудностями, ограничивающими применение кодов Грея, является непостоянство веса каждого разряда и изменение его знака. Выясним, как определяется вес и знак разряда кода Грея.

Выберем кодовые комбинации, содержащие только одну единицу: 0001, 0010, 0100, 1000. Этим комбинациям соответствуют числа в десятичной системе счисления: 1, 3, 7, 15, которые определяют вес каждого разряда.

С другой стороны, вес разряда может быть как положительным, так и отрицательным. Например, число 2 имеет код Грея 0011. Покажем ее представление с учетом веса каждого разряда:

$15*0 + 7*0 + 3*1 + (-1)*1 = 3-1 = 2$

Получили, что вес второго разряда положительный, а первого отрицательный (нумерация разрядов идет справа налево).

Рассмотрим еще один пример. Получим представление числа 10. Его код Грея 1111. С учетом весовых коэффициентов имеем:

$15*1 + (-7)*1 + 3*1 + (-1)*1 = 15 - 7 + 3 - 1 = 10$

Здесь два разряда имеют положительный вес, а два отрицательный.

Исследование особенностей построения кода Грея позволяет сделать вывод: его недостатком является то, что в нем затруднено, хотя и возможно, выполнение арифметических операций и цифроаналоговое преобразование.

Поэтому в тех случаях, когда эти операции необходимы, параллельный код Грея превращают в натуральный двоичный, а

уже затем осуществляют арифметические операции или цифроаналоговое преобразование.

Для перехода от кода Грея к натуральному двоичному коду используют следующее **правило**: если слева от данной цифры находится четное число единиц, то цифра сохраняется, в противном случае цифра меняется.

Например имеет код Грея некоторого числа: 1010. Необходимо получить само число. Рассмотрим разряды кода слева направо. Обозначим их q_1, q_2, q_3, q_4. Левее q_1 других цифр нет, значит она не меняется. Второй разряд q_2=0, слева от него находиться единственная единица, значит значение этого разряда меняется на на 1. Следующий разряд q_3 =1, в коде Грея ему также предшествует одна единица в разряде q_1. Следовательно цифра разряда q_3 поменяется на противоположную (q_3 = 0). А вот четвертому разряду кода Грея предшествуют уже две единицы в разрядах q_1 и q_3, соответственно, значение этого разряда должно измениться. Окончательно имеем следующий двоичный код: 1100. Это двоичное представление числа 12.

Вопросы для самопроверки:

1. Сформулируйте правила перехода от натурального двоичного кода к коду Грея.

2. Объясните алгоритм построения кода Грея с помощью графа.

3. Сформулируйте алгоритм перехода от кода Грея к натуральному двоичному коду.

Помехоустойчивое кодирование: основная идея, используемые понятия.

При ответе на данный вопрос необходимо дать понятие помехоустойчивого кода, рассказать о классификации помехоустойчивых кодов и основных правилах их построения.

Теория помехоустойчивого кодирования базируется на результатах исследований, проведенных Шенноном и сформулированных им в виде **основной теоремы для дискретного канала с шумом**: при любой скорости передачи двоичных символов меньшей, чем пропускная способность канала, существует такой код, при котором вероятность ошибочного декодирования будет сколь угодно мала; вероятность ошибки не может быть сделана произвольно малой, если скорость передачи больше пропускной способности канала.

Кодирование должно осуществляться так, чтобы сигнал, соответствующий принятой последовательности символов, после воздействия на него предполагаемой в канале помехи оставался ближе к сигналу, соответствующему конкретной переданной последовательности символов, чем к сигналам, соответствующим другим возможным последовательностям (степень близости обычно определяется по числу разрядов, в которых последовательности отличаются друг от друга).

Это достигается ценой введения при кодировании избыточности, которая позволяет так выбрать передаваемые последовательности символов, чтобы они удовлетворяли дополнительным условиям, проверка которых на приемной стороне дает возможность обнаружить и исправить ошибки. Коды, обладающие таким свойством, получили название **помехоустойчивых**. Они используются как для исправления ошибок (корректирующие коды), так и для их обнаружения.

У подавляющего большинства существующих в настоящее время помехоустойчивых кодов указанные выше условия являются следствием их алгебраической структуры. В связи с этим их называют **алгебраическими кодами**.

Алгебраические коды можно подразделить на два больших класса: блоковые и непрерывные.

В случае **блоковых** кодов процедура кодирования заключается в сопоставлении каждой букве сообщения (или последовательности из k символов, соответствующей этой букве) блока из n символов, причем в операциях по преобразованию принимают участие только указанные k символов и выходная последовательность не зависит от других символов в передаваемом сообщении.

Блоковый код называется **равномерным**, если n остается постоянным для всех букв сообщения. Различают разделимые и неразделимые блоковые коды. При кодировании **разделимыми** кодами выходные последовательности состоят из символов, роль которых может быть отчетливо разграничена. Это информационные символы, совпадающие с символами последовательности, поступающей на вход кодера канала, и избыточные (проверочные) символы, вводимые в исходную последовательность кодером канала и служащие для обнаружения и исправления ошибок.

При кодировании **неразделимыми** кодами разделить символы выходной последовательности на информационные и проверочные невозможно.

Непрерывными называются такие коды, в которых введение избыточных символов в кодируемую последовательность информационных символов осуществляется непрерывно, без разделения ее на независимые блоки.

Непрерывные коды также могут быть разделимыми и неразделимыми.

При взаимно независимых ошибках наиболее вероятен переход в кодовую комбинацию, отличающуюся от данной в наименьшем числе символов. Степень различия любых двух кодовых комбинаций характеризуется расстоянием между ними (по Хэммингу), или просто **кодовым расстоянием**. Оно выражается числом символов, в которых комбинации отличаются одна от другой, и обозначается через d.

Чтобы **рассчитать кодовое расстояние** между двумя комбинациями двоичного кода, достаточно подсчитать число единиц в сумме этих комбинаций по модулю 2. Например заданы две кодовые комбинации A и B. Требуется определить кодовое расстояние. Складывая по модулю 2 каждый разряд A и B, получаем некоторую комбинацию C. Непосредственный подсчет единиц определяет вес ϖ(C) кодовой комбинации C, который равен кодовому расстоянию d.

A: 1 0 0 1 1 1 1 1 0 1 ϖ(A)=7
+
B: 1 1 0 0 0 0 1 0 1 0 ϖ(B)=4
C: 0 1 0 1 1 1 0 1 1 1 ϖ(C)=7.

Следовательно, расстояние Хемминга для данных кодовых комбинаций d=7. Минимальное расстояние, взятое по всем парам кодовых комбинаций данного кода, называется **минимальным кодовым расстоянием**.

Декодирование после приема может производиться таким образом, что принятая кодовая комбинация отождествляется с той разрешенной, которая отличается от полученной в наименьшем числе символов.

Такое декодирование называется декодированием по методу **максимального правдоподобия**.

Вопросы для самопроверки:

1. Сформулируйте основную теорему для дискретного канала с шумом.

2. Расскажите о классификации помехоустойчивых кодов.

3. Вычислите расстояние Хемминга для следующей пары кодовых комбинаций: 10101001 и 11101101.

Шифрование. Основные понятия.

При ответе на данный вопрос необходимо раскрыть основные понятия теории шифрования (шифр, шифрование, ключ), дать классификацию шифров.

Шифр - система преобразования текста для обеспечения секретности передаваемой информации.

Сам термин происходит от арабского *sifr* - «ноль», преобразованного впоследствии во французское *chiffre* - «цифра». Шифры применяются для тайной переписки дипломатических представителей со своими правительствами, а также в вооруженных силах для передачи текста секретных документов по техническим средствам связи.

Шифр может представлять собой совокупность условных знаков (условная азбука из цифр или букв) либо алгоритм кодирования с использованием обычных цифр и букв. Процесс засекречивания сообщения с помощью шифра называется **шифрованием**. Наука о создании и использовании шифров называется **криптографией**. Наука о методах получения исходного значения зашифрованной информации называется

криптоанализ. Слово «криптограф» происходит от древнегреческих слов kryptos 'секрет' и graphos 'писание'. Исходное сообщение называется в криптографии **открытым текстом**, или **клером**. Засекреченное (зашифрованное) сообщение называется **шифротекстом**, или **шифрограммой**, или криптограммой.

Важным параметром любого шифра является **ключ** — параметр криптографического алгоритма, обеспечивающий выбор одного преобразования из совокупности преобразований, возможных для этого алгоритма. В современной криптографии предполагается, что вся секретность криптографического алгоритма сосредоточена в ключе, но не деталях самого алгоритма (принцип Керкгоффса).

Шифры могут использовать один ключ для шифрования и дешифрования или два различных ключа. По этому признаку различают:

- **Симметричный** шифр использует один ключ для шифрования и дешифрования. Ключ алгоритма должен сохраняться в секрете обеими сторонами. Алгоритм шифрования выбирается сторонами до начала обмена сообщениями. Классическим примером таких алгоритмов являются **симметричные криптографические алгоритмы**: простая перестановка, одиночная перестановка по ключу, двойная перестановка, перестановка «Магический квадрат».
- **Асимметричный** шифр использует два различных ключа. Асимметричная система шифрования и/или электронной цифровой подписи (ЭЦП), при которой **открытый ключ** передаётся по открытому (то есть незащищённому, доступному для наблюдения) каналу, и используется для проверки ЭЦП и для шифрования сообщения. Для генерации ЭЦП и для расшифровки сообщения используется **секретный ключ**.

Криптографические системы с открытым ключом в настоящее время широко применяются в различных сетевых протоколах, в частности, в протоколах TLS и его предшественнике SSL (лежащих в основе HTTPS), в SSH.

Шифры могут быть сконструированы так, чтобы либо шифровать сразу весь текст, либо шифровать его по мере поступления. Таким образом существуют:

- **Блочный** шифр, который обрабатывает открытый текст блоками по несколько (как правило 8 или 16) байт за одну итерацию. Если исходный текст (или его остаток) меньше размера блока, перед шифрованием его дополняют. Преобразование текста должно использовать **принципы рассеивания** (изменение любого знака открытого текста или ключа влияет на большое число знаков шифротекста, что скрывает статистические свойства открытого текста) и **перемешивания** (использование преобразований, затрудняющих получение статистических зависимостей между шифротектстом и открытым текстом). К достоинствам блочных шифров относят похожесть процедур шифрования и расшифрования, которые, как правило, отличаются лишь порядком действий. Это упрощает создание устройств шифрования, так как позволяет использовать одни и те же блоки в цепях шифрования и дешифрования.
- **Поточный** шифр шифрует информацию и выдает шифротекст по мере поступления, таким образом имея возможность обрабатывать текст неограниченного размера используя фиксированный объем памяти. При использовании поточного шифра каждый символ открытого текста преобразуется в символ шифрованного текста в зависимости не только от используемого ключа, но и от его расположения в потоке открытого текста.

Естественно, что блочный шифр можно превратить в поточный, разбивая входные данные на отдельные блоки и шифруя их по отдельности. Важнейшим достоинством поточных шифров перед блочными является высокая скорость шифрования, соизмеримая со скоростью поступления входной информации; поэтому, обеспечивается шифрование практически в реальном масштабе времени вне зависимости от объема и разрядности потока преобразуемых данных. В синхронных поточных шифрах (в отличие от блочных) отсутствует эффект размножения ошибок, то есть число искаженных элементов в расшифрованной последовательности равно числу искаженных элементов зашифрованной последовательности, пришедшей из канала связи. Структура поточного ключа может иметь уязвимые места, которые дают возможность криптоаналитику получить дополнительную информацию о ключе (например, при малом периоде ключа криптоаналитик может использовать найденные части поточного ключа для дешифрования последующего закрытого текста). Поточные шифры в отличие от блочных достаточно часто взламываются при помощи линейной алгебры. Также для взлома поточных шифров весьма успешно применяется линейный и дифференциальный анализ.

Вопросы для самопроверки:

1. *Приведите примеры шифра и шифрования.*
2. *Определите предмет и объект исследования криптографии и криптоанализа.*
3. *Расскажите о классификации шифров.*

Основные криптографические алгоритмы. Алгоритм замены.

При ответе на данный вопрос необходимо рассказать о криптографическом алгоритме замены и проиллюстрировать способы его построения на конкретных примерах.

К основным криптографическим алгоритмам относят алгоритмы замены, перестановки и дробления. Эти три базовых преобразования самостоятельно или в сочетании друг с другом используются в большинстве систем шифрования для создания очень сложных шифровальных алгоритмов.

Рассмотрим подробнее алгоритм замены. Пусть требуется зашифровать следующее сообщение (открытый текст):

МАМА МЫЛА РАМУ.

Один из способов шифрования – **простая замена**, при которой каждая буква открытого текста заменяется на какую-то букву алфавита (возможно, на ту же самую). Для этого отправитель сообщения должен знать, на какую букву в шифротексте следует заменить каждую букву открытого текста. Часто это делается путем сведения нужных соответствий букв в виде двух алфавитов, например так, как показано ниже в таблице:

Открытый алфавит	А	Л	М	Р	У	Ы
Шифровальный алфавит	А	Б	В	Г	Д	Е

Шифрограмма получается путем замены каждой буквы открытого текста на записанную непосредственно под ней букву шифровального алфавита:

ВАВА ВЕБА ГАВД

Две алфавитные последовательности, используемые в процессе шифрования, называются, соответственно, **открытым** и **шифровальным компонентом системы**. Чтобы получатель шифрограммы мог восстановить открытый текст и прочитать сообщение, ему необходимо иметь копию вышеприведенной таблицы. Дешифровщик повторяет в обратном порядке все действия шифровальщика, раскрывая тем самым содержание сообщения.

В вышеприведенном примере использовался алгоритм побуквенной замены. Этот метод называется простой, или **моноалфавитной заменой**. Ключ к данному шифру состоит из таблицы, содержащей открытый и шифровальный алфавиты, в которой указывается, на какую букву в шифротексте следует заменить букву открытого текста. В такой криптографической системе предполагается, что алгоритм шифрования общеизвестен, тогда как ключ доступен только отправителю и получателю соответствующих сообщений.

Для формирования шифровального алфавита можно использовать ключевое слово (ключевую фразу). В этом случае в начале шифровального алфавита записывается начало ключевого слова или первого слова ключевой фразы, потом алфавит формируется с опущением всех тех букв, что уже появились в этом слове (или в первом слове этой фразы), а после этого вписываются остающиеся буквы алфавита в обычном порядке, опять же с опущением всех ранее появившихся букв. Так, если в качестве ключевой мы используем фразу У ЛУКОМОРЬЯ ДУБ ЗЕЛЕНЫЙ, то шифровальный алфавит будет выглядеть следующим образом:

У Л К О М Р Ь Я Д Б З Е Н Ы Й А В Г Ё Ж И П С Т Ф Х Ц

Ч Ш Щ Ъ Э Ю

С помощью ключевого слова (фразы) при шифровании можно перемешать любую алфавитную последовательность. Использование ключевых слов облегчает восстановление открытого и шифровального компонента системы, поскольку при этом необходимо запомнить только соответствующее ключевое слово (фразу). Нет необходимости записывать (или разгадывать) какие бы то ни было таблицы: если помнить ключевое слово, то алфавитную последовательность всегда можно восстановить по памяти.

В вышеприведенной шифрограмме между словами сохранены пробелы, однако шифровку можно сделать более защищенной (или, как говорят криптографы, устойчивой, или стойкой ко взлому; шифр считается тем более стойким, чем дольше он не поддается вскрытию) путем удаления межсловных пробелов из окончательного шифротекста. Согласно установившейся практике, шифротекст принято делить на группы из пяти букв каждая. Если убрать пробелы между словами, то нашу шифрограмму можно было бы записать так:

ВАВАВ ЕБАГА ВД

Вопросы для самопроверки:

1. Поясните понятия «открытый алфавит», «шифровальный алфавит».

2. Приведите пример алгоритма простой замены.

3. Приведите пример шифрования при помощи ключевого слова.

Основные криптографические алгоритмы. Алгоритм перестановки.

При ответе на данный вопрос необходимо рассказать о криптографическом алгоритме перестановки и проиллюстрировать способы его построения на конкретных примерах.

В криптографическом **алгоритме перестановки** все **буквы** открытого текста остаются без изменений, но **переставляются** согласно заранее оговоренному правилу. Здесь также удобно использовать ключ, управляющий процедурой шифрования. Возьмем в качестве ключа для сообщения МАМА МЫЛА РАМУ слово БАЙТ. Пронумеруем буквы ключевого слова в порядке их следования слева направо в русском алфавите:

Буквы ключевого слова	Б	А	Й	Т
Порядковые номера букв в алфавите	2	1	3	4

Далее под полученной числовой последовательностью в строках, равных по длине ключевому слову, запишем открытый текст.

Буквы ключевого слова	Б	А	Й	Т
Порядковые номера букв в алфавите	2	1	3	4
Открытый текст	М	А	М	А
	М	Ы	Л	А
	Р	А	М	У

В процессе шифрования текст будем выписывать уже по

отдельным столбцам в порядке, определяемом данной числовой последовательностью

АЫАММРМЛМААУ

Этот метод перестановки называется **перестановкой столбцов**, но можно избрать и другие «маршруты» перестановки, например выписывать шифротекст следуя по диагонали (слева направо или справа налево, или же чередуя левое и правое направления) или по спирали и т.п. Кроме того, буквы шифротекста могут записываться в виде различных геометрических фигур или любыми другими способами. Один из них состоит в двойном шифровании путем повторной перестановки столбцов. При этом и в первом, и во втором блоках перестановки может быть использовано одно и то же ключевое слово, хотя лучше использовать разные ключевые слова. Такой шифр, называющийся двойной перестановкой, получил широкое распространение.

Вопросы для самопроверки:

1. Объясните алгоритм метода перестановки столбцов.

2. Зашифруйте сообщение У ЛУКОМОРЬЯ ДУБ ЗЕЛЕНЫЙ методом перестановки столбцов с ключевым словом БУКВА.

*3. Зашифруйте сообщение У ЛУКОМОРЬЯ ДУБ ЗЕЛЕНЫЙ по следующим маршрутам перестановки: текст вписывается в таблицу размером 5*5 клеток по спирали, начиная с верхнего левого угла по часовой стрелке, а выписывается по столбцам сверху вниз, начиная с последнего столбца.*

Основные криптографические алгоритмы. Алгоритм дробления.

При ответе на данный вопрос необходимо рассказать о

криптографическом алгоритме дробления и проиллюстрировать способы его построения на конкретных примерах.

В **алгоритме дробления** каждой букве открытого текста сопоставляется более одного символа шифротекста, после чего **символы перемешиваются** (переставляются) в определенном порядке.

Рассмотрим подробнее алгоритм дробления. Сначала составляется шифровальная таблица размером 6∗6 клеток, куда построчно вписывается шифровальный алфавит с ключевой фразой. Пусть ключевая фраза будет У ЛУКОМОРЬЯ ДУБ ЗЕЛЕНЫЙ. Также как и в алгоритме замены запишем в таблицу начало первого слова ключевой фразы, а далее сформируем шифровальный алфавит, исключая уже записанные буквы. После окончания ключевой фразы впишем в таблицу остающиеся буквы алфавита в обычном порядке, но с учетом ранее появившихся букв:

	1	**2**	**3**	**4**	**5**	**6**
1	У	Л	К	О	М	Р
2	Ь	Я	Д	Б	З	Е
3	Н	Ы	Й	А	В	Г
4	Ё	Ж	И	П	С	Т
5	Ф	Х	Ц	Ч	Ш	Щ
6	Ъ	Э	Ю	+	-	*

Получается, что букве У соответствует пара чисел (1, 1), букве Л — пара (1, 2) и т.д. Зашифруем с помощью этого шифровального алфавита фразу МАМА МЫЛА РАМУ. Разобъем полученный открытый текст на группы по шесть символов и

выпишем в столбец сответствующую каждому символу пару чисел (координат) из таблицы:

М	**А**	**М**	**А**	**М**	**Ы**
1	3	1	3	1	3
5	4	5	4	5	2

Л	**А**	**Р**	**А**	**М**	**У**
1	3	1	3	1	1
2	4	6	4	5	1

Теперь прочитаем пары чисел из каждой шестерки уже построчно. В результате такого преобразования получим для первой шестерки — КККЧЧХ, а для второй — ККУБ+Ф.

При таком шифровании координата строки и координата столбца каждой буквы оказываются разъединенными, что характерно именно для раздробляющего шифра.

Вопросы для самопроверки:

1. Объясните алгоритм метода дробления.

2. Зашифруйте сообщение У ЛУКОМОРЬЯ ДУБ ЗЕЛЕНЫЙ методом перестановки столбцов с ключевой фразой МАМА МЫЛА РАМУ.

Литература

1) Берлекэмп Э. Алгебраическая теория кодирования. М.: Мир, 1971. 478 с.

2) Вольфовиц Д. Теоремы кодирования теории информации. М.: Мир, 1967. 248 с.

3) Информатика: Учебник/ под ред. Н.В.Макаровой. М.: Финансы и статистика, 1997. 768 с.

4) Касами Т. и др. Теория кодирования. М.: Мир, 1978. 576с.

5) Кодирование информации: методические указания / сост.: В. Д. Горбоконенко, В. Е. Шикина. – Ульяновск: УлГТУ, 2006. – 56 с.

6) Кузьмин И.В., Кедрус В.А. Основы теории информации и кодирования. Киев: ВИЩА ШКОЛА, 1986. 240 с.

7) Мазур М. Качественная теория информации. М.: Мир, 1974. 239 с.

8) Марков А.А. Введение в теорию кодирования. М.: Наука, 1982. 364 с.

9) Могилев А.В., Пак Н.И., Хеннер Е.К. Информатика: Учеб. пособие для студ. пед. вузов. М.: ИЦ "Академия", 1999. 816 с.

10) Нечаев В.И. Элементы криптографии (Основы теории защиты информации): учебное пособие для университетов и педвузов. /Под.ред. В.А. Садовничего. - М.: Высш. шк., 1999. 256 с.

11) Стариченко Б.Е. Теоретические основы информатики. М.: Горячая линия — Телеком, 2003. 312 с.

12) Стратонович Р.Л. Теория информации. М.: Советское радио, 1975. 424 с.

13) Файнстейн А. Основы теории информации. М.: ИЛ, 1960. 233 с.

14) Чисар И., Кернер Я. Теория информации: теоремы кодирования для дискретных систем без памяти. М.: Мир, 1985. 400 с.

15) Шеннон К. Математическая теория связи. В кн.: Работы по теории информации и кибернетике. М.: ИЛ, 1963. С. 243-332.

Интернет-источники

1) Теория кодирования [Электронный ресурс] // GOUSPRO — студенческий портал!:сайт. - URL: http://gouspro.ru/?page_id=22 (дата обращения 16.01.2012)

2) Леонов Д. Файлы, которым мы доверяем /Свет в Интернет: интернет-еженедельник, - №33(145)/04.11.03. URL: http://www.lightnet.obninsk.ru/Review/Security/145.shtml (дата обращения 13.12.2011)

3) Бекман И.Н. Компьютеры в информатике: курс лекций [Электронный ресурс] // Лекции. - URL: http://profbeckman.narod.ru/EVM.htm (дата обращения 15.12.2011)

Приложение 1. *Оригинальный код Бодо.*

(«o» — наличие сигнала, «.» - отсутствие сигнала)

Управляющие символы								
o. ...	пробел, перейти к таблице букв							
.o ...	пробел, перейти к таблице цифр							
oo ...	удалить последний знак							
таблица букв					*таблица цифр*			
.. o..	A	oo o..	K		.. o..	1	o. o..	.
.. oo.	É	oo oo.	L		.. .o.	2	o. .o.	9/
.. .o.	E	oo .o.	M		.. ..o	3	o. ..o	7/
.. .oo	I	oo .oo	N		.. o.o	4	o. o.o	2/
.. ooo	O	oo ooo	P		.. ooo	5	o. ooo	'
.. o.o	U	oo o.o	Q		.. oo.	1/	o. oo.	:
.. ..o	Y	oo ..o	R		.. .oo	3/	o. .oo	?
.o ..o	B	o. ..o	S		.o o..	6	oo o..	(
.o o.o	C	o. o.o	T		.o .o.	7	oo .o.	)
.o ooo	D	o. ooo	V		.o ..o	8	oo ..o	-
.o .oo	F	o. .oo	W		.o o.o	9	oo o.o	/
.o .o.	G	o. .o.	X		.o ooo	0	oo ooo	+
.o oo.	H	o. oo.	Z		.o oo.	4/	oo oo.	=
.o o..	J	o. o..	—		.o .oo	5/	oo .oo	£

Приложение 2. Таблица частотности букв русского языка.

Буква	Частота	Буква	Частота
пробел	0,175	о	0,090
и	0,062	т	0,053
р	0,040	в	0,038
м	0,026	д	0,025
я	0,018	ы	0,016
б	0,014	г	0,013
х	0,009	ж	0,007
ц	0,004	щ	0,003
Буква	**Частота**	**Буква**	**Частота**
е, ё	0,072	а	0,062
н	0,053	с	0,045
л	0,035	к	0,028
п	0,023	у	0,021
з	0,016	ъ, ь	0,014
ч	0,012	й	0,010
ю	0,006	ш	0,006
э	0,003	ф	0,002