Гимназия № 446

Домбровский В.Г.
Лимаренко А.И.

Информатика и ИКТ

Методическое пособие
для подготовки к экзамену
(10-11 класс)

Санкт –Петербург
Колпино
2007 г

Настоящая методическая разработка предназначена для подготовки к экзаменам по профильному курсу Информатики и ИКТ (10-11 класс). Методическая разработка содержит теоретические сведения по основным темам профильного уровня: информация и ее кодирование, арифметические и логические основы компьютера, устройство компьютера, программное обеспечение, файловая система, моделирование, графика, текстовый процессор, электронные таблицы, СУБД, компьютерные сети, алгоритмизация и программирование на Паскале.

Оглавление

ГЛАВА 1. Информатизация общества.

Современную эпоху называют "веком информации" или "веком технологий", что одно и то же. Что это значит? Оказывается наибольшая ценность, которой владеет то или другое государство сейчас заключается не в золоте, нефти, алмазах и т.п. ("товарах"), а в информации - знаниях, как сделать тот или другой товар (это называется технологиями). Поэтому самые богатые государства США и Япония, а Россия, к сожалению, пока проигрывает экономическое соревнование.

ОПРЕДЕЛЕНИЕ Информатика - это наука, изучающая все аспекты получения, хранения, преобразования, передачи и использования информации.

Информатика стала развиваться с середины нашего столетия, когда появился компьютер - устройство, ориентированное на хранение и обработку информации.

ОПРЕДЕЛЕНИЕ Информация - это содержание сообщения, сигнала, памяти, а также сведения, содержащиеся в сообщении, сигнале или памяти.

Это простейшее определение, в котором не ставится вопрос о ценности информации. Если остановиться только на этом определении, то это приведет к так называемому "информационному" загрязнению окружающей среды. Остановимся на этом подробнее.

Информатизация общества. - термин который родился в 60-х годах, когда человечество осознало наличие информационного взрыва. Что это такое? Количество информации, циркулирующее в обществе в XX веке, стало стремительно возрастать в связи с быстрым ростом научных и технических знаний, разработки современных технологий. Закон увеличения информации в обществе представляет собой показательную функцию, что и позволило говорить об так называемом "информационном взрыве". Часто можно услышать утверждение, что каждые 10 лет количество информации в мире удваивается. Появилась уверенность, что человек не сможет справиться с такой лавиной информации без специальных средств обработки, хранения и использования информации. Все шире и шире для хранения информации, обработки и поиска нужной информации стали использоваться ЭВМ. Это конечно правильное решение, но без четкого осознания того, какая информация является ценной, а какая только "загрязняет" окружающую среду (например, школьную программу), даже с помощью ЭВМ справиться с информационным взрывом невозможно.

Очевидно, что ценной, является только востребованная информация - та которую используют в своей жизни люди не являющиеся ее создателями. Какая информация не является ценной и не нужна конкретному информационному потребителю? Возможны три варианта:

- тривиальная (слишком "простая") информация, очевидные факты и так уже ему известные;
- "непонятная" (слишком "сложная") информация, недоступная людям для восприятия из-за их неподготовленности к ней;
- "ненужная", как не соответствующая кругу интересов конкретного человека.

Исходя из вышеизложенного определения ценности информации:

ОПРЕДЕЛЕНИЕ №2 понятия информации:

информация - это "снятая неопределенность" в знаниях у получателя информации, которая произошла после получения сообщения.

Так как у всех людей разные интересы и разный уровень образованности, то определение истинной ценности информации и соответственно борьба с информационным загрязнением окружающей среды становится весьма сложной задачей. На практике это сводится к тому что:

- каждому лично ежедневно приходится решать для усвоения какой информации стоит прикладывать усилия, а для какой нет;
- многим организациям все время надо решать - распространение какой информации следует поощрять, чтобы все могли иметь к ней доступ. Это связано с тем, что поиск нужной информации занимает все большее и большее время.

В мире существует следующий простой способ определения ценности информации:

сосчитать сколько человек использовало ее в своей работе. Поэтому в научном мире уже давно принято в конце печатной работы указывать список использованной литературы. Чем больше таких ссылок на конкретное издание в независимых публикациях - тем больше его ценность! Очень хорошим результатом считается 5-7, а большинство остается невостребованными никогда! Подсчетом таких ссылок занимаются специальные коммерческие структуры.

Как известно сейчас уже любую информацию можно найти с помощью международной компьютерной сети InterNet, где она в основном находится на так называемых WWW-страничках спецсерверов, называемых "сайтами". В компьютерной сети особенно актуальна проблема ценности информации. Обычно ее решают так: при обращении к конкретной страничке автоматически включается счетчик фиксирующий обращение, а также повторное обращение данного абонента. Непопулярные странички "снимают", а популярные даже могут заработать денежные средства, например? за счет коммерческого размещения рекламных объявлений.

Современных виды информационного обслуживания.

Факсимильная связь. - передача преобразованного в электрические сигналы изображения документа по электрическим линиям связи (например, телефонная линия) с помощью специального устройства - факса. Факсимиле - это система электронной передачи документов (схем, чертежей фотографий). С помощью телефакса можно передавать изображения и текст (например коммерческий договор с круглой печатью - он считается юридическим документом). Факс включен чаще всего круглосуточно. Габариты факсимильного аппарата небольшие, часто факсы можно использовать и для ксерокопирования документов.

Конференции. - способ общения между абонентами компьютерной сети. Проводятся по темам (например "рок музыка", "программирование" и пр.). Каждый "подписавшийся" на данную конференцию может послать письмо (в режиме "почтового ящика"), которое автоматически будет послано всем участникам конференции. Конференции позволяют быстро находить нужную информацию и знакомиться с людьми близкими к тебе по духу. Наиболее известные сети - *Интернет* и *Фидонет*. Интернет - коммерческая сеть, а ФидоНет - самая популярная в России не коммерческая международная любительская сеть. Ее абоненты называют себя "фидошники", и у них очень специфический сленг.

Конечно более интересно общаться не по электронной почте, а непосредственно в режиме "реального времени". При этом, в принципе, можно даже использовать видеокамеры, хотя возможности большинства существующих сетей не позволяют этого сделать (из-за быстродействия). Если все же картинка и звук передаются, то такая конференция уже называется видеоконференцией ..

Телеконференция - это способ общения между двумя и более людьми, который предусматривает использование компьютера для передачи разговорной речи участников конференции по телефонной сети в реальном масштабе времени. Чаще всего речь пока передается в виде текста.

Необходимые технические средства. компьютер, модем, цифровая видеокамера, звуковая карта с микрофоном и акустическими системами.

Применение: передача речевой, текстовой и видеоинформации; совместная работа нескольких пользователей с документами; медицинские консилиумы; удаленная диагностика оборудования; дистанционное обучение. К сожалению современная телефонная сеть не позволяет проводить видеоконференции (требуется оптоволокно или спутниковая связь), поэтому сейчас телеконференции проводятся или в текстовом режиме или осуществляется только голосовая связь. Телеконференции уже давно проводятся на производстве и без использования ЭВМ, но там они называются "селекторное совещание" и проводятся с использованием телефонной сети и специального устройства "селектора", который стоит на столе у каждого более-менее большого начальника (главного инженера, директора завода, министра) и можем одновременно соединить много абонентов. До эпохи ЭВМ в Питере существовал (и сейчас еще есть) "телефонный эфир" - определенные телефонные номера, по которым могут связываться

одновременно несколько абонентов, общаться и знакомиться.

Электронная почта - это система пересылки и хранения сообщений между пользователями сети ЭВМ; предназначена для доставки писем, подготовленных на ЭВМ по телефонной сети с помощью модема и разветвленной сети серверов, обрабатывающих почту. Адресат должен иметь ЭВМ, модем и обладать электронным адресом (по-английски e-mail), зарегистрированным на любом сервере данной компьютерной сети. Адрес состоит из 2 частей объединенных знаком "@".

Телекс - это международная служба телесвязи, обеспечивающая обмен сообщениями между телетайпами абонентов через телеграфную сеть (используется периодическими изданиями для передачи корреспонденции, почтой, военными).

Телетайп - устройство для передачи и приема данных по телеграфным каналам связи.

Телетекст - система одноканальной широковещательной передачи текстовой информации. Сейчас передается телетекст по всей России одновременно с передачей телевизионных программ, но чтобы его прочитать нужен не совсем простой телевизор или специальное устройство.

Автоматизированные рабочие места (АРМ) - оснащенные компьютером и средствами связи рабочие места людей, чья работа связана с обработкой и хранением информации. В зависимости от специальности компьютер оснащается соответствующим программным обеспечением (бухгалтер, экономист - ЭТ и БД, секретарь - текстовый редактор, инженер-конструктор - AutoCad, инженер-программист - язык программирования и т.д.). Рабочее место конструктора дополнительно оснащается графопостроителем (плоттером), продавца - устройством считывания штрих-кода и кассовым аппаратом и т.п.

ЭВМ широко используются в издательском деле (подготовка макетов книг, журналов, газет) и в литературоведении (грамматический и статистический анализ, составление словарей, проверка грамматики и орфографии, определение авторства и т.п.). Сейчас можно не очень дорого приобрести целую малогабаритную настольную компьютерную издательскую систему с лазерным принтером (для изготовления оригинал макетов), ризографом (для тиражирования) и приспособлениями для переплета книг и брошюр.

Информационно-справочные системы позволяют ускорить поиск данных при работе с большим объемом фактического материала (выписки, цитаты, даты, имена, афоризмы, рефераты, наблюдения, факты,...).

Экспертные системы - интеллектуальные системы, призванные играть роль прибора-советчика построенного на базе опыта и знаний эксперта. Ядро системы - база знаний в определенной предметной области.

Эксперт - квалифицированный специалист в некоторой области (например - в медицине, военном деле, и т.п.), человек, обладающий глубокими теоретическими знаниями и богатым практическим опытом. Его знания и оценки закладываются в базу знаний на ЭВМ.

Научно-технические и патентные информационные системы используются при проектировании и конструировании новой техники (подбор материалов, готовых узлов, анализ научных исследований, а также патентный поиск с целью защитить новое изделие авторским правом).

Примеры применения ЭВМ

в научных исследованиях, проектировании, производстве, обслуживании, управлении:

1) САПР (системы автоматизированного проектирования) используются для проектирования механизмов. Например, на "Ижорском заводе" в "Ижора-Картекс" стоит английская система с помощью которой проектируются карьерные экскаваторы - в ней существуют огромные библиотеки стандартных деталей на центральном сервере, специализированные программы типа "Чертежник", системы подготовки сопровождающей документации и много плоттеров и принтеров;

2) ЧПУ (числовое программное управление) станками, роботами, сборочными автоматами, транспортными системами. В мире получили самое широкое распро-

странение, благодаря чему вчерашний школьник через 1-2 года справляется с такой сложной станочной работой, которой раньше надо было учиться 10 лет! На "Ижоре" это реальность! Программа составляется очень просто, состоит из команд для управления инструментами станка.

3) Автоматизированные обучающие системы. Пример: программа "Инструктор", но она очень старая. Сейчас существуют современные интерактивные (диалоговые) мультимедийные обучающие системы на лазерных дисках, тренажеры для пилотов истребителей или операторов ядерных станций, программы обучающие английскому и т.п.

4) АСУП (автоматизированная система управления производством). В единый центр по сети ЭВМ поступают данные о состоянии производственного процесса (напр. сколько сделали продукции, сколько запасов на складе, сколько сотрудников заболело и т.п.). По этим данным осуществляется учет и контроль, они передаются в экспертную систему которая выдает рекомендации по управлению производством. Обычно в АСУП входят программные пакеты "Кадры", "Склад", "Бухгалтерия" и другие;

5) АСУТП (автоматическая система управления технологическим процессом).

Обычно управляет каким-то одним замкнутым процессом, например производством серной кислоты;

6) СМО (системы массового обслуживания), например:

- справочная служба точного времени,
- система кредитных банковских карточек,
- системы продажи билетов для железнодорожного и воздушного транспорта.

ГЛАВА 2. Развитие вычислительной техники.

Первое счетное "устройство" у человека - пальцы.

Первое специализированное счетное устройство - аббак ("каменные счеты"), который представлял собой доску с горизонтальными канавками в которые помещались счетные камешки. Аббак - прообраз известных всем счетов. Эти первые вычислительные приспособления изобретены на Древнем Востоке.

Первое "механическое" счетное устройство придумал в XVII веке великий французский богослов, философ, физик и математик Блез Паскаль. В истории это устройство осталось под именем "паскалина" - она была прообразом современного арифмометра (вроде кассового аппарата "с ручкой"). Но прошло более 100 лет прежде чем был изобретен принцип программного управления. В XIX веке в Англии на протяжении всей своей жизни великий ученый Чарльз Бэббидж работал над созданием "аналитической машины". Это устройство уже можно назвать не просто "калькулятором", а первой механической вычислительной машиной, так как в первые в ней был реализован принцип программного управления. В аналитической машине Бэббиджа были такие важные устройства как:

- "склад" (прообраз ОЗУ);
- "мельницу" (прообраз АЛУ);
- "контора" (прообраз устройства управления ЦП).

К сожалению, аналитическая машина полностью не была реализована из-за своей громоздкости, но для нее были написаны первые в мире программы. Их создателем была графиня Ада Лавлейс, дочь знаменитого английского поэта Байрона.

В конце XIX века началась эпоха электричества, появились электромеханические устройства типа реле. В США был объявлен конкурс на создание машины для обработки результатов переписи населения, которые хранились в виде таблиц. В этом конкурсе победил американский инженер-предприниматель Генрих Холлерит (или Голлерит), который разработал "статистический табулятор", воспользовавшись идеями Чарльза Бебиджа. В дальнейшем фирма, которую он основал, была после ряда реорганизаций в 1924 году переименована в IBM (Интернациональные Бизнес Машины). Сейчас это самая крупная компания в США.

До начала первой мировой войны все счетные машины были не "электронными", а

"электромеханическими", т.к. в них не использовались "электронные" устройства, то есть специальные устройства для обработки информации. В этих машинах не было ОЗУ, программа работы записывалась на перфокартах и выполнялась по шагам в режиме непосредственного счета. Поэтому они были крайне медлительны и громоздки. К тому же реле - очень не надежный элемент, т.к. его контакты быстро загрязняются.

Первое поколение ЭВМ:

- отличалось от последующих тем, что в качестве основных элементов в ЭВМ этого поколения использовались радиолампы (электровакуумные приборы с нитью накаливания). По сравнению с релейными машинами быстродействие повысилось более чем в 1000 раз, но были налицо и следующие недостатки радиоламп:

1) Большие габариты.
2) Высокая температура внутри ЭВМ.
3) Высокое напряжение питания (сотни вольт).
4) Низкая надежность, как следствие 2 и 3 пунктов.
5) Низкое быстродействие.
6) Высокая стоимость.
7) Большой штат обслуживающего персонала.

Машин были единицы, работали они "на войну" - сначала рассчитывали таблицы для управления артиллерийской стрельбой (США), расшифровывали вражеские секретные шифры (Англия). После войны рассчитывали траектории полета баллистических ракет.

Первой ЭВМ считается сконструированный в США в 1946 г., "ЭНИАК". Эта машина содержала около 18000 ламп, быстродействие у нее было в 1000 раз большее чем у релейных ВМ. Недостаток - она копировала архитектуру электромеханических ВМ. Программа задавалась штекерным методом, т.е. соединением блоков ЭВМ в определенной последовательности.

На основе анализа возможностей ЭНИАК Джон Фон Нейман предложил ряд новых принципов, в том числе - идею хранимой программы (в памяти ЭВМ). 1949 г., ЭДСАК, Великобритания, первая ЭВМ с хранимой программой. Входное устройство позволяло осуществить ввод в память ЭВМ программы и исходных данных (в двоичном коде). Вводу информации в ЭВМ предшествовало кодирование ее двоичным кодом.

Первая серийная ЭВМ с хранимой программой была введена в эксплуатацию в 1951 году (УНИВАК).

Применение первых ЭВМ:

- расчеты баллистических характеристик оружия;

- научные расчеты (математика - определенные интегралы, таблицы тригонометрических и других функций; физика - теория Эйнштейна; астрономия - траектории движения планет и их искажения, открытие новых небесных тел в результате теоретических расчетов).

Второе поколение ЭВМ (1951-1958 гг.).

В этот период наблюдалось постепенное совершенствование электронных ламп (миниатюризация) и постепенный переход к полупроводниковым приборам (транзисторам). Представитель - СЕАК, США 1950 г. Быстродействие 1 млн. оп/с.

Транзистор в отличие от радиолампы в сотни раз более миниатюрен, не нагревается, так как в нем нет нити накала и работает от совсем низкого напряжения (до 1 вольта).

Также были созданы новые *запоминающие устройства*:

- на ферритовых сердечниках (использовалось свойство материалов изменять полярность при намагничивании электрическим током).

- долговременная память на магнитных дисках. В программировании внедрены новшества:

- появление ассемблера (каждая машинная команда в этом языке программирования

заменялась мнемоническим обозначением из 2-х или 3-х букв.

- появление алгоритмических языков;
- операторные методы программирования;
- концепция крупноблочного программирования.

Структура ЭВМ изменилась:

- увеличение плотности монтажа электронных элементов в 10 раз;
- уменьшение размеров в 10 раз;
- уменьшение энергопотребления в 10-100 раз;
- срок службы транзистора, а значит и ЭВМ стал больше в 100 раз.

Архитектура ЭВМ улучшилась:

- появление многопроцессорных систем;
- разделение времени при доступе нескольких программ;
- одновременный ввод-вывод информации и ее обработка;
- появились новые устройства ввода-вывода информации ЭВМ - перфокарты.

Быстродействие ЭВМ возросло до 1 млн оп/с.

Третье поколение ЭВМ (1965 г.)

Связано с созданием интегральных схем (ИС) на базе транзисторов. Укрепление на рынке компьютеров позиций фирмы IBM. Появилась совместимость ЭВМ на уровне машинных языков. Интегральная схема - объединение в одном корпусе на специальном кремниевом кристалле (ЧИПе) сначала нескольких десятков, затем сотен, тысяч и т.д. соединенных между собой транзисторов.

В программировании:

- создание операционных систем, которые позволили облегчить обслуживание ЭВМ, создать систему виртуальных машин (одна ЭВМ - много пользователей);
- продолжает использоваться ассемблер;
- появляются языки высокого уровня (Паскаль, Фортран, Алгол, ...).

В архитектуре и структуре ЭВМ:

- минимизация аппаратуры центрального вычислителя;
- создание универсальных ЭВМ, способных управлять объектом в реальном масштабе времени;
- совершенствуются устройства ввода-вывода, появился алфавитно-цифровой терминал, позволивший работать с ЭВМ в диалоговом режиме.

Быстродействие ЭВМ увеличилось до 10 млн. оп./сек.

Четвертое поколение ЭВМ (с 1975 г.)

Появление этих компьютеров связано с созданием больших ИС (БИС) и микропроцессоров, а также микро ЭВМ на их основе. Микропроцессор (МП) - это БИС, выполняющая функцию центрального процессора (АЛУ + устройство управления + РОНы). Первый микропроцессор появился в 1971 г. и был выпущен фирмой Intel, США. На базе МП создавались также и супер-ЭВМ (многопроцессорные) производительностью до 1 млрд. оп/сек.

Революционный переворот в информатике произвело появление персональных ЭВМ (ПЭВМ). Первая ПЭВМ - 1974 г. "Альтаир 8800". На его базе разработан первый массовый персональный компьютер "Apple 2". До сих пор на рынке ПЭВМ компьютеры этой фирмы ("Эпл-Макинтош") успешно конкурируют с IBM, т.к. их память не имеет деления на базовую, дополнительную и расширенную и все они "белой сборки" (поэтому и дороже)

Кроме ПЭВМ микропроцессоры по настоящему широко стали встраиваться в производственную, связную, медицинскую, военную, бытовую и др. аппаратуру.

В программировании появились:

- СУБД;
- развиваются и совершенствуются языки высокого уровня (процедурные ЯВУ);

- появление объектного программирования;
- дисковые операционные системы.

В архитектуре ЭВМ: - цветной графический дисплей;

- устройства графического ввода-вывода (цветные принтеры, сканеры, графические планшеты, ...).
- устройства ввода-вывода звуковой информации.

Пятое поколение ЭВМ (1990 - ...)

На взгляд многих специалистов новое поколение компьютеров еще не разработано. В целях рекламы, маркетинга ЭВМ пятого поколения называют мультимедийные компьютеры большой мощности и быстродействия, в которых начинают появляться признаки так называемого "искусственного интеллекта", реализуются принципы "виртуальной реальности". Появление этих компьютеров связано с созданием сверхбольших ИС (СБИС), МП и микро ЭВМ на их основе.

Появился мультимедийный интерфейс - близкий к естественному (речевой и визуальный ввод-вывод информации).

В программном обеспечении - появление баз знаний и экспертных систем, доступ к которым можно осуществлять с помощью ЭВМ высокой производительности.

В системах связи - глобальные высокоскоростные сети (Интернет).

Мультимедийные устройства ввода-вывод: звуковые карты, микрофон, акустические системы, цифровые видеокамеры.

На самом деле новое поколение ЭВМ появится только когда будет разгадана загадка поразительных интеллектуальных способностей человека и появятся не "элементы", а полноценный искусственный интеллект, в чем то, наверно, существенно отличный от человеческого, но будем надеяться что дружественный!

Современная компьютерная техника.

Основные характеристики:

1. Габариты и вес персонального компьютера. Различают настольный и портативный (Notebook) варианты компьютеров, причем последние имеют в 3 - 5 раз большую стоимость.
2. Энергопотребление (оно меньше у портативных компьютеров).
3. Производительность ПК. Определяется типом процессора и тактовой частотой. Тип процессора можно определить по числам: 286,386,486 или названию и Pentium I,II или III. Тактовая частота уже приближается к гигагерцам. Производительность - это количество операций, выполняемых за секунду. Быстродействие ЭВМ определяется не только производительностью ЦП и частотой (для частоты 50 МГц - около 10 млн.оп./сек), но и другими факторами, например, разрядностью ЦП, разрядностью и быстродействием шины, винчестера и пр.
4. Объем ОЗУ ПК. Чем больше объем ОЗУ, тем более современное программное обеспечение может быть использовано.(в современных ПК объем ОЗУ может быть от 1 до 64 Мб и более)
5. Объем жесткого диска ("винчестера"), его скорость передачи данных (от 100 кб/сек до 2 Мб/сек), время перехода читающей головки от одной дорожки к другой - около 5 мс).
7. Наличие НГМД (1.44 Мб, 2.88 Мб или 1.2 Мб).
8. Мультимедийное оборудование (CD-ROM и звуковая карта, а также акустические системы, микрофон, графический ускоритель для просмотра видеофильмов, цифровая видеокамера для ввода динамических графических изображений).
9. Тип монитора. Различают цветные и черно-белые мониторы. Цветные мониторы отличаются друг от друга количеством цветов, размером диагонали, размером зерна и количеством воспроизводимых точек по горизонтали и вертикали.

Размер точек (зерна) 0.39 или 0.28 мм.

Типы мониторов:
- CGA (640x200x4 цвета)- устарели;
- EGA (640x350x16 цветов) - устарели;
- VGA (640x480x256 цветов) - устаревают;
- SVGA (1024x768x64000 цветов и более) .

Размер диагонали у портативных ПК 8-11 дюймов, у настольных - 14-20 дюймов.

10. Цветные принтеры с фотографическим качеством печати (лазерные или светодиодные или струйные).

ГЛАВА 3. Функциональная схема ЭВМ. Устройство компьютера.

Современные ЭВМ построены согласно так называемых "принципов фон Неймана", которые он изложил в 1946 году:

1. ЭВМ должна создаваться на электронной основе и работать в двоичной системе счисления.

2. В состав ЭВМ должны входить:

- арифметическое устройство, с представлением чисел в форме с плавающей точкой;
- центральное устройство управления;
- запоминающее устройство (в т.ч. ОЗУ для чисел и команд и связанное с ним внешнее ЗУ большей емкости для долговременного хранения программ и данных);
- устройства ввода данных и вывода результатов на печать.

3. В системе команд ЭВМ должны быть команды условной и безусловной передачи управления для создания циклических и ветвящихся алгоритмов.

Эти принципы реализованы с помощью т.н. "архитектуры ЭВМ".

Архитектура ЭВМ - это совокупность характеристик компьютера, которые важны для потенциального пользователя машины. К ним относятся система команд, структура данных, интерфейс и его возможности, объем памяти и ее виды.

Самый главный элемент ЭВМ - микропроцессор, специальная большая интегральная схема, которая управляет работой компьютера и всех его устройств.

Для увеличения производительности работы во многих компьютерах используется не один, а несколько процессоров.

Многопроцессорная система - это вычислительная система, состоящая из двух или большего количества взаимосвязанных процессоров, работающих с общей памятью.

Однородные многопроцессорные системы - в которых все процессоры одинаковы и полностью взаимозаменяемы.

Неоднородные многопроцессорные системы - системы, в которых имеются специализированные процессоры, предназначенные для решения определенных типов задач (обработка символьной информации, управление базой данных и т.д.)

Многомашинная архитектура (компьютерная сеть) - это объединение двух или более ЭВМ, соединенных при помощи каналов передачи данных (линий проводной или радиосвязи, оптических линий связи). Различают глобальные и локальные компьютерные сети.

Классическая архитектура ЭВМ - архитектура с одним процессором и стандартным интерфейсом (устройствами ввода-вывода информации).

Параллельный процессор (транспьютер) - это вычислительное устройство, выполненное в виде одной сверхбольшой интегральной схемы, содержащее процессор, запоминающее устройство и коммуникационные связи для соединения с другими транспьютерами. Транспьютер может использоваться для построения параллельных высокопроизводительных систем.

Например, ЭВМ с 286 или 386 микропроцессором можно было дополнить еще так называемым "блоком расширенной арифметики", который брал на себя рутинные вычисления, например связанные с обработкой анимационной графики или вычислений с повышенной точностью. Этот блок содержал дополнительный "математический" сопроцессор. У микропроцессоров более старших поколений этот блок встроен в состав само-

го микропроцессора. С этой же точки зрения можно рассматривать также графические ускорители, обрабатывающие графическую информацию. ЭВМ с такими устройствами можно отнести к неоднородным многопроцессорным системам.

К сожалению в этом издании не удалось поместить схему микропроцессора.

В самом упрощенном виде он состоит из трех основных блоков:

- арифметико-логическое устройство (АЛУ) выполняет арифметические и логические операции над машинными словами в виде двоичных кодов;
- регистры общего назначения (РОНы) хранят промежуточные результаты работы АЛУ и исходные для его операций;
- устройство управления управляет работой АЛУ и РОНов.

При ответе надо также изобразить функциональную схему ЭВМ с общей шиной и рассказать про магистрально - модульный принцип ее работы.

Современные персональные компьютеры (ПК) построены по так называемому "магистрально-модульному принципу" или еще их называют компьютерами с "открытой архитектурой". В таком ПК каждый модуль имеет свое назначение и выполнен как отдельное устройство, которое можно легко присоединить к общей шине (магистрали).

Магистраль - это набор электрических соединений на печатной плате посредством которых передается информация (адреса, данные и управляющие сигналы) от одного модуля к другому. Разрядность магистрали (количество проводников) определяют по шине данных (бывают 8, 16, 32, 64-хразрядные шины).

ПК с открытой архитектурой - это такой ПК, в котором можно изменить состав и количество модулей: в любой момент мы можем вынуть один из модулей не трогая остальные и заменить его на более совершенный. Кроме этого, мы можем добавить новые модули и сделать машину более подходящей для выполнения каких-либо специфических задач (написание музыки, научные исследования, издательская деятельность), не лишая себя при этом возможности решать и другие задачи.

У ЭВМ с открытой архитектурой основной недостаток - магистраль может соединить одновременно только 2 устройства, поэтому канал часто "занят" и устройствам желающим обменяться информацией приходиться "выстраиваться в очередь", что безусловно снижает быстродействие. Отчасти с этим недостатком можно бороться, если использовать так называемое кэширование информации.

Кэшированием называют временное хранение крупных блоков информации в памяти, а затем передачу этого блока целиком по каналу связи.

Информация в кэш-блоке накапливается постепенно по мере ее обработки и пока кэш не заполнится не передается. В результате за счет передачи информации не маленьким "кусочкам", а большими блоками сокращается время когда магистраль занята. Кэширование - понятие многоуровневое. Свой внутренний кэш есть у современных микропроцессоров, кэш может быть и дополнительный в виде отдельной микросхемы на материнской плате. Также кэш создается и на винчестере и в ОЗУ с помощью специальных драйверов.

ВНУТРЕННИЕ УСТРОЙСТВА ЭВМ.

По каким критериям разделить устройства ЭВМ на внутренние или внешние? Неверно считать, что внутренние устройства находятся "внутри" системного блока, а внешние снаружи. Например, винчестер - внутри, но это внешнее устройство.

Внутренние устройства - это те устройства с которыми центральный процессор обменивается информацией без посредников (адаптеров) и без которых он принципиально не сможет работать. Какие это устройства? В первую очередь сам ЦП, и внутренние устройства ввода-вывода информации - ОЗУ для хранения программ и их результатов, ПЗУ с программой начальной загрузки (например BIOS), а также общая шина (магистраль), соединяющая между собой эти устройства. Представьте себе беспилотную ракету-автомат, летящую под управлением ЭВМ. Бортовой ЭВМ дисплей не нужен, клавиату-

ра тоже. Устройство ввода для нее - радар, устройство вывода - рули.

Внешние устройства - устройства ввода-вывода информации, предназначенные для организации обмена информацией с пользователями ЭВМ. Без этих устройств или с другими устройствами компьютер в принципе может функционировать. Эти устройства соединяются с общей шиной не напрямую, а через адаптеры или специальные порты ввода-вывода.

Характеристики основных внутренних устройств.

1. Центральный процессор (CPU) - управляет работой ПК, выполняет арифметические и логические операции по программе, которая размещена в ОЗУ. ЦП является "сердцем" любого компьютера. Обычно указывается его тип и тактовая частота с которой он работает. Тактовая частота измеряется в мегагерцах и сегодня может уже достигать 300-400 Мгц. ЦП в современных компьютерах это одна большая интегральная схема (БИС), которую называют также микропроцессором. Внутри него находится чип ("осколок") - кристалл кремния, внутри которого находятся миллионы соединенных между собой транзисторов.

2. ОЗУ (RAM - Random Access Memory) - оперативное запоминающее устройство (кратковременная память),хранит все программы и данные с которыми работает процессор. Содержимое ОЗУ можно легко изменять в процессе работы с ПК, при выключении питания ПК ОЗУ очищается. Оперативной этот вид памяти называется потому, что это быстродействующая память и обращаясь к ней процессор практически не простаивает. Объем ОЗУ 1-64 Мб и более (сейчас покупать ЭВМ с ОЗУ менее 8М не целесообразно). ОЗУ - основной вид внутренней памяти ЭВМ.

3. ПЗУ (ROM - Read Only Memory) - постоянное запоминающее устройство (долговременная память), программируется обычно на заводе изготовителе. Во всех ЭВМ в ПЗУ хранится BIOS - базовая система ввода-вывода (часть ДОС), которая загружается после включения ЭВМ и уже с ее помощью с системного диска загружается ядро операционной системы. Изменить содержимое ПЗУ нельзя. Информация храниться в ПЗУ и при выключенном питании. В микросхему ПЗУ встроен также таймер (компьютерные часы) с питанием от аккумулятором. Там же хранятся параметры ЭВМ, которые называют "параметры SetUp" тоже имеют аккумуляторное питание (в первую очередь - тип винчестера).

У современных ЭВМ для хранения BIOS уже используют не ПЗУ, а перепрограммируемое постоянное запоминающее устройство (ППЗУ), в него можно записать другую версию BIOS или, например, повысить тактовую частоту! Это можно сделать с помощью специальной программы-утилиты. К сожалению, в это устройство может внедриться и вирус, например, "Чернобыль" и перепрограммировать ("сломать") BIOS.

ПЗУ и дисковая память являются гораздо более "медленной" памятью, чем ОЗУ. Поэтому все программы при их запуске сначала копируются с диска или из ПЗУ в ОЗУ, и только после этого начинают выполнятся.

4. Общая магистраль или шина . - совокупность электрических связей (проводов) объединяющих между собой блоки ЭВМ. Делится на шину данных, шину адреса и шину управления.

Разрядность ЭВМ определяется шиной данных - сколько двоичных разрядов (проводов) в машинных словах, которыми обмениваются между собой блоки ЭВМ.

Магистрально-модульный принцип построения ЭВМ - это использование для обмена информации между ее модулями одной общей универсальной магистрали. В результате применения в архитектуре ЭВМ общей шины ввода-вывода информации ("магистрали") стало возможна компоновка компьютера любыми стандартными внешними блоками без изменения схемы. Для Windows 95 даже не надо ничего дополнительно устанавливать из программного обеспечения - обозначение у шины "PnP" означает что

она поддерживает стандарт "Plug and Play" (установи и играй), т.е. при установке нового блока он распознается автоматически. Недостаток такого аппаратного решения - канал часто занят, т.к. его одновременно могут использовать только два устройства. Поэтому страдает быстродействие, что компенсируется высокой тактовой частотой и кэшированием.

ПЕРЕЧЕНЬ ВНЕШНИХ (периферийных) УСТРОЙСТВ.

Число внешних устройств определяется практической целесообразностью и Вашими финансовыми возможностями.

1. Клавиатура и мышь - устройства ввода информации в ПК. Правильное название мыши - настольный манипулятор. Предпочтительнее оптические мыши, как более надежные.

2. Монитор (дисплей) - устройство вывода текстовой и графической информации из ЭВМ, который работает совместно с видео картой. На видео карте установлено видео ОЗУ.

3. Принтер - устройство вывода текстовой и графической информации из ЭВМ на бумагу. Наиболее распространены матричные, струйные и лазерные.

4. Винчестер или НЖМД (HDD), накопитель на жестких магнитных дисках - HDD, предназначен для долговременного хранения данных и программ, необходимых для работы на том ПК, где он установлен. Этот тип памяти работает медленнее чем ОЗУ, процессор подолгу простаивает. Емкость винчестеров от 10 Мб до десятка Гб.

5. Дисководы или НГМД (FDD), накопители на гибких магнитных дисках, предназначены для долговременного хранения данных и программ, достаточно редко используемых в процессе работы, архивных копий важных программ и документов, для переноса программ с одного ПК на другой, для коммерческого распространения программ. Дисководы бывают двух видов - рассчитанные на дискеты размером 133 мм (5,25 дюйма) емкостью 360, 720, 1200 кб, и 89 мм (3,5 дюйма) емкостью 720, 1440, 2880 кб. Отличие от винчестера - меньшая скорость работы, меньшая емкость и возможность механической защиты от записи и стирания.

6. CD-ROM - устройство для чтения компакт-дисков, обладающих большой емкостью (до 1000 Мб); скорость работы - между винчестером и НГМД, диски обладают высокой надежностью. Недостаток - нет возможности перезаписи информации. Различаются количеством скоростей; самые простые - двухскоростные.

7. Сетевой адаптер - позволяет подключить машину к локальной компьютерной сети.

8. Модем и факс-модем - позволяет обмениваться данными между компьютерами через телефонную сеть. (Факс-модем - между факсимильным аппаратом и ПК).

9. Звуковая карта, акустические колонки - расширяет возможности ПК в области воспроизведения звука.

10. Сканер - позволяет считывать с бумаги в ПК фотографии, рисунки и тексты.

11. Плоттер (графопостроитель) - устройство для вывода на бумагу чертежей.

12. Магнитооптические диски - позволяют хранить большие объемы информации, обладают возможностью многократной перезаписи информации. Предназначен для хранения копии винчестера.

13. Стример - предназначен для хранения информации на магнитной ленте (назначение как у магнитооптических дисков).

14. Цифровые видеокамеры - позволяют проводить сеансы видеосвязи по компьютерной сети или создавать видеосюжеты для мультимедийных программных продуктов.

15. Источники бесперебойного питания - предохраняют информацию записанную в памяти ЭВМ от скачков напряжения в сети и аварийного отключения питания.

16. Фото считки - устройства для ввода данных с перфоленты, перфораторы устройства для вывода данных на перфоленты.

17. Фото сканеры – устройства для создания компьютерных фотографий минуя этап печати на фотобумаге.

18. Трехмерные сканеры – устройства для ввода в ПК объемных изображений с глубиной до 20 см. Используются в торговле для считывания полосковых кодов с этикеток товара. Позволяют также получить объемное изображение с любого предмета: деталей машин, экспонатов музеев, вещественных доказательств и т.д.

ГЛАВА 4. Информация. Единицы измерения информации

Информация, количество информации, ценность информации.

ОПРЕДЕЛЕНИЕ №1 понятия информации на интуитивном ("бытовом") уровне: информация - содержание сообщения, сигнала или памяти.

ОПРЕДЕЛЕНИЕ №1 единицы количества информации на этом же уровне (его еще называют "техническим" или "инженерным"):
количество информации измеряется объемом памяти, необходимой для записи этой информации. Согласно этому определению, 1 бит информации - это такое количество информации, которое можно записать в элементарную ячейку памяти (например триггер). А записать туда можно или логический ноль, или логическую единицу.

Но при этом сразу возникает парадокс, например такой: обычно, чтобы сосчитать количество информации в какой-нибудь книге вычисляют сколько в ней символов, включая пробелы, и умножают на 1 байт, т.к. 1 символ в ЭВМ кодируется обычно с помощью кода ASC II восьмиразрядными двоичными числами (1 байт=8 бит, а 1 бит - элементарная ячейка памяти ЭВМ, в которую можно записать минимальное количество информации "Логический 0" или "Логическую 1"). Получиться, что необходима, например, целая дискета; для хранения книги "Я" одного зазнайки, в которой на каждой странице во всех строчках записано только это выдающееся слово. Так неужели в такой книге более 1 Мб (мегабайта) информации ?!

С точки же зрения математической теории информации или КИБЕРНЕТИКИ, созданной в 40-х годах Винером, давать определение информации надо исходя из других принципов. В этой теории принят **ЭНТРОПИЙНЫЙ** (или **"вероятностный"**) подход.

ЭНТРОПИЯ - общая мера неопределенности системы или, говорят, "степень беспорядка".

Например из физики известно понятие **"ТЕПЛОВАЯ СМЕРТЬ ВСЕЛЕННОЙ",** когда все звезды погаснут, а все вещество равномерно перемешается - во всех точках вселенной все будет одинаково, энтропия такой системы будет максимальна. Количество информации в такой системе минимально. Понятно, что чем энтропия меньше, чем вселенная разнообразнее, тем информации в ней хранится больше (больше планет, книг, ЭВМ и т.д.).

Воспользовавшись этим подходом, можно сказать, что в книге "Я" информации мало, т.к. энтропия в ней большая, неопределенности почти нет.

Теория информации также говорит, что нет "абстрактной" информации, она всегда связана с "получателем" информации, т.е. она всегда кому-то предназначается. Поэтому принята следующая модель:

- получатель информации всегда имеет определенное представление о возможных наступлениях некоторых событий (например, игрок в карты, знает из каких карт состоит колода и вытаскивая из нее одну карту, предполагает, какая это может быть карта);
- когда событие произошло (игрок вытянул и посмотрел на одну карту из 32) общая неопределенность, в которой находился получатель информации уменьшилась - он получил информацию.

Напротив, если игроку просто дали все 32 карты, которые он и так знает, то это тривиальное сообщение, оно не изменяет ожидаемых вероятностей и не несет для получателя никакой информации! Хотя "что-то" и передавалось - для него это не ИНФОРМАЦИЯ, как часто и случается для некоторых учеников на уроках!

Таким образом, исходя из этого подхода, можно дать другое определение информации.

ОПРЕДЕЛЕНИЕ №2 понятия информации:
информация - это "снятая неопределенность" в знаниях у получателя информации, которая произошла после получения сообщения!

Обратите внимание, как много при этом зависит от "получателя", от его знаний! Например Вы не знаете, где включается свет в здании, но знаете где в здании располагается 128 выключателей и предполагаете что это один из них, Вам потребуется 7 бит информации чтобы выяснить какой из них нужный (но только если Вы станете правильно ставить вопросы методом деления неопределенности пополам; первый вопрос - "Нужный выключатель в первой половине выключателей или во второй?" и т.д.). Но если Вы знаете в какой комнате включается свет в здании, а в ней всего 4 выключателя, то Вам понадобится только 2 бита информации (т.к. Ваши "старые" знания были больше, т.е. неопределенность меньше).

ОПРЕДЕЛЕНИЕ №2 количества информации:
в сообщении содержится столько информации, сколько "правильных" вопросов надо задать получателю информации, чтобы полностью устранить неопределенность в своих знаниях по данному вопросу.

Ответ на каждый вопрос может быть "Да" или "Нет". Если вопрос "правильный", т.е. он устраняет неопределенность в знаниях у получателя информации передается один бит информации. Все остальное в сообщении - "вода". Такое определение можно назвать - "принцип отбрасывания тривиальной информации".

Существует много игр, основанных на этом принципе, например - "загадай число".

ОПРЕДЕЛЕНИЕ №3 количества информации ("не игровое", "математическое") - формула Хартли:

$$H = \log m$$ ***по основанию 2, где***

H - количество информации;
m - число возможных выборов.

Для примера с выключателями: log 128=7; log 4=2.
Для примера с картами log 32=5 (потребуется 5 вопросов, например для дамы пик :

- Красной масти? Нет.
- Трефы? Нет.
- Одна из четырех старших? Да.
- Одна из двух старших? Нет.
- Дама? Да.

Такой выбор можно описать двоичным числом 00101, а для его записи потребуется 5 элементарных ячеек, т.е. 5 бит! Нашли точки соприкосновения с инженерным подходом!

Исходя из вышеизложенного - появляется новое понятие:

ЦЕННОСТЬ ИНФОРМАЦИИ - это ее основное свойство, позволяющее устранить неопределенность выбора у ее получателя, т.е. уменьшить его энтропию.

Информация является ценной для получателя, если она снимает имеющиеся у него вопросы. Получатель информации должен быть ***заинтересован*** в ней и понимать, что ему передают! Иначе, даже если передавать "сверх умную" информацию, до получателя доходит ноль! Научиться ***задавать правильные вопросы*** - основная задача каждого обучающегося человека - и тогда не придется всю жизнь "барахтаться" или в слишком "простой" или слишком "сложной" информации. К сожалению очень много ее не доходит до получателя, а представляет собой "***информационный шум***"т.к. игнорируются вышеизложенные принципы.

Зависимость ценности информации от уровня подготовки к ее восприятию

Исходя из сказанного выше понятно, что ***ценность информации*** напрямую связана с ***уровнем подготовки получателя***, причем двояко:

1) 1 случай - передается что-то очень умное, но получатель совсем к такой информации не подготовлен, он ничего не понимает. У него просто нет неопределенности в знаниях в этой области - по другому, у него нет вопросов, на которые он хотел бы получить ответы. Ценность такой информации для него - ноль!
2) 2 случай (противоположный) - передается что-то такое, что и так получателю известно (так называемая тривиальная информация), т.е. он наоборот подготовлен слишком хорошо для этой информации. Ценность такой информации для него тоже ноль.

Чтобы передавать действительно ценную информацию (например на уроке) приходится все время искать золотую середину! Проблема "информационного взрыва", "информационного загрязнения" окружающей среды (в частности телекоммуникационной сети INTERNET или, например, школьного расписания) напрямую связана с этой проблемой. Это одна из самых актуальных проблем человечества, которую должна помочь решить информатика.

Виды и основные свойства информации.

Виды информации.

Информацию можно разделить по множеству критериев.

Информацию можно различать ***по способу ее восприятия*** человеком:

- зрительная;
- звуковая;
- осязательная;
- обонятельная;
- вкусовая;
- телепатическая (шутка?!).

Можно различать ***по способу ее передачи*** в виде сигнала:

- ***аналоговая*** (в виде непрерывного сигнала, изменяющегося во времени, например сигнал от звуковой дорожки гибкой грампластинки, он повторяет бугорки и впадины дорожки);
- ***цифровая*** (в виде дискретного сигнала, например сигнал от лазерного аудиодиска, в виде "лог 1" и "лог 0". Таким же образом информация хранится, перерабатывается с помощью ЭВМ.)

Для ***преобразования*** информации из цифровой в аналоговую и обратно используются специальные устройства - адаптеры (например звуковые карты, видео карты, модемы, ***аналого-цифровые преобразователи*** (АЦП), ***цифро-аналоговые преобразователи*** (ЦАП) и пр.)

Можно различать информацию (данные) ***по способу хранения*** в памяти ЭВМ и ***по объему памяти***, которая требуется для ее хранения:

1) или как файлы, по расширению:
   - графические (.pcx, .bmp, gif);
   - текстовые (.txt, .doc, .wri)
   - файлы данных для БД (.dat, .dbf)
   - машинные коды (.exe, .com) и пр.

2) или как виды переменных (констант) в Паскале: integer, real, char, string, file, array, record, set и др.

Графическая и аудио информация занимает обычно очень много места, так же как и текстовая, созданная с помощью средств Windows.

Также информацию можно разделить ***на конфиденциальную*** (секретную) и ***открытую***, ***уникальную и повседневную, мировую и региональную, газетную и компьютерную*** и по многим другим критериям т.к. мы живем в век информации, а сейчас происходит настоящая ***информационная революция***.

Наиболее рациональным способом получения информации в наши дни становятся услуги ***мировой компьютерной телекоммуникационной сети InterNet***, в частности услуги ***WWW-серверов*** (специальных мощных сетевых машин, с которых можно "взять" практически любую информацию в виде так называемых WWW-страничек). Говорят, что WWW-серверы "опутали весь мир своей паутиной". Но для этого нужен режим on-line, за который надо платить! Теперь самое ценное в мире - информация, или, как говорят чаще, - технология ("как сделать") и революцию называют ***технологической***.

Свойства информации (наиболее важные):

1. ценность или конструктивность (рассматривали);
2. доступность (долой цензуру, архивы надо рассекретить, а цены на газеты и спутниковое телевидение снизить!);
3. оперативность (новости не должны опаздывать, быть "горячими");
4. полнота (скрытая цензура - позволяет утаить часть информации);
5. объективность (не врать в сообщениях, не навязывать своего мнения);
6. уникальность (открытия и изобретения - патентованная информация);
7. правдивость (не должна содержать неправды);
8. ложность (ложь - дезинформация);
9. противоречивость (не должна содержать внутренних противоречий);
10. читабельность, наглядность (должна легко читаться, иметь графическое оформление);
11. художественность (должна иметь художественные достоинства);
12. интерактивность (возможность получателя информации вступить в диалог с источником информации, задать вопросы).

Единицы измерения объема информации.

Что такое ***1 бит***? Исходя из вышеизложенного можно привести два определения.

ОПРЕДЕЛЕНИЕ 1 ("инженерное"):
1 бит - один символ двоичного алфавита, минимальный объем памяти ЭВМ (элементарная ячейка памяти ЭВМ), которую можно представить в виде ключа (вентиля или триггера), находящегося в состоянии или 0 (ВЫКЛ) или 1 (ВКЛ).

ОПРЕДЕЛЕНИЕ 2 (с точки зрения теории информации, учитывая ее ценность):
1 бит - устранение минимальной неопределенности в знаниях получателя информации.

Где точки соприкосновения между этими двумя определениями?

В элементарную ячейку памяти ЭВМ можно записать ***минимальное количество информации***, как говорят, один выбор из двух возможных вариантов (1 или 0). 1 байт=8 бит - один символ, например кода ASC II, который можно представить в виде восьмиразрядного двоичного числа, например 01000001 - код символа А (заглавное, латинское). Это один выбор из 256 комбинаций.

Более крупные единицы измерения:

- 1 Кбайт = 1024 бит (1024 - 2 в десятой степени!).
- 1 Кбайт сокращенно 1 Кб.
- 1 Мбайт (1 Мб)=1024 Кб.
- 1 Гбайт (1 Гб)=1024 Мб.

Емкость ГМД диаметром 3,5 дюйма - 1,44 Мб, емкость современных HDD (винчестеров) составляет гигабайты, емкость ОЗУ - 1 Мб, 4 Мб, 8 Мб, 16 Мб, 32 Мб или даже 64 Мб. В школе ОЗУ (RAM) 1-4 Мб на РМУ и 8-32 Мб на РМП.

Задача 1 на определение объема информации.

На один из выходных дней ближайшей недели Вам назначено свидание в условленном месте в определенное время. Партнер по модему передал на Вашу ЭВМ название этого дня, например "суббота". Сколько информации получили Вы и сколько Ваша ЭВМ?

Вы получили один бит т.к. устранение Вашей неопределенности - один из двух возможных вариантов (log 2=1), а ЭВМ, как минимум 7 байт * 8 бит=54 бит, если не учитывать "шапку" письма.

Задача 2 на определение объема информации.

Уже рассматривали выше - поместиться ли информация, записанная в данной книге на ГМД или в ОЗУ Вашей ЭВМ?

Задача 3 на определение объема информации.

Сколько места на диске займет картинка? Простейшее решение: все зависит от способа кодировки и от качества записи изображения (числа точек на которые мы или сканер ее разобьет и число цветов):

1. Предположим, что каждая точка картинки кодируется тремя числами - две координаты и цвет.
2. Если размеры картинки 640*480 точек, а цветов 16, то первое число должно быть 10-разрядным, второе 9-разрядным, третье 4-разрядным, если использовать комбинацию 0000.
3. Таким образом, для кодирования и записи в память ЭВМ одной точки понадобится 23 двоичных разряда или 23 бита.
4. Точек всего 640*480=307200, следовательно понадобится 307200*23=7065600 бит или 7065600 : 8 = 883200 байта или 883200 : 1024 = 862,5 Кб.

Такой способ кодирования очень не рационален (представьте, что Вы записываете один большой одноцветный прямоугольник и он займет так много места! Отсюда следует вывод: необходимо использовать более рациональные методы кодирования. Но это уже другая история ...

ГЛАВА 5. Программное обеспечение компьютера.

Современный компьютер можно условно разделить на две части:

- аппаратная часть ("hardware" - "жесткие", "тяжелые" средства), аппаратное обеспечение ЭВМ - микропроцессор, ОЗУ, ПЗУ и пр.
- программная часть ("software" - "мягкие" средства), математическое обеспечение

ЭВМ.

Современная тенденция развития ЭВМ состоит в том, что все больше средств вкладывают в "software", оно стоит все дороже и дороже, а "hardware" относительно дешевеет. Наглядная иллюстрация - Бил Гейтс, президент компании MicroSoft, производящей "software" для ПЭВМ, который недавно стал самым богатым человеком в мире!

Персональные компьютеры - это универсальные устройства, позволяющие обрабатывать информацию. Для этого необходимо составить для компьютера на понятном ему языке точную и подробную последовательность инструкций, как надо обрабатывать информацию. Такую инструкцию называют программой. Сам по себе компьютер не обладает знаниями ни в одной из областей своего применения, все эти знания сосредоточены в выполняемых на ПК программах.

Меняя программы для ПК, можно превратить его в рабочее место человека любой специальности, связанной с хранением и обработкой информации. Поэтому такое огромное значение для ЭВМ имеет программное обеспечение. Это основная компонента так называемых "компьютерных технологии".

Компьютерные (информационные) технологии - совокупность аппаратного и программного обеспечения ЭВМ предназначенного для обеспечения деятельности человека в какой либо области человеческой деятельности. Аппаратная часть при этом чаще всего является универсальной частью (персональный компьютер), а ПО - товар "штучный", поэтому дорогой. Основные виды программного обеспечения (ПО): системное, прикладное и инструментальное.

Системное программное обеспечение.

ОПРЕДЕЛЕНИЕ. Системным называется программное обеспечение, предназначено для обеспечения нормального функционирования ЭВМ. Под "нормально функционирующей" мы подразумеваем ЭВМ позволяющую пользователю найти и загрузить нужную ему программу, найти свои старые файлы, сохранить новые файлы и надеяться, что он их там еще найдет!

К современному системному обеспечению предъявляются, кроме "просто" функционирования ЭВМ следующие дополнительные требования:

- простота работы для неквалифицированного пользователя;
- графический интерфейс;
- простота установки дополнительных программных средств и периферийного оборудования. Наиболее полно эта возможность реализована в "Windows 95" и носит название "plug and play" ("установи и работай") - после запуска setup (программы установки) система автоматически самонастраивается.

Системное программное обеспечение состоит из дисковой операционной системы, и разнообразных ее оболочек (например Norton Commander, Dos Navigator, разные версии Windows). Кроме этого к нему относится огромное число драйверов для управления периферийными устройствами и утилит для обработки данных.

Драйверы и утилиты могут быть как стандартными, то есть входящими в состав DOS или оболочек, или дополнительными. Примеры утилит - архиваторы данных, антивирусные программы. Примеры драйверов - русификатор, драйвер мышки, принтера.

Надо заметить, что Windows 95 и выше уже включает в себя ДОС, поэтому и называется "операционной оболочкой".

Дисковая операционная система - это пакет программ, осуществляющих диалог с пользователем, управление ресурсами ЭВМ (оперативной памятью, местом на дисках), запуск других программ на выполнение. Ядро ДОС загружается в ОЗУ при включении компьютера.

Кроме MS DOS также распространены операционные системы UNUX, OS/2.

ДРАЙВЕРЫ - компоненты ДОС - программы для управления различными устройствами. "Основные" драйверы "вшиты" в файл *io.sis* - один из файлов ядра MS DOS. Но для "нестандартных" устройств необходимо при включении ЭВМ запускать для каждого тако-

го устройства свой драйвер (чаще всего с помощью файла *autoexec.bat*). Чаще всего приходиться сталкиваться с драйверами русификаторами, драйверами мышки, принтера, расширенной памяти т.п.

УТИЛИТЫ - программы для утилизации (обработки) данных.

Утилиты входят в состав ДОС, "Norton Commander" или распространяются отдельно, как дополнительные приложения, расширяющие возможности по управлению и обслуживанию ПК.

Утилиты входящие в ДОС - специальные программы, выполняющие внешние команды DOS, например:

FORMAT. - инициализация дисков,

UNFORMAT - восстановление информации после форматирования диска,

UNDELETE - восстановление стертых файлов и каталогов и другие.

Примеры дополнительных утилит:

- архиваторы (программы - упаковщики) - позволяют сжимать информацию за счет специальных методов т.е. создавать копии файлов меньшего размера (PKUNZIP, ARJ, RAR, и др.);

- антивирусные программы - предназначены для предотвращения заражения ПК компьютерным вирусом и ликвидации последствий его работы (Doctor Weber, ADinf, Aidstest и др).

- программы для оптимизации дисков - позволяют обеспечить более быстрый доступ к информации на диске за счет оптимизации размещения данных на диске.(Speed Disk)

- программы для диагностики компьютеров - позволяют проверить конфигурацию ПК и работоспособность его устройств (Check-It).

Многие драйверы и утилиты, строго говоря, входят в состав ДОС, но так как часто их надо туда устанавливать дополнительно мы выделили их отдельно.

Системное ПО должен устанавливать системный оператор, а разрабатывает его системный программист, т.н. "системщик" (одна из специальностей, требующая очень высокой квалификации, знания ассемблера).

ОБОЛОЧКИ ОПЕРАЦИОННОЙ СИСТЕМЫ

- программные средства, предназначенные для повышения удобства работы на ЭВМ пользователей (по сравнению с ДОС). Пример наиболее известной оболочки - "Norton Commander", хотя сейчас многие предпочитают "ДОС-навигатор". Эти оболочки работают в текстовом режиме. В графическом режиме работать намного проще. В этом режиме работает многооконная интерфейсная система пользователя "Windows 3.1" и ее дальнейшие модификации "Windows 95" и другие. Эти системы уже называют "операционными оболочками".

Операционные оболочки в отличие от простых программ-оболочек предоставляют новые возможности для запускаемых программ:

- графический интерфейс (меню, окна, кнопки, пиктограммы);

- мультипрограммный режим (возможность одновременного выполнения нескольких программ);

- расширенные средства для обмена информацией между программами.

Эти системы включают в себя уже и минимальный набор стандартных пользовательских программ ("реквизитов"). Необходимо отметить, что "Windows 95" включает в себя и ДОС. Но для работы с "Windows" необходима хорошая ЭВМ и достаточный объем ОЗУ (не менее 4М, для более-менее комфортной работы).

Прикладное программное обеспечение

ОПРЕДЕЛЕНИЕ Прикладным называется программное обеспечение, предназначенное для решения с помощью ЭВМ каких-либо прикладных задач. Эти программы непосредственно обеспечивают выполнение необходимых пользовательских работ, т.е. автоматизируют большинство операций на рабочем месте.

Прикладное ПО бывает универсальным и специализированным.

Универсальное ПО можно использовать для решения целых классов задач одного

типа (например, электронные таблицы). Специализированное ПО разрабатывается для какой-то конкретной задачи (например, какой-нибудь бухгалтерский пакет). Преимущество специализированного ПО - учет всех пожеланий заказчика, максимальная простота использования. Недостаток - дороговизна, а также "закрытость" - если необходимо что-то изменить в ПО, то надо обращаться к разработчику. Использовать универсальное ПО сложнее - необходима определенная компьютерная грамотность. Существует следующая тенденция: для простых задач пользователь использует универсальное ПО в режиме "оператора"; для более сложных задач приглашают специалиста, который решает их с помощью "программных" возможностей универсального ПО (макросов). Полученный таким образом программный продукт более "открыт" и универсален.

На сегодняшний день наиболее широко используемое универсальное прикладное программное обеспечение - так называемая "большая тройка" ПО:

- текстовые процессоры;
- табличные процессоры (электронные таблицы);
- системы управления базами данных (СУБД).

Наиболее известна "троица" разработанная фирмой MicroSoft. Она входит в состав всемирно знаменитого пакета программ MicroSoft Office - пакет деловых программ для офиса. В него входят: текстовый процессор Word, табличный процессор Excel и системы управления базами данных Access. В школах С.-Петербурга много лет изучается пакеты FrameWork и Works. Идеи, заложенные в этом пакете нашли свое продолжение и развитие в MicroSoft Office, поэтому переход к новому пакету происходит очень просто.

Основное преимущество пакета программ MicroSoft Office - это его интегрированность. Это означает, что пакет позволяет "связывать" данные в из отдельных приложений друг с другом в одном документе. Можно так "связать" данные, что, например, если в отделе кадров используют базу данных Access и в ней пропала какая-то фамилия (сократили сотрудника), то она автоматически исчезнет и в бухгалтерии, если там используют для расчета зарплаты таблицы Excel.

Во все приложения встроен язык высокого уровня (Visual Basic), позволяющий, например, создавать свои собственные "кнопки" (макросы) и вложенные меню.

Существует возможность встраивать в документ "объекты" в виде фрагментов из других приложений "MicroSoft Office" или Windows, в том числе картинки, "звуки", анимацию.

Все приложения пакета имеют одинаковый интерфейс.

Основные компоненты пакета программ MicroSoft Office

1. Текстовый процессор Word

- система для подготовки текстов, позволяют вводить, корректировать, оформлять и распечатывать текстовые документы на бумаге (отечественный аналог - "Лексикон"). Обладает в отличии от "простых" текстовых редакторов (например, Write, из реквизитов Windows) уникальными возможностями:

- создавать "кадры", свободно перемещаемые по документу и накладывающиеся друг на друга;
- создавать "типовые письма";
- использовать многоколончатую печать;
- использовать встроенный графический редактор и библиотеку картинок;
- использовать простые встроенные электронные таблицы;
- создавать гипертекст (текст со "ссылками", например WWW странички);
- использовать библиотеку "стилей" и "мастеров";
- использовать глобальный поиск и замену информации в тексте.

Более подробно можно посмотреть в главе "Текстовые редакторы".

2. Табличный процессор Excel., (чаще его называют "электронные таблицы"), обеспечивают работу с данными организованными в виде таблиц.

Электронные таблицы предназначены в основном для расчетов.

Отличие Excel от других известных таблиц - часть таблицы можно "объявить" базой данных и эффективно использовать в составе таблицы.

3. Система управления базами данных (СУБД) Access

- позволяют управлять большими информационными массивами - базами данных - составлять по ним отчеты, производить в них поиск необходимой информации с помощью фильтрации. Все современные СУБД управляют только реляционными (табличными) базами данных, т.к. иерархические и сетевые базы данных слишком сложны.

Другие известные СУБД: Карат, Ребус, DBase, Clipper, Clarion, Paradox, FoxPro.

Преимущество Clipper и FoxPro по сравнению с другими СУБД – возможность создания специализированных баз данных по индивидуальным требованиям заказчика в виде выполнимых (.exe) файлов. При использовании таких баз не надо загружать саму СУБД!

Сейчас состав MicroSoft Office уже намного расширился. Хотелось бы упомянуть также PowerPoint - программное средство для создания и просмотра слайд-шоу, часто используемых при проведении различных презентаций.

Другие виды пользовательского ПО:

1) Графические редакторы;

- позволяют создавать многоцветные графические изображения, обрабатывать картинки введенные с помощью сканера; готовые изображения можно распечатать или использовать при издании книг, журналов, рекламы, ...

Самый простой - "PaintBrush", входящий в состав реквизитов Windows. Главный его недостаток - малое разрешение (число точек), поэтому линии идут "ступеньками". Наиболее популярен (но сложен) - "Corel Draw".

Для создания анимационных (мультипликационных) изображений рекомендуется использовать редактор "3-D Studio", а для обработки изображении введенных в ЭВМ с помощью сканера "PhotoShop".

2) Музыкальные редакторы - синтез музыки и звуковых эффектов.

3) Обучающие программы - в школах и институтах (например, тренажер клавиатуры др.)

4) Моделирующие программы, например тренажеры для обучения водителей, летчиков, космонавтов, операторов атомных станций или программы моделирующие поведение механизмов в экстремальных ситуациях (авариях).

5) Системы автоматизированного проектирования (САПР) - создание новой техники и исследование ее характеристик на моделях (например, Auto Cad – чертежи деталей, PCad - электрические схемы).

6) Бухгалтерские программы, программы для организации маркетинга.

7) Программы управления оборудованием, например, технологическим оборудованием (станками с ЧПУ, обрабатывающими центрами, технологическими линиями), вооружением кораблей, самолетов, полетом летательных аппаратов, работой экскаваторов, электросетей и т.п. Очень часто для уменьшения веса, габаритов и стоимости, если нет требования универсальности, такое ПО совмещает системные и потребительские функции (например, ПО для стиральной машины или цифровые часы). Специалисты, способные создавать такие мини-компьютеры наиболее ценятся и их очень мало.

Конечно, если это экономически допустимо, проще установить стандартный персональный компьютер и воспользоваться стандартным инструментальным ПО для создания нужной программы управления.

8) Игровые программы - для организации досуга.

Необходимо отметить, что число видов пользовательского ПО неисчислимо, как и число профессий людей.

Инструментальное программное обеспечение

- программы и пакеты программ предназначенные для создания системного и при-

кладного программного обеспечения.

К ним относятся:

- ЯВУ - языки высокого уровня (Паскаль, Си, Бэйсик и многие другие);
- ЯНУ - языки низкого уровня (Ассемблер, машинные коды);
- СУБД, ЭТ. если они используются в режиме программирования для создания пользовательских программ.

В ЯВУ операторы максимально приближению к "человеческому" языку, каждый оператор после компиляции преобразуется иногда в сотни машинных команд. В ЯНУ программирование идет непосредственно в машинных командах микропроцессора, программа максимально приближена к аппаратуре ("hardware"), поэтому она оптимальна по быстродействию и длине (программа не транслируется). При программировании на ЯНУ необходимо хорошо знать "hardware", адреса всех устройств, векторы прерывания и т.п. С помощью ЯНУ обычно пишут системное ПО, драйверы, вирусы или "вставки" в ЯВУ оптимизирующие работу программы. В ассемблере каждая команда - мнемоническое обозначение одной машинной команды, программирование идет на уровне РОНов (регистров общего назначения) и регистров состояния устройств ЭВМ.

P.S. В Windows 95 встроена программа – просмотрщик ("браузер") "Explorer". С ее помощью можно просматривать в InterNet WWW-странички.

ГЛАВА 6. Файловая система.

Понятие "файл." имеет корнями своего происхождения папку с документами в английском делопроизводстве. Однако сейчас понятие "папка" используется в Windows 95 и выше вместо понятия "каталог", а "файлу" сопоставляется понятие "документ". Файлы могут хранится на гибких и жестких дисках, на магнитной ленте стримера, на ленте видеомагнитофона, в ПЗУ или в ППЗУ (например, на картриджах записывают игры), на магнитных барабанах на вычислительных центрах, на перфолентах (для станков с ЧПУ). Поэтому одно из распространенных определений:

ФАЙЛ - поименованная область внешней памяти, выделенная для хранения однотипной информации. Имя файла записано в каталоге диска.

Каталог - список имен файла и его "реквизитов".

Исторически раньше на дисках был только один каталог, но когда объем диска вырос оказалось очень неудобно хранить все файлы в одном каталоге. Во первых трудно найти нужный файл, а во вторых нельзя хранить в одном и том же каталоге разные файлы с одинаковыми именами. Поэтому появились подкаталоги, которые на дисках образуют древовидную ("иерархическую") структуру.

Это необходимо по следующим причинам:

1) Удобство хранения и поиска файлов - все они разложены по нужным папкам (каталогам). Папка может находится внутри другой папки, глубина таких "вложений" не ограничена. Если каталог находится внутри другого он называется "подкаталогом", но для краткости его все равно зовут каталогом.

2) В разных папках могут бать файлы с одинаковыми именами, но разным содержанием.

Самый "верхний" (начальный) каталог на каждом диске называется "корневым" или "главным". Его имя совпадает с именем диска после которого ставится двоеточие и знак "\", который называется "обратный слеш". Например, "C:\" - корневой каталог системного диска винчестера.

Реквизиты файла, которые хранятся в каталогах.

1) Имя.
2) Расширение, три символа после точки, указывают тип файла.
3) Время и дата создания.
4) Объем в битах.
5) 4 атрибута: R - read only, "только для чтения";
H - hiden, "скрытые" файлы;
S - системные файлы;

A - архивные.

Информация на магнитном диске записывается и считывается магнитной головкой вдоль концентрических окружностей - дорожек. Их количество зависит от типа диска и накопителя на МД. Каждая дорожка на МД разбита на сектора. В одном секторе дорожки обычно 512 байт данных. Обмен данными между диском и внутренней памятью ЭВМ осуществляется последовательно целым числом сектором. Минимальное число таких секторов, которые могут быть считаны или записаны называется КЛАСТЕРОМ. Таким образом КЛАСТЕР - минимальная единица размещения информации на диске. Обычно это 512 байт. В ОЗУ - 1 бит! Понятно, что дисковое пространство при этом расходуется не рационально. Например, короткие файлы содержат совсем немного бит информации, а на диске все равно занимают пол килобайта! Если у Вас много очень маленьких файлов и полно каталогов они могут "съесть" пол-винчестера.

Почему же так все устроено? Если размер кластера сделать более маленьким магнитная головка дольше считывает информацию и очень сильно диск "тормозит" работу ЭВМ.

Файл может хранится в разных местах диска. Это не очень хорошо, т.к. замедляется время его считывания - головка магнитная "скачет" по дорожкам. Поэтому периодически надо информацию на винчестере "сжимать" ликвидируя свободные места между занятыми секторами с помощью специальной программы-утилиты.

Самая главная дорожка - нулевая. На ней хранятся дисковые компоненты – логическое имя, идентификатор диска, компоненты операционной системы, если диск системный и корневой каталог. В этом каталоге хранятся имена файлов и подкаталогов (если они есть), а также все реквизиты файлов (время создания, атрибуты, размер).

В каких секторах на диске лежат отдельные "куски" файла и их "контрольные суммы" записано в двух специальных таблицах расположения файлов "FAT". Одна из них резервная. Они расположены тоже на нулевой дорожке. Эта дорожка "главная" - служебная.

Что происходит при стирании файла или каталога? Стирается только первая буква в его имени в каталоге, поэтому его легко восстановить, но только до того пока сверху не записана другая информация.

Магнитный диск на котором хранится ДОС называется "системным". Если на ЭВМ есть винчестер, это диск "C". Для борьбы с вирусами и для устранения других аварийных ситуаций рекомендуется всегда иметь также и гибкий магнитный диск с ДОС.

Полное имя файла - имя файла с указанием пути поиска файла. Например, C:\windows\my.txt означает, что файл my.txt находится на диске "C" (винчестере) в каталоге "windows". При работе в ДОС длина имени файла не должна превышать 8 символов, "запрещенные" символы в имени файла "пробел", точка, запятая, "?", "*", надежнее использовать только английские буквы и т.п. В Windows 95 эти проблемы снимаются, можно давать файлам имена длиной до 256 символов, можно их именовать по-русски, с пробелами и пр.

Расширение чаще всего присваивается файлу автоматически той программой с помощью которой он был создан. Обычно расширения используются для определения типа файла. Существует взаимная договоренность о том, каким типам данных соответствует то или иное расширение. Такая договоренность удобна для универсализации способов обработки данных хранящихся в файлах.

"Программами" обычно называют исполнимые (выполнимые) файлы, которые запускаются из ДОС "сразу" по имени, без загрузки никаких других программ.

Такие файлы имеют расширение .exe, .com, .bat:

.exe, .com - исполнимые в машинных кодах

.bat - командные (исполнимые, но текстовые, они состоят из команд ДОС).

Некоторые другие общепринятые расширения файлов.

.txt, .doc , .hlp , .lst, - текстовые (.doc из под Word, .txt в кодах ASCII),

.pas,.bas,.c,.cpp - текстовые файлы, в которых программы на TP, Basic, C, C++,

.sys - системные в машинных кодах,

.pcx простейший графический,
.bmp графический под Windows (PaintBruch),
.htm - текстовый, стандарт HTML, например WWW-страничка,
.gif – анимационный.

Необходимо отметить, что файловая структура хранения информации на магнитных дисках соответствует иерархической модели баз данных. С этой точки зрения файл - это запись, а его реквизиты - поля записи.

ГЛАВА 7. Дисковая операционная система.

ДОС - это "посредник", связующее звено между аппаратной частью ЭВМ (hardware) и пользовательским программным обеспечением.

Без ДОС невозможно ни запустить, ни найти нужную программу, не будет функционировать ни монитор ни любое другое внешнее устройство ЭВМ. Поэтому ЭВМ продают с установленной ДОС. ДОС относится к системной части программного обеспечения ЭВМ (soft ware). Без ДОС функционирование современного персонального компьютера невозможно.

До ДОС существовала ленточная операционная система очень неудобна, т.к. был очень медлителен поиск нужного файла и не возможен "свопинг" т.е. временное хранение данных на носителе, которым являлась бумага (перфолента). С появлением магнитных носителей была внедрена файловая система. При этом дан ные хранятся на диске в виде отдельных файлов - поименованных областей дискового пространства. Файлы и каталоги образуют иерархическую систему хранения информации на дисках. Самый "верхний" (начальный) каталог на каждом диске называется "корневым" или "главным". Его имя совпадает с именем диска после которого ставится двоеточие и знак "\", который называется "обратный слеш". Например, "C:\" - корневой каталог системного диска винчестера. Магнитный диск на котором хранится ДОС называется "системным". Если на ЭВМ есть винчестер, это диск "C". Для борьбы с вирусами и для устранения других аварийных ситуаций рекомендуется всегда иметь также и гибкий магнитный диск с ДОС, его имя тогда будет "A".

ДОС предназначена для:

- управления всеми устройствами ЭВМ (посредник между программными средствами и аппаратурой, хотя некоторое ПО и непосредственно руководит некоторыми периферийными устройствами, например, принтером)
- организации взаимодействия файловой системы с ЯВУ (посредник между ЯВУ и машинными кодами)
- организацией хранения данных, их копирование, перенос и т.п.
- тестирование ЭВМ, парковка HDD, часики и пр. мелочи

Состав ДОС

Наиболее распространенной ДОС в мире на персональных компьютерах является MS DOS, разработанная фирмой Micro Soft. Для работы в сетях используется чаще "полуось" OS/2 или Unix. Эти системы в отличие от MS DOS многозадачные, то есть одновременно могут работать с несколькими пользователями. MS DOS обслуживает нескольких пользователей (например, принтер и CD ROM) последовательно, поэтому медленнее, просто это часто мало заметно.

Сейчас операционных систем становится все больше и больше.

Рассмотрим состав ДОС на примере MS DOS.

- BIOS (базовая система ввода/вывода) "зашита" в ПЗУ (ROM) на заводе изготовителе. Это большая специализированная интегральная схема на материнской плате. Часть этой микросхемы представляет собой ОЗУ, подпитанное аккумулятором или батарейкой. В этой ее части хранятся параметры BIOS - ее изменяемую часть. Эти параметры мож-

но изменить с помощью программы "SetUp", которая хранится в этой же микросхеме и вызывается клавишей Delete в начале загрузки ЭВМ. В первую очередь это тип HDD, НГМД, установка компьютерных часов (таймера) и др. Таймер находится в этой же микросхеме и питается от того же аккумулятора.

BIOS тестирует ЭВМ и "учит" ее, как обращаться к нулевой дорожке системного диска, где находится уже системный загрузчик SYSTEM BOOT., который начинает загружать основное ядро ДОС, файлы io.sys и msdos.sys, находящиеся в коренном каталоге системного диска. Затем загружается:

- командный процессор command.com, который может выполнять все внутренние команды ДОС (dir, md, rd, copy, ren, cd и т.п.). Потом запускается:

- config.sys, файл который состоит из установок ДОС (его можно редактировать F4, как текстовый в Norton Commander). Затем запускается:

- autoexec.bat, командный файл "автозагрузчик", тоже текстовый, состоящий из команд ДОС и обычно запускающий Norton Commander и :

- драйверы (программы управляющие периферийными устройствами, например, мышкой - mouse.com, клавиатурой - руссификатор rk1.com, звуковой картой). Еще есть:

- утилиты (программы для обработки информации), отвечающие за выполнение внешних команд, например, format.exe (их имена совпадают с именами команд), архиваторы и пр. Они обычно находятся в каталоге DOS.

Ситуация с Windows 95 и выше несколько особая. Это уже "операционные оболочки". Это значит, что они включают в себя ДОС. Поэтому, конечно, ее организация несколько другая. Выйти из Windows 95 в ДОС можно или запустив Norton Commander или через кнопку "Пуск".

КОМАНДЫ ДОС

Команды в ДОС записываются в командной строке после "приглашения", которое указывает какой диск и какой каталог является рабочим.

Рабочим или текущим называется тот каталог с которым мы работаем. При обращении к этому каталогу в командах ДОС не надо прописывать пути его поиска. Например команда DIR выведет на экран список файлов текущего каталога, а "DIR .." родительского каталога (прописан неполный путь). Путь к рабочему каталогу прописан в приглашении.

Пример приглашения: C:\MY\PROBA>_ Его анализ говорит нам, что рабочим является диск "C", а на нем рабочий каталог "PROBA" внутри каталога "MY". Рабочим является тот каталог, на чье имя оканчивается приглашение.

Все команды ДОС делятся на внутренние и внешние.

Всего команд больше ста.

Некоторые внутренние команды ДОС:

1) DIR - просмотр содержания (списка файлов) рабочего каталога (директория). Например, C:\MY\PROBA>DIR на экране появится содержимое каталога PROBA

2) "Имя диска и двоеточие" (например C:\>A:) - смена рабочего диска. После такой команды приглашение изменится на A:\>_ (если в дисководе есть гибкий диск!)

3) CD (change dir) - смена рабочего каталога. Например, после команды C:\MY>CD PROBA приглашение изменится на C:\MY\PROBA, если конечно внутри каталога "MY" существует каталог "PROBA"!

Очень удобная команда "CD .." - возвращение в предыдущий ("верхний" или "родительский" каталог), в результате он становится рабочим. Например, после команды "C:\MY\PROBA>CD .." приглашение изменится на "C:\MY>_" и рабочим станет каталог MY, а не PROBA

4) MD (make dir) - создание нового каталога. Например, после команды C:\MY>MD PROBA2 в каталоге "MY" появится новый каталог "PROBA", в чем можно убедиться выполнив команду DIR.

5) DEL - удаление файла или группы файлов,

например, C:\MY\PROBA>DEL my.pas удаляется файл my.pas в каталоге PROBA

6) RD - удаление каталога, например, C:\MY\PROBA>RD MY удаляется каталог MY внутри каталога PROBA, если он там есть.

7) COPY - копирование файла или группы файлов, например,

C:\MY\PROBA>COPY my.pas .. копирование файла my.pas из каталога PROBA в родительский каталог MY или

C:\MY\PROBA>COPY my.pas my.bak копирование файла my.pas из каталога PROBA в этот же каталог, но с изменением расширения файла на .bak или

C:\MY\PROBA>COPY my*.pas my.bak копирование всех файлов начинающихся на my и с расширением .pas в один файл my.bak. Так "сливают" много файлов в один, или

C:\MY\PROBA>COPY my.pas+fire.pas my.bak копирование (слияние) двух файлов в один.

8) TIME - время по компьютерным часам

9) DATE - дата по компьютерным часам

10) PATH - создание "прозрачных" каталогов, например, PATH c:\tp60

Все вышеперечисленные команды - "внутренние", т.е. для их выполнения не нужны никакие внешние файлы. Эти команды выполняются под управление командного процессора command.com, который постоянно загружен в ОЗУ.

ВНЕШНИЕ КОМАНДЫ.

Команды для выполнения которых необходимы дополнительные файлы (обычно одноименные) называются "внешними", а соответствующие им файлы "утилитами".

Пример внешней команды: C:\MY\PROBA>arj e my.arj (разархивация файла "my.arj"). Ей соответствует файл (утилита) "arj.exe" (архиватор "arj").

Внутренние команды более "быстрые", т.к. для их выполнения не должны запускаться утилиты.

КАК УСТАНОВИТЬ ДОС НА КОМПЬЮТЕР?

ДОС невозможно скопировать на диск, например с дискеты на испорченный винчестер. Существует несколько способов установки ДОС.

1) Использовать внешнюю команду SYS, которая переносит системные компоненты, например a:\>SYS a: c: скопирует системные компоненты с дискеты на винчестер. При этом на дискете должна быть утилита sys.exe

2) Отформатировать диск с установкой системных компонент, например с помощью утилиты format, например c:\>format a: /s В этой команде "/s" - ключ команды format, который должен использоваться для создания системного диска. Форматируется дискета. На винчестере должен быть файл format.sys или в корневом или в "прозрачном" каталоге. В процессе форматирования надо правильно ответить на вопросы ЭВМ. После форматирования ЭВМ сообщит сколько на диске плохих блоков, каков объем свободного дискового пространства, идентификатор диска и его имя.

3) Воспользоваться возможностями ДОС Навигатора или Windows.

ДЛЯ ЧЕГО РЯДОВОЙ ПОЛЬЗОВАТЕЛЬ ДОЛЖЕН ХОТЯ БЫ НЕМНОГО УМЕТЬ РАБОТАТЬ В ДОС?

1) Чтобы уметь создать простейший командный файл, например, автоматически архивирующий в конце рабочего дня все его файлы на дискету.

2) Уметь редактировать autoexec.bat и config.sys

3) Уметь загрузится с системной дискеты, если ЭВМ не загружается с винчестера и запустить антивирусную программу.

ГЛАВА 8. Арифметические основы построения эвм.

История возникновения системы счисления. Классификация систем счисления.

Система счисления - совокупность приемов и правил изображения чисел цифровыми знаками. Другими словами - это способ кодирования числовой информации.

В древности, когда в мире еще не существовало "единого информационного пространства" люди пользовались самыми разнообразными мерами счета. Например, в древнем Вавилоне час делили на 60 минут, окружность на 360 градусов. А в римской системе счисления максимальное число было 40 000. Англосаксы считают дюжинами. На Руси считали по 40 (например, вспомните выражение "сорок сороков"). Французы использовали двадцатки.

Первая система счисления, по видимому возникла в Египте. У египтян существовали иероглифы: один, десять, сто, ... , десять миллионов. Все остальные числа записывались с их помощью и операции сложения. Недостаток римской и египетской систем счисления состоит в том, что от величины числа зависит количество используемых для его записи знаков. Иероглифа более 10.000.000 не было, египтяне так же, как и римляне для записи больших чисел вынуждены как-то приспосабливаться. Самый большой недостаток таких систем счисления, которые называются "непозиционные" - очень сложно умножать числа, а делить почти невозможно (например, воспользоваться алгоритмом "деление уголком" Вам не удастся). Египетская система очень громоздкая - для записи 9 надо 9 раз написать иероглиф "1". В римской системе счисления все же уже заложена идея современных позиционных систем счисления.

Позиционной системой счисления называется такая система счисления, в которой значение знака (цифры) в числе зависит от позиции, в которой находится этот знак.

В средневековой Руси была принята славянская алфавитная нумерация, которая также как и римская, напоминала современную позиционную систему счисления, но числа в ней были закодированы буквами, а чтобы избежать путаницы над ними ставился специальный знак ~ - "титло" , например, 1 = А , 2 = Б , 3 = В и т.д.

Одной буквой в славянской системе счисления кодировались числа от 1 до 9, двумя 10, 20, ... 90 и т.д. Например, современная десятичная система счисления пришла к нам из Индии, где она появилась не позднее VI века нашей эры. Любое, сколь угодно большое, число в этой системе кодируется с помощью не более чем 10 цифр и его можно представить в виде суммы степеней числа 10 множенных на число соответствующее цифре в данном разряде. Показатель степени при этом - позиция разряда, разряды нумеруются справа налево, начиная от 0 (!). Например для числа 589, этот принцип можно выразить с помощью следующей формулы:

Номера разрядов: 2 1 0
Число: (5 8 9) D = 5*100 + 8*10 + 9*1,

где 100, 10, 1 - веса разрядов ($100=10^2$, $10=10^1$, $1=10^0$), соответствующие номеру разряда степени основания системы счисления 10. Во избежании путаницы числа записываются с указанием системы счисления:

буквой D - обозначают числа, записанные в десятичной системе счисления;
буквой B - числа, записанные в двоичной системе счисления;
буквой C - числа, записанные в восьмеричной системе счисления;
буквой H - числа, записанные в шестнадцатеричной системе счисления.

Таким образом любое десятичное число можно представить в виде суммы единиц, десятков, сотен, тысяч и т.д. Минимальное число, которое может быть записано в каждом конкретном разряде называется ВЕСОМ РАЗРЯДА. В десятичной системе счисления для нулевого разряда это - единица, для первого - десять, для второго сто и так далее.

По аналогии, для всех других позиционных систем счисления можно записать следующую "универсальную формулу" для выражения любого числа в виде суммы слагаемых:

Номера разрядов: 3 2 1 0
Число: (... d c b a) $K = a*K^0 + b*K^1 + c*K^2 + d*K^3 + ...$

где К является основанием конкретной позиционной системы счисления.

ОСНОВАНИЕМ позиционной системы счисления называется количество знаков, которые используются для кодирования чисел. Например, для двоичной системы счисления это - 2, для десятичной - 10.

ВЕСОМ РАЗРЯДА в этой формуле является степень основания системы счисления, в которой показатель степени - номер разряда. Для двоичной системы счисления веса разрядов до 10 разряда: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024. Вес десятого двоичного разряда служит мерой измерения количества информации:

- один килобайт (1 К) это не 1000 байт, а 1024 !
- один мегабайт (1 М) - 1024 К;
- один гигабайт (1 Г) - 1024 М.

В универсальной формуле каждое слагаемое - это произведение веса разряда на цифру записанную в этом разряде.

Десятичная система счисления выбрана человечеством в качестве основной из соображений удобства устного счета, т.к. у нас 10 пальцев на руках. В древнем Вавилоне, также была принята позиционная, но 60-ричная система счисления, и сначала в ней не было 0. Ее недостаток - огромная таблица умножения, ее никто не мог запомнить. Эту таблицу умножения всегда приходилось "держать под рукой" при умножении.

Преимущества и недостатки использования двоичной системы счисления.

Двоичная система счисления была придумана математиками в XVII-XIX веках и потом забыта, и только в 1936-1938 г.г. Клод Шенон применил ее для конструирования электронных схем.

Достоинства этой системы счисления - цифры, которые в ней используются очень легко закодировать (представить) в виде простейших электрических сигналов - "Лог.0" (низкий уровень сигнала), "Лог.1" (высокий уровень сигнала). Во всех современных ЭВМ обмен информацией и управление всеми устройствами осуществляется с помощью машинных слов (двоичных чисел). Каждый разряд передается по отдельному проводу. Если их не хватает (мала разрядность ЭВМ), то они передаются за несколько приемов последовательно по одним и тем же проводам.

Разрядность ЭВМ определяется разрядностью шины данных канала ввода - вывода (общей шины), а также разрядностью центрального процессора (обычно они совпадают). Начиналось все с 8-ми разрядных, у нас 16-ти, а современные 32-х разрядные ЭВМ уже везде. Уже появились и 64-х разрядные! Недостаток двоичной системы счисления - для записи даже небольших чисел надо использовать много цифр. Например 98 в двоичной системе счисления - 1100010, цифр в 3 с половиной раза больше, чем в десятичной. Второй недостаток - не специалисту трудно в этой системе разобраться.

Перевод чисел из десятичной системы счисления в системы с другим основанием и обратно.

Рассмотрим перевод двоичного числа 1100010 в десятичное. Для перевода числа из двоичной системы счисления в десятичную необходимо пронумеровать справа налево все разряды двоичного числа. Необходимо запомнить, что нумерация начинается не от "1", а от "0"! Для рассматриваемого примера номера разрядов следующие - 0, 1, 2, 3, 4, 5, 6. Для перевода числа в десятичную систему счисления необходимо, согласно "универсальной формуле", перемножить цифру находящуюся в каждом разряде двоичного числа на вес данного разряда и сложить полученные результаты:

6 5 4 3 2 1 0 - номера разрядов
(1 1 0 0 0 1 0) B = 1*64+1*32+0*16+0*8+0*4+1*2+1*1 = 64+32+0+0+0+2+1 = (99) D

Для перевода дробной части двоичного числа в десятичное применяется следующее правило:

1) необходимо пронумеровать двоичные разряды числа от запятой слева направо, причем отсчет ведется от -1 с шагом -1. Например, для числа (0,1011) номера разрядов -1,-2,-3,-4

2) веса десятичных разрядов расположенных слева от запятой также как и для целой части числа представляют собой степень основания системы счисления, у которой показатель степени - номер разряда, но так как они теперь отрицательные, веса разрядов приобретают значения 1/2, 1/4 , 1/8, 1/16 и так далее.

Согласно "универсальной формуле":

(0,1011) B = 1*(1/2)+0*(1/4)+1*(1/8)+1*(1/16) = (8+2+1)/16 = 11/16 = (0,6875) D

Восьмеричная и шестнадцатеричная системы счисления.

При программировании часто, кроме двоичной системы счисления, используют также восьмеричную и шестнадцатеричную системы счисления. В этих системах счисления запись числа более компактна, поэтому ее часто используют при выводе данных на экран ЭВМ, при программировании в машинных кодах или на ассемблере.

В восьмеричной системе счисления 8 цифр. Используются арабские цифры от 0 до 7. Восьмерка в восьмеричной системе счисления - это уже 10, так как младший разряд переполняется! Девятка - (11) C.

В шестнадцатеричной системе счисления 16 цифр. Так как арабских цифр для нее не хватает, используются первые заглавные латинские буквы:

10-тичная	16-ричная
10	A
11	B
12	C
13	D
14	E
15	F

Число 16 в шестнадцатеричной системе счисления - (10) H, а 17 - (11) H.

Перевод чисел из восьмеричной системы счисления в десятичную.

Переведем число 1327 из восьмеричной системы счисления в десятичную, воспользовавшись "универсальной формулой":

(1327) C = 7*1+2*8+3*64+1*512 = 7+16+192+512 = (727) D

Вышеуказанный способ перевода восьмеричных чисел трудоемкий, так как надо помнить степени восьмерки, поэтому программисты чаще переводят число сначала из восьмеричной системы счисления в двоичную, а затем в десятичную.

Алгоритм перевода из восьмеричной системы счисления в десятичную через двоичную:

1) каждую цифру в восьмеричном числе надо слева направо преобразовать в двоичное число содержащее 3 цифры (триаду). Если получившиеся число содержит менее трех разрядов, то слева у него надо дописать необходимое количество нулей. Необходимо чтобы каждое число обязательно была записано в виде двоичной триады! Полученные триады записываются друг за другом. Для нашего числа

(1327) C = (001 011 010 111) B.

В результате мы перевели исходное число в двоичную систему счисления!
2) перевести полученное двоичное число в десятичное по известному алгоритму:

1*512+0*256+1*128+1*64+0*32+1*16+0*8+1*4+1*2+1*1 =
= 512+128+64+16+4+2+1 = (727)D
Результат совпадает!

Перевод чисел из шестнадцатиричной системы счисления в десятичную.

Переведем число A1C из шестнадцатеричной системы счисления в десятичную, воспользовавшись "универсальной формулой":

(A1C) H = A*256+1*16+C*1 = 10*256+1*16+12*1 = 2560+16+12 = (2588) D

Вышеуказанный способ перевода шестнадцатиразрядных чисел очень трудоемок, так как надо помнить степени шестнадцати, поэтому программисты чаще сначала тоже, как и для восьмеричных чисел, переводят искомое число в двоичную систему счисления, а только затем в десятичную. Алгоритм перевода:
1) каждую цифру в шестнадцатеричном числе надо слева направо преобразуется в двоичное число содержащее 4 цифры (тетраду). Если получившееся число содержит менее четырех разрядов, то слева у него надо дописать необходимое количество нулей. Необходимо чтобы каждое число обязательно было записано в виде двоичной тетрады! Полученные тетрады записываются друг за другом. Для нашего числа

(A1C) H = (1010 0001 1100) H

В результате мы перевели исходное число в двоичную систему счисления!
2) перевести полученное двоичное число в десятичное по известному алгоритму:

1*2048 + 0*1024 + 1*512 + 0*256 + 0*128 + 0*64 + 0*32 + 1*16 + 1*8 + 1*4 + 0*2 + 0*1 =
2048 + 512 + 16 + 8 + 4 = (2588) D
Результат совпадает!

Алгоритмы перевода чисел в десятичную систему счисления.

Перевод из десятичной системы счисления в двоичную. Если число небольшое всегда это можно сделать устно, определив какой набор максимальных степеней числа 2 есть в исходном числе. Например:

(25) D = 16+8+1 = $2^4 + 2^3 + 2^0$ = (11001) B

В остальных случаях надо воспользоваться следующем алгоритмом:
1) Последовательно делить исходное число на 2 с отбрасыванием остатка до тех пор, пока результат не станет меньше двух. Остаток, ноль или единицу записывают слева в той же строке где и результат, но в отдельный столбец. Например, для числа 99:

Остаток	Деление
0	98:2=49
1	49:2=24
0	24:2=12
0	12:2=6

0 6:2=3
1 3:2=1

2) Получившиеся остатки и результат от последнего деления и представляют собой искомое двоичное число. Записывать результат надо от результата последнего деления снизу вверх: (99) D = (1100010) B. Это число рассматривалось выше!

Для перевода дробной части десятичного числа в двоичную используют следующее правило, например для числа 0,6875:

1) исходное число последовательно умножают на 2, причем целая часть результата записывается в левый столбик. Если целая часть - единица, то она отбрасывается при следующем умножении.

Целая часть	Умножение
1	0,6875*2=1,3750
0	0,3750*2=0,7500
1	0,7500*2=1,5000
1	0,5000*2=1,0000

Умножение надо производить до тех пор пока после запятой не появятся одни нули. Однако, очень многие десятичные дробные числа не удается точно переводить в двоичные и справа никогда не появятся одни нули. В этом случае перевод производят с определенной точностью, за некоторое разумное число шагов.

2) Левый столбик - искомое число. Его надо записать после запятой, начиная от верхней цифры. Таким образом 0,6875 =(1011)2, это число уже использовалось выше.

Перевод восьмеричных и шестнадцатеричных чисел в десятичные.

Если при переводе чисел из десятичной системы счисления в восьмеричную и шестнадцатеричную системы воспользоваться тем же принципом что и при переводе в двоичную, то есть последовательно делить нацело исходное число на 8 или 16 то это будет очень трудоемко. Поэтому программисты сначала переводят исходное число в двоичную систему счисления, а затем полученное двоичное число в десятичное.

Правило перевода десятичных чисел в восьмеричные.

1) Десятичное число переводится в двоичное.

2) Двоичное число разбивается на группы по 3 разряда (триады) справа налево. Если в самой левой группе меньше чем 3 цифры ее можно дополнить слева нулями.

3) Каждая группа (триада) отдельно преобразуется в восьмеричную цифру.

ПРИМЕР. Исходное число (727) D

1) Переведя его в двоичное, получим 1011010111.

2) Разбиваем его на триады: 001_011_010_111.

3) Каждую триаду заменяем восьмеричной цифрой: 1327 - это и есть искомое число в восьмеричной системе счисления.

Правило перевода десятичных чисел в шестнадцатеричные.

1) Десятичное число переводится в двоичное.

2) Двоичное число разбивается на группы по 4 разряда (тетрады) справа налево. Если в самой левой группе меньше чем 4 цифры ее можно дополнить слева нулями.

3) Каждая группа (тетрада) отдельно преобразуется в шестнадцатеричную цифру.

ПРИМЕР. Исходное число (2588) D

1) Переведя его в двоичное, получим 101000011100.

2) Разбиваем его на тетрады 1010_0001_1100.

3) Каждую тетраду заменяем шестнадцатеричной цифрой: A1C - это и есть искомое число в шестнадцатиричной системе счисления.

Арифметические действия над вещественными числами в двоичной системе.

СЛОЖЕНИЕ ДВОИЧНЫХ ЧИСЕЛ.

В двоичной системе счисления:
1) 0+0=0
2) 1+0=1
3) 1+1=10,
так как младший разряд переполняется, в более старший разряд переходит единица переноса. Переведя 10 в десятичную систему счисления можно убедиться, что это 2, то есть результат верен!

Используя эти три правила легко складывать двоичные числа в столбик.
Например:

Перенос:	1	1	1	1			Проверка
1-е число:		1	1	0	1	1	=1+2+8+16=27
2-е число:		1	0	1	1	0	=2+4+16=22
Сумма:	1	1	0	0	0	1	=1+16+32=49

ВЫЧИТАНИЕ ДВОИЧНЫХ ЧИСЕЛ.

В двоичной системе счисления:
1) 0-0=0;
2) 1-0=1;
3) 1-1=0;
4) 0-1=1,
если есть возможность произвести заем в более старшем разряде. При выполнении операции "заем" из более старшего разряда "занимают" единицу и она "распадается" на две единицы в более младшем разряде. После вычитания из этих двух единиц одной остается одна. Если слева от рассматриваемого разряда "занять" не удается (там ноль), то надо занимать в более старшем разряде. Учтите что единица, которую удастся занять распадается на две и только одну из них надо переносить в более младший разряд, а вторая остается! Эта процедура может повторятся несколько раз.

ПРИМЕР

Заем:	1	1	1	1	1			ПРОВЕРКА
Уменьшаемое:	1	0	0	0	1	0	1	=1+4+64=69
Вычитаемое		1	0	1	1	1	0	=2+4+8+32=46
Разность	0	0	1	0	1	1	1	=1+2+4+16=23

Операция вычитания более сложная чем сложение чисел. Основная арифметическая операция которую "умеет" выполнять арифметико-логическое устройство центрального процессора компьютера - сложение двоичных чисел. Все другие арифметические и алгебраические действия сводятся к сложению. Вычитание в ЭВМ сводится к сложению, для чего вычитаемое число записывается в дополни- тельном коде (см. ниже).

Понятие кода (прямой, обратный, дополнительный).

Рассмотрим кодирование целых чисел в компьютере. Прямой код - запись числа в двоичной системе счисления. При этом знак числа кодируется с помощью старшего разряда. Единица в старшем разряде означает, что закодировано отрицательное число.

Дополнительным или обратным называется такой код числа для которого справедливо утверждение: " При сложении прямого и обратного кода одного и того же числа все младшие разряды переполняются, в них появляются нули, единица появиться в следующем старшем разряде." Например, прямой код 0111, дополнительный 1001, их сумма 10000.

Существует простой алгоритм преобразующий прямой код в обратный:

1) Получить инверсию прямого кода (инверсный код). Для числа 0111 – это 1000;
2) Образовать дополнительный код числа путем добавления 1 к полученному инверсному коду: получится 1001.

ПРАВИЛО ВЫЧИТАНИЯ ДВОИЧНЫХ ЧИСЕЛ С ИСПОЛЬЗОВАНИЕМ ДОПОЛНИТЕЛЬНОГО КОДА.

Предположим, уменьшаемое 1000101, вычитаемое 101110.

1) Прямой код вычитаемого необходимо преобразовать в дополнительный: инверсный код вычитаемого 010001, дополнительный 010001+1=010010
2) Сложить прямой код уменьшаемого и дополнительный код вычитаемого:

```
    1  0  0  0  1  0  1
+      0  1  0  0  1  0
   1  0  1  0  1  1  1
```

3) В двоичном числе, которое получилось в результате сложения отбросить самый старший разряд: 010111 - это и есть искомое число! Полученное правило намного проще представленного выше.

Представление числа в виде с фиксированной и плавающей запятой.

Числа в ЭВМ представляются для экономии памяти в так называемой потенциальной форме, или, иначе говоря, "с плавающей запятой". При этом число разбивается на две составляющих: мантисса и порядок числа. Напр.: 0.22E-5 означает 0,22*10 в -5 степени т.е 0,00022. Такая запись кроме того экономит место на экране. Например, если число не помещается в ячейку электронной таблицы, в Works оно будет выводиться в ней в потенциальной форме.

В памяти ЭВМ мантисса и порядок числа храниться отдельно в разных частях машинного слова. Таким образом хранятся только вещественные числа (real).

В ЭВМ с фиксированной запятой числа в памяти не хранятся (не удобно), но на экран выводить в таком виде предпочтительней. Например, в Паскале: write(x:8:2); будет выведено значение переменной x, для числа будет отведено 8 знакомест, из них 2 для дробной части; число будет выведено с фиксированной запятой.

ГЛАВА 9. Логические основы построения ЭВМ.

Понятие "логическое высказывание". Представление об алгебре логики.

Логика.- наука, изучающая методы установления истинности или ложности одних высказываний на основе истинности или ложности других высказываний.

Логические высказывания (утверждения) - объекты логики, которые могут быть или истинными или ложными. Под высказыванием понимается всякое предложение, в котором содержится смысл утверждения (истинности) или отрицания (ложности). Рассматриваются только два значения высказывания: истинное или ложное (1 или 0). Такое условие алгебры логики приводит к соответствию между логическими высказываниями и двоичными цифрами в двоичной системе счисления, что позволяет описывать работу

схем и блоков ЭВМ, проводить их анализ и синтез с помощью алгебры логики. Высказывания могут быть сложными и простыми.

Сложные высказывания состоят из простых, соединенных логическими связками (NOT, OR, AND)..

Пример простого высказывания: А- "лампа горит". Если лампа действительно горит, то А=1, иначе А=0. Другой пример из математики: высказывание А – "X>Y", если X=5, а Y=1, то значение А=0 (ложь).

Пример сложного высказывания из математики. Дана разрывная функция:

Y = -1, если X<-2;

Y = 0, если -2<X<2;

Y = 1, если X>2.

При построении графика этой функции по точкам необходимо каждый раз проверять истинность или ложность высказываний, стоящих после слов "если".

При построении части графика Y=0, проверяется истинность высказывания "X>-2 и X<2". На языке алгебры логики это высказывание можно записать следующим образом: (X>-2) AND (X<2).

Алгебра логики (математическая логика) - наука (раздел математики), которая устанавливает истину не с помощью экспериментов, а на основании уже установленных фактов чисто формально с помощью рассуждений.

Основными логическими операциями алгебры логики являются:

1) логическое отрицание (операция "НЕ", инверсия): NOT A; если имеется некоторое высказывание А и оно истинно, то NOT A - ложно и наоборот. Другими словами для логического высказывания справедливо следующее правило: NOT (0) = 1, NOT (1) =0.

2) логическое умножение (операция "И", конъюнкция): A AND B; принято читать А и В. Высказывание A AND B истинно только в том случае, если А истинно (А=1), и В истинно (В=1). Во всех остальных случаях это высказывание ложно. Другими словами при логическом умножении справедливы следующие правила:

1 AND 1 = 1, 0 AND 1 = 0, 1 AND 0 = 0, 0 AND 0 = 0. Это правило справедливо и для большего числа сомножителей.

3) Логическое сложение (операция "ИЛИ", дизъюнкция): A OR B; принято читать А или В. Высказывание A OR B истинно в том случае, если хотя бы одно из высказываний А или В истинно (А=1 или В=1). В противном случае это высказывание ложно. Другими словами при логическом сложении справедливы следующие правила: 1 OR 1 = 1, 0 OR 1 = 1, 1 OR 0 = 1, 0 OR 0 = 0. Это правило справедливо и для большего числа слагаемых.

4) Сложение по модулю 2 (операция "исключающее ИЛИ"): A XOR B. Высказывание A XOR B истинно в том случае, если только одно из высказываний А или В истинно (А=1, В=0 или А=0, В=1). В противном случае это высказывание ложно.

Другими словами при сложении по модулю 2 справедливы следующие правила:

1 XOR 1 = 0, 0 XOR 1 = 1, 1 XOR 0 = 1, 0 XOR 0 = 0.

Предмет логики - анализ различных логических связей и методы построения на их основе правильных логических рассуждений.

Основные базовые логические элементы ЭВМ.

В ЭВМ логические операции над двоичными числами выполняются схемами, называемыми логическими элементами. Электрическое напряжение в этих схемах бывает двух уровней: высокое (соответствует двоичной 1) и низкое (соответствует двоичному 0). Представление чисел с помощью электрических сигналов позволяет конструировать различные электронные логические схемы.

Для описания законов работы логических схем используется математический аппарат алгебры логики. Переменные x1, x2,..., xn называются двоичными, если они могут принимать значения 0 или 1. Функция от двоичных переменных f (x1, x2,..., xn) называется переключательной функцией (булевой функцией). Эта функция также может принимать значения только 1 или 0.

Сумматор.- электронное логическое устройство, аппаратным путем складывающее

поразрядно двоичные числа и вырабатывающее сигнал переноса при переполнении. Состоит из соединенных между собой одноразрядных сумматоров, которые складывают только два двоичных разряда Xi, Yi и сигнал переноса из младшего разряда Pi-1, вырабатывают сигнал переноса в старший разряд Pi.

Триггер (бистабильная ячейка) - логическое электронное устройство с двумя устойчивыми состояниями. Триггер - элементарная ячейка памяти (ОЗУ),позволяющая хранить 1 бит информации. Простейший триггер - имеет 2 выхода и 2 входа. Если на одном выходе "1", то на другом - "0"! Вход установки триггера в "1" (S-вход) устанавливает триггер в "1" (на прямом выходе), если на него кратковременно подать управляющий сигнал. И, напротив, R-вход (ReSet) "сбрасывает", обнуляет триггер. Обычно при включении питания все триггеры сбрасываются.

Триггеры чаще всего используются в микропроцессоре в составе РОНов, а в основном ОЗУ сейчас используют другие более быстрые приборы, например, используют заряд конденсаторов, но их надо все время подзаряжать, поэтому такое ОЗУ называют дина-

Элемент "И" (y=x1 AND x2)

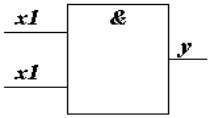


Таблица истинности

x1	x2	y = x1 AND x2
0	0	0
0	1	0
1	0	0
1	1	1

Элемент "ИЛИ" (y=x1 OR x2)

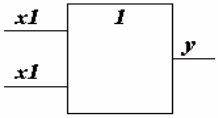


Таблица истинности

x1	x2	y = x1 OR x2
0	0	0
0	1	1
1	0	1
1	1	1

Элемент "НЕ" (y= NOT x)

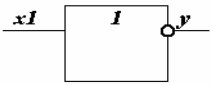


Таблица истинности

x	y = NOT x
0	1
1	0

Элемент "И-НЕ": y=NOT(x1 AND x2)

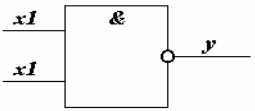


Таблица истинности

x1	x2	y = NOT(x1 AND x2)
0	0	1
0	1	1
1	0	1
1	1	0

Элемент "ИЛИ-НЕ": y=NOT (x1 OR x2)

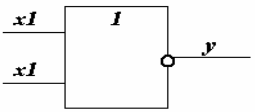


Таблица истинности

x1	x2	y = NOT (x1 OR x2)
0	0	1
0	1	0
1	0	0
1	1	0

Одноразрядный сумматор

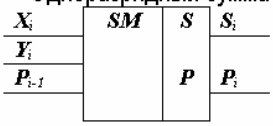


Таблица истинности

X_i	Y_i	P_{i-1}	S_i	P_i
0	0	0	0	0
0	0	1	1	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

мическим.

Триггеры - основной элемент "сверхбыстрого" ОЗУ - регистров общего назначения (РОНов), которые находятся непосредственно в микропроцессоре, где он (ЦП) хранит все промежуточные результаты выполнения микрокоманд.

Регистр - устройство, предназначенное для приема и запоминания N - разрядного кода, представляет собой совокупность триггеров у которых объединены некоторые входы (например, обнуление и тактовый). Сколько в машинном слове двоичных разрядов (разрядность ЭВМ) - столько в регистре триггеров.

В ЦП 16 РОНов (их обозначают буквами A, B, C, D и т.д. и используют при программировании на ассемблере или в машинных кодах) к ним непосредственно обращаются из

Триггер

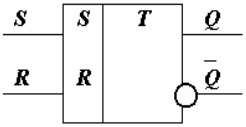


Таблица истинности

t	t	t+1	Примечание
R	S	Q	
0	0	0/1	Хранение 0 или 1
0	1	1	Установка 1
1	0	0	Установка 0
1	1	--	Запрещено

программы. Эти языки называют языками "низкого уровня" - потому что они максимально приближен к архитектуре ЭВМ (нет "посредников" вроде DOS или ЯВУ) и программы написанные с их помощью самые короткие, компактные и быстрые.

Регистры и триггеры обычно тактируют, т.е. они воспринимают информацию только во время воздействия тактового импульса (для синхронизации работы всех частей ЭВМ).

Произвести анализ схемы - это значит написать логическое высказывание по известной схеме.

1. На схеме обозначить промежуточные логические высказывания после каждого элемента.
2. Для этих высказываний в таблице отвести отдельные графы.
3. Последовательно слева направо заполнить таблицу.

Синтез схемы логического элемента - построение по таблице истинности логической схемы:

1. Подчеркнуть "1" в результирующем столбике.
2. Входные воздействия в каждой из этих строк объединить связкой "И", причем, если входной сигнал "0", надо его инвертировать.
3. Полученные высказывания объединить связкой "ИЛИ".
4. По полученному логическому высказыванию строим логическую схему, начиная

например от выхода, изображая логические элементы в обратном порядке действий, либо наоборот от входов, объединяя их с помощью логических элементов в порядке действий.

Упрощение схемы:

Упростить полученную схему можно используя теорему Де Моргана (заменить для отдельных частей схемы "или" на "и" или наоборот, при этом инвертировав входные и выходные сигналы), либо раскрыв скобки в логическом выражении и произведя логические действия.

Основные правила преобразования формул в алгебре логики:

(Для логических операций умножения и сложения допускается использовать знаки "*" и "+" соответственно).

1. A * not(A) = A * 0 = 0.
2. A + not(A) = A + 1 = 1.
3. A + A = A * A = A * 1 = A + 0 = A.
4. A * B = B * A;A + B = B + A.
5. (A * B) * C = A * (B * C) =A * B * C.
6. (A + B) + C = A + (B + C) =A + B + C.
7. (A * B) + C = (A + C) * (B + C).
8. (A + B) * C = (A * C) + (B * C).
9. Not(A * B) = not(A) + not(B); not(A + B) = not(A) * not(B);
10. A xor B = (A * not(B)) + (not(A) * B) = (A + B) * (not(A) + B)

Порядок построения логической схемы:

1. По словесному описанию составить таблицу истинности.
2. Синтезировать схему по этой таблице.

Глава 10. Модели и моделирование.

Содержание понятия "модель".

Информатика имеет дело с реальными и абстрактными объектами. Информация, циркулируя в реальном виде, присутствует в различных физических процессах, но в информатике она выступает как некоторая абстракция. Таким образом вместо реальных объектов в компьютерах надо использовать специальные абстрактные (формализованные) модели той физической среды, в которой "живет" информация в реальном мире.

Под **моделью** некоторого объекта понимают другой объект (реальный, знаковый или воображаемый), отличный от исходного, который обладает существенными для целей моделирования свойствами и, в рамках этих целей, полностью заменяет исходный объект.

Моделированием называется область знаний, занимающаяся разработкой разнообразных моделей, их теорией и использованием.

Основные виды моделирования:

- **Математическое** моделирование (это множество M, состоящее из элементов произвольной природы, на которых определено конечное множество отношений R1,R2,...,Rn. Эти отношения задают взаимосвязь между элементами моделируемого объекта.). Пример: уравнения электрической цепи, механической системы, ...
- **Имитационное (динамическое)** моделирование изучает объекты, явления и

процессы, изменяющиеся во времени. Пример: обучение диспетчеров ж/д.

- Модели внешнего подобия **(физические)** - используются для проведения испытания (модели самолетов, кораблей, игрушки, манекены).
- **Имитирующие** модели - используются для изучения поведения объектов в различных ситуациях (тренажеры).
- **Функциональные** модели заменяют реальные объекты при выполнении определенных функций (протезы, искусственные почки, сердце, манипуляторы).
- **Исследовательские** модели заменяют реальные объекты в ходе научных исследований.

Вычислительный эксперимент.

Проверить идеи, гипотезы, теоретические решения, работу новых технических конструкций и аппаратов лучше всего с помощью эксперимента.

Виды экспериментов:

1. **Лабораторный** (используются в физике и химии), необходимо создать экспериментальную установку.
2. **Натурный** (используется в технике), подвергают испытаниям новые аппараты, чтобы определить их свойства и возможности. Лабораторный и натурный эксперименты дороги и требуют много времени (новые установки, измерительное оборудование, материалы).
3. **Вычислительный** (машинный) эксперимент сокращает затраты и ускоряет исследование "новинки", так как вместо экспериментальных образцов и установок используется для исследования математическая модель объекта.(пример: движение тела под действием приложенных к нему сил.)

Математическая модель в вычислительном эксперименте представляет собой совокупность системы уравнений, описывающих изучаемый процесс, алгоритма ее численного решения на ЭВМ и набора программ, при помощи которых исследователь может получать решение задачи.

Любое явление может зависеть от большого количества факторов. Если попытаться учесть все эти факторы, то математическая модель сильно усложнится. Если же попытаться обойтись самыми существенными факторами, то модель будет выдавать приблизительные результаты. Необходимо уметь выделять среди этих факторов главные и второстепенные для того, чтобы пренебречь последними. (Пример: падение камня и падение кленового листа - разное сопротивление воздуха и масса)

Вычислительный эксперимент особенно важен там, где нельзя провести натурный эксперимент (например, моделирование ситуации после ядерной войны).

Уметь графически представить простую классификационную модель.

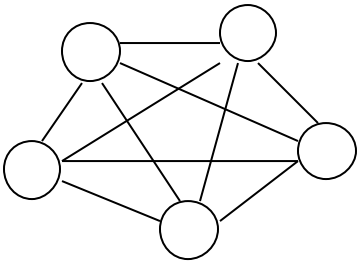
Модели можно описывать с помощью графов. **Граф** - это совокупность точек (кружков, прямоугольников) и линий, соединяющих между собой эти очки. Точки называются **вершинами** графа, линии - **ребрами** графа.

Примеры графов:

1. Схема метрополитена. В данном случае станции метро - вершины графа, а железнодорожные линии – ребра графа.
2. Схема соединений нескольких компьютеров, принтеров, сканеров в вычислительную сеть.
3. Электрическая схема соединений различных устройств (например, елочная гирлянда).

Часто графы используются для *графического решения* задач. Например: сколько матчей будет сыграно в школьном чемпионате по футболу, если в нем принимают участие 5 команд и все команды должны встретиться только по одному разу? Ответ далеко не очевиден, но может быть легко получен если под рукой оказалась бумага и карандаш.

На рисунке вершины графа (кружки) обозначают команды, каждое ребро графа – матч между соответствующими командами. Ответ – 10 матчей.



ГЛАВА 11. ТЕКСТОВЫЕ РЕДАКТОРЫ.

Программные средства для автоматизированной обработки текстов называют ***текстовыми редактором или текстовыми процессорами***. Их существует огромное множество и каждый пользователь выбирает на свой вкус. Наиболее известные из них:

- ***Edit***, встроенный редактор оболочки Norton Comander;
- ***Блокнот и Write*** в Widows 3.1;
- ***Блокнот и WordPad*** в Windows 95;
- редакторы встроенные в ЯВУ, например редактор ***Edit в паскале***;
- ***Лексикон, Слово и дело*** - отечественные текстовые процессоры
- текстовый процессор ***WORD*** из пакета MS Office

Основное назначение текстовых редакторов - набор, хранение, редактирование и распечатка текстов с помощью ПК. Статистика использования ПЭВМ показывает, что более 70% всех применяемых ПК связаны именно с подготовкой текстов. Объясняется это вполне понятными причинами. Во-первых, появление дешевых ПК и принтеров стимулировало использование их в делопроизводстве, журналистике, редакционно-издательском процессе, где подготовка текстовых документов - это основной вид работы. Во-вторых, любая работа связана с подготовкой самых разных текстовых документов: статей, писем, заявлений, технических описаний, отчетов, записных книжек, дневников и т.п.

Использование текстовых редакторов позволяет повысить скорость печатания; резко увеличить эффективность обработки текстовых документов; при этом исчезает страх сделать ошибки, т.к. они легко исправляются. Наиболее очевидные достоинства систем подготовки текстов на ПК по сравнению с обычной пишущей машинкой следующие:

- удобная коррекция текста (удаление/вставка букв, слов, строк; замена слов в целом тексте, в том числе и автоматическая замена) ;
- возможность многократного использования фраз, частей текста, или текста в целом ;
- возможность "выравнивания" (форматирования) текста с автоматическим переносом по правилам грамматики. Обычно используются четыре вида форматирования: "по левому краю", " по правому краю", " по обеим краям одновременно" (как в книгах) и "по центру");
- автоматическое разбиение текста на страницы;
- возможность поиска в тексте фрагмента по маске;
- возможность замены в во всем тексте одного фрагмента на другой .

Существуют некоторые признаки позволяющие отличить того, кто совсем недавно сел за компьютер и ничему еще не научился от хорошего специалиста:

- профессионал нажимает клавишу "Enter" только в конце абзаца, а не в конце каждой строки, как на пишущей машинке. Компьютер сам форматирует (выравнивает) строки, переносит слова не поместившиеся в конце строки на новую строку;
- профессионал почти после всех знаков препинания ставит пробел, поэтому у него слова не “склеиваются”;
- неопытный пользователь с ужасом смотрит на “разрезанный” после случайного нажатия клавиши “Enter” текст и не может соединить “разрезанные” куски (“склеить строки”). Строки соединяются ("склеивается") клавишей "BackSpace" ("Забой"), если установить курсор в начале строки которую мы хотим соединить с предыдущей. Однако в некоторых случаях этот способ не срабатывает по ряду причин. Второй, более надежный способ - установить курсор в конце той строки к которой надо “приклеивать” нижние строки и необходимое число раз нажать клавишу "Delete";
- профессионал знает, что абзац - это часть текста, которая начинается с красной строки и отделяется от остального текста невидимым символом. Этот символ вводится в текст при нажатии клавиши "Enter" (на "советских” ЭВМ это клавиша назы-

валась “ВК” - возврат каретки). Поэтому смысл "склеивания" строки - уничтожение этого символа. В процессоре “Word” есть специальная кнопка, которая для удобства делает этот символ видимым. На этой кнопке изображен значок напоминающий греческую букву “ПИ”;

- профессионал знает, что для абзаца можно установить следующие параметры - отступ для “красной” строки, а также “левый” и “правый” отступы от краев листа;
- профессионал предпочитает работать в режиме “ВСТАВКА”, в котором, если установить курсор внутри слова и нажать любой символ на клавиатуре набранный текст раздвинется. Переключиться из режима “ВСТАВКА” в режим “ЗАМЕНА” и обратно можно с помощью клавиши "Insert";
- профессионал свободно пользуется клавишами “Delete” и “BackSpace” при удалении фрагментов текста;
- профессионал знает, что при работе под управлением Windows “русская” запятая и точка находятся рядом с правой клавишей “Shift”, а руссификатор включается одной из трех комбинаций - “две клавиши Shift, Ctrl+Shift правый/левый или Alt+Shift правый/левый, в зависимости от настройки клавиатуры.
- - параметры страницы: число строк, ширина, высота, ориентация (обычная или альбомная), границы от края (верхний, нижний, левый и правый), может быть межстрочный интервал.

В чем же отличия текстовых процессоров от простых текстовых редакторов? Конечно в первую очередь своей сложностью. Они имеют множество дополнительных возможностей по сравнению с обычными редакторами, но в связи с этим занимают значительно больше места на диске и в ОЗУ. Четкую грань между процессорами и редакторами провести сложно.

Наиболее характерные отличия текстового процессора от редактора:

- возможность проверки лексики;
- возможность построения таблиц и вести не сложные расчеты прямо в тексте;
- возможность брать текст в рамки (обрамление текста);
- встроенный графический редактор;
- наличие "мастеров" (заготовок для разных текстов);
- многооконность и возможность обмена информацией между окнами;
- возможность вставки объектов по технологии OLE;
- наличие библиотеки картинок;
- огромный выбор шрифтов и их размеров, цвета и ориентации;
- наличие спецэффектов (различное расположение и масштабирование текстов);
- возможность помещать фрагменты в "кадр", который существует как отдельный документ. Кадр может перемещаться в любое место, накладываться первым или вторым планом на основной текст или другие кадры;
- возможность использовать "многоколончатую печать", при этом задается необходимое число колонок, как в газете, и текст автоматически в них форматируется
- возможность позади текста располагать "водяные знаки" ("обои");
- возможность создавать "музыкальные" письма со встроенным звуком;;
- возможность создавать типовые письма.
- для использования возможностей текстового процессора можно не "рыться" в меню, а использовать "инструменты" ("кнопки"), расположенные в виде рамки, как в графредакторе "Paint Brush".
- наличие встроенного языка программирования, позволяющего создавать макросы. Макрос - это например, список наиболее часто повторяемых команд, который, может вызываться с помощью новой кнопки на панели инструментов. В процессоре Word язык программирования - Vizual Basic;
- возможность создавать WWW странички в стандарте HTML для установки в Интернет. Созданный в этом стандарте документ представляет собой так называемый ***“гипертекст”.*** Гипертекст - это текст со "ссылками" - выделенными цветом словами или специальными значками. Ссылкой может быть и определенное место на гра-

фическом изображении. Признак ссылки - при установке на нее указателя мышки он изменяет свою форму. Если после этого нажать левую клавишу, то автоматически производится переход на другую часть текста или вызывается встроенное в текст приложение, например зазвучит музыкальный фрагмент или будет демонстрироваться видео ролик. Гипертекст давно используется в справочных системах, например в “Help” в Паскале или Windows. Все WWW-странички и многие мультимедийные проекты созданы в этом формате. Однако надо заметить, что если создавать HTML-документ вручную, например просто в блокноте прописывая “теги”, а не с помощью каких-то специальных редакторов, то его код получается более компактным. Ситуация аналогичная с “обычным” программированием - наиболее “быстрая” и компактная программа пишется на языке низкого уровня - ассемблере, а не на Паскале или Си. Впрочем ведь можно использовать в ЯВУ и ассемблерные вставки.

ИЗДАТЕЛЬСКИЕ СИСТЕМЫ.

С помощью текстовых процессоров вполне можно заниматься издательской деятельностью, но для этого разработаны еще более сложные редакторы, которые называются ***"настольные издательские системы"***. Их самое главное отличие - удобство изготовления макетов газет с их спецификой нерегулярной многоколончатой печати (“спуск полос”) и макетов брошюр. Ведь у брошюры на первом листе располагается первая, вторая, последняя и предпоследняя страница.

Обычно в состав ***издательских систем*** входит и дополнительное аппаратное оборудование:

- "быстрый" и качественный принтер (лучше лазерный);
- сканер;
- множительная аппаратура, например ризограф;
- приспособления для переплета и брошюровки.

Одно из популярных программных средств - "Page Maker" и “MS Puplich”. С их помощью более удобно производить компьютерную верстку - расположение материала на отдельном листе и во всем издании с учетом свободного места, эргономики, эстетики, стиля издания и пр.(особенно на газетном листе).

Основной этап издания любого печатного материала - разработка оригинал-макета. ***Оригинал-макет*** - это высококачественный образец печатного издания, который используется для изготовления с него копий типографским методом, с помощью ризографа или ксерокопировального аппарата. Ризограф - это копировальный аппарат с достаточно низкой стоимостью изготовления одного листа - от 20 коп. и ниже в зависимости от числа копий. Ксерокопирование обходиться примерно в 5 раз дороже. Для типографии оригинал-макет изготавливается обычно на пленке или в виде диапозитивов.

Этапы подготовки текста к изданию:

- набрать текст без его окончательного форматирования;
- проверить ошибки;
- подобрать иллюстрации;
- выбрать формат листа, отступы от краев;
- выбрать шрифты;
- разработать стиль издания или воспользоваться одним из стандартных стилей;
- придумать колонтитулы ("шапки" листов);
- если это необходимо - определить, где и какие будут "буквицы";
- отформатировать текст;
- расположить текст и "картинки" в издании;
- разбить текст на страницы;
- придумать, что делать с "дырками" и с "наездами";
- сдвинуть заголовки в начало листов;
- подвинуть, если надо, иллюстрации;
- снова отформатировать нужные фрагменты текста и т.д и т.д.

ГЛАВА 12. МАШИННАЯ ГРАФИКА. ГРАФИЧЕСКИЕ РЕДАКТОРЫ.

Назначение графических редакторов: - создание и редактирование графических изображений. Например, с их помощью могут быть ***созданы***:

- товарные знаки - логотипы;
- элементы оформления документов;
- поздравительные открытки;
- титульные листы книг и рефератов;
- рекламные материалы;
- ярлыки для обозначения документов и программ в Windows;
- иллюстрации для текстовых документов, электронных таблиц и баз данных;
- таблички для кабинетов и др.

Очень широко графические редакторы используются ***для редактирования*** уже готовых рисунков, которые могут быть отсканированы или выбраны из библиотек рисунков. В этом случае у исходного рисунка можно:

- вырезать фрагмент;
- изменить размер;
- изменить ориентацию;
- изменить цветовую гамму;
- размножить фрагмент рисунка;
- добавить надписи;
- изменить контрастность.

ТИПЫ РЕДАКТОРОВ

Графические редакторы можно классифицировать используя разные критерии.

Критерий №1 - простота редактора.

Простые редакторы, например встроенный в Windows стандартный редактор Paint или PaintBrush, просты в использовании для неопытного пользователя, занимают мало места на диске. Сложные редакторы, например CorelDraw, предназначены скорее для профессионалов - они очень громоздки.

Критерий №2 - формат созданного графического изображения.

Различают векторный и растровый ("точечный" или "битовый") формат изображений. Простейшие редакторы создают точечные изображения - это означает, что в файле хранятся координаты каждой точки и ее цвет. Сканеры, цифровые видеокамеры или фотоаппараты создают кадры в этом формате. Поэтому файлы этого формата занимают на диске очень много места. Второй недостаток точечных форматов изображений - при их трансформации (изменении размеров или вращение) обычно графика сильно искажается. Файлы точечных форматов имеют стандартные расширения .bmp, .pic, .pcx и др.

Изображения векторного формата занимают на диске значительно меньше места за счет того, что "картинка" разбивается на векторы для каждого из которых на диск записываются координаты только двух точек - начала и конца вектора. Файлы в этом формате используются для передачи по Интернет, так как их быстрее можно передать по сети.

Примеры редакторов создающих файлы в векторном формате - CorelDraw (файлы с расширением .jpg и .gif) и встроенный графический редактор текстового процессора Word. Удобство встроенного графредактора Word - очень удобно привязывать вектора к текстовым строкам.

Дополнительная информация для тех, кто хочет заниматься компьютерным дизайном, графикой.

На самом деле векторный принцип описан выше очень упрощенно. На самом деле используются не векторы, а "кривые Безье", названые так по имени французского математика. Он знаменит тем, что впервые рассчитал так называемые кривые - "сплайны" для корпусов автомобилей. На диск при использовании векторного метода записываются математические описания объектов - примитивы. Сплайны - кривые

третьего порядка. Суть сплайна - любую кривую можно построить зная четыре коэффициента, соответствующим четырем точкам на плоскости, следовательно для каждой "маленькой" кривой, на которые разбивается изображение, надо записать координаты не двух, а четырех точек. Две крайних, а две рассчитанные "хитрым" способом. К сожалению фирмы-разработчики эти способы не документируют из понятных соображений и поэтому "картинка" созданная с помощью одного редактора может сильно искажаться другим редактором или просмотрщиком - ее не удается конвертировать в другой формат. Например в CorelDraw есть заливки не реализованные нигде более. Вывод из вышеизложенного - компьютерный дизайнер должен быть еще и хорошим математиком. Отметим, что с помощью сплайнов построены также так называемые векторные шрифты TrueType и PostScript, поэтому очень удобно изменять их размер.

Критерий №3 - создание и редактирование анимационных сюжетов.

Существуют редакторы позволяющие преобразовать видеосигнал, например от видеомагнитофона, в файл, например с расширением .avi Но для этого должен использоваться специальный дорогой видеоадаптер, позволяющий подключить видеомагнитофон.

Нетрудно найти редактор позволяющий вращать объемные буквы или цифры или простые рисунки. Один из первых "Fanta".

Критерий №4 - специализированные редакторы для обработки фотографий, например редактор PhotoShop (фотомагазин).

Эти редакторы имеют также встроенные возможности управлять процессом сканирования.

Критерий №5 - возможность создания и обработки трехмерной графики, например пакет 3D-Studio.

Такие пакеты позволяют работать с объемными изображениями - так называемой 3D-графикой. Быстрая обработка 3D-графики - одна из основных проблем современных компьютеров. Современные игровые программы, некоторые интернет-приложения, дизайнерские пакеты написаны с использованием этой графики. Для ее поддержки в компьютере прежде необходим был видеоаксельратор - весьма дорогое устройство. Новые микропроцессоры берут на себя его функции, однако "под них" еще не написаны соответствующие приложения, хотя все еще впереди.

Сейчас популярно создание с помощью 3D виртуальных городов в InterNet. Посещая WWW-серверы на которых размещены эти города можно знакомится с ними и их достопримечательностями.

Основные отличия графических изображений друг от друга:

- количество цветов и оттенков, приемлемо 256, а бывают миллионы использованных оттенков цвета;
- разрешение - число точек доступных для построения изображения на экране, приемлемо 1024*768, можно и больше;
- степень сжатия картинки при записи на диск;
- наличие анимации;
- используемый графический формат (векторная графика или точечная).

Опытный пользователь может продемонстрировать и объяснить действия ***простейших графических инструментов*** графредактора:

- линии;
- прямоугольника пустого и с заливкой;
- круга и окружности;
- заливки замкнутых областей ("ведро");
- разбрызгивателя;
- карандаша;
- ввод с клавиатуры текста на поле изображения, выбор размера, типа шрифта, цвета и стиля символов;
- "резиновой линии" (очень напоминает вышеупомянутый сплайн);

- резинки простой и цветной;
- замкнутого многоугольника;
- выделение фрагмента изображения в виде прямоугольника и произвольное выделение фрагмента (“ножницы”).

Опытный пользователь знает:

- в чем отличие цвета переднего и заднего плана,
- как изменить палитру и толщину линий,
- как сохранить изображение на диске,
- как приблизить изображение для редактирования мелких деталей.

Он умеет производить с выделенными фрагментами изображения следующие действия:

- “буксировать” их мышью не затирая фона;
- изменять из размер;
- изменять их ориентацию;
- изменять наклон;
- копировать их в буфер обмена (Clipboard);
- возвращать скопированные фрагменты из буфера обмена в исходное изображение или другой документ.

ГЛАВА 13. ЭЛЕКТРОННЫЕ ТАБЛИЦЫ

Электронные таблицы (ЭТ) - пакет программ предназначенный для обработки табличной информации.

Примеры информации, которую можно представить в виде таблицы:

- таблицы логарифмов, умножения, артиллерийских стрельб;
- классный журнал, дневник;
- ведомость на зарплату;
- различные счета на оплате товаров и услуг;
- бухгалтерские книги, отчеты;
- данные экспериментов и пр.

Задачи, которые можно решить с помощью ЭТ:

- числовая обработка табличных данных по формулам записанным в ячейках ЭТ;
- числовое моделирование;
- создание связанных с БАЗАМИ ДАННЫХ отчетов;
- визуализация данных с помощью графиков;
- сортировка данных в ЭТ по определенным критериям;

Основные отличия ЭТ от реляционной (табличной) системы управления базами данных (СУБД):

1) В табличных базах данных, если она представлена виде таблицы, сразу видно что “столбцы”, которые в базах называют “полями” имеют имена, а на озаглавлены буквами латинского алфавита. Строки в базах называются “записями” и в отличие от таблиц строго однородны - это означает, что все записи имеют одинаковые поля и информация записанная них однотипна. Казалось бы, что и в таблице можно также записывать информацию. Это действительно так, но в отличие от баз в таблицах нет эффективных способов обработки такой информации.

2) Так как в базах данных в записях хранятся однотипные данные, можно с помощью МЕНЮ изменять ВИД базы данных, что невозможно в таблицах. Существуют виды: “таблица”, “форма”, “запрос”, “отчет”. Запросы и отчеты - это особые способы обработки информации, которых нет в ЭТ.

3) Информацию записанную в ЭТ нельзя представить в виде формы или запроса, однако отчеты с помощью таблиц все же создают, но только по связанным с ними базам данных. В базах данных отчеты заполняются почти автоматически, а в таблицах

все формулы надо прописывать вручную.

4) ЭТ предназначены для числовой обработки табличных данных, т.е. для расчетов. В базах данных тоже возможны расчеты, но там без использования особых приемов, например отчетов, нельзя в формулах использовать "вертикальные" ссылки, т.е. ссылаться на поля не той записи, в которой мы пишем формулу. Это связано с тем что однозначного номера у записи нет. Поэтому, если в базе мы пишем формулу, она автоматически действует в том поле в котором была записана на все записи базы.

Наиболее популярные ЭТ - Excel из интегрированного пакета MicroSoft Office, SuperCalk, Works и другие.

Основные возможности ЭТ:

- ***вычислительные*** - автоматическое выполнение расчетов при любом изменении ЭТ, большой набор различных функции. Расчеты производятся с помощью формул, которые могут копироваться. В формулах могут использоваться указания на ячейки ЭТ трех видов - относительные, абсолютные и смешенные. При копировании относительные ссылки изменяются методом параллельного переноса, а абсолютные не изменяются. Копирование данных и формул можно производить индивидуальным или групповым способом;
- заполнения ряда ячеек данными с определенным заданным шагом;
- оформительские - представление данных в виде удобном для чтения с использованием различных узоров, стилей форматирования, шрифтов, защиты данных Таким образом удобно оформлять бланки, например счета за квартплату;
- перенос данных и формул. При выполнении этих операций относительные и абсолютные ссылки в формулах не изменяются;
- сортировка данных (убывающая и возрастающая) может производится одновременно по трем столбцам;
- построение красочных графиков и диаграмм;
- поиск и автоматическая замена данных в ЭТ по маске;

Электронные таблицы Excel

Электронная таблица — это прямоугольная матрица, состоящая из ячеек, каждая из которых имеет свой адрес. Адрес ячейки (или **ссылка** на ячейку) определяется обычным координатным способом, например, ячейка B3, ячейка C5 (сначала указывается буква—название столбца а затем—номер строки).

В каждую из ячеек можно записать **число**, **формулу** (арифметическое выражение) или **текст**. Операндами формулы могут быть математические функции, константы, ссылки на ячейки; в последнем случае реальный операнд — содержимое ячейки с указанной ссылкой.

Тексты и числовые константы, занесенные в ЭТ, сами по себе в ходе работы таблицы никогда не изменятся. Что же касается формул, то программа по специальной команде производит вычисление их значений и отображает его на экране (в той же ячейке, куда занесена формула).

Задаваемый обычным образом адрес ячейки называется относительным адресом или **относительной ссылкой** (например C3).

При некоторых операциях копирования, удаления, вставки электронная таблица автоматически изменяет адреса ячеек. Иногда возникает необходимость не менять адрес ячейки. В таких случаях используют абсолютный адрес или **абсолютную ссылку**. Абсолютный адрес ячейки создается с помощью знака доллара $(например, F5, F$5 $F5).

Группа ячеек {диапазон) задается через двоеточие, например, B3:D4 (или B3..D4), и образует прямоугольник, включающий ячейки B3, C3, D3, B4, C4, D4.

Ячейки могут содержать следующие типы данных:

1. **Символьные** (текстовые) данные. Могут включать в себя алфавитные, числовые и специальные символы. По умолчанию символьные данные при вводе выравниваются по правому краю ячейки. При необходимости их можно выровнять по левому краю или по центру.

2. **Числовые** данные. Могут содержать только числовые данные. Исключением являются десятичная точка (или запятая) и знак числа, стоящий перед ним. Числовые данные по умолчанию выравниваются при вводе по левому краю ячейки.

3. **Формулы**. Формула может включать ряд арифметических, логических и других действий, производимых с данными из других ячеек. На экране после ввода формулы отображается результат вычислений, а сама формула отображается в строке ввода над таблицей. Ввод формул начинается со знака равенства.

4. **Функции**. Функции могут самостоятельно присутствовать в ячейках, а также могут входить в состав формул. Различают математические, статистические, логические, финансовые и другие функции.

5. **Даты**. В этом типе данных возможны такие функции, как добавление к дате числа (пересчет даты вперед и назад) или вычитание разности двух дат (длительность периода).

Названия ссылок	Обозначения	Результат при копировании или переносе формул
Частичная абсолютная ссылка	$A5	Не меняется номер столбца
	A$5	Не меняется номер строки
Полная абсолютная ссылка	A5	Не меняется ни номер строки, ни номер столбца

Excel имеет два окна — программное (внешнее) и рабочее (внутреннее). Внутреннее окно содержит рабочую страницу (таких страниц несколько, они образуют книгу), представляющую двумерную прямоугольную таблицу. Справа и внизу на рабочей странице расположены линейки со стрелками прокрутки, позволяющие с помощью мыши быстро перемещаться по странице.

В окне Excel под зоной заголовка находится область меню. Чуть ниже находится основная линейка инструментов. Кнопки линейки инструментов позволяют быстро и легко вызывать различные функции Excel. Их можно вызывать так-же через меню.

Чтобы выполнить какое-либо действие с данными, помещенными в ячейки (ввод, копирование, удаление, форматирование и т.п.), необходимо их выделить. Чтобы выделить ячейку, укажите на нее и нажмите кнопку мыши. При нажатой левой кнопке можно выделить диапазон ячеек. После выделения необходимой области нажмите правую кнопку мыши, вызывая контекстное меню, которое позволяет выполнить ряд команд: *Вырезать, Копировать, Вставить* и т.п.

Изменение данных проводят прямо в ячейке или в строке редактирования (нажать F2). Перемещение или копирование содержимого ячеек можно осуществить перетаскиванием его с помощью мыши. Чтобы скопировать (а не переместить), держите нажатой клавишу CTRL.

Создание формулы начинается с ввода знака равенства (=). Формула содержит встроенные функции, адреса ячеек, константы. В случае затруднений с формированием формулы используйте Мастер функций.

Подобный сервис есть и при оформлении дизайна таблицы. Перед печатью таблиц (кнопка Печать) удобно осуществить предварительный просмотр (кнопка Предварительный просмотр в меню Сохранить).

Excel работает с несколькими листами книги. Например, на одном листе можно разместить итоговые оценки класса за пять лет обучения, а на пяти следующих — данные за каждый год обучения. Листы книги могут служить местом для размещения графических иллюстраций, диаграмм.

Для отображения числовых данных в графической форме используют линии, полосы, столбцы, сектора и другие маркеры, а также их объединенные варианты. Для размещения диаграммы рядом с данными создают внедренную диаграмму на том же листе.

Можно создать диаграмму на отдельном листе диаграммы. Если нет времени, желания и возможности для относительно сложных построений, используют автоматическое оформление диаграмм с помощью команды *Автоформат* в меню *Формат*.

Помимо того, что имеется большая встроенная библиотека построения графических образов (графиков, диаграмм, гистограмм), Excel содержит мощный встроенный графический редактор.

Excel не только «дружен» с текстовыми и графическими системами, но и поддерживает основные действия, характерные для систем управления базами данных (СУБД). Более того, у него развит аппарат импортирования и экспортирования данных из других программных систем.

Для форматирования данных в ячейках электронной таблицы используется меню Формат\Ячейки, для настройки ширины и высоты ячеек — меню Формат\Строка и Формат\Столбец.

ГЛАВА 14. СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Система управления базами данных (СУБД) - программное средство, позволяющее создавать, хранить, редактировать и обрабатывать ***базы данных.***

Базы данных (БД)- совокупность специально организованных данных, которые хранятся на каком-либо материальном носителе.

Данные, хранящиеся в одной базе данных, должны быть связаны между собой по смыслу, логически. Это достигается путем группировки (объединения) их по определенным признакам (параметрам). Такой ***процесс*** получил название ***структурирования данных.*** В результате структурирования данных появляется база данных в компьютерном или некомпьютерном варианте. Это определяется тем, был ли при этом использован компьютер для хранения базы данных. У любой базы данных должно быть имя.

Примеры баз данных:

1) ***"Кадры"*** - имя БД ,
   - логический смысл, связывающий данные - сведения о сотрудниках предприятия;
   - материальный носитель для хранения данных:
2) ***"Книги"*** - имя БД
   - логический смысл, связывающий данные - сведения о книгах в библиотеке;
   - материальный носитель для хранения данных, определяющий вид БД:
3) ***"Учащиеся"*** - имя БД;
   - логический смысл данных - сведения об учащихся гимназии.
   - материальный носитель для хранения данных, определяющий вид БД:

Итак, определение базы данных.

База данных - *поименованная совокупность структурированных данных предметной области.*

В основном все СУБД - ***интерпретаторы.*** Это означает, что без оболочки созданный с их помощью программный продукт (база данных) работать не может. Поэтому пользователю нужно сначала загрузить СУБД, а затем открывать с помощью ее оболочки нужную ему базу данных или создавать новую. Базы данных хранятся на диске отдельно от СУБД в виде отдельных файлов. Эти файлы могут быть использованы и некоторыми другими СУБД, а не только той, с помощью которой они были созданы.

Однако существуют СУБД, например ***FoxPro***, которые могут также работать и в режиме ***компилятора***. При отладке базы такая СУБД работает как интерпретатор, а затем, когда база готова, она компилируется и передается заказчику в виде выполнимого ("exe") файла. Достоинство такого подхода - база становится более быстродействующей, так как при работе не тратится время на ее ***трансляцию***. Недостаток - для пользователя база является закрытой, он не может изменить в ней почти ничего.

Базы данных можно создавать и с помощью языков высокого уровня, например таких как Паскаль, Си, новые версии Basic или Java. Первые три из них компиляторы, а

последний - интерпретатор. С помощью языка Java создают базы для компьютерных сетей.

Несколько баз данных можно объединить в ***банк данных.***

Любая база данных состоит из структурных элементов, называемых ***записями***, которые в свою очередь могут быть разбиты на ***поля.***

Единица хранения информации в БД называется ЗАПИСЬЮ.

Каждая запись соответствует одному из объектов базы данных. Все объекты базы должны быть однотипны, поэтому и записи однотипны. Например, для базы "Книги" записью будут являться данные, описывающие признаки (параметры) конкретной книги. Сколько книг мы внесем в базу - столько в ней будет и записей.

Данные, которые заносятся в запись чаще всего разбивают на части, в соответствии с целями, которые ставит перед разработчиком заказчик. Например, если для базы "Книги" потребуется производить поиск книг по фамилии автора, то понятно, что эти данные надо отделить от названия книги. Такие области, в которых находятся отделенные друг от друга внутри записи данные, называются ***ПОЛЯМИ***.

ПОЛЕ - наименьшая единица поименованных данных внутри записи.

Примеры полей:

- название книги;
- фамилия автора
- жанр книги и т.п.

ЗАПИСЬ - набор логически связанных данных, состоящих из ПОЛЕЙ.

Внутри базы данных записи могут быть связаны между собой самыми разнообразными способами. Выбор способа связи между записями зависит от удобства работы с ними. Способы связи между записями внутри БД называют ***моделями баз данных.***

МОДЕЛЬ БАЗЫ ДАННЫХ - это описание связей между данными в виде логической схемы или таблицы, в которой содержится вся информация о всех связях между записями.

ТРИ ОСНОВНЫЕ МОДЕЛИ БАЗ ДАННЫХ

- ***иерархическая или древовидная***

Как пример БД такой модели можно привести способ организации файловой системы на магнитном диске. С этой точки зрения пакеты программ Norton Commander и DOS это СУБД, управляющие базой данных о файлах на диске. Все сведения о файле составляют запись, а каждый параметры файла (имя, тип, размер, дата создания и др.) - поля;

- ***сетевая***

Такую модель данных приблизительно можно сравнить с ***WWW-страничками*** или любой справочной системой, например по Windows или MS Office. Они представляет собой ***гипертекст***, содержащий перекрестные ссылки. ***Ссылки*** - это связи между записями описывающими понятия справочной системы. В тексте такие ссылки выделяют цветом. Если курсор мыши установить на такую ссылку он изменяет свою форму, а если нажать клавишу мыши компьютер автоматически произведет переход к данным на которые указывает ссылка, то есть на другую запись;

- ***реляционная или табличная***

Эта модель наиболее распространенная, т.к. существует мощный математический аппарат для обработки данных представленных в виде таблицы. Например, простейшая база данных на ЯВУ - это одномерный массив состоящий из данных типа "запись". Недостаток такой базы данных - ограничение размерности массива, то есть числа записей в базе.

Все существующие СУБД - табличного типа. Создание иерархических или сетевых баз требует особого искусства программирования. В дальнейшем мы будем рассматривать только табличные БД.

ОТЛИЧИЯ ТАБЛИЧНЫХ БД ОТ ЭЛЕКТРОННЫХ ТАБЛИЦ

На первый взгляд кажется что отличия ЭТ от БД невелики, однако при работе с достаточно большими базами они очень существенны. Базы данных предназначены для обработки ***очень*** больших объемов информации (иногда миллионы записей). Отсюда и

их особенности.

Если БД выводится на экран в табличном виде, то сразу бросается в глаза:

- столбцы имеют не повторяющиеся имена - это ***поля*** базы данных;
- данные находящиеся в ***записях*** (строках) однородны - в разных записях, но одних и тех же полях данные имеют один и тот же тип;
- если попытаться написать формулу в какой ничуть ячейке табличной формы базы данных и написать знак "=", а затем перемещать курсор по другим ячейкам, то видно, что "все записи на одно лицо" - у них нет номера. Поэтому в формулах, записанных в поле определенной записи базы данных нельзя использовать данные из другой записи. Из-за этого и формула в одном поле может быть только одна, она автоматически начинает действовать для всех записей базы стоит только ее набрать для одного поля. Поэтому не надо копировать формулы для всех записей базы данных. Если для другой записи задать в том же поле другую формулу предыдущая формула исчезнет. По этой же причине, если Вы захотите рассортировать записи по данным какого то поля не надо выделять эти записи цветом. Действительно, представьте, что в Вашей базе миллион записей - тогда сама попытка выделять в ней что-то мышкой является бессмысленной.

В базах есть три основных преимущества, три способа обработки информации, которых нет в ЭТ:

- организация ***запросов*** к базам данных (***фильтрация*** данных);
- автоматическая ***генерация отчетов;***
- создание ***типовых писем***.

Первое преимущество баз данных - возможность организовывать к ним ***запросы***. Эту операцию называют также ***фильтрацией*** данных - вывод на экран не всех записей базы данных, а только их части, той которая удовлетворяет определенным требованиям заданным пользователем. В СУБД Works возможно задать одновременно три критерия по разным полям, по которым производится фильтрация записей. Эти критерии могут быть объединены логическими связками "и" или "или". Например, для базы "книги" можно произвести запрос "все романы написанные, Толстым Л.Н.", произведя фильтрацию по полю "автор"="Толстой Л.Н." и полю "жанр"="роман".

После выполнении запроса к базе данных на экране можно будет увидеть только ту часть информации, те ее записи, которые соответствуют критериям запроса. Остальная часть базы становится не видна, но из ОЗУ она не исчезает и может на экране быть восстановлена.

Второе преимущество баз данных - возможность создания ***отчетов***.

Отчет по БД - специальный вид базы данных, в котором в специальную таблицу заноситься информация из табличной формы базы. В отчет из каждой записи могут переносится информация не из всех полей, а только из тех которые интересуют пользователя. Информация выводится на принтер в табличной форме, причем возможно создание красочного заголовка и именования столбцов.

В отчетах возможно использование формул, которые обрабатывают все записи (такие формулы невозможно задать в табличной форме). Например, можно просуммировать данные по любому полю или вычислить среднее значение.

В отчет можно помещать не все записи. Для этого сначала надо произвести запрос к базе данных.

В Works есть возможность поименовать и хранить на диске по восемь отчетов и запросов.

Для того чтобы увидеть, как будет выглядеть отчет на бумаге надо перейти в режим "предварительный просмотр".

Третье преимущество баз данных - возможность создавать ***типовые письма***.

Типовые письма - один из видов отчета по БД, в котором в текстовый документ из соответствующих полей БД помещается необходимая информация. Для этого надо прописать или вставить в тесте имена соответствующих полей базы данных. При печати на принтере, или при печати "в файл", или при пересылке по электронной почте либо в режиме "предварительный просмотр" в эти места текста будет подставлена информация из соответствующих полей БД. Причем будет подготовлено столько писем, сколько запи-

сей в базе данных. Число писем можно ограничить, если предварительно организовать к базе запрос и отфильтровать не нужные записи.

Кроме этого база данных, работающая под управлением Windows, позволяет создавать с помощью электронных таблиц, текстовых процессоров и других приложений так называемые ***связанные отчеты*** на основе ***OLE-технологии.*** В этих приложениях информация из базы данных будет автоматически изменяться при любом изменении информации в исходной базе. Для организации подобной связи надо в приложении воспользоваться пунктом меню “Поместить”, “Объект” и выбрать поле интересующей Вас базы.

Кроме вышеизложенных особенностей СУБД, они подобно ЭТ, позволяют выполнять с данными следующие операции:

- оформительские - представление данных в виде удобном для чтения с использованием различных узоров, стилей форматирования, шрифтов, защиты данных Таким образом очень удобно оформлять базы в режиме “форма” и “отчет” - в их состав можно включать разнообразные объекты, в том числе графические;
- сортировка данных (убывающая и возрастающая) может производится одновременно по трем полям;
- поиск и автоматическая замена данных в БД по маске.

В БД не имеют смысла понятия “абсолютная” и “относительная” ссылка, так как необходимости копирования формул нет. В СУБД Works нет возможности построения графиков.

ГЛАВА 15. ОРГАНИЗАЦИЯ КОМПЬЮТЕРНЫХ СЕТЕЙ

КОМПЬЮТЕРНАЯ СЕТЬ - объединение двух или более машин посредством каналов передачи данных. Компьютерную сеть называют также “***вычислительная сеть***" или "***телекоммуникационная сеть***".

Каналами связи могут быть как проводные или оптические линий связи, так и линии радиосвязи. Из них самые дешевые - проводные линии связи. К ним относиться:

- коаксиальный кабель;
- бифилярное соединение - витая пара проводов;
- телефонный кабель;
- оптоволоконная линия связи.

По оптоволокну сигнал распространяется в виде световых импульсов, в других линиях связи - в виде электрических сигналов.

Приемущество оптоволокна:

- отсутствие электрических помех;
- высокая скорость передачи информации;
- огромная информационная емкость одного канала.

Например, одновременно разговаривать по телефону по одному оптоволоконному каналу связи могут несколько тысяч абонентов. По оптоволоконному каналу сигнал передается с помощью луча лазера. В Интернет оптоволокно - лучший способ соединения абонентов.

Недостатки передачи сигнала в виде электрических импульсов - влияние на него помех от посторонних электроцепей, низкая скорость передачи информации и небольшая информационная пропускная способность. С точки зрения помех – коаксиальный кабель для передачи таких сигналов предпочтительней. Для локальных сетей для повышения их быстродействия лучше использовать витую пару, но для этого надо приобрести специальное устройство - ***маршрутизатор***.

Связь между компьютерами с помощью радиосвязи повышает их мобильность, используется для ЭВМ установленных на автомобилях, самолетах, спутниках или для переносных моделей (notebook). Ее стоимость сравнима с сотовой телефонной связью.

ПРИНЦИПЫ ОРГАНИЗАЦИИ КОМПЬЮТЕРНЫХ СЕТЕЙ

Все компьютерные сети можно разделить по признаку расстояния, на которой передается информация:

- ***локальные вычислительные сети (ЛВС) -*** объединение компьютеров, расположенных на небольших расстояниях друг от друга. Их длина как правило не превышает 2 км, по ним информация передается внутри одной организации и не предназначена для широкого (открытого) распространения для большого числа пользователей. Пример - сеть в учебном компьютерном классе или в банке;
- ***региональные сети -*** объединение компьютеров и локальных сетей для решения проблемы регионального масштаба. Пример - межвузовская сеть города;
- ***корпоративные сети -*** объединение компьютеров и сетей в пределах одной корпорации сетей Пример - сеть российских паспортных столов;
- ***глобальные сети -*** такие сети, которые обычно охватывают город, регион, страну или весь мир и предназначаются для достаточно большого числа пользователей.

Локальные сети могут быть связаны с другими сетями. При этом могут возникнуть большие проблемы с обеспечением сохранности, поэтому системным администраторам сети приходиться прикладывать значительные усилия для их защиты от несанкционированного доступа.

Что необходимо для создания компьютерной сети?

Если это сеть локальная, то сигналы между компьютерами обычно передаются по коаксиальному кабелю или бифилярной паре. На каждом компьютере (они называются "***сетевыми станциями***") должны быть установлены специальные дополнительные платы - ***сетевые адаптеры***. Если на станциях стоит Windows 95 и выше, можно обойтись без дополнительного программного обеспечения.

Существует несколько вариантов ***топологических решений*** при организации локальной сети:

- ***звездообразная сеть,*** сервер в центре, пользовательские ЭВМ подключены "звездой". Недостаток такого решения - требуется провести много проводов, на сервере нужен особый адаптер. Преимущество - если одна машина выходит из строя, то она всю сеть не портит
- ***кольцевая сеть***, адаптеры у всех машин однотипные, проводов много меньше, но если сеть зависает в сети не работают все машины;
- ***магистральная сеть***, построена по такому же принципу что и кольцевая, только кольцо не замкнуто, а на обоих концах сети установлены специальные заглушки.

Если Вы хотите подсоединиться к ***глобальной компьютерной сети*** необходимо:

- подсоединить Ваш компьютер к телефонной сети через ***модем***. Модем - устройство для согласования между собой сигналов компьютера и телефонной сети. Модемы используются для организации обмена информацией между двумя компьютерами по телефонной сети. Модемы производят ***модуляцию*** (преобразование) дискретного двоичного компьютерного сигнала в непрерывный аналоговый сигнал для того, чтобы его можно было передать по телефонной сети. Поэтому первые две буквы в этом слове - "***мо***". Следующие три буквы соответствуют слову "***демодуляция***" - для модема это процесс преобразования сигнала из телефонной линии, посланного с другого компьютера, в дискретный цифровой сигнал. Демодуляция - процесс обратный модуляции сигнала.;
- знать по каким номерам телефона можно соединиться с другим компьютерами. Существует множество глобальных сетей, мы рассмотрим их специфику ниже.

Примеры организации компьютерных сетей.

1) ***Простейшая локальная сеть*** - организуется через параллельные порты (длинные разъемы на материнской плате). С помощью витой пары проводов между собой соединяются два компьютера для передачи информации. Как программное средство для обеспечения связи можно использовать оболочку Norton Commander, в которой ре-

жим связи включается с помощью пункта меню "linc" (“линкование”). По такой небольшой локальной сети удобно, например переписывать информацию на вновь отформатированный винчестер с исправного компьютера. Недостаток такой сети - между собой соединяются только два компьютера, причем один из них является пассивным - его винчестер мы видим на одной из панелей Norton Commander другой машины.

2) ***Организация решения всех задач пользователей ЛВС на одной мощной, центральной машине.***

Центральная высокопроизводительная машина называется ***сервером локальной сети.***

Конечное устройство, с которым соединен сервер называется ***ТЕРМИНАЛОМ*** - это более простая персональная машина, общий лазерный принтер, сканер или другое периферийное оборудование, которое служит для ввода и отображения данных. Простейшие машины при этом называют ***дисплейными станциями***, а зал где они установлены "дисплейным". В этом случае центральная машина работает в режиме "***разделения времени***", т.е. по очереди выполняет программы введенные и запущенные с каждого терминала. При этом пользователь, работающий на своем терминале, часто не замечает задержек времени в своей работе. Ему кажется, что он работает на персональной машине в режиме реального времени. Преимущество таких сетей - хотя центральная машина "дорогая", зато терминалы "дешевые", поэтому общая стоимость оборудования низкая. Машинное время у центрального компьютера используется эффективно, потому что пользователи терминалов большую часть времени только вводят данные или их "осознают". Когда компьютеры были еще громоздкими и очень дорогими, так организовывались ***вычислительные центры*** (ВЦ). Сотрудники предприятия или учебного заведения ходили работать на такой центр в "дисплейные залы". В лучшем случае дисплейная станция была на рабочем месте в отделе.

Недостатки этого способа работы:

- пользователи владеют не всеми ресурсами ЭВМ;
- сбой центральной ЭВМ или зависание сети нарушает работу всех пользователей.

Когда появились дешевые мощные персональные компьютеры, необходимость в перекладывании на центральную машину вычислительных функций отпала, однако использование серверов и ЛВС наоборот расширилось по следующим причинам:

- передача данных в виде файлов по сети;
- резервное копирование данных (архивирование) на специальном оборудовании, например магнитных барабанах;
- совместное использование дорогостоящего периферийного оборудования (лазерных принтеров, плоттеров, сканеров, факсов и т.п);
- организация выхода из ЛВС в глобальные сети;
- объединение вычислительных возможностей компьютеров (см. ниже).

3) Решение одной вычислительной задачи с помощью нескольких ЭВМ.

Теперь, когда все машины в ЛВС примерно равны по мощности, стало возможным распараллеливание решения одной сложной вычислительной задачи на группу машин подсоединенных к сети. При этом ее выполнение ускоряется во много раз. Быстродействия или памяти одной машины для решения таких задач часто не хватает. Пример - расчет прочности сложной конструкции, например аэробуса.

Также работа множества сетевых компьютеров под управлением одной и той же программы очень часто бывает необходимо для обеспечения одновременного доступа к ней большого числа пользователей. Классический пример - система продажи авиационных или железнодорожных билетов. Для решения таких задач можно использовать и объединенные ресурсы Интернет. Сейчас очень распространены сетевые базы данных - эти базы предназначены для всех пользователей сети, причем любое изменение в базе сразу становится доступным всем абонентам. Доступ к такой базе для ее редактирования предоставляется администратором базы.

4) связь между ЭВМ с помощью телефонной сети в терминальном режиме.

Если у Вас есть модем и Вы хотите обменяться информацией со своим товарищем, у которого тоже есть компьютер с модемом, то Вы можете с помощью терминальной программы, входящей в стандартную поставку Windows, установить с ним связь. С ее помощью Вы можете обмениваться текстовыми сообщениями, пересылать друг другу файлы, играть по сети в сетевую игру.

5) связь между Вашим компьютером и узлом BBS.

На любой машине имеющей модем можно установить специальную программу и превратить ее в узел BBS. Расшифровка аббревиатуры “BBS” - “Bulleten Board System” - "электронные доски объявлений” - это телекоммуникационная программа, дающая удаленным пользователям регулируемый доступ к ресурсам Вашей ЭВМ. Для этого придется запустить эту программу и не выключать Ваш компьютер целые сутки или на какое то заранее оговоренное время. В этот период любой желающий может в терминальном режиме через модем позвонить по Вашему телефону и получить доступ к ресурсам Вашей ЭВМ (BBS). На экране компьютера обратившегося к BBS появляется список услуг, которые предоставляется данной станцией и условия их предоставления. Можно ограничить доступ к ним за плату по паролю. Некоторые BBS предоставляют информацию имеющую коммерческий интерес, занимаются переводами с иностранных языков, рассылают компьютерные версии периодических изданий (газет или журналов), осуществляют консультирование по компьютерным технологиям, сертификации, организуют клубы знакомств или предоставляют другие сетевые информационные услуги. Список российских и зарубежных BBS распространяется по глобальным сетям, среди них встречается достаточно много некоммерческих бесплатных станций организованных энтузиастами. Многие BBS организовывались радиостанциями, телекомпаниями или газетами в рекламных целях для связи со своими читателями или слушателями. Станции BBS стали появляться сразу же с появлением в продаже модемов.

6) объединение компьютеров в глобальных компьютерных сети

Следующим шагом стало объединение машин, предоставляющих информационные услуги, между собой по специальным соглашениям и протоколам связи. Такие объединения получили название глобальных компьютерных сетей. В мире множество таких сетей, они возникали исходя из повседневных задач, которые ставили перед собой их организаторы. Примером таких глобальных сетей могут служить ФидоНет и ИнтерНет.

ГЛАВА 16. ГЛОБАЛЬНАЯ КОМПЬЮТЕРНАЯ СЕТЬ ИНТЕРНЕТ

Фирмы, желающие сохранить свою информацию в тайне создавали ***корпоративные сети***, а те кто стремился к максимальной открытости и доступности создали ***Фидонет***, сеть любителей. Наиболее известная всемирная сеть ***Интернет*** первоначально разрабатывалась в США, как сеть Пентагона и преследовала цель сделать работу максимально надежной. Дословный перевод этого словосочетания - “сеть сетей” или “меж сеть”.

Интернет - система, объектами которой являются компьютерные сети. Сейчас в этой сети объединено более чем 35 миллионов компьютеров, более чем в 135 странах. У этой сети нет президента, директора или главного редактора - в этом ее уникальность. Она работает благодаря наличию определенных стандартов взаимодействия компьютеров. Если по Fido информация распространяется в основном через компьютеры установленные дома у “фидошников“, то в корпоративных сетях и Интернет существуют очень мощные “главные” компьютеры (***хост-компьютеры или серверы***), соединенные между собой оптоэлектронными кабелями или с помощью спутниковой связи. Эти компьютеры должны обладают большим числом телефонных входов - номеров. Их число называют ***модемным пулом***. Чем он больше, тем проще связаться с хост-ЭВМ. Кроме номеров телефона у каждого хоста в сети имеется свое уникальное имя. Хост-компьютеры отличаются друг от друга ***пропускной способностью*** - объемом информации, который может обрабатываться ими в единицу времени. Если пропускная способность низкая - информация в часы максимальной загрузки хоста может поступать

на ваш компьютер более медленно.

Все глобальные сети можно разделить на:

- коммерческие, например Интернет;
- некоммерческие, например Фидонет.

За предоставление услуг связи в коммерческих сетях надо платить, зато качество, надежность, пропускная способность, число информационных услуг в них во много раз выше, чем в Фидонет. Несмотря на это, многие называют Фидо - "Интернет без графики". В Фидо работают в основном в текстовом режиме. Требования к рабочим характеристикам компьютера для работы в Фидо весьма низкие. Специфика Фидонет - объединение близких по духу людей ("фидошников") на некоммерческой основе. Фидо - это скорее способ жизни, некая современная субкультура для "продвинутых" пользователей ЭВМ.

Интернет в современном обществе занимает в сознании людей все больше и больше места. Работа в Интернет, по сравнению с Фидонет не требует углубленного знания ЭВМ и специфического сленга, а рассчитана на рядового пользователя ("домохозяйку"). Надежность работы Интернет намного выше, шире круг предоставляемых информационных услуг, но соответственно намного жестче требования к мощности вашего компьютера, если Вы желаете работать не только с почтой. В Интернет обычно работают в графическом режиме.

Интернет - это всемирная (глобальная) коммерческая компьютерная сеть.

Многие другие глобальные мировые компьютерные сети входят в ее состав, например сеть российской академии наук, банковские сети и пр. Вычислительные мощности подобных сетей являются основой работы Интернет. Услуги предоставляемые Интернет практически безграничны, причем ежедневно появляются все новые и новые. Это целый воображаемый (виртуальный) мир в котором очень просто можно заблудиться, но без которого в развитом обществе человек уже не сможет прожить. Часто сеть Интернет сравнивают с ***информационной паутиной*** каналов связи, которые оплели весь мир. Революционная привлекательность Интернет - возможность мгновенного доступа к практически любой информации в любом уголке мира для любого человека. По существу, ***свободный доступ к информации*** - одно из самых необходимое условий, которое делает человека по настоящему ***свободным -*** оно гарантирует человеку ***право свободного выбора.*** Вместе с другими основными демократическими гарантиями, это право способствует построению в нашей стране и во всем мире ***открытого общества***.

За все услуги предоставляемые Интернет надо платить. Для того чтобы начать работать в Интернет, кроме необходимого оборудования и программного обеспечения, Вам надо заключить договор на обслуживание с тем, кто уже имеет доступ в сеть. Фирма предоставляющая такие услуги называется "***провайдером***". У хорошего провайдера - мощный хост-компьютер, часто не один, с большим модемным пулом. За рубежом государство и коммерческие организации оказывают большую материальную поддержку развитию Интернет в учебных заведениях. Часть этой помощи поступает и в российские Вузы. Например, в государственном Санкт-Петербургском Университете в Петродворце есть мощный Интернет-узел, через который есть выход в школьную информационную сеть (ШИС). Частные лица и коммерческие организации за доступ в Интернет должны платить. Условия договора могут быть различными. Существуют два основных вида работы в Интернет - "***режим почтового ящика***" и "***режим непосредственного доступа***". Рассмотрим их особенности.

1. Режим почтового ящика (e-mail)

- автономный режим работы, в котором компьютер пользователя Интернет связывается с хост-компьютером провайдера только для передачи и приема писем. Это самый дешевый режим работы, часто даже бесплатный для пользователя. В этом режиме можно работать даже не под управлением Windows, например популярная программа-почтовик "Uupc" и просмотрщик писем "Dmail" работают под ДОС и не требовательны к характеристикам компьютера.

Этот режим работы не диалоговый, не активный, поэтому, конечно, не очень интересный. Письмо можно написать в любом текстовом редакторе на любой ЭВМ, их

часто пишут в коде ASC II - тогда получателю не надо их расшифровывать. Затем с помощью программы – "почтовика", настроенной на хост-ЭВМ Вашего провайдера, через модем письмо передается по телефонной линии. Хост-ЭВМ провайдера автоматически просматривает новую почту и передает письмо по указанному адресу. Очень быстро письмо дойдет в любую точку мира. Таким образом можно послать не только письмо в ASC II коде, но и файл в машинных кодах ("exe" файл) или заархивированный пакет программ, либо графический файл или файл другого формата. Для этого программа – просмотрщик писем должна перекодировать такой файл необходимым образом и присоединить к письму. К сожалению, нередко приходится сталкиваться с ситуацией, когда письмо содержащие такой файл не удается передать или принять по сети. Это происходит из-за того, что часто такие файлы получаются довольно большого объема - мегабайты. Российские телефонные сети очень низкого качества. Даже хорошие, "быстрые" модемы не могут в таких сетях работать с большой скоростью. Также, из-за ошибок в передаче информации, такое большое письмо модему приходится передавать сначала все снова и снова. В результате на это может уйти половина рабочего дня или вообще письмо не удастся передать или принять.

Почтовые адреса в Интернет.

В почтовом адресе Интернет обязательно есть знак "@" - коммерческое "эт". Часто этот знак называют просто "собачка". В Internet каждый абонент имеет свой адрес с уникальным именем, его называют "***e-mail***", почтовый ящик ("mail"-ящик). Адрес почтового ящика присваивается администратор Интернет узла. Адрес нашей гимназии ***dvg@sc446.nit.spb.su***

Состав почтового адреса

Почтовый адрес состоит из двух частей - правой и левой, которые разделены знаком "@".

Правая часть почтового адреса - адрес машины, куда будут направляются письма. Он состоит из кратких названий зарегистрированных телекоммуникационных узлов связи. Адрес читается справа налево. Для нашего примера:

- su - Советский Союз. Это имя хост-компьютера, который связан мощным оптоволоконным кабелем с зарубежными партнерами;
- nit - образовательная сеть, хост-компьютер в школьной информационной сети;
- sc446 - этот домен - наш компьютер, на который поступают наши письма.

Все части адреса разделяются точками, в других адресах их может быть больше или меньше. Пробелов в адресе нет.

Левая часть почтового адреса - название индивидуального ящика на машине, которая принимает письма через модем. На одной машине может быть много адресатов, то есть почтовых адресов. Эти адреса будут различаться друг от друга только левой частью. Для провайдера их число не имеет никакого значения. Для каждого адресата на компьютере организовываются отдельные одноименные каталоги, в которые будут приходить личные письма абонентов Интернет. Возможна организации рассылки этих писем по локальной сети.

2. Режим непосредственного доступа в Интернет (режим "on-line")

В режиме непрсредственного доступа пользователь получает максимальную возможность воспользоваться информационным ресурсам Интернет. Это основной режим работы в Интернет. В этом режиме Пользователь работает в интерактивном, диалоговом режиме с хост-компьютерами сети, сам находит и выбирает источники необходимой ему информации и информационные услуги. В этом режиме можно пользоваться и почтовыми услугами. Удобнее всего обратиться на один из бесплатных почтовых серверов, которые предоставляют почтовые ящики с Web-интерфейсом.

Примеры информационных услуг, предоставляемых Интернет в режиме on-line.

- Web - страницы;

- телеконференции между абонентами сети;
- поисковые системы;
- проведение денежных расчетов с помощью магнитных карточек или паролей;
- осуществление банковских операций;
- курс валют;
- биржевые новости;
- продажа товаров по сети с доставкой на дом (виртуальные магазины);
- рекламная деятельность;
- дистанционное обучение;
- заочная сертификация знаний;
- трудоустройство с помощью сети;
- телефонные переговоры;
- интерактивное сетевое телевидение;
- трансляция по сети аудио или видео концертов;
- сетевые видеотеки и библиотеки;
- варианты компьютерных изданий известных периодических газет и журналов;
- консультации по сети, например медицинские или юридические;
- заказ и бронирование билетов на самолеты, поезда и корабли;
- бронирование номеров в гостиницах;
- астрологические прогнозы;
- кулинарные рецепты;
- прогноз погоды в любом уголке мира;
- службы знакомств;
- полезные советы;
- телефонные справочники;
- мониторинг экологической обстановки в различных регионах мира;
- различные Интернет – проекты, например “Анти-спид”;
- сетевые компьютерные игры.

Этот перечень можно продолжать очень долго. Надо заметить, что многие из перечисленных информационных услуг осуществляются в виде специализированных WWW-страничек (web-интерфейс), могут быть реализованы с помощью телеконференций или поисковыми системами, поэтому остановимся на этих услугах более подробно.

ГЛАВА 17. WEB – СТРАНИЦЫ. ГИПЕРТЕКСТ

Еще до возникновения компьютерных сетей, более 25 лет назад появился новый вид организации данных - ***гипертекст***. С его помощью до сих пор строится справочная система большинства программных продуктов, например HELP языка Турбо-Паскаль. Представьте себе расположенный на экране ЭВМ текст, в котором Ваше внимание привлекают особенные объекты:

- слова выделенные цветом или подчеркиванием;
- графические элементы.

Если установить указатель мыши на такой объект, указатель изменит свою форму. Если затем щелкнуть левой клавишей мыши, то текст находящийся на экране исчезнет и появиться новая заинтересовавшая Вас информация.

Гипертекст - документ, содержащий ссылки на другие документы.

Гипперссылка - выделенный объект, связанный с другим документом и реагирующий на щелчок мыши.

С появлением глобальных сетей метод организации информации с помощью гипертекста стал основным. С помощью гиперссылок можно мгновенно вызывать на экран своего компьютера информацию, не только из других частей просматриваемого документа, но и с серверов сети расположенных в любых других уголках планеты. Таким обра-

зом, каждый объект расположенный в сети соединяется с множеством связей - нитей с другими объектами. Возникает ассоциация с паутиной, которая соединяя информацию находящуюся на разных серверах, опутывает весь мир. Паутина по-английски “Web” (произносится “веб”). Так в 1991 году появилась всемирная паутина - Word Wide Web, сокращенно WWW. WWW связывает миллионы гипертекстовых документов, которые в данном случае называются по другому - ***Web-документы*** или ***Web-страницы (***иногда ***WWW-страницы).*** Они, в отличии от обычных текстовых документов, кроме текста, могут содержать графические, анимационные и музыкальные данные.

Как разместить страничку в сети? Самый лучший способ - создать в Интернет свой сервер, тогда Вы сами будете на нем хозяином и сможете располагать там даже чужие странички на выгодных для Вас условиях. Таким образом поступают крупные фирмы, всемирно известные музеи, крупные магазины, университеты и научные институты. Они могут приобрести для этой цели мощный компьютер и подключить его к оптоволоконной линии связи или покупать дополнительные телефонные линии (номера). Только тогда клиенты, которые обратятся к их серверу, смогут до них дозвониться. Если линий связи мало, то лучше установить страничку у своего провайдера.

Также надо установить на этом компьютере специальную ***программу-сервер***, которая будет предоставлять пользователям информационные услуги. Эта программа называется ***Web-сервер***. Именно те компьютеры на которых она установлена и образуют Всемирную Информационную Паутину, WWW. Их называют ***Web-узлами***.

Большинство пользователей размещают свои странички на чужих серверах на коммерческой основе, хотя есть достаточно много мест в сети, где это можно сделать бесплатно. Но так как “бесплатным бывает только сыр в мышеловке”, обычно предоставляя место для бесплатной странички, провайдер оставляет за собой право располагать в ней свою рекламу.

Какие цели преследуют создатели страничек?

Все странички в сети можно разделить на коммерческие и бесплатные.

Попадая на коммерческую страничку, надо быть готовым к тому, что Вам может прийти дополнительный счет на оплату информационных услуг, например за новую версию компьютерной игры или юридическую справку. Но большинство страниц в сети - бесплатные для посещения. Почему?

В современном мире Web-страница - визитная карточка организации, современный быстрый способ общения со своими партнерами:

- весь мир получает информацию о Вас;
- Вы становитесь интегрированы во всемирное информационное пространство;
- каждый может оставить Вам на гостевой странице свое сообщение;
- Вы можете узнать, сколько человек посетило Вашу страницу за день;
- можно оставить сообщение на гостевой странице.

По каким другим причинам создают Web-страницы?

- многочисленные гуманитарные и образовательные программы, например ”Музеи мира”. Создаются за счет государственных бюджетов, бюджетов образовательных учреждений, ЮНЕСКО, средств спонсоров;
- политическая борьба. Финансируются за счет партий и госбюджетов;
- прямая или скрытая реклама товаров, услуг, фирм и т.п;
- предоставление сетевых услуг рассмотренных выше. При посещении этих страниц счет будет прислан только после заказа или предоставления услуги;
- самоутверждение их создателей, желание заявить о себе всему миру.

Программа – браузер

Web-страницы хранятся в файлах. Создаются эти файлы одними программами, а просматриваются чаще всего другими. ***Просматривать*** - по-английски ***to browser***, соответствующее существительное ***browser*** (по-русски произносится ”***браузер***”).

Браузер - программа, служащая для просмотра Web-документа, т.е. обеспечивающая переход на другой объект в соответствии с гиперссылкой.

Не все браузеры поддерживают все возможности Web-страниц. Например простейшие браузеры не поддерживают графику. Наиболее известные и конкурирующие между собой браузеры:

- ***Mozilla FireFox;***
- ***Google Chrome;***
- ***Opera;***
- ***Microsoft Intetnet Exporer***.

Как просмотреть в сети нужную Web-страницу?

После загрузки браузера в окне "адрес" ввести местонахождение документа в сети - ***Uniform Resoform Location***, сокращенно ***URL***. URL - это комбинация названия протокола передачи документа, адреса сервера, каталога и имени файла.

URL - адрес документа в сети.

Если адрес станицы не известен, можно обратиться на один из поисковых серверов и искать нужную страницу в сети по ключевым словам интересующей Вас темы.

В Интернет принята числовая система адресации, подобный почтовому индексу. Адрес разделен на четыре блока, например 194.135.81.440. Первое число слева означает к какой сети из входящих в Интернет относится этот адрес.

Но людям удобнее работать с именами, их легче запоминать. Поэтому в Интернете была введена также удобная система имен. Она разделяет компьютеры по тематическим или географическим областям. Область - по-английски ***domain*** - произносится "домен". Поэтому такая система называется ***доменной***. Имя в ней читается справа налево. Первым стоит домен верхнего уровня, затем их уровни убывают. Рассмотрим, например адрес: ***sc446.nit.spb.su***

- домен высшего "su" означает, что компьютер с этим именем находится в бывшем Советском Союзе;
- домен "spb" означает - в Санкт-Петербурге;
- третий домен "nit" общеобразовательная сеть города;
- последний домен "sc446" - реальный компьютер установленный в гимназии 446.

Домен каждого уровня отвечает за назначение имен компьютерам находящимся в его ведении. Например, домен "sc446" сам назначает и обеспечивает доступность Интернети именам своих пользователям, например компьютерам в локальной сети или просто пользователям из разных каталогов. Адреса ***dvg@sc446.nit.spb.su*** и ***asm@sc446.nit.spb.su*** назначены администратором сети гимназии, их может быть сколько угодно. Если в адресе присутствует значек "@", значит это почтовый адрес, иначе этот адрес надо использовать в режиме "on-line".

На основании доменного принципа присваивания имен была создана база данных ***DNS***, с помощью которой по имени компьютера определяют его местонахождение в сети. Аббревиатура DNS читается "***дэ-эн-эс***", без перевода на русский язык (***Domain Name System***).

Создание Web-документа.

Web-документы создаются с помощью ***языка HTML*** (произносится "аш-тэ-эм-эль" или "эй-ти-эм-эль") - это аббревиатура английских слов HyperText Markup Language.

Язык HTML - язык разметки гипертекста.

Этот язык был создан для того чтобы создавая документ не надо было думать о различии в программном и аппаратном обеспечении компьютеров, на которых этот документ будет просматриваться. Именно создание таких независимых файлов особенно актуально в сети, потому что там объединяются принципиально разные компьютерные системы - такие как Macintosh, IBM и пр.

Web-документ проще можно создавать с помощью любого текстового редактора, который способен создавать файлы в ASC II коде. Отличия от простого текстового файла будут заключаться в том, что в текст документа будут вставлены, так называемые,

“***флаги разметки***” или “***теги***”, от английского слова “***tag***”. Их можно назвать также командами или кодами HTML.

Тег - инструкция браузеру, указывающая способ отображения текста.

Тег отделяется от остального текста знаками меньше (“<“) и больше(“>“), например, после интерпретации тега
 браузер отобразит горизонтальную линию, идущую через весь экран.

Существует множество редакторов, упрощающих создание HTML документов. Если воспользоваться ими, то не понадобится вставлять теги вручную, однако полученный код не будет оптимален, поэтому профессиональные Web-мастера их не любят.

Сейчас разработан язык высокого уровня ***Java***, который позволяет в HTML формате создавать не только простые странички, но и сложные пользовательские программы, например ***распределенные сетевые базы данных***, файл – клиенты или файл-серверы. Преимущество его использования - аппаратная и программная независимость полученного программного кода. Этот язык принципиально - интерпретатор, так как один и тот же откомпилированный машинный код не может управлять в сети компьютерами изготовленным на разных аппаратных платформах.

Телеконференции.

Телеконференциями называют обсуждения или коллективные дискуссии на разные темы, проводимые с помощью средств связи. В основном общение между участниками конференций происходит по электронной почте. Однако в режиме on-line возможно проведение конференций и в режиме реального времени с использованием видео изображений участников и аудио сигнала, однако для большинства пользователей сети это пока не доступно. Поэтому, даже в режиме on-line при непосредственном общении на, так называемом, “чате” (специализированном сервере телеконференций) участники обмениваются текстовыми сообщениями.

Информация на серверах телеконференций хранится в архивах, организация которых носит иерархический (древовидный) характер, что отражается в названии телеконференции. Они состоят из разделенных точками частей, называемых доменами, в которых отображаются темы конференций. Например:

- relkom.music;
- relkom.music.rock;
- relkom.music.beatles.

Работая сервером телеконференций, можно заказать услугу, которая называется ***“список рассылки”.*** Этот список включает в себя список пользователей которые подписаны на эту конференцию. Так можно найти людей интересующихся теми же вопросами.

Кроме списка рассылки можно работать с ***группами новостей***. Это описания и названия статей, которые можно заказать себе. Любой пользователь может послать свою статью в телеконференцию. Конференции бывают управляемыми и открытыми. В управляемых конференциях специальный человек, “***модератор***”, решает поместить или нет ту или иную статью в конференцию. Для работы с телеконференциями можно использовать обычный почтовик или, например, программу ***Internet News***.

Глава 18. Компьютерные вирусы. Методы распространения и профилактика заражения.

Определение

Компьютерный вирус – это программа, которая использует слабые места дисковой операционной системы для выполнения действий, направленных на бесконтрольное размножение своего “тела” (т.е. программного кода вируса), подрыв работоспособности компьютера и порчу информации, хранящейся на данном компьютере.

С другой стороны, компьютерный вирус - это небольшая программа, которая припи-

сывает себя в конец исполняемых файлов, драйверов, или “поселяющаяся” в загрузочном (BOOT) секторе магнитного диска.

При запуске зараженных программ и драйверов вначале происходит выполнение программы – вируса, а уже потом управление передается самой программе. Если же вирус “поселился” в загрузочном секторе магнитного диска, то его активизация происходит в момент загрузки операционной системы с такого диска.

В тот момент, когда управление принадлежит вирусу, обычно выполняются неприятные для пользователя, но необходимые для продолжения жизни данного вируса действия:

- ❖нахождение и заражение других программ;
- ❖порча данных на магнитном диске;
- ❖шифрование таблицы размещения файлов (FAT);
- ❖проникновение в уязвимые области памяти BIOS и их порча (вирус “Чернобыль”) и т.д.

После окончания работы вируса управление передается зараженной программе, которая продолжает работать “как ни в чем не бывало”. Поэтому часто очень трудно вовремя обнаружить присутствие вируса в системе без использования специальных средств (антивирусных программ). Когда же вирус проявит себя – скорее всего большинство программ уже будет заражено и потери от его зловредных действий будут уже очень велики!

Помимо этого вирус может остаться в памяти резидентно (т.е. постоянно) и продолжать вредить до перезагрузки компьютера.

Внешние признаки заражения компьютера.

Наиболее часто вирусы проявляют себя следующим образом:

- ✓сильное замедление работы компьютера;
- ✓появление на экране в самый неподходящий момент различных фраз или рисунков;
- ✓появление различных видеоэффектов (например, перевертывание экрана);
- ✓пропадание информации с экрана (“опадающие” буквы, “поедание” букв гусеницей)
- ✓генерация различных звуков (например, играет гимн Советского Союза);
- ✓на дисках “исчезает” свободное пространство;
- ✓некоторые программы перестают работать, а другие ведут себя очень странно;
- ✓на дисках вдруг появляется большое количество испорченных файлов данных, текстовых файлов;
- ✓вдруг разом рушится вся файловая система на одном из дисков;
- ✓неожиданно операционная система перестает видеть винчестер;
- ✓произвольно изменяется длина отдельных файлов.

Типы вирусов

Вирусы отличаются друг от друга по способу проникновения в систему, по способу размножения и функционирования.

1.Вирусы-спутники. Это примитивные вирусы, которые создают для каждого файла с расширением .exe файл с тем же именем, но с расширением .com. При запуске файла ДОС сначала ищет .com файлы, а потом уже .exe файлы, поэтому управление вначале получает вирус.

2.Файловые вирусы. Поражают файлы с расширениями .com, .exe, .sys. Эти вирусы дописывают свое тело в начало, середину или конец файла и изменяют программный код таким образом, чтобы первыми получить управление. Чаще всего зараженный файл непоправимо портится вирусом.

3.Загрузочные вирусы – поражают загрузочные сектора дисков. Инфицирование новых дисков происходит, когда в зараженный компьютер вставляют новую дискету и начинают с ней работать. Такие вирусы могут поражать как файлы, так и загрузочные секторы, а после запуска продолжают оставаться в памяти резидентно.

4.Вирусы DIR. Изменяют файловую систему диска: в таблице размещения фай-

лов (FAT) заменяют ссылки на все исполняемые файлы ссылками на тело вируса. При запуске любой программы управление получает вирус, который восстанавливает правильные ссылки на исполняемый файл, а сам остается в памяти резидентно, получая возможность заражать другие диски. Когда зараженная дискета попадает на чистый компьютер, то считать с нее информацию оказывается невозможным, но стоит запустить хотя бы одну программу – работоспособность дискеты восстанавливается. На самом же деле происходит активизация вируса и заражение еще одного компьютера.

5.Макро-вирусы. Могут заражать .doc файлы, созданные в текстовом процессоре MicroSoft Word. Заражение текстовых документов стало возможным благодаря использованию в программных продукта фирмы MicroSoft языка программирования высокого уровня Visual Basic Application. На этом языке записываются макрокоманды для автоматизированного выполнения некоторых операций с текстами. Естественно, что авторы вирусов использовали и эту возможность для своих черных целей.

6.Стелс-вирусы (stealth). Это вирусы-невидимки, которые используют специальные ухищрения, затрудняющие их обнаружение. Эти вирусы остаются в памяти резидентно и при обращении к зараженным файлам подменяют информацию так, что “заказчик” получает ее в незараженном виде. Увидеть такой вирус можно, если загрузиться с незараженной дискеты.

7.Вирусы-призраки – это самомодифицирующиеся вирусы, которые используют другой способ маскировки: они хранят свое тело в закодированном виде, причем способ кодирования все время меняется. При переносе такого вируса с компьютера на компьютер код вируса изменяется таким образом, что уже не имеет ничего общего со своим предыдущим вариантом.

Методы распространения вирусов.

Заражение компьютера вирусом может произойти в одном из следующих случаев:

1.При запуске на компьютере исполняемой программы, зараженной вирусом.

2.При загрузке компьютера с дискеты, зараженной загрузочным вирусом.

3.При подключении к системе зараженного драйвера.

4.При установке на компьютере зараженной операционной системы.

5.При обработке на компьютере зараженных текстовых файлов .doc.

Компьютер не может быть заражен, если:

1.на него переписывались текстовые, графические, информационные файлы и файлы данных;

2.на нем производилось копирование с одной дискеты на другую при условии, что не один из файлов с этой дискеты не запускался;

3.на компьютере производится обработка принесенных извне графических, информационных файлов и файлов данных.

Профилактические меры по предупреждению заражения компьютера.

Чтобы избежать заражения, необходимо выполнять ограниченное число правил, но выполнять их всегда, без исключения. Ниже приводятся эти правила.

1.Хранить все важные программы и данные вне компьютера, на дискетах или CD-ROM.

2.Результаты своей текущей работы необходимо ежедневно сохранять на дискетах.

3.Проверять все новые программы и драйверы перед первым запуском с помощью программ – детекторов (AIDSTEST, Dr.Weber, и т.д.)

4.Если программы попали к вам в заархивированном виде, перед их проверкой необходимо развернуть архив.

5.Перед загрузкой операционной системы с чужой дискеты – проверьте ее на наличие вирусов.

Если заражение уже произошло, то необходимо:

1.Выключить компьютер, сохранив предварительно результаты текущей работы на винчестере.

2.Загрузить компьютер с системной дискеты, защищенной от записи механически. На дискете должна находиться антивирусная программа.

3.Запустить с дискеты антивирусную программу и проверить винчестер на наличие вирусов. Вылечить все зараженные программы и удалить те, которые восстановить невозможно.

4.После проверки – переписать на винчестер системные файлы с помощью команды SYS C:

5.Перезагрузить компьютер с винчестера (дискету вынуть из дисковода).

6.Повторить проверку системы, используя возможно большее количество антивирусных программ.

7.Оценить потери от действий вируса, если необходимо – проверить целостность файловой системы с помощью программы SCANDISK.

8.Восстановить из архива все необходимые файлы и программы, которые были испорчены вирусом.

9.Повторить проверку системы на наличие вирусов.

10.Желательно произвести проверку вашего архива.

ГЛАВА 19. Алгоритм как управляющая информация

Понятие алгоритма.

Электронные вычислительные машины - это высокопроизводительные средства обработки информации, предназначенные для решения большого круга самых разнообразных задач . Однако ЭВМ - это всего лишь автомат (хотя и достаточно 'интеллектуальный'), который быстро и точно выполняет предписания, составленные человеком. Разработка таких предписаний, т.е. заведомое проектирование всего хода решения задач, - неотъемлемая часть деятельности, связанной с использованием вычислительных машин.

Современные вычислительные машины могут хранить программы вычислений в памяти, и тогда человеку остается только составить программу и поместить ее в память машины, а все остальное - собственно решение задачи - машина сделает автоматически, т.е. без участия человека.

Общим для вычислительной техники является то, что решение задачи на ней осуществляется посредством составления программы. В основу программы для вычислительной машины кладется алгоритм решения данной задачи, т.е. Точное предписание о последовательности действий, которые должны быть произведены для получения результата. Алгоритм является более общим понятием, чем программа, в этом смысле программа для вычислительной машины - это запись алгоритма решения некоторой задачи в виде, пригодном для данной вычислительной машины. Отсюда следует, что основная часть процесса решения задач с помощью программно управляемой техники - это разработка алгоритмов решения этих задач. Когда алгоритм решения задач ясен, он без особого труда может быть представлен на языке программирования.

АЛГОРИТМ - фундаментальное, основное, неопределяемое понятие, которому нельзя дать точное определение. Можно лишь приблизительно описать это понятие, например так:

АЛГОРИТМ - это предписание, однозначно задающее процесс преобразования исходной информации в виде последовательности элементарных дискретных шагов, приводящих за конечное число их применений к результату.

Небольшая историческая справка. Слово "алгоритм" - это не что иное, как латинская транслитерация (Algorithmi) имени знаменитого среднеазиатского ученого Мухаммеда бен Мусы аль-Хорезми. (787 - 850 годы). Его основополагающие трактаты по арифметике и алгебре, переведенные в XII веке на латинский язык, оказали существенное влияние на развитие математики в Западной Европе. В обыденной жизни мы постоянно сталкиваемся с алгоритмами как последовательностью действий, приводящих к достижению поставленной цели,- это правила перехода улицы, рецепты приготовления различных блюд, поиск нужного слова в словаре и др. Алгоритмы в математике - это правила нахождения корней квадратных алгебраических уравнений, правила выполнения арифметических действий, правила нахождения наибольшего общего делителя и другие.

Исполнитель - как средство реализации алгоритма

Разработанный алгоритм может исполнять как человек, так и техническое устройство, которое "понимает" правила записи алгоритма и умеет выполнять предписанные им действия. Это может быть, например, ЭВМ, программируемый калькулятор, робот на производстве, станок с ЧПУ (числовым программным управлением), а также различная бытовая техника, обладающая встроенными микропроцессорами (кухонные комбайны, автоматические стиральные машины, посудомоечные машины, видеомагнитофоны, телевизоры, СВЧ-печи).

Каждый исполнитель обладает рядом свойств:

1. система команд, т.е. некоторый список команд, которые исполнитель в состоянии понять и выполнить. Список команд связан с техническими особенностями того или иного исполнителя.
2. среда исполнителя - это та обстановка окружающего мира, в которой он может выполнять возложенную на него задачу (стиральная машина - автомат: вода, порошок, белье; станок с ЧПУ: заготовки деталей, стружка, масла, охлаждающая жидкость).
3. элементарные действия - это те действия, которые производит исполнитель, когда получает правильные и понятные для него команды. Каждая такая команда приводит к выполнению всего одного элементарного действия. Чаще всего выполняются действия по включению или отключению некоторого исполнительного механизма (например, в видеомагнитофоне - включение или отключение электромотора для перемотки ленты).

Понятие программы как алгоритма, записанного на языке, "понятном" исполнителю; способы записи алгоритма.

Свойства алгоритма
(дискретность, детерминированность, результативность, массовость).

Программа - описание процесса обработки информации на языке программирования, определяющее последовательность действий ЭВМ.

Программа - это способ описания алгоритма на языке, понятном исполнителю. Программирование - это процесс перевода алгоритма, записанного одним из известных способов, на язык, понятный машине.

Например, когда речь идет о человеке, алгоритм для него можно записать на естественном языке, в виде схемы или рисунка. Для вычислительной и другой техники, использующей микропроцессоры, придется записать программу на языке программирования или даже на машинном языке в двоичных кодах.

Таким образом, различают несколько способов записи алгоритмов – словесное (вербальное), графическое (с помощью схем) и табличное:

1. вербальная форма (словесное описание алгоритма на естественном языке). Сло-

весное описание имеет минимум ограничений и является наименее формализованным. Однако при этом алгоритм получается и наименее строгим, допускающим появление неопределенностей. Позволяет описать алгоритм с помощью слов и предложений.

2. схема алгоритма представляет собой графическое изображение алгоритма с помощью определенным образом связанных между собой блоков. Под блоком понимается любой конечный этап вычислительного процесса, принимаемый в данной схеме как целое. Внутри каждой геометрической фигуры дается описание операций, содержащихся в данном блоке. При этом используются символы математических операций и операций отношений.

3. табличное задание служит для представления алгоритма в форме таблицы и расчетных формул. Такой способ наиболее часто используют в экономических расчетах. Исходные данные и выходные данные (результаты решения) вносят в заголовки граф. Результаты вычисления значений выходных данных по формулам записывают в соответствующие графы.

Свойства алгоритмов:

Дискретность - это разбиение алгоритма на ряд отдельных законченных действий - шагов. Дискретность означает, что выполнение алгоритма разбивается на последовательность законченных действий - шагов. Каждое действие должно быть завершено исполнителем прежде, чем он перейдет к выполнению следующего. Значения величин в каждом шаге алгоритма получаются по определенным правилам из значений величин, определенных на предыдущем шаге.

Детерминированность - (или определенность). Под определенностью понимается то обстоятельство, что каждое правило алгоритма настолько четко и однозначно, что значение величин, получаемые на каком-либо шаге, однозначно определяются значениями величин, полученными на предыдущем шаге, при этом точно известно, какой шаг будет выполнен следующим. Другими словами, при одних и тех же исходных данных будут получены одни и те же результаты.

Результативность - обязательное получение результата за конечное число шагов. Результативность или конечность алгоритма предполагает, что его исполнение сводится к выполнению конечного числа действий и всегда приводит к некоторому результату. В качестве одного из возможных результатов является и установление того факта, что задача решения не имеет.

Массовость - применимость алгоритма к решению целого класса однотипных задач. Под массовостью понимается, что алгоритм решения задачи разрабатывается в общем виде так, чтобы его можно было применить для целого класса задач, различающихся лишь наборами исходных данных. При этом исходные данные могут выбираться из некоторой области, называемой областью применимости алгоритма. В свойстве массовости заключена основная практическая ценность алгоритмов.

Типы алгоритмов. Виды ошибок.

Правильным можно назвать такой алгоритм, который обладает всеми выше перечисленными свойствами. Другими словами, правильный алгоритм обеспечивает получение некоторого результата при любых исходных данных.

При выполнении алгоритма могут возникать различного типа ошибки:

1. семантическая ошибка - это смысловая ошибка в программе, не связанная с нарушением синтаксиса. К числу таких ошибок относятся: неправильное описание алгоритма решения задачи, неверное определение переменных, неверное использование пра-

вильно записанных конструкций языка и т.д.

2. синтаксическая ошибка - это ошибка этапа программирования, заключающаяся в нарушении грамматических правил языка; неправильная запись конструкций языка (эти ошибки, в отличие от семантических, обнаруживаются транслятором).

Основные типы алгоритмов:

1. Линейный алгоритм - не содержащий ветвей и циклов. Все команды такого алгоритма выполняются последовательно, в естественном порядке.

2. Разветвляющийся алгоритм - в котором происходит выбор одной из нескольких последовательностей команд (ветвей) при проверке некоторого условия (или тот, в котором происходит выбор из двух или более альтернатив). При движении по каждой ветви может встретиться следующее ветвление и т.д. Различают 4 вида ветвлений:

а) простое ветвление (другое название - "неполное" или структура "обход");
Пример на Паскале: if (x) and (y) thenwrite(x,y);

б) полное ветвление. Пример на Паскале:
```
if (x) and (y) then write('1')
else write('0');
```

в) множественное неполное ветвление (структура "выбор" ,без "else") :
Пример на Паскале:
```
case k of
    1: menu;
    2: input_record;
    3: output_basa;
    4..9: Quit;
end;
```

г) множественное полное ветвление (структура "выбор" с "else") :
Пример на Паскале:
```
case k of
    1: menu;
    2: input_record;
    3: output_basa;
else Quit;
end;
```

3. Циклический алгоритм - это алгоритм, содержащий один или несколько циклов (многократно повторяемых участков, состоящих из одной или нескольких команд). Типы циклов: с предусловием ("пока" - while..), с постусловием ("до" - repeat .. until) и с заданным числом повторений (или с параметром, цикл "ДЛЯ"- For..). Подробнее см. главу 9.

4. Вспомогательный алгоритм (процедура, функция или внешний модуль) - алгоритм являющийся частью другого алгоритма и выполняющий ряд действий. Вспомогательный алгоритм выполняется при вызове его из основной программы по имени. Вызов можно осуществить неограниченное число раз.

Функция отличается от процедуры тем что:

а) всегда входит в состав какого-либо оператора;

б) всегда возвращает в то место программы, откуда она была вызвана только одно значение;

в) всегда имеет определенный тип.

Процедура:

а) может ничего не возвращать (например, "рисовать" текстовое окно), а может возвращать одно или несколько значений переменных;
б) вызывается автономно, по имени, как любой оператор.

5. Рекурсивный алгоритм - вспомогательный алгоритм, вызывающий сам себя.
Пример на Паскале:

```
Function factorial (n:integer):integer;
  begin
    if n>1 then
      begin
        factorial:=factorial(n)*(n-1);
        n:=n-1;
      end;
  end;

 var k:integer;
 begin
    readLn (k);
    write (factorial(k));
 end.
```

ГЛАВА 20. Величины. данные

ОПРЕДЕЛЕНИЯ.

Переменная величина - элементарное средство представления данных в памяти ЭВМ.
У величины есть три атрибута - имя, значение и тип.

Физическая модель переменной величины - ячейка в оперативном запоминающем устройстве ЭВМ (ОЗУ, на английском RAM). Так как все данные в ЭВМ кодируются двоичным кодом, на самом деле это не одна ячейка, а совокупность элементарных ячеек ("бит"), в каждой из которых может быть записана логическая единица (Лог.1), либо логический ноль (Лог.0)

Имя переменной величины - комбинация из букв и цифр ("идентификатор") позволяющая разместить значение переменной величины в определенной ячейке ОЗУ, хранить его там и при необходимости считывать и обновлять. На модели переменной имя можно представить как ярлык на передней стенке ящика шкафа.

Типом величины называется множество ее допустимых значений и множество операций, которые можно совершать над этой величиной.
Тип характеризуется способом представления каждой величины - сколько знаков может быть в записи величины и какие комбинации знаков допустимы. Так как все величины кодируются в ЭВМ с помощью двоичного кода, от ее типа зависит сколько элементарных ячеек ОЗУ ("бит") будет занимать в памяти эта величина. Для ограничения путаницы между величинами разных типов и из-за физической невозможности для каждого типа есть жестко ограниченный список допустимых операций. Например, ясно, что величины символьного типа нельзя перемножать и т.п.
Некоторые операции производят "преобразование типов" над исходными данными. Например, результат деления - всегда "real", даже если число делится нацело!

ПРИМЕРЫ некоторых стандартных типов данных:
1. Тип "integer" (целые числа) в языке программирования Паскаль. Множество до-

пустимых значений чисел этого типа лежит в диапазоне от - 32768 до +32767. Допустимы все комбинации целых чисел в этом диапазоне. Для кодирования этих чисел достаточно 16 двоичных разрядов (один разряд используется для кодирования знака числа) или, как говорят, для хранения этих чисел надо два байта памяти.

2. Для вещественных ("real"). чисел диапазон значительно шире и допустимо бесконечное множество чисел в этом диапазоне, так как число чисел после запятой не ограничено. Однако на практике в ЭВМ точность представления вещественных чисел конечно ограничена, т.к. они кодируются специальным (см. ниже) образом с помощью двоичных разрядов.

В структурированных языках программирования используется очень большое многообразие типов величин. Это вызвано несколькими важными причинами: 1. повышение читабельности и простоты обработки данных разных типов.

Например массивы удобно использовать для хранения однотипных данных, а записи для создания баз данных;

2. экономия памяти. Хотя емкость памяти у современных ЭВМ очень большая она может быстро переполниться. Если данные занимающие много места размещать их в памяти надо рационально;

3. повышение быстродействия. ЭВМ производит действия с "большими" машинными словами, состоящими из большого числа разрядов, значительно медленнее, чем с "короткими".

Типы данных

1. boolean - логический, один бит

2. integer - целый, 16 бит (посмотрите в help TP70 интервал значений и подтипы!). Старший разряд - хранит знак числа (1-отрицательное число).

Два символьных типа:

3. char - литерный, 1 символ, 1 байт (8 бит, 256 комбинаций кода ASCII)

4. string - строковый, строка до 256 символов, до 256 байт (в описании этого типа можно ограничить его длину, если указать в квадратных скобках максимальное число символов в строке, например var name:string[10], тогда это уже 10 байт!).

5. real - вещественный, в ЭВМ представлен в виде чисел с плавающей запятой. Если выводить на экран значение величины типа real, то мы увидим, например такую запись: 3.24E+0008. На самом деле это 324 000 000. А такая запись: .12E-0005 означает 0,12 умноженное на 10 в минус 5 степени т.е. 0,00012. Такие числа состоят из мантиссы и порядка. Слева от "E" находится мантисса, а справа - порядок. У мантиссы один знак отводится под целую часть числа и 14 (!) под дробную. Порядок указывает на сколько знаков надо "сдвинуть" запятую в записи мантиссы. Если "+" сдвигать надо вправо, если "-", то влево. Переменные вещественного типа занимают в памяти целых 6 байт, т.е. 3 машинных слова для 16 разрядных ЭВМ. Мантисса и порядок хранятся в памяти отдельно. Обрабатываются такие величины за несколько машинных тактов. Если требуется большая точность или больший диапазон чисел, то можно использовать подтипы "real", их можно увидеть в "help" TP70 в соответствующей таблице.

Всем рассмотренным выше данным естественно могут соответствовать в программе на Паскале переменные, описанные в одном из разделов программы. Эти типы переменные, в отличии от динамических переменных, которые мы не рассматриваем, в ОЗУ имеют строго фиксированные адреса, однозначно связанные с именем переменной.

Адреса по которым размещаются переменные в ОЗУ задаются после запуска программы при распределении памяти ЭВМ.

Оператор присваивания

Оператор присваивания состоит из двух частей:

1. в левой части находится всегда имя переменной ("ячейки" ОЗУ), куда будут помещаться данные после выполнения правой части.

2. в правой части находиться выражение, значение которого и заносится в

переменную из левой части после производства вычислений.

NB! ВАЖНОЕ ПРАВИЛО!

Тип переменной величины в левой части оператора присваивания всегда должен соответствовать типу величины выработанной после выполнения правой части присваивания.

Из этого правила возможны исключения, когда происходит, так называемое, "преобразование типов"! Например, в левой части переменная вещественного типа, а в правой целого или слева строчная переменная, а справа литерная. В этом случае действует следующий принцип: переменная слева может "вобрать" в себя величину другого типа, если она занимает в памяти меньше места и кодируется тем же способом.

Рекуррентные выражения в операторах присваивания.

Если в выражении из правой части оператора присваивания есть в наличие имя переменной такое же что и в левой части, то это выражение называется "РЕКУРРЕНТНЫМ". Например: x:=x*10/y; Такие выражения часто используют в циклах. Простейшие случаи - "счетчики", увеличивающие или уменьшающие свое значение с заданным шагом. Например: a:=a+1; Для объяснения работы такого счетчика надо четко представлять себе, что в его правой части находиться "старое" значение переменной, которое изменяется с заданным шагом, а в левой – "новое", которое примет переменная после выполнения оператора присваивания.

NB! При использовании рекуррентных выражений не забывайте до начала цикла задать начальные значения переменным, которые используются в этих счетчиках !

Надо заметить, что и в цикле "for…" уже есть такой "скрытый" счетчик, автоматически изменяющий параметр цикла с шагом "+1" или "-1" от начального до конечного значения.

Все вышесказанное относительно типов данных полностью соотносится не только с языком программирования паскаль, но и с электронными таблицами и базами данных или другими пользовательскими программами, потому что там ис- пользуются для вычислений те же математические функции и способы кодирования информации. Там есть и другие типы данных, необходимые пользователю, например ВРЕМЯ или ДАТА, ГОД. Такие типы можно создать и на паскале, используя специальный раздел описания пользовательских типов "Type".

ПРИМЕР использования раздела описания пользовательских типов "Type".
(см. Help TP70 - F1)

ПОРЯДКОВЫЕ ДАННЫЕ.

Все данные можно разделить на порядковые и не порядковые.

К порядковым относятся данные тех типов, которые располагаются в определенном порядке и изменяют свое значение дискретно. Данные этого тип имеют ограниченное число значений. К данным этого типа относятся, например, данные типа "integer", "char", "boolean". Тип "real" к порядковым не относится, так как на числовой прямой между двумя точками, с которыми идентифицируются вещественные (действительные) числа, может находиться бесконечное число других точек.

ГЛАВА 21. Циклические процессы. способы организации цикла

Синоним слова "цикл" - "повторение".

Циклические процессы очень часто встречаются в окружающем нас мире. Можно сказать, что они положены в основу процессов, происходящих в природе.

В качестве примеров можно привести следующие процессы:

1. Развитие Вселенной Сначала был "Большой взрыв", границы вселенной начали расширяться. Из вещества стали образовываться планеты и светила. Зародились раз-

личные формы жизни. После того как вселенная достигнет определенных размеров - она начнет уменьшаться, вещество сожмется в одну точку и вновь произойдет взрыв. Все повторится.

2. Смена времен года в связи с циклическим движением Земли вокруг Солнца.
3. Круговорот воды в природе.
4. Движения животного при ходьбе, птицы в полете, рыбы в воде.

А вот примеры циклических процессов, являющихся результатом человеческой деятельности:

1. В экономике: кризис - оживление - подъем - перепроизводство - застой - кризис - оживление и т.д.
2. Работа двигателя внутреннего сгорания.
3. Работа светофора.
4. Колебания электронов в цепи переменного тока.
5. Оклеивание стен обоями (плиткой, деревянными панелями).
6. Процесс книгопечатания (многократная печать одинаковых страниц с помощью печатного станка, ксерокса или ризографа).
7. Занятия в тренажерном зале.

ОПРЕДЕЛЕНИЯ

Циклом называется любой повторяющийся периодический процесс.

Циклический алгоритм - порядок действий, который может периодически повторяться.

Существует два основных (базовых) типа цикла - цикл <ПОКА> и цикл <ДО>:

1) Цикл с предусловием ("ПОКА", на языке Паскаль - "while") - цикл, в котором логическое выражение проверяется в начале. Проверка значения логического выражения ("условия") в таком цикле стоит перед телом цикла.

Цикл повторяется при значении логического выражения TRUE (истина).

Пример использования цикла <ПОКА>:

ЗАДАЧА.

Определить номер первого четного элемента массива.

```
i:=1;
while x[i]/2<>int(x[i]/2) do i:=i+1;
write('ОТВЕТ=',i);
```

ЗАДАЧА.

Вычислить сколько раз в заданной строке встречается заданная подстрока.

```
var stroca, substroca:string;      {описание переменных}
begin
  readln(stroca, substroca);       {ввести строку и подстроку с клавиатуры}
  n:=0;                            обнулить счетчик числа подстрок в строке;}
 while pos (substroca,stroca)<> 0 do        {пока подстрока есть в строке,
  Begin                                     повторять тело цикла}
delete(stroca,pos(substroca,stroca),lenth(substroca));
                                   {удалять подстроку в строке}
 n:=n+1;                           {увеличить значение счетчика на 1 end;}
 write(n);                         {вывести значение счетчика}
  end.
```

2) Цикл с постусловием (цикл "ДО", на языке Паскаль "repeat .. until") - цикл в котором логическое выражение находится в конце цикла.

Отличия цикла <ДО> от цикла <ПОКА>:

1. проверка значения логического выражения ("условия") в таком цикле стоит после тела цикла .
2. цикл повторяется при значении логического выражения FALSE (ложь).
3. в любом случае цикл выполнится хотя бы один раз.
4. тело цикла можно не брать в операторные скобки.

Пример использования цикла "ДО".

ЗАДАЧА.

Определить номер первого четного элемента массива.

```
i:=0;
repeat
i:=i+1;
until x[i]/2=int(x[i]/2);
write('ОТВЕТ=',i);
```

ЗАДАЧА.

Вычислить, сколько раз в заданной строке встречается заданная подстрока.

```
var stroca, substroca:string;
  begin
    readln(stroca, substroca);        {ввод подстроки и строки с клавиатуры}
    n:=-1;                            {установка исходного значения счетчика}
     repeat
       delete(stroca,pos(substroca,stroca),lenth(substroca));
       N:=N+1
     until pos (substroca,stroca)=0;
     write('ОТВЕТ=',N);
```

Как мы видим один цикл с успехом заменяет другой и какой из них использовать - дело вкуса. Если в языке программирования нет операторов "while" и "repeat", то их можно "сконструировать" используя оператор ветвления "if" и оператор безусловного перехода на метку ("goTo"). В тех языках программирования где есть операторы цикла "ПОКА" и "ДО" оператор "goTo" не рекомендуется использовать в программах, чтобы "не портить" их структурированность и "стиль" программирования учащихся.

Циклы "ДО" и "ПОКА" используют, когда заранее не известно сколько раз будет выполняться цикл. Если это известно, то лучше воспользоваться третьим типом цикла "ДЛЯ".

3) Цикл с параметром "ДЛЯ" (FOR).

В заголовке этого цикла присутствует переменная порядкового типа ("параметр"), которая при выполнении цикла автоматически изменяет свое значение с шагом +1 или -1 от своего начального значения до конечного. Поэтому, если шаг +1, конечное значение параметра цикла должно быть больше или равно начальному, а если шаг -1, то наоборот. В противном случае цикл не будет выполняться ни разу. Если конечное значение равно начальному цикл "ДЛЯ" будет выполнен один раз. Из этого следует, что цикл "ДЛЯ" - частный случай цикла "ПОКА".

К порядковым относятся данные тех типов, которые располагаются в определенном порядке и изменяют свое значение дискретно. Данные этого тип имеют ограниченное число значений. К данным этого типа относятся, например данные типа "integer", "char", "boolean", а тип "real" к порядковым не относится.

Синтаксис оператора цикла "ДЛЯ" на Паскале:

```
n:=0;
for i:=-3 to 2 do
```

```
  begin
    n:=n+1;
    write(i:3);
  end; {for}
```

На экране появиться " -3 -2 -1 0 1 2", "n" примет значение "6".

Для изменения параметра цикла с шагом - 1 "to" надо заменить на "downTo".

NB! Ограничения на использование параметра цикла:

а) не изменяйте параметр цикла внутри цикла!

б) по окончании цикла не используйте значение параметра цикла! Эти ограничения позволяют сделать программу более структурированной, т.е. читабельной и легко отлаживаемой.

ВЛОЖЕННЫЕ ЦИКЛЫ.

Вложенными называются циклические алгоритмы в которых внутри одного цикла ("внешнего") есть один или несколько других циклов. Эти циклы называются внутренними.

Порядок выполнения вложенных циклов - пока внутренний цикл не закончился во внешнем цикле ничего не изменяется.

Порядок работы вложенных циклов "ДЛЯ" параметр внешнего цикла принимает исходное значение, затем полностью выполняется внутренний цикл. Параметр внешнего цикла изменяет свое значение и снова полностью выполняется внутренний цикл и так до тех пор, пока параметр внешнего цикла не достигнет своего конечного значения.

Например:

```
for i:=1 to 3 do            { внешний цикл по "i"}
 begin
  for j:=1 to 4 do write(i:2,j:2,'*');      внутренний цикл по "j"}
  writeLn;                  { перевод курсора на новую строку }
 end; { for i }             { конец внешнего цикла}
```

В результате выполнения алгоритма оператор write(i:2,j:2,'*') будет выполнен 12 раз и на экране последовательно появятся строки:

```
1 1* 1 2* 1 3* 1 4*
2 1* 2 2* 2 3* 2 4*
3 1* 3 2* 3 3* 3 4*
```

Правило использования вложенных циклов "ДЛЯ" в структурированных программах: Для того чтобы программа была читабельной, не используйте во вложенных циклах одноименные параметры!

Практический пример использования вложенных циклов - заполнение значениями двумерного массива.

"БЕСКОНЕЧНЫЕ ЦИКЛЫ"

Несколько отдельно от трех рассмотренных базовых циклических алгоритмов находится, так называемые, "бесконечные циклы".

"Бесконечным" принято называть такие алгоритмы, выполнение которых можно прервать только выключив ЭВМ или нажав "Ctrl"+"Break", при выполнении такого алгоритма на Паскале. Строго говоря, бесконечные циклы использовать в программах неверно, т.к. общий алгоритм становится нерезультативным - рядовой пользователь долго будет сидеть и глядеть на "черный экран" и не догадываться, что надо нажать "Ctrl"+"Break"! Поэтому в паскале существуют специальные операторы прерывания цикла, например "exit", позволяющие этого избежать.

Для организации "бесконечного цикла" проще всего использовать оператор безусловного перехода "goto" на метку в программе. Например:

```
1: goTo 1;
```

где "1" - метка в программе, описанная в разделе описания меток, для нашего примера label 1; Метка может быть и символьного типа.

Как уже отмечалось оператор "goTo" и метки не рекомендуется использовать в программах, поэтому для организации "бесконечного цикла" принято использовать операторы "пока" или "до" с логическими константами true, false,

например:

```
while true do    repeat
  begin...
  .... или ...
  end;until false;
```

ОРГАНИЗАЦИЯ ЦИКЛОВ С ПОМОЩЬЮ РЕКУРСИВНЫХ АЛГОРИТМОВ.

К циклическим алгоритмам относятся также и рекурсивные алгоритмы.

Рекурсивными называются такие алгоритмы, которые вызывают сами себя по своему имени.

Из определения следует, что в программах написанных на Паскале рекурсивными могут быть процедуры и функции, так как они поименованы. Возможность создавать рекурсивные алгоритмы появилась только с появлением структурированных языков программирования в которых существуют параметры - переменные..

Пример: рекурсивная функция, вычисляющая факториал.

```
uses crt;
var k:integer;

Function factorial(n:integer):integer;
 begin
if n=0 then
 begin
write('*');
factorial:=1
  end

   else
  begin
write('-');
factorial:=factorial(n-1)*n;
  end; { if }
 write('+');
  end; { Function }

  begin
clrScr;
readLn(k);
write(factorial(k));
repeat until keypressed;
  end.
```

Что же появится на экране, если задать с клавиатуры, например, "k" равное 5? Кроме правильного значения факториала, появится "----*++++++", пять минусов, одна звездочка и шесть плюсов. Сколько же это повторений, чего и почему получается верный результат?

При рекурсивном вызове функции или процедуры в любом структурированном языке

программирования реализован, так называемый, "принцип стековой памяти". Он действует по правилу: "Сколько раз вошли в рекурсивный алгоритм (в стек) - столько раз из него вышли".

Поэтому в правой части оператора "factorial:=factorial(n-1)*n" функция "factorial(n-1)" будет вызываться 5 раз, каждый раз с новым значением параметра-переменной "n". При этом выполнение оператора присваивания, пока еще полностью не заканчивается и его левой части ничего не присваивается. Параметр-переменная "n" при каждом новом вызове функции уменьшается на единицу. Что записывается в стековую память? Кроме адреса места в программе откуда была вызвана функция, позволяющего потом выйти из рекурсии, в стек записываются значения параметров переменных, с которыми происходит "вход" в рекурсию. Для нашей задачи в него последовательно будут записаны величины 5,4,3,2,1.

Когда значение "n" уменьшится до нуля, значению функции будет присвоена единица и на экран выводится "*". Только теперь завершается выполнение оператора ветвления "if" и печатается первый значек "+".

Конец выполнения функции. И тут компьютер "вспоминает", что он то вызывал эту функцию уже пять раз, значит он столько раз должен из нее выйти. Где он ее вызывал? Это он определяет по содержанию стека - в правой части оператора "factorial:=factorial(n-1)*n". Теперь, согласно принципу стековой памяти, надо из стека выйти столько раз сколько мы туда вошли, то есть пять раз. При этом начинает работать оператор присваивания "factorial:=factorial(n-1)*n" в плане присваивания значения его левой части.

При выходе из стека действует правило "последним вошел - первым вышел".Поэтому, из стековой памяти в правую часть этого оператора будут последовательно, при каждом новом выходе из рекурсии, подставлены значения "n" в обратном порядке по сравнению со входом в стек, то есть 1,2,3,4,5. На каждом новом шаге это очередное значение "n" умножается на предыдущее значение функции и присваивается ей же. На первом шаге 1*1, на втором 1*2, третьем 2*3, четвертом 6*4, пятом 24*5. Таким образом получается искомое значение факториала. При выполнении каждой из этих операций выводится знак "+", поэтому всего их на экране шесть.

На первый взгляд, в использовании рекурсии выигрыша, по сравнению с классическим использованием цикла "ДЛЯ", нет. Для этого примера это действительно так. Но есть большой класс задач, в которых можно значительно сократить число повторений при использовании рекурсий. В первую очередь это "переборные" задачи, например задача перебора ветвей бинарных деревьев. Или "шахматные" задачи. При их решении рекурсия позволяет отказаться от простого перебора всех вариантов с помощью вложенных циклов "ДЛЯ". Владение рекурсией определяет класс программиста.

ГЛАВА 22. Ветвления. способы организации ветвления

Ветвящиеся процессы встречаются достаточно часто как в окружающей нас природе, так и при решении всевозможных задач в повседневной жизни человека. Ветвление - это результат наступления некоторого события, которое изменяет обычный ход протекающего процесса. В простейшем случае различают две ветви: та, по которой протекал бы процесс, если бы событие не наступило, и та, по которой будет протекать процесс, если указанное событие наступит. В алгоритмах, которые создаются человеком для управления исполнителями или решения других задач часто необходимо принимать решения и выбирать один из путей решения задачи в зависимости от значения логических высказываний, поэтому языки программирования высокого уровня снабжаются специальными средствами проверки наступления событий, которые могут изменить ход алгоритма.

ОПРЕДЕЛЕНИЕ

Разветвляющимся называется такой алгоритм, в котором происходит выбор из двух или большего числа возможных ветвей выполнения алгоритма ("альтернатив"). В зависи-

мости от этого выбора выполнятся одна из альтернативных (возможных) ветвей алгоритма. Простейший случай - выбор из двух альтернатив (возможностей). При этом ветвление осуществляется в зависимости от значения логического выражения ("условия"), которое может принимать одно из двух значений: TRUE ("ИСТИНА") или FALSE ("ЛОЖЬ"). Существует две формы такого ветвления:

1) Неполное ветвление.

В зависимости от значения логического выражения выполняется определенное действие или осуществляется его обход. Поэтому неполное ветвление еще называют "алгоритмическая структура обход". Действие выполняется, если логическое выражение принимает значение TRUE, а если его значение FALSE, то выполняется обход и тогда начинает выполнятся действие идущие после ветвления.

Пример 1: допустим, в программе "Устный счет" в переменных "x", "y" - случайные числа, а в переменной "z" пусть находится ответ пользователя на вопрос, сколько будет "x*y". Необходимо выяснить - правильно ли он ответил, и если верно, то поздравить его с удачей и увеличить содержимое счетчика правильных ответов на единицу, а если он ошибся сообщить ему об этом:

```
  if x*y=z then
begin
 write('Молодец,', name, '!');
 s:=s+1
end;
  if x*y<>z then write('Неверно', name, '!');
```

Пример 2: предположим, "s" - переменная, в которой содержится число правильных ответов в программе "Устный счет". Необходимо выставить оценку. Было задано "n" вопросов. Можно организовать четыре простых ветвления:

```
  if s=n then write('Ваша оценка - 5');
  if (s<>n) and (s>=0.8*n) then write('Ваша оценка - 4');
  if (s<0.8*n) and (s>=0.5*n) then write('Ваша оценка - 3');
  if (s<0.5) then write('Ваша оценка - неуд.');
```

2) Полное ветвление.

В таком разветвляющемся алгоритме уже не одно, а два действия. Если логическое выражение принимает значение TRUE, то выполняется первое из этих двух действий, иначе ("else"), если выражение принимает значение FALSE, выполняется второе действие.

Пример 3: условие такое же, как и в примере 1.

```
  if x*y=z then
begin
 write('Молодец,', name, '!');
 s:=s+1
end
else
 write('Неверно', name, '!');
```

Как видно, алгоритм короче и яснее.

ВНИМАНИЕ!!! Перед "else" не надо ставить ";", т.к. оператор "if" еще не закончился!

Пример 4.: условие такое же, как и в примере 2.

```
  if s=n then
```

```
write('Ваша оценка - 5')
  else
if  s>=0.8*n then
 write('Ваша  оценка  -  4')
else
 if s>=0.5*n then
  write('Ваша оценка - 3')
 else
  write('Ваша оценка - неуд.');
```

Алгоритм короче, исчезли "and". Такая алгоритмическая структура называется "вложенные полные ветвления" (их здесь три).

3) "Множественное" ветвление, полная форма (с "else").

Фактически, в выше рассмотренном примере 4 присутствует уже "множественное" ветвление, только реализованное с помощью нескольких операторов <if>.

ОПРЕДЕЛЕНИЕ

Множественным. называется ветвление в котором происходит выбор из нескольких альтернатив (возможностей). Такие алгоритмы называют "структура выбор".

В предыдущем примере он реализован с помощью трех вложенных друг в друга ветвлений. В развитых ЯВУ существует специальный оператор "выбор" позволяющий такой алгоритм записать короче. На языке Паскаль это оператор "case", который также как оператор <if> имеет полную и неполную формы. В операторе используется переменная - "ключ", в зависимости от значения которого алгоритм выполняется по той или иной ветви. Если "ключ" не принимает ни одно из указанных значений, то выполняются действия после "else".

NB! ВНИМАНИЕ! Ключ не может быть величиной вещественного типа (<real>)!

Пример 5: условие такое же, как и в примере 2.

```
case s of
                    n : write('Ваша оценка - 5');
          int(0.8*n)..n : write('Ваша оценка - 4');
 int(0.5*n)..int(0.8*n) : write('Ваша оценка - 3')
else write('Ваша оценка - неуд.');
end;
```

NB! У этого оператора есть <end> без <begin>!

В этом примере в качестве значений ключа использованы величины интервального типа.

Как видно из примера, "case" позволяет сделать алгоритм намного короче и понятней, чем если его реализовать с помощью "if"!

4) Множественное ветвление, неполная форма (без "else").

В этой форме, если "ключ" не принимает ни одно из указанных в операторе значений, то выполняются действия следующие после оператора "case".

Пример 6: условие такое же, как и в примере 2.

```
  case s of
                    n : write('Ваша оценка - 5');
          int(0.8*n)..n : write('Ваша оценка - 4');
int(0.5*n)..int(0.8*n) : write('Ваша оценка - 3');
          0..int(0.5*n) : write('Ваша оценка - неуд.');
 end;
```

На экзамене необходимо изобразить блок схемы всех рассмотренных выше разветвляющихся алгоритмов.

ГЛАВА 23. Массивы как способ организации данных

Самые простые массивы - одномерные.. В одномерных массивах все элементы имеют общее имя и свой индекс. Существуют также многомерные массивы, например, двумерные, трехмерные… У многомерных массивов число индексов соответствует их размерности. Одномерные массивы очень часто называют также ЛИНЕЙНЫМИ ТАБЛИЦАМИ, а двумерные массивы ПРЯМОУГОЛЬНЫМИ ТАБЛИЦАМИ или МАТРИЦАМИ. Эти названия распространены так широко, потому что именно в эти таблицы удобно записывать значения элементов соответствующих массивов. Еще одно название массивов - ВЕКТОР. Если массив одномерный - это линейный вектор, для двумерного массива - это вектор на плоскости, т.е. двумерный вектор. и т.д.

Название "вектор" связано с направлением, которое указывает, где в пространстве находится тот или другой элемент массива относительно его начала. С этой точки зрения индексы - это некоторые упорядоченные точки в пространстве определенной размерности. Для одномерного массива - одномерное пространство (линия), для двумерного - плоскость, для трехмерного - декартово пространство, а для других уже многомерные пространства Лобачевского.

ОПРЕДЕЛЕНИЕ 1
Массив - тип данных с составными значениями.

Это значит, что внутри массива еще что-то находится! Ограничение назначения, которые находятся внутри массива - они все должны быть одного типа (например, или все целые числа, или все строки). Этим массивы кардинально отличаются от типа данных "запись", у которой "внутри" могут быть данные разных типов (например, и целые числа и строки, что очень удобно для организации баз данных!).

ОПРЕДЕЛЕНИЕ 2
Массив - совокупности однотипных данных, имеющих общее имя и различающихся друг от друга с помощью индексов.

Индекс элемента массива - признак, по которому можно найти внутри массива необходимые данные (значения элементов массива). Массив очень удобно можно представить в виде линейной таблицы, поэтому их так часто и называют.

ХАРАКТЕРИСТИКИ МАССИВОВ

РАЗМЕРНОСТЬ - число элементов массива. Задается при описании массива.
При запуске программы в памяти ЭВМ резервируется необходимое место. К сожалению, после запуска программы нельзя изменять число элементов массива.
ИМЯ МАССИВА - задается по тем же правилам, что и имя любой переменной.
ТИП ЭЛЕМЕНТОВ МАССИВА - все элементы массива должны быть одного типа, который задается при описании массива.

Пример массива: В фирме 100 сотрудников, их табельные номера от 1 до 100. Можно создать массив, состоящий из заработных плат всех сотрудников фирмы, а в качестве индексов для этого массива использовать их табельные номера.

Индексы массива не обязательно должны начинаться с 1, но так удобнее. На языке Паскаль индекс записывается в квадратных скобках. Например: X [3], A [C].
NB! Самое важное в идее использования массивов состоит в том, что индекс масси-

ва может быть переменной величиной!

Индекс массива обязательно должен быть ПЕРЕЧИСЛИМОГО ТИПА!

Например, индекс элемента массива может быть целого (integer или его подтипы) типа или литерного (char). В качестве индекса может выступать и выражение, но только если его значение соответствует вышеуказанным типам!

Индекс массива НЕ МОЖЕТ БЫТЬ вещественного типа "REAL"!

Массивы нашли столь широкое распространение, т.к. индекс массива может быть переменной величиной, которую очень удобно изменять в цикле (перебирать элементы массива).

Массивы на языке Паскаль описываются в разделе переменных следующим образом:

```
const n=10;
var x, y: array[1..n] of integer;
```

Требование - начальный индекс должен быть меньше конечного! Массив можно сразу задать и в разделе констант:

```
const x: array [1..5] of integer=(4,54,2,1,56);
```

ОПРЕДЕЛЕНИЕ 1

Структура данных "массив", элементы которого сами являются массивами называется "двумерными массивами" (массив массивов). Исходя из этого определения описать двумерный массив можно так:

```
const n=10,  m=5;

var x:array[1..n] of array[1..m] of integer;
```

Но так не очень удобно, поэтому чаще дают такое определение:

ОПРЕДЕЛЕНИЕ 2

Массив у которого элементы имеют "двойной" индекс называют двумерными (по аналогии трехмерными и т.д.).

Двумерный массив удобно представить в виде прямоугольной таблицы, состоящей из столбцов и строк (ее называют также "матрицей"). Исходя из этого определения описать двумерный массив можно так:

```
const n=10, m=5;
  var x:array[1..n,1..m] of integer;
```

Отличия одномерного массива от двумерного массива (матрицы) с точки зрения организации хранения их в памяти и доступа в них к отдельным переменным:

1) В памяти (ОЗУ) массивы располагаются компактно, занимают соседние ячейки, одну за другой. Для двумерных массивов: за строкой идет строка.

2) Для организации доступа к элементу одномерного массива надо указать только один индекс, а к элементу двумерного - два индекса (номер столбца и строки)

Стандартные алгоритмы обработки массивов: заполнение, вывод значений элементов на экран, нахождение минимального и максимального элементов, основные виды сортировок элементов массива.

Создание и вывод на экран элементов двумерного массива заполненного случайными числами:

```
randomize;
For i:=1 to n do
  For j:=1 to m do
    begin
      x[i,j]:=random(10);
      GoToXY(i*2+10,j);
      write(X[i,j]);
    end;
```

Способы создания массивов:

1) В разделе констант.

2) С помощью оператора присваивания, в частности, например заполнение случайными числами, что удобно для отладки программ (см. выше).

3) Ввод данных с клавиатуры с помощью оператора ReadLn (x[i]);

4) Ввод данных с ГМД из файла данных с помощью файловой переменной и переопределения действия оператора read;

СТАНДАРТНЫЕ АЛГОРИТМЫ обработки массивов.

1. Поиск минимального значения из всех величин элементов массива.

```
min:=x[1,1];            {"эталон" для сравнения}
For i:=1 to n do
  For j:=1 to m do
    if min>x[i,j] then min:=x[i,j];
  writeLn ('Min элемент среди всех величин элем. массива=',min);
```

2. Основные виды сортировок элементов массива:

1) Метод перестановок. ("обменная" сортировка или метод "пузырька", "камешка"). При первом проходе сравнивают соседние элементы от первого до предпоследнего ("текущий", "i" элемент и следующий "i+1"). Если предыдущий элемент меньше следующего ("легче") - поменяем их местами через промежуточную переменную. Эти операции производится при первом проходе внутреннего цикла! Эти проходы необходимо повторять (n-1) раз с помощью внешнего цикла (т.к. число пар в массиве n-1!). Но ведь крайние правые элементы уже упорядочены, поэтому во внутреннем цикле достаточно сравнивать не все пары, а каждый раз на одну меньше!

```
for j:=1 to n-1 do        { цикл повторяется ст. раз, ск. надо "поднять пузырьков"}.
  for i:=1 to n-j do      {"подъем пузырька" снизу до незанятого места}
    if x[i]>x[i+1] then
  begin
    s:=x[i];              {если "нижний" (предыдущий) элемент меньше
    x[i]:=x[i+1];         последующего, он меняется с ним с
    x[i+1]:=s             помощью промежуточной ячейки s}
  end;
writeLn('Упорядоченный массив:');
For i:=1 to n do write('X[',i,']=',X[I],' ');
```

Преимущества метода - программа самая простая. Недостатки метода - самый медленный, т.к. много лишних обменов! Почему бы не найти самый большой, и только потом переместить его на край? Это уже "метод выбора".

2) Метод выбора.

```
for j:=1 to n-1  do          {повторить n-1 раз поиск максимального элемента}
  begin
    max:=x[1];k:=1;          {"образец" для поиска и его номер}
    for i:=2 to n-j+1 do     {поиск максимума от 2 до "не рассортированного"}
      if x [i] > max then
        begin
          max:=x[i];
          k:=i;
        end;                 {конец внутреннего цикла}
      s:=x[k];               {обмен максимального
      x[k]:=x[n-j+1];        с крайним "не рассортированным"}
      x[n-j+1]:=s
  end;                       {конец внешнего цикла}
writeLn ('Упорядоченный массив:');
For i:=1  to n do write (X [I],' ');
```

Существует еще множество видов сортировок. Главное их отличие друг от друга - быстродействие, т.к. время на сортировку зачастую растет в геометрической прогрессии в зависимости от числа элементов массива. Для каждого типа данных и для каждого вида сортировки (а ведь надо часто сортировать сразу по нескольким критериям) существует свой метод сортировки. Это очень большая проблема для баз данных. Если Вы хотите стать профессионалом, постарайтесь найти классический источник, книгу Н. Вирт, Алгоритмы и структуры данных, М. Мир, 1989, в нем описано около 40 алгоритмов сортировки.

ГЛАВА 24. Технология решения задач с помощью ЭВМ

Для решения любой задачи с помощью ЭВМ необходимо произвести шесть последовательных операций - звеньев технологической цепочки (технология - способ решения поставленной задачи). Рассмотрим эти этапы на примерах расчета зарплаты для сотрудников некой фирмы и построения на ЭВМ модели электрической схемы с помощью ЭТ.

1) Формирование у заказчика и исполнителя исходного представления об объекте.

Необходимо получить сведения о природе рассматриваемых объектов, их составе, взаимодействии их компонентов в общем виде не вдаваясь в детали. Для первого примера - это механизм начисления зарплаты в общих чертах, состав фирмы; для второго - электросхема, ее компоненты, характер внешних воздействий.

2) Формулировка цели исследования.

Очень часто считают, что "компьютер сам все сделает", но от него можно получить только то, что попросишь. Следовательно - определить цель которую должна решить программа - это очень важный, длительный и ответственный этап, от которого зависит работоспособность готового программного продукта. Нередко забывают, что ЭВМ не только должна выдать какой то конкретный результат, например число или список, но и корректно обработать все возможные ситуации, когда конкретного результата получить невозможно.

К примеру при вычислении зарплаты, предположим, что из фирмы все сотрудники уволились. Тогда может произойти сбой при вычислении средней зарплаты при делении на ноль, если этого заранее не предусмотреть. Такие ситуации называют "не штатными", "экстремальными", а в математике - "граничными условиями". Все такие ситуации заранее часто невозможно предусмотреть, к ним придется многократно возвращаться при отладке программного продукта. На этом этапе закладывается свойство "результативности" алгоритма - для любых комбинаций исходных данных необходимо получить определенный осмысленный, правильный результат, а не "черный экран" или непонятное для пользователя сообщение транслятора.

Цель исследования для первого примера - вычисление зарплаты сотрудников по отделам, а также поиск нужного сотрудника по фамилии. Если такого сотрудника нет, должно выдаваться сообщение об этом, а не чистый экран.

Цель исследования для второго примера - знакомство учащихся с закономерностями протекания постоянного тока по цепи, состоящей из нескольких резисторов и иллюстрация пожароопасных ситуаций в таких цепях.

3) Постановка задачи.

Конкретно описывается, исходя из поставленной цели, какие данные вводятся в ЭВМ и в какой форме (виде) и какие данные (или воздействия на исполнительные механизмы) и в какой форме должны быть получены после решения задачи. Этот этап называется **"разработка технического задания (ТЗ)".**

Входные данные для первого примера: список сотрудников, их табельные номера, номера отделов, должности, тарифные ставки (сколько они зарабатывают в час), число отработанных нормо-часов.

Выходные данные: в зависимости от номера отдела на экран должен выводиться список сотрудников со всеми данными о нем, включая зарплату, рассчитанную в согласно отработанным нормо-часам и среднюю зарплату по отделу. Предусмотреть возможность поиска сотрудника по фамилии, при этом фамилия вводится с клавиатуры и не обязательно полностью.

Входные данные для второго примера: напряжение источника питания, величина сопротивления резистора и амперметра, величина опасного тока. Выходные данные - ток в цепи и сообщение о его опасной величине.

Если бы цель исследования была другая, то и техническое задание пришлось бы изменить, например, нужен отчет по зарплате перед финансовыми органами. Тогда в финансовых органах надо узнать, какие они требуют формы и исходя из них изменить ТЗ.

Если цепь для второго примера - не иллюстрировать свойства электросхемы, а ее полное исследование, то необходимо учитывать сопротивление вольтметра и каждого провода отдельно.

4) Построение информационной модели.

- ***Информационная модель - набор величин, содержащих всю необ-***

ходимую информацию о исследуемых процессах.

Для описания этих величин предварительно необходимо построить математическую и, если это необходимо, физическую модели задачи. Это означает, что процессы влияющие на результат решения задачи должны быть адекватно описаны математическими формулами (построена математическая модель), часто с использованием приемов физического моделирования. Снова должны быть сделаны некие упрощения процесса, исходя из цели и ТЗ.

Для электросхемы, используя закон Ома, надо описать токи и напряжения в схеме с помощью формул. Для начисления зарплаты необходимы бухгалтерские формулы (порядок начисления подоходного налога). Необходимо решить, например, куда девать доли копеек?

Затем необходимо выбрать инструментальное программное средство с помощью которого эта задача может быть наиболее эффективно реализована. При выборе можно использовать следующие критерии:

- **простота реализации** (время создания, простота отладки, дешевизна ПО). Нам проще обе задачи решить с помощью ЭТ и БД;
- **простота использования потребителем** . Потребителю проще использовать ".exe" файл, чтобы не знакомится с оболочкой ЭТ и БД. Для этого пришлось бы создавать для первой задачи БД например созданную на Паскале, а для второй использовать графические возможности Паскаля. Это намного более трудоемко, если не использовать Delphi, Visual Basic и т.п.

Предположим, что первую задачу реализуем на Паскале, вторую с помощью ЭТ.

5) Построение алгоритма.

Понятно, что в первом случае алгоритм громоздкий и сложный, а во втором только отдельные алгоритмические конструкции используются в ячейках ЭТ в виде формул (хотя и там можно написать целую программу, используя встроенный алгоритмический язык). Рекомендуется сначала записать алгоритм с помощью структурной схемы в виде специальных графических обозначений и только потом реализовать его в виде программы!

6) Компьютерная реализация.

Программа набирается в редакторе языка программирования, электронная таблица заполняется данными и формулами . Для получения результатов программа и ЭТ "должны быть отлажены".

Принципы отладки и тестирования алгоритмов и программ.

- ***ОТЛАДКА - процесс поиска и исправления ошибок в программе или другом программном продукте.***

Отладку производят с помощью программных средств, которые называют **ТРАНСЛЯТОРАМИ.**

- ***ТРАНСЛЯТОР - программное средство, переводящими текстовую информацию в машинные коды, понятные центральному***

процессору, для их выполнения.

Трансляторы можно разделить на две группы, для которых отладка происходит по разному:

1. ИНТЕРПРЕТАТОРЫ - программные средства**,** выполняющие программу по строчкам.

Интерпретаторы обрабатывают операторы по очереди сверху вниз, слева направо, переводят их из текстового вида в машинный код и сразу его выполняют. Таким образом программа отлаживается по строчкам, это удобно, т.к. не надо ждать пока она целиком преобразуется в код и будет запущена на выполнение. Программы написанные с помощью интерпретаторов могут храниться на диске только в виде текстовых файлов (например созданные с помощью "старого" Бейсика) или в специальных форматах (электронные таблицы, СУБД и т.п). Без запуска интерпретатора и загрузки в него таких программ, они не могут быть выполнены.

2. КОМПИЛЯТОРЫ - программные средства **,** которые сначала всю программу переводят в машинные коды, а затем выполняют ее.

- **Достоинство компиляторов** - можно получать выполнимые (.exe) файлы. Недостаток компиляторов проявляется при отладке очень больших программ - надо относительно долго ждать результата выполнения программы, поэтому с этой точки зрения интерпретаторы относительно удобнее.
- **Достоинство интерпретаторов** - более высокая скорость отладки по сравнению с компиляторами, недостаток - невозможность создания выполнимого файла. Поэтому программа, которая выполняется под управлением интерпретатора, после запуска начинает работать позже, чем компилированная, так как тратится время на ее интерпретацию. Однако, если для программы написанной с помощью компилятора исходным является не "exe" файл, отрезок времени между запуском и началом выполнения программы для компилятора будет практически таким же как и для интерпретатора.

Паскаль - компилятор, почти все ЭТ и БД - интерпретаторы. Все интерпретаторы не работают без оболочки, т.е. сначала необходимо загрузить оболочку, потом программу и только после этого ее запускать (не то что выполнимый ".exe" файл!). Только некоторые СУБД, например FoxPro, Clipper, могут работать в двух режимах - при отладке, как интерпретаторы, а когда все отлажено полностью - как компиляторы.

Исторически первые языки программирования были интерпретаторами и только затем появились компиляторы. Однако сейчас бурно развивается язык для создания сетевых приложений Juva (ЯВА), который принципиально - компилятор. Это связано с тем что обычно в сетях могут работать совершенно разные компьютеры, например и IBM совместимые и Aplle (Mac). Для управления их центральными процессорами нужны совершенно различные машинные коды, то есть разные "exe" файлы. На Java программа создается в текстовом виде (кодирование в ASC II коде), но в специальном HTML стан-

дарте. В этом стандарте создаются, например WWW-странички для международной компьютерной сети Internet. С ними работают с помощью программ – просмотрщиков (так называемых “броузеров”). Наиболее популярные сейчас броузеры - “Explouer” и “NetScape”. Explouer фирма MicroSoft включила в состав Windows 98. Броузеры - интерпретаторы HTML кода. Кроме WWW-страничек с помощью Java создают различные сетевые программы, например распределенные базы данных, которыми могут пользоваться одновременно множество абонентов сети.

Типы ошибок в программах и других программных средствах.

Ошибки в программах бывают двух типов - ***синтаксические и семантические.***

1. Синтаксические ошибки - связанны с нарушением синтаксиса, т.е. правил языка программирования, заполнения ЭТ и т.д. Все трансляторы сначала проверяют эти ошибки и указывают места в программе, где они возникли.

2. Семантические ошибки - связанны с содержимым и смыслом программы. Некоторые из них трансляторы вылавливают, например ошибки "типов" - тип переменной не соответствует типу величины, которая записывается в нее.

Но многие семантические ошибки остаются и после успешной трансляции исходного кода. Для их "отлавливания" используют специальные приемы.

- Запускают программу на контрольных примерах - тестах. Например для массивов мы использовали случайные числа и запускали программу много раз с разными их комбинациями исходных данных.
- Отладка программы "по частям". Стремятся при программировании разбить программу на независимые процедуры и функции, которые отлаживаются отдельно.
- Трассировка выполнения программы - пошаговое выполнение с выводом на экран промежуточных значений переменных.
- Временные задержки выполнения отдельных частей для контроля и осмысливания промежуточных результатов.

Тестирующие программы должны быть достаточно полными:

- тестировать все "веточки" алгоритма;
- проверить все возможные комбинации входных данных.

Т.к. все комбинации не перебрать обычно надо обязательно проверить те комбинации, ответ на которые заранее известен и "экстремальные" входные данные, например деление на ноль.

Иногда приходится писать специальную программу, тестирующую другую программу. Чтобы обнаруженные ошибки устранить, часто приходится корректировать техническое задание или информационную (математическую) модель, алгоритм, т.е. начинать все сначала и так много раз. Даже после того как программа отлажена и продана ее приходится часто

"сопровождать" некоторое время, т.к. все ошибки можно найти только при эксплуатации ПО. Невеселая шутка программистов - в любой полностью отлаженной программе всегда содержится еще одна ошибка.

ГЛАВА 25. Принципы структурной алгоритмизации.

Структурированная программа - это программа составленная в соответствии с правилами структурного программирования, понятная программисту, не являющемуся ее разработчиком.

Структурное программирование - это метод программирования, предусматривающий создание понятных, локально простых и удобочитаемых программ.

Характерные особенности структурированных программ:

- модульность;
- ограниченное использование глобальных переменных и повсеместное использование локальных переменных;
- ограничение использования оператора "goto .." (переход к метке) для организации циклов в программе;
- специальное расположение текста программы по правилу "лесенка" для читабельности текста.

Программа называется ***"модульной"***, если она организована в виде совокупности модулей - ***вспомогательных алгоритмов*** - связанных между собой с помощью основной программы.

Вспомогательный алгоритм - произвольный алгоритма, снабженный заголовком, позволяющим вызывать этот алгоритм из других алгоритмов. Вспомогательные алгоритмы имеют ту же структуру что и основной алгоритм за исключением двух особенностей - у них другой заголовок и они не оканчиваются точкой. Переменные, которые описываются внутри вспомогательного алгоритма называются ***локальными.*** Их действие распространяется только на вспомогательный алгоритм. Переменные описанные в основном алгоритме называются ***глобальными.*** Эти переменные могут использоваться и во вспомогательных алгоритмах, но это считается плохим стилем программирования, так как тогда вспомогательный алгоритм теряет свою автономность и его нельзя отлаживать без основного. Если имена локальных переменных совпадают с глобальными, то во вспомогательном алгоритме они имеют приоритет, а в основном - наоборот.

Чтобы воспользоваться вспомогательным алгоритмом из любого другого его снабжают заголовком. После заголовка обычно указывают имена переменных, которые являются ***формальными параметрами*** алгоритма. С их помощью вспомогательный алгоритм обменивается информацией с другими алгоритмами - связываются между собой глобальные локальные переменные. Для вызова вспомогательного алгоритма в основном алгоритме необходимо указать его заголовок и ***фактические параметры*** вызова, после чего управление передается вспомогательному алгоритму. При этом значение фактиче-

ских параметров передается формальным параметрам, которые при необходимости передают их локальным переменным.

После того, как вспомогательный алгоритм выполнит свою задачу, управление вновь перейдет к алгоритму, вызвавшему его. При это возможен обратный обмен информацией через параметры алгоритма.

Правило хорошего стиля программирования - во вспомогательных алгоритмах не должны использовать глобальные переменные.

В Паскале ***вспомогательные алгоритмы - это процедуры (procrdure), функции (function),*** а также ***внешние модули***, которые подключаются к программе в разделе внешних процедур после ключевого слова uses ..., например модуль "crt", с помощью которого организуют работу с текстовыми окнами и с палитрой в текстовом режиме.

Процедура - полностью законченная вспомогательная программа, со структурой почти полностью повторяющей основную программу. Процедура вызывается также как и любой оператор по своему имени из той программы внутри которой она описана.

Отличия процедуры от основной программы.

- Процедура описывается в разделе описаний, после слова "procedure" указывается имя процедуры, а конце описания ставится ";"
- После имени могут быть в круглых скобках описаны возвращаемые или не возвращаемые параметры-переменные (возвращаемые со словом "var")

ПРИМЕР 1

Процедура, вычисляющая корни квадратного уравнения.

Обратите внимание, что имена локальных и глобальных переменных совпадают!

```
uses crt;
var x1,x2,x3:real;              { описание глобальных переменных }
procedure korni(a,b,c:real); { имя процедуры и описание невозвращаемых в основную
                                  программу переменных-параметров }
 var x1,x2:real;                { локальные переменные } begin
 if sqr(b)-4*a*c > 0 then
     begin
        x1:=(-b+sqrt(sqr(b)-4*a*c))/(2*a);
        x2:=(-b-sqrt(sqr(b)-4*a*c))/(2*a);
         writeLn('x1',x1:5:2,'x2':5,x2:5:2)
     end
     else   writeLn('Корней нет!');
end; {конец процедуры}

begin
 clrScr;
 write('a='); readLn(x1); write('b='); readLn(x2); write('c');
```

```
readLn(x3); korni(x1,x2,x3);
repeat until KeyPressed;
end.
```

ПРИМЕР 2

Использование процедуры возвращающей в основную программу значение параметров-переменных. Эта процедура меняет местами значения двух переменных.

```
uses crt;
var x1,x2: real;                                  {глобальные переменные}

procedure change(var a,b:real); {имя процедуры и описание возвращаемых}
                                  { параметров - переменных }
var c: real;                      {локальная переменная}

begin
   c:=a; a:=b; b:=c;
end;                              {конец описания процедуры}

begin
 readLn(x1,x2);
 change(x1,x2);
 writeLn(x1:5,x2:5);
repeat until KeyPressed;
end.
```

NB!!! Обратите внимание, что значения параметров-переменных "a", "b" передаются наружу из процедуры во внешнюю программу (в переменные "x1" и "x2") за счет использования в описании этих параметров слова "var" в заголовке процедуры.

Функция заданная пользователем - вспомогательный алгоритм находящийся в разделе описаний, который вызывается из основной программы по своему имени, причем обязательно из какого-нибудь оператора. При вызове функции в нее передаются значения параметров-переменных, вычисляется значение функции и в то место основной программы откуда она была вызвана возвращается вычисленное значение функции. Возвращаемое значение соответствует типу функции.

Эти функции традиционно называют "заданными пользователем" (вернее программистом разработчиком программы), чтобы отличить их от "стандартных функций", например sin.., cos.., int.., round.., rnd.. и прочих, "встроенных" в Паскаль.

ПРИМЕР 3

Использование функции возводящей заданное число ("a") в заданную

степень ("b"):

```
uses crt;
var x: real; k: integer; {глобальные переменные}

function st(a:real; b:integer):real; {имя функции, описание ее параметров - переменных и в самом конце описание типа значения функции, которое возвращается в то место программы откуда эта функция была вызвана }
 var i: integer;                     {локальные переменные}
     d: real;
  begin
    d:=a;

    For i:=1 to b-1 do d:=d*a;

    st:=d; {обязательная строка в каждой функции - ей надо присвоить значение
            которое возвращается туда откуда функция была вызвана}
  end;    {окончание описания функции}

begin
  readLn(x,k); writeLn(st(x,k)); { вызов функции "st" внутри оператора "writeLn"}
                                  repeat until KeyPressed;
end.
```

NB!!! Обратите внимание на отличия функции от процедуры:

- Функция всегда "вызывается" из основной программы находясь внутри какого-нибудь оператора. Процедура всегда "гуляет сама по себе", т.е. вызывается также как и любой оператор по имени. Процедура не имеет типа.
- Функция всегда "возвращает" в то место программы откуда была вызвана свое значение, поэтому функция всегда имеет тип. Тип функции указывается в конце ее заголовка после ":".
- Функция возвращает в основную программу, в то место из которого была вызвана (или, говорят, в тот оператор из которого она была вызвана) только одно значение. Процедура, напротив, может возвратить в основную программу (в глобальные переменные) много значений или вообще ничего не возвращать.
- Внутри раздела операторов функции всегда должен быть хотя бы один оператор присваивания, задающий значение функции, которое возвращается в основную программу. При этом после имени функции не надо указывать значения ее параметров-переменных.

В "профессиональной" программе раздел операторов очень короткий, а целые "километры" листинга занимают процедуры и функции заданные поль-

зователем (вернее программистом). Такую программу можно писать и отлаживать по кускам (даже разным людям!). Однако для этого надо не забывать о еще одной важной особенности структурированных программ - о различиях между локальными и глобальными переменными.

ГЛОБАЛЬНЫЕ И ЛОКАЛЬНЫЕ ПЕРЕМЕННЫЕ. ОБМЕН ЗНАЧЕНИЯМИ МЕЖДУ НИМИ ВО ВСПОМОГАТЕЛЬНЫХ АЛГОРИТМАХ.

Глобальными называются переменные описанные в основной программе. К этим переменным имеется доступ (т.е. к ним можно обратиться или изменить их значение), как в основной программе, так и внутри любой процедуры или функции. Однако, если имя глобальной переменной внутри процедуры или функции совпадает с именем локальной переменной данной процедуры или функции, что внутри этого вспомогательного алгоритма приоритет отдается локальной переменной, а снаружи - глобальной.

Локальными называются переменные описанные внутри процедур или функций. Они действуют только внутри данной процедуры или функции, доступа к ним напрямую из основной программы нет.

Ранее, когда ЯВУ еще не были "структурированными", понятия "локальная переменная" не существовало. Поэтому в достаточно большой программе было очень просто запутаться (особенно если ее писали несколько человек) и ее было очень тяжело отладить. С появлением локальных переменных появилась возможность простой отладки программы "по кускам", а для "стыковки" готовых кусков стали использовать "параметры-переменные".

Параметры-переменные - это переменные описанные в круглых скобках после имени переменных. Значение параметров-переменных передается внутрь функции или процедуры. В процедурах, если это необходимо значения параметров-переменных может возвращаться в основную программу (за счет использования в описании этих параметров слова "var" в заголовке процедуры, пример процедуры change смотри выше).

В функциях параметры обязательны. В основную программу возвращается значение функции.

Процедуры могут параметров не иметь, например, простейшая процедура GoodBay очищающая экран и выводящая надпись "ПРОЩАЙ!":

```
Uses Crt;
 Procedure GoodBay;
  begin
     Window(35,19,45,21);
     TextBackground(0);
     ClrScr;
     Window(34,18,44,20);
      TextBackground(red);
      TextColor(15);
```

```
    ClrScr;
    GoToXY(2,2);
    Wrire('Good bay!!!')
end;
```

После разработки структурированных языков программирования, в которых появилось независимые локальные переменные, у программистов появилась возможность создавать так называемые "***рекурсивные алгоритмы***". Эти алгоритмы работают по принципу "вызов самого себя", таким образом зацикливая программу. Каждый раз такой вызов осуществляется с новыми значениями параметров-переменных, а значение которое должно вернуться в основную программу запоминается в стековой памяти. Рекурсивные алгоритмы работают по принципу "***сколько раз вошел - столько раз и вышел***". В этих алгоритмах очень важно организовать правильный выход. Любые задачи можно решать как с помощью рекурсивных алгоритмов как и "простыми" циклами, но часто оказывается, что если надо сделать много "переборов" данных это эффективнее (быстрее и "элегантнее") это получается с помощью рекурсии.

Пример рекурсивного алгоритма приведен в главе "Циклические алгоритмы".

Модульное проектирование программы

предполагает выполнение следующих действий:

- Глобальная задача разбивается на ряд более простых задач (модулей), выполнение которых под силу одному программисту.
- Для каждой простой задачи (модуля) разрабатывается свой алгоритм и пишется программа, которую обычно оформляют в виде вспомогательного алгоритма (процедуры или функции) или нескольких вспомогательных алгоритмов, которые составляют из себя библиотеку.
- Главный алгоритм может вызывать любой другой вспомогательный алгоритм из библиотеки для выполнения конкретной задачи и служит для объединения модулей в единую программу.

Пошаговая детализация программы

предполагает выполнение следующих действий:

- отдельная отладка и тестирование каждого модуля.
- компоновка главного алгоритма.
- отладка и тестирование основного алгоритма "по шагам" для проверки правильности "стыковки" модулей. При этом проверяют правильно или нет модули обмениваются значениями параметров-переменных и как значения функций и локальных переменных возвращаются в основную программу.

В современных алгоритмических языках существуют обширные библиотеки вспомогательных алгоритмов для решения различного класса задач: нахождение значений тригонометрических функций, решение систем уравнений, оформление программ с помощью оконного интерфейса и т.д. Таким об-

разом можно построить программу из отдельных модулей, используя библиотеку в виде своеобразного конструктора. Недостающие модули можно проработать и включить в библиотеку, а затем использовать во всех других задачах.

Примечание: отдельные модули при отладке программы можно было заменить "заглушкой" на то время, пока он не был проработан. Кроме этого можно было поручить написание модулей разным людям, что могло существенно сократить сроки создания программы.

Виды технологий программирования

- ***Нисходящее программирование*** базируется на идее постепенного разбиения исходной задачи на ряд подзадач. Сначала исходная задача представляется в виде набора подзадач и строится программа, в которой эти подзадачи выступают как некоторые именованные процедуры, к которым можно организовать обращение. Те подзадачи, которые еще не проработаны временно заменяются программистскими ***"заглушками"***, что позволяет иметь действующий вариант программы, годный для первого шага отладки и поиска улучшенных вариантов. После этого по отношению к подзадачам применяется такой же прием.
- ***Восходящее программирование*** основано на противоположном процессе. Сначала пишутся и отлаживаются программы (процедуры) самого нижнего уровня. Затем из них собираются более крупные блоки. Эта последовательность действий заканчивается тогда, когда вся программа будет собрана и отлажена.