**Лекция 3 24.09. Внутреннее представление информации в ЭВМ**

**План лекции:**

1. Равномерное алфавитное двоичное кодирование. Байтовый код
2. Кодирование графической информации
3. Двоичное кодирование звуковой информации
4. Двоичное кодирование звуковой информации

**Кодирование-- представление символов одного алфавита символами другого. Простейшим алфавитом, достаточным, для кодирования любого другого, является алфавит, состоящий из двух символов 0 и 1(двоичный алфавит).**

**Двоичное кодирование** – один из распространенных способов представления информации. В вычислительных машинах, в роботах и станках с числовым программным управлением, как правило, вся информация, с которой имеет дело устройство, кодируется в виде слов двоичного алфавита.

Двоичный алфавит состоит из двух цифр 0 и 1.

Цифровые ЭВМ (персональные компьютеры относятся к классу цифровых) используют двоичное кодирование любой информации. В основном это объясняется тем, что построить техническое устройство, безошибочно различающее 2 разных состояния сигнала, технически оказалось проще, чем то, которое бы безошибочно различало 5 или 10 различных состояний.

К недостаткам двоичного кодирования относят очень длинные записи двоичных кодов, что затрудняет работу с ними.

1. **Равномерное алфавитное двоичное кодирование. Байтовый код**

В этом случае двоичный код первичного алфавита строится цепочками равной длины, т.е. со всеми знаками связано одинаковое количество информации равное **I0**. Передавать признак конца знака не требуется, поэтому для определения длины кодовой цепочки можно воспользоваться формулой: **K(2) http://it.kgsu.ru/TI_3/images/ris11_1.giflog2N или N>=2m, где N – количество независимых кодируемых значений, m – количество двоичных разрядов необходимых для кодирования N значений.** Приемное устройство просто отсчитывает оговоренное заранее количество элементарных сигналов и интерпретирует цепочку (устанавливает, какому знаку она соответствует). Правда, при этом недопустимы сбои, например, пропуск (непрочтение) одного элементарного сигнала приведет к сдвигу всей кодовой последовательности и неправильной ее интерпретации; решается проблема путем синхронизации передачи или иными способами. С другой стороны, применение равномерного кода оказывается одним из средств контроля правильности передачи, поскольку факт поступления лишнего элементарного сигнала или, наоборот, поступление неполного кода сразу интерпретируется как ошибка.

    Примером равномерного алфавитного кодирования является телеграфный код Бодо, пришедший на смену азбуке Морзе. Исходный алфавит должен содержать не более 32-х символов; тогда **K(2) = log2 32 = 5**, т.е. каждый знак содержит 5 бит информации. Условие **N http://it.kgsu.ru/TI_3/images/ris11_2.gif32**, очевидно, выполняется для языков, основанных на латинском алфавите (**N** = 27 = 26+”пробел”), однако в русском алфавите 34 буквы (с пробелом) – именно по этой причине пришлось "сжать" алфавит (как в коде Хаффмана) и объединить в один знак "е" и "ё", а также "ь" и "ъ". После такого сжатия **N** = 32, однако, не остается свободных кодов для знаков препинания, поэтому в телеграммах они отсутствуют или заменяются буквенными аббревиатурами; это не является заметным ограничением, поскольку, как указывалось выше, избыточность языка позволяет легко восстановить информационное содержание сообщения. Избыточность кода Бодо для русского языка **Q(r) = 0,129**, для английского **Q(e) = 0,193**.

    Другим важным для нас примером использования равномерного алфавитного кодирования является представление символьной информации в компьютере. Чтобы определить длину кода, необходимо начать с установления количество знаков в первичном алфавите. Компьютерный алфавит должен включать:

* 26http://it.kgsu.ru/TI_3/images/ris11_3.gif2=52 букв латинского алфавита (с учетом прописных и строчных);
* 33http://it.kgsu.ru/TI_3/images/ris11_3.gif2=66 букв русского алфавита;
* цифры 0...9 – всего 10;
* знаки математических операций, знаки препинания, спецсимволы http://it.kgsu.ru/TI_3/images/ris11_4.gif20.

    Получаем, что общее число символов **N 148**. Теперь можно оценить длину кодовой цепочки: **K(2) http://it.kgsu.ru/TI_3/images/ris11_1.giflog2148 http://it.kgsu.ru/TI_3/images/ris11_1.gif7,21**. Поскольку **K(2)** должно быть целым, очевидно, **K(2)= 8**. Именно такой способ кодирования принят в компьютерных системах: любому символу ставится в соответствие цепочка из 8 двоичных разрядов (8 бит). Такая цепочка получила название ***байт***, а представление таким образом символов – ***байтовым кодированием***.

    Байт наряду с битом может использоваться как единица измерения количества информации в сообщении. Один байт соответствует количеству информации в одном символе алфавита при их равновероятном распределении. Этот способ измерения количества информации называется также объемным. Пусть имеется некоторое сообщение (последовательность знаков); оценка количества содержащейся в нем информации согласно рассмотренному ранее вероятностному подходу (с помощью формулы Шеннона ) дает **Iвер**, а объемная мера пусть равна **Iоб**; соотношение между этими величинами:

**Iвер http://it.kgsu.ru/TI_3/images/ris11_2.gifIоб**

    Именно байт принят в качестве единицы измерения количества информации в международной системе единиц СИ. 1 байт = 8 бит. Наряду с байтом для измерения количества информации используются более крупные производные единицы:

***1 Кбайт = 210 байт = 1024 байт (килобайт)***

***1 Мбайт = 220 байт = 1024 Кбайт (мегабайт)***

***1 Гбайт = 230 байт = 1024 Мбайт (гигабайт)***

***1 Тбайт = 240 байт = 1024 Гбайт (терабайт)***

    Использование 8-битных цепочек позволяет закодировать 28=256 символов, что превышает оцененное выше **N** и, следовательно, дает возможность употребить оставшуюся часть кодовой таблицы для представления дополнительных символов.

    Однако недостаточно только условиться об определенной длине кода. Ясно, что способов кодирования, т.е. вариантов сопоставления знакам первичного алфавита восьмибитных цепочек, очень много. По этой причине для совместимости технических устройств и обеспечения возможности обмена информацией между многими потребителями требуется согласование кодов. Подобное согласование осуществляется в форме стандартизации кодовых таблиц. Первым таким международным стандартом, который применялся на больших вычислительных машинах, был **EBCDIC> (Extended Binary Coded Decimal Interchange Code) – *"расширенная двоичная кодировка десятичного кода обмена"***. В персональных компьютерах и телекоммуникационных системах применяется международный байтовый код **ASCII (American Standard Code for Information Interchange – *"американский стандартный код обмена информацией"*)**. Он регламентирует коды первой половины кодовой таблицы (номера кодов от 0 до 127, т.е. первый бит всех кодов 0). В эту часть попадают коды прописных и строчных английских букв, цифры, знаки препинания и математических операций, а также некоторые управляющие коды (номера от 0 до 31). Ниже приведены некоторые **ASCII**-коды:

|  |  |  |
| --- | --- | --- |
| **Знак, клавиша** | **Код двоичный** | **Код десятичный** |
| **пробел** | **00100000** | **32** |
| **A (лат)** | **01000001** | **65** |
| **B (лат)** | **01000010** | **66** |
| **Z** | **01011010** | **90** |
| **0** | **00110000** | **48** |
| **1** | **00110001** | **49** |
| **9** | **00111001** | **57** |
| **Клавиша ESC** | **00011011** | **27** |
| **Клавиша Enter** | **00001101** | **13** |
|  |  |  |

    Вторая часть кодовой таблицы – онасчитается расширением основной – охватывает коды в интервале от 128 до 255 (первый бит всех кодов 1). Она используется для представления символов национальных алфавитов (например, русского или греческого), а также символов псевдографики. Для этой части также имеются стандарты, например, для символов русского языка это ***КОИ–8, КОИ–7*** и др.

    Как в основной таблице, так и в ее расширении коды букв и цифр соответствуют их лексикографическому порядку (т.е. порядку следования в алфавите) – это обеспечивает возможность автоматизации обработки текстов и ускоряет ее.

    В настоящее время появился и находит все более широкое применение еще один международный стандарт кодировки – **Unicode**. Его особенность в том, что в нем использовано 16-битное кодирование, т.е. для представления каждого символа отводится 2 байта. Такая длина кода обеспечивает включения в первичный алфавит 65536 знаков. Это, в свою очередь, позволяет создать и использовать единую для всех распространенных алфавитов кодовую таблицу.

# 2. Кодирование графической информации

**Графическая информация на экране монитора представляется в виде растрового изображения,** которое формируется из определенного количества строк, которые, в свою очередь, содержат определенное количество точек.

Давайте посмотрим на экран компьютера через увелечительное стекло.

В зависимости от марки и модели техники мы увидим либо множество разноцветных прямоугольничков, либо множество разноцветных кружочков.

И те, и другие группируются по три штуки, причем одного цвета, но разных оттенков.

Они называются ПИКСЕЛЯМИ (от английского PICture's ELement).

Пиксели бывают только трех цветов - зеленого, синего и красного.

Другие цвета образовываются при помощи смешения цветов.

Рассмотрим самый простой случай - каждый кусочек пикселя может либо гореть (1), либо не гореть (0).

Тогда мы получаем следующий набор цветов:



Из трех цветов можно получить восемь комбинаций.

Для получения богатой палитры цветов базовым цветам могут быть заданы различные интенсивности, тогда количество различных вариантов их сочетаний, дающих разные краски и оттенки, увеличивается.

Шестнадцатицветная палитра получается при использовании 4-разрядной кодировки пикселя: к трем битам базовых цветов добавляется один бит интенсивности. Этот бит управляет яркостью всех трех цветов одновременно.

Число цветов, воспроизводимых на экране монитора (**N**), и число бит, отводимых в видеопамяти на каждый пиксель (**I**), связаны формулой: **N=2I**

Величину **I** называют битовой глубиной или глубиной цвета.

Чем больше битов используется, тем больше оттенков цветов можно получить.



Итак, любое графическое изображение на экране можно закодировать c помощью чисел, сообщив, сколько в каждом пикселе долей красного, сколько - зеленого, а сколько - синего цветов.

Также **графическая информация может быть представлена в виде векторного изображения.**

Векторное изображение представляет собой графический объект, состоящий из элементарных отрезков и дуг.

Положение этих элементарных объектов определяется координатами точек и длиной радиуса.

Для каждой линии указывается ее тип (сплошная, пунктирная, штрих-пунктирная), толщина и цвет.

Информация о векторном изображении кодируется как обычная буквенно-цифровая и обрабатывается специальными программами.

Качество изображения определяется разрешающей способностью монитора, т.е. количеством точек, из которых оно складывается.

Чем больше разрешающая способность, т.е. чем больше количество строк растра и точек в строке, тем выше качество изображение.

**Пример 1.** Рисунок построен с использованием палитры 256 цветов на экране монитора с графическим разрешением 1024 x 768. Рассчитать объем памяти необходимый для хранения этого рисунка.

Решение: 256=2I  I=8

 V=1024 \* 768 \* 8 бит = 1024 \* 768 байт = 768 Кбайт.

Ответ: 768 Кбайт.

**Пример 2.** Рассчитайте объем памяти, необходимый для хранения рисунка построенного при графическом разрешении монитора 800 x 600 с палитрой 32 цвета.

Решение

800 \* 600 \*5 бит = 100 \* 3000 байт http://ivanovff.21419s01.edusite.ru/uchin/images/clip_image003.gif300 Кбайт

**Пример 3.** Каков информационный объем книги, если в ней 200 страниц текста (на каждой странице 50 строк по 80 символов) и 10 цветных рисунков. Каждый рисунок построен при графическом разрешении монитора 800 x 600 с палитрой 16 цветов.

Решение:

1 символ – 1 байт

80 \* 50 \* 200 байт http://ivanovff.21419s01.edusite.ru/uchin/images/clip_image003.gif80 \* 10 Кбайт = 800 Кбайт – для хранения текста

16 = 24

10 \* (800 \* 600 \* 4) бит = 10 \* 100 \* 2400 байт http://ivanovff.21419s01.edusite.ru/uchin/images/clip_image003.gif2400 Кбайт для хранения рисунков

2400 + 800 = 3200 Кбайт http://ivanovff.21419s01.edusite.ru/uchin/images/clip_image003.gif3, 2 Мб

Ответ: http://ivanovff.21419s01.edusite.ru/uchin/images/clip_image003.gif3, 2 Мб

1. **Двоичное кодирование звуковой информации**

**Временная дискретизация звука.** Звук представляет собой звуковую волну с непрерывно меняющейся амплитудой и частотой. Чем больше амплитуда сигнала, тем он громче для человека, чем больше частота сигнала, тем выше тон. Для того чтобы компьютер мог обрабатывать звук, непрерывный звуковой сигнал должен быть превращен в последовательность электрических импульсов (двоичных нулей и единиц).

В процессе кодирования непрерывного звукового сигнала производится его *временная дискретизация*. Непрерывная звуковая волна разбивается на отдельные маленькие временные участки, причем для каждого такого участка устанавливается определенная величина амплитуды.

Таким образом, непрерывная зависимость амплитуды сигнала от времени A(t) заменяется на дискретную последовательность уровней громкости. На графике это выглядит как замена гладкой кривой на последовательность "ступенек" - рис. 1.9.

|  |
| --- |
| http://www.5byte.ru/11/images/inf23.gif |
| Рис. 1.9 Временная дискретизация звука |

Каждой "ступеньке" присваивается значение уровня громкости звука, его код (1, 2, 3 и так далее). Уровни громкости звука можно рассматривать как набор возможных состояний, соответственно, чем большее количество уровней громкости будет выделено в процессе кодирования, тем большее количество информации будет нести значение каждого уровня и тем более качественным будет звучание.

Современные звуковые карты обеспечивают 16-битную глубину кодирования звука. Количество различных уровней сигнала (состояний при данном кодировании) можно рассчитать по формуле (2.1):

N = 2I = 216 = 65536, где I - глубина звука.

Таким образом, современные звуковые карты могут обеспечить кодирование 65536 уровней сигнала. Каждому значению амплитуды звукового сигнала присваивается 16-битный код.

При двоичном кодировании непрерывного звукового сигнала он заменяется последовательностью дискретных уровней сигнала. Качество кодирования зависит от количества измерений уровня сигнала в единицу времени, то есть *частоты дискретизации*. Чем большее количество измерений производится за 1 секунду (чем больше частота дискретизации), тем точнее процедура двоичного кодирования.

*Качество двоичного кодирования звука определяется* ***глубиной кодирования*** *и* ***частотой дискретизации****.*

Количество измерений в секунду может лежать в диапазоне от 8000 до 48 000, то есть частота дискретизации аналогового звукового сигнала может принимать значения от 8 до 48 кГц. При частоте 8 кГц качество дискретизированного звукового сигнала соответствует качеству радиотрансляции, а при частоте 48 кГц - качеству звучания аудио-CD. Следует также учитывать, что возможны как моно-, так и стерео-режимы.

Можно оценить информационный объем стереоаудиофай-ла длительностью звучания 1 секунда при высоком качестве звука (16 битов, 48 кГц). Для этого количество битов, приходящихся на одну выборку, необходимо умножить на количество выборок в 1 секунду и умножить на 2 (стерео):

16 бит × 48 000 × 2 = 1 536 000 бит = 192 000 байт = 187,5 Кбайт.

Стандартное приложение Звукозапись играет роль цифрового магнитофона и позволяет записывать звук, то есть дискретизировать звуковые сигналы, и сохранять их в звуковых файлах в формате WAV. Эта программа позволяет редактировать звуковые файлы, микшировать их (накладывать друг на друга), а также воспроизводить.

1. **Двоичное кодирование звуковой информации**

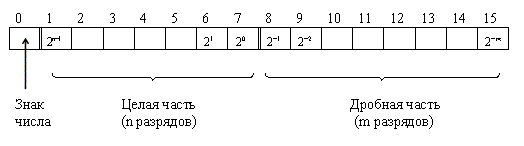
При проектировании ЭВМ, создании инструментального и прикладного программного обеспечения разработчикам приходится решать вопрос о представлении в ЭВМ числовых данных. Для решения большинства прикладных задач обычно достаточно использовать целые и вещественные числа. Запись целочисленных данных в запоминающем устройстве ЭВМ не представляет затруднений: число переводится в двоичную систему и записывается в прямом коде. Диапазон представляемых чисел в этом случае ограничивается количеством выделенных для записи разрядов. Для вещественных данных обычно используются две формы записи: число с фиксированной точкой (ЧФТ) и число с плавающей точкой (ЧПТ).

3.1. Числа с фиксированной точкой.

Форма записи числа с фиксированной точкой использовалась в основном на ранних этапах развития вычислительной техники. Запись числа с фиксированной точкой обычно имеет знаковый и цифровой разряды. Фиксированная точка означает, что на этапе конструирования ЭВМ было определено, сколько и какие разряды машинного слова отведены под изображение целой и дробной частей числа. Запятая в разрядной сетке может быть зафиксирована, в принципе, после любого разряда.

Пример.

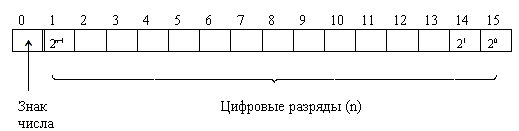
Ячейка с целой и дробной частью.



Как частный случай числа с фиксированной точкой может быть рассмотрена запись целого числа (в этом случае все разряды, кроме знакового, используются для записи целой части).

Пример.

Ячейка с записью целого числа.



К достоинствам использования чисел с фиксированной точкой относятся простота выполнения арифметических операций и высокая точность изображения чисел. К недостаткам - небольшой диапазон представления чисел.

3.2 Числа с плавающей точкой.

Для представления чисел с плавающей точкой (ЧПТ) используется полулогарифмическая форма записи числа:

*N = ± mq± p*

где *q*- основание системы счисления,

*p*- порядок числа,

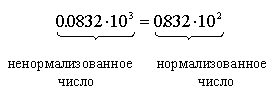
*m*- мантисса числа*N*.

Положение точки определяется значением порядка  *p*. С изменением порядка точка перемещается (плавает) влево или вправо. Пример.

12510=12.5\*101=1.25\*102=0.125\*103=0.0125\*104=...

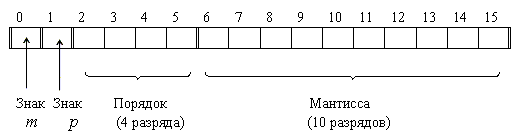
Для установления однозначности при записи чисел принята **нормализованная форма**записи числа. Мантисса нормализованного числа может изменяться в диапазоне:  1/q ≤ | m | < 1. Таким образом в нормализованных числах цифра после точки должна быть значащей.

Пример.

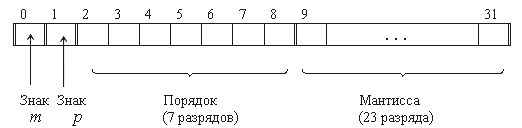


Для представления чисел в машинном слове выделяют группы разрядов для изображения мантиссы, порядка, знака числа и знака порядка:

а) представление чисел в формате полуслова

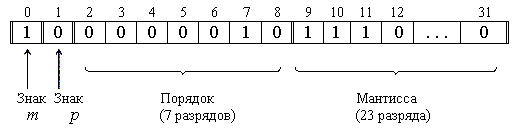


б) представление чисел в формате слова

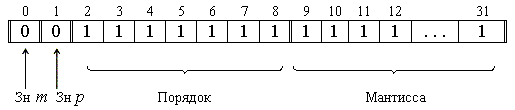


Наиболее типично представление ЧПТ в формате слова (32 разряда). Пример.

Число А=-3.510=-11.12=-0.111·1010



Максимальным числом представимым в формате слова будет A=(0.1111...1·101111111)2https://studfile.net/html/2706/262/html_fzJv0XOL_P.Hbmn/img-Rpu9Y3.png(1·2127)10.



Таким образом числа с плавающей точкой позволяют увеличить диапазон обрабатываемых чисел, но при этом точность изображения чисел определяется только разрядами мантиссы и уменьшается по сравнению с числами с фиксированной точкой. При записи числа в формате слова диапазон представимых чисел будет от -1·2127до 1·2127(2127https://studfile.net/html/2706/262/html_fzJv0XOL_P.Hbmn/img-U_jG5q.png1038), а точность определяться мантиссой, состоящей из 23 разрядов. Точность может быть повышена путем увеличения количества разрядов мантиссы. Это реализуется путем представления чисел с так называемой двойной точностью (используется формат двойного слова):

