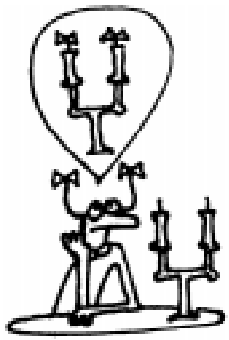
Столяр Сергей Ефимович

ЗАНЯТИЕ 9. КОДЫ И СЖАТИЕ ДАННЫХ

Наука базируется на опыте. На основе опыта и его анализа принимаются определенные соглашения, на которых строятся все дальнейшие рассуждения. Так, на основе опыта, мы безусловно считаем, что существование информации без ее материального носителя невозможно. Ибо, предположив иное, мы оказываемся в положении Алисы из Страны Чудес [1]:



И она попыталась представить себе, как выглядит пламя свечи после того, как свеча потухнет. Насколько ей помнилось, такого она никогда не видела.

Для обмена информацией (еще говорят: «для передачи сообщения») между двумя носителями (назовем их *источником* и *приемником*) нужен так называемый *канал связи*.

Обсуждение физических принципов работы каналов связи относится к курсу физики. Нам же для дальнейших рассуждений достаточно принять лишь одно утверждение: передача информации с помощью «промежуточного» *материального* носителя возможна только, благодаря изменениям во времени его физических характеристик. Поэтому вводятся понятия: *сигнал*, отражающее этот процесс, и *параметр сигнала*, обозначающее собственно изменяющуюся (и несущую сообщение) характеристику.

Будем рассматривать работу канала как некоторое преобразование (отображение) **F** исходного сообщения (пакета данных, набора данных) $\mathbf{A_{in}}$ в выходной пакет $\mathbf{B_{out}}$

$$\mathbf{F : A_{in} \to B_{out}}$$

или, в другой форме записи,

$$\mathbf{B_{out} = F(A_{in})}.$$

Мы пока говорим не о цели такого преобразования, но лишь о возможности его осуществления. Скажем, если вы, сочтя мои рассуждения малоинтересными, станете аккуратно замазывать каждое очередное слово черной пастой, ваши действия вполне можно считать таким F-преобразованием...

Впрочем, надеюсь, этого не происходит, поскольку дальше нам может понадобиться обратное преобразование $\mathbf{F^{-1}}$, и у вас, в этом случае, возникнут проблемы.

Говорят, что преобразование **F** является взаимно однозначным, если существует обратное преобразование

$$\mathbf{F^{-1} : B_{out} \to A_{in}},$$

восстанавливающее исходное сообщение *без изменений*.

За примерами можно обратиться к школьному курсу математики, и вы обнаружите в нем как функции, осуществляющие обратимые преобразования, так и необратимые функции.

В курсе информатики, когда речь заходит о теории баз данных, вводится понятие *связь*, во многом «похожее» на отображение. Типы связей подразделяют на 1:1 («один к одному»), 1:*M* («один ко многим»), *M*:1 («много к одному») и *M*:*N* («много ко многим»).

В такой терминологии «восстановление без изменений» возможно только для отображения 1:1. Но и другие отображения (связи) вполне актуальны. Проиллюстрирую это, скажем, для связи 1:*M*, следующим примером.

Шекспир начинает монолог принца Гамлета словами:

To be, or not to be: that is' the question:
Whether 'tis nobler in the mind to suffer
<...>

Из множества переводов, которые можно найти в [2], я выбрал варианты, наиболее близкие к этому тексту.

Быть или не быть – таков вопрос;
Что благородней: духом покоряться
<...>

(М. Лозинский)

Быть иль не быть – вот в чем вопрос.
Что благороднее: сносить удары
<...>

(П. Гнедич)

Быть иль не быть? Вот в чем вопрос.
Что выше:
Сносить в душе с терпением удары
<...>

(К. Р.)

Даже из «похожих на оригинал» переводов очевидно, что «восстановить» английский текст в обратном переводе было бы по силам только самому Шекспиру. Это ни в коем случае не значит, что мы можем предъявлять претензии переводчикам: таковы законы поэтического жанра. Но, с точки зрения «передачи сообщения по каналу связи», наличие искажений и потерь несомненно.

Задача 1.

Уровень 1.

1. Сентиментальные и суеверные туристы бросают по монете в главный городской фонтан, рассчитывая, благодаря этому, когда-нибудь вновь посетить замечательный город. Во избежание засорения стоков городские службы регулярно организуют «сбор мелочи» в фонтане. Можно ли, рассматривая процесс бросания-сбора монет как «передачу сообщения», считать соответствующее преобразование **F** взаимно однозначным?

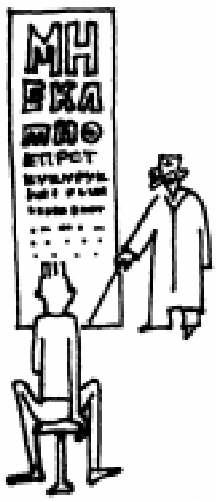
2. Юный любитель природы Вася, отмечая свое посещение парка, всякий раз вырезает на стволе дерева надпись «Здесь был Вася!». Что можно сказать относительно преобразования **F** в этом примере? (Относительно Васи решение пусть принимают компетентные органы.)

3. Та же ситуация с Петей, но он, в отличие от Василия, добавляет к надписи еще и дату посещения. Вопрос прежний.

Обоснуйте свои ответы.

Так как носителем сигналов является некий физический процесс, передача пакета информации *без искажений* далеко не всегда возможна в принципе (да и не всегда необходима! – отмечу в скобках). Так, любое физическое измерение параметров объекта с помощью приборов всегда связано с ограниченными возможностями самого измерительного прибора.

Нужен другой пример? – Пожалуйста. Некоторые из моих учеников склонны «совмещать с полезным приятное»: нет ли и у вас на голове в данный момент наушников любимого плейера? – Есть!? Прекрасно. Добавьте громкость... Еще чуть-чуть... Еще... Продолжайте читать этот текст. Через некоторое время, рискну предположить, ваш «канал связи» начнет давать сбои. Если *помехи* (это тоже характеристика канала связи) покажутся вам все еще малозначительными, попробуйте для продолжения эксперимента отодвинуть текст как можно дальше от глаз... И при этом продолжайте читать...



Наше (и ваше – в данный момент) восприятие потока звуковой и зрительной информации связано с получением так называемых *аналоговых сигналов*. Подробное рассмотрение аналоговых сигналов здесь в нашу задачу не входит, это может быть темой отдельного развернутого обсужде-

ния. Но, учитывая наш интерес к сжатию данных любых типов, отмечу эту возможность в отношении и аналоговых сигналов. Однако, как правило, для них преобразования **F** (сжатие, компрессия) и $\mathbf{F^{-1}}$ (восстановление, декомпрессия) не являются взаимно однозначными, иначе говоря, почти неизбежны потери части информации. С подобными механизмами мы встречаемся, когда обращаемся к аналоговой или цифровой аудио и видеозаписи. Вообще говоря, использованию любых современных мультимедийных средств сопутствуют определенные потери качества звука и «картинки» в сравнении с оригинальным звучанием и изображением.

Итак, далее мы намеренно ограничиваем себя рассмотрением *дискретных сигналов*. Это понятие определяет сигналы, параметр которых может принимать лишь *конечное число* значений. Соответственно, возможность передачи сообщения дискретными сигналами приводит к понятию *дискретное сообщение*.

Попробуем установить, насколько «критичны» потери для дискретных сообщений, в том числе, при преобразовании типа сжатия.

Исторический прогресс (естественную эволюцию я сейчас не рассматриваю) в значительной степени определяется возможностью анализировать «неудачные решения» (проще говоря, «учиться на ошибках»). Такой анализ осуществим, благодаря сохранению информации о предшествующих «решениях». Располагая предоставленными Природой физическими каналами связи, человеческий ум разработал много различных способов их использования.

По-видимому, наиболее популярным является «зрительный» канал. Так, вышеупомянутый Василий, желая зафиксировать для потомков факт своего пребывания в парке, рассчитывает именно на этот канал связи. Будучи образованным человеком, он использует, во-первых, *язык* как систему обозначений (систему соглашений!), во-вторых, так называемую *письменную речь*.

Здесь я бы не хотел даже пытаться *формально* определить понятие *язык*. От этой попытки меня удерживает как нежелание уходить в сторону от основной темы, так и возможность сослаться на «описательное» определение, предлагаемое нам философом Павлом Флоренским [3]:

> *Итак, что же разумеется <...> под языком? Понятие о языке далеко не покрывается понятием о членораздельной речи, ибо сюда входит еще речь письменная, язык мимики, язык жестов, язык рисунка, язык красок и музыкальных тонов, язык природы, язык фактов и т. п. Но наиболее зафиксированной формой языка является язык письменный и язык рисунка.*

В письменной речи мы используем последовательный метод записи *знаков письма* («научное» название таких знаков – *графемы*), размещая их друг за другом. Полный (не бесконечный!) набор используемых знаков составляет *алфавит* языка, при этом разные языки отличаются, в том числе, *мощностью* (количеством элементов) своих алфавитов. Сравните: в современном русском языке алфавит содержит 33 буквы, в английском – 26, в греческом – 24.

Впрочем, и *устную речь* можно расчленить на элементарные составные части – *фонемы*.

> *Учитель. Однако не будем задерживаться на теории < ...>. Мы еще вернемся к этому позже... а может, и не вернемся... Как знать...*
> *Ученица (восторженно). О да, мсье.*
> *Учитель. Итак, знайте и помните до самого своего смертного часа <...> еще одно основное положение: любой язык есть в конечном счете не что иное, как речь, и, следовательно, состоит из звуков, или...*
> *Ученица. Или фонем...* [4]

Рассматривая фонемы в качестве знаков или элементов соответствующего алфавита, несложно договориться относи-

тельно способа их графического отображения, то есть установить соответствующее F-преобразование. В результате мы получаем специальный «письменный язык для устной речи». Цивилизованный мир уже прошел этот путь и установил определенный «стандарт» для указанного отображения **F**.

Еще в 1886 г. в Париже была создана Международная фонетическая ассоциация. Ее целью стало введение так называемой *фонетической нотации* (иначе говорят: *фонетической транскрипции*) при изучении языков. В результате родился Международный фонетический алфавит IPA (International Phonetic Alphabet). Любопытно, что последние изменения, внесенные в него, сделаны сравнительно недавно – в 1989 году, и сейчас этот алфавит включает более сотни элементов [5].

Разумеется, при такой мощности алфавита поместить здесь весь IPA невозможно, да и смысла в этом нет. Но несколько символов в качестве примера приведу:

æ ø ß œ ^ ð è é e q ʟ a ¶

Не правда ли, вы уже не раз с ними сталкивались, когда пользовались словарями, например, [6]?

Заметим, что мощность фонетического алфавита языка не совпадает с мощностью соответствующего алфавита письменного. Я очень удивлюсь, если вы подберете фонетическую транскрипцию к букве «Ъ» и сумеете ее озвучить.

И этот пример отнюдь не экзотический. Так, произношение звука, соответствующего «письменному Р», в русском и английском языке заметно различается, что вызывает немалые проблемы при изучении неродного языка. Почему написанию «EAU» соответствует лишь один «фонетический символ», знают только французы. Нас поражает количество «шипящих» в польской речи. Вы вполне можете продолжить перечень...

Легко представить себе проблемы, с которыми сталкиваются переводчик и актер, дублирующий фильм, когда пытаются втиснуть русскоязычный текст в уста киноперсонажа, соблюдая при этом визуальную и звуковую артикуляцию. Как там обстоит дело с пресловутыми «потерями»?

Задача 2.

Задан входной текст и символьно-фонетическая таблица из *N* строк. В каждой строке ее записан символ или группа символов и, через пробел, фонетический эквивалент. Предполагается, что если группа символов, для которых в таблице есть фонетический эквивалент, входит в более длинную группу символов, также имеющую свой фонетический эквивалент, то приоритет отдается более длинной группе. Таблица упорядочена в лексикографическом (то есть в словарном) порядке.

Уровень 1. Разработайте алгоритм, следуя которому можно перевести исходный текст в фонетический вид.

Уровень 2. Напишите программу **Sym2Phon**, переводящую исходный текст в фонетический вид.

Формат ввода:

Текстовый файл, в первой строке которого задано значение *N*; в следующих *N* строках размещена таблица; в остальных – текст для перевода.

Формат вывода:

Фонетический вид.

Пример.

Ввод:

```
6
A æ
AL ol
ALL ol
C k
E i
OR o
CALL ME AT 10 OR 11 PM
```

Вывод:
kol Mi æT 10 o 11 PM

Вообще говоря, невозможность построить взаимно однозначное отображение для алфавита «письменного» и алфавита «устного» достаточно очевидна. Об этом можно судить, учитывая мощности этих алфавитов. Потому попытка зафиксировать в письменном виде речь устную, состоящую из более мощного множества фонем, чем «обычный» алфавит языка, и вызывает появление специальных обозначений для звуков.

Но при этом фонетическая транскрипция не настолько распространена, чтобы можно было пользоваться ею в любом случае. Скажем, стандартные раскладки клавиатуры и таблицы шрифтов используемого мною текстового редактора не позволяют мне привести здесь некоторые из фонетических символов.

Имеют ли рассуждения о «посимвольном» соответствии алфавитов какой-нибудь практический смысл? Судите сами! Например, принятый в международном телеграфном обмене алфавит не содержит символов кириллицы, и сообщение, которое мы не хотим переводить на английский, принято «писать» латинскими буквами. Язык, на который мы переводим таким образом русский свой текст, называют *волапюк*. Этот язык популярен и при электронной переписке, когда используется код КОИ-7.

Естественно, несовпадение мощностей исходного (русского) и выходного (латинского) алфавитов заставляет кодировать некоторые символы кириллицы комбинациями латинских букв. На этот счет есть определенные соглашения, но именно «соглашения», а не жесткие правила. Скажем, символ «Я», встречающийся в фамилии автора, в его паспорте переведен как «IA», в соответствии с нормой французского языка. Чиновник же американского консульства, выдавая визу, руководствуется другой языковой нормой и фиксирует вариант «YA».

К счастью, хоть буквы «Щ» в моей фамилии нет. Один мой знакомый, которому «повезло меньше», ставя свою подпись в электронной переписке, набирает взамен такое сочетание символов латиницы: «SCHTSCH» (!?)

Пожалуй, здесь уместно ввести одно необходимое нам далее понятие. Используя для объемов входного и выходного сообщений, соответственно, обозначения $\mathbf{L_{in}}$ и $\mathbf{L_{out}}$ (обычно это – длины файлов), будем называть коэффициентом сжатия данных величину

$$\mathbf{K = L_{out} / L_{in}}.$$

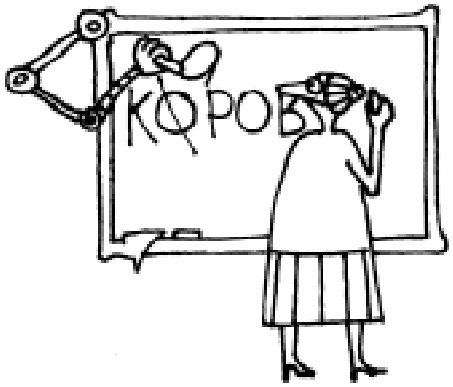
Отметим, что встречаются и альтернативные варианты определения:

$\mathbf{K = (L_{in} - L_{out}) / L_{in}}$ и $\mathbf{K = L_{out} / L_{in}}$, но мы будем пользоваться первым вариантом. Кстати, коэффициент сжатия часто представляется в процентах.

Возвращаясь к примеру с буквой «Щ», можно заключить, что если бы мой знакомый ограничивался в передаваемом тексте только подписью, то коэффициент «растяжения», если принять за длину число символов, составлял бы 7, или 700%! Естественно, реальное сжатие имеет место лишь при $\mathbf{K} < 1$, и к этому мы еще вернемся.

Хочу здесь привести один пример попытки «упорядочения» отображения, связывающего речь письменную и устную. Сейчас его легко принять как анекдотический, но в свое время ситуация выглядела совсем иначе. Я имею в виду результаты деятельности академической комиссии, которая в начале 1960-х годов готовила реформу русского языка с целью «упростить письмо». Развернутый проект предлагавшейся тогда реформы вполне

серьезно обсуждался не только в научных кругах, но и был вынесен на широкое обсуждение и опубликован в прессе. Основной принцип изменений – по счастью, не внедренных – состоял в формуле «как слышится – так и пишется». Как образец историко-языкового казуса я храню газету с полным изложением всего проекта. В нем, например, предлагается такая языковая письменная норма: «кАрова». Или еще более революционное нововведение: «цыган, сидящих на цыпочках» и «наставляющих непослушных цыплят» планировалось вывести из состава знакомых нам исключений, переведя в их общество все остальные слова, включающие сочетание «ЦИ». Вот такой «цЫрк» получался...



Уважаемые коллеги-учителя! Призываю вас: снижайте оценки юным программистам, когда обнаруживаете допущенные ошибки в «русскоязычных» фрагментах программ. Язык программирования – всего лишь подмножество Языка.

Возможно, у реформаторов были вполне благие намерения, но куда вел избранный ими путь, представить несложно. А ведь еще Владимир Даль предупреждал [7]:

Надо, кажется, сохранять такое правописание, которое бы всегда напоминало о роде и племени слова, иначе это будет звук без смысла.

В словаре Даля для каждой буквы алфавита в соответствующей словарной статье мы встречаем «полное название»: *Аз*, *Буки*, *Веди*, *Глагол*, *Добро*, *Есть* и т.д. И в этом смысле алфавит кириллицы не исключение. Так, знакомые нам названия большинства греческих букв никакого собственного смысла не имеют, но зато они происходят от финикийских предшественников, соответствовавших названиям предметов.

Читаешь Даля – и сразу становится очевидным, что слово «АзБука» в русском языке *информативней* современного американского аналога «ABC».

Лаконичность английского языка хорошо известна. Но даже ее не хватает американцам. Характерной иллюстрацией их стремления к сжатию данных служит аббревиатура AuH_2O, которую использовали в президентской рекламной кампании 1964 года сторонники сенатора Барри Голдуотера (GoldWater = золотая вода) [5].

Задача 3.

Предположим, что реформа русского языка осуществилась, и мы пишем «как слышится».

Задан входной текст и таблица из *N* строк. Текст состоит из слов. Под словом здесь понимается группа символов, «обрамленных» с обеих сторон пробелами, либо пробелами слева и синтаксическим символом справа. При этом в тексте встречаются только 6 синтаксических символов из набора {. , : ; ! ?}. Кроме того, в словах, состоящих более чем из одного слога, проставлено ударение, для чего используется служебный символ «подчеркивание» перед ударным слогом, например: «ко_рова».

Таблица в каждой строке содержит либо пару гласных – «дореформенную» из безударного слога в «длинном слове» и – через пробел – «послереформенную», либо правило перекодировки, определяющее «дореформенное» сочетание символов и – через пробел – соответствующее новое сочетание.

Таблица упорядочена в лексикографическом порядке. Предполагается, что возможная не единственность варианта перевода, связанная с содержанием текста

и таблицы, разрешается любым способом.

Уровень 1. Разработайте алгоритм, следуя которому можно перевести исходный текст в «послереформенный» вид.

Уровень 2. Напишите программу **WriteAsSpeak**, переводящую исходный текст в «послереформенный» вид.

Формат ввода:

Текстовый файл, в первой строке которого задано значение *N*; в следующих *N* строках дана таблица; в остальных – текст для перевода.

Формат вывода:

«Послереформенный вид»

Пример.

Ввод:

```
3
О а
ЦЫ ци
ЬО ьё
КТО ТАМ?
ЭТО Я, ПОЧТА_ЛЬОН _ПЕЧКИН.
```

Вывод:

```
КТО ТАМ?
ЭТО Я, ПаЧТАЛьёН ПЕЧКИН.
```

Разумеется, фонетический алфавит не является единственным образцом «аудио» азбук. Гораздо более распространенным алфавитом из этого семейства является нотная грамота. Вновь сошлюсь на авторитетное мнение специалиста – Людвига ван Бетховена [5]:

Я бы скорее написал десять тысяч нот, чем одну страницу буквами.

Зададимся естественным вопросом: является ли отображение музыкального произведения в нотную запись взаимно однозначным? Даже мое дилетантское представление о музыкальном творчестве заставляет предположить обратное. В противном случае концертная интерпретация одной и той же партитуры была бы совершенно одинаковой у разных дирижеров.

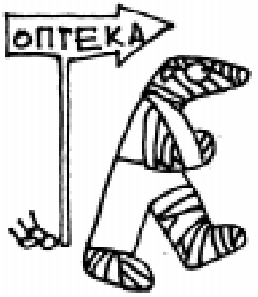
Можно предположить, что композитор, записывая свое сочинение в алфавите нотной грамоты, заведомо соглашается на некоторые будущие «потери» (более точно определить эту ситуацию я не берусь). Очевидно, что введенный нами коэффициент **K** уже не равен 1, что, впрочем, у меломанов вряд ли вызывает интерес.

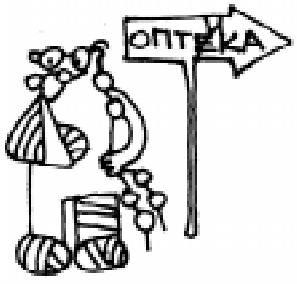
Но вернемся к более привычному для нас алфавиту и попытаемся выяснить наше отношение к потере символов в «письменном сообщении». В обыденной ситуации мы вынужденно миримся с нередкими орфографическими и прочими ошибками, особенно если они не влекут потери смысловые. Однако найти контрпримеры труда не составляет.

Известная всем преподавателям математики, и не только им, книга Дьердя Пойи (G.Polya) озаглавлена «Как решать задачу» (в оригинале – «How to Solve It») [8] – безо всякого синтаксического символа в конце. Допускаем маленькое интонационное искажение – и предложение превращается в коварный вопрос, адресованный ученикам. Соответствие изменению интонации достигается добавлением лишь одного символа «?».

Разумеется, отказываться от синтаксических символов рискованно. Что же касается данного примера, то некорректное сжатие-искажение «Как решать задачу? => Как решать задачу» просто не будет обратимым. При этом коэффициент **K** составляет 17/18, и «незначительная» потеря в 1/18 при сжатии заметно сказывается на смысле.

Еще пример. В книге Льва Успенского «Слово о словах» приводится драматический пример из военных времен. На указателе направления было написано «оптека», и правильный вариант «аптека» или «оптика» восстановить оказалось невозможно (при этом, **K** = 1). А это уже ситуация, когда потеря (искажение)





данных (сигнала) приводит к невосполнимой потере смысла (информации).

Итак, потерь при преобразовании, в том числе при сжатии дискретных сообщений, хотелось бы избегать. Так что пора нам ближе познакомиться с возможностями отображения-сжатия «без потерь».

Откуда, вообще говоря, возникает сама мысль сжимать данные?

Полагаю, причина таится в присущей человеку лени. О роли анализа «неудачных решений» в историческом прогрессе я уже упомянул. А разумная лень заставляет нас искать наиболее экономное решение, требующее минимальных, в широком смысле, усилий.

Какому-то чудаку надоело волочить за собой (или на себе) тяжести, и он сконструировал колесо. Другому чудаку показалось утомительным пересказывать бесконечные предания своего племени бездельнику-наследнику – и он изобрел письменность. Нерадивому ученику лень стало зубрить уроки – и он делает шпаргалку к экзамену.

Скорей всего, *Историк* оценил бы подобный взгляд на исторический прогресс как ненаучный, но ведь мы не покажем ему эти записи ☺.

Так вот, стремление к сжатию данных, то есть к экономии и места и времени, – в том же ряду.

ИСКЛЮЧЕНИЕ ИЗБЫТОЧНЫХ ЭЛЕМЕНТОВ ДАННЫХ

Такая возможность возникает в связи с определенными ограничениями, присущими данным, причем ограничения могут диктоваться либо содержанием всего набора данных, либо контекстом, либо и тем и другим. Рассмотрим подробнее несколько таких механизмов.

Применение ***первого механизма*** возможно в тех нередких случаях, когда обрабатываемый набор данных *заведомо* содержит только английский текст, включая символы пунктуации. Естественно, все соответствующие коды ASCII таблицы располагаются в ее первой половине. Это означает, что их номера не превосходят 127. Следовательно, старший разряд каждого байта содержит битовый нуль.

Почему бы не воспользоваться всеми «бесполезными» разрядами, разместив в них часть остальных данных? Напрашивается такая идея: каждую серию, состоящую из 8 последовательных байт, можно «упаковать» в 7 байт, разместив 7 бит последнего байта серии по одному в старшие позиции предыдущих байт. Если последняя серия короче 8 байт, то проще всего дополнить ее до «стандартной» длины нулевыми байтами. Экономия при таком механизме сжатия составит, очевидно, 1/8. Иначе говоря, коэффициент сжатия **K** = 87.5%.

Описанное F-преобразование сжатия (компрессии) естественно назвать алгоритмом *7-битного кодирования* (*сжатия*).

В литературе, посвященной алгоритмам и программам сжатия данных, для обозначения программы, реализующей прямое преобразование **F**, нередко используется название *кодер*; соответственно, для обратного преобразования – *декодер*; для программы, предназначенной осуществлять оба преобразования – *кодек*.

Отметим как положительный момент, что алгоритм *однопроходный*, то есть для его выполнения достаточен однократный просмотр входного файла. Восстановление исходного набора данных (декомпрессия) реализуется с помощью вполне очевидного механизма. И здесь также потребуется лишь один проход по файлу.

Задача 4.

Уровень 1. Примените алгоритм 7-битного кодирования к первой строке приведенного выше фрагмента монолога принца Гамлета (естественно, к тексту Шекспира). Как будет выглядеть выходной ASCII текст?

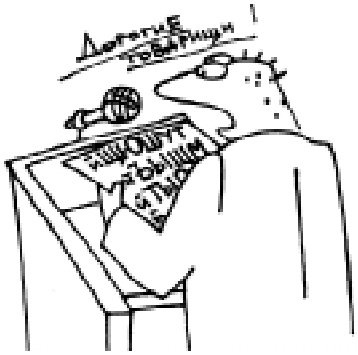
Уровень 2. Напишите программу-кодер **Bit7Code**, осуществляющую 7-битное кодирование.

Формат ввода:

Текстовый файл, заведомо не содержащий символы с кодами более 127.

Формат вывода:

Файл закодированного текста.



Пример.

Ввод:

CALL ME AT 10 OR 11 PM

Вывод:

CA╨L ME┴Ta10a 11 PM

Задача 5.

Уровень 1. На вход декодирующего устройства подается ASCII текст, полученный при работе алгоритма 7-битного кодирования. Опишите алгоритм работы декодера. Продемонстрируйте на примере из предыдущей задачи работоспособность вашего алгоритма.

Уровень 2. Напишите программу-декодер **Bit7Decode**, осуществляющую декодирование файла, сформированного при работе кодера.

Формат ввода:

Файл закодированного текста.

Формат вывода:

Файл декодированного текста.

Пример.

Ввод:

CA╨L ME┴Ta10a 11 PM

Вывод:

CALL ME AT 10 OR 11 PM

Второй механизм работоспособен, когда возможность исключения избыточных элементов из набора данных продиктована контекстом.

Цифровая запись 3.14 вполне заменяет нам длинный аналог *три целых четырнадцать сотых*.

Также хорошо известны общеупотребительные сокращения мер веса, расстояния и прочие, и появление на шоссе указателя 3.14 км понимается нами однозначно. «Контекстом» здесь является само шоссе и многочисленные дорожные знаки. Надпись на ценнике товара 3.14 мы будем трактовать, скорее всего, как «*3 рубля 14 копеек*» или, что также вполне возможно, как «*3 доллара 14 центов*». При других же обстоятельствах мы вполне резонно рассудим, что запись 3.14 относится к числу p (3.1415926...).

Иначе говоря, замена «полного обозначения» на «сокращенное» далеко не всегда соответствует отображению типа 1:1. Стало быть, возможность восстановления данных однозначным образом вне контекста не гарантируется.

В 1818 году шведский химик Йенс Якоб Берцелиус ввел в обиход науки систему обозначений для химических элементов, действующую и поныне [5]. Поскольку с тех пор были открыты новые элементы, для них добавлялись и новые обозначения. Вот несколько примеров:

S – сера, Ru – рутений,

Ir – иридий, F – фтор,

Fr – франций.

Обычно выбор сокращений диктовался либо написанием полных названий, либо более древней традицией, алхимической.

В 1926 году, в соответствии с «Международным соглашением по регулирова-

нию дорожного движения», ввели знаки для обозначения принадлежности автомобиля определенной стране (IVR – International Vehicle Registration). Позднее сфера применений этих знаков расширилась, и сейчас мы встречаем их и на товарах, и в рекламе, и на почтовых отправлениях. В системе IVR те же обозначения расшифровываются уже совсем иначе [5]:

S – Швеция, RU – Бурунди,

IR – Иран, F – Франция,

FR – Фарерские острова.

Если механизм сокращения названий для «старых» государств достаточно очевиден, то вот с Бурунди, видимо, возникли трудности, поскольку Бирма успела раньше «захватить» BUR.

Но, главное, вряд ли сотрудник дорожной службы, учитывая «контекст», решит, что владельцем автомобиля с номерами FR-314-15 является либо французхимик, либо математик.

Итак, для общеупотребительных сокращений существование восстанавливающего данные обратного отображения $\mathbf{F^{-1}}$ определяется, как мы видим, контекстом и нашим знакомством с соответствующей системой сокращений.

Между прочим, понятие «общеупотребительность» весьма рискованно трактовать буквально. Так, школьник знаком с сокращенными обозначениями единиц измерения системы СИ. Но знает ли он, как расшифровать само название системы? А как расшифровывается аббревиатура ASCII?

Если же, в целях экономии места, применяется система обозначений, принятая «в узких кругах», то однозначно интерпретировать данные становится невозможно. И никакой контекст тут не поможет.

Тогда на первый план выходит ***третий механизм*** исключения избыточности.

Кто из вас не наблюдал хотя бы однажды за соревнованиями атлетов, различить которых на большом расстоянии можно только по номерам на спортивной форме. В распоряжении судей соревнования имеется специальный протокол с фамилиями и номерами участников, играющий роль *таблицы соответствия*. Телевизионный редактор, формируя кадр, непременно подставит рядом с номером спортсмена сопутствующие сведения из собственной аналогичной таблицы.

Кстати, выше в нашем тексте вы уже встретились с применением таблиц соответствия – в задачах 2 и 3.

В *системах управления базами данных* (*СУБД*) такие таблицы часто называют *кодификаторами*. В компьютерном представлении кодификаторы могут храниться как отдельные файлы. Их единственное назначение состоит в том, чтобы информацию, которая при обработке циркулирует в виде укороченных кодов, можно было привести в «читаемый» вид для вывода на экран или принтер. В так называемых *реляционных базах данных* таблица-кодификатор состоит из строк-*кортежей*, содержащих всего по паре значений. Очевидно, что в этом случае мы вновь имеем пример связи вида 1:1.

В какой степени подобные кодификаторы обеспечивают достижение основной цели – сжатия? Ведь ясно, что они сами являются *избыточными данными*. Значит, их избыточность должна компенсироваться сокращениями в других наборах.

Здесь вновь будет уместен пример с автомобильными номерами. Сотни и сотни тысяч зарегистрированных в Санкт-Петербурге машин несут на своих бамперах обозначение региона 78. На бамперах авто из Ленинградской области мы видим число 47. Для других регионов России выделены свои двузначные коды.

Легко оценить, с точки зрения ведения компьютерных баз данных, рентабельность подобных сокращений в номерных знаках. Существенно и то, что *поле данных* кода региона имеет фиксированный формат (ширину), в отличие от самих названий регионов. Благодаря этому, обеспечиваются более удобные механизмы поиска информации в базе данных. Кстати говоря, экономия места и краски при изготовлении собственно номерных

знаков автомобиля также свидетельствует в пользу сжатия данных.

Задача 6.

Задана таблица соответствия. В общем случае, содержательная информация в строках таблицы имеет разную длину. При непосредственном использовании удобно размещать таблицу в оперативной памяти как двумерный массив. В файл же поместим ее в упакованном виде. Для этого «склеим» все строки в одну последовательность, подставив перед каждой из них соответствующее двузначное число – длину строки до упаковки.

Уровень 1. Разработайте алгоритм, следуя которому можно указанным способом упаковать таблицу-массив в текстовый файл.

Уровень 2. Напишите программу **TablePack**, реализующую этот алгоритм.

Формат ввода:

Текстовый файл с таблицей, разбитой на строки.

Формат вывода:

Упакованная таблица соответствия.

Пример.

Ввод:

RSM_Сан-Марино
RU_Бурунди
RUS_Россия

Вывод:

14RSM_Сан-Марино10RU_Бурунди10
RUS_Россия

Задача 7.

Уровень 1. Разработайте алгоритм обратного преобразования файла, полученного в предыдущей задаче, в таблицу.

Уровень 2. Напишите программу **TableUnPack**, осуществляющую обратное преобразование файла, полученного в предыдущей задаче, в таблицу.

Формат ввода:

Упакованная таблица соответствия.

Формат вывода:

Текстовый файл с таблицей, разбитой на строки.

Пример.

Ввод:

14RSM_Сан-Марино10RU_Бурунди10
RUS_Россия

Вывод:

RSM_Сан-Марино
RU_Бурунди
RUS_Россия

Обсуждая ранее использование кодификаторов, мы выяснили, что их приходится хранить отдельно от исходного и сжатого наборов. И рентабельность обслуживания дополнительного набора проявляется лишь при многочисленных заменах «названия» на «код». Если же количество замен невелико, то можно использовать ***четвертый механизм*** ликвидации избыточных данных, который мы здесь назовем *обратной ссылкой*.

Продемонстрируем ее непосредственно на примере. Воспользуемся словарем [6] и откроем его, скажем, на 512-й (грешен! – люблю степени двойки) странице первого тома. Там можно обнаружить статью **data** (вот он – «рояль в кустах»). Цитирую:

data I n pl

1. pl от datum

2. употр. тж. с гл. в ед. ч. (исходные) данные, факты; информация; this ~ эти данные <...>

Как видите, здесь перемешаны сокращения русских и английских лингвистических терминов, текст набран разными шрифтами. Разумеется, все обозначения приведены в начале тома, то есть без таблицы соответствия здесь не обойтись. Хотя, вообще говоря, они более-менее стандартны для большинства словарей. Но сейчас нам интереснее другое – замена заглавного слова словарной статьи символом ~ (тильда). Вот она – обратная ссылка. Риторический вопрос: насколько вырос бы объем издания (только в 1-м томе – 832 стр., а томов – 3), если исключить подобное сжатие «data».

Несколько более сложный ***пятый механизм*** ликвидации избыточности по идее похож на предыдущий. Он применим в тех случаях, когда данные упорядочены, и можно организовать цепочку ссылок. Например, при построении файла-словаря компьютерной программы-переводчика построение слов происходит из добавляющихся друг за другом частей слова.

Подробнее рассматривать этот механизм я здесь не стану, а интересующийся читатель может заглянуть, например, в книгу Дж. Мартина [9].

Но применение механизма обратной ссылки словарями не ограничивается. Предположим, нам нужно сохранить в числовом файле набор натуральных чисел, каждое из которых помещается в два байта, например:

1020 1018 1013 1013 1013 1008 1010 1020 1036 1050 1035 ...

Замечаем, что разница между соседними элементами каждой пары составляет не более ±128, то есть величину, которая может быть записана как однобайтовая. Оказывается, в этом случае можно хранить не сами очередные «длинные» значения, а лишь их приращения по отношению к предыдущему элементу. Естественно, для первого элемента файла, который далее выступает в роли *базы*, нужно предусмотреть возможность кодирования в «правильном» виде. Например, можно в качестве базы завести фиктивный, нулевой, элемент. Для приведенного примера получаем:

1000 +20 –2 –5 0 0 –5 +2 +10 +16 +14 –15 ...

Таким образом, если не учитывать механизм кодирования 1-го элемента, на остальных мы выигрываем по байту на каждом. Соответственно, коэффициент сжатия составит **K** = 50%.

Этот алгоритм сжатия данных применяется, например, для записи в файл аналоговых сигналов после их так называемой дискретизации.

Типичный пример – запись показаний прибора через равные промежутки времени. Представьте себе работу самописца, вычерчивающего на непрерывно поступающем рулоне бумажной ленты график хода некоего процесса, например, изменения уровня воды в реке Неве.

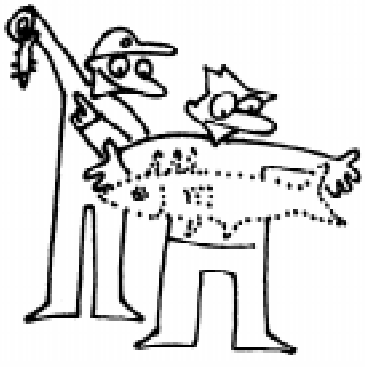
В компьютерной форме хранение такой информации затруднительно. Взамен из ординат графика (в нашем примере – уровня воды) через фиксированный шаг абсциссы (интервал времени) формируется либо таблица (при использовании оперативной памяти – это линейный массив), либо последовательность значений (если ведется запись файла).

При записи данных можно применять описанную выше упаковку, и такой алгоритм, очевидно, является однопроходным. Если далее понадобится восстановить непрерывный график поведения сни-

маемой характеристики процесса, то используется один из вариантов *интерполяции*. Следует, однако, иметь в виду, что применение интерполяции при восстановлении данных связано в общем случае с неизбежными потерями. Иначе говоря, преобразование дискретизации не является взаимно однозначным.

Задача 8.

Нередко при исследовании процесса, который описывается указанным образом, интерес представляют некоторые его обобщенные характеристики.



Среди них: значение наименьшего *min* и наибольшего *max* элемента последовательности, размах последовательности $r = max - min$, количество локальных экстремумов N (при описанном способе упакованного представления данных эта характеристика определяется легко – как количество перемен знака между соседними элементами*).*

Задана целочисленная последовательность, представляющая запись значений некой характеристики процесса с фиксированным шагом по времени. При этом начальный элемент последовательности следует рассматривать как базу, а все последующие элементы – как приращения по сравнению с предыдущим значением.

Уровень 1. Разработайте алгоритм вычисления размаха последовательности r и количества перемен знака N.

Уровень 2. Напишите программу **NumSequence**, вычисляющую значения r, N.

Формат ввода:

Целочисленная последовательность, элементы которой отделены друг от друга символом «пробел» и/или «перевод строки».

Формат вывода:

В первую и вторую строки выходного текстового файла поместить, соответственно, целочисленные значения r и N.

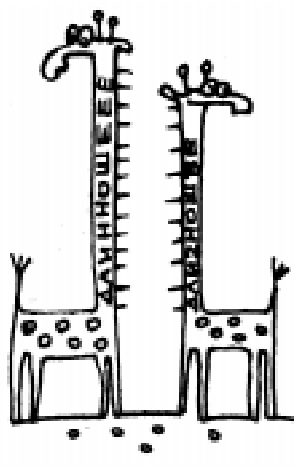
Пример.

Ввод:

0 1 2 0 6

Вывод:

6 2



Наконец, еще один механизм, ликвидирующий избыточность, уже ***шестой*** по счету в нашем рассмотрении, основан на использовании повторов «соседних» данных. Например, в приведенной выше последовательности встречается *серия* из трех повторяющихся друг за другом значений 1013. Можно попытаться использовать это обстоятельство, заменяя каждую серию *парой* величин – коэффициентом повторения (в примере: 3) и собственно значением (1013). Ясно, что придется либо применить такой механизм ко всему набору, либо всякую *пару* сопровождать признаком ее появления, чтобы отличать от обычных элементов. Очевидно, для нашего примера коэффициент сжатия данных в обоих случаях превышает 1, то есть фактического сжатия не происходит.

Тем не менее, идея оказывается плодотворной, если ее применить в отношении серий повторяющихся символов. В литературе соответствующий алгоритм носит название RLE. Эта аббревиатура у разных авторов заменяет либо полное название *Repeated Running Length Encoding*, либо короче – *Run Length Encoding*. К сожалению, привести канонический вариант и первоисточник я не могу. По-русски же этот алгоритм обычно называют *сжатием последовательностей одинаковых символов.*

Применим описанный подход к символьной последовательности, например, такой:

ABBCCCDDDDEEEEE

Очевидно, результатом преобразования будет более короткая последовательность

1A2B3C4D5E

Задача 9.

Уровень 1. Имеется цифровая таблица размера 9x9. Ее главная диагональ, то есть диагональ из верхнего левого угла к нижнему правому, заполнена нулями. В каждой строке элементы слева от диагонального содержат номер строки, а элементы справа – номер столбца. «Склейте» все 9 строк таблицы последовательно в одну текстовую строку и примените затем RLE-преобразование. Как будет выглядеть выходной ASCII текст?

Уровень 2. Напишите программу-кодер **RLECode**, осуществляющую RLE-преобразование входного текстового файла в выходной текстовый файл.

Формат ввода:

Произвольный текстовый файл.

Формат вывода:

Результат RLE-преобразования.

Пример.

Ввод:

ABBCCCDDDDEEEEE

Вывод:

1A2B3C4D5E

Задача 10.

Уровень 2. Напишите программу **RLEDecode**, осуществляющую восстановление упакованного предыдущей программой файла.

Формат ввода:

RLE-упакованный файл.

Формат вывода:

Исходный текстовый файл.

Пример.

Ввод:

1A2B3C4D5E

Вывод:

ABBCCCDDDDEEEEE

Для того чтобы избежать перемешивания в последовательности числовых значений коэффициентов и собственно символов, целесообразно кодировать коэффициенты как символы ASCII с соответствующими кодами. Так, вместо *серии*, включающей 65 рядом стоящих символов S, получаем *пару* AS, поскольку под порядковым номером 65 в таблице ASCII значится символ A.

Коэффициент сжатия здесь составит **K** = 10/15 = 66.7%.

Сразу возникает вопрос, как же закодировать серию из 256 и более одинаковых символов, ведь резервов ASCII таблицы уже не хватает. Проблема решается произвольным разбиением длинной серии на несколько коротких так, чтобы каждая включала менее 256 символов. Скажем, серию из 320 символов S можно представить как ZSZSZS2S (код Z – 90_{10}, код 2 – 50_{10}), а последовательностью SSSSSSSS, при желании, можно заменить 332-символьную серию букв S (код S – 83_{10}).

Задача 11.

Уровень 1. Вы получили сообщение: HELLO! – представляющее результат упаковки RLE-преобразованием ASCII текста. Восстановите исходное содержание.

Достаточно очевиден тот факт, что для обычного текста применение RLE-преобразования приводит не к сжатию, а к противоположному результату. Действительно, сжимаются лишь серии длиной не менее 3 символов, а даже 2 одинаковых символа рядом – скорее исключение, чем правило.

Выходит, описанный механизм представляет лишь теоретический интерес? Не совсем так. До недавнего времени широкой популярностью пользовался файловый формат сохранения изображений, разработанный еще в 80-е годы фирмой ZSoft. Если кодировать цвет знакоместа или пикселя экрана одним байтом, то строки изображения будут содержать большое количество *серий* одинаковых символов. Указанный формат изначально был рассчи-

тан на использование видеоадаптеров, обеспечивающих палитру из 64 цветов. Расширение палитры до 256 цветов осуществляется специальным приемом, но это не относится к предмету нашего рассмотрения. При диапазоне значений цвета от 0 до 63 в байте оказываются задействованными лишь 6 младших бит, с номерами от 0 до 5, а биты 6 и 7 заведомо свободны, то есть содержат нули.

Дальнейшее – очевидно, но кодирование PCX-файлов имеет и некоторую специфику. Если длина серии менее 2, то элементы изображения не подвергаются сжатию. В противном случае признаком кодирования для байта-счетчика служат две битовых 1 в старших разрядах, а остальные 6 бит содержат двоичную длину серии, содержание которой определяет следующий байт пары. Поэтому кодировать таким образом можно серии длиной не более 63_{10}.

Идея использовать «лишние» биты позволяет продемонстрировать ***еще один вариант*** применения RLE-сжатия – в отношении 7-битных ASCII наборов. Помните, мы с ними уже встречались, обсуждая алгоритм *7-битного кодирования*.

Теперь, если встречается *серия* длиной не менее трех одинаковых символов, мы заменим ее *парой*, и в первый байт со счетчиком запишем 1 в старший бит. Это то же самое, что добавление 128_{10}. Соответственно, отсутствие такого признака будет означать, что очередной байт переносится в выходной набор без изменений. Естественно, длина *серии* не должна превосходить 127_{10}.

Можно попытаться применить RLE-механизм к двоичным последовательностям. В этом случае, вместо *пары*, заменяющей очередную *серию*, достаточно в выходной набор помещать лишь *коэффициент повторения*, поскольку алфавит {0,1} однозначно определяет чередование содержания *серий*.

Как при этом отличать двоичные последовательности, начинающиеся с 0, от последовательностей, началом которых является 1? Можно просто *договориться*, что первым всегда стоит счетчик для 0.

Например, согласно ASCII таблице, 6-символьная последовательность, составляющая имя принца Датского – Hamlet – представляется шестнадцатеричными кодами

48 61 6D 6C 65 74

или в двоичном виде

0100 1000 0110 0001 0110 1101 0110 1100 0110 0101 0111 0100

При замене двоичного кода на последовательность десятичных счетчиков получаем

112142411212111212322111113112

К сожалению, больших успехов на этом пути не добиться. Проблема состоит в том, что коэффициенты повторения должны сами представляться в двоичном коде. Соответственно, для них нужно выделить *зоны записи* одинаковой длины. Скажем, если *договориться* о предоставлении для каждого счетчика по 2 битовых позиции, то значение 4 из приведенного примера туда уже не поместится. Приходится разбивать эту 4-серию на 3 других, с *пустой* посередине, например, так: 301. В результате последовательность счетчиков удлинится и превратится в такую:

1121301230111212111212322111113112

или вновь в двоичном виде –

0101100111000110110001010110011 0010101100110111010010101011101 0110

Итого, вместо исходных 48 бит, получили 67, то есть коэффициент сжатия **K** = 67/48 > 1, что нас вряд ли может устроить.

Впрочем, попробуйте поэкспериментировать с реальными файлами, меняя размер зоны записи коэффициента повторения.

Задача 12.

Уровень 2. Напишите программу-кодер **RLEBinCode**, осуществляющую RLE-преобразование входного файла в выходной набор. Входной текстовый файл программа рассматривает как двоичный набор, полученный преобразованием каждого символа в его двоичный ASCII код. Кодирование осуществляется в 4-битную

зону записи, то есть в каждый очередной байт выходного файла помещается последовательно по два счетчика; если последний счетчик не имеет «партнера» для упаковки, то он дополняется четверкой битовых нулей.

Формат ввода:

Произвольный текстовый файл.

Формат вывода:

Результат модифицированного RLE-преобразования.

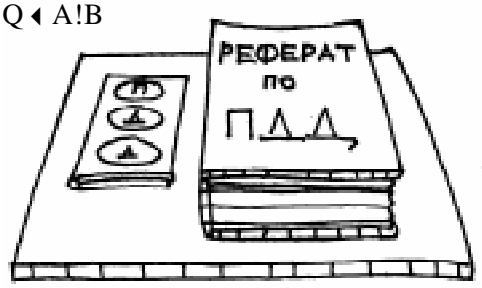
Пример.

Ввод:

ABC

Вывод:

◂Q◂A!B



Задача 13.

Уровень 2. Напишите программу-декодер **RLEBiNDecode**, осуществляющую восстановление данных – из входного упакованного предыдущей программой файла в выходной распакованный файл.

Формат ввода:

Результат модифицированного RLE-преобразования.

Формат вывода:

Произвольный текстовый файл.

Пример.

Ввод:

◂Q◂A!B

Вывод:

ABC

Подводя итог обсуждению RLE-механизмов, остается признать, что особых достижений в общем случае этот метод сжатия не дает. Но его эффективность проявляется на наборах, содержащих достаточно длинные и многочисленные серии повторяющихся элементов. Если удается исходный набор предварительно «подготовить», то рентабельность RLE-сжатия заметно вырастает. В ряде современных программ-архиваторов эта идея находит свое применение.

Но пора обсудить и другие подходы к сжатию данных.

Обратимся к методам, не связанным с непосредственным использованием ASCII таблицы. Очевидно, использование ASCII-кода базируется на двух *соглашениях*. Во-первых, принят двухэлементный – двоичный – алфавит, и, во-вторых, все его символы равноправны, то есть применяется так называемое n-разрядное кодирование ($n = 8$).

Можно предположить, что оба соглашения являются во многом данью традиции. Еще в 1580 году, во времена Шекспира, английский философ Фрэнсис Бэкон (Fransis Bacon) использовал двухэлементный алфавит, предложив *двоичный 5-разрядный код* [10], то есть $n = 5$.

Между прочим, эта его работа была связана с вопросами *криптографии*, ныне активно развивающимся разделом информатики. Код Бэкона предназначался для шифрования английского текста, в котором каждая буква заменялась в соответствии с таблицей 1.

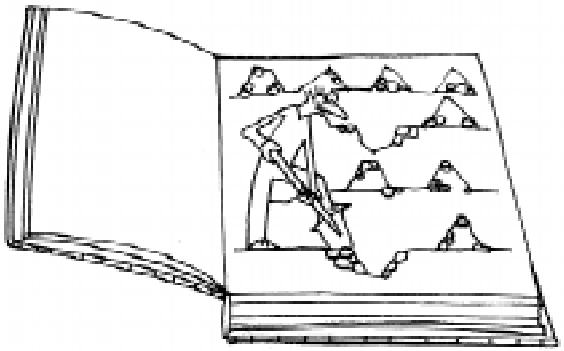
Вообще говоря, обыкновенная *таблица* является наиболее удобной, наглядной, формой представления взаимно однозначного отображения для *дискретных конечных наборов*.

a	AAAAA	g	AABBA	N	ABBAA	T	BAABA
b	AAAAB	h	AABBB	O	ABBAB	V	BAABB
c	AAABA	I	ABAAA	P	ABBBA	W	BABAA
d	AAABB	K	ABAAB	Q	ABBBB	X	BABAB
E	AABAA	L	ABABA	R	BAAAA	Y	BABBA
f	AABAB	m	ABABB	S	BAAAB	Z	BABBB

Таблица 1.

И вы, конечно, сразу заметили, что букв здесь 24, а не 26. Дело в том, что символы *j* и *u* в таблице отсутствуют, поскольку во времена Шекспира они не использовались.

Бэкон, шифруя сообщение, применял 2-символьный алфавит {A,B}. При использовании *n*-разрядного кодирования, чем маломощней алфавит, тем длиннее текст. При шифровании по Бэкону закодированный текст заметно удлиняется, очевидно, в 5 раз.



Но нам ведь известна возможность битового кодирования, и, вместо символьного алфавита {A,B}, мы можем применить равномощный битовый алфавит {0,1}. То есть, используя взаимно однозначное отображение

$$\mathbf{F : \{A,B\}_{sym} \rightarrow \{0,1\}_{bit}},$$

легко построить аналогичный 5-битовый код. С его помощью кодируются $2^5 = 32$ различных символа, что, с точки зрения большинства современных приложений, маловато. Но и 5-разрядные коды находят применение, например, именно таков международный телеграфный код CCIT-2.

Примером 6-разрядного кодирования является рельефный шрифт для слепых француза Луи Брайля. Мощность этого алфавита, очевидно, составляет $2^6 = 64$ символа. На родине изобретателя шрифт Брайля был официально принят спустя год после его смерти (в 1853 году) и ныне используется в книгоиздании [5]. Специфической особенностью системы кодирования Брайля является не последовательное расположение двоичных позиций кодируемого символа, а в виде матрицы 3x2. Вот как, к примеру, записывается имя HAMLET:

```
·▫   ·▫   ··   ·▫   ·▫   ▫·
··   ▫▫   ▫▫   ·▫   ▫·   ··
▫▫   ▫▫   ·▫   ·▫   ▫▫   ·▫
H    A    M    L    E    T
```

Знак · соответствует выдавленной на плотной бумаге точке. Матрицы, то есть символы, расположены построчно, на одинаковом расстоянии.

Если мы сопоставим таким точкам двоичное значение 1, а остальным позициям – битовый 0, то для алфавита Брайля легко построить взаимно однозначное отображение в алфавит $\{0,1\}_{bit}$ с разрядностью представления символов $n = 6$. Например, можно «склеить» построчно элементы матрицы.

Для того же текста получим последовательность:

101100 100000 110010 101010 100100 011110

Промежутки между 6-битовыми кодами здесь оставлены только для наглядности.

Задача 14.

Уровень 2. Написать программу **Code2x3**, которая, получая на вход таблицу соответствия между 64-символьным алфавитом и 6-битными кодами и текст, формирует соответствующий выходной файл в «формате Брайля».

Формат ввода:

Текстовый файл, в котором задана упорядоченная в алфавитном порядке левого столбца таблица соответствия между 64-символьным алфавитом и 6-битными кодами. В каждой из 64 строк записан ASCII-символ и – через пробел – соответствующая комбинация из 6 нулей и единиц. В следующих строках файла содержится произвольный текст, но состоящий только из тех же 64 символов.

Формат вывода:

Каждый символ из входного текста должен быть заменен битовой комбинацией и размещен в матрице 3x2; между соседними матрицами вставлять матрицу из 6 нулей, обозначающую промежуток между символами.

Пример.

Ввод:

```
...
A  100000
...
E  100100
...
HAMLET
```

Вывод (промежутки между знаками здесь сделаны для наглядности):

```
10 00 10 00 11 00 10 00 10 00 01
11 00 00 00 00 00 10 00 01 00 11
00 00 00 00 10 00 10 00 00 00 10
```

Задача 15.

Уровень 2. Напишите программу **UNCode2x3**, которая, получая на вход кодовую таблицу из предыдущего задания и текст в «формате Брайля» (выходной для того же задания), восстанавливает исходный ASCII текст.

Формат ввода:

Текстовый файл, в котором задана упорядоченная в алфавитном порядке левого столбца таблица соответствия между 64-символьным алфавитом и 6-битными кодами. В каждой из 64 строк записан ASCII-символ и – через пробел – соответствующая комбинация из 6 нулей и единиц. Следующие его строки содержат текст в «формате Брайля».

Формат вывода:

Исходный ASCII текст.

Пример.

Ввод (промежутки между знаками в последней строке сделаны для наглядности):

```
...
A  100000
...
E  100100
...
10 00 10 00 11 00 10 00 10 00 01
11 00 00 00 00 00 10 00 01 00 11
00 00 00 00 10 00 10 00 00 00 10
```

Вывод:

```
HAMLET
```

Если 64 символов мало, то, оставаясь в рамках двоичного алфавита и увеличивая разрядность до $n = 7$, мы получаем в распоряжение код ISO-7, принятый Международной организацией по стандартизации (International Standards Organization). Фактически на его основе заполнена первая половина ASCII таблицы. Как выяснилось, и его $2^7 = 128$ символов не хватает, и 256 (при $n = 8$) еще удобнее для записи многообразной информации.

Но и это – не предел. В комитетах ISO рассматривался проект перехода к стандарту 4-байтного, то есть 32-битного, кодирования. Не знаю, впрочем, чем дело закончилось, и закрыт ли вопрос...

Очевидно, что «механическое» увеличение разрядности и принятие соответствующего *соглашения* позволяет расширять диапазон используемого алфавита. Но этот путь вызывает обоснованные возражения. Ясно, что далеко не все используемые символы встречаются одинаково часто. Скажем, в русском языке наиболее «популярной» буквой в начале слова, заметно опережающей все остальные, является П. Легко убедиться в этом без специального анализа, лишь обратившись к четырехтомному словарю Даля, весь третий том которого составлен из таких слов.

А вот противоположный пример, из английского языка. Много ли найдется слов, начинающихся с X? Или удастся ли вам обнаружить слова с начальной буквой Q, непосредственно за которой не следует символ U?

Итак, вновь возвращаясь к общепринятой «пока» системе кодирования на основе ASCII таблицы, мы вполне уверенно можем утверждать, что ее использование, вместо того же 5-битного кода CCIT-2 или какого-нибудь 7-битного алфавита, далеко не всегда оправдано контекстом. Более того, ASCII–алфавит отнюдь не обеспечивает одинаково компактного кодирования в каждом конкретном случае.

Но если использование 8-битного алфавита определяется, как мы установили, лишь принятым соглашением, притом далеко не идеальным, то почему не под-

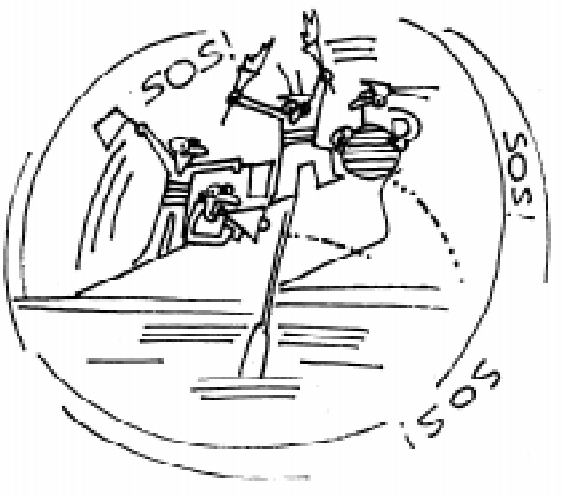
вергнуть сомнению и второе соглашение – относительно равноправия символов при выделении им «битовой квоты»?

Полагаю, вы не станете возражать против небольшого исторического экскурса.

Американский художник-портретист Самюэль Финли Бриз Морзе (S.F.B.Morse, 1791-1852), помимо занятий живописью, любил путешествовать. И вот, занимаясь тем и другим одновременно, он неожиданно, если учитывать к тому же еще и зрелый возраст, в 1832 г. заинтересовался проблемами электромагнетизма. В 1837 г. он создает свою первую «телеграфную машину», а спустя всего несколько месяцев – 6 января 1838 г. – передает первое сообщение, используя специальный алфавит. Позднее, как вы догадались, этот алфавит назвали в его честь *азбукой Морзе*. Летом 1841 г. С. Морзе получил в Нью-Йорке патент на свое изобретение.

У Морзе не было приоритета в изобретении электрического телеграфа, но он оказался первым, кто применил систему передачи алфавитного кода. Позднее Морзе усовершенствовал свою азбуку, изменив ряд кодов и добавив новые.

А затем, в 1851 году, на проходившей в Европе конференции был утвержден так называемый *международный Морзе*. В нем окончательно остались лишь 2 знака – точка и тире, и все сигналы комбинируются из этих двух знаков.



На этом развитие систем передачи сообщений на основе азбуки Морзе не остановилось. Британские военные моряки в 1865 г. стали применять ее в «беспроволочном» телеграфе, используя днем флажки, а ночью – фонари. В 1897 г. с той же целью стали применять прожектора, закрывая и открывая специальные

1	2	3	4	5	6
1	E	5	.	Е,Ё	6,7
2	T	20	–	Т	20
3	I	9	..	И	10
4	A	1	.–	А	1
5	N	14	–.	Н	15
6	M	13	––	М	14
7	S	19	...	С	19
8	R	18	.–.	Р	18
9	D	4	–..	Д	5
10	U	21	..–	У	21
11	W	23	.––	В	3
12	G	7	––.	Г	4
13	K	11	–.–	К	12
14	O	15	–––	О	16
15	H	8	....	Х	23
16	L	12	.–..	Л	13
17	F	6	..–.	Ф	22
18	B	2	–...	Б	2
19	V	22	...–	Ж	8
20	C	3	–.–.	Ц	24
21	P	16	.––.	П	17
22	X	24	–..–	Ь,Ъ	28,29
23	Z	26	––..	З	9
24	Y	25	–.––	Ы	30
25	Q	17	––.–	Щ	27
26	J	10	.––-	Й	11
27			––-.	Ч	25
28			————	Ш	26
29			..––	Ю	32
30			.–.–	Я	33
31			..–..	Э	31
32	1		.———–		
33	2		..––-		
34	3		...––		
35	4		....–		
36	5		.....		
37	6		–....		
38	7		––...		
39	8		———..		
40	9		————.		
41	0		————–		

Таблица 2. *Колонка 1 – порядковый номер строки таблицы; 2 – символ латиницы; 3 – номер символа в латинском алфавите; 4 – кодовые слова Морзе; 5 – символ кириллицы; 6 – номер символа в русском алфавите.*

жалюзи; использовалась и звуковая связь – с помощью сирен. Наконец, в 1901 г. в России и в 1905 г. в Европе применили азбуку Морзе в радиосвязи.

В таблице 2 представлена часть азбуки Морзе в порядке увеличения длины кодовой комбинации, или так называемого кодового слова. В приведенную таблицу включены коды букв международной азбуки Морзе для латинского алфавита и коды цифр; там же приведены соответствующие кодам символы кириллицы, причем их, естественно, больше. Обратите внимание, что заглавные и прописные буквы не различаются. Кстати говоря, я не знаю, можно ли передавать с помощью азбуки Морзе «смешанный» текст, состоящий одновременно из символов латиницы и кириллицы. Если найдется компетентный читатель, с удовольствием воспользуюсь его консультацией...

В чем же основная идея нововведения Морзе?

Он обратил внимание на «неправильный» порядок расположения букв пишущей машинки. Чаще используемые буквы были сконцентрированы (как и теперь!) в окрестностях центра клавиатуры. Соответственно, «редким» буквам он назначил более длинные кодовые комбинации, часто используемым – более короткие.

Любопытно сравнить «качество» подбора кодовых слов. Ч. Мидоу [11], ссылаясь на три различных источника, приводит таблицу букв, упорядоченную по их встречаемости в письменном английском языке. Эти три варианта выглядят так:

1) ETAONISRHLDC<...>,
2) EAOIDHNRSTUY<...>,
3) ETAONRISHDLF<...>,

и, как видим, полного совпадения между ними нет.

Кстати говоря, согласно [10], есть еще и вариант

4) ETAONIRSHDLU<...>.

Последний порядок отражен в клавиатуре типографского устройства под названием «линотип»; его в 1886 г. изобрел Оттмар Мергенталер (O. Mergenthaler).

К вопросу «наилучшего порядка» символов в алфавите мы еще вернемся. Пока же выясним, как, в отсутствие фиксированного значения n в качестве разрядности, различать кодируемые символы. Для знакомого уже примера код Морзе выглядит так:

....	**.–**	**––**	**.–..**	**.**	**–**
H	A	M	L	E	T

Стоит убрать здесь пробелы, и преобразование текста в код Морзе перестает быть обратимым, так как установить *единственным образом*, где заканчивается кодовое слово очередного символа и начинается другое, не удается. Скажем, «сцепленные» друг за другом 6 точек вполне можно расшифровать как SS или IH.

Эту проблему Морзе разрешил, установив между передаваемыми в сообщении буквами и словами специальные *паузы*, причем интервалы между точками и тире, между буквами и между словами различаются своей продолжительностью. Очевидно, что *алфавит Морзе* $A_{Morse} = \{. –\}$ является двухэлементным, но собственно *код Морзе* считать таковым уже нельзя, так как он предусматривает возможность передачи пауз неалфавитным способом.

Стало быть, напрашивающаяся идея построить обратимое отображение

$$\mathbf{F : A_{Morse} \rightarrow \{0,1\}_{bit}}$$

и использовать его для сжатия компьютерных текстов оказывается малопродуктивной.

Литература.

1. Льюис Кэрролл. Приключения Алисы в Стране Чудес. Сквозь зеркало и что там увидела Алиса, или Алиса в Зазеркалье. М.: Наука, 1978. – [Литературные памятники]
2. Шекспир У. Гамлет. Избранные переводы: Сборник / Сост. А. Н. Горбунов. М.: Радуга, 1985.
3. Флоренский П.А. Имена: Сочинения. М.: ЗАО Изд-во ЭКСМО-Пресс; Харьков: Изд-во Фолио, 1998.

4. Ионеско Эжен. Собрание сочинений. Между жизнью и сновидением: Пьесы. Роман. Эссе. / Пер. с франц. СПб.: «Симпозиум», 1999.
5. Фоли Дж. Энциклопедия знаков и символов. М.: Вече, АСТ, 1997.
6. Новый Большой англо-русский словарь: в 3-х т. / Апресян Ю.Д. и др. М.: Рус. яз., Т.1 A–F, 1993.
7. Даль Владимир. Толковый словарь живого великорусского языка: Т.1–4. М.: Рус. яз., Т.1. А–З, 1981.
8. Пойа Д. Как решать задачу. Изд-е 2-е. М., 1961.
9. Мартин Дж. Организация баз данных в вычислительных системах. Изд-е 2-е, доп. М.: Мир, 1980.
10. Бауэр Ф. Л., Гооз Г. Информатика. Вводный курс: В 2-х ч. Ч.1. Пер. с нем. М.: Мир, 1990.
11. Мидоу Ч. Анализ информационно-поисковых систем. Введение для программистов. Пер. с англ. М.: Мир, 1970.
12. Григорьев Олег. Птица в клетке. СПб.: Изд-во Ивана Лимбаха, 1997.

Столяр Сергей Ефимович,
учитель информатики,
гимназия № 470,
лицей «Физико-Техническая школа»,
Санкт-Петербург.