ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»
Кафедра «Математика и теоретическая механика»

Методические рекомендации для самостоятельной работы обучающихся по дисциплине

Информатика

Направление подготовки (специальность) 27.03.04 Управление в технических системах

Профиль образовательной программы «Системы и средства автоматизации технологических процессов»

Форма обучения очная

СОДЕРЖАНИЕ

1. ОРГАНИЗАЦИЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1.1. Организационно-методические данные дисциплины

№ п. п.	Наименование темы	Общий объем часов по видам самостоятельной работы				
		подготовка курсового проекта (работы)	подготовка реферата/ эссе	Индивидуальные домашние задания (ИДЗ)	самостоятельное изучение вопросов (СИВ)	подготовка к занятиям (ПкЗ)
1	2	3	4	5	6	7
1	Информация и информационные процессы. Представление информации	-	-	-	2	3
2	Информационно-логические основы построения персонального компьютера	-	-	-	2	3
3	Состав и структура ЭВМ и ПЭВМ	-	-	-	2	3
4	Программное обеспечение ЭВМ	-	-	-	2	3
5	Текстовые и графические редакторы	-	-	-	2	5
6	Электронные таблицы и базы данных	-	-	-	2	4
7	Основы моделирования, алгоритмизации и программирования	-	-	-	2	4
8	Локальные и глобальные сети ЭВМ. Основы защиты информации	-	-	-	2	4

2. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО САМОСТОЯТЕЛЬНОМУ ИЗУЧЕНИЮ ВОПРОСОВ

2.1 Качество информации

Информационные ресурсы в силу определения понятия "информация", которое связано с уменьшением степени неопределенности в системе "передатчик – приемник" непосредственно у потребителя, могут быть охарактеризованы рядом свойств или признаков, выявляемых и оцениваемых самими потребителями, а поэтому эти оценки весьма субъективными.

Каждый потребитель, исходя из имеющегося него объема знаний и опыта работы с информацией в определенной предметной области, может выделить те свойства информации, которые, по его мнению, определяют смысловую сущность этого ресурса. Последний при этом может иметь различные цели использования, различную степень новизны и другие свои характеристики, которые по-разному могут влиять на процесс принятия решения по выполнению каких-либо действий со стороны потребителя.

При работе с информацией возможны случаи, при которых для разных потребителей один и тот же ресурс может иметь различные выявленные признаки и разную степень оценки каждого из них. Так, к примеру, для одного потребителя предлагаемый информационный ресурс может оказаться совершенно новым, чрезвычайно актуальным и полезным материалом с точки зрения получения новых знаний и их использования на практике. Другой же потребитель с учетом степени его квалификации или в силу сложившихся обстоятельств может не оценить смысл содержащегося в том же материале и не придать должного значения предлагаемой информации.

Кроме того, качественная сторона информации может рассматриваться как с точки зрения потребителя конкретной информации, так и с точки зрения информационной службы, информационной системы или эксперта-специалиста, передающих данный ресурс для дальнейшего его применения. Учитывая это, оценку следует проводить на основе качественных и количественных признаков, характеризующих различные факторы и обстоятельства производства определенного информационного ресурса, запроса потребителя на последний, процессов его передачи, получения и непосредственного использования в научно-теоретической, производственно-практической или социальной деятельности человека.

Следует отметить, что качество одной и той же информации при реализации различных целей или видов деятельности (в технике, метрологии, экономике, социологии и др.) различно. Отличаются и наборы параметров (показателей), и методики определения качества информации в разных предметных областях знаний.

Характеристики качества информации определяют существенные свойства данного объекта, который может находиться в разных стадиях информационных технологий: сбора, хранения, переработки, передачи, получения и использования. Именно с таких позиций и рассмотрим основные характеристики качества.

Этап сбора или отбора данных (возникновение информации) сопровождается чрезвычайно важной характеристикой информации – ее репрезентативностью, связанной с определенными правилами сбора, отбора и формирования данных таким образом, чтобы последние наиболее правильно отражали исследуемые стороны и свойства объекта и представляли в дальнейшем этот объект адекватно. Информация об объекте отражает его структуру, свойства, внутренние и внешние связи, реальные процессы, в которых он участвует приблизительно, лишь стремясь к истинному и полному отражению действительности.

Нарушение репрезентативности при формировании информации приводит нередко к существенным ее погрешностям и сказывается на основных характеристиках информации – *точности и достоверности*.

Точность информации характеризует степень приближения этой информации к реальному состоянию отображаемого объекта, процесса, явления или окружающей действительности.

Достоверность (адекватность, истинность, истинность) информации определяется ее свойством отражать реально существующие объекты с необходимой точностью.

Любая информация об объекте или явлении отражает действительность с определенной степенью погрешности. Это связано с несовершенством применяемых методов и средств сбора информации или измерения информативных параметров. Кроме того, при регистрации данных или сигналов, несущих информацию, всегда присутствует уровень посторонних шумов. В результате указанные факторы могут в различной степени

влиять на качество отображения объекта, а следовательно, на достоверность получаемой информации вплоть до полного ее несоответствия реальному объекту.

Следует отметить, что в настоящее время разработаны и активно развиваются методы восстановления информации на фоне даже сильных помех (зачастую по уровню превышающих полезный сигнал) и повышения тем самым ее достоверности.

С другой стороны достоверность созданной (первичной) информации можно понизить, применяя в дальнейшем несоответствующие методы и средства ее хранения, переработки или неправильного использования.

Допустимая степень погрешности, как правило, определяется целевой установкой при реализации конкретной деятельности и зависит от уровня изученности объекта на данный момент времени.

При формировании информации весьма важный характеристикой является его смысловая содержательность, которая отражает количество информации, несущей некий смысл сообщения, или объем содержащихся в нем знаний по отношению к общему объему сообщения. В данном случае указанное отношение может находиться в пределах от 0 (в сообщении нет смысла и оно полностью избыточно) до 1 (все данные сообщения осмысленны, а избыточность равна нулю).

Отмеченные характеристики в общем случае весьма трудно подвергнуть объективной оценке, так как они относятся к области, охватывающей признаки семантического порядка, и обуславливаются свойствами познания как процесса отражения и воспроизведения действительности в мышлении человека. Такие качественные характеристики можно выразить в виде упорядоченного ряда или в шкале оценочного порядка (например: высокая, средняя, низкая или отличная, хорошая, удовлетворительная, плохая).

Временные показатели характеризуют различные временные аспекты информации: моменты ее возникновения, моменты ввода конкретных информационных ресурсов в обращение, в том числе по коммуникационным каналам, временные периоды накопления и представления данных (календарный, налоговый, финансовый год, конец соответствующего года и т.п.) и т.д.

В общем, эта характеристика определяет связь между содержанием информации об объекте и ее соответствием реальному состоянию объекта на текущий момент времени.

Информация имеет свойства *старения*, так как она подвержена влиянию времени, следовательно, ин имеют определенный "жизненный" срок.

На *этапе хранения* информации характеристики должны отражать его индивидуальные особенности как единицы хранения и отношение к месту и времени хранения.

Источник определяет происхождение информационных ресурсов и может выступать в качестве конкретного лица (специалиста, эксперта), организации, собрания документов (информационный центр, база данных, библиотека, архив, фонд ит.д.), единичной публикации в печатном или электронном формате (книга, журнальная статья, энциклопедия, официальные и научные отчеты, технологическая документация и др.), а также в качестве измерительного датчика и т. д.

Тематическая принадлежность отражает принадлежность информационных ресурсов к определенной предметной области знаний, что позволяет проводить систематизацию и структуризацию ресурсов в соответствии с классификационными признаками объектов хранения.

Содержание определяет тематическую сущность представляемых знаний (тему, идею, теорию, методику) в определенной предметной области.

На *этапах передачи – получения* информации его качество может изменяться в худшую сторону в зависимости от характеристик канала связи и условий передачи (возможны искажения информации по причине появления случайных или систематических помех, ограниченной пропускной способности, отсутствия

необходимого протокола и т.п.). Отсюда вытекают требования установления тех свойств, которые отражают все стороны взаимодействия ИР в процессе его передачи, по каналам связи по направлению к получателю.

Из всех характеристик информационных ресурсов прагматического толка при передаче можно установить только один общепринятый количественный показатель – это цена. Остальные показатели, как это было показано выше, выявляются и оцениваются потребителем в зависимости от характера передаваемой информации и ее целевой направленности, а также в зависимости от степени информированности потребителя в той сфере знаний, к которой принадлежит рассматриваемый ресурс.

На *этапе переработки информации* характеристики PIP определяются целью и алгоритмом выполняемых с ним действий. К примеру, достоверность информации может снижаться в случае переработки числовых данных при их возможном округлении с неверно выбранной малой точностью, и, наоборот, при необоснованно высокой точности обработки необходимы большие массивы числовых данных, что повышает вероятность внесения случайных ошибок.

На *этапе непосредственного использования* информационных ресурсов как информационного продукта потребления (конечная цель информационных технологий) он должен характеризоваться свойствами, которые обусловливают способность удовлетворять определенные общественные или личные потребности на практике. Следовательно, характеристики качества в данном случае отражают его отношения к индивидуальному получателю и пользователю информацией и носят в основном прагматический характер.

Исходя из этого, эффективность использования информации обуславливается такими основными ее потребительскими **показателями качества, как *полезность, важность, актуальность, своевременность, соответствие запросу, цена***.

При этом приведенные характеристики могут не являться отражением чисто природных свойств информационных ресурсов, а проявляются только в результате предметно-практического взаимодействия объекта и субъекта. Таким образом, рассматриваемые характеристики качества информационных ресурсов имеет только прагматический смысл и оторваны от других его свойств синтаксического и семантического характера.

Полезность характеризует способность приносить пользу в интересах кого-нибудь, в соответствии с чьими-нибудь выгодами и определяется абсолютной или относительной величиной полученного эффекта (например, экономического, технологического, социального и т.д.) в результате использования конкретного ИР по отношению к результату, достигнутому без использования данного ресурса.

Особенность информации заключается в том, что в ряде случаев в ситуациях, при которых участники преследуют противоположные интересы (ситуационные конфликты: игра, аукционы, арбитраж, конкуренция производств или товаров, спортивные состязания, военное дело, политика), ложная или недостоверная информация, невольно или преднамеренно переданная одним участником другому (конкуренту, сопернику, противнику и т.п.), в случае ее использования может принести ущерб одной стороне, а значит, явиться полезной для другой стороны.

Важность определяет степень влияния используемого информационных ресурсов в процессе анализа складывающейся ситуации и принятия решения на пути достижения поставленной цели по отношению к влиянию другой используемой при этом информации. Эта характеристика выражает, насколько нужна и значительна именно данная информация для принятия решений.

Некоторые исследователи, определяя это качество, вводят понятие ценности информации, подразумевая при этом ее важность или нужность в общем для принятия решений. Но в данном случае трудно подобрать критерии определения ценности конкретных видов информации при принятии решений.

Актуальность отражает степень важности и значительности содержательной сущности полученной информации в момент ее использования, в том числе для анализа складывающейся ситуации и принятия решения по управлению наилучшим образом.

Эта характеристика подчеркивает востребованность информационных ресурсов именно в определенный момент времени и зависит от динамики изменения его свойств и от периода времени, прошедшего с момента возникновения этого ресурса. Дело в том, что со временем многие ИР устаревают и полностью или частично теряют свою актуальность, например, в процессах управления объектами и производственными процессами, в рыночной экономике, военном деле и т.п. Поэтому с актуальностью часто связывают коммерческую ценность информации, т.е. ее цену.

При этом следует отметить, что использование информации, полностью или частично потерявшей актуальность, может привести к ошибочным решениям. Поэтому в процессах управления каким-либо объектом стремятся получить информацию о нем до момента его изменения таким образом, чтобы успеть выработать необходимые решения по последующему управлению. В этом случае информация адекватно и достоверно отражает действительность и, следовательно, сохраняет свою актуальность для использования.

Характеристику информации – *актуальность* – можно уподобить "жизни" данного ресурса. По прошествии некоторого времени информация может стать неактуальной и ненужной для дальнейшего использования. Это свойство часто используют в системах защиты объектов от несанкционированного проникновения, например, путем частой смены паролей или пропусков. В системах защиты информации разрабатывают к применяют ключи доступа к ним с весьма сложными схемами кодирования к шифрования, с тем, чтобы продолжительность возможных попыток вскрытия таких ключей была достаточно велика по сравнению с периодом встроенных процедур обновления кодовых ключей.

Своевременность характеризует факт поступления информации в пределах временного периода между моментом появления потребности в данной информации (момент спроса на ИР) и моментом выполнения аналитических процедур и принятия конкретного решения по управлению. Иначе говоря, эта характеристика означает поступление информации не позже заранее назначенного времени, согласованного со временем решения поставленной задачи, когда данная информация еще может повлиять на результат принятия решения.

Непоступление востребованного ИР создает большую неопределенность в процессе анализа складывающейся ситуации и повышает вероятность принятия неоптимального или нерационального, или неверного решения на пути достижения поставленной цели.

С другой стороны, более раннее поступление ИР, чем в момент появления потребности в данной информации, может снизить ее актуальность за счет старения и понижения объективности и достоверности отражения действительности на момент принятия решения.

Необходимо отметить, что несвоевременно поступившая информация, опоздавшая к моменту принятия решения, может и не терять полностью свою актуальность, а значит, может быть использована в аналитических целях оценки эффективности ранее принятых решений, их возможной коррекции, совершенствования системы управления и в других возможных случаях (даже повторного использования).

Соответствие запросу – содержательная тематическая характеристика ИР, которую определяет потребитель при получении ресурса определенной тематической направленности в ответ на его конкретный тематический запрос передающей стороне. В данном случае учитывается полнота или достаточность объема знаний, заложенных в ИР, которые необходимы для эффективного решения поставленных задач.

Цена определяет денежное выражение стоимости ИР в том случае, если последний выступает в виде продаваемого товара, который удовлетворяет специфическим потребностям пользователей. При этом цена в созданном ИР характеризует овеществленный общественный труд его производителей. В ее основе также заложена рыночная стоимость, складывающаяся с учетом признанных обществом на рынке затрат труда на подготовку ИР и его потребительских свойств с учетом востребованности, полезности, важности и других характеристик.

В некоторых обстоятельствах может проявиться целый ряд дополнительных характеристик, отражающих различные стороны в процессе достижения результатов практического использования ИР. Среди них: *риск неполучения запланированного результата или недостижения поставленной цели, опасность нанесения ущерба здоровью человека, ущерба животным, опасность нарушения экологии окружающей среды* и др.

2.2 Понятие переключательной и коммутационной схемы. Примеры схем

В вычислительных и других автоматических устройствах широко применяются электрические схемы, содержащие множество переключательных элементов: реле, выключателей и т. п. При разработке таких схем с успехом может быть использован аппарат алгебры логики.

Переключательная схема – схематическое изображение некоторого устройства, состоящего из переключателей и соединяющих их проводников, а также входов и выходов, на которые подается и с которых снимается электрический сигнал.

Каждый переключатель имеет только два состояния: замкнутое и разомкнутое. Переключателю X поставим в соответствие логическую переменную x, которая принимает значение 1 только в том случае, когда переключатель X замкнут и схема проводит ток; если же переключатель разомкнут, то переменная x равна нолю. При этом два переключателя X и X связаны таким образом, что когда X замкнут, то X разомкнут, и наоборот. Следовательно, если переключателю X поставлена в соответствие логическая переменная x, то переключателю X должна соответствовать переменная x.

Всей переключательной схеме также можно поставить в соответствие логическую переменную, равную единице, если схема проводит ток, и равную нолю — если не проводит. Эта переменная является функцией от переменных, соответствующих всем переключателям схемы, и называется функцией проводимости.

Рассмотрим функции проводимости F некоторых переключательных схем:

Схема не содержит переключателей и проводит ток всегда, следовательно, F – 1;

Схема содержит один постоянно разомкнутый контакт, следовательно, F – 0;

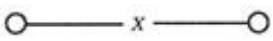


Схема проводит ток, когда переключатель x замкнут, и не проводит, когда x разомкнут, следовательно, $F(x) = x$;

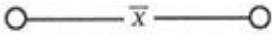


Схема проводит ток, когда переключатель x разомкнут, и не проводит, когда x замкнут, следовательно, $F(x) = \bar{x}$;

Схема проводит ток, когда оба переключателя замкнуты, следовательно, $F(x) = x \cdot y$;

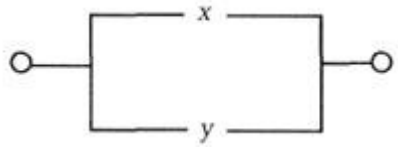


Схема проводит ток, когда хотя бы один из переключателей замкнут, следовательно, $F(x) = x \vee y$.

При рассмотрении переключательных схем решают, как правило, одну из основных задач: синтез или анализ схемы.

Синтез переключательной схемы по заданным условиям ее работы сводится к следующим трем этапам:

· составление функции проводимости по заданным условиям;

· упрощение этой функции;

· построение соответствующей схемы.

Анализ схемы характеризуется следующими этапами:

определение значений функции проводимости при всех возможных наборах входящих в эту функцию переменных;

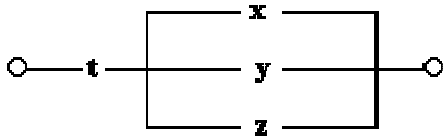
получение упрощенной формулы.

Рассмотрим примеры решения задач синтеза и анализа несложных переключательных схем.

Примеры.

1. Построим схему, содержащую 4 переключателя x, y, z и t, такую, чтобы она проводила ток тогда и только тогда, когда замкнут контакт переключателя t и какой-нибудь из остальных трёх контактов.

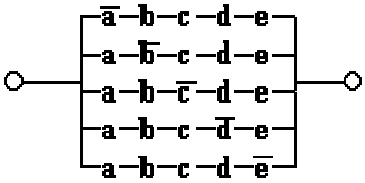
Решение. В этом случае можно обойтись без построения таблицы истинности. Очевидно, что функция проводимости имеет вид **F(x, y, z, t) = t · (x v y v z)**, а схема выглядит так:



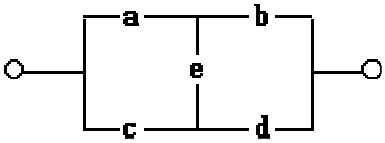
2. Построим схему с пятью переключателями, которая проводит ток в том и только в том случае, когда замкнуты ровно четыре из этих переключателей.

Решение : $F(a,b,c,d,e) = \bar{a}\cdot b\cdot c\cdot d\cdot e \vee a\cdot \bar{b}\cdot c\cdot d\cdot e \vee a\cdot b\cdot \bar{c}\cdot d\cdot e \vee a\cdot b\cdot c\cdot \bar{d}\cdot e \vee a\cdot b\cdot c\cdot d\cdot \bar{e}$

Схема имеет вид:



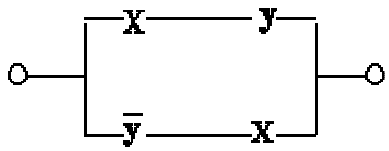
3. Найдем функцию проводимости схемы:



Решение. Имеется четыре возможных пути прохождения тока при замкнутых переключателях a, b, c, d, e : через переключатели a, b; через переключатели a, e, d; через переключатели c, d и через переключатели c, e, b. Функция проводимости

F(a, b, c, d, e) = a · b v a · e · d v c · d v c · e · b.

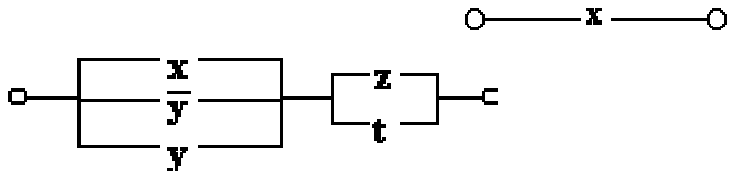
4. Упростить переключательные схемы:



а)

Решение: $F(x;y) = x \cdot y \vee \bar{y} \cdot x = x \cdot (y \vee \bar{y}) = x \cdot 1 = x$.

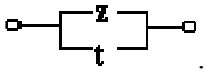
Упрощенная схема:



б)

Решение: $F(x;y;z;t) = (x \vee \bar{y} \vee y) \cdot (z \vee t) = 1 \cdot (z \vee t) = z \vee t$.

Упрощенная схема:



.

2.3 Дисковая память. Флэш-память

Носителями информации являются поверхности гибких и жестких дисков, в качестве немагнитных основ которых используются соответственно майлар (как и в магнитных лентах) и алюминиевые (в ряде случаев стеклянные) круги (диски). Стеклянные диски являются менее критичными к температурным изменениям и позволяют увеличить плотность записи информации. В настоящее время наиболее широкое распространение получили диски с напыленным магнитным слоем, а точнее, с металлической пленкой (например, кобальт).

Перед осуществлением записи на магнитный диск он должен быть специальным образом инициализирован – отформатирован. В результате форматирования на поверхности образуются концентрические окружности (синхронизирующие метки диска), называемые **дорожками** (track). Количество дорожек зависит от типа диска. Дорожки разбиваются на участки фиксированной длины, называемые **секторами.** Количество секторов на дорожке определяется типом и **форматом** диска, и они в основном одинаковы для всех дорожек. IBM PC-совместимые ПК могут работать с несколькими размерами секторов от 128 до 1024 байт. Стандартным сектором считается сектор из 512 байт. Данные любого размера (разрядности) размещаются в секторах с фиксированным размером, а дисковые операции записи и считывания производятся с целыми секторами.

Дорожки и сектора нумеруются с нуля, начиная с внешнего края диска, при этом сектор с нулевым номером на каждой дорожке резервируется для системных целей. Диски имеют две стороны. Так как накопители на жестких дисках могут состоять из нескольких дисков (стопка), то совокупность всех дорожек, по одной на каждой стороне с одинаковыми номерами, образует **цилиндр** с номером соответствующей дорожки.

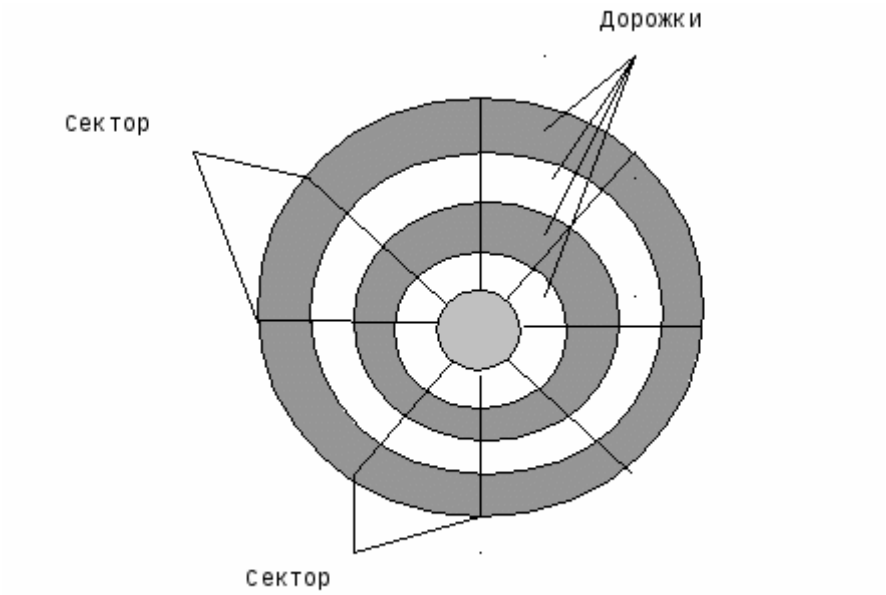


Рис. 1. Расположение дорожек и секторов на магнитном диске

Память на гибких магнитных дисках

Все НГМД, применяемые в PC, независимо от типа и размера имеют одинаковый *интерфейс* и *унифицированные разъемы*. Используемый в PC кабель-шлейф имеет перевернутый фрагмент из 7 проводов с номерами 10-16. Этот поворот позволяет подключать к контроллеру одним шлейфом до двух НГМД, причем адрес накопителя определяется его положением на шлейфе: для привода A: фрагмент перевернут, для B: – нет. *Контроллер* накопителей на гибких дисках *FDC* (Floppy Drive Controller) является всегда внешним по отношению к накопителю и обычно располагается на одной плате с контроллером или адаптером жестких дисков. Контроллер *FDC XT* поддерживает до четырех накопителей (FDD), хотя многие контроллеры имеют интерфейсные схемы только для одного шлейфа, то есть для двух накопителей. Эти контроллеры обеспечивают скорость передачи данных 250 и 300 Кбит/с.Контроллер *FDC AT* поддерживает только два накопителя, но обеспечивает более высокую скорость 500 Кбит/с. Современные контроллеры обеспечиваю скорость 1000 Кбит/с. В карте ресурсов AT имеется место под два контроллера НГМД. Контроллеры вырабатывают запрос *аппаратного прерывания* IRQ6 (BIOS INT OEh) по окончании выполнения внутренних операций. Для обмена данными может использоваться канал DMA2.

Память на жестких магнитных дисках

В отличие от накопителей на гибких дисках и их контроллеров, жестко стандартизованных и поэтому легко конфигурируемых, в PC применяется множество типов накопителей на жестких дисках, их интерфейсов и контроллеров, различающихся и способами конфигурирования.Накопители на жестких магнитных дисках НЖМД (HDD), появились с машинами PC/XT. Первые накопители имели интерфейс, являющийся расширением интерфейса НГМД, и подключались к специальной плате контроллера с модулем дополнительной BIOS, хранящей всю информацию об установленных жестких дисках. В машинах класса AT поддержку стандартного контроллера включили в системную BIOS, параметры используемых жестких дисков стали хранить в памяти CMOS. *Традиционные версии BIOS* поддерживают *до двух накопителей* на жестких дисках и хранят их параметры в ячейках памяти CMOS. *Расширенные версии BIOS* для современных двухканальных контроллеров ATA поддерживают *4 жестких диска* и хранят их параметры. Для дисков ATA используются следующие режимы адресации:

- CHS (целиндр-головка-сектор, традиционная трехмерная адресация данных на диске);
- ECHS (расширенная трехмерная адресация);

- LBA (линейная адресация данных на диске через логический адрес блока).

Учитывая, что в соответствии с форматом вызова функций дискового сервиса, одно устройство может иметь 2^{10}=1024 цилиндра, 2^8=256 головок, 2^6-1=63 сектора. Таким образом, при трехмерной адресации (CHS) и размере сектора в 512 байт максимальный объем диска не может превышать 7.875 Гбайт.$HDD_{max}(CHS) = [2^{10} * 2^8 * (2^6-1)] * 512 = 136\ 902\ 082\ 560$ байт = 7.875Gb (~8,4ГБ).

Все современные винчестеры используют LBA-адресацию. В режиме LBA параметры стандартных вызовов транслируются в т.н. линейный адрес, который вычисляется однозначно в «естественном» порядке счета секторов, т.е. сектору с нулевым лог. адресом соответствует первый сектор нулевой головки нулевого цилиндра. В этом случае номер каждого сектора представляет собой 28-битное число и максимальным диском для LBA будет:

$HDD_{max}(LBA) = 2^{28} * 512 = 137.4Gb$ (128ГБ).

В тоже время большинство ПО использует CHS-адресацию. Поэтому с появлением HDD с LBA адресацией, чтоб не модернизировать имеющееся ПО, поступили следующим образом. BIOS в случае определения LBA-винчестера, переводит его параметры в CHS-версию и ОС работает с ним с CHS-винчестером. Т.е. 28-битное значение адреса LBA «раскладывается» следующим образом: 16 бит – цилиндр, 8 бит – сектор, 4 бита – головка. В результате, при получении запроса на работу с диском, BIOS переводит для контроллера это значение LBA-адрес :

Возможно также подключение дисковых устройств и к параллельному порту, но через устройство, обеспечивающее один из вышеперечисленных интерфейсов. О дисках с интерфейсом USB говорить пока рано, а интерфейс FireWire является родственником SCSI-3.

Кэширование диска

Время доступа к различным блокам информации на HDD является переменной величиной, складывающейся из временами подвода магнитной головки (МГ) к искомой дорожке, времени успокоения вибрации МГ и времени подвода искомого сектора под МГ. Кэш или буфер HDD необходим, чтоб по возможности сократить время доступа к диску за счет, во первых, предварительной выборки данных, и во-вторых, за счет организации поблочного доступа. Для организации буфера используются два вида кэш-памяти аппаратная и программная.

Аппаратная кэш-память представляет собой значительный объем памяти и имеет архитектуру полного ассоциативного отображения. Она строится на плате кэш-контроллера HDD с использованием модулей высокопроизводительной памяти и имеет собственный процессор.

Программная кэш-память - это некоторая область системной памяти, зарезервированная для дискового кэша и управляемая утилитой (например, Windows SmartDrive).Объем программной кэш-памяти рекомендуется ограничивать четвертью объема системной памяти).

В многозадачных системах выгодно иметь HDD с мультисегментной кэш-памятью (для каждой задачи отводится своя часть кэша – сегмент). В адаптивной системной кэш-памяти для повышения производительности размер и количество сегментов могут изменяться. Для эффективной работы кэш (HDD) необходимо часто оптимизировать (дефрагментировать) диск, так как при этом относящиеся к одному и тому же файлу сектора будут расположены в физически близких секторах, что приведет к большей эффективности функционирования кэш.

Флеш-память (англ. *flash memory*) - разновидность полупроводниковой технологии электрически перепрограммируемой памяти (EEPROM). Это же слово используется в электронной схемотехнике для обозначения технологически законченных решений постоянных запоминающих устройств в виде микросхем на базе этой

полупроводниковой технологии. В быту это словосочетание закрепилось за широким классом твердотельных устройств хранения информации.

Благодаря компактности, дешевизне, механической прочности, большому объёму, скорости работы и низкому энергопотреблению, флеш-память широко используется в цифровых портативных устройствах и носителях информации. Серьёзным недостатком данной технологии является ограниченный срок эксплуатации носителей, а также чувствительность к электростатическому разряду.

Принцип работы полупроводниковой технологии флеш-памяти основан на изменении и регистрации электрического заряда в изолированной области («кармане») полупроводниковой структуры.

Изменение заряда («запись» и «стирание») производится приложением между затвором и истоком большого потенциала, чтобы напряженность электрического поля в тонком диэлектрике между каналом транзистора и карманом оказалась достаточна для возникновения туннельного эффекта. Для усиления эффекта туннелирования электронов в карман при записи применяется небольшое ускорение электронов путём пропускания тока через канал полевого транзистора (явление инжекции горячих носителей).

Чтение выполняется полевым транзистором, для которого карман выполняет функцию затвора. Потенциал плавающего затвора изменяет пороговые характеристики транзистора, что и регистрируется цепями чтения.

Эта конструкция снабжается элементами, которые позволяют ей работать в большом массиве таких же ячеек.

Технологические ограничения

Запись и чтение ячеек различаются в энергопотреблении: устройства флеш-памяти потребляют большой ток при записи для формирования высоких напряжений, тогда как при чтении затраты энергии относительно малы.

Ресурс записи

Изменение заряда сопряжено с накоплением необратимых изменений в структуре и потому количество записей для ячейки флеш-памяти ограничено. Типичные количества циклов стирания-записи составляют от десятков и сотен тысяч до тысячи или менее, в зависимости от типа памяти и технологического процесса. Гарантированный ресурс значительно более низок при хранении нескольких бит в ячейке (MLC и TLC) и при использовании техпроцессов класса "30 нм" и более современных.

Одна из причин деградации - невозможность индивидуально контролировать заряд плавающего затвора в каждой ячейке. Дело в том, что запись и стирание производятся над множеством ячеек одновременно - это неотъемлемое свойство технологии флеш-памяти. Автомат записи контролирует достаточность инжекции заряда по референсной ячейке или по средней величине. Постепенно заряд отдельных ячеек рассогласовывается и в некоторый момент выходит за допустимые границы, которые может скомпенсировать инжекцией автомат записи и воспринять устройство чтения. Понятно, что на ресурс влияет степень идентичности ячеек. Одно из следствий этого - с уменьшением топологических норм полупроводниковой технологии создавать идентичные элементы все труднее, поэтому вопрос ресурса записи становится все острее.

Другая причина - взаимная диффузия атомов изолирующих и проводящих областей полупроводниковой структуры, ускоренная градиентом электрического поля в области кармана и периодическими электрическими пробоями изолятора при записи и стирании. Это приводит к размыванию границ и ухудшению качества изолятора, уменьшению времени хранения заряда.

Срок хранения данных

Изоляция кармана неидеальна, заряд постепенно изменяется. Срок хранения заряда, заявляемый большинством производителей для бытовых изделий, не превышает 10-20 лет, хотя гарантия на носители дается не более чем на 5 лет. При этом память MLC имеет меньшие сроки, чем SLC.

Специфические внешние условия, например, повышенные температуры или радиационное облучение (гамма-радиация и частицы высоких энергий), могут катастрофически сократить срок хранения данных.

У современных микросхем NAND при чтении возможно повреждение данных на соседних страницах в пределах блока. Осуществление большого числа (сотни тысяч и более) операций чтения без перезаписи может ускорить возникновение ошибки. По данным Dell, длительность хранения данных на SSD, отключенных от питания, сильно зависит от количества прошедших циклов перезаписи (P/E) и от типа флеш-памяти и в худших случаях может составлять 3-6 месяцев.

Иерархическая структура

Стирание, запись и чтение флеш-памяти всегда происходит относительно крупными блоками разного размера, при этом размер блока стирания всегда больше, чем блок записи, а размер блока записи не меньше, чем размер блока чтения. Собственно это - характерный отличительный признак флеш-памяти по отношению к классической памяти EEPROM. Как следствие - все микросхемы флеш-памяти имеют ярко выраженную иерархическую структуру. Память разбивается на блоки, блоки состоят из секторов, секторы из страниц. В зависимости от назначения конкретной микросхемы глубина иерархии и размер элементов может меняться. Например, NAND-микросхема может иметь размер стираемого блока в сотни кбайт, размер страницы записи и чтения - 4 кбайт. Для NOR-микросхем размер стираемого блока варьируется от единиц до сотен кбайт, размер сектора записи - до сотен байт, страницы чтения - единицы - десятки байт.

2.4 Файловые менеджеры, утилиты и архиваторы

Файловый менеджер (англ. file mannager) - компьютерная программа, предоставляющая интерфейс пользователя для работы с файловой системой и файлами. Файловый менеджер позволяет выполнять наиболее частые операции - копирования, переноса, удаления, редактирования текстовых файлов, гибкого запуска программ для работы с этими файлами…

Помимо основных функций, многие файловые менеджеры включают ряд дополнительных возможностей, например, таких как работа с сетью (через FTP, NFS и т. п.), резервное копирование, управление принтерами и пр.

Существует два вида файловых менеджеров - навигационные и ортодоксальные. Основное отличие - в последних имеется две панели, реализована соответствующая модель работы.

Наиболее известные ортодоксальные файловые менеджеры: Norton Commander, Dos Navigator, Volkov Commander, PIE Commander, FAR Manager, Total Commander, POSIX (Linux, BSD и т. д.), Midnight Commander, Krusader, GNOME Commander.

Навигационные файловые менеджеры: проводник Windows (англ. Windows Explorer) -- встроен в Windows, Mac OS X, Finder, Path Finder, POSIX (Linux, BSD и т.д.), Konqueror -- поставляется с KDE, Nautilus (файловый менеджер) -- поставляется с GNOME

Современный файловый менеджер должен: обеспечивать удобную возможность работы с файлами, копировать, удалять, перемещать, создавать редактировать текстовые файлы, запускать внешние программы для работы с разными типами файлов, позволять с легкостью и удобством работать как клавиатурой, так и с помощью мышки, поддерживать технологию плагинов и настройку цветовых схем.

В состав базового дистрибутива файлового менеджера должны входить ряд модулей, плагинов, которые непосредственно связаны с работой с файлами:

1. Просмотр и редактирование текстовых файлов, подцветка синтаксиса, поддержка разных кодировок (включая Unicode)

2. Поиск и замена по множеству файлов, множественное переименование файлов, просмотр картинок, работа с архивами.

Утилиты

Термин «утилита» происходит от английского слова utility – полезный.

Утилиты можно рассматривать как «развитые» внешние команды операционной системы, имеющие хорошо организованный графический интерфейс, ориентированный на работу с мышью. Они служат для расширения возможностей ОС (предоставление различного сервиса), а их функции носят специализированный характер.

Системные утилиты – это обслуживающие программы вспомогательного назначения.

Утилиты дополняют возможности ОС, обеспечивая выполнение различных вспомогательных действий. Обычно некоторое количество утилит поставляется в составе соответствующей ОС, но немало утилит создано независимыми разработчиками и поставляется отдельно от ОС.

Утилиты часто используют низкоуровневые механизмы функционирования ОС, поэтому они могут работать только в тех ОС, на которые рассчитаны. Т.к. применение утилит в «чужой» для них ОС может привести, например, к уничтожению данных (это относится, прежде всего, к программам обслуживания дисков).

Примером может служить комплект стандартных утилит, встроенных в MSWindows(группа «Служебные»). Туда включен стандартный набор приложений, обеспечивающих выполнение следующих функций:

- проверка и восстановление сбойных дисков;
- оптимизация расположения файлов на диске (дефрагментация диска);
- получение информации о компьютере;
- восстановление файлов на диске;
- очистка диска и др.

К утилитам относят и два блока приложений: архиваторы и антивирусные пакеты.

Архиваторы

Архиватор (упаковщик) – программа, позволяющая за счет применения специальных методов сжатия информации создавать копии файлов меньшего размера, а также объединять копии нескольких файлов в один архивный файл.

Все существующие на сегодняшний день архиваторы можно разделить на три группы, которые можно условно назвать файловые, программные и дисковые.

1. Файловые архиваторы – позволяют упаковывать один или несколько файлов в единый архив. Размер архива, как правило, меньше чем суммарный размер исходных файлов. Воспользоваться архивными данными и программами пока они находятся в архиве нельзя. Для распаковки архива требуется разархиватор, который совмещен с архиватором в одной программе.

Кроме этого практически в любой программе архиваторе имеется возможность создания самораспаковывающихся файлов, который имеет расширение exe. Он содержит кроме упакованных данных разархивирующий модуль. (Rar, Zip, Ice, Ain)

2. Программные архиваторы – позволяют упаковать за один прием один единственный файл – выполняемую программу exe типа, которая при запуске самораспаковывается в оперативной памяти и тут же начинает работу. Программа становиться в два раза меньше и при этом сохраняет работоспособность. (LZEXE – UNLZEXE, EXEPACK — UPACKEXE)

3. Дисковый архиватор – представляет собой резидентный драйвер, который незаметно для пользователя архивирует любую записываемую на диск информацию и распаковывает ее обратно при чтении. При этом на диске создается огромный архив, который отображается как еще один логический раздел винчестера.

Принцип работы архиватора состоит в том, что программа ищет повторяющиеся фрагменты в файлах, после чего все найденные повторения заменяются ссылками на первые фрагменты. При этом, записывая информацию подобным образом, архиватор

обязательно должен запомнить, что и откуда он «отрезал», что и куда он «приклеил», и что за чем стоит в этой очереди.

Естественно, для различных типов файлов степень архивации будет разная. Файлы, содержащие текстовые данные, сжимаются максимально. Файлы-программы имеют очень маленькую степень сжатия из-за малого количества повторяющихся значений. Графические файлы, содержащие простые графические объекты черно-белого цвета, можно сжать в пять-шесть раз. Графические файлы типа TIFF Compressed не сжимаются вообще, потому что сжимаются при создании.

Хотя изначально сложно определить процент сжатия, но все же использование архиватора дает положительный результат, если вы пытаетесь экономно расходовать место на диске.

Самыми популярными программами архивирования, пожалуй, можно считать ZIP, RAR, ARJ. Основным недостатком архивов является невозможность прямого доступа к данным. Их сначала необходимо извлечь из архива или распаковать. Операция распаковки, впрочем, как и упаковки, требует некоторых системных ресурсов. Это не мгновенная операция. Поэтому архивы в основном применяют со сравнительно редко используемыми данными. Например, для хранения резервных копий или установочных файлов.

Методы сжатия архиваторов

Разработано большое количество разнообразных методов, их модификаций и подвидов для сжатия данных. Современные архиваторы, как правило, одновременно используют несколько методов одновременно. Можно выделить некоторые основные.

Кодирование длин серий (RLE - сокращение от run - length encoding - кодирование длин серий).

Очень простой метод. Последовательная серия одинаковых элементов данных заменяется на два символа: элемент и число его повторений. Широко используется как дополнительный, так и промежуточный метод. В качестве самостоятельного метода применяется, например, в графическом формате BMP .

Словарный метод (LZ - сокращение от Lempel Ziv - имена авторов).

Наиболее распространенный метод. Используется словарь, состоящий из последовательностей данных или слов. При сжатии эти слова заменяются на их коды из словаря. В наиболее распространенном варианте реализации в качестве словаря выступает сам исходный блок данных.

Основным параметром словарного метода является размер словаря. Чем больше словарь, тем больше эффективность. Однако для неоднородных данных чрезмерно большой размер может быть вреден, так как при резком изменении типа данных словарь будет заполнен неактуальными словами. Для эффективной работы данного метода при сжатии требуется дополнительная память. Приблизительно на порядок больше, чем нужно для исходных данных словаря. Существенным преимуществом словарного метода является простая и быстрая процедура распаковки. Дополнительная память при этом не требуется. Такая особенность особенно важна, если необходим оперативный доступ к данным.

Энтропийный метод (Huffman - кодирование Хаффмена, Arithmetic coding - арифметическое кодирование)

В этом методе элементы данных, которые встречаются чаще, кодируются при сжатии более коротким кодом, а более редкие элементы данных кодируются более длинным кодом. За счет того, что коротких кодов значительно больше, общий размер получается меньше исходного.

Широко используется как дополнительный метод. В качестве самостоятельного метода применяется, например, в графическом формате JPG .

Метод контекстного моделирования (CM - сокращение от context modeling - контекстное моделирование)

В этом методе строится модель исходных данных. При сжатии очередного элемента данных эта модель выдает свое предсказание или вероятность. Согласно этой вероятности, элемент данных кодируется энтропийным методом. Чем точнее модель будет соответствовать исходным данным, тем точнее она будет выдавать предсказания, и тем короче будут кодироваться элементы данных.

Для построения эффективной модели требуется много памяти. При распаковке приходится строить точно такую же модель. Поэтому скорость и требования к объему оперативной памяти для упаковки и распаковки почти одинаковы. В данный момент методы контекстного моделирования позволяют получить наилучшую степень сжатия, но отличаются чрезвычайно низкой скоростью.

PPM (PPM - Prediction by Partial Matching - предсказание по частичному совпадению).

Это особый подвид контекстного моделирования. Предсказание выполняется на основании определенного количества предыдущих элементов данных. Основным параметром является порядок модели, который задает это количество элементов. Чем больше порядок модели, тем выше степень сжатия, но требуется больше оперативной памяти для хранения данных модели. Если оперативной памяти недостаточно, то такая модель с большим порядком показывает низкие результаты. Метод PPM особенно эффективен для сжатия текстовых данных.

Предварительные преобразования или фильтрация.

Данные методы служат не для сжатия, а для представления информации в удобном для дальнейшего сжатия виде. Например, для несжатых мультимедиа данных характерны плавные изменения уровня сигнала. Поэтому для них применяют дельта-преобразование, когда вместо абсолютного значения берется относительное. Существуют фильтры для текста, исполняемых файлов, баз данных и другие.

Метод сортировки блока данных (BWT - сокращение от Burrows Wheeler Transform - по имени авторов).

Это особый вид или группа преобразований, в основе которых лежит сортировка. Такому преобразованию можно подвергать почти любые данные. Сортировка производится над блоками, поэтому данные предварительно разбиваются на части. Основным параметром является размер блока, который подвергается сортировке. Для распаковки данных необходимо проделать почти те же действия, что и при упаковке. Поэтому скорость и требования к оперативной памяти почти одинаковы. Архиваторы, которые используют данный метод, обычно показывают высокую скорость и степень сжатия для текстовых данных.

Непрерывные блоки или непрерывный режим (Solid mode - непрерывный режим).

Во многих методах сжатия начальный участок данных или файла кодируется плохо. Например, в словарном методе словарь пуст. В методе контекстного моделирования модель не построена. Когда количество файлов большое, а их размер маленький, общая степень сжатия значительно ухудшается за счет этих начальных участков. Чтобы этого не происходило при переходе на следующий файл, используется информация, полученная исходя из предыдущих файлов. Аналогичного эффекта можно добиться простым представлением исходных файлов в виде одного непрерывного файла.

Этот метод используется во многих архиваторах и имеет существенный недостаток. Для распаковки произвольного файла необходимо распаковать и файлы, которые оказались в начале архива. Это необходимо для правильного заполнения словаря или построения модели. Существует и промежуточный вариант, когда используются непрерывные блоки фиксированного размера. Потери сжатия получаются минимальными, но для извлечения одного файла, который находится в конце большого архива, необходимо распаковать только один непрерывный блок, а не весь архив.

Сегментирование.

Во всех методах сжатия при изменении типа данных собственно сам переход кодируется очень плохо. Словарь становится не актуальным, модель настроена на другие данные. В этих случаях применяется сегментирование. Это предварительная разбивка на однородные части. Затем эти части кодируются по отдельности или группами.

Особо хочется подчеркнуть, что существует большое количество методов сжатия. Каждый метод обычно ориентирован на один вид или группу реальных данных. Хорошие результаты показывает комплексное использование методов.

2.5 Векторные и растровые форматы

Компьютерная графика представляет собой одну из современных технологий создания различных изображений с помощью аппаратных и программных средств компьютера, отображения их на экране монитора и затем сохранения в файле или печати на принтере.

Существует два способа представления графических изображений: **растровый** и **векторный.** Соответственно различают растровый и векторный форматы графических файлов, содержащих информацию графического изображения.

Растровые форматы хорошо подходят для изображений со сложными гаммами цветов, оттенков и форм. Это такие изображения, как фотографии, рисунки, отсканированные данные. Векторные форматы хорошо применимы для чертежей и изображений с простыми формами, тенями и окраской.

Растровая графика. Наиболее просто реализовать растровое представление изображения. **Растр**, или растровый массив (bitmap), представляет совокупность битов, расположенных на сетчатом поле-канве. Бит может быть включен (единичное состояние) или выключен (нулевое состояние). Состояния битов можно использовать для представления черного или белого цветов, так что, соединив на канве несколько битов, можно создать изображение из черных и белых точек.

Растровое изображение напоминает лист клетчатой бумаги, на котором каждая клеточка закрашена черным или белым цветом, в совокупности формируя рисунок.

Основным элементом растрового изображения является **пиксель (pixel**). Под этим термином часто понимают несколько различных понятий: отдельный элемент растрового изображения, отдельная точка на экране монитора, отдельная точка на изображении, напечатанном принтером. Поэтому на практике эти понятия часто обозначают так:

Пиксель - отдельный элемент растрового изображения;

видеопиксель - элемент изображения на экране монитора;

точка - отдельная точка, создаваемая принтером или фотонаборным автоматом. Цвет каждого пикселя растрового изображения - черный, белый, серый или любой из спектра - запоминается с помощью комбинации битов. Чем больше битов используется для этого, тем большее количество оттенков цветов для каждого пикселя можно получить. Число битов, используемых компьютером для хранения информации о каждом пикселе, называется битовой глубиной или глубиной цвета.

Наиболее простой тип растрового изображения состоит из пиксель, имеющих два возможных цвета - **черный и белый**. Для хранения такого типа пиксель требуется один бит в памяти компьютера, поэтому изображения, состоящие из пиксель такого вида, называются 1-битовыми изображениями. Для отображения большего количества цветов используется больше битов информации. Число возможных и доступных цветов или градаций серого цвета каждого пикселя равно двум в степени, равной количеству битов, отводимых для каждого пикселя. 24 бита обеспечивают более 16 миллионов цветов. О 24-битовых изображениях часто говорят как об изображениях с естественными цветами, так как такого количества цветов более чем достаточно, чтобы отобразить всевозможные цвета, которые способен различать человеческий глаз.

Основной недостаток растровой графики состоит в том, что каждое изображение для своего хранения требует большое количество памяти. Простые растровые картинки,

такие как копии экрана компьютера или черно-белые изображения, занимают до нескольких сотен килобайтов памяти. Детализированные высококачественные рисунки, например, сделанные с помощью сканеров с высокой разрешающей способностью, занимают уже десятки мегабайтов.

Для разрешения проблемы обработки объемных (в смысле затрат памяти) изображений используются два основных способа:

1. увеличение памяти компьютера;

2. сжатие изображений.

Другим недостатком растрового представления изображений является снижение качества изображений при масштабировании.

Векторная графика. Векторное представление, в отличие от растровой графики, определяет описание изображения в виде линий и фигур, возможно, с закрашенными областями, заполняемыми сплошным или градиентным цветом. Хотя это может показаться более сложным, чем использование растровых массивов, но для многих видов изображений использование математических описаний является более простым способом.

В векторной графике для описания объектов используются комбинации компьютерных команд и математических формул для описания объектов. Это позволяет различным устройствам компьютера, таким как монитор и принтер, при рисовании этих объектов вычислять, где необходимо помещать реальные точки.

Векторную графику часто называют **объектно-ориентированной** или **чертежной графикой.** Имеется ряд простейших объектов, или примитивов, например: эллипс, прямоугольник, линия. Эти примитивы и их комбинации используются для создания более сложных изображений. Если посмотреть содержание файла векторной графики, обнаруживается сходство с программой. Он может содержать команды, похожие на слова, и данные в коде АЗСИ, поэтому векторный файл можно отредактировать с помощью текстового редактора. Приведем в условном упрощенном виде команды, описывающие окружность:

объект — окружность;

центр — 50, 70; радиус — 40;

линия: цвет — черный, толщина — 0.50;

заливка — нет.

Данный пример показывает основное достоинство векторной графики — описание объекта является простым и занимает мало памяти. Для описания этой же окружности средствами растровой графики потребовалось бы запомнить каждую отдельную точку изображения, что заняло бы гораздо больше памяти.

Кроме того, векторная графика в сравнении с растровой имеет следующие преимущества:

- простота масштабирования изображения без ухудшения его качества;

- независимость объема памяти, требуемой для хранения изображения, от выбранной цветовой модели.

Недостатком векторных изображений является их некоторая искусственность, заключающаяся в том, что любое изображение необходимо разбить на конечное множество составляющих его примитивов.

Растровая и векторная графика существуют не обособлено друг от друга. Так, векторные рисунки могут включать в себя и растровые изображения. Кроме того, векторные и растровые изображения могут быть преобразованы друг в друга — в этом случае говорят о конвертации графических файлов в другие форматы. Достаточно просто выполняется преобразование векторных изображений в растровые. Не всегда осуществимо преобразование растровой графики в векторную, так как для этого растровая картинка должна содержать линии, которые могут быть идентифицированы программой конвертации (типа CorelTrace в составе пакета CorelDraw) как векторные примитивы. Это

касается, например, высококачественных фотографий, когда каждый пиксель отличается от соседних.

Разрешающая способность - это количество элементов в заданной области. Этот термин применим ко многим понятиям, например, таким как:

- разрешающая способность графического изображения;
- разрешающая способность принтера как устройства вывода;
- разрешающая способность мыши как устройства ввода.

Например, разрешающая способность лазерного принтера может быть задана 300 dpi (dot per inche - точек на дюйм}, что означает способность принтера напечатать на отрезке в один дюйм 300 отдельных точек. В этом случае элементами изображения являются лазерные точки, а размер изображения измеряется в дюймах.

Разрешающая способность графического изображения измеряется в пикселях на дюйм. Отметим, что пиксель в компьютерном файле не имеет определенного размера, так как храпит лишь информацию о своем цвете. Физический размер пиксель приобретает при отображении на конкретном устройстве вывода, например, на мониторе или принтере.

Разрешающая способность технических устройств по-разному влияет на вывод векторной и растровой графики.

Так, при выводе векторного рисунка используется максимальное разрешение устройства вывода. При этом команды, описывающие изображение, сообщают устройству вывода положение и размеры какого-либо объекта, а устройство для его прорисовки использует максимально возможное количество точек. Таким образом, векторный объект, например, окружность, распечатанная на принтерах разного качества, имеет на листе бумаги одинаковые положение и размеры. Однако более гладко окружность выглядит при печати на принтере с большей разрешающей способностью, так как состоит из большего количества точек принтера.

Значительно большее влияние разрешающая способность устройства вывода оказывает на вывод растрового рисунка. Если в файле растрового изображения не определено, сколько пиксель на дюйм должно создавать устройство вывода, то по умолчанию для каждого пикселя используется минимальный размер. В случае лазерного принтера минимальным элементом служит лазерная точка, в мониторе - видеопиксель. Так как устройства вывода отличаются размерами минимального элемента, который может быть ими создан, то размер растрового изображения при выводе на различных устройствах также будет неодинаков.

Масштабирование изображений. Масштабирование заключается в изменении вертикального и горизонтального размеров изображения. Масштабирование может быть пропорциональным — в этом случае соотношение между высотой и шириной рисунка не изменяется, а меняется общий размер, и непропорциональным — в этом случае оба измерения изменяются по-разному.

Масштабирование векторных рисунков выполняется просто и без потери качества. Так как объекты векторной графики создаются по их описаниям, то для изменения масштаба векторного объекта, достаточно изменить его описание. Например, чтобы увеличить в два раза векторный объект, следует удвоить значение, описывающее его размер.

Масштабирование растровых рисунков является намного более сложным процессом, чем для векторной графики, и часто сопровождается потерей качества. При изменении размеров растрового изображения выполняется одно из следующих действий:

- одновременное изменение размеров всех пиксель (в большую или меньшую сторону);
- добавление или убавление пиксель из рисунка для отражения производимых в нем изменений, называемое выборкой пиксель в изображении.

Простейший способ изменения масштаба растрового рисунка состоит в изменении размера всех его пиксель. Так как внутри самого рисунка пиксели не имеют размера и

приобретают его уже при выводе на внешнее устройство, то изменение размера пиксель растра в сильной степени похоже на масштабирование векторных объектов — необходимо сменить только описание пикселя, а остальное выполнит устройство вывода.

Устройство вывода для создания пикселя определенного физического размера использует столько своих минимальных элементов (лазерных точек — для лазерного принтера, видеопиксель — для монитора), сколько сможет. При масштабировании изображения количество входящих в него пиксель не меняется, а изменяется количество создаваемых устройством вывода элементов, идущих на построение отдельного пикселя изображения. На рисунке 8 показан пример масштабирования растрового изображения — увеличения его в два раза по каждому измерению.

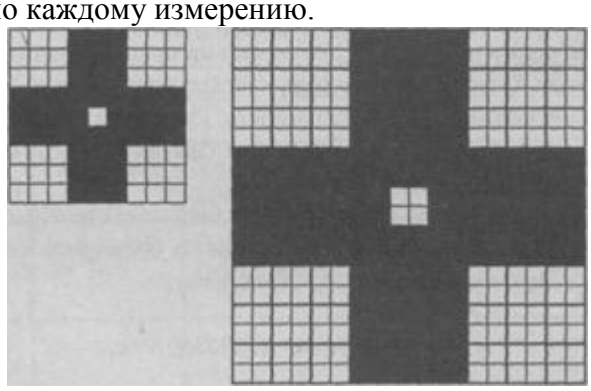


Рисунок 8 - Масштабирование растрового изображения

Выборка растрового рисунка может быть сделана двумя различными способами.

1) По первому способу просто дублируется или удаляется необходимое количество пиксель. При этом в результате масштабирования, как правило, ухудшается качество изображения. Например, при увеличении размера рисунка возрастают его зернистость и дискретность. При уменьшении размера рисунка потери в качестве не столь заметны, однако при последующем восстановлении уменьшенного рисунка до прежнего размера опять возрастают зернистость и дискретность. Это связано с тем, что при уменьшении размера рисунка часть пиксель была удалена из исходного изображения и потеряна безвозвратно, а при последующем восстановлении размеров рисунка недостающие пиксели дублировались из соседних.

2) По второму способу с помощью определенных вычислений можно создать пиксели другого цвета, определяемого цветами первоначального пикселя и его окружения. Этот метод называется интерполяцией и является более сложным, чем простое дублирование. При интерполяции кроме дублируемых пиксель, отбираются и соседние с ними, с помощью которых вновь создаваемые пиксели получают от существующих усредненный цвет или оттенок серого. В результате переходы между пикселями становятся более плавными, что позволяет убрать или уменьшить эффект «пилообразности» изображения.

2.6 Реляционная модель базы данных

Почти все современные системы основаны на **реляционной** (relational) модели управления базами данных. Название **реляционная** связано с тем, что каждая запись в такой базе данных содержит информацию, относящуюся только к одному конкретному объекту.

В **реляционной** СУБД все обрабатываемые данные представляются в виде плоских таблиц. Информация об объектах определенного вида представляется в табличном виде: в столбцах таблицы сосредоточены различные атрибуты объектов, а строки предназначены для сведения описаний всех атрибутов к отдельным экземплярам объектов.

Модель, созданная на этапе инфологического моделирования, в наибольшей степени удовлетворяет принципам реляционности. Однако для приведения этой модели к реляционной необходимо выполнить процедуру, называемую **нормализацией**.

Теория нормализации оперирует с пятью **нормальными формами**. Эти формы предназначены для уменьшения избыточности информации, поэтому каждая

последующая нормальная форма должна удовлетворять требованиям предыдущей и некоторым дополнительным условиям. При практическом проектировании баз данных четвертая и пятая формы, как правило, не используются. Мы ограничились рассмотрением первых четырех нормальных форм.

Введем понятия, необходимые для понимания процесса приведения модели к реляционной схеме.

Отношение - абстракция описываемого объекта как совокупность его свойств. Проводя инфологический этап проектирования, мы говорили об абстракции объектов и приписывали им некоторые свойства. Теперь же, проводя концептуальное проектирование, мы переходим к следующему уровню абстракции. На данном этапе объектов, как таковых, уже не существует. Мы оперируем совокупностью свойств, которые и определяют объект.

Экземпляр отношения - совокупность значений свойств конкретного объекта.

Первичный ключ - идентифицирующая совокупность атрибутов, т.е. значение этих атрибутов уникально в данном отношении. Не существует двух экземпляров отношения содержащих одинаковые значения в первичном ключе.

Простой атрибут - атрибут, значения которого неделимы.

Сложный атрибут - атрибут, значением которого является совокупность значений нескольких различных свойств объекта или несколько значений одного свойства.

Требования к реляционным моделям

Рациональные варианты концептуальной схемы базы данных должны удовлетворять третьей нормальной форме, а также следующим требованиям:

- Выбранный перечень отношений должен быть минимален. Отношение используется, если только его необходимость обусловлена задачами.
- Выбранный перечень атрибутов должен быть минимален. Атрибут включается в отношение только в том случае, если он будет использоваться.
- Первичный ключ отношения должен быть минимальным. То есть невозможно исключить ни один атрибут из идентифицирующей совокупности атрибутов, не нарушив при этом однозначной идентификации.
- При выполнении операций над данными не должно возникать трудностей.

Графическая интерпретация реляционной схемы

Концептуальная модель, реализованная в виде реляционной схемы, имеет свои правила графического представления.

- Отношение представляется в виде полоски, содержащей имена всех атрибутов. Имя отношения пишется над ней.
- Первичный ключ отношения должен быть выделен жирной рамкой.
- Связи, определенные между отношениями, должны быть показаны линиями, проведенными между связующими атрибутами. Значения экземпляров связующих атрибутов должны совпадать.

Реляционная модель не единственный метод хранения и манипулирования данными. Существуют альтернативные варианты: иерархическая, сетевая, а также звездообразная модели данных. У каждой из них свои преимущества при решении задач определенного типа. Например, применение реляционной модели в обработке данных с иерархической организацией недостаточно хорошо изучено, для решения подобных задач используют специально созданную звездообразную модель данных. Однако гибкость и эффективность реляционной модели делают ее наиболее популярным инструментом для разработки баз данных. В этой книге мы будем рассматривать только реляционную модель, на которой основаны механизмы СУБД Microsoft , Jet и Microsoft SQL Server.

В общих чертах основные принципы реляционных систем баз данных можно сформулировать так:

*Все данные на концептуальном уровне представляются в виде упорядоченной организации, определенной в виде строк и столбцов и называемой отношением.

* Все значения являются скалярами. Это означает, что для любой строки и столбца любого отношения существует одно и только одно значение.

*Все операции выполняются над целым отношением, и результатом выполнения этих операций также является целое отношение.Этот принцип называется замыканием.

Если у вас имеется опыт работы с базами данных Microsoft Access, вы, конечно, догадались, что в данном случае представляет собой отношение - это набор записей или, в терминах SQL Server, набор результатов. Формулируя принципы реляционной модели, доктор Код выбрал термин «отношение» (relation), потому что он однозначен (в то время как, например, термин «таблица» имеет множество дополнительных значений). Весьма распространено следующее заблуждение: реляционная модель названа так потому, что она определяет отношения между таблицами. На самом деле название этой модели происходит от отношений, лежащих в ее основе. В рамках реляционной модели данные представлены в виде отношения на концептуальном уровне; однако при этом совсем не дается никаких указаний, каким образом данные будут реализованы на физическом уровне. Разделение концептуального и логического уровней давно стало привычным для нас; но всего лишь 30 лет назад этот метод произвел настоящий переворот в области программирования баз данных. Ранее программирование баз данных сводилось в основном к написанию программного кода для физического управления устройствами, предназначенными для хранения данных. В действительности отношения не нуждаются в физическом представлении. Некий набор записей может соответствовать некоей физической таблице, размещенной на диске, а может быть и сформирован из столбцов нескольких десятков таблиц с вычисляемыми полями, значения которых вообще нигде не хранятся. Такой набор записей является отношением, поскольку организован в виде строк и столбцов, и его значения - скаляры. Его существование абсолютно никак не зависит от физической реализации. Принцип замыкания заключается в том, что и базовые таблицы, и результаты операций над ними на концептуальном уровне представляются как отношения. Он позволяет непосредственно использовать результаты одной операции в качестве исходных данных для выполнения другой. Таким образом, и Microsoft Jet, и SQL Server дают возможность использовать результаты одного запроса для составления нового. Этот принцип реализует в области разработки баз данных функциональность, аналогичную подпрограммам в процедурном программировании - возможность инкапсуляции сложных или часто повторяющихся операций для повторного использования. Предположим, вы составили запрос с именем FullNameQuery, выполняющий операцию конкатенации над данными и помещающий имя и фамилию физического лица в вычисляемое поле FullName. Вы можете составить следующий запрос, в котором FullNameQuery будет выступать в качестве источника. Вычисляемое поле FullName используется в этом случае точно так же, как любое другое поле базовой таблицы. Заново выполнять вычисления, в результате которых будет получено имя физического лица, не нужно. Требование, чтобы все значения в отношении являлись скалярами, может иногда создавать дополнительные трудности и соблюдается не абсолютно строго. Принцип существования одного и только одного значения для любой строки и любого столбца субъективен и зависит от семантики модели данных. Например, имя и фамилия физического лица в одной модели могут быть представлены как одно значение, а в других моделях - разбиты на несколько отдельных значений (например, имя и фамилию или обращение, имя, отчество и фамилию). С точки зрения абстрактной теории ни один из этих вариантов не является более правильным, чем остальные; представление данных зависит от выбранного варианта реализации системы. Примерами зарубежных реляционных СУБД для ПЭВМ являются следу- ющие: (dBaseIII Plus и dBase IY ,DB2, Fox Pro более поздних версий, FoxBASE, Paradox и Acess, Clarion, Ingres(ASKComputer Systems) иОгас1е (Огас1е). К отечественным СУБД реляционного типа относятся системы: ПАЛЬМА (ИК АН УССР), а также система НуТech(МИФИ). Заметим, что последние версии реляционных СУБД имеют некоторые свойства объектно-ориентированных систем. Такие СУБД часто называют объектно-реляционными.

Примером такой системы можно считать продукты Orac1e 8.x. Системы предыдущих версий вплоть до Orac1e 7.x считаются «чисто» реляционными.

Элементы реляционной базы данных

Элемент реляционной модели	Форма представления
Отношение	Таблица
Схема отношения	Строка заголовков столбцов таблицы(заголовок таблицы)
Кортеж	Строка таблицы
Сущность	Описание свойств объекта
Атрибут	Заголовок столбца таблицы
Домен	Множество допустимых значений атрибута
Значение атрибута	Значение поля в записи
Первичный ключ	Один или несколько атрибутов
Тип данных	Тип значений элементов таблицы

Отношение является важнейшим понятием и представляет собой двумерную таблицу, содержащую некоторые данные. Сущность есть объект любой природы, данные о котором хранятся в базе данных. Данные о сущности хранятся в отношении. Атрибуты представляют собой свойства, характеризующие сущность. В структуре таблицы каждый атрибут именуется и ему соответствует заголовок некоторого столбца таблицы. На рис/ приведен пример представления отношения СОТРУДНИК. В общем случае порядок кортежей в отношении, как и в любом множестве, не определен. Однако в' реляционных СУБД для удобства кортежи все же упорядочивают. Чаще всего для этого выбирают некоторый атрибут, по которому система автоматически сортирует кортежи по возрастанию или убыванию. Если пользователь не назначает атрибута упорядочения, система автоматически присваивает номер кортежам в порядке их ввода.

ФИО	Отдел	Должность	Д.Р.
Иванов	002	Начальник	27,09,51
Петров	001	Заместитель	15,04,55
Сидоров	002	Инженер	13,01,30

Формально, если переставить атрибуты в отношении, то получается новое отношение. Однако в реляционных БД перестановка атрибутов не приводит к образованию нового отношения. Домен представляет собой множество всех возможных значений определенного атрибута отношения. Отношение СОТРУДНИК включает 4 домена. Домен 1 содержит фамилии всех сотрудников, домен 2 - номера всех отделов фирмы, домен 3 - названия всех должностей, домен 4 - даты рождения всех сотрудников. Каждый домен образует значения одного типа данных, например, числовые или символьные. Отношение СОТРУДНИК содержит 3 кортежа. Кортеж рассматриваемого отношения состоит из 4 элементов, каждый из которых выбирается из соответствующего домена. Каждому кортежу соответствует строка таблицы. Схема отношения (заголовок отношения) представляет собой список имен атрибутов. Например, для приведенного примера схема отношения имеет вид СОТРУДНИК (ФИО, Отдел, Должность, Дата Рождения). Множество собственно кортежей отношения часто называют содержимым (телом) отношения. Первичным ключом (ключом отношения, ключевым атрибутом) называется атрибут отношения, однозначно идентифицирующий каждый из его кортежей. Например, в отношении СОТРУДНИК (ФИО, Отдел, Должность, Дата Рождения)

ключевым является атрибут «ФИО». Ключ может быть составным (сложным), то есть состоять из нескольких атрибутов. Каждое отношение обязательно имеет комбинацию атрибутов, которая может служить ключом. Ее существование гарантируется тем, что отношение это множество, которое не содержит одинаковых элементов - кортежей. То есть в отношении нет повторяющихся кортежей, а это значит, что по крайней мере вся совокупность атрибутов обладает свойством однозначно идентификации кортежей отношения. Во многих СУБД допускается создавать отношения, не определяя ключи. Возможны случаи, когда отношение имеет несколько комбинаций атрибутов, каждая из которых однозначно определяет все кортежи отношения. Все эти комбинации атрибутов являются возможными ключами отношения. Любой из возможных ключей может быть выбран как первичный. Если выбранный первичный ключ состоит из минимально необходимого набора атрибутов, говорят, что он является не избыточным. Ключи обычно используют для достижения следующих целей:

1) исключения дублирования значений в ключевых атрибутах (остальные атрибуты в расчет не принимаются);

2) упорядочения кортежей. Возможно упорядочение по возрастанию или убыванию значений всех ключевых атрибутов, а также смешанное упорядочение (по одним - возрастание, а по другим - убывание);

3) ускорения работы к кортежами отношения

4) организации связывания таблиц

Реляционная модель накладывает на внешние ключи ограничение для обеспечения целостности данных, называемое ссылочной целостностью. Это означает, что каждому значению внешнего ключа должны соответствовать строки в связываемых отношениях. Поскольку не всякой таблице можно поставить в соответствие отношение, приведем условия, выполнение которых позволяет таблицу считать отношением.

1. Все строки таблицы должны быть уникальны, то есть не может быть строк с одинаковыми первичными ключами.

2. Имена столбцов таблицы должны быть различны, а значения их простыми, то есть недопустима группа значений в одном столбце одной строки.

3. Все строки одной таблицы должны иметь одну структуру, соответствующую именам и типам столбцов.

4. Порядок размещения строк в таблице может быть произвольным. Наиболее часто таблица с отношением размещается в отдельном файле.

Операции, выполняемые над отношениями, можно разделить на две группы. Первую группу составляют операции над множествами, к которым относятся операции: объединения, пересечения, разности, деления и декартова произведения. Вторую группу составляют специальные операции над отношениями, к которым, в частности, относятся операции: проекции, соединения, выбора. В различных СУБД реализована некоторая часть операций над отношениями, определяющая в какой-то мере возможности данной СУБД и сложность реализации запросов к БД. В реляционных СУБД для выполнения операций над отношениями используются две группы языков, имеющие в качестве своей математической основы теоретические языки запросов, предложенные Э. Коддом:

* реляционная алгебра;

* реляционное исчисление.

Эти языки представляют минимальные возможности реальных языков манипулирования данными в соответствии с реляционной моделью и эквивалентны друг другу по своим выразительным возможностям. Существуют не очень сложные правила преобразования запросов между ними. В реляционной алгебре операнды и результаты всех действий являются отношениями.

Языки реляционной алгебры являются процедурными, так как отношение, являющееся результатом запроса к реляционной БД, вычисляется при выполнении последовательности реляционных операторов, применяемых к отношениям. Операторы

состоят из операндов, в роли которых выступают отношения, и реляционных операций. Результатом реляционной операции является отношение.

Языки исчислений, в отличие от реляционной алгебры, являются непроцедурными (описательными, или декларативными) и позволяют выражать запросы с помощью предиката первого порядка (высказывания в виде функции), которому должны удовлетворять кортежи или домены отношений. Запрос к БД, выполненный с использованием подобного языка, содержит лишь информацию о желаемом результате. Для этих языков характерно наличие наборов правил для записи запросов. В частности, к языкам этой группы относится SQL.

Достоинство реляционной модели данных заключается в простоте, понятности и удобстве физической реализации на ЭВМ. Именно простота и понятность для пользователя явились основной причиной их широкого использования. Проблемы же эффективности обработки данных этого типа оказались технически вполне разрешимыми. При проектировании реляционной БД применяются строгие правила, базирующие на математическом аппарате. Полная независимость данных. При изменении структуры реляционной изменения, которые требуют произвести в прикладных программах, минимальны.

Для построения запросов и написания прикладных программ нет необходимости знания конкретной организации БД во внешней памяти.

Основными недостатками реляционной модели являются следующие: отсутствие стандартных средств идентификации отдельных записей и сложность описания иерархических и сетевых связей. Относительно низкая скорость доступа и большой объем внешней памяти. Трудность понимания структуры данных из-за появления большого кол-ва таблиц в результате логического проектирования. Далеко не всегда предметную область можно представить в виде совокупности таблиц.

2.7 Программы циклической структуры

При решении подавляющего большинства задач (в том числе и весьма несложных) в программе практически невозможно задать в явном виде все операции, которые необходимо выполнить. В самом деле, пусть необходимо вычислить сумму первых n членов гармонического ряда: Y= 1+ 1/2+ 1/3 + …+ 1/ n

Если значение n не фиксируется, а является исходным данным, вводимым в процессе выполнения программы (и даже константой, описанной в программе), то аналогичный оператор присваивания записать невозможно. Ибо запись вида Y:= 1+1/2+1/3+…+1/ n в языках программирования недопустима.

Для устранения возникающих трудностей служат операторы цикла. Они позволяют повторять выполнение отдельных частей программы. Можно выделить четыре оператора цикла:

- простой арифметический оператор цикла (цикл с параметром с шагом 1),
- сложный арифметический оператор цикла (цикл с параметром произвольного шага),
- итерационный оператор цикла с предусловием,
- итерационный оператор цикла с постусловием.

Простой арифметический оператор цикла Паскаля (цикл с параметром)

Вернемся к рассмотренной выше задаче вычисления суммы первых n членов гармонического ряда, правила которой невозможно задать в виде арифметического выражения, если значение n заранее не фиксировано.

На самом деле вычисление этой суммы можно осуществить по очень простому и компактному алгоритму: предварительно положим y=0 (с помощью оператора присваивания y:=0), а затем выполним оператор присваивания y:= y+1/ i для последовательных значений i= 1,2,…, n. При каждом очередном выполнении этого оператора к текущему значению y будет прибавляться очередное слагаемое. Как видно, в

этом случае процесс вычислений будет носить циклический характер: оператор y:= y+1/i должен выполняться многократно, т.е. циклически, при различных значениях i.

Этот пример циклического вычислительного процесса является весьма типичным; его характерные особенности состоят в том, что

- число повторений цикла известно к началу его выполнения (в данном случае оно равно значению n, которое предполагается заданным к этому времени);
- управление циклом осуществляется с помощью переменной порядкового типа, которая в этом циклическом процессе принимает последовательные значения от заданного начального до заданного конечного значений (в нашем случае – это целочисленная переменная i, принимающая последовательные значения от 1 до n).

Для компактного задания подобного рода вычислительных процессов и служит оператор цикла с параметром. Чаще всего используется следующий вид этого оператора В Паскале:

For V:= E1 to E2 do S,

где for (для), to (увеличиваясь к) и do (выполнять, делать) – служебные слова, V – переменная порядкового типа, называемая параметром цикла, E1 и E2 – выражения того же типа, что и параметр цикла, S – оператор, который и выполняется многократно в цикле, называемый телом цикла.

Заметим, что в Паскале после do должен стоять один оператор, если необходимо выполнить несколько действий, то они должны быть объединены в один составной оператор путем заключения в операторные скобки.

Этот оператор цикла Паскаля предусматривает присваивание параметру цикла V последовательных значений от начального значения, равного значению выражения E1, до конечного значения, равного значению выражения E2, т.е. при каждом повторении выполняется оператор присваивания V:= succ(V), и выполнение оператора S при каждом значении параметра цикла V. При этом значения выражений E1 и E2 вычисляются один раз, при входе в оператор цикла, а значение параметра цикла V не должно изменяться в результате выполнения оператора S. Если заданное конечное значение меньше начального значения (что допустимо), то оператор S не выполняется ни разу.

В Паскале считается, что при нормальном завершении выполнения оператора цикла значение параметра цикла не определено.

С использованием оператора цикла с параметром алгоритм вычисления суммы первых n членов гармонического ряда может быть задан следующим образом:

Пример кода программы для суммирования первых n членов гармонического ряда

```
Readln(n); Y:= 0;
For i:= 1 to n do y:= y+1/i;
```

В некоторых случаях бывает удобно, чтобы параметр цикла Паскаля принимал последовательные, но не возрастающие, а убывающие значения. Для таких случаев в Паскале предусмотрен оператор цикла с параметром следующего вида:

For V:= E1 downto E2 do S,

где downto (уменьшаясь к) – служебное слово, а все остальные слова и выражения имеют прежний смысл. Изменение параметра цикла от большего значения к меньшему происходит при выполнении присваивания V:=pred(V). Заметим, что начальное значение может быть меньше конечного значения. В этом случае оператор S не выполнится ни разу. Значение параметра цикла по завершении выполнения такого цикла так же считается неопределенным.

Следует запомнить и то, что для обоих вариантов записи цикла с параметром справедливо: если начальное и конечное значения равны, то тело цикла (оператор S) выполнится один раз.

Арифметический оператор цикла Паскаля с произвольным шагом

Естественным усложнением простого арифметического цикла Паскаля, является цикл, в котором параметр цикла изменяется не на 1, а на произвольную величину – шаг

приращения. При этом в процессе выполнения цикла шаг изменяется по заданному закону. Стандартные операторы для реализации такого цикла есть в Форте, в других языках их приходится организовывать из простейшего арифметического цикла.

Итерационные операторы цикла Паскаля

Итерационные циклы отличаются от циклов с параметром тем, что в них заранее неизвестно число повторений.

Рассмотрим теперь математическую задачу. Пусть нам необходимо вычислить сумму первых членов гармонического ряда, удовлетворяющих условию 1/i>= e, где 0< e<1, а i=1,2,3…...Эту задачу можно решить по следующему алгоритму: положить предварительно y=0 и i=0, а затем в цикле увеличивать i на 1, к значению y добавлять очередное слагаемое 1/ i до тех пор, пока текущее значение 1/ i впервые окажется больше заданного значения 0< e<1.

Очевидно, что число повторений этого цикла заранее не известно. В подобного рода случаях мы можем лишь сформулировать условие, при выполнении которого процесс добавления к сумме очередного слагаемого должен завершиться.

Для задания таких вычислительных процессов и служит оператор цикла Паскаля с постусловием. Этот оператор имеет вид:

Repeat S1; S2;…; Si until B,

где repeat (повторять) и until (до) – служебные слова, через Si обозначен любой оператор Паскаля, а через B – логическое выражение.

При выполнении этого оператора цикла последовательность операторов, находящихся между словами repeat и until, выполнится один или более раз. Этот процесс завершается, когда после очередного выполнения заданной последовательности операторов логическое выражение B примет (впервые) значение true. Таким образом, с помощью логического выражения B задается условие завершения выполнения оператора цикла. Поскольку в данном случае проверка условия производится после выполнения последовательности операторов (тела цикла), этот оператор цикла и называется оператором цикла с постусловием.

С использованием этого вида оператора цикла Паскаля задача о суммировании первых членов гармонического ряда, удовлетворяющих заданному условию, может быть реализована следующим образом:

Заметим, что оператор цикла с постусловием является более общим, чем оператор цикла с параметром — любой циклический процесс, задаваемый с помощью цикла с параметром можно представить в виде цикла с постусловием. Обратное утверждение неверно. Например, задача о суммировании первых n членов гармонического ряда, рассмотренная ранее, с оператором цикла с постусловием будет выглядеть так:

Оператор цикла Паскаля с предусловием

В случае оператора цикла Паскаля с постусловием входящая в него последовательность операторов заведомо будет выполняться хотя бы один раз. Между тем довольно часто встречаются такие циклические процессы, когда число повторений цикла тоже неизвестно заранее, но при некоторых значениях исходных данных предусмотренные в цикле действия вообще не должны выполняться, и даже однократное выполнение этих действий может привести к неверным или неопределенным результатам.

Пусть, например, дано вещественное число M. Требуется найти наименьшее целое неотрицательное число k, при котором 3 k> M. Эту задачу можно решить по следующему алгоритму: предварительно положить y=1 и k=0; затем в цикле домножать значение y на 3 и увеличивать значение k на 1 до тех пор, пока текущее значение y впервые окажется больше значения M. На первый взгляд, здесь можно воспользоваться оператором цикла с постусловием:

Для задания подобного рода вычислительных процессов, когда число повторений цикла заранее неизвестно и действия, предусмотренные в цикле, могут вообще не

выполняться, и служит оператор цикла с предусловием. Этот оператор цикла имеет в Паскале следующий вид:

While B do S,

где while (пока), do (делать, выполнять) – служебные слова, B – логическое выражение, S – оператор. Здесь оператор S выполняется ноль или более раз, но перед каждым очередным его выполнением вычисляется значение выражения B, и оператор S выполняется только в том случае, когда значение выражения B true. Выполнение оператора цикла завершается, когда выражение B впервые принимает значение false. Если это значение выражение B принимает при первом же его вычислении, то оператор S не выполнится ни разу.

Оператор цикла Паскаля с предусловием можно считать наиболее универсальным – с использованием таких операторов можно задать и циклические процессы, определяемые операторами цикла с параметром и постусловием.

Отметим отличия и особенности хорошего стиля работы с рассмотренными циклическими операторами.

Цикл с предусловием While (пока условие истинно)	Цикл с постусловием Repeat (до истинности условия)
1. До начала цикла должны быть сделаны начальные установки переменных, управляющих условием цикла, для корректного входа в цикл	
2. В теле цикла должны присутствовать операторы, изменяющие переменные условия так, чтобы цикл через некоторое число итераций завершился	
3. Цикл работает пока условие истинно (пока True)	3. Цикл работает пока условие ложно (пока False)
4. Цикл завершается, когда условие становится ложным (до False)	4. Цикл завершается, когда условие становится истинным (до True)
5. Цикл может не выполниться ни разу, если исходное значение условия при входе в цикл False	5. Цикл обязательно выполнится как минимум один раз
6. Если в теле цикла требуется выполнить более одного оператора, то необходимо использовать составной оператор	6. Независимо от количества операторов в теле цикла, использование составного оператора не требуется
Цикл со счетчиком (с параметром) For	
• Начальная установка переменной счетчика цикла до заголовка не требуется	
• Изменение в теле цикла значений переменных, стоящих в заголовке не допускается	
• Количество итераций цикла неизменно и точно определяется значениями нижней и верхней границ и шага приращения	
• Нормальный ход работы цикла может быть нарушен оператором goto или процедурами Break и Continue	
• Цикл может не выполниться ни разу, если шаг цикла будет изменять значение счетчика от нижней границы в направлении, противоположном верхней границе	

Оператор, который выполняется в цикле, сам может быть циклом. Это относится ко всем видам циклов. В результате мы получаем вложенные циклы. Механизм работы вложенных циклов удобнее всего рассмотреть на примере вложенных циклов с параметром. Пусть нам нужно описать работу электронных часов, начиная с момента времени 0 часов, 0 минут, 0 секунд. Значение минут станет равным 1 только после того, как секунды «пробегут» все последовательные значения от 0 до 59. Часы изменят свое значение на 1 только после того, как минуты «пробегут» все последовательные значения

от 0 до 59. Таким образом, вывод всех значений времени от начала суток до конца суток может быть представлен следующим фрагментом программы:

```
For h:=0 to 23 do
For m:=0 to 59 do
For s:=0 to 59 do
Writeln(h,':',m,':',s);
```

Для удобства реализации циклических структур на Паскале в последних версиях языка введены операторы break и continue, применяемые внутри циклов. Они расширяют возможности использования циклов и улучшают структуру программы.

В процессе выполнения тела цикла до полного завершения цикла могут возникнуть дополнительные условия, требующие завершения цикла. В этом случае цикл может быть прекращен оператором break.

В ходе выполнения цикла может возникнуть условие, при котором необходимо пропустить все или некоторые действия, предусмотренные в цикле, не прекращая работу цикла совсем. Для этого используется оператор continue, который передает управление в ту точку программы, где проверяется условие продолжения или прекращения цикла.

2.8 Криптографическая защита информации

Проблема защиты информации путем ее преобразования, исключающего ее прочтение посторонним лицом, волновала человеческий ум с давних времен. История криптографии - ровесница истории человеческого языка. Более того, первоначально письменность сама по себе была криптографической системой, так как в древних обществах ею владели только избранные. Священные книги Древнего Египта, Древней Индии тому примеры.

Криптографические методы защиты информации – это специальные методы шифрования, кодирования или иного преобразования информации, в результате которого ее содержание становится недоступным без предъявления ключа криптограммы и обратного преобразования. Криптографический метод защиты, безусловно, самый надежный метод защиты, так как охраняется непосредственно сама информация, а не доступ к ней (например, зашифрованный файл нельзя прочесть даже в случае кражи носителя). Данный метод защиты реализуется в виде программ или пакетов программ.

Современная криптография включает в себя четыре крупных раздела:

1. *Симметричные криптосистемы*. В симметричных криптосистемах и для шифрования, и для дешифрования используется один и тот же ключ. (Шифрование - преобразовательный процесс: исходный текст, который носит также название открытого текста, заменяется шифрованным текстом, дешифрование - обратный шифрованию процесс. На основе ключа шифрованный текст преобразуется в исходный);
2. *Криптосистемы с открытым ключом*. В системах с открытым ключом используются два ключа - открытый и закрытый, которые математически связаны друг с другом. Информация шифруется с помощью открытого ключа, который доступен всем желающим, а расшифровывается с помощью закрытого ключа, известного только получателю сообщения. (Ключ - информация, необходимая для беспрепятственного шифрования и дешифрования текстов);
3. *Электронная подпись*. Системой электронной подписи. называется присоединяемое к тексту его криптографическое преобразование, которое позволяет при получении текста другим пользователем проверить авторство и подлинность сообщения.
4. *Управление ключами*. Это процесс системы обработки информации, содержанием которых является составление и распределение ключей между пользователями.

Основные направления использования криптографических методов - передача конфиденциальной информации по каналам связи (например, электронная почта),

установление подлинности передаваемых сообщений, хранение информации (документов, баз данных) на носителях в зашифрованном виде.

Требования к криптосистемам

Процесс криптографического закрытия данных может осуществляться как программно, так и аппаратно. Аппаратная реализация отличается существенно большей стоимостью, однако ей присущи и преимущества: высокая производительность, простота, защищенность и т.д. Программная реализация более практична, допускает известную гибкость в использовании. Для современных криптографических систем защиты информации сформулированы следующие общепринятые требования:

- зашифрованное сообщение должно поддаваться чтению только при наличии ключа;
- число операций, необходимых для определения использованного ключа шифрования по фрагменту шифрованного сообщения и соответствующего ему открытого текста, должно быть не меньше общего числа возможных ключей;
- число операций, необходимых для расшифровывания информации путем перебора всевозможных ключей должно иметь строгую нижнюю оценку и выходить за пределы возможностей современных компьютеров (с учетом возможности использования сетевых вычислений);
- знание алгоритма шифрования не должно влиять на надежность защиты;
- незначительное изменение ключа должно приводить к существенному изменению вида зашифрованного сообщения даже при использовании одного и того же ключа;
- структурные элементы алгоритма шифрования должны быть неизменными;
- дополнительные биты, вводимые в сообщение в процессе шифрования, должен быть полностью и надежно скрыты в шифрованном тексте;
- длина шифрованного текста должна быть равной длине исходного текста;
- не должно быть простых и легко устанавливаемых зависимостью между ключами, последовательно используемыми в процессе шифрования;
- любой ключ из множества возможных должен обеспечивать надежную защиту информации;
- алгоритм должен допускать как программную, так и аппаратную реализацию, при этом изменение длины ключа не должно вести к качественному ухудшению алгоритма шифрования.

Симметричные криптосистемы

Все многообразие существующих криптографических методов в симметричных криптосистемах можно свести к следующим 4 классам преобразований:

- подстановка - символы шифруемого текста заменяются символами того же или другого алфавита в соответствии с заранее определенным правилом;
- перестановка - символы шифруемого текста переставляются по некоторому правилу в пределах заданного блока передаваемого текста;
- аналитическое преобразование - шифруемый текст преобразуется по некоторому аналитическому правилу, например гаммирование - заключается в наложении на исходный текст некоторой псевдослучайной последовательности, генерируемой на основе ключа;
- комбинированное преобразование - представляют собой последовательность (с возможным повторением и чередованием) основных методов преобразования, применяемую к блоку (части) шифруемого текста. Блочные шифры на практике встречаются чаще, чем “чистые” преобразования того или иного класса в силу их более высокой криптостойкости. Российский и американский стандарты шифрования основаны именно на этом классе.

Системы с открытым ключом

Как бы ни были сложны и надежны криптографические системы - их слабое мест при практической реализации - проблема распределения ключей. Для того, чтобы был возможен обмен конфиденциальной информацией между двумя субъектами ИС, ключ должен быть сгенерирован одним из них, а затем каким-то образом опять же в конфиденциальном порядке передан другому. Т.е. в общем случае для передачи ключа опять же требуется использование какой-то криптосистемы. Для решения этой проблемы на основе результатов, полученных классической и современной алгеброй, были предложены системы с открытым ключом. Суть их состоит в том, что каждым адресатом ИС генерируются два ключа, связанные между собой по определенному правилу. Один ключ объявляется открытым, а другой закрытым. Открытый ключ публикуется и доступен любому, кто желает послать сообщение адресату. Секретный ключ сохраняется в тайне. Исходный текст шифруется открытым ключом адресата и передается ему. Зашифрованный текст в принципе не может быть расшифрован тем же открытым ключом. Дешифрование сообщения возможно только с использованием закрытого ключа, который известен только самому адресату. Криптографические системы с открытым ключом используют так называемые необратимые или односторонние функции, которые обладают следующим свойством: при заданном значении x относительно просто вычислить значение f(x), однако если y=f(x), то нет простого пути для вычисления значения x. Множество классов необратимых функций и порождает все разнообразие систем с открытым ключом. Однако не всякая необратимая функция годится для использования в реальных ИС. В самом определении необратимости присутствует неопределенность. Под необратимостью понимается не теоретическая необратимость, а практическая невозможность вычислить обратное значение используя современные вычислительные средства за обозримый интервал времени. Поэтому чтобы гарантировать надежную защиту информации, к системам с открытым ключом (СОК) предъявляются два важных и очевидных требования:

1. Преобразование исходного текста должно быть необратимым и исключать его восстановление на основе открытого ключа.
2. Определение закрытого ключа на основе открытого также должно быть невозможным на современном технологическом уровне. При этом желательна точная нижняя оценка сложности (количества операций) раскрытия шифра.

Алгоритмы шифрования с открытым ключом получили широкое распространение в современных информационных системах. Так, алгоритм RSA стал мировым стандартом де-факто для открытых систем. Вообще же все предлагаемые сегодня криптосистемы с открытым ключом опираются на один из следующих типов необратимых преобразований:

- Разложение больших чисел на простые множители;
- Вычисление логарифма в конечном поле;
- Вычисление корней алгебраических уравнений.

Здесь же следует отметить, что алгоритмы криптосистемы с открытым ключом (СОК) можно использовать в следующих назначениях:

1. Как самостоятельные средства защиты передаваемых и хранимых данных.
2. Как средства для распределения ключей.

Алгоритмы СОК более трудоемки, чем традиционные криптосистемы. Поэтому часто на практике рационально с помощью СОК распределять ключи, объем которых как информации незначителен. А потом с помощью обычных алгоритмов осуществлять обмен большими информационными потоками. Один из наиболее распространенных - система с открытым ключом - RSA. Криптосистема RSA, разработанная в 1977 году и получила название в честь ее создателей: Рона Ривеста, Ади Шамира и Леонарда Эйдельмана. Они воспользовались тем фактом, что нахождение больших простых чисел в вычислительном отношении осуществляется легко, но разложение на множители произведения двух таких чисел практически невыполнимо. Доказано (теорема Рабина), что раскрытие шифра RSA эквивалентно такому разложению. Поэтому для любой длины ключа можно дать нижнюю

оценку числа операций для раскрытия шифра, а с учетом производительности современных компьютеров оценить и необходимое на это время. Возможность гарантированно оценить защищенность алгоритма RSA стала одной из причин популярности этой СОК на фоне десятков других схем. Поэтому алгоритм RSA используется в банковских компьютерных сетях, особенно для работы с удаленными клиентами (обслуживание кредитных карточек).

Электронная подпись

В чем состоит проблема аутентификации данных? В конце обычного письма или документа исполнитель или ответственное лицо обычно ставит свою подпись. Подобное действие обычно преследует две цели. Во-первых, получатель имеет возможность убедиться в истинности письма, сличив подпись с имеющимся у него образцом. Во-вторых, личная подпись является юридическим гарантом авторства документа. Последний аспект особенно важен при заключении разного рода торговых сделок, составлении доверенностей, обязательств и т.д. Если подделать подпись человека на бумаге весьма непросто, а установить авторство подписи современными криминалистическими методами - техническая деталь, то с подписью электронной дело обстоит иначе. Подделать цепочку битов, просто ее скопировав, или незаметно внести нелегальные исправления в документ сможет любой пользователь. С широким распространением в современном мире электронных форм документов (в том числе и конфиденциальных) и средств их обработки особо актуальной стала проблема установления подлинности и авторства безбумажной документации. В разделе криптографических систем с открытым ключом было показано, что при всех преимуществах современных систем шифрования они не позволяют обеспечить аутентификацию данных. Поэтому средства аутентификации должны использоваться в комплексе и криптографическими алгоритмами.

Управление ключами

Кроме выбора подходящей для конкретной ИС криптографической системы, важная проблема - управление ключами. Как бы ни была сложна и надежна сама криптосистема, она основана на использовании ключей. Если для обеспечения конфиденциального обмена информацией между двумя пользователями процесс обмена ключами тривиален, то в ИС, где количество пользователей составляет десятки и сотни управление ключами - серьезная проблема. Под ключевой информацией понимается совокупность всех действующих в ИС ключей. Если не обеспечено достаточно надежное управление ключевой информацией, то завладев ею, злоумышленник получает неограниченный доступ ко всей информации. Управление ключами - информационный процесс, включающий в себя три элемента:

- генерацию ключей;
- накопление ключей;
- распределение ключей.

Рассмотрим, как они должны быть реализованы для того, чтобы обеспечить безопасность ключевой информации в ИС.

Генерация ключей

В самом начале разговора о криптографических методах было сказано, что не стоит использовать неслучайные ключи с целью легкости их запоминания. В серьезных ИС используются специальные аппаратные и программные методы генерации случайных ключей. Как правило используют датчики ПСЧ. Однако степень случайности их генерации должна быть достаточно высоким. Идеальным генераторами являются устройства на основе “натуральных” случайных процессов. Например случайным математическим объектом являются десятичные знаки иррациональных чисел, которые вычисляются с помощью стандартных математических методов.

Под накоплением ключей понимается организация их хранения, учета и удаления. Поскольку ключ является самым привлекательным для злоумышленника объектом,

открывающим ему путь к конфиденциальной информации, то вопросам накопления ключей следует уделять особое внимание. Секретные ключи никогда не должны записываться в явном виде на носителе, который может быть считан или скопирован. В достаточно сложной ИС один пользователь может работать с большим объемом ключевой информации, и иногда даже возникает необходимость организации мини-баз данных по ключевой информации. Такие базы данных отвечают за принятие, хранение, учет и удаление используемых ключей. Итак, каждая информация об используемых ключах должна храниться в зашифрованном виде. Ключи, зашифровывающие ключевую информацию называются мастер-ключами. Желательно, чтобы мастер-ключи каждый пользователь знал наизусть, и не хранил их вообще на каких-либо материальных носителях. Очень важным условием безопасности информации является периодическое обновление ключевой информации в ИС. При этом переназначаться должны как обычные ключи, так и мастер-ключи. В особо ответственных ИС обновление ключевой информации желательно делать ежедневно. Вопрос обновления ключевой информации связан и с третьим элементом управления ключами - распределением ключей.

Распределение ключей

Распределение ключей - самый ответственный процесс в управлении ключами. К нему предъявляются два требования:

- Оперативность и точность распределения;
- Скрытность распределяемых ключей.

В последнее время заметен сдвиг в сторону использования криптосистем с открытым ключом, в которых проблема распределения ключей отпадает. Тем не менее распределение ключевой информации в ИС требует новых эффективных решений. Распределение ключей между пользователями реализуются двумя разными подходами:

1. Путем создания одного ли нескольких центров распределения ключей. Недостаток такого подхода состоит в том, что в центре распределения известно, кому и какие ключи назначены и это позволяет читать все сообщения, циркулирующие в ИС. Возможные злоупотребления существенно влияют на защиту.
2. Прямой обмен ключами между пользователями информационной системы. В этом случае проблема состоит в том, чтобы надежно удостоверить подлинность субъектов. Для обмена ключами можно использовать криптосистемы с открытым ключом, используя тот же алгоритм RSA.

В качестве обобщения сказанного о распределении ключей следует сказать следующее. Задача управления ключами сводится к поиску такого протокола распределения ключей, который обеспечивал бы:

- возможность отказа от центра распределения ключей;
- взаимное подтверждение подлинности участников сеанса;
- подтверждение достоверности сеанса механизмом запроса-ответа, использование для этого программных или аппаратных средств;
- использование при обмене ключами минимального числа сообщений.

Реализация криптографических методов

Проблема реализации методов защиты информации имеет два аспекта:

- разработку средств, реализующих криптографические алгоритмы;
- методику использования этих средств.

Каждый из рассмотренных криптографических методов могут быть реализованы либо программным, либо аппаратным способом. Возможность программной реализации обуславливается тем, что все методы криптографического преобразования формальны и могут быть представлены в виде конечной алгоритмической процедуры. При аппаратной реализации все процедуры шифрования и дешифрования выполняются специальными электронными схемами. Наибольшее распространение получили модули, реализующие комбинированные методы. Большинство зарубежных серийных средств шифрования основано на американском стандарте DES. Отечественные же разработки, такие как,

например, устройство КРИПТОН, использует отечественный стандарт шифрования. Основным достоинством программных методов реализации защиты является их гибкость, т.е. возможность быстрого изменения алгоритмов шифрования. Основным же недостатком программной реализации является существенно меньшее быстродействие по сравнению с аппаратными средствами (примерно в 10 раз). В последнее время стали появляться комбинированные средства шифрования, так называемые программно-аппаратные средства. В этом случае в компьютере используется своеобразный “криптографический сопроцессор” - вычислительное устройство, ориентированное на выполнение криптографических операций (сложение по модулю, сдвиг и т.д.). Меняя программное обеспечения для такого устройства, можно выбирать тот или иной метод шифрования. Такой метод объединяет в себе достоинства программных и аппаратных методов.

Таким образом, выбор типа реализации криптозащиты для конкретной ИС в существенной мере зависит от ее особенностей и должен опираться на всесторонний анализ требований, предъявляемых к системе защиты информации.

Идентификация и аутентификация

Идентификацию и аутентификацию можно считать основой программно-технических средств безопасности. Идентификация и аутентификация - это первая линия обороны, "проходная" информационного пространства организации.

Идентификация позволяет субъекту - пользователю или процессу, действующему от имени определенного пользователя, назвать себя, сообщив свое имя. Посредством аутентификации вторая сторона убеждается, что субъект действительно тот, за кого себя выдает. В качестве синонима слова "аутентификация" иногда используют сочетание "проверка подлинности". Субъект может подтвердить свою подлинность, если предъявит по крайней мере одну из следующих сущностей:

- нечто, что он знает: пароль, личный идентификационный номер, криптографический ключ и т.п.;
- нечто, чем он владеет: личную карточку или иное устройство аналогичного назначения;
- нечто, что является частью его самого: голос, отпечатки пальцев и т.п., то есть свои биометрические характеристики;
- нечто, ассоциированное с ним, например координаты.

Главное достоинство парольной аутентификации - простота и привычность. Пароли давно встроены в операционные системы и иные сервисы. При правильном использовании пароли могут обеспечить приемлемый для многих организаций уровень безопасности. Тем не менее по совокупности характеристик их следует признать самым слабым средством проверки подлинности. Надежность паролей основывается на способности помнить их и хранить в тайне. Ввод пароля можно подсмотреть. Пароль можно угадать методом грубой силы, используя, быть может, словарь. Если файл паролей зашифрован, но доступен на чтение, его можно перекачать к себе на компьютер и попытаться подобрать пароль, запрограммировав полный перебор.

Пароли уязвимы по отношению к электронному перехвату - это наиболее принципиальный недостаток, который нельзя компенсировать улучшением администрирования или обучением пользователей. Практически единственный выход - использование криптографии для шифрования паролей перед передачей по линиям связи.

Тем не менее, следующие меры позволяют значительно повысить надежность парольной защиты:

- наложение технических ограничений (пароль должен быть не слишком коротким, он должен содержать буквы, цифры, знаки пунктуации и т.п.);
- управление сроком действия паролей, их периодическая смена;
- ограничение доступа к файлу паролей;

- ограничение числа неудачных попыток входа в систему, что затруднит применение метода грубой силы;
- обучение и воспитание пользователей;
- использование программных генераторов паролей, которые, основываясь на несложных правилах, могут порождать только благозвучные и, следовательно, запоминающиеся пароли.

Хэш-функции

В самых различных отраслях информационных технологий находят свое применение хэш-функции. Они предназначены для того, чтобы, с одной стороны, значительно упростить обмен данными между пользователями и обработку файлов, используемых в тех или иных целях, с другой - оптимизировать алгоритмы обеспечения контроля доступа к соответствующим ресурсам. Хэш-функция - один из ключевых инструментов обеспечения парольной защиты данных, а также организации обмена документов, подписанных с помощью ЭЦП. Существует большое количество стандартов, посредством которых может осуществляться кэширование файлов. Многие из них разработаны российскими специалистами. В каких разновидностях могут быть представлены хэш-функции? Каковы основные механизмы их практического применения?

Для начала исследуем понятие хэш-функции. Под данным термином принято понимать алгоритм преобразования некоторого объема информации в более короткую последовательность символов посредством математических методов. Практическую значимость хэш-функции можно проследить в самых разных областях. Так, их можно задействовать при проверке файлов и программ на предмет целостности. Также криптографические хеш-функции задействуются в алгоритмах шифрования. Характеристики Рассмотрим ключевые характеристики исследуемых алгоритмов. В числе таковых: наличие внутренних алгоритмов преобразования данных исходной длины в более короткую последовательность символов; открытость для криптографической проверки; наличие алгоритмов, позволяющих надежно шифровать изначальные данные; адаптированность к расшифровке при задействовании небольших вычислительных мощностей. В числе иных важнейших свойств хэш-функции:

Требования к хэш-функциям

Существует ряд требований к хэш-функциям, предназначенным для практического задействования в той или иной области. Во-первых, соответствующий алгоритм должен характеризоваться чувствительностью к изменениям во внутренней структуре хешируемых документов. То есть в хэш-функции должны распознаваться, если речь идет о текстовом файле, перестановки абзацев, переносы. С одной стороны, содержимое документа не меняется, с другой — корректируется его структура, и этот процесс должен распознаваться в ходе хеширования. Во-вторых, рассматриваемый алгоритм должен преобразовывать данные так, чтобы обратная операция (превращение хэша в изначальный документ) была на практике невозможна. В-третьих, хэш-функция должна предполагать задействование таких алгоритмов, которые практически исключают вероятность формирования одинаковой последовательности символов в виде хэш, иными словами - появления так называемых коллизий. Их сущность мы рассмотрим чуть позже.

В числе главных требований к рассматриваемым алгоритмам - обеспечение однонаправленности шифрования. Человек, имеющий в распоряжении только хэш, практически не должен иметь возможности получить на его основе исходный документ. В какой структуре может быть представлена используемая в подобных целях хеш-функция? Пример ее составления может быть таким: H (hash, то есть, хэш) = f (T (текст), H1), где H1 — алгоритм обработки текста T. Данная функция хеширует T таким образом, что без знания H1 открыть его как полноценный файл будет практически невозможно. Использование хэш-функций на практике: скачивание файлов Изучим теперь подробнее варианты использования хэш-функций на практике. Задействование соответствующих

алгоритмов может применяться при написании скриптов скачивания файлов с интернет-серверов.

В большинстве случаев для каждого файла определяется некая контрольная сумма - это и есть хэш. Она должна быть одинаковой для объекта, располагающегося на сервере и скачанного на компьютер пользователя. Если это не так, то файл может не открыться либо запуститься не вполне корректно. Хэш-функция и ЭЦП Использование хэш-функций распространено при организации обмена документами, содержащими электронно-цифровую подпись. Хэшируется в данном случае подписываемый файл, для того чтобы его получатель мог удостовериться в том, что он подлинный. Хотя формально хэш-функция не входит в структуру электронного ключа, она может фиксироваться во флеш-памяти аппаратных средств, с помощью которых подписываются документы, таких как, например, eToken. Электронная подпись представляет собой шифрование файла при задействовании открытого и закрытого ключей. То есть к исходному файлу прикрепляется зашифрованное с помощью закрытого ключа сообщение, а проверка ЭЦП осуществляется посредством открытого ключа. Если хэш-функция обоих документов совпадает - файл, находящийся у получателя, признается подлинным, а подпись отправителя распознается как верная.

Хеширование не является непосредственно компонентом ЭЦП, однако позволяет весьма эффективно оптимизировать алгоритмы задействования электронной подписи. Так, шифроваться может, собственно, только хэш, а не сам документ. В итоге скорость обработки файлов значительно возрастает, одновременно становится возможным обеспечивать более эффективные механизмы защиты ЭЦП, так как акцент в вычислительных операциях в этом случае будет ставиться не на обработке исходных данных, а на обеспечении криптографической стойкости подписи. Хэш-функция к тому же делает возможным подписывать самые разные типы данных, а не только текстовые.

Проверка паролей

Еще одна возможная область применения хеширования - организация алгоритмов проверки паролей, установленных для разграничения доступа к тем или иным файловым ресурсам. Каким образом при решении подобных задач могут быть задействованы те или иные виды хеш-функций? Очень просто. Дело в том, что на большинстве серверов, доступ к которым подлежит разграничению, пароли хранятся в виде хэшированных значений. Это вполне логично — если бы пароли были представлены в исходном текстовом виде, хакеры, получившие доступ к ним, могли бы запросто читать секретные данные. В свою очередь, на основе хэш вычислить пароль непросто.

Пароль, вводимый пользователем, сверяется с тем, что зафиксирован в хэш-функции, что хранится на сервере. Если значения текстовых блоков совпадают - пользователь получает необходимый доступ к ресурсам. В качестве инструмента проверки паролей может быть задействована самая простая хэш-функция. Но на практике IT-специалисты чаще всего используют комплексные многоступенчатые криптографические алгоритмы. Как правило, они дополняются применением стандартов передачи данных по защищенному каналу - так, чтобы хакеры не смогли обнаружить либо вычислить пароль, передаваемый с компьютера пользователя на сервера - до того, как он будет сверяться с хешированными текстовыми блоками.

3. Методические рекомендации по подготовке к занятиям

3.1 Практические занятия по теме «Информация и информационные процессы. Представление информации»

При подготовке к занятию необходимо обратить внимание на следующие моменты.

1. Информатика как наука. Информация: определение, формы представления. Передача информации.

2. Сигналы и данные. Основные структуры данных. Файлы.
3. Виды информации. Свойства информации.
4. Количество информации (вероятностный и объёмный подходы). Единицы измерения информации.
5. Кодирование информации. Кодирование различных форм представления информации (числовой, текстовой).
6. Кодирование информации. Кодирование различных форм представления информации (графической, звуковой).
7. Качество информации. Ее основные потребительские показатели.

3.2 Практические занятия по теме «Информационно-логические основы построения персонального компьютера»

При подготовке к занятию необходимо обратить внимание на следующие моменты.

1. Представление информации в компьютере.
2. Позиционные и непозиционные системы счисления.
3. Двоичная, восьмеричная, шестнадцатеричная системы счисления.
4. Перевод чисел из двоичной, восьмеричной, шестнадцатеричной систем счисления в десятичную и обратно.
5. Арифметические операции в позиционных системах счисления.
6. Построение коммутационных схем на основе алгебры логики.

3.3 Практические занятия по теме «Состав и структура ЭВМ и ПЭВМ»

При подготовке к занятию необходимо обратить внимание на следующие моменты.

1. Принципы фон Неймана. Классификация ЭВМ.
2. Поколения ЭВМ и перспективы их развития. Архитектура ЭВМ.
3. Магистрально-модульный принцип построения компьютера
4. Процессор: назначение, устройство, важнейшие характеристики.
5. Структура памяти компьютера. Внутренняя память.
6. Внешняя память. Основные носители информации и их важнейшие характеристики.

3.4 Практические занятия по теме «Программное обеспечение ЭВМ»

При подготовке к занятию необходимо обратить внимание на следующие моменты.

1. Программы. Виды программного обеспечения.
2. Системное программное обеспечение.
3. Служебные приложения windows (утилиты).
4. Прикладные программы.
5. Операционная система. Состав операционной системы
6. Загрузка операционной системы
7. Операционная система MS DOS. Основные команды MS DOS.

3.5 Практические занятия по теме «Текстовые и графические редакторы»

При подготовке к занятию необходимо обратить внимание на следующие моменты.

1. Функциональные возможности текстового процессора.
2. Создание простых текстовых документов. Установка параметров страницы, колонтитулы, нумерация страниц.
3. Создание составных текстовых документов: работа с таблицами, диаграммы. Формулы, вычисляемые таблицы.

4. Создание маркированных, нумерованных и многоуровневых списков. Создание оглавления
5. Создание гипертекстового документа. Запись макроса.
6. Основы компьютерной графики. Векторные и растровые форматы.
7. Профессиональные графические редакторы.
8. Форматы графических файлов.

3.6 Практические занятия по теме «Электронные таблицы и базы данных»

При подготовке к занятию необходимо обратить внимание на следующие моменты.

1. Электронные таблицы. Основные манипуляции с таблицами. Блоки, абсолютная и относительная адресация, запись чисел и формул.
2. Функции: математические; статистические; функции даты.
3. Условные (логические) функции.
4. Сортировки данных. Фильтрация данных в списке. Промежуточные итоги.
5. Сводные таблицы. Консолидация.
6. Моделирование. Информационные модели. Этапы построения компьютерной информационной модели.
7. Формы представления и структуры информационных моделей.
8. Основные понятия БД. Классификация.
9. СУБД Microsoft Access – основные возможности. Базовые объекты СУБД MS Access.
10. Типы данных в MS Access. Связи в БД. Целостность БД.

3.7 Практические занятия по теме «Основы моделирования, алгоритмизации и программирования»

При подготовке к занятию необходимо обратить внимание на следующие моменты.

1. Классификация и формы представления моделей.
2. Методы и технологии моделирования.
3. Понятие алгоритма и его свойства.
4. Способы записи алгоритмов. Блок-схема алгоритма.
5. Основные алгоритмические конструкции. Алгоритмы линейной, разветвляющейся и циклической структуры.
6. Языки программирования, их назначение, особенности.
7. Классификация языков программирования. Транслятор и компилятор.
8. Структурное программирование.
9. Основные понятия объектно-ориентированного программирования

3.8 Практические занятия по теме «Локальные и глобальные сети ЭВМ. Основы защиты информации»

При подготовке к занятию необходимо обратить внимание на следующие моменты.

1. Передача информации. Локальные компьютерные сети.
2. Классификация сетей. Сетевые топологии.
3. Глобальные сети. Основные возможности, предоставляемые сетью Интернет.
4. Технические и программные ресурсы Интернета.
5. Информационная система. Основные свойства информационных систем.
6. Лицензионные, условно бесплатные и бесплатные программы.
7. Защита информации. Криптографические средства защиты информации.