МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
БОРИСОГЛЕБСКИЙ ФИЛИАЛ
(БФ ФГБОУ ВО «ВГУ»)

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ
Теоретические основы информатики

1. Код и наименование направления подготовки:

44.03.01 Педагогическое образование

2. Профиль подготовки:

Информатика и информационные технологии в образовании

3. Квалификация (степень) выпускника:

Бакалавр

4. Форма обучения:

Заочная

5. Кафедра, отвечающая за реализацию дисциплины:

Кафедра прикладной математики, информатики, физики и методики их преподавания

6. Составитель:

В. В. Волков, кандидат физико-математических наук, доцент

7. Методические указания для обучающихся по освоению дисциплины

Приступая к изучению учебной дисциплины, прежде всего обучающиеся должны ознакомиться с учебной программой дисциплины. Электронный вариант рабочей программы размещён на сайте БФ ВГУ.

Знание основных положений, отраженных в рабочей программе дисциплины, поможет обучающимся ориентироваться в изучаемом курсе, осознавать место и роль изучаемой дисциплины в подготовке будущего педагога, строить свою работу в соответствии с требованиями, заложенными в программе.

Основными формами контактной работы по дисциплине являются лекции, практические и лабораторные занятия, посещение которых обязательно для всех студентов (кроме студентов, обучающихся по индивидуальному плану).

В ходе лекционных занятий следует не только слушать излагаемый материал и кратко его конспектировать, но очень важно участвовать в анализе примеров, предлагаемых преподавателем, в рассмотрении и решении проблемных вопросов, выносимых на обсуждение. Необходимо критически осмысливать предлагаемый материал, задавать вопросы как уточняющего характера, помогающие уяснить отдельные излагаемые положения, так и вопросы продуктивного типа, направленные на расширение и углубление сведений по изучаемой теме, на выявление недостаточно освещенных вопросов, слабых мест в аргументации и т.п.

Не следует дословно записывать лекцию, лучше попытаться понять логику изложения и выделить наиболее важные положения лекции в виде опорного конспекта. Рекомендуется использовать различные формы выделения наиболее сложного, нового, непонятного материала, который требует дополнительной проработки: можно пометить его знаком вопроса (или записать на полях сам вопрос), цветом, размером букв и т.п. – это поможет быстро найти материал, вызвавший трудности, и в конце лекции (или сразу же, попутно) задать вопрос преподавателю (не следует оставлять непонятый материал без дополнительной проработки, без него иногда бывает невозможно понять последующие темы). Материал уже знакомый или понятный нуждается в меньшей детализации – это поможет сэкономить усилия во время конспектирования.

На практических занятиях необходимо активно участвовать в решении предлагаемых задач, начиная уже с этапа анализа условия и поиска путей решения. Студенту, вызванному для решения задачи к доске, следует подробно комментировать ход решения задачи, а стальным студентам — выполнять основные этапы решения предложенной задачи самостоятельно, но при этом контролируя ход решения на доске. При возникновении проблем с решением какой-либо задачи, рекомендуется сразу же задать вопрос преподавателю: непонимание, возникшее, при решении одной задачи, может помешать решать последующие.

При выполнении лабораторных работ следует пользоваться конспектом лекций и тетрадью с решением задач с практических занятий. Решения оформляются с использованием текстового процессора (например, MS Word) и содержат, помимо ответов, подробное решение каждой задачи. Также при решении задач рекомендуется активно использовать для проведения необходимых расчётов и вычислений возможности табличных процессоров (например, MS Excel) и (по желанию) известных обучающимся языков программирования.

При подготовке к промежуточной аттестации необходимо повторить пройденный материал в соответствии с учебной программой, примерным перечнем вопросов, выносящихся на зачет. Рекомендуется использовать конспекты лекций и источники, перечисленные в списке литературы в рабочей программе дисциплины, а также

ресурсы электронно-библиотечных систем. Необходимо обратить особое внимание на темы учебных занятий, пропущенных по разным причинам. При необходимости можно обратиться за консультацией и методической помощью к преподавателю.

8. План лекционных занятий

Тема №1. Теоретическая информатика

- Информатика как наука. Структура и место в системе наук.
- Основные понятия теоретической информатики.
- Основы теории кодирования

Тема №2. Теория кодирования информации.

- Основные понятия. Постановка задачи кодирования

Тема №3. Представление информации в памяти компьютера

- Представление информации с использованием равномерного кодирования
- Представление числовой информации. Целые числа.
- Представление текстовой информации (символов). Кодировки.
- Представление графической информации.
- Представление звуковой информации.

Тема №4. Эффективное и помехоустойчивое кодирование

- Эффективное кодирование.
- Методы эффективного кодирования.
- Оценка эффективности кодирования.
- Помехоустойчивое кодирование. Простейшие методы построения помехоустойчивых кодов.

9. Методические материалы к практическим занятиям по теме «Теория кодирования информации»

Теория кодирования информации является одним из разделов теоретической информатики. К её основным задачам относятся следующие:

- разработка принципов *наиболее экономичного* кодирования информации;
- согласование параметров передаваемой информации с особенностями канала связи;
- разработка приемов, обеспечивающих *надёжность* передачи информации по каналам связи, то есть отсутствие потерь информации.

Для представления дискретной информации используется некоторый *алфавит*. Однозначное соответствие между информацией и алфавитом отсутствует. Другими словами, одна и та же информация может быть представлена посредством различных алфавитов. Поэтому возникает проблема перехода от одного алфавита к другому, причем такое преобразование не должно приводить к потере информации. Условимся называть алфавит, с помощью которого представляется информация до преобразования, *первичным*; алфавит конечного представления — *вторичным*.

Введем ряд определений:

Код — правило, описывающее соответствие знаков или их сочетаний одного алфавита знакам или их сочетаниям другого алфавита, а также знаки вторичного алфавита, используемые для представления знаков или их сочетаний первичного алфавита.

Кодирование — перевод информации, представленной посредством первичного алфавита, в последовательность кодов.

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

Операции кодирования и декодирования называются *обратимыми*, если их последовательное применение обеспечивает возврат к исходной информации без каких-либо её потерь.

Виды кодирования

В зависимости от целей кодирования, различают следующие его виды:

1) *Кодирование по образцу* — каждый знак дискретного сигнала представляется знаком или набором знаков того алфавита, в котором выполняется кодирование. Используется, в частности, всякий раз для ввода информации в компьютер для ее внутреннего представления.
2) *Эффективное* (оптимальное) кодирование — для устранения избыточности данных путем снижения среднего числа символов кодового алфавита для представления одного исходного символа. Именно этот вид кодирования используется для сжатия.
3) *Помехоустойчивое* (помехозащитное) кодирование — для обеспечения заданной достоверности в случае, когда на сигнал накладывается помеха. Используется для защиты от помех при передаче информации по каналам связи.
4) *Криптографическое* кодирование (шифрование) — используется, когда нужно защитить информацию от несанкционированного доступа.

Приведём математическую постановку задачи кодирования.

Пусть первичный алфавит A содержит N знаков со средней информацией на знак, определённой с учетом вероятностей их появления, $I_1(A)$ (нижний индекс отражает то обстоятельство, что рассматривается первое приближение, а буква в скобках указывает алфавит). Вторичный алфавит B пусть содержит M знаков со средней информационной ёмкостью $I_1(B)$. Пусть также исходное сообщение, представленное в первичном алфавите, содержит n знаков, а закодированное сообщение — m знаков. Если исходное сообщение содержит $I(A)$ информации, а закодированное — $I(B)$, то условие обратимости кодирования, то есть *неисчезновения* информации при кодировании, очевидно, может быть записано следующим образом:

$$I(A) \le I(B).$$

Смысл неравенства в том, что *операция обратимого кодирования может увеличить количество формальной информации в сообщении, но не может его уменьшить*. Каждая из величин в данном неравенстве может быть заменена произведением числа знаков на среднюю информационную емкость знака:

$$I_1(A) \le \frac{m}{n} I_1(B), \quad I_1(A) = -\sum_{i=1}^{N} p_i \log_2 p_i.$$

Отношение $K(B) = m / n$, очевидно, характеризует *среднее число знаков вторичного алфавита, которое приходится использовать для кодирования одного знака первичного алфавита* — будем называть его *длиной кода*.

В частном случае, когда появление любых знаков вторичного алфавита равновероятно, согласно формуле Хартли $I_1(B) = \log_2 M$, и тогда

$$\frac{I_1(A)}{\log_2 M} \le K(B).$$

10. Методические материалы к лекционным и практическим занятиям по теме «Представление числовой информации в памяти компьютера»

Вся информация, с которой работает (хранит, обрабатывает и передает по сетям) компьютер представлена в цифровой форме. Вне зависимости от типа информации (текстовая, числовая, графическая, звуковая или видеоинформация), во внутреннем представлении компьютера она обрабатывается в виде двоичного кода — последовательностей нулей и единиц (битов).

Использование двоичного кода объясняется многими причинами: это и высокая вычислительная эффективность представления любой дискретной информации в двоичном коде, и возможность реализации любой операции над двоичными кодами с помощью базовых операций математической логики, а также удобство при реализации логических элементов на основе различных физических процессов и явлений, наличие (единица) или отсутствие (ноль) которых легко различить.

Для представления числовой информации в памяти компьютера, разумеется, также используется двоичная система счисления.

Представление целых чисел в компьютере

Целые числа могут представляться в компьютере со знаком или без знака.

Целые числа без знака обычно занимают в памяти один или два байта и принимают в однобайтовом формате значения от 00000000_2 до 11111111_2, а в двухбайтовом формате — от 00000000 00000000_2 до 11111111 11111111_2.

Разрядность (в битах)	**Диапазон**		**Тип в ЯП Turbo Pascal**
8	от 0 до 255	от 0 до 2^8–1	Byte
16	от 0 до 65535	от 0 до 2^{16}–1	Word

В общем случае максимальное значение беззнакового числа $N_{\max} = 2^k - 1$, где k — число разрядов, отводимых под запись числа.

Целые числа со знаком обычно занимают в памяти компьютера один, два или четыре байта, при этом самый левый (старший) разряд содержит информацию о знаке числа. Знак «плюс» кодируется нулем, а «минус» — единицей.

Диапазоны значений целых чисел со знаком (определенные в ЯП Turbo Pascal):

Разрядность (в битах)	**Диапазон**		**Тип в ЯП Turbo Pascal**
8	от -128 до 127	от -2^7 до 2^7–1	Shortint
16	от -32 768 до 32 767	от -2^{15} до 2^{15}–1	Integer
32	от -2 147 483 648 до 2 147 483 647	от -2^{31} до 2^{31}–1	Longint

Отметим, что существуют целые числа и с большей разрядностью.

В общем случае минимальное значение целого числа со знаком $N_{\min} = -2^{k-1}$, а максимальное — $N_{\max} = 2^{k-1} - 1$, где k — число разрядов, отводимых под запись числа.

Рассмотрим особенности записи целых чисел со знаком на примере *однобайтового формата*, при котором для знака отводится один разряд, а для цифр абсолютной величины числа — семь разрядов.

В компьютерной технике применяются три формы записи (кодирования) целых чисел со знаком: *прямой* код, *обратный* код, *дополнительный* код. Последние две формы применяются особенно широко, так как позволяют упростить конструкцию арифметико-логического устройства компьютера путем замены разнообразных арифметических операций операцией сложения.

Положительные числа в прямом, обратном и дополнительном кодах записываются одинаково — двоичными кодами с цифрой 0 в знаковом разряде.

Отрицательные числа в прямом, обратном и дополнительном кодах записываются по-разному:

Прямой код. В знаковый разряд помещается цифра 1, а в разряды цифровой части числа — двоичный код его абсолютной величины.

Например, прямой код числа –42: 10101010.

Обратный код получается инвертированием всех цифр двоичного кода числа, исключая разряд знака: нули заменяются единицами, а единицы — нулями.

Обратный код числа –42: 11010101.

Дополнительный код получается образованием обратного кода с последующим прибавлением единицы к его младшему ряду.

Дополнительный код числа –42: 11010110.

Сложение и вычитание целых чисел

В большинстве компьютеров операция вычитания не используется. Вместо нее производится сложение уменьшаемого с обратным или дополнительным кодом вычитаемого. Это позволяет существенно упростить конструкцию АЛУ (арифметико-логического устройства).

При *сложении обратных кодов* чисел A и B в случае, если A положительное, B отрицательное и по абсолютной величине меньше A, или A и B отрицательные, будет неправильный результат, который компьютер исправляет переносом единицы из знакового разряда в младший разряд суммы.

При *сложении дополнительных кодов* получается сразу правильный результат.

Отсюда следует, что время выполнения сложения для дополнительных кодов чисел меньше, чем для их обратных кодов, потому что в таком сложении нет переноса единицы из знакового разряда в младший разряд результата.

Следует помнить, что при сложении может возникнуть ситуация, когда старшие разряды результата операции не помещаются в отведенной для него области памяти. Такая ситуация называется переполнением разрядной сетки формата числа. Это происходит, если знаки складываемых чисел совпадают, а сумма их модулей превышает 2^{k-1}, где k — количество разрядов формата чисел.

Представление в компьютере вещественных чисел

Вещественными числами (в отличие от целых) называются числа, имеющие дробную часть.

Любое число N в системе счисления с основанием q можно записать виде $N = M \cdot q^p$, где M называется *мантиссой* числа, а p — *порядком (экспонентой)*. Такой способ записи чисел называется представлением числа *с плавающей точкой (запятой)*.

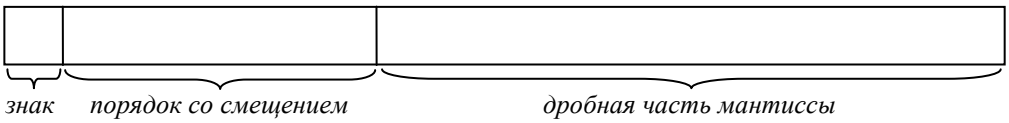
С точки зрения «экономии» разрядов, отводимых под мантиссу, наиболее выгодной является запись, не содержащая нулей в начале или конце числа. Поэтому порядок выбирают таким образом, чтобы мантисса находилась в диапазоне $M \in [1, q)$, где q — основание системы счисления.

Такое наиболее выгодное для компьютера представление вещественных чисел называется *нормализованным*.

Например, десятичное число 123.4 в нормализованном виде будет записано как $1.234 \cdot 10^2$.

На сегодняшний день наиболее используемым стандартом для представления действительных чисел в двоичном коде с плавающей точкой является стандарт IEEE 754.

Согласно стандарту IEEE 754, вещественное число записывается в памяти компьютера в виде последовательности битов, где самый старший бит отводится под знак числа, следующие несколько битов — под смещённый порядок, и оставшиеся биты — под дробную часть мантиссы:



IEEE 754 определяет несколько форматов представления чисел с плавающей запятой, которые отличаются числом разрядов, отводимых под запись числа, и, следовательно, разной точностью представления числа.

Отметим также, что чем больше разрядов отводится под запись мантиссы, тем выше точность представления числа. Чем больше разрядов занимает порядок, тем шире диапазон от наименьшего, отличного от нуля числа, до наибольшего числа, представимого в машине при заданном формате.

Для примера рассмотрим формат представления с двойной точностью (double precision), под запись числа в котором отводится 64 бита (в ЯП Turbo Pascal этому формату соответствует тип Double).

В этом формате 52 бита (с 0 по 51) отводятся под запись дробной части мантиссы (так как для нормализованного двоичного числа целая часть всегда равна единице и эту единицу можно не хранить), 11 бит (с 52 по 62) предназначены для хранения показателя, сложенного с числом $1111111111_2 = 1023_{10}$ (чтобы не хранить знак показателя), а самый старший бит (63) определяет знак числа.

Арифметические действия над нормализованными числами в компьютере

Сложение и вычитание. При сложении и вычитании сначала производится подготовительная операция, называемая *выравниванием кодов*.

В процессе выравнивания порядков мантисса числа с меньшим порядком, сдвигается в своем регистре вправо на количество разрядов, равное разности порядков операндов. После каждого сдвига порядок увеличивается единицу.

В результате выравнивания порядков одноименные разряды чисел оказываются расположенными в соответствующих разрядах обоих регистров, после чего мантиссы складываются или вычитаются (складываются с отрицательными числами). В случае необходимости полученный результат нормализуется.

Умножение. При умножении двух нормализованных чисел их порядки складываются, а мантиссы перемножаются.

Деление. При делении двух нормализованных чисел из порядка делимого вычитается порядок делителя, а мантисса делимого делится на мантиссу делителя. Затем в случае необходимости полученный результат нормализуется.

11. Методические материалы к лекционным и практическим занятиям по теме «Представление текстовой, графической и звуковой информации в памяти компьютера»

Как уже упоминалось, вся информация, с которой работает компьютер, представлена в цифровой форме — в виде двоичного кода.

Как известно, информация передается в виде сообщений. Дискретная информация записывается с помощью некоторого конечного набора знаков, которые обычно называют буквами, не вкладывая в это слово привычного ограниченного значения. *Буква* в данном расширенном понимании — любой из знаков, которые некоторым соглашением установлены для общения (элемент некоторого конечного множества (набора) отличных друг от друга знаков). Множество знаков, в котором определен их порядок, называют *алфавитом*.

В канале связи сообщение, составленное из символов (букв) одного алфавита, может преобразовываться в сообщение из символов (букв) другого алфавита. Правило, описывающее однозначное соответствие букв алфавитов при таком преобразовании, называют *кодом*. Саму процедуру преобразования сообщения называют *(пере)кодировкой*. Подобное преобразование сообщения может осуществляться в момент поступления сообщения от источника в канал связи (*кодирование*) и в момент приема сообщения получателем (*декодирование*).

Во внутреннем представлении компьютера кодирование любых символов основывается на сопоставлении каждому из них определенной группы двоичных знаков. При этом из технических соображений обычно пользуются равномерными кодами, то есть двоичными группами равной длины.

Существуют международные стандарты и методы кодирования текстовой, графической и звуковой информации.

Представление текстовых данных

Рассмотрим кодирование текстовой информации. Оценим, сколько различных кодов потребуются для представления необходимых символов.

В англоязычных странах используются 26 прописных и 26 строчных букв (A ... Z, a ... z), 9 знаков препинания (.,:!";?()), пробел, 10 цифр, 5 знаков арифметических действий (+-*/^), специальные символы (№%_#$&><|\), а также управляющие команды (удаление символа, переход на новую строку и т.п.) — всего чуть больше 100 символов.

Чтобы определить, сколько различных кодовых комбинаций можно получить, используя равномерный двоичный код длины k, можно воспользоваться несложной формулой:

$$N = 2^k .$$

Таким образом, для кодирования перечисленных выше символов можно ограничиться 7-разрядным двоичным числом (от 000000 до 1111111 или от 0 до 127_{10}), так как $2^7 = 128$, а $2^6 = 64$, чего недостаточно.

Первой такой 7-разрядной кодовой таблицей была ASCII (American Standard Code for Information Interchange — Американский стандартный код для обмена информацией), опубликованная как стандарт еще в 1963 г.

Однако в такую 7-битную кодировку не входят символы языков, использующих отличные от латинских буквы, в частности, — русского. Для национальных алфавитов было предложено использовать 8-битную кодовую таблицу, которая могла содержать дополнительно еще 128 символов. При этом первые 128 символов оставались теми же, что и в 7-битной таблице, а символы с 128 по 255 отводились для неанглийских символов. Для таких таблиц было введено понятие «*кодовая страница*» (англ. code page, CP).

Так как для всех символов национальных алфавитов отводилось только 128 кодов, то для каждой страны была принята своя кодовая страница, как правило, не одна. Так, например, наиболее известными кодировками русских букв являются:

- CP1251 (в операционной системе MS Windows);
- KOI8-R (в операционной системе Unix);
- CP866 (в операционной системе MS DOS).

При этом текст, набранный в одной из кодировок, будет невозможно прочитать при просмотре в другой кодировке.

Кроме того, любая 8-битная кодовая страница может включать только 256 символов. Поэтому при наборе текста может быть использован только один язык (не считая английского).

В 1991 году был разработан стандарт Unicode (Юникод), позволяющий увеличить количество кодов в кодовой таблице и хранить в ней сразу все национальные (и другие при необходимости) символы.

Первая версия Unicode отводила для хранения каждого символа 2 байта и, таким образом, позволяла хранить одновременно 65 536 (2^{16}) различных символов. Современные версии Unicode теоретически позволяют кодировать до 2 147 483 648 (2^{31}) кодовых позиций, но из соображений совместимости разрешено использовать лишь 1 112 064. При этом на практике в версии Unicode 6.0, принятой в 2010 году, используется чуть менее 110 000 кодовых позиций.

Юникод имеет несколько форм представления (англ. Unicode transformation format, UTF): UTF-8, UTF-16 и UTF-32.

В системах Microsoft Windows семейства NT используется UTF-16 (размером 2 байта или 4 байта в зависимости от кодируемого символа) для внутреннего представления текста.

В UNIX-подобных операционных системах (а также в последнее время все чаще для представления текстовой информации на HTML-страницах) используется форма UTF-8, имеющая переменный размер от 1 до 4 байт и обеспечивающая совместимость с ASCII. В UTF-8 все символы разделены на несколько групп по значению первых битов. Символы с кодами менее 128 кодируются одним байтом и соответствуют символам таблицы ASCII, следующие 1920 символов — двумя байтами и содержат национальные символы. Последующие более редко используемые символы кодируются тремя и четырьмя байтами.

Представление графических изображений

Для кодирования графической информации используют два принципиально разных метода: *растровое кодирование* и *векторное кодирование*.

В *векторном* формате изображение разделяется на *примитивы* — прямые линии, многоугольники, окружности и сегменты окружностей, параметрические кривые, залитые определенным цветом или шаблоном, связные области, набранные определенным шрифтом отрывки текста и т.д. Для пересекающихся примитивов задается порядок, в котором один из них перекрывает другой. Некоторые форматы, например, PostScript, позволяют задавать собственные примитивы аналогично тому, как в языках программирования можно описывать подпрограммы. Такие форматы часто имеют переменные и условные операторы и представляют собой полнофункциональный (хотя и специализированный) язык программирования.

Каждый примитив описывается своими геометрическими координатами. Точность описания в разных форматах различна, нередко используются числа с плавающей точкой двойной точности или с фиксированной точкой и точностью до 16-го двоичного знака.

Координаты примитивов бывают как двух-, так и трехмерными. Для трехмерных изображений, естественно, набор примитивов расширяется, в него включаются и различные поверхности — сферы, эллипсоиды и их сегменты, параметрические многообразия и др.

Двумерные векторные форматы очень хороши для представления чертежей, диаграмм, шрифтов и отформатированных текстов. Такие изображения удобно редактировать — изображения и их отдельные элементы легко поддаются масштабированию и другим преобразованиям. Примеры двухмерных векторных форматов — PostScript (язык описания страниц, в основном используемый в настольных издательских системах), WMF (Windows MetaFile), SVG (Scalable Vector Graphics), Adobe Flash.

Однако преобразование реальной сцены (например, полученной оцифровкой видеоизображения или сканированием фотографии) в векторный формат представляет собой сложную и, в общем случае, неразрешимую задачу. Создание фотореалистичных изображений в векторном формате хотя теоретически и возможно, на практике требует большого числа очень сложных примитивов. Гораздо более практичным для этих целей оказался другой подход к оцифровке изображений, который использует большинство современных устройств визуализации: растровые дисплеи и многие печатающие устройства.

В *растровом* формате изображение разбивается на прямоугольную матрицу элементов, называемых *пикселями* (слегка искаженное PICture ELement — элемент картинки). Матрица называется *растром*. Для каждого пикселя определяется его яркость и, если изображение цветное, цвет. Если, как это часто бывает при оцифровке реальных сцен или преобразовании в растровый формат (растеризации) векторных изображений, в один пиксель попали несколько элементов, их яркость и цвет усредняются с учетом занимаемой площади.

Количество пикселей, приходящихся на единицу длины, называется *разрешением* растрового изображения. Одна из наиболее распространённых единиц, в которых измеряется разрешение, — число точек на дюйм (DPI, Dots Per Inch).

Количество битов в двоичном коде цвета каждого пикселя называется *глубиной цвета*. Для черно-белых изображений достаточно одного бита на пиксель, для градаций яркости серого или цветовых составляющих изображения необходимо несколько битов.

Если глубина цвета равна i, то равномерным кодом может быть закодировано $N = 2^i$ различимых оттенков цвета.

Информационную ёмкость несжатого растрового изображения в этом случае можно найти по простой формуле $I = h \times w \times i$, где $h \times w$ — размер изображения в пикселях.

Если число цветов в изображении невелико (не более 256), то применяют кодирование с палитрой. *Палитра* — таблица, содержащая используемые цвета и их коды.

Если же требуется поддержка большого количества цветов в изображении, то применяют кодирование цвета с помощью цветовых моделей.

Самой известной является цветовая модель RGB, в которой цвет формируется за счет смешивания трех базовых цветов разной интенсивности: красного (red), зеленого (green) и синего (blue). Если интенсивность каждого базового цвета кодируется 8 битами, то глубина цвета равна 24, а палитра содержит $2^{24} = 16\,777\,216$ оттенков цвета (система точной цветопередачи True Color). В настоящее время используется вариант системы True Color, использующий 32 бита — дополнительные 8 бит резервируются для альфа-канала (прозрачность).

Кроме RGB широко используются также цветовые модели CMYK (Cyan, Magenta, Yellow, Key color — голубой, пурпурный, желтый, «ключевой» цвет — как правило, чёрный) и HSV (Hue, Saturation, Value — тон, насыщенность, значение) и другие.

Наиболее известные форматы растровых рисунков: BMP (BitMap Picture — растровое изображение), JPEG (Joint Photographic Experts Group — объединённая группа экспертов по фотографии), GIF (Graphics Interchange Format — формат для обмена изображениями), PNG (Portable Network Graphics — портативные сетевые изображения).

Представление звуковой информации

Звук можно упрощенно представить как акустическую волну с непрерывно меняющейся амплитудой и частотой. Амплитуда сигнала определяет его громкость, а частота — тон: чем больше частота сигнала, тем выше тон. Важно также подчеркнуть, что существует определенный диапазон частот, к которому принадлежат звуковые волны: примерно от нескольких десятков герц до величины немного более 20 кГц. Значения этих границ определяются возможностями человеческого слуха.

Существуют два способа представления звуковой информации в памяти компьютера:

- *цифровая запись (оцифровка)*, когда реальные звуковые волны преобразуются в цифровую информацию путем измерения параметров звуковых колебаний тысячи раз в секунду;
- *инструментальное кодирование (MIDI-запись)* (MIDI — Musical Instrument Digital Interface — интерфейс цифровых музыкальных инструментов), когда записываются определенные команды-указания синтезатору.

Для записи звуковых сигналов с целью последующего воспроизведения необходимо как можно точней сохранить форму кривой зависимости интенсивности звука от времени. При этом возникает одна очень важная и принципиальная трудность: звуковой сигнал непрерывен, а компьютер способен хранить в памяти только дискретные величины. Отсюда следует, что в процессе сохранения звуковой информации она должна быть *оцифрована*, то есть из аналоговой непрерывной формы переведена в *цифровую* (*дискретную*). Данную функцию выполняет специальный блок, входящий в состав звуковой карты, который называется *аналого-цифровой преобразователь* (АЦП).

АЦП, во-первых, производит дискретизацию записываемого звукового сигнала по времени. Это означает, что измерение уровня интенсивности звука ведется не непрерывно, а, напротив, в определенные фиксированные моменты времени. Частоту,

характеризующую периодичность измерения звукового сигнала, принято называть *частотой дискретизации*. Ее выбор в значительной степени зависит от спектра сохраняемого сигнала: согласно теореме Котельникова, частота «оцифровки» звука должна как минимум в 2 раза превышать максимальную частоту, входящую в состав спектра сигнала. Так как максимальной частотой звука, который способен слышать человек, считается частота порядка 20 кГц, для высококачественного воспроизведения звука частота звукозаписи принимается в 44,1 кГц. Также используются частоты 48 кГц, 96 кГц, 192 кГц. Однако если высокое качество не требуется, частоту дискретизации можно значительно снизить. Например, для записи речи достаточно частоты дискретизации 8 кГц.

Во-вторых, АЦП производит дискретизацию амплитуды (определяющей громкость) звукового сигнала (дискретизация по уровню): при измерении имеется шкала стандартных уровней (например, 256 или 65536 — это количество характеризует *глубину кодирования*), и текущий уровень измеряемого сигнала округляется до ближайшего из них. Человек различает примерно 110 уровней громкости, для которых постоянной является не разность, а отношение соседних уровней (логарифмическая шкала дискретизации). Кроме логарифмической, используются и равномерная дискретизация амплитуды. Наиболее часто используется глубина кодирования 16 до 24 бит.

В простейшем случае для оценки информационной ёмкости несжатого звукового файла можно воспользоваться формулой: $I = h \cdot f \cdot t \cdot n$, где h — глубина кодирования, f — частота дискретизации, t — длительность звучания, n — число звуковых каналов.

Воспроизведение закодированного звука осуществляется при помощи *цифро-аналогового преобразователя* (ЦАП). Двоичные числа, кодирующие звук, подаются на вход с точно такой частотой, как и при дискретизации, и ЦАП преобразует их в электрические напряжения обратно тому, как это делал АЦП. Ступенчатый сигнал, выходящий из ЦАП, сглаживается при прохождении через аналоговый фильтр, а затем преобразуется в звук с помощью усилителя и динамика.

Для сжатых аудио-файлов вводят характеристику, называемую *битрейтом* (bit rate, битовая частота) — она определяет количество передаваемой за единицу времени информации.

Некоторые из распространённых аудио-форматов: WAV (Waveform Audio File Format), MP3 (MPEG Layer 3), WMA (Windows Media Audio).

12. Методические материалы к лекционным и практическим занятиям по теме «Эффективное кодирование»

Используя обозначения для математической постановки задачи кодирования, введем в рассмотрение новую величину — *относительную избыточность кода* (Q):

$$Q = 1 - \frac{I(A)}{I(B)} = 1 - \frac{I_1(A)}{K(B) I_1(B)}.$$

Данная величина показывает, насколько операция кодирования увеличила длину исходного сообщения. Очевидно, чем меньше Q (то есть чем ближе она к *0* или, что тоже, $I(B)$ ближе к $I(A)$), тем более выгодным оказывается код и более *эффективной* операция кодирования.

Отметим, что на практике для оценки средней длины кода пользуются формулой из теории вероятностей:

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

где k_i — длина кода для символа с частотой p_i.

Для теории связи важнейшее значение имеет следующая теорема.

Первая теорема Шеннона о передаче информации, которая называется также *основной теоремой о кодировании при отсутствии помех*, формулируется следующим образом:

При отсутствии помех передачи всегда возможен такой вариант кодирования сообщения, при котором избыточность кода будет сколь угодно близкой к нулю.

Важно, что теорема открывает принципиальную возможность оптимального кодирования. Однако необходимо сознавать, что из теоремы никоим образом не следует, как такое кодирование осуществить практически.

С практической точки зрения наиболее просто реализуемый вариант – ситуация, когда $M = 2$, то есть для представления кодов в линии связи используется лишь два типа сигналов. Подобное кодирование называется *двоичным*. Знаки двоичного алфавита принято обозначать «*0*» и «*1*», но нужно воспринимать их как буквы, а не цифры. Удобство двоичных кодов и в том, что при равных длительностях и вероятностях каждый элементарный сигнал (*0* или *1*) несет в себе 1 бит информации ($\log_2 M = 1$); тогда

$$I_1(A) = K(2) ,$$

и первая теорема Шеннона получает следующую интерпретацию:

При отсутствии помех передачи средняя длина двоичного кода может быть сколь угодно близкой к средней информации, приходящейся на знак первичного алфавита.

Тогда формула избыточности для двоичного кодирования дает:

$$Q = 1 - \frac{I_1(A)}{K(2)} .$$

Итак, первая теорема Шеннона открывает возможность эффективного кодирования. Рассмотрим подходы к построению эффективных кодов.

Согласно теории информации Шеннона, количество получаемой с сообщением информации тем больше, чем неожиданнее данное сообщение. Этот тезис использован при эффективном кодировании кодами переменной длины: исходные символы, имеющие большую вероятность (или частоту), имеют код меньшей длины и наоборот.

Таким образом, эффективное кодирование предполагает использование кодов разной длины — коротких для часто встречающихся символов и более длинных — для редких. Но в таком случае возникает проблема отделения кодов разных символов друг от друга — нужно знать, когда закончился один код и начался другой.

Для этого можно вести специальный *код-разделитель* (например, в коде Морзе для этого используется пауза), но это понизит эффективность кода, так как увеличит его длину. Однако очевидно, что эффективность данного способа кодирования невелика. Легко заметить, что одной из причин этого является присутствие в коде каждого символа «лишних» бит — разделителя, не несущего информации.

Можно найти такой вариант кодирования сообщения, при котором последующее выделение из него каждого отдельного знака (то есть декодирование) оказывается

однозначным без специальных указателей разделения знаков. Наиболее простыми и употребимыми кодами такого типа являются так называемые *префиксные коды*, которые удовлетворяют следующему условию (*условие Фано*):

Неравномерный код может быть однозначно декодирован, если никакой из кодов не совпадает с началом какого-либо иного более длинного кода.

Например, если имеется код *110*, то уже не могут использоваться коды *1*, *11*, *1101*, *110101* и пр. Если условие Фано выполняется, то при прочтении (расшифровке) закодированного сообщения путём сопоставления со списком кодов всегда можно точно указать, где заканчивается один код и начинается другой.

Таким образом, использование префиксного кодирования позволяет делать сообщение более коротким, поскольку нет необходимости передавать разделители знаков. Однако условие Фано не устанавливает способа формирования префиксного кода и, в частности, наилучшего из возможных.

Метод Шеннона-Фано

Один из первых методов построения префиксных кодов.

Для построения кода методом Шеннона-Фано можно использовать таблицу или кодовое дерево. Опишем процесс построения кода с помощью таблицы.

Исходное множество символов упорядочивается по невозрастанию частот (вероятностей) их встречаемости в тексте, и выполняются следующие шаги.

1. Список символов делится на две части так, чтобы суммы частот обеих частей были точно или примерно равны.
2. Кодовым комбинациям первой части дописываются единицы, второй части — нули.
3. Если первая часть содержит только один символ, работа с ней заканчивается, и осуществляется переход к шагу 4, иначе — переход к шагу 1 и алгоритм применяется к первой части как к целому алфавиту.
4. Если вторая часть содержит только один символ, работа с ней заканчивается, и осуществляется переход к шагу 5, иначе — переход к шагу 1 и алгоритм применяется ко второй части как к целому алфавиту.
5. Если оставшийся список пуст (в каждой группе по одному символу) — код построен, работа алгоритма заканчивается. Иначе — выполняется шаг 1 для оставшейся группы символов.

Рассмотрим пример.

Пусть даны символы a, b, c, d с частотами $f_a = 0.3;\quad f_b = 0.4; f_c = 0.1; f_d = 0.2$. Построим эффективный код методом Шеннона-Фано.

Процесс построения кода представлен в таблице.

Исходные символы s	Частоты f_s	I	II	III	**Код**
b	0.4	1			**1**
a	0.3	0	1		**01**
d	0.2		0	1	**001**
c	0.1			0	**000**

Метод Хаффмана

Метод кодирования Хаффмана относится к префиксным кодам и является неулучшаемым, то есть невозможно построить префиксный код символьного кодирования с большей эффективностью.

Опишем алгоритм кодирования Хаффмана.

Исходное множество символов упорядочивается по невозрастанию частот (вероятностей) их встречаемости в тексте, и выполняются следующие шаги.

1. Объединение частот:
   - две последние (самые маленькие) частоты складываются, а соответствующие символы исключаются из списка;
   - оставшийся после исключения символов список пополняется суммой частот и вновь упорядочивается;
   - предыдущие шаги повторяются до тех пор, пока не получится единица в результате суммирования и список не уменьшится до одного символа.
2. Построение кодового дерева:
   - строится двоичное кодовое дерево: корнем его является вершина, равная 1; остальные вершины соответствуют либо суммарным, либо исходным частотам, причём для каждой вершины левая подчинённая вершина соответствует большему слагаемому, а правая — меньшему;
   - рёбра дерева кодируются: каждое левое кодируется единицей, каждое правое — нулем.
3. Формирование кода: для получения кодов листьев (исходных кодируемых символов) продвигаются от корня к нужной вершине и записывают веса проходимых ребер.

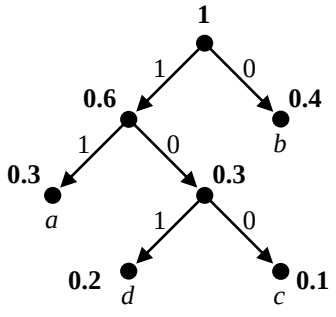
Рассмотрим пример.

Пусть даны символы a, b, c, d с частотами $f_a = 0.3;\ \ f_b = 0.4;\ f_c = 0.1;\ f_d = 0.2$. Построим эффективный код методом Хаффмана.

1. Объединение частот.

Исходные символы s	Частоты f_s	Этапы объединения		
		первый	второй	третий
b	0.4	0.4	0.6	1
a	0.3	0.3	0.4	
d	0.2	0.3		
c	0.1			

2. Построение кодового дерева.



3. Формирование кода.

b: 0;

a: 11;

d: 101;

c: 100.

Блочное двоичное кодирование

При алфавитном кодировании наилучший результат (наименьшая избыточность) может быть получен при кодировании по методу Хаффмана – для русского алфавита избыточность составит менее *1%*. При этом известно, что код Хаффмана улучшить невозможно.

Для повышения эффективности кодирования от алфавитных методов переходят к *блочному кодированию*. 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.

Например, пусть даны символы a и b с частотами, соответственно, 0.9 и 0.1. Построим эффективный код методом Хаффмана для блоков из двух символов.

Сформируем список возможных блоков и их частот (при этом частоту блока будем рассчитывать как произведение частот символов, входящих в блок) и построим код по методу Хаффмана. Тогда имеем:

Блоки	**Частоты**	**Код**
aa	0.81	1
ab	0.09	00
ba	0.09	011
bb	0.01	010

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

$$K_2(2) = 0.81*1 + 0.09*2 + 0.09*3 + 0.01*3 = 1.28.$$

Поскольку в блоке 2 символа, то для одного символа:

$$K(2) = K_2(2) / 2 = 1.28 / 2 = 0.64.$$

При посимвольном кодировании для эффективного кода потребуется по одному двоичному разряду. Таким образом, при блочном кодировании выигрыш составил $1 - 0.64 = 0.36$ двоичных разрядов на один кодируемый символ в среднем.

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

Однако, несмотря на кажущиеся преимущества, применение блочного и словесного 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. При кодировании данным способом реальных словарей, 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. По указанным причинам на практике, как правило, применяется алфавитное кодирование.

13. Методические материалы к лабораторным работам по теме «Представление информации в памяти компьютера»

Задание 1. Представление текстовой информации

1. Создайте простой текстовый файл (с помощью программы Блокнот), содержащий по пять символов с последовательно идущими кодами, начинающимися с кодов 31+№ и 127+№, где № — номер варианта (всего десять символов в одну строку без пробелов). Для генерации символа с заданным кодом можно воспользоваться комбинацией клавиш Alt+код (набирать на цифровой клавиатуре).

С новой строки в этом же файле сгенерируйте ещё десять символов с теми же кодами, но набирая символ «0» (ноль) перед кодом каждого символа: Alt+0код.

С использованием программы просмотра, имеющей режим отображения шестнадцатеричных кодов, просмотрите символы, содержащиеся в файле в кодировках Windows (CP-1251) и MS-DOS (CP-866). Для этого можно использовать встроенные просмотрщики файловых менеджеров FAR, Norton Commander, Total Commander и другие (режим просмотра содержимого файла включается по F3).

Зафиксировать и представить в виде таблиц значения:

a. шестнадцатеричных кодов символов;
b. символов в кодировках Windows и MS-DOS.

Проанализировать полученные результаты и сделать выводы. К чему приводит набор нуля перед кодом символа?

2. Запишите шестнадцатеричные и двоичные коды для символов вашей *Фамилии*, используя кодировку ASCII для кодовых страниц Windows (CP-1251) и DOS (CP-866).

Сравните результаты и сделайте выводы.

3. Декодируйте заданный шестнадцатеричными кодами текст, используя кодировку ASCII. Какая кодовая страница (CP-1251 или CP-866) была использована?

C0 EB E3 EE F0 E8 F2 EC E8 F7 E5 F1 EA E8 E9 20 FF E7 FB EA

4. Рассчитайте размер простого текстового файла, содержащего ваши:

Фамилию Имя Отчество

(каждое слово с новой строки, без пробелов и запятых).

Создайте с помощью программы Блокнот описанный выше текстовый документ (формат *.txt). Сравните результаты своих расчётов с фактическим размером созданного файла. Сделайте выводы.

Задание 2. Представление числовой информации

1. Запишите прямой код числа, интерпретируя его как восьмибитовое целое без знака:

224

2. Запишите дополнительный код чисел, интерпретируя их как восьмибитовое целое со знаком:

а) 115; б) –34

Задание 3. Представление графической информации

Создайте с помощью MS Paint пустой графический документ (рисунок) (размером 740×850), сохраните его в формате bmp (24-разрядный).

Нарисуйте в этом же файле произвольный рисунок и сохраните его в четырёх форматах bmp: 24-разрядный рисунок; 256-цветный рисунок; 16-цветный рисунок; монохромный рисунок.

Рассчитайте размер, который будет занимать каждый из этих файлов на жёстком диске. Сравните результаты расчетов с фактическим размером файлов. Сравните размер пустого файла и файлов с рисунком. Сделайте выводы.

Задание 4. Представление звуковой информации

Рассчитайте объем (в килобайтах) звукового файла без сжатия с параметрами: частота дискретизации 48 кГц, глубина кодирования 16 бит, число каналов: стерео, время звучания 41 с

Кодовая таблица ASCII

Кодировка Windows (CP-1251):

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				©	Ё	§	Є	·		°						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2		!	"	#	$	%	&		(	)	*	+				/
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}]	~	□
	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	Ъ	Ѓ	‚	ѓ	„	…	†	‡		‰	Љ	‹	Њ	Ќ	Ћ	Џ
	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	ђ	‘	’	"	"		–	—		™	љ	›	њ	ќ	ћ	џ
	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A		Ў	ў	J	¤	Ґ	¦	§	Ё	©	Є	«		-	®	Ї
	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	°	±	I	i	ґ	µ		·	ё	№	є	»	j	S	s	ї
	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	▪ 223
E	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Кодировка DOS (CP-866):

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺	☻	♥	♦	♣	♠	•		○						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	►	◄					↕		↑	↓	→	←		↔	▲	▼
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}]	~	□
	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	░	▒	▓	│	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	╨	╤	╥	╙	╘	╒	╓	╫		┘	┌	▉	▄	▌	▐	▀
	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	Ё	ё	Є	є	Ї	ї	÷	≈	°	∙	·	√	№	¤	■	□
	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255