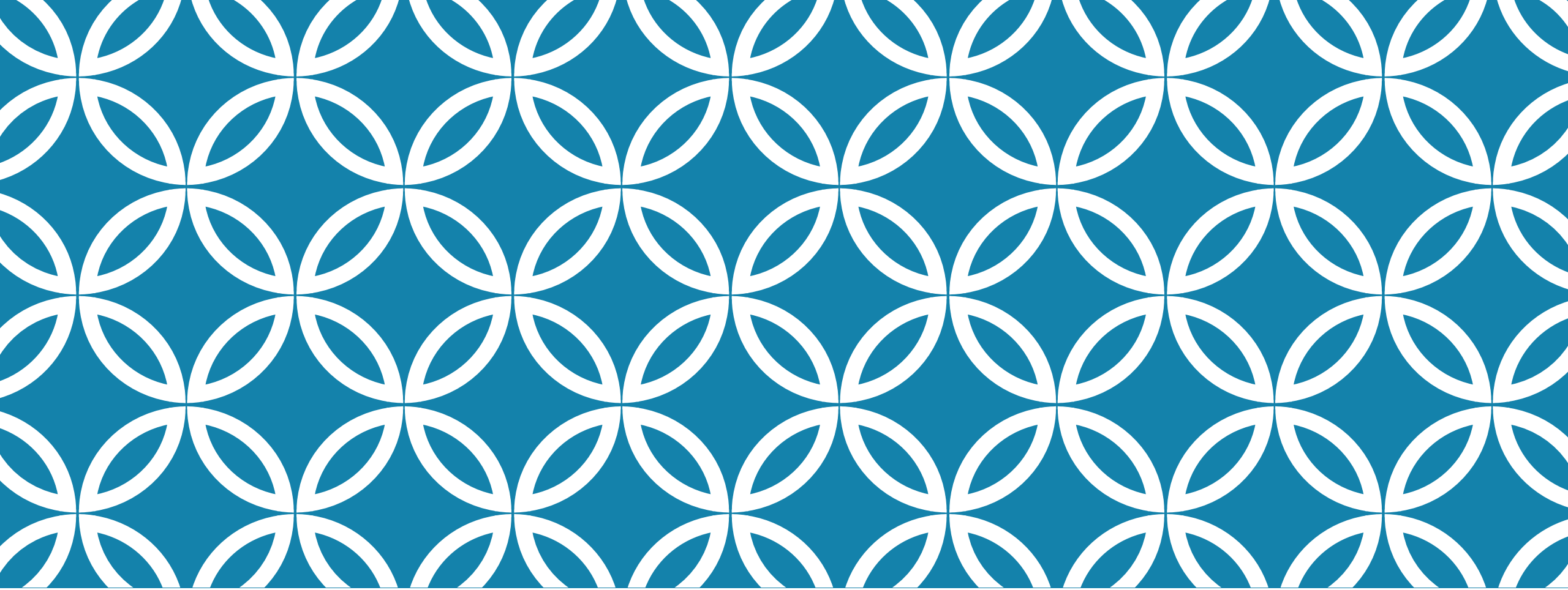


ПЕРЕДАЧА ИНФОРМАЦИИ

Источник связывает информацию с ее материальным носителем. Источник, порождающий информацию, для передачи должен представить ее в виде сообщения, т.е. последовательности сигналов. При этом для представления информации он должен использовать некоторую систему кодирования. Устройство, выполняющее операцию кодирования информации, может являться подсистемой источника .

Рис. 5.1. Общая схема передачи информации

После преобразователя сигналы поступают и распространяются по *каналу связи*. Понятие канала связи включает в себя *материальную среду, а* также *физический* или иной *процесс,* посредством которого осуществляется передача сообщения, т.е. распространение сигналов в пространстве с течением времени. Ниже приведены примеры некоторых каналов связи.

Канал связи	Среда	Носитель сообщения	Процесс, используемый для передачи сообщения
Почта, курьеры	Среда обитания человека	Бумага	Механическое перемещение носителя
Телефон, компьютерные сети	Проводник	Электрический ток	Перемещение электрических зарядов
Радио, телевидение	Электромагнитное поле	Электромагнитные волны	Распространение электромагнитных волн
Зрение	Электромагнитное поле	Световые волны	Распространение световых волн
Слух	Воздух	Звуковые волны	Распространение звуковых волн
Обоняние, вкус	Воздух, пища	Химические вещества	Химические реакции
Осязание	Поверхность кожи	Объект, воздействующий на органы осязания	Теплопередача, давление

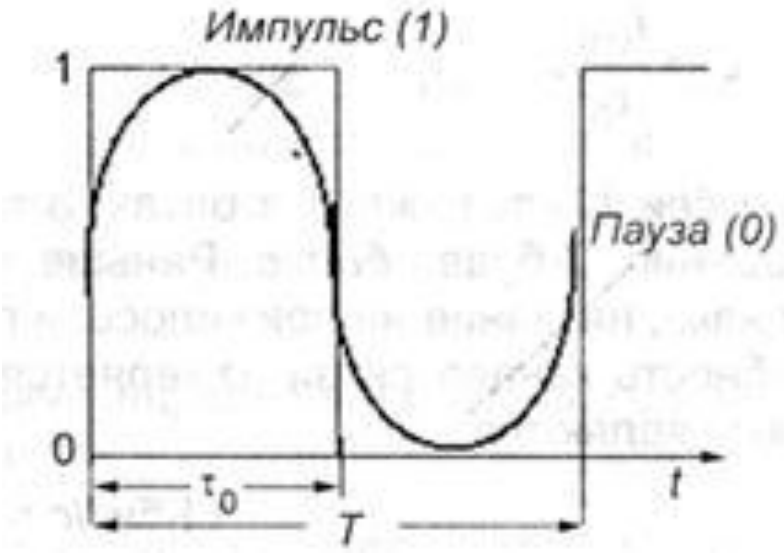
Любой *реальный* канал связи подвержен внешним воздействиям, а также в нем могут происходить внутренние процессы, в результате которых искажаются передаваемые сигналы и, следовательно, связанное с ними сообщение. Такие воздействия называются ***шумами*** (**помехами**).

После прохождения сообщения по каналу связи сигналы с помощью приемного преобразователя переводятся в последовательность кодов, которые декодирующим устройством представляются в форме, необходимой приемнику информации. На этапе приема, как и при передаче, преобразователь может быть совмещен с декодирующим устройством (например, радиоприемник или телевизор) или существовать самостоятельно (например, модем).

Характеристиками любой линии связи являются скорость, с которой возможна передача сообщения в ней, а также степень искажения сообщения в процессе передачи.

Интервал частот, используемый данным каналом связи для передачи сигналов, называется ***шириной полосы пропускания.***

Длительность элементарного импульса может быть определена из следующих соображений. Если параметр сигнала меняется синусоидально, то, как видно из рисунка, за один период колебания *T* сигнал будет иметь одно максимальное значение и одно минимальное.



Если аппроксимировать синусоиду прямоугольными импульсами и сместить начало отсчета на уровень минимального значения, получатся, что сигнал принимает всего два значения - максимальное (обозначим его «1») - *импульс,* и минимальное (можно обозначить «0») - *пауза*. Импульс и паузу можно считать элементарными сигналами; при выбранной аппроксимации их длительности, очевидно одинаковы и равны:

$$\tau_0 = \frac{T}{2} = \frac{1}{2\nu_m}. \qquad (5.1)$$

Если же импульсы порождаются тактовым генератором, имеющим частоту ν_m, то

$$\tau_0 = \frac{1}{\nu_m}. \qquad (5.2)$$

ν_m определяет *длительность элементарного сигнала* τ_0, используемого для передачи сообщения.

Если с передачей одного импульса связано количество информации I_{imp}, а передается оно за время τ_0, отношение Ii_{mp} к τ_0, очевидно, будет отражать *среднее количество информации, передаваемое по каналу за единицу времени* - эта величина является характеристикой канала связи и называется *пропускной способностью канала C*:

$$C = \frac{I_{imp}}{\tau_0}. \qquad (5.3)$$

Если I_{imp} выражено в битах, а τ_0 - в секундах, то единицей измерения *C* будет *бит/с*. Раньше такая единица называлась *бод*, однако, название не прижилось, и по этой причине пропускная способность канала связи измеряется в бит/с. Производными единицами являются:

1 Кбит/с = 10^3 бит/с,
1 Мбит/с = 10^6 бит/с,
1 Гбит/с = 10^9 бит/с.

Величину I_{imp} в (5.3) можно установить из следующих соображений: если первичный алфавит содержит *N* знаков с вероятностями появления их в сообщении p_i, то по формуле Шеннона (2.17) можно найти среднюю информацию на знак первичного алфавита I_1, для представления которого используется двоичный код длиной $K^{(2)}$, то, очевидно:

$$I_{imp} = \frac{I_1}{K^{(2)}}.$$

При подстановке в (5.3) получаем: $$C = \frac{I_1}{K^{(2)} \cdot \tau_0}. \qquad (5.4)$$

Пусть по каналу связи за время *t* передано количество информации *I*. Можно ввести величину, характеризующую быстроту передачи информации - *скорость передачи информации J*:

$$J = \frac{I}{t}.$$

Размерностью *J*, как и *C*, является бит/с. Поскольку τ_0 - минимальная длительность элементарного сигнала, очевидно, *C* соответствует *максимальной скорости передачи информации* по данной линии связи, т.е. $J \leq C$ или $C = J_{max}$.

Максимальная скорость передачи информации по каналу связи равна его пропускной способности.

Отличие реального канала от идеального в том, что шумы приводят к *снижению пропускной способности канала*. Покажем это для частного случая использования двух элементарных сигналов равной длительности. Каждый из них на входе канала связи несет $I_{imp} = 1$ бит информации. После прохождения сигнала по идеальному каналу и на выходе импульс (1) интерпретируется именно как импульс, а пауза (0) - как пауза и, связанная с ними информация не меняется в количественном отношении.

Иная ситуация в реальном канале: из-за шумов при передаче может произойти искажение сигнала, в результате чего вместо 1 на выходе будет получен 0, а вместо 0 - 1. Пусть вероятности таких искажений для 0 и 1 одинаковы, т.е. $p_{1\to0} = p_{0\to1} = p$. Тогда вероятность того, что исходный сигнал придет без искажений, очевидно, равна $1 - p$.

Следовательно, при интерпретации (распознавании) конечного сигнала возникает *неопределенность,* которая, как следует из формул (2.4) и (А.8), $H = -p\log_2 p - (1-p)\log_2(1-p)$. ей энтропией:

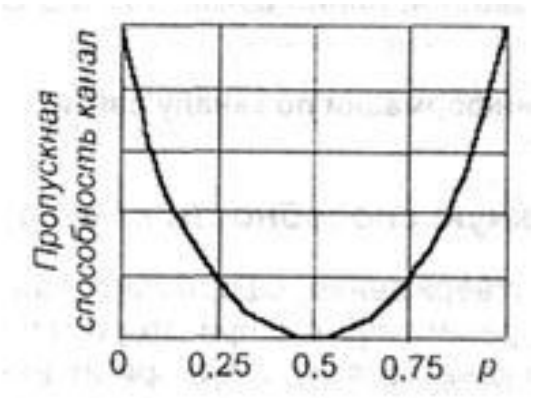
Эта неопределенность приведет к *уменьшению* количества информации, содержащейся в сигнале, на величину *H,* т.е.

$$(I_{imp})' = I_{imp} - H.$$

Поскольку длительность импульса τ_0 определяется частотой ν_m и не зависит от наличия шумов, пропускная способность реального канала C_R оказывается меньше, чем аналогичного идеального *C*:

$$C_R = \frac{(I_{imp})'}{\tau_0} = \frac{I_{imp} + p \cdot \log_2 p + (1-p) \cdot \log_2 (1-p)}{\tau_0} \le C. \qquad (5.5)$$

На графике показана зависимость $C_R(p)$. Как следует из (5.5), при $p = 0{,}5$ $(I_{\text{iтр}})' = 0$ и, следовательно, $C_R = 0$.



Влияние шумов определяется соотношением мощности сигнала и мощности помех

$$C_R = \nu_m \cdot \log_2\left(1 + \frac{N_S}{N_n}\right),$$

Ns - средняя мощность сигнала, а Nn- средняя мощность по- мех. Из этого соотношения, в частности, видно, что для увеличения пропускной способности канала связи необходимо увеличивать полосу пропускания, либо улучшать отношение мощности сигнала к мощности помех.

Приведем характеристики некоторых каналов связи.

Вид связи	ν_m, Гц	N_s/N_n	C_R, бит/с
Телеграф	120	$\approx 2^6$	640
Телефон	$3 \cdot 10^3$	$\approx 2^{17}$	$5 \cdot 10^4$
Телевидение	$7 \cdot 10^6$	$\approx 2^{17}$	$130 \cdot 10^6$
Компьютерная сеть			до 10^9
Слух человека	$20 \cdot 10^3$		$5 \cdot 10^4$
Глаза человека			$5 \cdot 10^6$

Вторая теорема Шеннона относится к реальным каналам связи и гласит следующее:

> При передаче информации по каналу с шумом всегда имеется способ кодирования, при котором сообщение будет передаваться со сколь угодно высокой достоверностью, если скорость передачи не превышает пропускной способности канала.

В любых случаях при превышении скорости передачи пропускной способности канала возможна потеря информации. Смысл данной теоремы в том, что при передаче по реальным каналам можно закодировать сообщение таким образом, что действие шумов не приведет к потере информации. Это достигается за счет повышения избыточности кода (т.е. увеличения длины кодовой цепочки); безусловно, возрастает время передачи, что следует считать платой за надежность.

Относительная избыточность сообщения - это характеристика, показывающая, во сколько раз требуется удлинить сообщение, чтобы обеспечить его надежную (безошибочную) передачу (хранение).

Наличие шумов в канале связи ведет к частичной потере передаваемой информации на величину возникающей неопределенности, которая при передаче одного бита исходного сообщения составляет:

$$H = -p \cdot \log_2 p - (1 - p) \cdot \log_2 (1 - p),$$

где p - вероятность появления ошибки в сообщении. Для восстановления информационного содержания сообщения, очевидно, следует дополнительно передать количество информации не менее величины ее потерь, т.е. вместо передачи каждого 1 бит информации следует передавать $1 + H$, бит. В этом случае избыточность сообщения составит

$$L_{min} = \frac{1+H}{1} = 1 - p \cdot \log_2 p - (1-p) \cdot \log_2 (1-p). \qquad (5.7)$$

Приведенную избыточность следует считать *минимальной* (это указывает ее индекс), поскольку при передаче сообщения по каналу, характеризуемому вероятностью искажения p, при избыточности, меньшей L_{min} восстановление информации оказывается невозможным.

В компьютерных линиях связи используются два способа передачи:

- параллельный, когда передаются одновременно все биты машинного слова;
- последовательный, когда биты передаются поочередно, начиная с младшего.

Для одновременной передачи нескольких сигналов требуется линия связи, количество проводников в которой совпадает с числом передаваемых сигналов. Такая линия связи называется **шиной**.

Количество проводников определяет ширину или разрядность шины. Например, во внутренних линиях компьютера могут использоваться 16-ти и 32-х разрядные шины. На рис. 5.2. показана линия параллельной передачи, связывающая регистр АЛУ и ячейку памяти компьютера

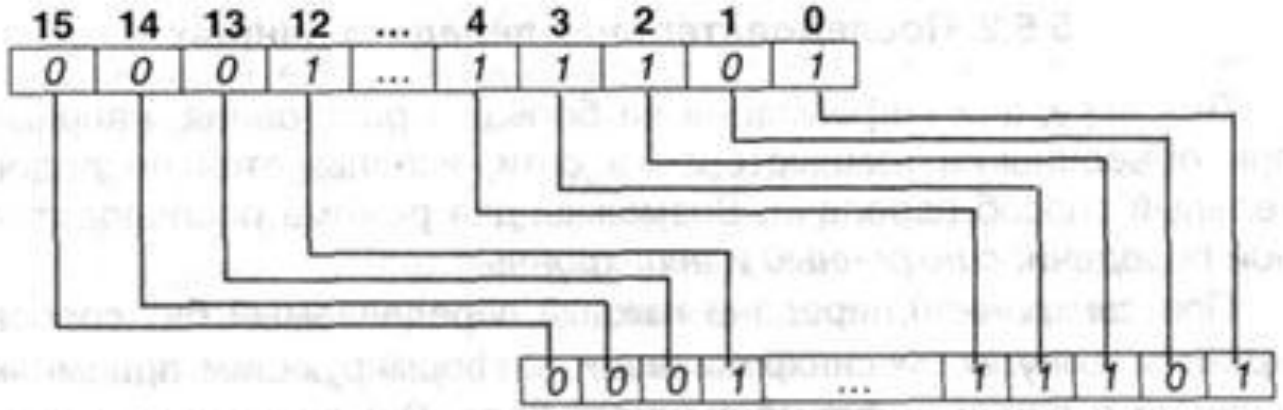


Рис.5.2. Параллельный способ передачи данных в компьютере

Шина обеспечивает наиболее быстрый способ передачи информации, поскольку за два такта синхрогенератора компьютера передается целое машинное слово.

В общем случае, если **Vm** - частота генератора, **h** - разрядность шины, а **п** - число тактов, за которые осуществляется передача, пропускная способность канала параллельной передачи С будет равна:

$$C = \frac{\nu_m \cdot h}{n}. \qquad (5.8)$$

Параллельный способ передачи используется во внутренних линиях связи, а также для связи с внешними устройствами, подключаемыми к параллельному порту компьютера (LPT-порту): принтером, плоттером (графопостроителем) и др.

Данный способ использовался и при объединении компьютеров в локальные сети, например, в отечественном КУВТ-86, которым раньше оснащались школы.

К **недостаткам** параллельного способа передачи относят, во-первых, то, что им невозможна передача на большие расстояния (более нескольких метров), поскольку между параллельными проводниками имеется электроемкость, увеличивающаяся с их длиной, существование которой приводит к тому, что при протекании импульса по какому-либо одному проводнику возникают наводки в других. Во-вторых, данный способ требует многожильных специальных проводов для связи, что существенно повышает стоимость линии.

Для передачи информации на большие расстояния используется последовательный способ передачи. Возможны два режима последовательной передачи: *синхронный* и *асинхронный*.

При ***синхронной*** **передаче** каждый передаваемый бит сопровождается импульсом синхронизации, информирующим приемник о наличии в линии информационного бита. Синхронизирующие импульсы управляют приемом информации. Между передатчиком и приемником должны быть протянуты минимум три провода: один - для передачи данных, второй - для передачи синхроимпульсов, третий - общий заземленный. Если же расстояние между источником и приемником составляет несколько метров, то каждый из сигналов приходится посылать по экранированному кабелю, что увеличивает стоимость линии связи. Кроме того, такая передача оказывается целесообразной, если передается некоторый массив символов. Оба перечисленных обстоятельства приводят к тому, что синхронный способ связи не получил широкого распространения.

Асинхронный **способ** передачи не требует синхронизации действий приемника и передатчика; по этой причине для связи достаточно линии из двух проводников, причем, оказывается возможным использование даже телефонных линий, т.е. неспециализированных компьютерных. При этом источник и приемник информации должны быть согласованы по формату и скорости передачи.

Передача производится машинными словами (информационными битами), дополненными несколькими служебными. Рассмотрим пример передачи 8-битного слова с одним контрольным битом. В отсутствии передачи в линии поддерживается уровень сигнала, соответствующий логической единице (например, +5 В).

Передатчик может начать пересылку в любой момент посредством генерации ***стартового бита***, который переводит линию в состояние логического нуля на время продолжительности элементарного сигнала τ_0 - по нему приемник узнает, что передача началась.

Затем происходит передача информационных битов, начиная с младшего (0-го). За ними передается контрольный бит четности. Наконец, за ним следует столовый *бит* (их может быть два), который вновь переводит линию в состояние ожидания, т.е. логической единице. Вся передаваемая цепочка сигналов от стартового до стопового бита называется ***кадром***.

Передача следующего кадра может начаться сразу после стопового бита, причем, новый стартовый бит может быть послан в любой момент времени.

Применение телефонных линий или радиоканала для осуществления связи между удаленными на большие расстояния компьютерами диктуется экономическими соображениями. Использование двухпроводных линий и большие расстояния однозначно определяют способ передачи - **последовательный**.

Однако телефонные линии предназначены для передачи аналоговых электрических сигналов с частотой человеческого голоса (как указывалось выше, верхняя граница составляет 3400 Гц). При передаче по таким линиям сигналов с частотами, на которых работает компьютер (100-500 МГц), они будут испытывать значительные искажения и быстро затухать. По этой причине передача осуществляется в соответствии со схемой, представленной на рис. 5.3.

Рис. 5.3. Компьютерная линия связи с применением модемов

На передающем конце линии связи сначала осуществляется преобразование параллельного компьютерного кода в асинхронный последовательный. Затем посредством другого устройства - ***модема*** -производится ***модуляция*** двухуровневых импульсных компьютерных сигналов в аналоговые сигналы, которые без большого затухания и искажения могут распространяться в телефонных линиях.

На приемном конце происходит обратная цепочка: модемом аналоговый сигнал переводится в двухуровневый последовательный код (происходит демодуляция), у которого затем удаляются все неинформационные биты и он преобразуется в параллельный.

Асинхронный преобразователь располагается в самом компьютере в виде блока, осуществляющего обмен последовательным способом с любым устройством - этот блок называется ***последовательным портом*** (COM-портом). Помимо модема к нему присоединяются также внешние манипуляторы (мышь, джойстик, дигитайзер). В модеме может использоваться амплитудная, частотная или фазовая модуляция электрических аналоговых сигналов с несущей частотой до 3000 Гц.

Для увеличения скорости (в современных модемах до 56 Кбит/с) передачи применяют более сложные методы модуляции, при которых каждый передаваемый элементарный сигнал кодирует не один, а несколько бит.

При передаче данных по последовательным линиям связи возможны три режима: ***симплексный***, ***полудуплексный*** и ***дуплексный***.

Симплексная линия обеспечивает передачу только в одном направлении, например, от датчика к устройству обработки.

Полудуплексная связь обеспечивает передачу и получение информации в обоих направлениях, но не одновременно - передает один модем, другой в это время получает, затем они могут поменяться ролями.

Дуплексная связь обеспечивает передачу и получение данных в обоих направлениях одновременно. Поскольку связь осуществляется по двухпроводной телефонной линии, приходится использовать вторую пару частот для связи в обратном направлении, например, 2025 Гц для представления нуля и 2225 Гц для представления единицы.

ГЛАВА 6. ХРАНЕНИЕ ИНФОРМАЦИИ

Данные - это сведения, характеризующие какую-то систему, явление, процесс или объект, представленные в определенной форме и предназначенные для дальнейшего использования.

Данные - это конкретная форма представления содержания информации (например, информацию о результатах наблюдения за температурой окружающей среды можно представить в виде числового массива (таблицы), но можно и в виде графика, и в виде текстового описания посредством некоторого языка);

• в отличие от ненаправленной (неадресной, рассеянной) информации, существующей в природе независимо от нас и наших потребностей в ней, **данными** называется только такая информация, которая имеет значение для потребителя и, следователь- но, предусматривается ее использование для решения каких- либо задач; другими словами, практический статус и важность данных выше, нежели у природной информации.

Данным приписываются несколько классификационных при-знаков. Важнейшим из них является тип данных.

Тип данных определяет:

- набор их допустимых значений;
- правила их обработки (преобразования);
- порядок их размещения в ОЗУ и ВЗУ при хранении;
- порядок доступа к ним (т.е. обращение и извлечение при необходимости с места хранения)

Допустимый набор типов данных и их особенности определяются программной системой либо языком программирования, на котором система написана. При всем этом способности языков по обилию допустимых типов данных, также построению новых типов различаются очень.

Чем более широкой и гибкой оказывается типизация данных в программной системе либо языке, тем больше способностей предоставляется юзеру в решении задачки рационального представления, хранения и внедрения данных.

Типизация данных *оказывает влияние и на компактность самой исполняемой программки*. К примеру, в языке BASIC отсутствует тип данных *«записи»;* в итоге для сотворения и использования базы данных пришлось бы организовывать параллельную обработку нескольких массивов.

Последующим признаком является деление данных на *простые* (одиночные, обыкновенные) и *структурированные* (сложные).

К ***простым данным*** относятся знаки, числа (целые и вещественные) и логические данные. Общей и неотклонимой особенностью одиночных данных будет то, каждое из их имеет одно *значение* и *собственное имя*.

Значение — это содержимое тех ячеек памяти, где данное размещается. *Имя* (его именуют также *идентификатор*) — **это обозначение данного в тексте программки.** Правила построения идентификаторов простых данных определяются языком программирования написанной программки.

Информационный массив, объединяющий данные и связи (дела) меж ними именуется **структурированными данными.**

Список объединяемых одиночных данных, их свойства, также особенности связей меж ними образуют ***структуру данных*****.** писок допустимых структур данных, как уже было сказано, определяется языком программирования либо прикладной программкой.

Сложные данные, как и простые, имеют *значения* и идентификаторы. Значения располагаются в ячейках ОЗУ по определенным схемам .Правила построения идентификаторов инсталлируются языком программирования либо программной системой. Исключение составляют правила формирования названий файлов — **они задаются операционной системой и должны соблюдаться всеми работающими в ней программками и языками.**

По способности конфигурации значений данных (как обычных, так и структурированных) в процессе общей обработки их подразделяют на *переменные* и *неизменные (константы)*. Из наименования разумеется, что *переменные* могут изменять свое значение по ходу выполнения программки, а *константы* — нет.

Зависимо от того, на каком шаге обработки данные употребляются, они разделяются на ***начальные*** (***входные***), *промежуточные* и *выходные*.

К ***начальным*** относятся данные, нужные для выполнения программки и вводимые в нее до либо в процессе работы. Начальные данные могут быть за ранее записаны на неком носителе и вводиться с него, поступать по линиям связи от каких-либо датчиков либо с других компов, вводиться юзером программки средством устройств ввода.

Промежуточные данные формируются в процессе выполнения программки и, в большинстве случаев, юзеру недосягаемы; они не показываются на устройствах вывода, но есть в ОЗУ либо на ВЗУ. Идентификаторы промежуточным данным присваивает разработчик программки либо задает сама программка по заложенным в нее правилам.

Выходные данные являются результатом работы программки — ради их и делается обработка входных. Выходные данные, созданные для человека, представляются в требуемой для него форме (тексты, картинки, звуки); при хранении выходных данных на носителях либо передаче по сетям сохраняется двоичный компьютерный формат их представления

Представление данных при их хранении и обработке просит **решения 3-х главных задач:**

- найти методы представления простых (обычных) данных;
- найти методы объединения данных в структуры;
- установить методы размещения инфы на вещественном носителе.

Выделяют три *уровня* представления данных — ***концептуальный, логический и физический.***

На ***концептуальном*** уровне определяется общая структура информационного массива — она именуется моделью *данных*. Известны и употребляются несколько моделей данных: *иерархическая, сетевая, реляционная, объектно-ориентированная*. В согласовании с избранной моделью данных строится информационная система, в какой данные будут храниться, также программки, ведущие их обработку *(манипулирование данными)*.

Логический **уровень** определяет методы представления простых данных, их список при объединении в структуру, также нрав связей меж ними в рамках избранной модели данных.

Физический **уровень** определяет форматы размещения сделанной логической структуры данных на наружных носителях инфы (**магнитных либо оптических дисках, бумаге, в памяти компьютера**). Представление данных является принципиальным фактором, обеспечивающим малогабаритный (т.е. экономичный исходя из убеждений расходования носителя) метод записи инфы при хранении и резвый доступ к необходимым данным при их использовании. Дальше подвергнутся рассмотрению варианты решения перечисленных задач в компьютерных системах.

Для представления *значений* элементарных данных в памяти компьютера используется *машинное слово;* этот термин в информатике применяется в двух значениях:

Машинное слово - *(1) совокупность двоичных элементов, обрабатываемая как единое целое в устройствах и памяти компьютера;*

(2) данные, содержащиеся в одной ячейке памяти компьютера.

С технической точки зрения машинное слово объединяет запоминающие элементы, служащие для записи 1 бит информации, в единую *ячейку памяти*. Количество таких объединяемых элементов кратно 8, т.е. целому числу байт. Доступ к машинному слову в операциях записи и считывания осуществляется по номеру ячейки памяти, который называется *адресом ячейки.*

Запоминающие устройства, в которых доступ к данным осуществляется по адресу ячейки, где они хранятся, называются устройствами с ***произвольным доступом.***

Для логического уровня важно то, что представление значений любых элементарных данных должно быть ориентировано на использование машинных слов определенной и единой для данного компьютера длины, поскольку их представление на физическом уровне производится именно в ячейках ОЗУ .

Рассмотрим особенности представления всех типов элементарных данных с помощью 16-
битного машинного слова. Порядок размещения значений данных представлен на рис.6.1.

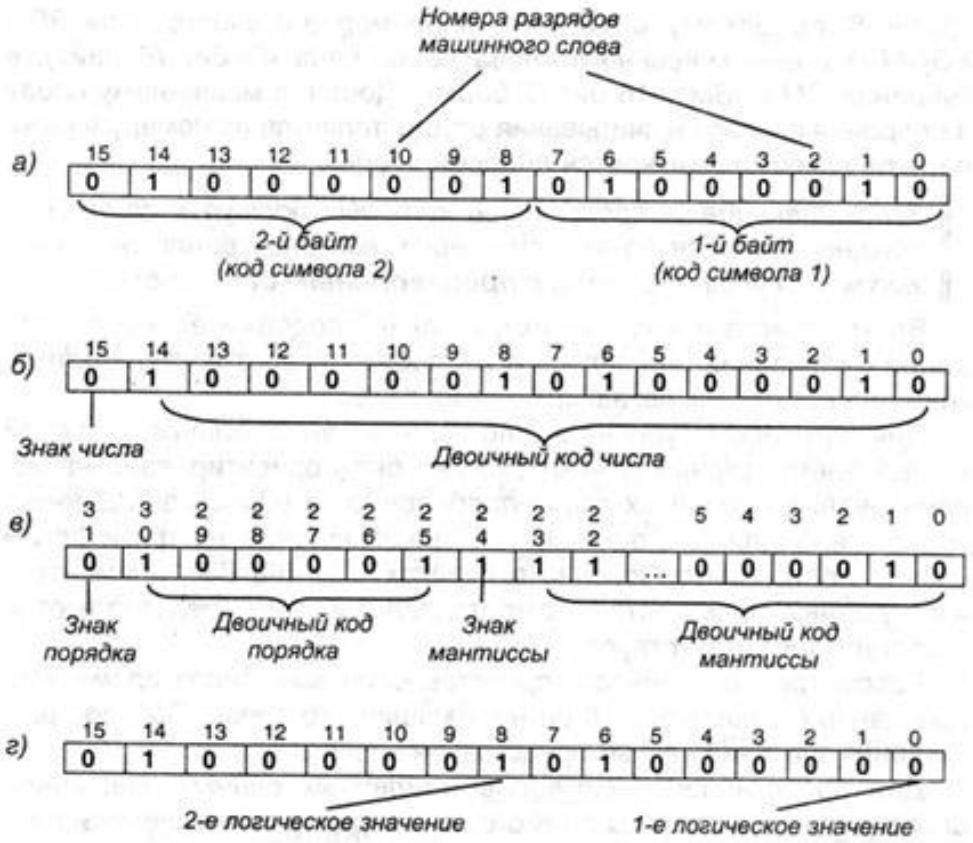


Рис. 6.1. Представление значений элементарных данных с помощью 16-битных машинных слов:
а – символы; *б* – целые числа со знаком; *в* – вещественные числа с плавающей запятой; *г* – логические данные

Для представления символов (литерных данных) машинное слово делится на группы по 8 бит, в которые и записываются двоичные коды символов. Ясно, что в 16-битном машинном слове могут быть записаны одновременно два символа (рис. 6.1,а). Значениями одиночных литерных данных являются коды символов. Множество допустимых значений данных этого типа для всех кодировок, основанных на однобайтовом представлении, составляет 2^8 - 256; двубайтовая кодировка Unicode допускает 65536 значений.

***Совокупность символов образует алфавит**, т.е. для них установлен лексикографический порядок следования в соответствии с числовым значением кода; это, в свою очередь, позволяет определить над множеством символьных данных операции математических отношений >, <, =.*

Непосредственно над одиночной символьной переменной определена единственная операция - изменение значения с одного кода на другой. Все остальные действия производятся со сложными символьными данными, например, типа String в языке PASCAL (см. п.6.3.).

В <u>представлении целых чисел со знаком</u> *(например, тип Integer в языке PASCAL) старший бит (15-й), как уже обсуждалось ранее, отводится под запись знака числа (0 соответствует «+», 1 - «-»), а остальные 15 двоичных разрядов - под запись прямого (для положительного) или обратного (для отрицательного) двоичного кода числа*

При этом возможные значения чисел ограничены интервалом [-32768 + 32767].

*Наряду с описанным используется и другой формат представления целых чисел - беззнаковый; очевидно, он применим только для записи положительных чисел. В этом случае под запись числа отводятся все 16 двоичных разрядов, и интервал разрешенных значений оказывается [0 * 65535] (в PASCAL'e такой числовой тип называется Word).*

Помимо математических отношений над целыми числами определены операции сложения, вычитания и умножения (в тех случаях, когда они не приводят к переполнению разрядной сетки), а также целочисленного деления и нахождения остатка от целочисленного деления.

<u>Представление вещественных чисел</u> *с плавающей запятой также рассматривалось ранее. При записи числа оно переводится в нормализованную форму с выделением и отдельным хранением знака мантиссы, знака порядка, порядка и мантиссы. Для представления числа отводится несколько машинных слов. ».*

Благодаря применению плавающей запятой производится автоматическое масштабирование чисел в ходе вычислений, что снижает погрешность их обработки. Над вещественными числами определены все четыре арифметические операции. Помимо этого имеются операции преобразования вещественного типа к целому

Логические данные *могут принимать одно из двух значений - 0 или 1 (0 соответствует логическому False, 1 - True, причем, принимается False<True). Для их записи было бы достаточно отвести всего один двоичный разряд. Однако в ОЗУ компьютера отсутствует доступ к отдельному биту, поэтому для представления логических данных выделяется целый байт, в младший разряд которого и помещается значение.*

Таким образом, в машинном слове логические данные располагаются в 0-м и в 8-м битах (см. рис. 6.1, г). Над логическими данными определены операции: логическое умножение (конъюнкция, ^), логическое сложение (дизъюнкция, v), логическое отрицание (Ø).

Значения элементарных данных формируются в ходе исполнения программы и имеют физическое представление в ОЗУ. В отличие от них идентификаторы данных существуют только на уровне логического представления - они используются для обозначения данных в тексте программы.

При исполнении такой программы обращение к данным производится по адресу ячейки, а не идентификатору. Адреса могут быть абсолютными - в этом случае они не изменяются при загрузке программы в ОЗУ - именно такой способ адресации применяется в исполняемых программных файлах с расширением com.

В исполняемых файлах с расширением exe на этапе трансляции устанавливаются относительные адреса данных, которые конкретизируются при размещении программы в ОЗУ - это несколько замедляет начало исполнения, зато снимает указанное выше ограничение на размер программы.

Можно указать ряд причин, поясняющих необходимость и удобство использования данных, организованных в некоторую структуру:

· отражение в организации данных логики задачи, объективно существующей взаимосвязи и взаимообусловленности между данными;

· оптимизация последовательности обработки данных;

· широкое применение при обработке данных циклических конструкций - в них при переборе нельзя автоматически менять имя переменной, однако, можно изменять индексы;

· неудобство использования большого количества одиночных данных, поскольку это ведет к необходимости использования многих имен.

Перечисленные причины приводят к тому, что в современных языках и системах программирования резервируется широкий спектр различных структур данных и, помимо этого, предусматривается возможность создания структур удобных и необходимых пользователю.

Относительно структур данных необходимо сделать следующие **общие замечания**:

· логический уровень организации данных отражается в тексте программы - им определяется порядок обработки данных;

· физический уровень представления структур в ОЗУ имеет всего две разновидности: *последовательные списки* и *связные списки* (см. п.6.3.3); на *ВЗУ* все структуры представляются в виде *файлов;*

· обработка данных возможна только после их размещения в ОЗУ; с 63У определены только операции записи и чтения;

· идентификаторы, как и у одиночных данных, существуют только в тексте программы и на этапе трансляции переводятся в адреса ячеек памяти.

Структурирование данных предполагает существование (или установление) между ними каких-то отношений (связей). В зависимости от характера этих отношений можно выделить несколько классификационных признаков структур данных.

Первым из них рассмотрим отношение *порядка*. По порядку данных структуры делятся на ***упорядоченные* и *неупорядоченные***.

***Упорядоченных* структурах** элементы размещаются по порядку в соответствии со значением некоторого признака. Наиболее простым признаком является *порядковый номер элемента*:, установление порядка в соответствии с номером называется *нумерацией*. При этом если весь набор имеет один общий идентификатор, то отдельным данным присваиваются собственные идентификаторы - *индексы* .

Чаще всего индекс задается целым числом, хотя это необязательно -в качестве индекса может выступать любой знак из конечного алфавита. Лексикографический порядок индексов определяет *отношение следования* между элементами структуры. Примером структур, в которых упорядочение производится по номеру элемента, являются *массивы*. Порядковый номер элемента можно считать *внешним* признаком, который может присваиваться элементу независимо от его значения. Помимо нумерации в структурах данных используется упорядочение *по значению* некоторого *внутреннего признака*

Примером неупорядоченных структур являются, *множества* - в них не определен порядок элементов; единственное, что можно установить для каких-то конкретных данных, так это их принадлежность (или не принадлежность) выбранному множеству.

Следующим классификационным признаком структур является ***однородность***. К *однородным* относятся структуры, содержащие элементарные данные только одного типа. *Неоднородные* структуры объединяют данные разных типов. Примерами однородных структур являются массивы, множества, стеки. К неоднородным структурам относятся записи.

Еще одним признаком является *характер отношений* между элементами. По взаимной подчиненности элементов структуры данных подразделяются на ***линейные* и *нелинейные***.

***В линейных* структурах** все элементы *равноправны*. К ним относятся массив, множество, стек, очередь.

В *нелинейных* структурах между элементами существуют *отношения подчиненности* или они могут быть связаны логическими условиями. К ним относятся деревья, графы, фреймы.

Массив - *упорядоченная линейная совокупность однородных данных*.

Комментарии к определению:

· термин *«упорядоченная»* означает, что элементы массива пронумерованы;

· термин *«линейная»* свидетельствует о равноправии всех элементов;

· термин *«однородных»* означает следующее: в том случае, когда массив формируется из элементарных данных, это могут быть данные лишь одного какого-то типа, например, массив чисел или массив символов; однако, возможна ситуация, когда элементами массива окажутся сложные (структурные) данные, например, массив массивов - в этом случае «однородных» означает, что все элементы имеют одинаковую структуру и размер.

Количество индексов, определяющих положение элемента в массиве, называется ***мерностью*** *массива*.

Ели индекс единственный, массив называется ***одномерным****;* часто такой массив называют также *вектором,* строкой или столбцом. Для записи элементов одномерного массива используется обозначение m_i; в языках программирования приняты обозначения $m(i)$ или $m[i]$.

Массив, элементы которого имеют два индекса, называется ***двумерным*** **или** ***матрицей***. Пример обозначения: $G[3,5]$; при этом первый индекс является номером строки, а второй индекс - номером столбца, на пересечении которых находится данный элемент.

Массивы с тремя индексами называются ***трехмерными*** и т.д. Максимальная мерность массива может быть ограничена синтаксисом некоторых языков программирования, либо не иметь таких ограничений.

Максимальное значение индексов определяет ***размер* массива.** Размер массива указывается в блоке описания программы, поскольку при исполнении программы для хранения элементов массива резервируется необходимый объем памяти. Если в процессе исполнения программы размер массива не изменяется, то в этом случае говорят о массивах ***фиксированного размера***; если определение размеров массива или их изменение происходит по ходу работы программы, то такие массивы называются ***динамическими*** (динамически описываемыми).

Допустимый набор операций над элементами массива определяется типом данных, из которых массив сформирован. В некоторых языках программирования над массивом в целом определена операция присваивания в этом случае всем элементам массива присваивается одинаковое значение, равное вычисленному значению выражения; возможна также операция присваивания для двух одинаковых по типу, размеру и размерности массивов *M2:= M1* - производится *поэлементное* присваивание значений *(M2(i,j,k…) := M1(i,j,k…))*.

Особое место занимают символьные массивы - они называются ***строками* или *строковыми данными***. С ними возможен целый набор операций, неопределенных для одиночных символьных данных. В первую очередь, это операция ***конкатенации*** (объединения) строк с сформированием новой строки. Помимо этого имеются операции замещения части строки, а также определения ее числовых характеристик.

Стек (магазин) и очередь *являются упорядоченными, линейными, неоднородными структурами*. Эти структуры реализуются в виде специальным образом организованных областей ОЗУ компьютера либо в качестве самостоятельных блоков памяти. В стеке ячейки памяти соединяются друг с другом таким образом, что при занесении данных в первую ячейку содержимое всех остальных сдвигается в соседние вниз, при считывании - содержимое сдвигается вверх по ячейкам, как показано на рис. 6.2. Другими словами, вход в стек возможен только через первую ячейку *(вершину стека),* поэтому извлекаться первой будет та информация, которая была занесена последней.

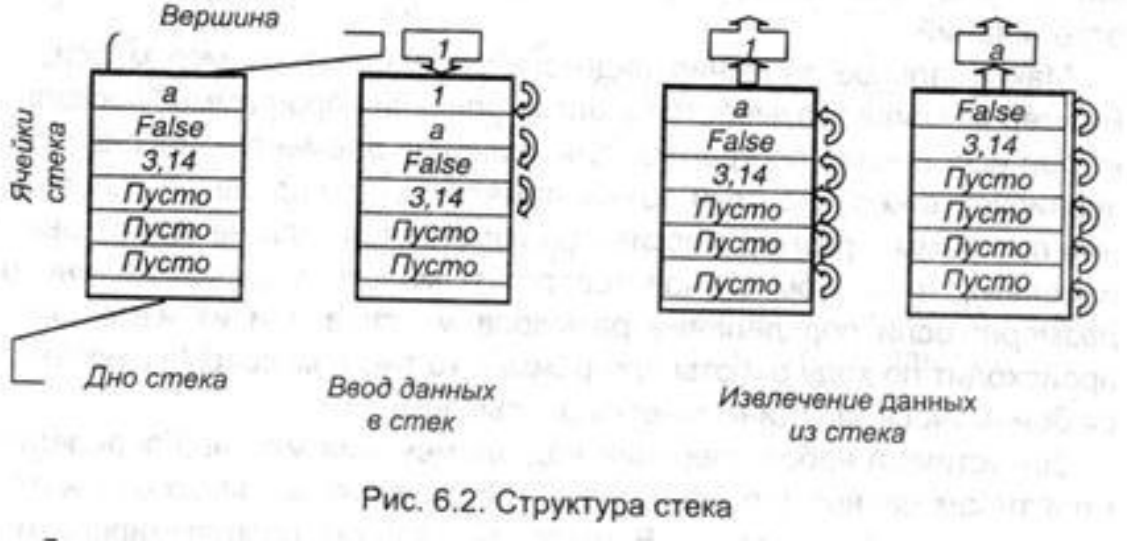


Рис. 6.2. Структура стека

Отличие очереди от стека только в том, что извлечение информации производится в порядке *«первым вошел - первым вышел»,* т.е. со дна стека.

Таким образом, данные имеют *порядок расположения* и они *равноправны* - поэтому структура является упорядоченной и линейной. Однако в общем случае в ячейках стека могут содержаться данные разных типов - по этому признаку структура оказывается неоднородной.

Дерево или ***иерархия*** является примером *нелинейной* структуры. В ней элемент каждого уровня (за исключением самого верхнего) входит в один и только один элемент следующего (более высокого) уровня. Элемент самого высокого уровня называется *корнем,* а самого нижнего уровня - ***листьями***.

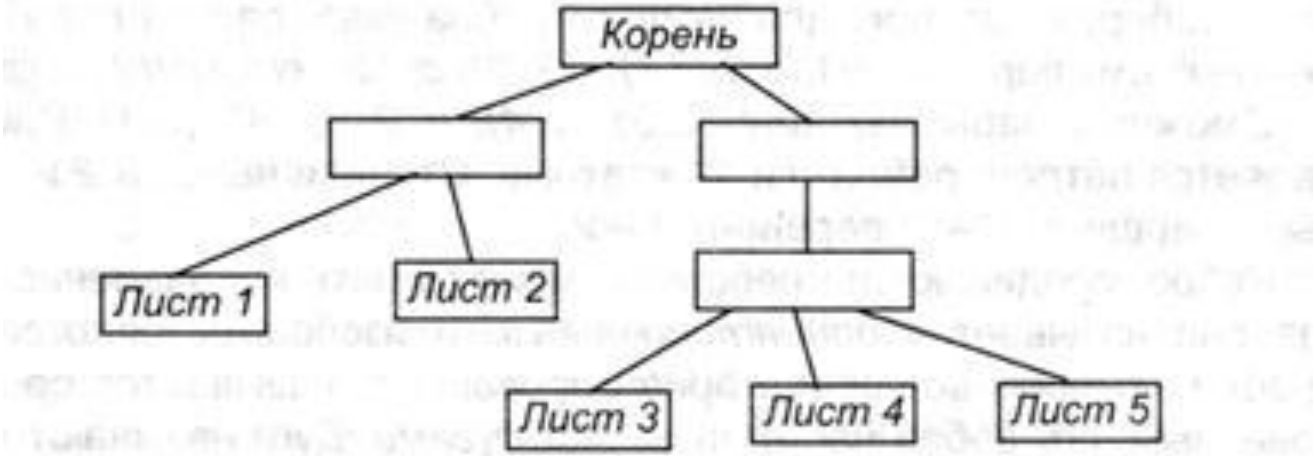


Схема такой структуры показана на рисунке. Отдельные элементы могут быть однородными или нет. Примером подобной организации служат файловые структуры на внешних запоминающих устройствах компьютера.

Часто отношения между данными представляются в виде ***графа*** -совокупности точек и линий, в которой каждая линия соединяет две точки. В информатике точка получает смысл элемента структуры а линии - смысл отношения между элементами.

Точки называются ***вершинами*** графа, линии - ***ребрами***. Если ребро соединяет две вершины, то говорят, что ребро *инцидентно* этим вершинам, а сами вершины называются *смежными*. Число ребер, инцидентных вершине, называется *степенью* вершины. Если два ребра инцидентны одной и той же паре вершин, они называются *кратными*. Ребро, у которого совпадают обе вершины, называется ***петлей***.

Ребро, соединяющие вершины, может иметь направление - тогда оно называется ***ориентированным*** и изображается стрелкой. Граф, в котором все ребра ориентированные, называется ориентированным; его ребра часто называют ***дугами***. Дуги называют *кратными,* если они соединяют одни и те же вершины и совпадают по направлению. При обозначении дуги всегда сначала указывается вершина, из которой она начинается

Маршрут - это последовательность ребер, в котором конец предыдущего ребра совпадает с началом следующего, например, *a*, *c, e* на *графе 1*. Маршрут, в котором конечная вершина совпадает с начальной, называется ***циклом***. Граф называется *связным,* если между любыми двумя его вершинами имеется маршрут. Связный граф с *n* вершинами содержит не менее *n*-1 ребер.

По рассмотренной ранее классификации *граф является упорядоченной, нелинейной, неоднородной структурой*. Понятие графа благодаря его наглядности и высокой общности в информатике выступает анных, систем, порядка выполнения действий

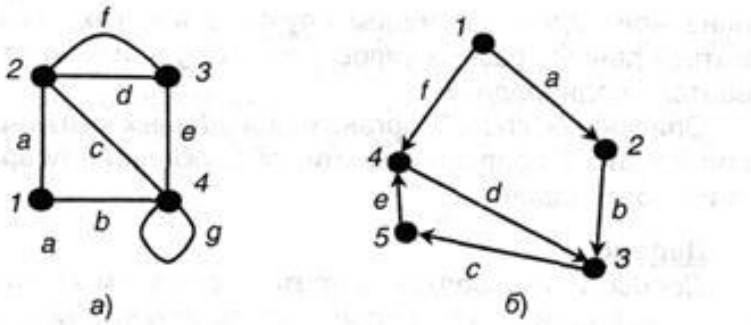


Рис. 6.3. Примеры графов:
а – граф 1 (неориентированный); *б* – граф 2 (ориентированный)

Логическая запись - *поименованная совокупность элементарных данных, имеющая смысловую завершенность*.

Пример записи - строка списка студентов:

Фамилия	Год рождения	Год поступления в вуз	Курс	Номер зачетной книжки

Определение требует ряда комментариев и дополнений:

1.Логическая запись объединяет не любые разрозненные (по смыслу) данные, а те, что относятся и характеризуют некоторую систему или объект - именно в этом плане следует понимать сочетание *«смысловая завершенность»* в определении. Запись в целом отражает различные свойства (атрибуты) системы.

2. Логическая запись имеют многоуровневую структуру. Элементами самого нижнего уровня являются элементарные данные - символы, числа, логические данные. Элементарные данные хранятся и считываются целиком, доступ к их частям невозможен. Совокупности элементарных данных, имеющих определенный смысл, но не обладающих смысловой завершенностью, образуют поля, каждое из которых соответствует одному атрибуту системы. Поле характеризуется типом элементарных данных, из которых оно строится, а также информационным размером

3. Поля записи связаны между собой. Связи между ними могут носить *функциональный характер* (значение одного поля посредством некоторого преобразования (правила) определяет значение другого; например, две первые цифры поля *«Номер зачетной книжки»* равны двум последним поля «Год *поступления»*), либо связи могут быть *причинно-следственными* (например, поле «Год *рождения»* определяется значением поля *«Фамилия»*).

4. Логические записи сами могут объединяться и образовывать структуры, которые определяются моделью данных. Например, совокупность указанных выше записей для всех студентов, обучающихся в одной группе, образуют массив, который называется базой данных (реляционного типа). Обращение к базе при сохранении и использовании осуществляется по ее идентификатору *(ГРУППА_101)*. Возможны и более высокие структурные объединения, например, структуры, элементами которых будут базы данных (объединение баз данных по всем группам факультета). Программные системы, позволяющие создавать и использовать базы данных называются системами управления базами данных (СУБД).

5. Логическая запись имеет собственный идентификатор, по которому можно обратиться к записи в целом (например, порядковый номер студента в группе). Поля также имеют идентификаторы, по которым они становятся доступны для просмотра или изменения значения. Идентификатор поля строится из идентификатора базы, идентификатора записи и собственно имени поля, например, *ГРУППА_101(13)*. *Фамилия*.

Таким образом, существует иерархическая многоуровневая структура данных, показанная на рис. 6.4.

Каждый выше расположенный уровень содержит низлежащий в качестве составных элементов. В этой иерархии запись является первым элементом структуры, обладающим смысловой завершенностью и, следовательно, самостоятельностью. Более высокие структуры образуются повторением записей с одинаковой и неизменной структурой.

Рис. 6.4. Иллюстрация многоуровневой иерархической структуры данных

Структура информационного массива определяется один раз на этапе его создания и в процессе использования уже не изменяется. В языках программирования это достигается описанием структуры ***в*** блоке описаний программы; в **СУБД** - установлением перечня и последовательности полей записи на начальном этапе создания базы данных. Всякое изменение структуры эквивалентно созданию новой структуры. Что же касается количества записей в структурированном информационном массиве, то при представлении его в ОЗУ компьютера возможны две ситуации: либо под него выделяется область ОЗУ фиксированного размера, либо размер области при необходимости может меняться.

В первом варианте в начале работы программы происходит резервирование областей ОЗУ для хранения информационных массивов. С этой целью в тексте программы указывается, какого типа и размера информационные массивы будут в дальнейшем использованы. В процессе выполнения программы могут меняться *значения* элементов информационного массива, но не его размер. По этой причине в случае, если размер массива не известен заранее, приходится осуществлять избыточное резервирование, что, безусловно, приводит к нерациональному использованию памяти компьютера. Отсутствие возможностей *динамического* описания массивов считается одним из существенных недостатков языка программирования.

Информационные массивы, допускающие изменение размера называются ***динамическими***. В этом случае данные могут иметь *последовательное* или *связное* представление в ОЗУ.

- ***Последовательное*** представление иллюстрируется рис. 6.5. В этом варианте данные размещаются в соседних последовательно расположенных ячейках памяти. На размещение одной записи может потребоваться несколько ячеек, но их количество *одинаково* для каждой из записей, поэтому идентификатор записи однозначно связывается с номером первой ячейки, начиная с которой запись размещается.

  Физический порядок следования полностью соответствует логическому. Такая совокупность записей называется ***последовательным списком***. Для его хранения в ОЗУ выделяется блок ячеек фиксированного размера. При выполнении команды обрабатывающей программы *«Добавить запись»* происходит увеличение размера массива на одну строку в конце блока и при необходимости производится перезапись массива в ОЗУ. В добавленной строке размещается новая запись, как это показано на рис. 6.5,*б*. При изъятии каких-то записей по команде *«Удалить запись»* соответствующая строка очищается и после перезаписи заполняется содержимым следующих ячеек (рис. 6.5,*в*).

- ***Связное*** представление данных основано на том, что в записи дописывается дополнительное поле, в котором размещается *указатель адреса,* т.е. ссылка на то место в ОЗУ, где располагается следующая запись. При этом физический порядок размещения записей не соответствует логическому - записи располагаются в любых свободных ячейках ОЗУ, причем, не обязательно подряд. Такие структуры называются *связными списками*. Их удобство состоит в гибкости структуры - без перезаписи остальных элементов можно легко добавлять новые или исключать имеющиеся - для этого достаточно лишь изменить состояние поля указателя адреса, что иллюстрируется рис. 6.6.

  Недостаток описанного способа представления информационного массива в ОЗУ состоит в том, что в нем невозможно напрямую обратиться к нужной записи - поиск ее осуществляется по цепочке переходов, безусловно, увеличивая время доступа к данным.

№ записи	№ ячейки	Содержание
1	3000	Запись *A*
2	3012	Запись *B*
3	3024	Запись *C*
…	…	…
N	3000 + (*N*–1)·12	Запись *N*

а)

№ записи	№ ячейки	Содержание
1	4000	Запись *A*
2	4012	Запись *B*
3	4024	Запись *C*
…	…	…
N	4000 + (*N*–1)·12	Запись *N*
N+1	4000 + *N*·12	Запись *N*+1

б)

№ записи	№ ячейки	Содержание
1	4000	Запись *A*
2	4012	Запись *C*
3	4024	Запись *D*
…	…	…
N–1	4000 + (*N*–2)·12	Запись *N*
N	4000 + (*N*–1)·12	Запись *N*+1

в)

Рис. 6.5. Схемы последовательного динамического размещения данных в ОЗУ:
а – размещение в ОЗУ; *б* – добавление записи; *в* – удаление записи.

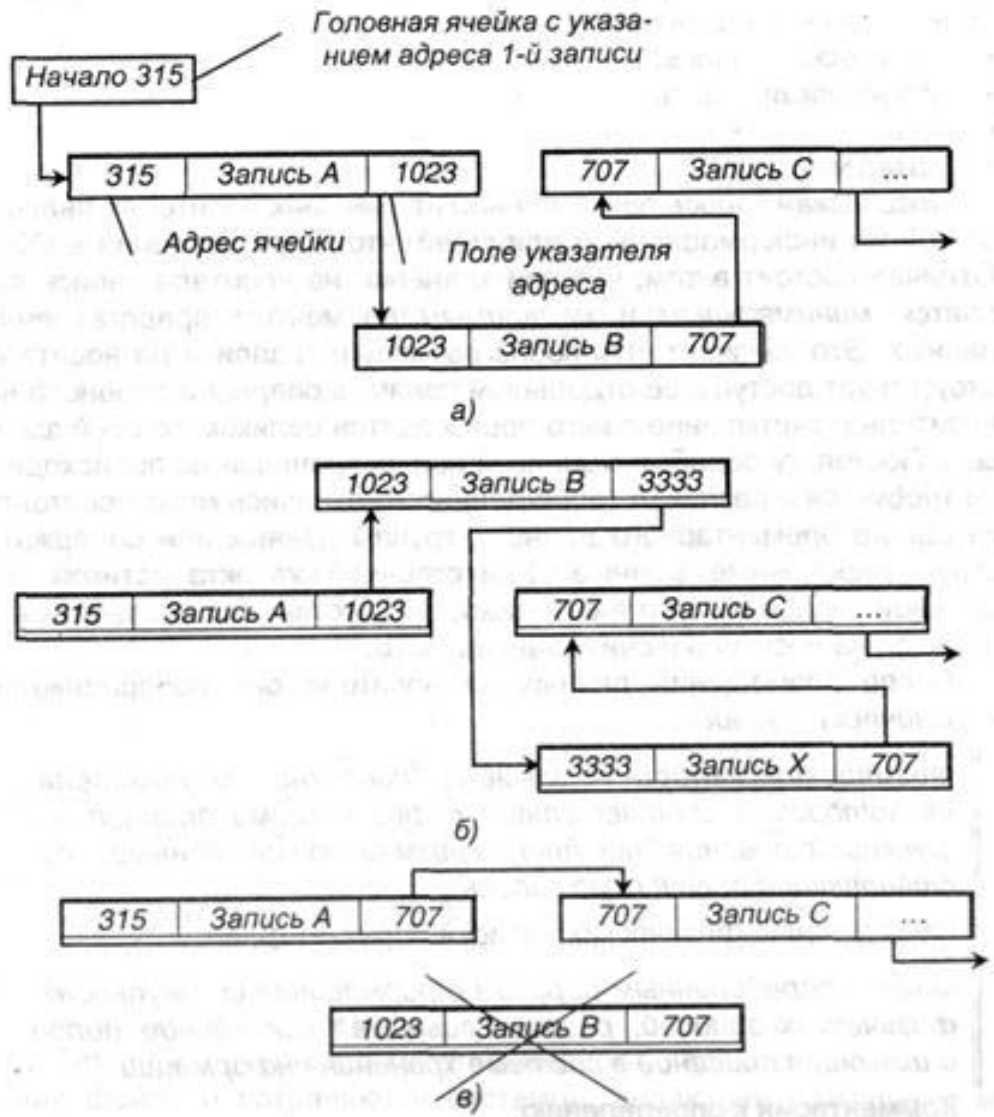


Рис. 6.6. Схема связного динамического размещения данных в ОЗУ:
а – организация связей между записями; *б* – добавление записи;
в – исключение записи

Основными информационными единицами при сохранении данных на внешних носителях являются:

– логическая запись;

– физическая запись;

– файл;

– каталог (папка).

- ***Логическая запись*** при хранении на внешних носителях является той же информационной единицей, что и при хранении в ОЗУ. При хранении на внешнем носителе запись является минимальным и неделимым элементом представления данных. Это означает, что после размещения записи на носителе отсутствует доступ к ее отдельным элементам, а операции переноса на носитель и считывание с него производятся целиком со всей записью.

Поскольку обработка записей при их хранении не происходит, не требуется и различия типов данных, т. е. запись может состоять из одного элементарного данного, группы данных или содержать структурированные данные. Единственной характеристикой отдельной записи является ее длина, а допустимыми операциями — перенос на носитель и считывание с него. После размещения данных на носителе они превращаются в физическую запись.

- **Физическая запись** — элемент поверхности носителя, на котором в соответствии с физическими принципами функционирования носителя размещаются данные, составляющие логическую запись.

  Объединение физических записей образует файл.

- **Файл** — определенным образом оформленная совокупность физических записей, рассматриваемая как единое целое и имеющая описание в системе хранения информации.

  На носителях хранятся не только сами файлы, но и сведения о них и их размещении. Эти сведения используются в операциях с файлами. Любые файлы содержат данные, закодированные с помощью двоичного алфавита. При этом способы кодирования и назначение файлов бывают различными.

  По этой причине файлам приписывается еще одна характеристика — тип. Тип входит в идентификатор файла и указывается в виде расширения имени.

Принципиально различными по типам следует считать программные (исполняемые) файлы и файлы данных.

Программные файлы содержат тексты программ в машинном коде, которые бывают загружены в ОЗУ и исполняться. Программные файлы имеют расширение com или exe. К этой же категории относятся так называемые командные файлы с расширением bat, содержащие в текстовом формате команды MS DOS, которые могут последовательно выполняться как программа.

Файлы данных формируются в результате работы какой-либо программы; они не являются исполняемыми и служат только в качестве хранилищ данных. Многие программные системы при формировании файлов данных приписывают им вполне определенные расширения — по ним можно установить, какой программой файл создан. Тип файла, как и его собственное имя, являются частью описания файла и сохраняются системой, ведающей размещением файлов на носителе.

Самым верхним уровнем представления данных на внешних носителях являются структуры файлов — **каталоги** — в них помещаются файлы, объединенные каким-то признаком. Как Каталоги допускают образование вложенных структур, т. е. подкаталогов. Каталоги образуют иерархическую структуру, в связи с этим правомочно использование термина дерево каталогов. При этом каталог, располагающийся на вершине иерархии, принято называть **корневым**.

Устройства, выполняющие операции, связанные с сохранения и считывания данных на материальном носителе, называются ***внешними запоминающими устройствами****(ВЗУ) или устройствами внешней памяти.*

Любое ВЗУ реализует один из двух возможных принципов размещения информации - ***последовательный доступ*** или ***прямой доступ***. Первый вариант используется при сохранении информации на ленточных носителях, например, магнитной или бумажной ленте - в этом случае записи размещаются одна за другой, т.е. последовательно. Считывание записей также производится *последовательно,* и для того, чтобы отыскать нужную запись, требует просмотреть все предыдущие.

Для реализации *прямого* доступа на носителе должны быть обозначены области для записи информации - такие области называются ***блоками***. Блок, подобно ячейке ОЗУ, служит контейнером для размещения данных. Обратиться к данным для записи-считывания можно по номеру блока.

Операция разбиения поверхности носителя на блоки называется ***форматированием*** - она производится в обязательном порядке и предшествует использованию носителя. Блок обычно имеет строго определенную для данного носителя информационную емкость. Блок может содержать только целое число физических записей - из-за этого часть блока длиной меньше, чем размер записи, оказывается пустой и не используется. На носителях большой емкости блоки объединяются в группы - ***кластеры*** -запись файлов производится в них и применяется адресация по номерам кластеров.

На дисковых носителях имена файлов хранятся отдельно от физических записей. В определенном месте диска при его форматировании создается специальная область, в которой располагается *таблица размещения файлов* - *FAT (**F**ile **A**llocation **T**able)*. В эту таблицу заносятся имена и атрибуты файлов.Обращение к файлу происходит в два этапа: сначала с помощью файловой таблицы по имени файла находится номер кластера, а затем считывающее-записывающая головка ВЗУ устанавливается над ним и производит операции. Ситуация иллюстрируется рис. 6.7.

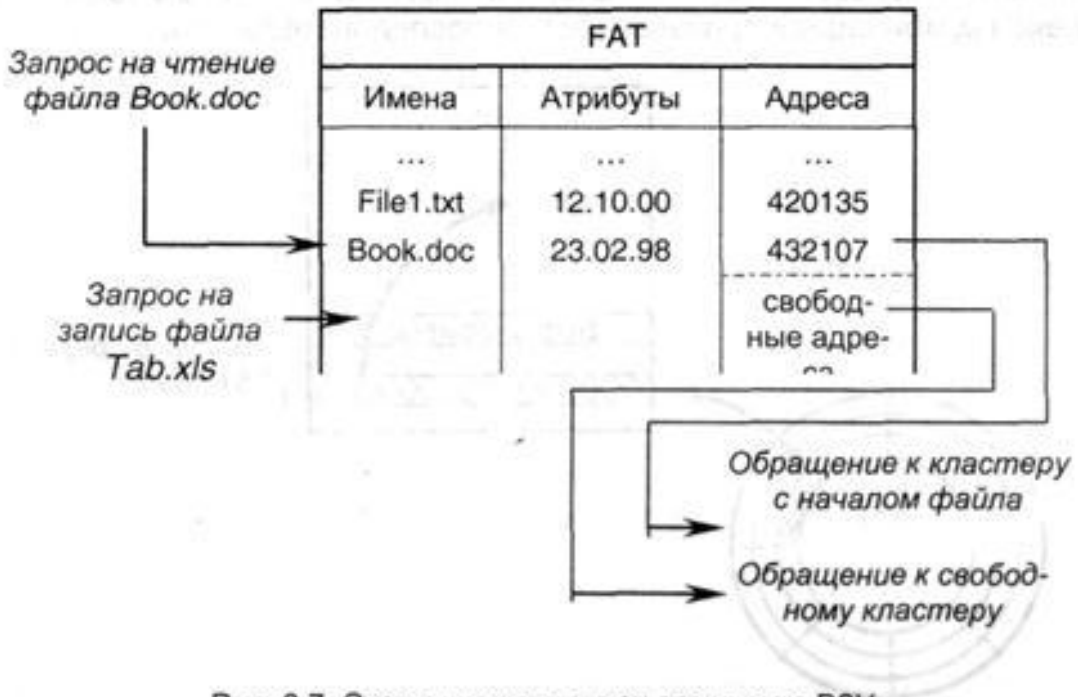


Рис. 6.7. Схема чтения-записи данных на ВЗУ

При обмене между ВЗУ и ОЗУ данные пересылаются не отдельными записями, а блоками, размер которых совпадает с размером блока ВЗУ - 512 байт; схема обмена представлена на рис.6.8.

Для организации обмена в ОЗУ выделяется специальная область - буфер обмена; размер буфера устанавливается при конфигурировании операционной системы компьютера. При пересылке из ОЗУ в ВЗУ данные сначала из ОЗУ пересылаются в буфер, пока он не заполнится, затем целым блоком отправляются в подготовленный блок ВЗУ. Считывание идет обратным путем. Обмен может идти минуя центральный процессор - в этом случае одновременно с обменом может производиться обработка данных (поступивших или иных).

Прямой доступ оказывается некой комбинацией произвольного и последовательного.

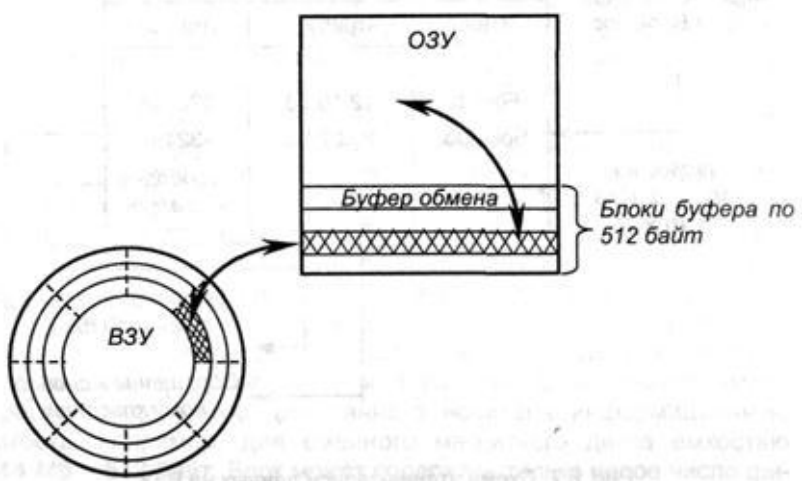


Рис. 6.8. Схема обмена блоками данных между ОЗУ и ВЗУ