

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»**

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ
ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ**

Б1.Б.14 Информатика

Направление подготовки 38.03.01 Экономика

Профиль образовательной программы Бухгалтерский учет, анализ и аудит

Форма обучения очная

1. КОНСПЕКТ ЛЕКЦИЙ

1.1 Лекция № 1 (2 часа).

Тема: «Введение. Основные понятия информатики»

1.1.1 Вопросы лекции:

1. Этапы информатизации общества
2. Цель и задачи информатики. Основные понятия
3. Структура информатики
4. Социальные, правовые и этические аспекты информатики

1.1.2 Краткое содержание вопросов:

1. Этапы информатизации общества

Деятельность отдельных людей, групп, коллективов и организаций сейчас все в большей степени начинает зависеть от их информированности и способности эффективно использовать имеющуюся информацию. Прежде чем предпринять какие-то действия, необходимо провести большую работу по сбору и переработке информации, ее осмыслению и анализу. Отыскание рациональных решений в любой сфере требует обработки больших объемов информации, что подчас невозможно без привлечения специальных технических средств.

Возрастание объема информации особенно стало заметно в середине XX в. Лавинообразный поток информации хлынул на человека, не давая ему возможности воспринять эту информацию в полной мере. В ежедневно появляющемся новом потоке информации ориентироваться становилось все труднее. Подчас выгоднее стало создавать новый материальный или интеллектуальный продукт, нежели вести розыск аналога, сделанного ранее. Образование больших потоков информации обуславливается:

- чрезвычайно быстрым ростом числа документов, отчетов, диссертаций, докладов и т.п., в которых излагаются результаты научных исследований и опытно-конструкторских работ;
- постоянно увеличивающимся числом периодических изданий по разным областям человеческой деятельности;
- появляются противоречия между ограниченными возможностями человека по восприятию и переработке информации и существующими мощными потоками и массивами хранящейся информации;
- существует большое количество избыточной информации, которая затрудняет восприятие полезной для потребителя информации;
- возникают определенные экономические, политические и другие социальные барьеры, которые препятствуют распространению информации. Например, по причине соблюдения секретности часто необходимой информацией не могут воспользоваться работники разных ведомств.

Эти причины породили весьма парадоксальную ситуацию – в мире накоплен громадный информационный потенциал, но люди не могут им воспользоваться в полном объеме в силу ограниченности своих возможностей. Информационный кризис поставил общество перед необходимостью поиска путей выхода из создавшегося положения. Внедрение ЭВМ, современных средств переработки и передачи информации в различные сферы деятельности послужило началом нового эволюционного процесса, называемого

информатизацией, в развитии человеческого общества, находящегося на этапе индустриального развития.

Информатизация общества – организованный социально-экономический и научно-технический процесс создания оптимальных условий для удовлетворения информационных потребностей и реализации прав граждан, органов государственной власти, органов местного самоуправления, организаций, общественных объединений на основе формирования и использования информационных ресурсов.

Рассмотрим этот процесс более подробно.

История развития информатизации началась в США с 60-х гг., затем с 70-х гг. – в Японии и с конца 70-х – в Западной Европе.

Современное материальное производство и другие сферы деятельности все больше нуждаются в информационном обслуживании, переработке огромного количества информации. Универсальным техническим средством обработки любой информации является компьютер, который играет роль усилителя интеллектуальных возможностей человека и общества в целом, а коммуникационные средства, использующие компьютеры, служат для связи и передачи информации. Появление и развитие компьютеров – это необходимая составляющая процесса информатизации общества.

Информатизация общества является одной из закономерностей современного социального прогресса. Этот термин все настойчивее вытесняет широко используемый до недавнего времени термин "компьютеризация общества". При внешней схожести этих понятий они имеют существенное различие.

При компьютеризации общества основное внимание уделяется развитию и внедрению технической базы компьютеров, обеспечивающих оперативное получение результатов переработки информации и ее накопление.

При информатизации общества основное внимание уделяется комплексу мер, направленных на обеспечение полного использования достоверного, исчерпывающего и своевременного знания во всех видах человеческой деятельности.

Таким образом, "информатизация общества" является более широким понятием, чем "компьютеризация общества", и направлена на скорейшее овладение информацией для удовлетворения своих потребностей. В понятии "информатизация общества" акцент надо делать не столько на технических средствах, сколько на сущности и цели социально-технического прогресса. Компьютеры являются базовой технической составляющей процесса информатизации общества.

Информатизация на базе внедрения компьютерных и телекоммуникационных технологий является реакцией общества на потребность в существенном увеличении производительности труда в информационном секторе общественного производства, где сосредоточено более половины трудоспособного населения. Так, например, в информационной сфере США занято более 60% трудоспособного населения, в СНГ – около 40%.

На основе вышесказанного можно сделать вывод: во второй половине XX века человечество вступило в новый этап своего развития. В этот период начался переход от **индустриального** общества к **информационному**. Информационное общество имеет следующие основные признаки:

1) Большинство работающих в информационном обществе (около 80%) занято в информационной сфере, т.е. сфере производства информации и информационных услуг.

2) Обеспечены техническая, технологическая и правовая возможности доступа любому члену общества практически в любой точке территории и в приемлемое время к нужной ему информации.

3) Информация становится важнейшим стратегическим ресурсом общества и занимает ключевое место в экономике, образовании и культуре.

Следовательно, **информационное общество** - это общество, в котором большая часть населения занята получением, обработкой, передачей и хранением информации.

Процесс, обеспечивающий переход от индустриального общества к информационному, получил название **информатизации**.

Информатизация общества – повсеместное внедрение комплекса мер, направленных на обеспечение полного и своевременного использования достоверной информации, и зависит от степени освоения и развития новых информационных технологий.

Неизбежность информатизации общества обусловлена резким возрастанием роли и значения информации. Информационное общество характеризуется высокоразвитой информационной сферой, которая включает деятельность человека по созданию, переработке, хранению, передаче и накоплению информации.

Процесс перехода от индустриального общества к информационному происходит одновременно в различных странах, он характеризуется также разными темпами развития. Первыми на этот путь встали в конце 50-х – начале 60-х годов XX века США, Япония и страны Западной Европы. В этих государствах, начиная с 60 – 70-х годов, проводится политика повсеместной информатизации всех сфер деятельности человека. В СССР в 1989 г. Разработана Концепция информатизации общества. По мнению специалистов, любая страна, насколько бы индустриально развитой она ни была, перейдет в разряд стран третьего мира, если опоздает с информатизацией.

Поэтому умение правильно использовать компьютерные технологии в своей профессиональной работе становится неотъемлемым показателем квалификации современного специалиста в любой области.

2. Цель и задачи информатики. Основные понятия

Научным фундаментом процесса информатизации общества является научная дисциплина – **информатика**.

Термин «информатика» был заимствован из французского языка и обозначал название области, связанной с автоматизированной обработкой информации с помощью электронных вычислительных машин. Именно развитие компьютерной техники способствовало выделению информатики в самостоятельную область человеческой деятельности.

Информатика – это наука, занимающаяся исследованием форм и методов сбора, хранения, накопления и передачи информации с помощью компьютерных и других технических средств.

Это дисциплина комплексная, тесно связанная с другими науками, у которых есть общий объект исследования – информация.

Задачами информатики являются:

1. Разработка и производство современных средств вычислительной техники.
2. Проектирование и внедрение прогрессивных технологий обработки информации.
3. Дальнейшая информатизация общества и повышение его информационной культуры.

Целью курса информатики является:

4. Помочь всем желающим (непрофессиональным пользователям) овладеть основами компьютерной грамотности.
5. Способствовать развитию логического мышления.
6. Познакомиться с аппаратными и программными средствами компьютера.
7. Изучить основы алгоритмизации и программирования.

Понятие «информатика» неразрывно связано с такими понятиями, как «информационная система» и «информационная технология». Информатика изучает свойства, структуру и функции информационных систем, а также происходящие в них информационные процессы.

Под **информационной системой** понимают – взаимосвязанную совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели.

Подавляющее большинство современных информационных систем являются автоматизированными. Компьютеры, оснащенные специализированными программными средствами, являются технической базой и инструментом для информационных систем. Информационная система немыслима без персонала, взаимодействующего с компьютерами и телекоммуникациями.

Информационная технология – это процесс, включающий совокупность способов сбора, хранения, обработки и передачи информации на основе применения средств вычислительной техники.

Каждая ИТ реализуется в рамках конкретной информационной системы. Таким образом ИС является средой обитания для соответствующей ИТ. В качестве инструментария ИТ выступают такие программные средства, как: текстовый редактор, электронные таблицы, системы управления базами данных, издательские системы и т.д.

Информатика тесно связана с кибернетикой, но не тождественна ей. **Кибернетика** – это наука об общих принципах управления в различных системах: технических, биологических, социальных и др. независимо от наличия или отсутствия компьютеров.

Информатика как научная дисциплина занимается изучением информационных процессов. **Информационными процессами** – называются процессы, связанные с получением, хранением, преобразованием и передачей информации.

Количество информации в современном обществе стремительно нарастает, человек оказывается погруженным в море информации. Чтобы быть востребованным, он должен обладать **информационной культурой**, т.е. знаниями и умениями в области информационных технологий, а также знать юридические и этические нормы, действующие в этой сфере.

Информатика как наука стала развиваться с середины нашего столетия, что связано с появлением ЭВМ и начинающейся компьютерной революцией.

Появление вычислительных машин в 50-е годы создало для информатики необходимую ей аппаратную поддержку, или, иначе говоря, благоприятную среду для ее развития как науки. Всю историю информатики принято разбивать на два больших этапа: **предыстория и история**.

Предыстория информатики такая же древняя, как и история развития человеческого общества. В предыстории, в свою очередь, выделяют ряд этапов. Каждый из этих этапов характеризуется по сравнению с предыдущим резким возрастанием возможностей хранения, передачи и обработки информации.

Начальный этап предыстории – появление устной речи. Освоение природы человеком началось, прежде всего, с присвоения готовых продуктов (растений, животных, плодов, тепла солнца и т.д.). Затем человек научился не только брать готовое у природы, но и воздействовать на нее, создавая новые вещества, новые виды энергии, приобретая новые знания. Появление устной речи позволило первобытному человеку делиться накопленными знаниями со своими соплеменниками. Таким образом, членораздельная речь, язык стал специфическим социальным средством хранения и передачи накопленных знаний.

По мере накопления знаний людей об окружающем мире возникла необходимость использования искусственных способов их фиксации. Так появилась письменность – второй этап предыстории. При этом, прежде всего, резко возросли (по сравнению с предыдущим этапом) возможности по хранению информации. Человек получил искусственную внешнюю память. Организация почтовых служб позволила использовать письменность как средство передачи информации. Кроме того, возникновение письменности было необходимым условием для начала развития наук (вспомним, например, Древнюю Грецию).

Третий этап – книгопечатание. Книгопечатание можно смело назвать первой информационной технологией. Воспроизведение информации было поставлено на поток, на промышленную основу. По сравнению с предыдущим этот этап не столько увеличил возможности по хранению, сколько повысил доступность информации и точность ее воспроизведения.

Четвертый и последний этап предыстории связан с успехами точных наук (прежде всего математики и физики) и начинающейся в то время научно-технической революцией. Этот этап характеризуется возникновением таких мощных средств связи, как радио, телефон и телеграф, к которым по завершению этапа добавилось и телевидение. Кроме средств связи появились новые возможности по получению и хранению информации – фотография и кино. К ним также следует добавить разработку методов записи информации на магнитные носители (магнитные ленты и диски).

В XVII веке появились первые счетные машины на зубчатых передачах, прообразы арифмометра, созданного в 70-х годах XIX века петербургским изобретателем В.Т.Однером и прослужившего человечеству почти столетие. В этот же период английский математик Чарльз Бэббидж попытался построить универсальное вычислительное устройство – аналитическую машину, которая должна была выполнять вычисления без участия человека. Попытка оказалась неудачной, но в этой машине были заложены основные идеи, используя которые в 1943 году Говард Эйкеп смог построить первую электромеханическую вычислительную машину "Марк-1". С этого же времени начинается и электронная эпоха развития информатики.

Возникновение информатики как науки, начало ее **истории** принято связывать с разработкой первых электронных вычислительных машин (ЭВМ) в середине 40-х годов. Для такой "привязки" имеется несколько причин. Во-первых, сам термин "информатика" появился на свет благодаря развитию вычислительной техники, и поначалу под ним понималась наука о вычислениях (первые ЭВМ большей частью использовались для проведения числовых расчетов). Во-вторых, выделению информатики в отдельную науку способствовало такое важное свойство современной вычислительной техники, как единая форма представления обрабатываемой и хранимой информации. Вся информация, вне зависимости от ее вида, хранится и обрабатывается на ЭВМ в двоичной форме. Так получилось, что компьютер в одной системе объединил хранение и обработку числовой, текстовой (символьной) и аудиовизуальной (звук, изображение) информации. В этом состояла иницирующая роль вычислительной техники при возникновении и оформлении новой науки.

Поколения ЭВМ

В соответствии с элементной базой и уровнем развития программных средств выделяют четыре реальных поколения ЭВМ, краткая характеристика которых приведена в таблице:

Параметры сравнения	Поколения ЭВМ			
	первое	второе	третье	четвертое
Период времени	1946 - 1959	1960 - 1969	1970 - 1979	с 1980 г.
Элементная база (для УУ, АЛУ)	Электронные (или электрические) лампы	Полупроводники (транзисторы)	Интегральные схемы	Большие интегральные схемы (БИС)
Основной тип ЭВМ	Большие		Малые (мини)	Микро
Основные устройства ввода	Пульт, перфокарточный, перфоленточный ввод	Добавился алфавитно-цифровой дисплей, клавиатура	Алфавитно-цифровой дисплей, клавиатура	Цветной графический дисплей, сканер, клавиатура
Основные устройства вывода	Алфавитно-цифровое печатающее устройство (АЦПУ), перфоленточный вывод		Графопостроитель, принтер	
Внешняя память	Магнитные ленты, барабаны, перфоленты, перфокарты	Добавился магнитный диск	Перфоленты, магнитный диск	Магнитные и оптические диски
Ключевые решения в ПО	Универсальные языки программирования, трансляторы	Пакетные операционные системы, оптимизирующие трансляторы	Интерактивные операционные системы, структурированные языки программирования	Дружественность ПО, сетевые операционные системы
Режим работы ЭВМ	Однопрограммный	Пакетный	Разделения времени	Персональная работа и сетевая обработка данных
Цель использования ЭВМ	Научно-технические расчеты	Технические и экономические расчеты	Управление и экономические расчеты	Телекоммуникации, информационное обслуживание

3. Структура информатики

По своей структуре информатика представляет собой единство трех взаимосвязанных частей – технические средства, программные средства, алгоритмические средства. В свою очередь, информатику как в целом, так и каждую часть обычно рассматривают с разных позиций: как отрасль народного хозяйства, как теоретическую науку, как прикладную дисциплину.

Информатика как отрасль народного хозяйства обеспечивает все другие отрасли необходимыми информационными ресурсами.

Информационные ресурсы – информация, используемая на производстве, в технике, управлении обществом, специально организованная и обрабатываемая на ЭВМ. Информационные ресурсы в объеме страны – национальные информационные ресурсы. Информационные ресурсы страны определяют ее научно-технический прогресс, научный потенциал, экономическую и стратегическую мощь.

Теоретическая информатика рассматривает все аспекты разработки автоматизированных информационных систем: их проектирования, создания и использования не только с формально-технической, но и с содержательной стороны, а также комплекс экономического, политического и культурного воздействия на социальную динамику. В орбиту анализа теоретической информатики попадают и традиционные системы преобразования информации и распространения знаний: средства и системы массовой информации, система лекционной пропаганды, кино, театры, справочные службы и т.д. При этом информатика рассматривает их с позиций получения

и использования информационного ресурса, форм и способов воздействия указанных систем на общественный прогресс.

Теоретическая информатика изучает информационный ресурс, законы его функционирования и использования как движущей силы социального прогресса, а также общие, фундаментальные проблемы информационных технологий как исторического феномена, выводящего общество на новую ступень развития.

Решающее значение для рождения теоретической информатики имеет появление информационных технологий высшего уровня, основанных на искусственном интеллекте.

Прикладная информатика изучает конкретные разновидности информационных технологий, которые формируются с помощью специальных информационных систем (управленческих, медицинских, обучающих, военных и др.). Информационные технологии в различных отраслях, имея общие черты, в то же время существенно различаются между собой. Разные операции и процедуры, различное оборудование, специализация критериев и показателей, даже разные носители информации, – все это становится объектом изучения конкретных функциональных и отраслевых информатик. Так рождаются ветви прикладной информатики, обслуживающие создание проектирующих систем, экспертных систем, управляющих и других функциональных систем. Поэтому наряду с теоретической информатикой развиваются ее конкретные ветви: экономическая информатика, военная информатика, медицинская информатика и многие другие.

4. Социальные, правовые и этические аспекты информатики

Социальные аспекты информатики

Термин “социальные аспекты” применительно к большей части наук. Однако, информатика – не только наука.

Мало какие факторы так влияют на социальную сферу обществ как информатизация.

Информатизация общества – процесс проникновения информационных технологий во все сферы жизни и деятельности общества. Она сильнее всего влияет на структуру экономики ведущих в экономическом отношении стран. В числе их лидирующих отраслей промышленности традиционные добывающие и обрабатывающие отрасли оттеснены максимально наукоемкими производствами электроники, средств связи и вычислительной техники. В этих странах постоянно растут капиталовложения в научные исследования, включая фундаментальные науки. Темпы развития сферы высоких технологий и уровень прибылей в ней превышают в 5-10 раз темпы развития традиционных отраслей производства. Такая политика имеет и социальные последствия – увеличение потребности в высокообразованных специалистах и связанный с этим прогресс системы высшего образования. Информатизация меняет и облик традиционных отраслей промышленности и сельского хозяйства. Промышленные роботы, управляемые ЭВМ, станки с ЧПУ стали обычным оборудованием. Новейшие технологии в сельскохозяйственном производстве не только увеличивают производительность труда, но и облегчают его, вовлекают более образованных людей.

Казалось бы, компьютеризация и информационные технологии несут в мир одну лишь благодать, но социальная сфера столь сложна, что последствия любого, даже гораздо менее глобального процесса, редко бывают однозначными. Рассмотрим, например, такие социальные последствия информатизации как рост производительности труда, интенсификацию труда, изменение условий труда. Все это, с одной стороны, улучшает условия жизни многих людей, повышает степень материального и интеллектуального комфорта, стимулирует рост числа высокообразованных людей, а с другой – является источником повышенной социальной напряженности. Например, появление на производстве промышленных роботов ведет к полному изменению

технологии, которая перестает быть ориентированной на человека. Тем самым меняется номенклатура профессий. Значительная часть людей вынуждена менять либо специальность, либо место работы – рост миграции населения характерен для большинства развитых стран. Государство и частные фирмы поддерживают систему повышения квалификации и переподготовки, но не все люди справляются с сопутствующим стрессом. Одним словом, жизнь в “информационном обществе” легче, по-видимому, не становится, а вот то, что она значительно меняется – несомненно

Правовые аспекты информатики

Деятельность программистов и других специалистов, работающих в сфере информатики, все чаще выступает в качестве объекта правового регулирования. Некоторые действия при этом могут быть квалифицированы как правонарушения (преступления).

Правовое сознание в целом, а в области информатики особенно, в нашем обществе находится на низком уровне. Все ли знают ответы на следующие вопросы:

- можно ли, не копируя купленную программу, предоставить возможность пользоваться ею другому лицу;
- кому принадлежит авторское право на программу, созданную студентом в ходе выполнения дипломной работы;
- можно ли скопировать купленную программу для себя самого, чтобы иметь резервную копию;
- можно ли декомпилировать программу, чтобы разобраться в ее деталях или исправить ошибки;
- в чем состоит разница между авторским и имущественным правом.

Вопросов, подобных этим, возникает множество, но остановимся на правовом регулировании в области информатики в России. К 1992 году был принят Закон Российской Федерации “О ПРАВОВОЙ ОХРАНЕ ПРОГРАММ ДЛЯ ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН И БАЗ ДАННЫХ”, содержащий обширный план приведения российского законодательства в сфере информатики в соответствие с мировой практикой. Действие этого Закона распространяется на отношения, связанные с созданием и использованием программ для ЭВМ и баз данных. Также предусматривалось внести изменения и дополнения в Гражданский кодекс РФ, в Уголовный кодекс РФ, другие законодательные акты, связанные с вопросами правовой охраны программ для электронных вычислительных машин и баз данных, привести решения Правительства РФ в соответствие с Законом, обеспечить пересмотр и отмену государственными ведомствами и другими организациями РФ их нормативных актов, противоречащих указанному Закону, обеспечить принятие нормативных актов в соответствии с указанным Законом и т.д. Главное содержание данного Закона – юридическое определение понятий, связанных с авторством и распространением компьютерных программ и баз данных, таких как Авторство, Адаптация, База данных, Воспроизведение, Декомпилирование. Использование, Модификация и т.д., а также установление прав, возникающих при создании программ и баз данных – авторских, имущественных, на передачу, защиту, регистрацию, неприкосновенность и т.д.

Авторское право распространяется на любые программы для ЭВМ и базы данных, представленные в объективной форме, независимо от их материального носителя, назначения и достоинства. Авторское право распространяется на программы для ЭВМ и базы данных, являющиеся результатом творческой деятельности автора. Творческий характер деятельности автора предполагается до тех пор, пока не доказано обратное.

Предоставляемая настоящим Законом правовая охрана распространяется на все виды программ для ЭВМ, которые могут быть выражены на любом языке и в любой форме, и на базы данных, представляющие собой результат творческого труда по подбору и организации данных. Предоставляемая правовая охрана не распространяется на идеи и принципы, лежащие в основе программы для ЭВМ и базы данных или какого-либо их

элемента, в том числе идеи и принципы организации интерфейса и алгоритма, а также языки программирования.

Авторское право на программы для ЭВМ и базы данных вступает в силу их создания. Авторское право на базу данных признается при условии соблюдения авторского права на каждое из произведений, включенных в базу данных. Автором программы для ЭВМ и базы данных признается физическое лицо, в результате творческой деятельности которого они созданы.

Этические аспекты информатики

Этика – система норм нравственного поведения человека.

Далеко не все правила, регламентирующие деятельность в сфере информатики, можно свести в правовым нормам. Очень многое определяется соблюдением неписаных правил поведения для тех, кто причастен к миру компьютеров. Впрочем, в этом отношении информатика ничуть не отличается от любой другой сферы деятельности человека в обществе.

Морально-этические нормы в среде информатиков отличаются от этики повседневной жизни несколько большей открытостью, альтруизмом. Большинство нынешних специалистов-информатиков сформировались и приобрели свои знания и квалификацию благодаря бескорыстным консультациям и содействию других специалистов. Очевидно, поэтому они готовы оказать бескорыстную помощь, дать совет или консультацию, предоставить компьютер для выполнения каких-либо манипуляций с дискетами и т.д. Ярким примером особой психологической атмосферы в среде информатиков является расширяющееся международное движение программистов, предоставляющих созданные ими программные средства для свободного распространения.

1.2 Лекция № 2 (2 часа).

Тема: «Понятие информации»

1.2.1 Вопросы лекции:

1. Место и роль понятия «информация» в курсе информатики. Методы получения информации
2. Виды, свойства и формы представления информации
3. Классификация информации
4. Кодирование информации. Единицы измерения информации

1.2.2 Краткое содержание вопросов:

1. Место и роль понятия «информация» в курсе информатики. Методы получения информации

В любой науке основные понятия определить достаточно сложно. Так и понятие «информация» невозможно определить через другие, более «простые» понятия. В каждой науке оно связано с различными системами понятий. Например, в определении понятия «информация» необходимо отразить, что она:

1. существует в неживой природе;
2. существует в биологических системах;
3. не всегда может быть выражена словами;
4. возникает в процессе общения;

5. хранится, обрабатывается, передается, и т.д.

В зависимости от области знания существуют различные подходы к определению понятия «информация».

Впервые как научное понятие термин «информация» стал применяться в теории журналистики в 30-х годах XX века, хотя в исследованиях по библиотечному делу он появился еще раньше. Под информацией понимались различные сведения, сообщения. Что соответствует переводу с латинского языка *informatio* – сведение, разъяснение, ознакомление.

В физике понятие информация рассматривается как антиэнтропия или энтропия с обратным знаком. Поскольку мерой беспорядка термодинамической системы является энтропия системы, то информация (антиэнтропия) является мерой упорядоченности и сложности системы. По мере увеличения сложности системы величина энтропии уменьшается, и величина информации увеличивается. Процесс увеличения информации характерен для открытых, обменивающихся веществом и энергией с окружающей средой, саморазвивающихся систем живой природы (белковых молекул, организмов, популяций животных и т.д.).

В неживой природе понятие информация связано с понятием отражения, отображения. В некоторых физических и химических теориях информация определяется как отраженное многообразие. Отражение заключается в таком изменении одного материального объекта под воздействием другого, при котором все особенности отражаемого объекта каким-либо образом воспроизводятся отражающим объектом. В процессе отражения и происходит передача информации. Т.е. информация - это результат отражения. В соответствии с этим взглядом информация существовала и будет существовать вечно, она содержится во всех элементах и системах материального мира. Информация, наряду с веществом и энергией, является неотъемлемым свойством материи.

Под информацией в технике понимают сообщение, передаваемое с помощью знаков и символов. В теории связи, например, под информацией принято понимать любую последовательность символов, не учитывая их смысл. В основанной американским ученым Клодом Шенноном математической теории информации под информацией понимались не любые сведения, а лишь те, которые снимают полностью или уменьшают существующую до их получения неопределенность (неизвестность). Каждому сигналу в теории Шеннона соответствует вероятность его появления. Например, при передаче текста телеграммы вероятность появления буквы «т» равна $1/33$. Чем меньше вероятность появления того или иного сигнала, тем больше информации он несет для потребителя. В обыденном понимании, чем неожиданнее новость, тем больше ее информативность.

Математическая теория информации не охватывает всего богатства содержания понятия информация, поскольку отвлекается от содержательной (смысловой, семантической) стороны сообщения. С точки зрения этой теории фраза из 100 слов, взятая из газеты, пьесы Шекспира или теории относительности Эйнштейна имеют приблизительно одинаковое количество информации.

Наш соотечественник математик Ю.А. Шрейдер оценивал информацию по увеличению объема знаний у человека под воздействием информационного сообщения. Академик А.А. Харкевич измерял содержательность сообщения по увеличению вероятности достижения цели после получения информации человеком или машиной. Таким образом, под информацией в семантической теории понимают сведения обладающие новизной.

В кибернетике – науке об управлении в живых, неживых и искусственных системах – понятие информации связывают воедино с понятием управления (Норберт Винер, Б.Н. Петров). Жизнедеятельность любого организма или нормальное функционирование технического устройства зависит от процессов управления, благодаря которым поддерживаются в необходимых пределах значения их параметров. Процессы управления включают в себя получение, хранение, преобразование и передачу информации.

Информация является обозначением содержания, полученного из внешнего мира в процессе приспособления к нему наших чувств. Информацию составляет та часть знания, которая используется для ориентирования, принятия решений, активного действия, управления, т.е. в целях сохранения, совершенствования и развития системы.

Данная концепция отрицает существование информации в неживой природе, не дает ответа на вопросы: являются ли информацией неиспользованные знания, являются ли информацией неосмысленная информация?

Для преодоления этих противоречий академик В.П. Афанасьев ввел понятие информационных данных. Информационные данные – это всякие сведения, сообщения, знания, которые могут храниться, перерабатываться, передаваться, но характер информации они приобретут лишь тогда, когда получат содержание и форму пригодную для управления и используются в управлении.

Дальнейшим развитием математического подхода к феномену информация послужили работы Р. Карнапа, И. Бар-Хиллела, А.Н. Колмогорова и многие др.

В этих теориях понятие информации не связано с содержанием сообщений, передаваемых по каналу связи. Информация – абстрактная величина, не существующая в физической реальности, подобно тому, как не существует мнимое число или не имеющая линейных размеров материальная точка.

В биологии, которая изучает живую природу, понятие «информация» связано с целесообразным поведением живых организмов. Такое поведение строится на основе получения и использования организмом информации об окружающей среде.

Понятие информация используется в связи с исследованием механизмов наследственности. В генетике сформулировано понятие генетической информации, которое определяется как программа (код) биосинтеза белков, представленных цепочками ДНК. Реализуется эта информация в ходе развития особи. Последнее обстоятельство позволило проводить научные эксперименты по клонированию, т.е. созданию точных копий организмов из одной клетки.

В социальных науках (социологии, психологии, политологии и др.) под информацией понимают сведения, данные, понятия, отраженные в нашем сознании и изменяющие наши представления о реальном мире. Эту информацию, передающуюся в человеческом обществе и участвующую в формировании общественного сознания, называют социальной информацией.

Под информацией в документалистике понимают все то, что так или иначе зафиксировано в знаковой форме в виде документов.

С точки зрения индивидуального человеческого сознания информация – это то, что поступает в наш мозг из многих источников в разных формах и, взаимодействуя там, образует структуру нашего знания. Под информацией в быту (житейский аспект) понимают сведения об окружающем мире и протекающем в нем процессах, воспринимаемые человеком или специальными устройствами. Информацией для человека являются не только сухие факты, строгие инструкции, но и то, что радует нас, волнует, печалит, заставляет переживать, восторгаться, презирать, негодовать. Более половины общего объема сведений, полученных в процессе разговора, приходится на так называемую бессмысловую информацию. Эту информацию говорящий по своему желанию, а иногда и произвольно, сообщает нам своей тональностью разговора, своей возбужденностью, жестикულიцией, выражением лица, глаз и т.д.

Понятие «информация» является основополагающим понятием информатики. Любая деятельность человека представляет собой процесс сбора и переработки информации, принятия на ее основе решений и их выполнения. С появлением средств вычислительной техники информация стала выступать в качестве одного из важнейших ресурсов научно-технического прогресса.

Информация является первичным и неопределяемым в рамках науки понятием, оно используется во всех без исключения сферах жизнедеятельности и, поэтому конкретное

толкование элементов, связанных с понятием «информация», зависит от метода конкретной науки и цели исследования. Термин «информация» в переводе с латинского означает разъяснение, осведомление, изложение. В широком смысле слова информация - это общенаучное понятие, включающее в себя обмен сведениями между людьми, обмен сигналами между живой и неживой природой, людьми и техническими устройствами.

Информация - сведения об объектах и явлениях окружающей среды их параметрах, свойствах и состоянии, которые уменьшают степень неопределенности, имеющуюся о них в реальности.

Наряду с информацией в информатике часто употребляется понятие «данные». Данные обычно рассматриваются как признаки или записанные наблюдения, которые не используются, а только хранятся. В том случае, когда появляется возможность использовать данные для уменьшения степени неопределенности чего-либо, они превращаются в информацию.

Пример 1. Напишите на листе десять номеров телефонов в виде последовательности десяти чисел и покажите их вашему другу. Он воспримет эти цифры как данные, так как они не предоставляют ему никаких сведений.

Затем против каждого номера укажите название фирмы и род деятельности. Для вашего друга непонятные цифры обретут определенность и превратятся из данных в информацию, которую он в дальнейшем мог бы использовать.

Выделяют три фазы существования информации:

1. Ассимилированная информация - представление сообщений в сознании человека, накопленное на систему его понятий и оценок.
2. Документированная информация - сведения, зафиксированные в знаковой форме на каком-то физическом носителе.
3. Передаваемая информация - сведения, рассматриваемые в момент передачи информации от источника к приемнику.

При работе с информацией всегда имеется ее источник и потребитель (получатель). Пути и процессы, обеспечивающие передачу сообщений от источника информации к ее потребителю, называются *информационными коммуникациями*.

Содержание информационного процесса: сбор, передача, обработка и накопление информации

Сбор информации – это процесс получения информации из внешнего мира и приведение ее к стандарту для данной информационной системы. Обмен информацией между воспринимающей ее системой и окружающей средой осуществляется посредством сигналов. Сигнал – средство передачи информации в пространстве и времени. В качестве носителя сигнала могут выступать звук, свет, электрический ток, магнитное поле и т.д. Сбор информации, как правило, сопровождается ее регистрацией, т.е. фиксацией информации на материальном носителе (документе или машинном носителе).

Передача информации осуществляется различными способами: с помощью курьера, пересылка по почте, доставка транспортными средствами, дистанционная передача по каналам связи. Для осуществления последней необходимы специальные технические средства. Дистанционно может передаваться как первичная информация с мест ее возникновения, так и результатная информация в обратном направлении. Поступление информации по каналам связи осуществляется двумя способами: на машинном носителе и непосредственно в компьютер при помощи специальных программных и аппаратных средств.

Преобразование (обработка) информации – внесение изменений в набор данных, вычисления, информационный поиск, сортировка, построение графиков и т.п.

В современных развитых информационных системах машинная *обработка информации* предполагает последовательно-параллельное во времени решение вычислительных задач. Это возможно при наличии определенной организации вычислительного процесса. Вычислительная задача по мере необходимости обращается с

запросами в вычислительную систему. Организация процесса предполагает определение последовательности решения задач и реализацию вычислений. Последовательность решения задается, исходя из информационной взаимосвязи, когда результаты решения одной задачи используются как исходные данные для решения другой.

Технология электронной *обработки информации* – человеко-машинный процесс исполнения взаимосвязанных операций, протекающих в установленной последовательности с целью преобразования исходной информации (первичной) в результатную. Операция представляет собой комплекс совершаемых технологических действий, в результате которых информация преобразуется. Технологические операции разнообразны по сложности, назначению, технике реализации, выполняются на различном оборудовании разными исполнителями.

Хранение и накопление информации вызвано многократным ее использованием, применением постоянной информации, необходимостью комплектации первичных данных до их обработки. Хранение осуществляется на машинных носителях в виде информационных массивов, где данные располагаются по установленному в процессе проектирования группировочному признаку.

2. Виды, свойства и формы представления информации

Все многообразие окружающей нас информации можно сгруппировать по различным признакам.

По признаку *«область возникновения»* информация делится на:

- элементарную – отражает процессы и явления неодушевленной природы;
- биологическую – отражает процессы растительного и животного мира;
- социальную – отражает процессы человеческого общества.

По *способу передачи и восприятия* различают информацию:

- визуальную – передается видимыми образами и символами;
- аудиальную – передается звуками;
- тактильную – передается ощущениями;
- органо-лептическую – передается запахами и вкусом;
- машинную – выдаваемую и воспринимаемую средствами вычислительной техники.

По *форме представления* различают информацию:

- текстовую;
- числовую;
- графическую;
- музыкальную;
- комбинированную.

Информацию, создаваемую и используемую человеком, по *общественному назначению* делят на виды:

- массовая;
- специальная;
- личная.

В информатике рассматривают две формы представления информации:

1) аналоговую (непрерывную), изменяющуюся по амплитуде и по времени непрерывно, например температура тела.

2) дискретную (прерывистую), принимающую конечное число значений, например времена года.

Информация имеет определенные свойства:

1. Полнота информации отражает ее достаточность для понимания и для принятия управленческого решения.

2. Достоверность - свойство информации не иметь скрытых ошибок, т.е. отражать истинное положение дел.
3. Актуальность - соответствие информации текущему моменту времени. С течением времени информация стареет.
4. Ценность - свойство информации, отражающее в какой степени она способствует достижению целей и задач ее потребителя.
5. Защищенность - свойство, характеризующее невозможность несанкционированного использования или изменения.
6. Эргономичность - свойство, характеризующее удобство формы или объема информации с точки зрения данного потребителя.

3. Классификация информации

Важным понятием при работе с информацией является классификация объектов.

Классификация - система распределения объектов (предметов, явлений, процессов, понятий) по классам в соответствии с определенным признаком.

Под объектом понимается любой предмет, процесс, явление материального или нематериального свойства. Система классификации позволяет сгруппировать объекты и выделить определенные классы, которые будут характеризоваться рядом общих свойств. Классификация объектов - это процедура группировки на качественном уровне, направленная на выделение однородных свойств.

Применительно к информации как к объекту классификации, выделенные классы называют информационными объектами (ИО).

Пример. Всю информацию об университете можно классифицировать по многим многочисленным информационным объектам, которые будут характеризоваться общими свойствами:

информация о студентах – в виде ИО «Студент»;

информация о преподавателях – в виде ИО «Преподаватель»;

информация о факультетах – в виде ИО «Факультет» и т.п.

Свойства ИО определяются информационными параметрами, называемыми реквизитами (представляются числовыми данными: вес, стоимость, год или признаками: цвет, фамилия).

Реквизит – логически неделимый информационный элемент, описывающий определенное свойство объекта, процесса, явления.

Пример. Информация о каждом студенте в отделе кадров университета систематизирована и представлена посредством одинаковых реквизитов: фамилия, имя, отчество, пол, год рождения, место рождения, адрес проживания, факультет и т.д.

Разработаны три метода классификации объектов: иерархический, фасетный, дескрипторный. Эти методы различаются разной стратегией применения классификационных признаков.

Иерархическая система классификации строится следующим образом:

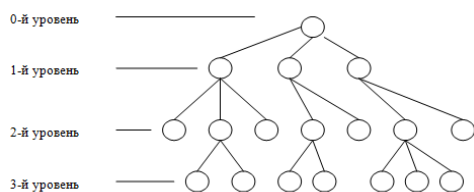


Рис.1. Иерархическая система классификации

- исходное множество элементов составляет 0-й уровень и делится в зависимости от выбранного классифицированного признака на классы, которые образуют 1-й уровень;
- каждый класс 1-го уровня в соответствии со своим, характерным для него классифицированным признаком делится на подклассы, которые образуют 2-й уровень;

- каждый класс 2-го уровня аналогично делится на группы, которые образуют 3-й уровень и т.д.

Учитывая достаточно жесткую процедуру построения структуры классификации, необходимо перед началом работы определить ее цель, т.е. какими свойствами должны обладать объединяемые в классы объекты. Эти свойства принимаются в дальнейшем за признаки классификации.

Количество уровней классификации, соответствующее числу признаков, выбранных в качестве основания деления, характеризует глубину классификации.

Достоинства иерархической системы классификации:

- простота построения;
- использование независимых классификационных признаков в различных ветвях иерархической структуры.

Недостатки:

- жесткая структура, которая приводит к сложности внесения изменений, т.к. приходится перераспределять все классификационные группировки;
- невозможность группировать объекты по заранее не предусмотренным сочетанием признаков.

Фасетная система классификации. В отличие от иерархической позволяет выбирать признаки классификации независимо как друг от друга. Признаки классификации называются *фасетами* (рамка). Каждый фасет (Φ_i) содержит совокупность однородных значений данного классифицируемого признака. Причем значения в фасете могут располагаться в произвольном порядке, хотя и предпочтительнее их упорядочение.

Схема построения фасетной системы: названия столбцов соответствуют выделенным классификационным признакам (фасетам) – цвет, размер одежды, вес и т.д. Произведена нумерация строк таблицы. В каждой клетке таблицы хранится конкретное значение фасета. Например, цвет фасета – Φ_2 содержит значения: красный, белый, зеленый, черный, желтый.

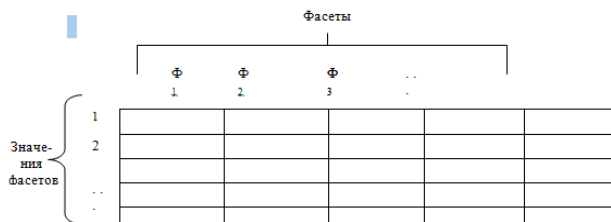


Рис.1. Фасетная система классификации

Процедура классификации состоит в присвоении каждому объекту соответствующих значений из фасетов. При этом могут быть использованы не все фасеты.

Достоинства фасетной системы классификации:

- возможность создания большой емкости классификации, т.е. использование большого числа признаков классификации и из значений для создания группировок
- возможность простой модификации всей системы классификации без изменения структуры существующих группировок

Недостатком фасетной системы классификации является сложность ее построения, т.к. необходимо учитывать все многообразие классификационных признаков.

Дескрипторная система классификации.

Для организации поиска информации, для ведения тезаурусов (словарей) эффективно используется дескрипторная (описательная) система классификации, язык которой приближается к естественному языку описания информационных объектов. Особенно широко она используется в библиотечной системе поиска.

Суть дескрипторного метода классификации:

- отбирается совокупность ключевых слов или словосочетаний, описывающих определенную предметную область или совокупность однородных объектов. Причем среди ключевых слов могут находиться синонимы;

- выбранные ключевые слова и словосочетания подвергаются нормализации, т.е. из совокупности синонимов выбирается один или несколько наиболее употребляемых;

- создается словарь дескрипторов, т.е. словарь ключевых слов и словосочетаний, отобранных в результате процедуры нормализации.

Между дескрипторами устанавливаются связи, которые позволяют расширить область поиска информации. Связи могут быть трех видов:

- *синонимические*, указывающие некоторую совокупность ключевых слов как синонимы (студент – учащийся – обучаемый);

- *родо-видовые*, отражающие включение некоторого класса объектов в более представительный класс (университет – факультет – кафедра);

- *ассоциативные*, соединяющие дескрипторы, обладающие общими свойствами (студент – экзамен – профессор – аудитория).

4. Кодирование информации. Единицы измерения информации

Информация может быть представлена в различных формах - звуковая, печатная и т. д.

В процессе обмена информацией часто приходится производить операции кодирования и декодирования информации.

Кодирование - это операция преобразования знаков или группы знаков одной знаковой системы в знаки или группы знаков другой знаковой системы:

- при вводе знака в компьютер путем нажатия соответствующей клавиши на клавиатуре происходит его *кодирование*, т.е. преобразование во внутренний компьютерный код,

- и наоборот, при выводе знака на экран монитора или принтер происходит обратный процесс - *декодирование*: знак из компьютерного кода преобразуется в его графическое изображение.

Средством кодирования служит таблица соответствия знаковых систем, которая устанавливает взаимно однозначное соответствие между знаками или группами знаков двух различных знаковых систем.

Компьютер может обрабатывать информацию следующих видов:

- числовую,
- текстовую,
- графическую,
- видео,
- звуковую.

Все эти виды информации преобразуются в последовательности электрических импульсов: есть импульс (1), нет импульса (0), т.е. в последовательности нулей и единиц. Такое кодирование информации легко реализуется технически и называется двоичным кодированием.

Существует много кодов: двоичный код обмена информации ДКОИ, восьмеричный код обмена информацией КОИ-8, в современных персональных компьютерах информация представляется в коде ASCII – американский стандартный код для обмена информацией. В России используется так называемая альтернативная ASCII кодировка, содержащая символы русского алфавита.

При вводе информации в компьютер каждый символ-буква русского или латинского алфавита, цифра, знак пунктуации или действия кодируется определенной последовательностью из восьми двоичных цифр в соответствии с таблицей кодирования.

Единицы измерения объема информации

Разные люди, получив одно и тоже сообщение, по-разному оценивают количество информации, содержащееся в нем. Оно зависит от того, насколько ново это сообщение для получателя. При этом подходе трудно выделить критерии, по которым можно было вывести единицу измерения информации.

2. Измерение информации в технике (информация – любая хранящаяся, обрабатываемая или передаваемая последовательность знаков).

А) В технике часто используют способ определения количества информации называемый объемным. Он основан на подсчете числа символов в сообщении, т.е. связан с его длиной и не зависит от содержания.

Б) В вычислительной технике (ВТ) применяют две стандартные единицы измерения:

бит (binary digit) и байт (bate). 1 байт = 8 бит

Бит – минимальная единица измерения информации, которая представляет собой двоичный знак двоичного алфавита {0;1}.

Байт – единица количества информации в СИ, представляющая собой восьмиразрядный двоичный код, с помощью которого можно представить один символ.

Информационный объем сообщения (информационная емкость сообщения) – количество информации в сообщении, измеренное в стандартных единицах или производных от них (Кбайтах, Мбайтах и т.д.).

1 байт = 8 бит

1 Кбайт = 2¹⁰ байт = 1024 байт

1 Мбайт = 2¹⁰ Кбайт = 2²⁰ байт

1 Гбайт = 2¹⁰ Мбайт = 2³⁰ байт

В теории информации количеством информации называют числовую характеристику сигнала, не зависящую от его формы и содержания, и характеризующую неопределенность, которая исчезнет после получения сообщения в виде данного сигнала. В этом случае количество информации зависит от вероятности получения сообщения о том или ином событии.

Для абсолютно достоверного события (событие обязательно произойдет, поэтому его вероятность равна 1) количество информации в сообщении о нем равно 0. Чем неожиданнее событие, тем больше информации он несет.

Лишь при равновероятных событиях: ответ «да» или «нет», несет 1 бит.

1.3 Лекция № 3 (2 часа).

Тема: «Технические средства реализации информационных процессов»

1.3.1 Вопросы лекции:

1. Классификация ЭВМ
2. Основные типы компьютеров. Конфигурации персональных компьютеров
3. Основные принципы функционирования ПК
4. Состав типового компьютера

1.3.2 Краткое содержание вопросов:

1. Классификация ЭВМ.

Классификация ЭВМ по принципу действия. По принципу действия вычислительные машины делятся на три больших класса: аналоговые (АВМ), цифровые (ЦВМ) и гибридные (ГВМ).

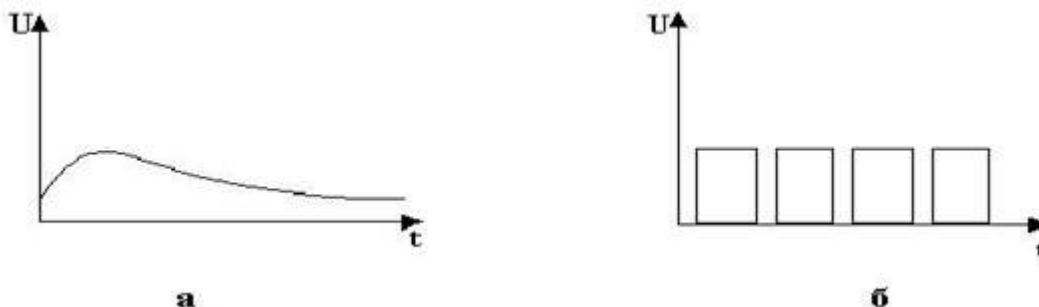
Критерием деления вычислительных машин на эти три класса является форма представления информации, с которой они работают.

Цифровые вычислительные машины (ЦВМ) - вычислительные машины дискретного действия, работают с информацией, представленной в дискретной, а точнее, в цифровой форме.

Аналоговые вычислительные машины (АВМ) - вычислительные машины непрерывного действия, работают с информацией, представленной в непрерывной (аналоговой) форме, т.е. в виде непрерывного ряда значений какой-либо физической величины (чаще всего электрического напряжения).

Аналоговые вычислительные машины весьма просты и удобны в эксплуатации; программирование задач для решения на них, как правило, нетрудоемкое; скорость решения задач изменяется по желанию оператора и может быть сделана сколь угодно большой (больше, чем у ЦВМ), но точность решения задач очень низкая (относительная погрешность 2-5 %).

На АВМ наиболее эффективно решать математические задачи, содержащие дифференциальные уравнения, не требующие сложной логики.



Две формы предоставления информации в машинах: а- аналоговая; б- цифровая импульсная.

Гибридные вычислительные машины (ГВМ) - вычислительные машины комбинированного действия, работают с информацией, представленной и в цифровой, и в аналоговой форме; они совмещают в себе достоинства АВМ и ЦВМ. ГВМ целесообразно использовать для решения задач управления сложными быстродействующими техническими комплексами.

Общепринятой классификацией компьютеров является классификация по поколениям ЭВМ, в основе которой лежит элементная база.

Первое поколение - электронные вакуумные лампы (1946-до середины 50-х годов XX века);

Второе поколение - полупроводниковые приборы, транзисторы (до середины 60-х годов XX века);

Третье поколение - интегральные схемы на полупроводниковых элементах (до конца 70-х годов);

Четвертое поколение - сверхбольшие интегральные схемы (с начала 80-х годов по настоящее время).

Пятое поколение отличительными чертами ЭВМ этого поколения являются новые технологии производства, переход к новым многопроцессорным архитектурам, новые способы ввода-вывода, искусственный интеллект и т.д.

Классификация компьютеров по назначению

Специализированные - предназначены для решения узкого круга специальных задач, например по управлению конкретными техническими устройствами, технологическими процессами (станками с числовым программным управлением, роботами и т.д.).

Универсальные - используются в различных сферах человеческой деятельности для решения самых разнообразных задач: инженерно-технических, экономических, математических, информационно-поисковых и других.

Традиционная классификация производится по: производительности, функциональному назначению и размерам, которая позволяет условно выделить два класса ЭВМ: большие ЭВМ (мэйнфреймы) и персональные компьютеры (мини-ЭВМ). Для каждого класса ЭВМ отличительными признаками также являются области применения, размерность решаемых задач, организационные формы использования, особенности технической архитектуры.

Современные большие ЭВМ. Современные большие ЭВМ называются мэйнфреймами или суперкомпьютерами. Эти ЭВМ характеризуются наивысшим уровнем производительности и надежности, рассчитанные на практически любые уровни нагрузки, обладающие высокой устойчивостью к сбоям и авариям. Они обеспечивают решение любых задач, требующих больших вычислительных ресурсов: от метеорологических прогнозов и изучения управляемого термоядерного синтеза до исследований генома человека и разведки нефти и газодобычи.

Название <мэйнфрейм> (mainframe) происходит от названия корпусов центрального процессора ЭВМ IBM System/360. Именно компьютеры первых моделей семейства ЭВМ System/360, о создании которых фирма IBM объявила в 1964 году, являются родоначальниками мэйнфреймов и первыми компьютерами третьего поколения. В России аналогичная серия машин носит название машин серии ЕС.

При разработке мэйнфреймов особое внимание уделяется техническим и технологическим решениям, которые обеспечивают принцип параллельного (одновременного) выполнения двух или более процессов (программ). Именно возможность параллельной работы различных устройств больших ЭВМ является основой ускорения выполнения вычислительных операций. Создаваемые в настоящее время мэйнфрейм (или суперЭВМ) относятся мощные многопроцессорные вычислительные машины с быстродействием сотни миллионов - десятки миллиардов операций в секунду.

Создать такую высокопроизводительную ЭВМ по современной технологии на одном микропроцессоре не представляется возможным ввиду ограничения, обусловленного конечным значением скорости распространения электромагнитных волн (300 000 км/с), ибо время распространения сигнала на расстояние несколько миллиметров (линейный размер стороны МП) при быстродействии 100 млрд. оп/с становится соизмеримым с временем выполнения одной операции. Поэтому мэйнфрейм создается в виде высокопараллельных многопроцессорных вычислительных систем (МПВС).

Высокопараллельные МПВС имеют несколько разновидностей:

- магистральные(конвейерные) МПВС, в которых процессоры одновременно выполняют разные операции над последовательным потоком обрабатываемых данных; по принятой классификации такие МПВС относятся к системам с многократным потоком команд и однократным потоком данных (МКОД или MISD - Multiple Instruction Single Data);

- векторные МПВС, в которых все процессоры одновременно выполняют одну команду над различными данными - однократный поток команд с многократным потоком данных (ОКМД или SIMD - Single Instruction Multiple Data);

- матричные МПВС, в которых МП одновременно выполняют разные операции над несколькими последовательными потоками обрабатываемых данных - многократный поток команд с многократным потоком данных (МКМД или MIMD - Multiple Instruction Multiple Data).

Условные структуры однопроцессорной (SISD - Single Instruction Single Data) и названных многопроцессорных вычислительных систем показаны на рисунке ниже.

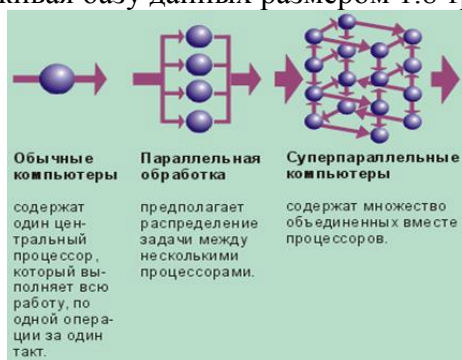
Крупнейшими российскими проектами в области создания суперкомпьютеров являются российский проект МВС и российско-белорусский СКИФ. Крупнейший

суперкомпьютер МВС-15000ВМ отечественной разработки включает 924 процессора Power PC и имеет пиковую производительность 8100 Gflops. Суперкомпьютер установлен в Межведомственном Суперкомпьютерном центре РАН (МСЦ). Основными заказчиками машинного времени на суперкомпьютерах выступают атомная, автомобильная, судостроительная, авиационная и нефтегазовая промышленность.

Суперпараллельные компьютеры (massively parallel computers), схема работы которых показана на рисунке включают в себя сложнейшие цепи процессоров.

Вместо методов параллельной обработки, где небольшое количество мощных, но дорогих специализированных процессоров связаны между собой, суперпараллельные компьютеры содержат сотни и тысячи недорогих обычных процессоров.

Такие ЭВМ достигают производительности суперкомпьютеров. Например, Wal-Mart Stores использует суперпараллельную машину для учета товаров и продаж, обслуживая базу данных размером 1.8 триллионов байт.



Компьютерная архитектура. Сравнение последовательной, параллельной и суперпараллельной обработки данных.

2. Основные типы компьютеров. Конфигурации персональных компьютеров

Персональные компьютеры представляют наиболее многочисленный и разнообразный по составу класс ЭВМ. Они используются при решении самых разных задач не только профессиональными программистами, но и специалистами других областей знаний и деятельности.

К ПК (или ПЭВМ) относится ЭВМ, управляемая одним пользователем и предоставляющая пользователю в каждом сеансе работы все свои ресурсы. К особенностям, отличающим ПК от других ЭВМ, следует отнести:

1. Универсальный характер использования, в соответствии с которым на ПК могут решаться экономические, научные, производственно-технические, конструкторско-технологические и другие задачи в различных сферах человеческой деятельности.

2. Модульный характер построения архитектуры ПК, позволяющий формировать техническую конфигурацию, определять состав внутренних и внешних устройств ПК в зависимости от характера решаемых задач, требований пользователя и финансовых возможностей.

3. Развитость и разнообразие программного обеспечения (ПО), направленные на решение задач из различных областей знаний и деятельности человека.

4. Небольшие габариты, высокая надежность работы, отсутствие специальных требований к условиям эксплуатации и наличие <дружественного> человеко-машинного интерфейса, дающие возможность устанавливать ПК на рабочие места пользователя.

Первоначально основным признаком ПК служило наличие в нем микропроцессора (МП), выполненного в виде одной микросхемы. В настоящее время этот признак перестал быть определяющим, так как МП используются во всех классах ЭВМ.

ПК классифицируются по следующим признакам:

1. По размеру ПК делятся на стационарные (Desktop) и переносные ПК. В состав переносных ПК включаются портативные (Laptop), блокнотного типа (Notebook) и карманные (Palmtop) ПК.

2. По типу используемых МП различают ПК, построенные на процессорах с расширенной системой команд - CISC-процессорах (CISC - Complete Instruction Set Computer), и ПК, основанные на процессорах с сокращенным набором команд - RISC-процессорах (RISC - Reduce Instruction Set Computer).

К первой группе ПК относятся ПК с Intel-совместимой архитектурой. Во вторую группу входят ПК Macintosh на МП Power PC (более ранние модели ПК Macintosh выпускались на базе МП Motorola 680x0), компьютеры SUN с процессорами SPARC-архитектуры, компьютеры DEC с МП Alpha и другие менее распространенные ПК.

Необходимо обратить внимание, что приведенные ПК представляют не отдельные модели, а программно совместимые <сверху-вниз> семейства поколений компьютеров.

3. Классификация ПК по поколениям МП включает для Intel-совместимых ПК следующие поколения: 8086/8088, 80286, 80386, 80486, Pentium, Pentium II, Pentium III, Pentium IV. В каждом поколении имеются различные модификации моделей МП, различающиеся структурой, техническими характеристиками и ценой.

4. Классификация ПК по назначению устанавливает четыре группы ПК:

- массовые ПК, к категории которого также относятся рабочие станции клиентов вычислительных сетей;
- специализированные ПК, ориентированные на работу в конкретных предметных областях, например в области медицины, дизайнерской работе и т.д.
- мультимедийные ПК;
- компьютеры, выполняющие функции серверов в вычислительных сетях.

3. Основные принципы функционирования ПК

ЭВМ определяется как комплекс взаимосвязанных программно-управляемых технических устройств, предназначенных для автоматизированной обработки данных с целью получения результатов решения вычислительных и информационных задач.

Основные принципы функционирования компьютера были сформулированные в 1945 году Джоном фон Нейманом.

1. В основе работы ЭВМ лежит программный принцип, согласно которому все вычисления выполняются путем последовательного выполнения команд программы ЭВМ.

2. Принцип хранимой программы означает, что программы и данные во время выполнения программы хранятся в одном адресном пространстве в оперативной памяти и различаются не по способу кодирования, а по способу использования. Программа может выступать также в качестве исходных данных (сагомодифицируемые программы).

3. Использование двоичного кодирования при хранении и обработке данных в ЭВМ. Отдельные разряды двоичного числа объединяются в более крупные единицы, называемые словами.

4. Информация размещается в ячейках различных запоминающих устройств. Каждая ячейка памяти имеет адрес, по которому происходят запись или считывание слов данных и программ.

К настоящему времени принципы фон Неймана дополнены рядом других принципов:

- открытая архитектура, которая означает, что в основе разработки новых ЭВМ лежат общедоступные стандарты, которые унифицируют взаимодействие различных типов оборудования и отдельных технических узлов ЭВМ. Использование при разработке оборудования открытых стандартов позволяет разным производителям разрабатывать для ЭВМ новые аппаратные средства, заменяющие или дополняющие существующее оборудование;

- модульность построения технической архитектуры состоит в том, что вся ЭВМ состоит из отдельных функционально и конструктивно законченных модулей. Соблюдение этого принципа упрощает процедуру замены устаревших или неработоспособных узлов ЭВМ на современные или рабочие;

- стандартизация технических устройств ЭВМ означает, что все устройства ЭВМ согласованы по своим электрическим, электромагнитным параметрам, протоколам работы, габаритам и т.д.;

- принцип микропрограммирования, заключающийся в том, что машинный язык не является конечной субстанцией, приводящей в действие процессы в ЭВМ. Процессор имеет в своем составе блок микропрограммного управления. Этот блок для каждой команды на машинном языке генерирует последовательность действий-сигналов для физического выполнения требуемой машинной команды. Можно также считать набор команд микропрограммами по отношению к операционной системе.

При этом под архитектурой ЭВМ понимают абстрактное представление ЭВМ, которое отражает ее структурную, схемотехническую и логическую организацию.

Понятие архитектуры является комплексным и включает:

- структурную схему ЭВМ;
- средства и способы доступа к элементам структурной схемы ЭВМ;
- организацию и разрядность интерфейсов ЭВМ;
- организацию и способы адресации памяти;
- способы представления и форматы данных ЭВМ;
- набор машинных команд ЭВМ;
- обработку прерываний.

Структура ЭВМ представляет совокупность конструктивных элементов (устройств), из которых состоит ЭВМ, и связей между ними.

Связь между различными устройствами, представляющую собой физическую магистраль, состоящую из многопроводной линии для передачи электрических сигналов, называют интерфейсной шиной. Различают шины для передачи адресов, управляющих сигналов и данных.

Перечисленные принципы функционирования ЭВМ предполагают обязательное наличие у ЭВМ следующих устройств:

- арифметико-логического устройства (АЛУ), выполняющее арифметические и логические операции;
- устройство управления, предназначенное для организации и синхронизации работы всех устройств ЭВМ;
- память для хранения данных;
- внешние устройства для обеспечения обмена информацией с человеком.

Обобщенная структурная схема ЭВМ представлена ниже

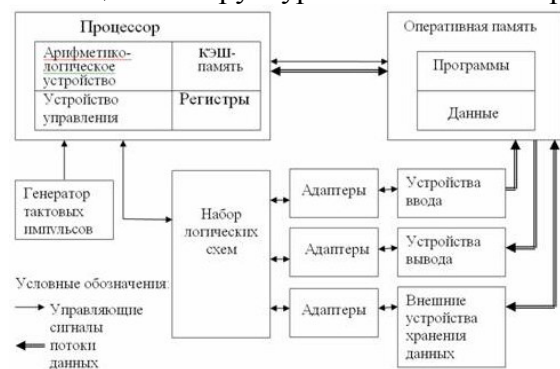


Рис. 1 Обобщенная структурная схема ЭВМ

В современных компьютерах арифметико-логическое устройство и устройство управления объединены в один блок - процессор, предназначенный для обработки данных

по заданной программе путем выполнения арифметических и логических операций и программного управления работой устройств компьютера.

Все арифметические и логические операции непосредственно выполняются арифметико-логическим устройством.

Устройство управления формирует и подает во все блоки ЭВМ управляющие импульсы, обусловленные выполняемой командой.

Для кратковременного хранения данных, непосредственно используемых в вычислениях, имеются специальные ячейки памяти процессора, называемые процессорной памятью или регистрами.

Под кэш - памятью понимают особый вид быстродействующей памяти, выполняющей в компьютере роль промежуточной памяти (буфера) при обмене данными между быстродействующим устройством ЭВМ и менее быстродействующим с целью уменьшения периодов ожидания более производительного устройства.

Программы и данные во время непосредственного сеанса работы хранятся в основной (оперативной) памяти компьютера.

Оперативная память состоит из ячеек памяти одинаковой длины.

Байт является наименьшей адресуемой единицей оперативной памяти. Для идентификации ячеек в оперативной памяти каждой из них присваивается адрес, представляющий собой номер ячейки.

Ячейки нумеруются числами из последовательного натурального ряда чисел. Организация оперативной памяти ЭВМ представлена на рис.4.2.

Запись в память данных осуществляется подачей на шину адреса сигналов, соответствующих адресам ячеек, в которые помещаются данные из шины записи.

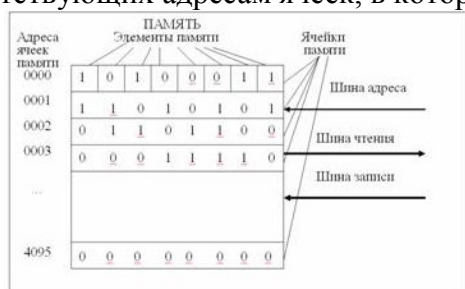


Рис.2 Организация оперативной памяти ЭВМ

При чтении данных из памяти по шине адреса передаются адреса читаемых ячеек, а сами данные из ячеек передаются по шине чтения. Возможность произвольного доступа к любой из ячеек памяти позволяет называть оперативную память, как память с произвольным доступом (RAM - Random Access Memory).

Тактовые импульсы вырабатываются генератором тактовых импульсов ЭВМ и используются для синхронизации процессов передачи информации между устройствами. Базовая последовательность импульсов задает тактовую частоту работы процессора и во многом определяет скорость работы ЭВМ.

Внешние устройства ввода-вывода и хранения данных подключаются к ЭВМ через адаптеры или контроллеры. Основное назначение адаптера состоит в управлении и синхронизации работы внешнего устройства с работой других устройств ЭВМ.

Устройства ввода обеспечивают считывание данных с определенных устройств (клавиатуры, сканера, графических манипуляторов и других) и преобразование их в последовательности электрических сигналов, воспринимаемых другими устройствами ЭВМ.

Устройства вывода представляют результаты обработки информации в форме, удобной для визуального восприятия. К таким устройствам относятся принтеры, мониторы, графопостроители.

Внешние устройства хранения предназначаются для организации долговременного хранения данных и программ. К устройствам внешнего хранения относятся накопители на

жестких и гибких дисках, DVD (Digital Video Disk) и CD (Compact Disk) накопители, накопители на магнитных лентах (стримеры), Flash - память и другие.

Управление работой внутренних и внешних устройств ЭВМ производится устройством управления процессора через основной набор логических схем компьютера.

Выполнение команд программы процессором.

Рассмотрим выполнение процессором команд программы.

В общем случае формат машинной команды состоит из двух частей. Одна часть содержит код операции, которую команда должна выполнить. Другая часть - адресная, содержащая адреса оперативной памяти операндов, над которыми эта операция должна быть выполнена и по какому адресу должен быть помещен результат выполнения команды. На рис. 4.3 представлен пример двухадресной команды ЭВМ.

Код операции	Адрес первого операнда	Адрес второго операнда и результата выполнения
--------------	------------------------	--

Рис. 3 Пример двухадресной команды ЭВМ

Выполнение программы начинается с загрузки программы и исходных данных в оперативную память с внешнего устройства хранения (если программа не резидентная) и происходит под управлением Операционной системы.

Процесс начинается с чтения и выборки первой команды программы из оперативной памяти. С этой целью в счетчике команд по управляющему сигналу процессора $<+1>$ формируется адрес первой выполняемой команды.

В конце цикла выполнения любой команды в счетчике команд всегда формируется адрес следующей команды, которая должна выполняться вслед за текущей.

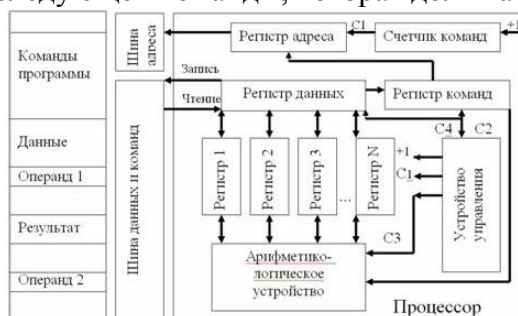


Рис. 4 Выполнение процессором команд программы

В случае, если текущей командой была команда перехода, то в качестве адреса следующей команды в счетчик команд, помещается адрес перехода.

Из счетчика команд адрес передается в регистр адреса, а оттуда поступает в адресную шину оперативной памяти для доступа к ячейкам памяти, содержащим команду.

Команда выбирается из оперативной памяти и через шину данных и команд поступает сначала в регистр данных процессора и далее в регистр команд.

Регистр данных выполняет роль буфера между памятью и остальными регистрами процессора; через него пересылаются команды и операнды из памяти и передаются в память результаты обработки.

Устройство управления считывает из регистра команд код выполняемой команды и переходит к ее выполнению. Выдается управляющий сигнал по которому из регистра команд адреса операндов последовательно передаются в регистр адреса, а затем в адресную шину.

Код операции из регистра команд передается в арифметико-логическое устройство. Операнды выбираются из памяти и помещаются сначала в регистр данных, а после в операционные регистры 1, ..., регистры N процессора.

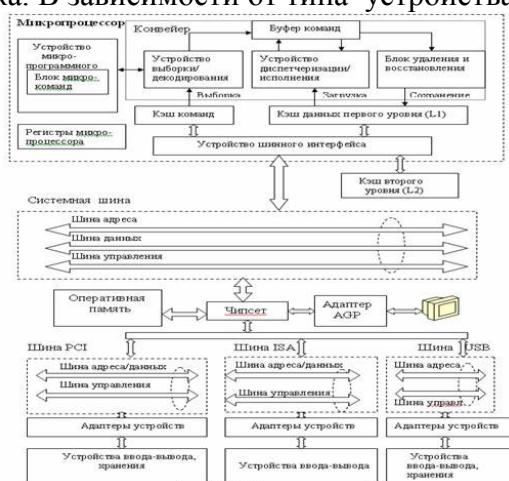
Устройство управления формирует и передает в арифметико-логическое устройство (АЛУ) сигнал на выполнение команды. Операнды последовательно выбираются в АЛУ, выполняется операция, результат которой помещается в один из операционных регистров и далее в регистр данных.

По сигналу устройства управления адрес результата передается из регистра команд в регистр адреса и далее в шину данных. Одновременно из регистра данных по шине данных передается и записывается в память результат решения. В цикле выполнения следующей команды все действия повторяются.

4. Состав типового компьютера

Структуру ПК рассмотрим на примере обобщенной схемы технической конфигурации ПК, представленной на рис. 7. Устройства, составляющие внутреннюю конфигурацию ПК, располагаются на системной плате.

К основным устройствам внутренней конфигурации относят: микропроцессор (МП), оперативную память, основной набор микросхем (ChipSets, чипсет), КЭШ-память, а также интерфейсные шины, используемые для связи устройств между собой: системную шину, шины ISA (Industry Standard Architecture - архитектура промышленного стандарта), PCI (Peripheral Component Interconnect - соединение внешних устройств), USB (Universal Serial Bus - универсальная последовательная шина), AGP (Advanced Graphic Port - расширенный графический порт), генератор тактовых импульсов (ГТИ), адаптеры и порты ввода-вывода. Внешние или периферийные устройства ПК подключаются к интерфейсным шинам через адаптеры (или контроллеры) и предназначены для организации интерфейса ПК с пользователем, осуществления операций ввода-вывода и хранения данных. Во многом, благодаря разнообразным внешним устройствам, обеспечивается универсальность использования ПК в различных областях деятельности человека. В зависимости от типа устройства,



Внешние устройства ПК.

Внешние устройства делятся на три группы:

- устройства ввода данных;
- устройства вывода данных;
- устройства хранения данных.

К устройствам ввода данных относят: клавиатуру, сканер, графический планшет, специальные манипуляторы: мышь, джойстик, трекбол, тачпад.

Устройства вывода включают: монитор, принтер, графопостроитель

Внешние устройства хранения данных составляют накопители на: жестком диске, компакт-диске, DVD, гибких дисках, магнитооптических устройствах. К этой же группе устройств относятся внешняя флэш-память и стример.

Состав и функции микропроцессора ПК.

МИКРОПРОЦЕССОР - самостоятельное или входящее в состав микроЭВМ устройство обработки информации, выполненное в виде одной или нескольких больших интегральных схем. Микропроцессор и средства вычислительной техники и автоматики на их основе применяются в системах автоматического управления, технологического и

контрольно-испытательного оборудования, транспортных средств, бытовых приборов и др.

Микропроцессор - это интегральная схема, смонтированная кремниевой пластине. Процессор содержит тысячи, или даже миллионы транзисторов, связанных между собой сверхтонкими алюминиевыми соединительными каналами, обеспечивающими их взаимодействие при записи и обработке данных, позволяя микропроцессору выполнять множество полезных функций. Конкретные задачи микропроцессора определяются программным обеспечением.

Микропроцессоры используются, прежде всего, в качестве "мозга" персональных компьютеров, но благодаря им становятся "разумными" и многие другие устройства. Например, оснащенный микропроцессором телефонный аппарат имеет возможность ускоренного и повторного набора номера, домашний термостат автоматически отключается по ночам, автомобиль становится безопаснее и потребляет меньше горючего.

Технологии, связанные с компьютерными микросхемами, используется сейчас повсеместно - от космических кораблей и компьютеров до светофоров и кофемолок. Если устройству можно "приказать" сделать что-нибудь, или запрограммировать его свойства, то в него наверняка встроена компьютерная микросхема.

Назначение и сложность микросхем может сильно отличаться. Один из самых совершенных типов микросхем - это микропроцессоры. Встроенные в них схемы могут выполнять сотни миллионов команд в секунду.

Микропроцессор характеризуется:

- тактовой частотой, определяющей максимальное время выполнения переключения элементов в ЭВМ;
- разрядностью, т.е. максимальным числом одновременно обрабатываемых двоичных разрядов

Разрядность МП обозначается $m/n/k/$ и включает:

m - Разрядность внутренних регистров, определяет принадлежность к тому или иному классу процессоров;

n - Разрядность шины данных, определяет скорость передачи информации;

k - Разрядность шины адреса, определяет размер адресного пространства.

Например, МП i8088 характеризуется значениями $m/n/k=16/8/20$;

- архитектурой. Понятие архитектуры микропроцессора включает в себя систему команд и способы адресации, возможность совмещения выполнения команд во времени, наличие дополнительных устройств в составе микропроцессора, принципы и режимы его работы выделяют понятия микроархитектуры и макроархитектуры.

Архитектура типового микропроцессора.

Микропроцессор или микрокомпьютер является практически законченной системой управления. Он имеет сложную архитектуру и представляет собой сверхбольшую интегральную схему, выполненную, как правило, на одном полупроводниковом кристалле. Различные типы микропроцессоров отличаются типом и размером памяти, набором команд, скоростью обработки данных, количеством входных и выходных линий, разрядностью данных. В самом общем виде структурная схема микропроцессора может иметь следующий вид рисунке ниже.



Центральный процессор (CPU) является обязательным узлом любого микропроцессорного устройства, его ядром. В его состав входит: арифметико-логическое

устройство (АЛУ); регистр-аккумулятор; логические устройства управления и синхронизации; внутренняя шина.

Арифметико-логическое устройство выполняет арифметические или логические операции над данными, представленными в двоичном или двоично-десятичном коде. Результат выполнения операции сохраняется в так называемом регистре-аккумуляторе. Регистр-аккумулятор представляет собой ячейки оперативной памяти, но, в отличие от ОЗУ, обмен информацией производится более короткими командами, т.е. регистр-аккумулятор является наиболее быстродействующим устройством памяти микропроцессора.

Устройство управления и синхронизации применяется для управления другими узлами микропроцессора, обеспечивая выполнение необходимых задач в соответствии с программой, хранимой в ПЗУ. Узел синхронизации обеспечивает синхронную работу всех узлов с помощью импульсов синхронизации и других управляющих сигналов. В состав устройства управления и синхронизации входит тактовый генератор и формирователь тактовых импульсов. Для генерации импульсов синхронизации используется кварцевый генератор, имеющий внешний кварцевый резонатор. Частота тактового генератора определяет быстродействие микропроцессора.

Связь между различными элементами микропроцессора осуществляется с помощью внутренней шины. Шина - это группа проводников, используемых в качестве линии связи для передачи цифровой информации. В микропроцессоре имеется три основных вида шин: это шина данных, адресная шина и шина управления.

Шина данных обеспечивает передачу данных между узлами процессора. Адресная шина используется для передачи адреса ячейки памяти с целью получить данные из постоянного запоминающего устройства или оперативного запоминающего устройства. Шина управления используется для передачи управляющих сигналов от микропроцессора к другим элементам системы.

Постоянное запоминающее устройство (ПЗУ) используется для хранения постоянной информации, которая вводится в него на этапе производства микропроцессора и не может быть изменена. Это значит, что записанные на заводе-изготовителе данные сохраняются неизменными при выключении питания микропроцессора. ПЗУ расположено на кристалле микропроцессора и состоит из большого количества ячеек. Каждая ячейка памяти имеет свой порядковый номер, называемый адресом. В этих ячейках хранятся коды команд - это и есть управляющая программа, исполняемая микропроцессором во время его работы. Информация вводится в ПЗУ на этапе изготовления микропроцессора, а процедура введения этой информации называется масочным программированием.

Оперативное запоминающее устройство (ОЗУ) используется для временного хранения промежуточных данных. Микропроцессор в процессе работы может изменять эти данные. При выключении питания информация, хранимая временно в ОЗУ, не сохраняется.

Устройство ввода/вывода (интерфейс ввода/вывода) обеспечивает связь с периферийными устройствами - микросхемами, клавиатурой и др. Подключение к внешним устройствам производится через специальные устройства, называемые портами. Они выполнены в виде набора двунаправленных линий. На структурной схеме показан параллельный 8-разрядный порт (выводы 0...7), который можно конфигурировать различным образом. Последовательный порт можно реализовать, используя две линии параллельного порта - одну для передачи, другую для приема необходимых данных. Количество портов может быть любое и зависит от выполняемых микропроцессором задач.

Классификация и характеристика видов памяти и запоминающих устройств ПК:

Жесткий диск

Жесткий диск (накопители на жестких магнитных дисках, НЖМД) - тип постоянной памяти. В отличие от оперативной памяти, данные, хранящиеся на жестком

диске, не теряются при выключении компьютера, что делает жесткий диск идеальным для длительного хранения программ и файлов данных, а также самых важных программ операционной системы. Эта его способность (сохранение информации в целостности и сохранности после выключения) позволяет доставать жесткий диск из одного компьютера и вставлять в другой.

Жесткие диски очень надежны для хранения большого объема информации и данных. Внутри запечатанного жесткого диска находятся один или больше несгибающихся дисков, покрытых металлическими частицами. Каждый диск имеет головку (маленький электромагнит), встроенную в шарнирный рычаг, который движется над диском при его вращении. Головка намагничивает металлические частички, заставляя их выстраиваться для представления нулей и единиц двоичных чисел. Моторы,двигающие диск и рычаг, обычно подвергаются износу. Избежать износа удастся только головке, поскольку она никогда не соприкасается с поверхностью диска.

Еще одна функция жесткого диска - симуляция оперативной памяти. Используя секции жесткого диска в качестве виртуальной памяти, Windows может запускать больше программ. Недостаток виртуальной памяти в ее медленности по сравнению с обычной памятью. Если поставить больше, работа компьютера замедляется.

Винчестер, или жесткий диск, - самая важная составляющая компьютера. На нем хранится операционная система, программы и данные. Без операционной системы Windows нельзя запустить компьютер, а без программ - ничего сделать, когда он уже загрузился. Без банка данных придется информацию каждый раз вводить вручную.

Жесткий диск - механическое устройство компьютера, и от него может быть больше проблем, чем от электронных устройств. На самом деле оно очень надежно. Диски собирают в чистых комнатах, в которых воздух постоянно фильтруется, и удаляются частички пыли. Собирают винчестеры из магниточувствительного материала. Перед тем как вынести диски из комнаты, их упаковывают и запечатывают. Если вы откроете свой жесткий диск из любопытства, то можете с ним попрощаться. Чтобы этого не случилось, никогда не делайте этого - их вскрывать нельзя.

Перед использованием новые жесткие диски нужно отформатировать. Этот процесс состоит в прокладывании магнитных концентрических дорожек и в их разбивке на маленькие сектора, как куски в торт. Будьте осторожны: если на жестком диске были записаны данные, то его форматирование ведет к полному их уничтожению.

За счет гораздо большего количества дорожек на каждой стороне дисков и большого количества дисков информационная емкость жесткого диска может в сотни тысяч раз превышать информационную емкость дискеты и достигать 150-200 Гбайт. Скорость записи и считывания информации с жестких дисков достаточно велика (может достигать 133 Мбайт/с) за счет быстрого вращения дисков (до 7200 об./мин).

В жестких дисках используются достаточно хрупкие и миниатюрные элементы (пластины носителей, магнитные головки и пр.), поэтому в целях сохранения информации и работоспособности жесткие диски необходимо оберегать от ударов и резких изменений пространственной ориентации в процессе работы.

Дисководы гибких магнитных дисков хранят данные примерно так же, как и жесткие диски - путем изменения магнитной полярности металлических частичек на их поверхности. Здесь сходство заканчивается. Первое отличие состоит в том, что дискеты можно вынимать из дисководов, то есть теоретически они предлагают неограниченное хранилище данных любому желающему. Но ограниченная емкость дискет (1,44 МБ в отличие тысяч мегабайтов жестких дисков) уменьшает их полезность.

CD-ROM / DVD-ROM

Пишущий CD-ROM может записывать информацию любого типа - музыку, изображение или текст. Есть записываемые диски, на которые можно записать информацию только один раз (CD-R). Но есть и перезаписываемые диски (CD-RW), они стоят дороже, но позволяют стирать информацию и добавлять новую. Однако, если вы

записываете музыку на перезаписываемый компакт-диск, вы можете его слушать только на ПК, а записываемый диск - на любом CD-плеере.

Накопители на магнитной ленте (стримеры) и накопители на сменных дисках

Стример (англ. tape streamer) - устройство для резервного копирования больших объемов информации. В качестве носителя здесь применяются кассеты с магнитной лентой ёмкостью 1 - 2 Гбайта и больше.

Стримеры позволяют записать на небольшую кассету с магнитной лентой огромное количество информации. Встроенные в стример средства аппаратного сжатия позволяют автоматически уплотнять информацию перед её записью и восстанавливать после считывания, что увеличивает объём сохраняемой информации.

Недостатком стримеров является их сравнительно низкая скорость записи, поиска и считывания информации. На данный момент стримеры являются устаревшими и поэтому используются они на практике очень редко.

Накопитель на магнито-оптических компакт-дисках CD-MO (Compact Disk - Magneto Optical). Диски CD-MO можно многократно использовать для записи. Ёмкость от 128 Мбайт до 2,6 Гбайт. Накопитель WARM (Write And Read Many times), позволяет производить многократную запись и считывание.

Состав и характеристика основных устройств, образующих внешнюю конфигурацию ПК:

Клавиатура

Клавиатура служит для ввода информации. Для подключения клавиатуры имеется специальный порт клавиатуры. Стандартная клавиатура имеет более 100 клавиш. Для удобства пользования клавиши объединены в несколько групп:

Группа алфавитно-цифровых клавиш служит для ввода чисел, букв и знаков препинания.

Группу функциональных клавиш составляют клавиши от F1 и F12. Действие этих клавиш зависит от того, какая программа загружена в компьютер.

Клавиши управления курсором - это четыре клавиши со стрелками указывающими влево, вправо, вверх и вниз .

Группа дополнительных цифровых клавиш используется в некоторых программах. В компьютерных играх эти клавиши могут использоваться вместо курсорных клавиш.

Клавиша с эмблемой Windows позволяет открыть Главное меню.

Монитор

Для представления буквенно-цифровой и графической информации служат мониторы. Монитор или дисплей является одним из основных блоков персонального компьютера (и одним из самых дорогих) и от его характеристик в значительной степени зависит эффект использования компьютера. Монитор подключается к компьютеру через плату видеоадаптера, а его нормальную работу обеспечивает набор специальных драйверов, поставляемых вместе с монитором.

В современных компьютерах используются два типа мониторов: на основе электронно-лучевых трубок (ЭЛТ) и жидкокристаллические дисплеи. В последнее время жидкокристаллические дисплеи используются все более широко и не только в портативных компьютерах.

Второй важной характеристикой видеомонитора является размер экрана. Обычно приводится размер его диагонали в дюймах. Наиболее распространенным является размер экрана 17 дюймов. Однако для профессионального использования графических пакетов и настольных издательских систем могут использоваться мониторы большей диагонали.

В настоящее время в персональных компьютерах используются исключительно цветные мониторы, черно-белые применяются только в специализированных системах.

Четкость изображения тем выше, чем меньше размер точек люминофора и соответственно расстояние между ними. В настоящее время наибольшее распространение получили дисплеи с расстоянием между пикселями (dot pitch) 0,28 мм.

До недавнего времени жидкокристаллические (ЖК) дисплеи использовались, в основном, в переносных компьютерах из-за их малых размеров, веса и низкого энергопотребления. Теперь же такие мониторы постепенно заменяют мониторы на ЭЛТ. Существуют две основные разновидности ЖК-дисплеев - с пассивными и активными матрицами. Последние обеспечивают более высокое качество изображения, малое время отклика, которое определяет скорость смены изображения на экране, и большой угол зрения. Наиболее популярный размер плоских дисплеев - 10,4 дюйма.

Видеоадаптер служит для предварительной обработки информации и подготовки ее для вывода на экран монитора. Физически он представляет собой отдельную плату, которая вставляется в соответствующий слот расширения материнской платы. На задней стороне платы имеется разъем для подключения монитора. В зависимости от объема видеопамати видеосистема может обеспечить различные параметры разрешающей способности и количества воспроизводимых цветов.

Мышь (трекбол, джойстик)

Мышь или графический манипулятор - это удобное средство, с помощью которого можно управлять объектами в графической операционной системе.

Мышь имеет шарик и две кнопки. При перемещении мыши шарик вращается, и в компьютер посылаются электрические сигналы. Как правило, мышь имеет две кнопки - основную и дополнительную. Существуют несколько конструкций мышей.

Наиболее распространенной и дешевой является в настоящее время оптико-механическая мышка. Основным элементом является тяжелый обрезиненный шарик, который "катается" по поверхности стола или специального коврика. К шарiku внутри корпуса прижаты два ролика с вертикальной и горизонтальной осями вращения. Главные враги мыши - грязь и пыль. Оптико-механические мыши чрезвычайно чувствительны к любым загрязнениям рабочей поверхности и довольно быстро выходят из строя.

Оптические мыши лишены этих недостатков. Они практически не имеют механических движущихся частей и потому очень долговечны. Такая мышь может работать только на специальном планшете, который покрыт мелкой сеткой перпендикулярных линий, нанесенных на светоотражающую поверхность. Благодаря своей конструкции оптическая мышь практически безотказна и имеет неопределенно большой срок работы.

Принтер

Принтер является основным устройством вывода информации для получения "твердой копии". В отличие от дисплея принтер обеспечивает получение документа, отпечатанного на бумаге. Наиболее распространенными принтерами являются лазерные и струйные.

Матричные принтеры - первый из изобретенных типов принтеров для персональных компьютеров. Они были относительно дешевы и обеспечивали удовлетворительное качество печати. Принцип их действия достаточно прост: конфигурация любого знака определяется положением стержней в печатающей головке. В настоящее время матричные принтеры не применяются ввиду низкой скорости и невысокого качества печати по сравнению со струйными и лазерными принтерами, изобретенными позднее.

Лазерные принтеры обеспечивают идеальное качество печати, что позволяет их использовать для подготовки полиграфических оригиналов для офсетной печати. Принцип их действия и устройство несколько схожи с ксероксами, однако вместо лампы в них используется маломощный лазерный луч.

В отличие от матричных лазерные принтеры практически бесшумные, обеспечивают высокую скорость печати, автоматическую подачу бланков и имеют многоуровневую систему меню, позволяющую изменять широкий спектр параметров печати, начиная от количества экземпляров каждого отпечатка, и кончая автоматическим счетчиком количества отпечатанных оттисков.

Лазерные принтеры могут обеспечивать как черно-белую печать, так и полноценную цветопередачу, что совершенно недоступно для матричных принтеров. Однако их стоимость в несколько раз выше матричных.

Струйные принтеры весьма популярный тип принтеров для персональных компьютеров. Они сочетают в себе высокое качество печати, достаточно высокое быстродействие при работе в графических средах (Windows-приложения) и невысокую стоимость, сравнимую с матричными принтерами.

Принцип их действия основан на управлении тонкой струей специальных чернил, которые выбрасываются из сопла.

Следует, однако, иметь в виду, что качество печати струйных принтеров сильно зависит от качества бумаги, и фирма-изготовитель гарантирует высококачественную печать лишь на специальной бумаге.

1.4 Лекция № 4 (2 часа).

Тема: «Программные средства реализации информационных процессов. Системное программное обеспечение»

1.4.1 Вопросы лекции:

1. Понятие программного обеспечения и классификация. Системное программное обеспечение
2. Понятие операционных систем, назначение, функции, классификация
3. Понятие файловой системы
4. Операционная система WINDOWS. Общие сведения о WINDOWS. Интерфейс пользователя WINDOWS. Основы работы в WINDOWS

1.4.2 Краткое содержание вопросов:

1. Понятие программного обеспечения и классификация. Системное программное обеспечение

Совокупность программ, предназначенная для решения задач на ПК, называется программным обеспечением. Состав программного обеспечения ПК называют программной конфигурацией.

Программное обеспечение, можно условно разделить на три категории:

- системное ПО (программы общего пользования), выполняющие различные вспомогательные функции, например создание копий используемой информации, выдачу справочной информации о компьютере, проверку работоспособности устройств компьютера и т.д.
- прикладное ПО, обеспечивающее выполнение необходимых работ на ПК: редактирование текстовых документов, создание рисунков или картинок, обработка информационных массивов и т.д.
- инструментальное ПО (системы программирования), обеспечивающее разработку новых программ для компьютера на языке программирования.



Системное ПО

Это программы общего пользования не связаны с конкретным применением ПК и выполняют традиционные функции: планирование и управление задачами, управления вводом-выводом и т.д.

Другими словами, системные программы выполняют различные вспомогательные функции, например, создание копий используемой информации, выдачу справочной информации о компьютере, проверку работоспособности устройств компьютера и т.п.

К системному ПО относятся:

- операционные системы (эта программа загружается в ОЗУ при включении компьютера)
- программы – оболочки (обеспечивают более удобный и наглядный способ общения с компьютером, чем с помощью командной строки DOS, например, Norton Commander)
- операционные оболочки – интерфейсные системы, которые используются для создания графических интерфейсов, мультипрограммирования и т.
- драйверы (программы, предназначенные для управления портами периферийных устройств, обычно загружаются в оперативную память при запуске компьютера)
- утилиты (вспомогательные или служебные программы, которые представляют пользователю ряд дополнительных услуг)

К утилитам относятся:

- диспетчеры файлов или файловые менеджеры
- средства динамического сжатия данных (позволяют увеличить количество информации на диске за счет ее динамического сжатия)
- средства просмотра и воспроизведения
- средства диагностики; средства контроля позволяют проверить конфигурацию компьютера и проверить работоспособность устройств компьютера, прежде всего жестких дисков
- средства коммуникаций (коммуникационные программы) предназначены для организации обмена информацией между компьютерами
- средства обеспечения компьютерной безопасности (резервное копирование, антивирусное ПО).

Необходимо отметить, что часть утилит входит в состав операционной системы, а другая часть функционирует автономно. Большая часть общего (системного) ПО входит в состав ОС. Часть общего ПО входит в состав самого компьютера (часть программ ОС и контролирующих тестов записана в ПЗУ или ППЗУ, установленных на системной плате). Часть общего ПО относится к автономным программам и поставляется отдельно.

Прикладное ПО

Прикладные программы могут использоваться автономно или в составе программных комплексов или пакетов. Прикладное ПО – программы, непосредственно обеспечивающие выполнение необходимых работ на ПК: редактирование текстовых документов, создание рисунков или картинок, создание электронных таблиц и т.д.

Пакеты прикладных программ – это система программ, которые по сфере применения делятся на проблемно – ориентированные, пакеты общего назначения и интегрированные пакеты. Современные интегрированные пакеты содержат до пяти функциональных компонентов: тестовый и табличный процессор, СУБД, графический редактор, телекоммуникационные средства.

К прикладному ПО, например, относятся:

- Комплект офисных приложений MS OFFICE
- Бухгалтерские системы
- Финансовые аналитические системы
- Интегрированные пакеты делопроизводства
- CAD – системы (системы автоматизированного проектирования)
- Редакторы HTML или Web – редакторы
- Браузеры – средства просмотра Web - страниц
- Графические редакторы
- Экспертные системы
- И так далее.

Инструментальное ПО

Инструментальное ПО или системы программирования - это системы для автоматизации разработки новых программ на языке программирования.

В самом общем случае для создания программы на выбранном языке программирования (языке системного программирования) нужно иметь следующие компоненты:

1. Текстовый редактор для создания файла с исходным текстом программы.
2. Компилятор или интерпретатор. Исходный текст с помощью программы-компилятора переводится в промежуточный объектный код. Исходный текст большой программы состоит из нескольких модулей (файлов с исходными текстами). Каждый модуль компилируется в отдельный файл с объектным кодом, которые затем надо объединить в одно целое.

3. Редактор связей или сборщик, который выполняет связывание объектных модулей и формирует на выходе работоспособное приложение – исполнимый код.

Исполнимый код – это законченная программа, которую можно запустить на любом компьютере, где установлена операционная система, для которой эта программа создавалась. Как правило, итоговый файл имеет расширение .EXE или .COM.

4. В последнее время получили распространение визуальный методы программирования (с помощью языков описания сценариев), ориентированные на создание Windows-приложений. Этот процесс автоматизирован в средах быстрого проектирования. При этом используются готовые визуальные компоненты, которые настраиваются с помощью специальных редакторов.

Наиболее популярные редакторы (системы программирования программ с использованием визуальных средств) визуального проектирования:

Borland Delphi - предназначен для решения практически любых задачи прикладного программирования

Borland C++ Builder – это отличное средство для разработки DOS и Windows приложений

Microsoft Visual Basic – это популярный инструмент для создания Windows-программ

Microsoft Visual C++ - это средство позволяет разрабатывать любые приложения, выполняющиеся в среде ОС типа Microsoft Windows

2. Понятие операционных систем, назначение, функции, классификация

Операционная система составляет основу программного обеспечения ПК. Операционная система представляет комплекс системных и служебных программных средств, который обеспечивает взаимодействие пользователя с компьютером и выполнение всех других программ.

С одной стороны, она опирается на базовое программное обеспечение ПК, входящее в его систему BIOS, с другой стороны, она сама является опорой для программного обеспечения более высоких уровней – прикладных и большинства служебных приложений.

Для того чтобы компьютер мог работать, на его жестком диске должна быть установлена (записана) операционная система. При включении компьютера она считывается с дисковой памяти и размещается в ОЗУ. Этот процесс называется загрузкой операционной системы.

Операционные системы различаются особенностями реализации алгоритмов управления ресурсами компьютера, областями использования.

Так, в зависимости от алгоритма управления процессором, операционные системы делятся на:

- Однозадачные и многозадачные
- Однопользовательские и многопользовательские
- Однопроцессорные и многопроцессорные системы
- Локальные и сетевые.

По числу одновременно выполняемых задач операционные системы делятся на два класса:

- Однозадачные (MS DOS)
- Многозадачные (OS/2, Unix, Windows)

В однозадачных системах используются средства управления периферийными устройствами, средства управления файлами, средства общения с пользователями. Многозадачные ОС используют все средства, которые характерны для однозадачных, и, кроме того, управляют разделением совместно используемых ресурсов: процессор, ОЗУ, файлы и внешние устройства.

В зависимости от областей использования многозадачные ОС подразделяются на три типа:

- Системы пакетной обработки (OS EC)
- Системы с разделением времени (Unix, Linux, Windows)
- Системы реального времени (RT11)
- Системы пакетной обработки предназначены для решения задач, которые не требуют быстрого получения результатов. Главной целью ОС пакетной обработки является максимальная пропускная способность или решение максимального числа задач в единицу времени.

Эти системы обеспечивают высокую производительность при обработке больших объемов информации, но снижают эффективность работы пользователя в интерактивном режиме.

В системах с разделением времени для выполнения каждой задачи выделяется небольшой промежуток времени, и ни одна задача не занимает процессор надолго. Если этот промежуток времени выбран минимальным, то создается видимость одновременного выполнения нескольких задач. Эти системы обладают меньшей пропускной способностью, но обеспечивают высокую эффективность работы пользователя в интерактивном режиме.

Системы реального времени применяются для управления технологическим процессом или техническим объектом, например, летательным объектом, станком и т.д.

По числу одновременно работающих пользователей на ЭВМ ОС разделяются на однопользовательские (MS DOS) и многопользовательские (Unix, Linux, Windows 95 - XP)

В многопользовательских ОС каждый пользователь настраивает для себя интерфейс пользователя, т.е. может создать собственные наборы ярлыков, группы программ, задать индивидуальную цветовую схему, переместить в удобное место панель задач и добавить в меню Пуск новые пункты.

В многопользовательских ОС существуют средства защиты информации каждого пользователя от несанкционированного доступа других пользователей.

Многопроцессорные и однопроцессорные операционные системы. Одним из важных свойств ОС является наличие в ней средств поддержки многопроцессорной обработки данных. Такие средства существуют в OS/2, Net Ware, Windows NT. По способу организации вычислительного процесса эти ОС могут быть разделены на асимметричные и симметричные.

Одним из важнейших признаков классификации ЭВМ является разделение их на локальные и сетевые. Локальные ОС применяются на автономных ПК или ПЭВМ, которые используются в компьютерных сетях в качестве клиента.

В состав локальных ОС входит клиентская часть ПО для доступа к удаленным ресурсам и услугам. Сетевые ОС предназначены для управления ресурсами ПК включенных в сеть с целью совместного использования ресурсов. Они представляют мощные средства разграничения доступа к информации, ее целостности и другие возможности использования сетевых ресурсов.

3. Понятие файловой системы

Все современные ОС обеспечивают создание файловой системы, которая предназначена для хранения данных на дисках и обеспечения доступа к ним.

Основные функции файловой системы можно разделить на две группы:

Функции для работы с файлами (создание, удаление, переименование файлов и т.д.)

Функции для работы с данными, которые хранятся в файлах (запись, чтение, поиск данных и т.д.)

Известно, что файлы используются для организации и хранения данных на машинных носителях. Файл – это последовательность произвольного числа байтов, обладающая уникальным собственным именем или поименованная область на машинных носителях.

Структурирование множества файлов на машинных носителях осуществляется с помощью каталогов, в которых хранятся атрибуты (параметры и реквизиты) файлов. Каталог может включать множество подкаталогов, в результате чего на дисках образуются разветвленные файловые структуры. Организация файлов в виде древовидной структуры называется файловой системой.

Принцип организации файловой системы – табличный. Данные о том, в каком месте на диске записан файл, хранятся в таблице размещения файлов (File Allocation Table, FAT).

Эта таблица размещается в начале тома. В целях защиты тома на нем хранятся две копии FAT. В случае повреждения первой копии FAT дисковые утилиты могут воспользоваться второй копией для восстановления тома.

По принципу построения FAT похожа на оглавление книги, так как операционная система использует ее для поиска файла и определения кластеров, которые этот файл занимает на жестком диске.

Наименьшей физической единицей хранения данных является сектор. Размер сектора 512 байт. Поскольку размер FAT – таблицы ограничен, то для дисков, размер которых превышает 32 Мбайт, обеспечить адресацию к каждому отдельному сектору не представляется возможным.

В связи с этим группы секторов условно объединяются в кластеры. Кластер является наименьшей единицей адресации к данным. Размер кластера, в отличие от размера сектора, не фиксирован и зависит от емкости диска.

Сначала для дискет и небольших жестких дисков (менее 16 Мбайт) использовалась 12-разрядная версия FAT (так называемая FAT12). Затем в MS-DOS была введена 16-разрядная версия FAT для более крупных дисков.

Операционные системы MS DOS, Win 95, Win NT реализуют 16 – разрядные поля в таблицах размещения файлов. Файловая система FAT32 была введена в Windows 95 OSR2 и поддерживается в Windows 98 и Windows 2000.

FAT32 представляет собой усовершенствованную версию FAT, предназначенную для использования на томах, объем которых превышает 2 Гбайт.

FAT32 обеспечивает поддержку дисков размером до 2 Тбайт и более эффективное расходование дискового пространства. FAT32 использует более мелкие кластеры, что позволяет повысить эффективность использования дискового пространства.

В Windows XP применяется FAT32 и NTFS. Более перспективным направлением в развитии файловых систем стал переход к NTFS (New Technology File System – файловая система новой технологии) с длинными именами файлов и надежной системой безопасности.

Объем раздела NTFS не ограничен. В NTFS минимизируется объем дискового пространства, теряемый вследствие записи небольших файлов в крупные кластеры. Кроме того, NTFS позволяет экономить место на диске, сжимая сам диск, отдельные папки и файлы.

По способам именования файлов различают “короткое” и “длинное” имя.

Согласно соглашению, принятому в MS-DOS, способом именования файлов на компьютерах IBM PC было соглашение 8.3., т.е. имя файла состоит из двух частей: собственно имени и расширения имени. На имя файла отводится 8 символов, а на его расширение – 3 символа.

Имя от расширения отделяется точкой. Как имя, так и расширение могут включать только алфавитно-цифровые символы латинского алфавита. Имена файлов, записанные в соответствии с соглашением 8.3, считаются “короткими”.

С появлением операционной системы Windows 95 было введено понятие “длинного” имени. Такое имя может содержать до 256 символов. Этого вполне достаточно для создания содержательных имен файлов. “Длинное” имя может содержать любые символы, кроме девяти специальных: \ / : * ? “ < > |.

В имени разрешается использовать пробелы и несколько точек. Имя файла заканчивается расширением, состоящим из трех символов. Расширение используется для классификации файлов по типу.

Уникальность имени файла обеспечивается тем, что полным именем файла считается собственное имя файла вместе с путем доступа к нему. Путь доступа к файлу начинается с имени устройства и включает все имена каталогов (папок), через которые проходит. В качестве разделителя используется символ “\” (обратный слеш - обратная косая черта). Например: D:\Documents and Settings\TBA\Мои документы\lessons-tva\robots.txt

Несмотря на то, что данные о местоположении файлов хранятся в табличной структуре, пользователю они представляются в виде иерархической структуры – людям так удобнее, а все необходимые преобразования берет на себя операционная система.

К функции обслуживания файловой структуры относятся следующие операции, происходящие под управлением операционной системы:

- создание файлов и присвоение им имен;
- создание каталогов (папок) и присвоение им имен;
- переименование файлов и каталогов (папок);
- копирование и перемещение файлов между дисками компьютера и между каталогами (папками) одного диска;
- удаление файлов и каталогов (папок);
- навигация по файловой структуре с целью доступа к заданному файлу, каталогу (папке);
- управление атрибутами файлов.

4. Операционная система WINDOWS. Общие сведения о WINDOWS. Интерфейс пользователя WINDOWS. Основы работы в WINDOWS

Операционная система Windows - это современная многозадачная многопользовательская 32 - разрядная ОС с графическим интерфейсом пользователя. Операционные системы семейства Windows являются наиболее распространенными ОС, которые установлены в домашних и офисных ПК.

Графическая оболочка ОС Windows обеспечивает взаимодействие пользователя с компьютером в форме диалога с использованием ввода и вывода на экран дисплея графической информации, управления программами с помощью пиктограмм, меню, окон, панелей (управления, задач, инструментов) и других элементов управления.

Основными элементами графического интерфейса Windows являются: Рабочий стол, Панель задач с кнопкой Пуск. Так как в Windows применен графический пользовательский интерфейс, то основным устройством управления программами является манипулятор мышь.

Основными элементами графического интерфейса Windows являются:

- Рабочий стол с пиктограммами
- Панель задач, на которой размещаются программные кнопки, индикаторы, Панель быстрого запуска
- Главное меню (кнопка Пуск)
- Контекстное меню (отображается при щелчке правой кнопкой мыши по выбранному объекту)

В ОС Windows применяются четыре типа меню (меню – это список команд, выводимых на экран и предлагаемых пользователю для выбора):

- Главное меню (открывается кнопкой Пуск)
- Строка меню в окнах приложения (все программы, входящие в стандартный пакет поставки Windows, имеют строку меню)
- Системное меню в окнах приложения (для изменения размеров окна и его положения)
- Контекстное меню

Работа с файлами

Все файлы, документы и программы в Windows хранятся в папках. В электронной папке, как правило, хранят файлы, сгруппированные по какому-либо признаку, типу и другие папки.

Папка – это контейнер для программ и файлов в графических интерфейсах пользователя, отображаемый на экране с помощью значка, имеющего вид канцелярской папки. Windows предоставляет средства для управления файлами и папками.

К таким средствам относятся программа Проводник и окно Мой компьютер. Приложение Проводник является главным инструментом Windows для просмотра файлов и папок, хранящихся на жестких и гибких дисках и других носителях информации.

Проводник отображает иерархическую структуру файлов, папок и дисков на ПК. В левой части проводника Windows использует иерархическое представление папок, файлов и других ресурсов, подключенных к компьютеру или сети.

Мой компьютер – программа,используемая для работы с файлами и папками, хранящимися на дисках компьютера. Мое сетевое окружение – программа, используемая для работы с сетевыми ресурсами в рабочей группе.

1.5 Лекция № 5 (2 часа).

Тема: «Программные средства реализации информационных процессов. Сервисное программное обеспечение»

1.5.1 Вопросы лекции:

1. Назначение и виды сервисных программ
2. Программы контроля и диагностики компьютера. Файловые менеджеры
3. Программы записи, обслуживания магнитных и компакт-дисков
4. Программы обслуживания ОС Windows

1.5.2 Краткое содержание вопросов:

1. Назначение и виды сервисных программ.

В процессе практической работы на компьютере у пользователя периодически возникают проблемы, связанные как с работой системы в целом, так и отдельных устройств и программ: замедление работы, нехватка места на диске, ошибки выполнения программ, их зависание и т.п. Для выявления таких проблем, устранения причин их возникновения и обслуживания отдельных устройств в состав системного программного обеспечения включают сервисные программы (утилиты).

Сервисное программное обеспечение – это совокупность программ, которые предназначены для тестирования устройств компьютера и обеспечения нормальной работы основных приложений.

Существует два типа сервисных программ – стандартные, входящие в состав операционной системы, и специализированные, которые часто объединяют в комплексы, подобные Norton Utilities или Acronis Power Utilities. Специализированные программы не только дублируют функции системных утилит в более удобном режиме, но и значительно расширяют их возможности по обслуживанию устройств компьютера, хранению и защите данных.

Перечень сервисных программ, представленных на рынке ПО, очень широк; многие из них дублируют друг друга. По функциональному назначению их можно разделить на несколько групп:

- программы контроля и диагностики компьютера;
 - файловые менеджеры;
 - антивирусные программы;
 - программы обслуживания дисков;
 - программы работы с архивами;
 - программы обслуживания операционной системы;
 - программы обслуживания сети.
2. Программы контроля и диагностики компьютера. Файловые менеджеры

Программы контроля и диагностики компьютера

Данный класс программ предназначен для проверки наличия и работоспособности устройств, определения их характеристик и технических параметров работы.

Существует три уровня тестирования устройств.

Первый уровень – самотестирование базовой системой ввода-вывода (BIOS) при включении компьютера (POST или Power-On-Self-Test). На этом уровне проверяется наличие и работоспособность основных компонентов компьютера: процессора, оперативной памяти, вспомогательных микросхем, приводов дисков, клавиатуры и видеоадаптера. Если тест успешно пройден, выдается один короткий звуковой сигнал и начинается загрузка ОС. Если же BIOS не находит основное устройство или возникает ошибка ввода-вывода, выдается сообщение на экран компьютера (например: «Keyboard error – ошибка клавиатуры») или последовательность звуковых сигналов, означающих определенную ошибку. Описание этих звуковых сигналов содержится в инструкции по эксплуатации материнской платы.

Второй уровень – тестирование операционной системой. ОС при загрузке определяет все подключенные устройства, наличие конфликтов между ними, соответствие им драйверов. При обнаружении ошибки выдается сообщение на экран монитора (например: «Обнаружено новое устройство» или «Ошибка принтера»), возможные причины и варианты их устранения.

Третий уровень – пользовательское тестирование, основанное на применении более основательных общих или специализированных тестов. Существуют программы для измерения производительности дисков, контроля температуры процессора, проверки качества изображения на мониторе и т.д. Имеется специализированное ПО для проверки работы различных устройств (например, *3DMark* – для тестирования видеосистемы) или тестовые программные комплексы (например, *SiSoftware Sandra*, *PCMark*).

В Windows имеется стандартная служебная программа *Сведения о системе*, которая собирает и отображает данные о ее параметрах. Сюда входит, в частности, информация о конфигурации оборудования, компонентах компьютера, установленном программном обеспечении.

Файловые менеджеры

В первых операционных системах для ПК, основанных на интерфейсе командной строки, было крайне неудобно выполнять ряд повседневных операций – отслеживать содержимое дисков и каталогов, копировать, удалять и перемещать файлы, запускать программы на исполнение. Поэтому очень скоро появились программы, которые визуально отображали списки файлов, позволяли нажатием одной-двух клавиш перемещаться по файловой системе и выполнять различные файловые операции.

Эти программы получили название «операционные оболочки», так как выполняли функцию посредника между операционной системой и пользователем. С появлением ОС, снабженных стандартным графическим интерфейсом, операционные оболочки превратились в часть системы и стали называться «диспетчерами файлов» или «файловыми менеджерами».

Файловые менеджеры – это сервисные программы, предназначенные для управления файлами и папками (каталогами) на дисках. Основными операциями управления являются: просмотр содержимого папок, навигация по файловой системе (переход из одной папки в другую), поиск, копирование, перемещение и удаление файлов и папок, запуск программ и т.д. Помимо перечисленных, большинство современных файловых менеджеров предоставляют дополнительные возможности: групповые операции с файлами, упорядочение (сортировка) объектов, возможность работы с файловыми архивами как с обычными папками, средства сравнения файлов и папок и многие другие.

Данный класс программ довольно широко представлен на рынке ПО; отдельные файловые менеджеры отличаются друг от друга по степени удобства работы и наличию дополнительных функций.

В состав стандартных служебных программ ОС Windows входит файловый менеджер, который называется *Проводник* (Explorer). Эта программа широко используется, так как очень проста и обладает некоторыми весьма удобными дополнительными функциями.

Программа *Norton Commander* – классический (со времен DOS) диспетчер файлов со средними возможностями (рис. 40). Первые версии этой программы стали образцом для создания других файловых менеджеров.

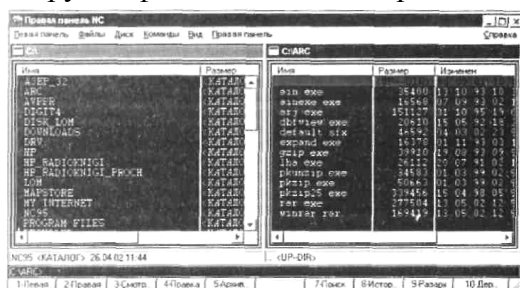


Рис. 1. Вид панелей Norton Commander

Среди специализированных файловых менеджеров следует упомянуть программы *Frigate* (Фрегат), *Total Commander* и *FAR Manager*. Все они используют две независимые друг от друга панели, позволяющие видеть одновременно на экране содержимое двух папок и легко копировать (или перемещать) файлы из одной папки в другую. Интерфейс данных программ включает также командную строку и адаптирован под работу с клавиатурными комбинациями, которые заметно ускоряют работу опытного пользователя.

Frigate (Фрегат) – это очень мощный и многофункциональный диспетчер файлов, разработанный российскими программистами. Он объединяет функции программы Проводник, архиватора, универсального просмотрщика файлов, мультимедиа-проигрывателя, internet-браузера, планировщика задач, FTP-клиента, некоторых утилит и многие другие. *Frigate* позволяет выполнять все файловые операции в фоновом режиме. Недостаток программы – довольно высокие требования к производительности компьютера.

Программа *Total Commander* – весьма популярный файловый менеджер с практически всеми необходимыми для таких программ функциями, в отличие от *Frigate* не предъявляет особых требований к аппаратным ресурсам ПК.

FAR Manager – бесплатно распространяемый диспетчер файлов с текстовым интерфейсом, высокой производительностью и дополнительными возможностями.

3. Программы записи, обслуживания магнитных и компакт-диск

Программы обслуживания магнитных дисков

Наиболее часто используемыми операциями обслуживания магнитных дисков являются их проверка и дефрагментация.

Программы проверки магнитных дисков. Для снижения риска потери информации, хранящейся на жестких и гибких дисках, рекомендуется периодически проверять их на наличие логических и физических ошибок. Физические дефекты возникают из-за механических повреждений, воздействия электромагнитных полей или старения магнитного покрытия дисков. Наличие таких дефектов на магнитной поверхности (секторах и кластерах) делает ее непригодной для хранения информации. Если в этой области записана информация, то полностью ее прочитать будет невозможно. Дефектные

сектора и кластеры диска должны быть исключены из дальнейшего использования программой проверки.

Логические дефекты диска связаны с ошибками в файловой системе диска, возникающими из-за отключения питания, сбоев и зависания ОС, воздействия компьютерных вирусов и других причин.

Возникновению физических дефектов наиболее подвержены дискеты. Иногда они повреждаются так, что полностью выходят из строя (невозможно прочитать даже содержимое дискеты). Поэтому дискеты рекомендуется проверять как можно чаще и желательно при хранении и передачи информации иметь несколько копий на разных дискетах.

Логические ошибки возникают и на жестких дисках, поэтому программа их проверки является очень важной сервисной программой. Она проверяет диски на правильность логической структуры и наличие сбойных участков на их поверхности, исправляет найденные ошибки. При этом производится анализ загрузочной записи, таблицы размещения файлов, структуры каталогов (папок), файловой структуры, потерянных кластеров. В случае обнаружения ошибок программа может их исправить в ручном и автоматическом режиме; по завершении работы выводится информация о результатах анализа и внесенных исправлениях.

Для проверки дисков в ОС Windows имеется стандартная утилита *Scandisk*, в пакете NU – *Norton Disk Doctor (NDD)*.

Когда пользователь начинает работу с чистым магнитным диском, ОС пишет на него файлы последовательно, один за другим. Однако при интенсивных манипуляциях с файлами (при модификации, удалении старых и записи новых файлов) нарастает уровень фрагментации диска: файл, который логически воспринимается системой как единое целое, физически разделен на множество цепочек кластеров, расположенных в разных местах диска. Это приводит к замедлению считывания файлов и повышает вероятность нарушения их целостности файла (например, при искажении ссылки на очередной кластер). Файл, который занимает на диске более одного непрерывного участка, называется фрагментированным.

Программа *дефрагментации диска* анализирует его и объединяет фрагментированные файлы и папки таким образом, чтобы каждый файл или папка занимали единое непрерывное пространство. В результате ускоряется чтение и запись файлов и папок. Одновременно объединяются в единое целое и свободные кластеры, что делает менее вероятной фрагментацию новых файлов. Время, необходимое для дефрагментации диска, зависит от его емкости, числа и размера файлов, степени фрагментации и доступных системных ресурсов.

Перед выполнением дефрагментации необходимо найти все фрагментированные файлы и папки, проанализировав диск. Затем можно оценить степень фрагментации диска и принять решение о целесообразности дефрагментации.

Степень фрагментации файлов и папок зависит в первую очередь от частоты и объема удаляемой с диска и записываемой на него информации. Поэтому анализ рекомендуется проводить регулярно, а дефрагментацию – после получения соответствующей рекомендации программы дефрагментации. Для дефрагментации дисков в ОС Windows имеется стандартная служебная утилита *Defrag*, в пакете NU – *Speed Disk*.

Программы уничтожения данных на магнитных дисках. Конфиденциальная информация должна не только храниться с соблюдением строгих правил, но и в случае необходимости надежно уничтожаться. Удаление файлов средствами операционной системы, форматирование разделов жестких дисков или их удаление не гарантируют невозможности восстановления информации специальными программными или аппаратными средствами. Существует несколько способов стирания данных с дисков:

- удаление файлов в специальную системную папку;
- удаление без помещения в системную папку;

- уничтожение файлов;
- форматирование дисков без возможности восстановления или очистка дисков.

При удалении файлов ОС обычно условно перемещает их в системную папку, что позволяет пользователю при необходимости их легко восстановить.

В случае удаления файлов без помещения в системную папку или при ее очистке операционная система не стирает данные с диска, а помечает пространство, занимаемое удаленными файлами, как свободное. Поэтому, если в освободившееся место не записаны новые данные, удаленную информацию в принципе можно восстановить на уровне файловой системы.

Процедура уничтожения файлов (shredding) предназначена для полного стирания пользователем данных с диска с исключением любой возможности их восстановления. Для этого используются специальные команды (например, Alt+Del в программе FAR manager) или программы (например, Shredder, которая стирает файлы, перезаписывая на их место несколько раз комбинацию случайных символов, а также уничтожает все резервные копии и временные файлы, связанные с удаляемым).

Обычное форматирование дисков, также как и удаление файлов, не исключает возможности восстановления информации. Для полного ее уничтожения необходимо использовать полное форматирование без возможности восстановления информации или очистку диска специальными программами (например, *Disk Cleanup*, из NU, *Acronis Drive Cleanser* или *Shredder*).

К программам восстановления файлов относятся средства, предназначенные для временного хранения удаленных файлов на диске, программы восстановления файлов и файловой системы.

Программы временного хранения удаленных файлов. Для снижения риска случайного уничтожения нужных для пользователя данных при удалении файлов используются средства их временного хранения. *Корзина* (Recycle Bin) – это системная папка операционной системы, предназначенная именно для этой цели. Файлы и папки при удалении перемещаются в корзину и хранятся в ней, продолжая занимать место на диске; при необходимости их можно вернуть на исходное место. При восстановлении автоматически создаются удаленные папки, из которых были удалены восстанавливаемые файлы.

Корзина имеет ограничение по максимальному объему занимаемого на диске места; этот объем задается пользователем при ее настройке. Когда отведенный объем корзины исчерпывается, новые удаленные файлы автоматически вытесняют из нее ранее удаленные. Поэтому вероятность восстановления пользователем удаленных файлов зависит от времени, прошедшего после удаления файла, объема данных, удаленных после него, и размера корзины. В любой момент времени (например, при нехватке места на жестком диске) пользователь может самостоятельно очистить корзину. Он может также полностью отключить ее использование. В этом случае (когда файлы удаляются без помещения в корзину) их нельзя извлечь из нее, но остается возможность их восстановления на уровне файловой системы.

Программа *Norton Protection* (из NU) предотвращает потерю файлов, которые не сохраняются в стандартной корзине ОС; она дает возможность восстановить копии перезаписанных файлов или файлов, удаленных по сети.

Программы восстановления файлов и файловой системы. Так как при удалении файлов без помещения в корзину или при ее очистке операционная система не стирает данные с диска, а лишь помечает пространство, занимаемое ими, как свободное, удаленную таким образом информацию можно восстановить. Главное при этом – чтобы на их место не были записаны новые данные. Впрочем, и в этом случае частичное восстановление возможно. Очень часто потерянную информацию можно восстановить даже после форматирования дисков, при повреждении файловой системы или появлении

неисправностей на носителях. Существует множество программ, созданных для этой цели: *Easy Recover*, *Tiramisu*, *Drive Rescue* и др.

Программы резервного копирования данных позволяют создавать резервные копии файлов на различных носителях информации или на другом компьютере, а также содержат надежные механизмы хранения, восстановления и менеджмента файлов (баз данных) на локальном компьютере или в локальной сети. Резервирование осуществляется в автоматизированном режиме, так что пользователь не обязан создавать резервные копии каких-либо данных вручную. Достаточно настроить планировщик заданий соответствующей программы, и все необходимые операции будут выполнены без участия пользователя. Многие программы позволяют делать сжатые резервные копии данных, проверяют результаты архивирования и генерируют отчеты; они особенно полезны для администраторов баз данных и серверов.

Программы записи и обслуживания компакт-дисков

Запись носителей CD или DVD можно выполнять на компьютере при помощи пишущих устройств (CD-RW и DVD+/-RW) и специальных служебных программ – *Easy CD Creator*, *WinOnCD*, *Nero Burning ROM*, *Direct CD* и др.

Процесс записи одной дорожки представляет собой единую операцию, которая не может быть прервана, иначе диск будет испорчен. Для равномерной передачи записываемой информации на лазер все приводы предварительно помещают данные в буфер. К сбою записи могут привести механические толчки привода и исчерпание данных в буфере, вызванное запуском параллельных процессов, зависанием программы записи или ОС.

Используются два основных метода записи на компакт-диски: сессионный (для CD-R и CD-RW) или с предварительным форматированием диска (только для CD-RW). Сессионную запись можно делать в одном из двух режимов: DAO (Disk At Once – весь диск за один прием) и TAO (Track At Once – одна дорожка за один прием). При записи методом TAO лазер включается в начале каждой дорожки и отключается в ее конце. При записи методом DAO он остается включенным в течение записи всего диска.

Диски, записанные за один прием, наиболее универсальны; их могут читать любые приводы CD-ROM, однако после записи на них невозможно дописывание новых данных. Этот режим используется при записи мастер-дисков для последующего тиражирования путем штампования.

В режиме TAO создают многосессионные диски формата CD-R, допускающие последующую дозапись данных; это также наиболее простой способ записи CD-DA с паузами между дорожками. Недостатком данного метода являются непроизводительные расходы (потери) дискового пространства: на первую сессию записи отводится 22,5 Мб на служебную информацию, на каждую последующую – 13,5 Мб. Если дозапись осуществляется более 10 раз, значительная часть емкости диска оказывается потерянной. Перед началом записи сессионным способом необходимо сформировать полный список входящих в сессию файлов; последующее добавление файлов на диск после записи возможно лишь в виде дополнительных сессий.

Метод с предварительным форматированием можно использовать только для записи RW-дисков. При этом сначала производится разбивка диска на секторы (форматирование), подобно магнитным дискам. После форматирования CD-RW может использоваться как обычный сменный диск; стандартные файловые операции копирования, удаления и переименования преобразуются драйвером привода CD-RW в серии операций перезаписи секторов. Благодаря этому для работы с дисками CD-RW не требуется специального программного обеспечения, кроме драйвера привода с поддержкой UDF (например, *Adaptec DirectCD*), программы начальной разметки (*Direct CD*) и обычного файлового менеджера.

4. Программы обслуживания ОС Windows

Программы оптимизации работы ОС позволяют повысить быстродействие компьютера за счет корректной установки и удаления программного обеспечения, более эффективного распределения оперативной памяти, оптимизации файлов реестра, отключения ненужных для использования компонент и служб операционной системы.

Чтобы программа правильно и без сбоев работала в ОС Windows, она должна быть корректно установлена на компьютер.

Установка программного обеспечения – это процедура добавления файлов и папок на жесткий диск и соответствующих данных в системный реестр Windows. После установки в меню и на рабочем столе появляются необходимые для запуска программ ярлыки, а сами программы настраиваются на работу с устройствами компьютера.

Программы устанавливаются с *дистрибутивных дисков* (как правило, это CD), на которых находится программа-установщик (инсталлятор) с совокупностью исходных файлов и папок (обычно в сжатом виде), а также разного рода дополнительная информация. Чаще всего инсталлятор носит имя *setup.exe* (или *install.exe*); для установки программы нужно запустить его на исполнение и следовать выдаваемым на экран инструкциям.

Неиспользуемые или неправильно работающие программы необходимо удалять из компьютера. *Удаление программного обеспечения (Uninstall, деинсталляция)* – это процедура удаления файлов и папок программы с жесткого диска и соответствующих данных из системного реестра Windows. К сожалению, стандартные средства Windows далеко не всегда осуществляют эту операцию корректно; зачастую в системных файлах остаются записи о давно удаленных программах. Со временем накапливающийся в конфигурационных файлах «мусор» начинает утяжелять систему, делая ее менее устойчивой и менее производительной.

Файлы реестра Windows представляют собой специализированную базу данных, в которой хранится информация об аппаратной и программной конфигурации компьютера. Программы оптимизации работы ОС (*RegCleaner, Customizer XP* и др.) проводят анализ правильности хранения данных в реестре, удаляют ссылки на отсутствующие программы и упорядочивают структуру данных, что приводит к определенному повышению быстродействия системы.

Некоторые программы (например, *Ashampoo Uninstaller*) специально предназначены для мониторинга изменений, происходящих при инсталляции программ, что облегчает в дальнейшем их корректное удаление из системы.

Существует ряд интегрированных программ (например, *WinXP Manager*), позволяющих осуществлять полную оптимизацию работы ОС: повышение скорости загрузки и выключения, очистку жесткого диска от ненужных файлов и реестра от ненужных ссылок, а также тонкую настройку параметров операционной системы (отключение неиспользуемых служб, ненужных компонентов и т.д.).

Для многих современных программ, в том числе для операционных систем, характерна большая сложность настройки; рядовому пользователю может потребоваться очень много времени, чтобы во всем этом разобраться. Поэтому появились программы тонкой настройки (*Tweak*), которые дают возможность пользователю в удобном и понятном для него диалоговом режиме изменять важные параметры работы ОС. Программы тонкой настройки существуют и для других видов ПО.

1.6 Лекция № 6 (2 часа).

Тема: «Основные понятия и принципы моделирования»

1.6.1 Вопросы лекции:

1. Понятие модели и моделирования

2. Назначение моделей
3. Основные этапы построения моделей
4. Понятие формализации

1.6.2 Краткое содержание вопросов:

1. Понятие модели и моделирования

Объекты и явления окружающего мира, даже самые на взгляд простые, на самом деле сложны. Чтобы понять, как объект или явление действует, приходится рассматривать их модели.

Модель – упрощенное представление, аналог реального объекта, процесса или явления. При построении модели сам объект называют *оригиналом*, или *прототипом*. Например, детская игрушка представляет реальный объект окружающего мира, но зачастую только одно или несколько свойств – внешний вид. Для любого объекта может существовать множество моделей, различных по сложности и степени сходства с оригиналом.

Модель необходима для того, чтобы:

- понять, как устроен реальный объект: каковы его структура, основные свойства, законы развития и взаимодействия с окружающим миром;
- научиться управлять объектом и процессом: определить наилучшие способы управления при заданных целях и критериях (оптимизация);
- прогнозировать прямые или косвенные последствия реализации заданных способов и форм воздействия на объект.

Моделирование – построение и изучение моделей в целях получения новых знаний или дальнейшего совершенствования характеристик объектов исследования.

Моделирование – это метод научного познания объективного мира с помощью моделей. Модель строится таким образом, чтобы она наиболее полно воспроизводила необходимые для изучения качества объекта. Модель должна быть проще объекта и удобнее для изучения. Таким образом, для одного и того же объекта могут существовать различные модели, классы моделей, соответствующие различным целям его изучения.

2. Назначение моделей

Моделировать можно внешний вид, структуру, поведение объекта, а также все возможные их комбинации. Это называется *аспект* моделирования.

Структурой объекта называют совокупность его элементов, а также существующих между ними связей.

Поведением объекта называют изменение его внешнего вида и структуры с течением времени в результате взаимодействия с другими объектами.

В процессе моделирования каждый аспект раскрывается через совокупность свойств, но отражаются только существенные с точки зрения целей моделирования.

Каждый аспект моделирования можно охарактеризовать набором свойств:

- внешний вид – набором признаков;
- структуру – перечнем элементов и указанием отношений между ними;
- поведение – изменением внешнего вида и структуры с течением времени.

Некоторые свойства объекта могут быть выражены числовыми значениями. Такие величины носят название *параметров модели*.

3. Основные этапы построения моделей

Основные этапы построения моделей

Процесс моделирования можно разбить на следующие этапы:

- определение цели моделирования;
- анализ объекта и выделение всех известных его свойств;
- анализ выделенных свойств с точки зрения цели моделирования и определение, какие из них следует считать существенными;
- выбор формы представления модели;
- формализация;
- анализ полученной модели на непротиворечивость;
- анализ адекватности полученной модели объекту и цели моделирования.

Если условия моделирования позволяют, то рекомендуется построить несколько моделей с разными наборами существенных свойств и затем оценить их на адекватность объекту и цели моделирования.

Классификация моделей:

Наиболее распространенными классификационными признаками для моделей являются:

- область использования;
- временной фактор (динамика);
- область знаний;
- способ представления.

Классификация моделей										
Способ представления								Вре- менной фактор	Об- ласть знаний	Область исполь- зования
Материаль- ные			Абстрактные							
Физические	Аналоговые	Пространственные	Мысленные (вербальные)	Информационные			Неформализованные	Статические	Динамические	Биологические, социальные, экономические, исторические...
				Формализован- ные						
				Знаковые						
				Аналитические	Статистические					
						Образные				
Учебные, опытные, научно-технические, игровые, имитационные										

Рис. 1. Классификация моделей

По области **использования** можно выделить следующие виды.

Учебные модели. К ним можно отнести обучающие программы, тренажеры.

Опытные модели – измененные в размерах копии проектируемых объектов, на основе которых исследуются и прогнозируются их будущие характеристики. К ним можно отнести модели корабля в бассейне, автомобиля или самолета в аэродинамической трубе и др.

Научно-технические модели используются для исследования процессов и явлений (стенд для тестирования телетехники, громоотвод и др.).

Игровые модели – это различные игры (военные, спортивные, деловые, экономические), проигрывающие реальные и потенциальные ситуации.

Имитационные модели представляют процессы реальной жизни с различной степенью точности. Эксперименты при различных исходных данных позволяют сделать выводы. Этот метод подбора правильного решения получил название «метод проб и ошибок».

По учету **фактора времени** выделяют следующие модели.

Статические, представляющие собой единовременный срез информации по данному объекту.

Динамические, представляющие картину изменения объекта во времени.

Один и тот же объект можно охарактеризовать и статической, и динамической моделями.

По области **знаний** или деятельности модели делятся на *биологические, социологические, экономические, исторические* и т.п.

По **способу представления** («из чего сделаны») модели делят на две группы: *материальные* и *абстрактные* (нематериальные). Последняя категория, в свою очередь, подразделяется на *мысленные* (вербальные) и *информационные* модели.

Обе группы содержат информацию об исходном объекте: цвет, форма, пропорция и т.п. В материальной модели эта информация имеет реальное воплощение, ее можно получить с помощью органов чувств и измерительными приборами. Исследователь материально воздействует на модель. Материальные модели могут быть *физическими, аналоговыми* и *пространственными*.

Физическое моделирование предназначено для воспроизводства динамики процессов, происходящих в реальных объектах (испытания объектов в аэротрубе).

Аналоговое моделирование основано на аналогии математического описания моделей и объектов, т.е. связано с использованием материальных моделей, имеющих иную физическую природу, чем изучаемый объект, но описывающихся теми же математическими соотношениями, что и изучаемый объект.

В пространственном моделировании используются модели, предназначенные для восприятия пространственных или геометрических свойств изучаемого объекта (макеты, глобусы).

В нематериальной модели та же информация представлена в абстрактной форме (мысль, формула, чертеж, схема) и основана на умозрительной связи между объектом и моделью. Этот вид моделей может быть *формализованным* и *неформализованным*.

В формализованном моделировании моделью служат системы знаков (знаковое моделирование) или образов (образное моделирование), вместе с коэффициентами задаются правила их преобразования и представления.

К *знаковому* моделированию относится математика, позволяющая для различных явлений применить одинаковые математические описания в виде совокупности формул, уравнений, подчиненных правилам логики и математики. В математических методах широко применяются *аналитические* и *статистические* модели. Каждый из этих типов имеет свои преимущества и недостатки.

Аналитические модели учитывают меньшее число факторов, всегда включают допущения и упрощения. Но результаты расчета отчетливее отображают основные закономерности явления.

Статистические модели более точны и подробны, позволяют учесть большее число факторов. К их недостаткам можно отнести громоздкость, плохую обобщимость и трудность поиска оптимального решения.

Наилучший результат получается при совместном применении аналитических и статистических моделей.

В *образном* моделировании модели строятся на наглядных элементах (фотография, рисунок, фильм или звукозапись). Анализ образных моделей осуществляется мысленно, поэтому они могут быть отнесены к формальному моделированию (например,

столкновение двух молекул, как соударение шаров). В физике широко используются «мысленные эксперименты».

Если отображение реальной действительности точно не зафиксировано (модель не формируется), а вместо нее используется нечто мысленное, служащее основой для рассуждения и принятия решения, то такой анализ проблем можно отнести к *неформализованному* моделированию.

Материальная и абстрактная модели взаимно дополняют друг друга.

4. Понятие формализации

В общем виде **формализация** понимается как сведение некоторого содержания (текста, научной теории, воспринимаемых сигналов и проч.) к выбранной форме описания. Например: книга – формализация средствами языковых конструкций мыслей автора; оглавление книги – это формализация ее содержательных частей; научная теория – совокупность текста, формул, графиков, схем, таблиц и т.д.

План действий в результате формализации переводится в алгоритм.

В основе понятия формализации лежит принцип существования принципиальной возможности разделения самого объекта и его обозначения, т.е. суть объекта не меняется от того, как он называется и описывается.

Языки описания бывают естественные и искусственные.

Язык характеризуется набором используемых знаков (слов, фраз, текстов), правилами образования из этих знаков языковых конструкций – синтаксических, семантических и прагматических.

Правила искусственного языка определяются строго и однозначно, поэтому такой язык называется **формализованным**.

1.7 Лекция № 7, 8, 9 (6 часов).

Тема: «Алгоритмизация вычислительных процессов»

1.7.1 Вопросы лекции:

1. Понятие алгоритмизации вычислительных процессов
2. Понятие алгоритма и его свойства, способы описания алгоритмов
3. Основные типы алгоритмов
 - 3.1. Алгоритм линейной структуры
 - 3.2. Алгоритм ветвящейся структуры
 - 3.3. Алгоритм циклической структуры

1.7.2 Краткое содержание вопросов:

1. Понятие алгоритмизации вычислительных процессов

Разработке алгоритма предшествуют такие этапы, как формализация и моделирование задачи. Формализация предполагает замену словесной формулировки решаемой задачи краткими символьными обозначениями, близкими к обозначениям в языках программирования или к математическим. Моделирование задачи является важнейшим этапом, целью которого является поиск общей концепции решения. Обычно моделирование выполняется путем выдвижения гипотез решения задачи и их проверке

любым рациональным способом (прикидочные расчеты, физическое моделирование и т.д.). Результатом каждой проверки является либо принятие гипотезы, либо отказ от нее и разработка новой.

При разработке алгоритма используют следующие основные принципы.

Принцип поэтапной детализации алгоритма (другое название - "проектирование сверху-вниз"). Этот принцип предполагает первоначальную разработку алгоритма в виде укрупненных блоков (разбиение задачи на подзадачи) и их постепенную детализацию.

Принцип "от главного к второстепенному", предполагающий составление алгоритма, начиная с главной конструкции. При этом, часто, приходится "достраивать" алгоритм в обратную сторону, например, от середины к началу.

Принцип структурирования, т.е. использования только типовых алгоритмических структур при построении алгоритма. Нетиповой структурой считается, например, циклическая конструкция, содержащая в теле цикла дополнительные выходы из цикла. В программировании нетиповые структуры появляются в результате злоупотребления командой безусловного перехода (GoTo). При этом программа хуже читается и труднее отлаживается.

Говоря о блок-схемах, как о средстве записи алгоритма, можно дать еще один совет по их разработке. Рекомендуется после внесения исправлений в блок-схему аккуратно перерисовывать ее с учетом этих исправлений. Аккуратность записи есть аккуратность мысли программиста. Аккуратно записанный и детализованный алгоритм упрощает его программирование и отладку.

Этапы решения задачи на компьютере.

Решение задачи разбивается на этапы:

1. Постановка задачи
2. Формализация (математическая постановка)
3. Выбор (или разработка) метода решения
4. Разработка алгоритма
5. Составление программы
6. Отладка программы
7. Вычисление и обработка результатов

1. При постановке задачи выясняется конечная цель и вырабатывается общий подход к решению задачи. Выясняется сколько решений имеет задача и имеет ли их вообще. Изучаются общие свойства рассматриваемого физического явления или объекта, анализируются возможности данной системы программирования.

2. На этом этапе все объекты задачи описываются на языке математики, выбирается форма хранения данных, составляются все необходимые формулы.

3. Выбор существующего или разработка нового метода решения (очень важен и, в то же время личностный этап).

4. На этом этапе метод решения записывается применительно к данной задаче на одном из алгоритмических языков (чаще на графическом).

5. Переводим решение задачи на язык, понятный машине.

2. Понятие алгоритма и его свойства, способы описания алгоритмов

"Алгоритм" является фундаментальным понятием информатики. Представление о нем необходимо для эффективного применения вычислительной техники к решению практических задач. Алгоритм - это предписание исполнителю (человеку или автомату) выполнить точно определенную последовательность действий, направленных на достижение заданной цели. Алгоритм - это сформулированное на некотором языке правило, указывающее на действия, последовательное выполнение которых приводит от исходных данных к искомому результату. Значение слова алгоритм очень схоже со

значением слов рецепт, процесс, метод, способ. Однако любой алгоритм, в отличие от рецепта или способа, обязательно обладает следующими свойствами.

Свойства алгоритма (отличающие его от любых других предписаний): понятность (для конкретного исполнителя); дискретность (команды последовательны, с точной фиксацией моментов начала и конца выполнения команды); точность (после выполнения каждой команды точно известно, завершено ли исполнение алгоритма или же какая команда должна выполняться следующей); результативность (после конечного числа шагов задача решается или же становится ясно, что процесс решения не может быть продолжен); массовость (алгоритм единым образом применяется к любой конкретной формулировке задачи, для которой он разработан).

1. Дискретность - разбиение алгоритма на ряд отдельных законченных действий - шагов. Выполнение алгоритма разбивается на последовательность законченных действий - шагов. Каждое действие должно быть закончено исполнителем алгоритма прежде, чем он приступит к исполнению следующего действия.

2. Точность - однозначные указания. На каждом шаге однозначно определено преобразование объектов среды исполнителя, полученной на предыдущих шагах алгоритма. Если алгоритм многократно применяется к одному и тому же набору исходных данных, то на выходе он получает каждый раз один и тот же результат. Запись алгоритма должна быть такой, чтобы на каждом шаге его выполнения было известно, какую команду надо выполнять следующей.

3. Понятность - однозначное понимание и исполнение каждого шага алгоритма его исполнителем. Алгоритм должен быть записан на понятном для исполнителя языке.

4. Результативность - обязательное получение результата за конечное число шагов. Каждый шаг (и алгоритм в целом) после своего завершения дает среду, в которой все объекты однозначно определены. Если это по каким-либо причинам невозможно, то алгоритм должен сообщать, что решение задачи не существует. Работа алгоритма должна быть завершена за конечное число шагов. Информатика оперирует только с конечными объектами и конечными процессами, поэтому вопрос о рассмотрении бесконечных алгоритмов остается за рамками теории алгоритмов.

5. Массовость - применение алгоритма к решению целого класса однотипных задач.

Порядок выполнения алгоритма:

- Действия в алгоритме выполняются в порядке их записи
- Нельзя менять местами никакие два действия алгоритма
- Нельзя не закончив одного действия переходить к следующему

Для записи алгоритмов используются специальные языки:

1. Естественный язык (словесная запись)
2. Формулы
3. Псевдокод
4. Структурограммы
5. Синтаксические диаграммы
6. Графический (язык блок-схем)

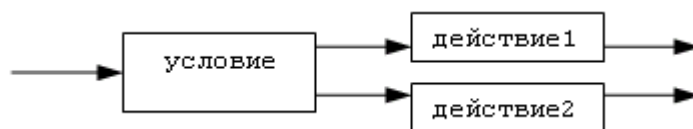
1. Естественный язык:

если условие то действие1 иначе действие2

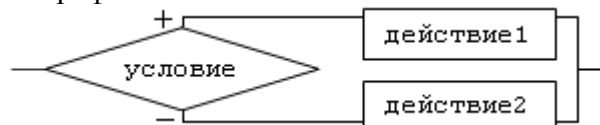
2. Структурограмма:



3. Синтаксическая диаграмма:



4. Графический язык:



Составление алгоритмов графическим способом подчиняется двум ГОСТам:

1. ГОСТ 19.002-80, соответствует международному стандарту ИСО 2636-73. Регламентирует правила составления блок-схем.

2. ГОСТ 19.003-80, соответствует международному стандарту ИСО 1028-73. Регламентирует использование графических примитивов.

Название	Символ (рисунок)	Выполняемая функция (пояснение)
1. Блок вычислений		Выполняет вычислительное действие или группу действий
2. Логический блок		Выбор направления выполнения алгоритма в зависимости от условия
3. Блоки ввода/вывода		Ввод или вывод данных вне зависимости от физического носителя
		Вывод данных на печатающее устройство
4. Начало/конец (вход/выход)		Начало или конец программы, вход или выход в подпрограмму
5. Предопределенный процесс		Вычисления по стандартной или пользовательской подпрограмме
6. Блок модификации		Выполнение действий, изменяющих пункты алгоритма
7. Соединитель		Указание связи между прерванными линиями в пределах одной страницы
8. Межстраничный соединитель		Указание связи между частями схемы, расположенной на разных страницах

Правила построения блок-схем:

Блок-схема выстраивается в одном направлении либо сверху вниз, либо слева направо

Все повороты соединительных линий выполняются под углом 90 градусов

3. Основные типы алгоритмов

3.1. Алгоритм линейной структуры

Линейным называется **вычислительный процесс, в котором предусматривается получение результата путем однократного выполнения последовательности действий при любых значениях исходных данных.** Характерной особенностью линейного вычислительного процесса является то, что направление вычислений не зависит от исходных данных и промежуточных результатов.

Алгоритм линейного вычислительного процесса графически может быть представлен **блоком следования – композицией** (объединением) нескольких следующих друг за другом блоков процесс (рис. 1).



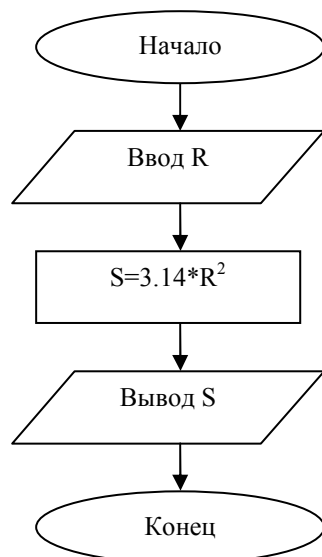
Рисунок 1- Блок-схема линейного алгоритма

Линии, соединяющие отдельные блоки на блок-схеме алгоритма и указывающие на последовательность действий, называются **линиями потока**. Направление линии потока сверху вниз принимается за основное и стрелкой не обозначается.

Рассмотрим пример задачи, решение которой представляет собой линейный вычислительный процесс, и составим блок-схему алгоритма.

Пример: Найдите площадь круга S при заданном значении радиуса R . Для вычисления площади использовать формулу $S=3,14 \cdot R^2$.

Блок-схема



Как правило, составление алгоритма линейного вычислительного процесса не вызывает затруднений, однако в "чистом виде" такие вычислительные процессы встречаются весьма редко. Чаще всего они являются составной частью более сложных вычислительных процессов. Таким образом:

- линейный вычислительный процесс является наиболее простым при реализации его на ЭВМ.
- блок-схема алгоритма линейного вычислительного процесса представляет собой **композицию** (объединение) нескольких следующих друг за другом блоков процесс.

3.2. Алгоритм ветвящейся структуры

Разветвляющимся вычислительным процессом называется процесс, направление вычислений в котором зависит от результата проверки некоторого условия (условий).

Алгоритм разветвляющегося вычислительного процесса предусматривает **выбор** одной из нескольких возможных **альтернатив** (последовательностей действий) в зависимости от значения исходных данных или промежуточных результатов. Каждую из

этих последовательностей называют **ветвью алгоритма**. Блок-схема алгоритма разветвляющегося вычислительного процесса содержит, по крайней мере, один блок РЕШЕНИЕ. Направления линий потока сверху вниз и слева направо на блок-схеме принимаются за основные и стрелками не обозначаются.

Графической интерпретацией алгоритма разветвляющегося вычислительного процесса является **блок разветвления алгоритма**, в котором может быть предусмотрен **полный** (рис. 2) или **неполный** (рис. 3) **выбор**.

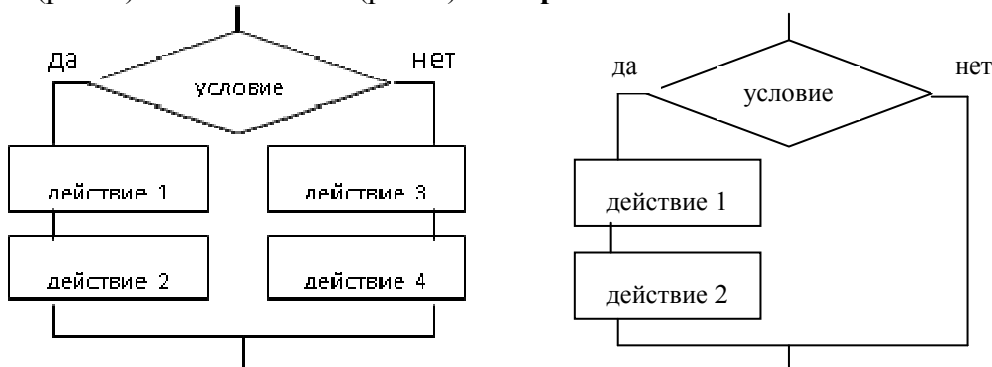


Рис. 2 Блок разветвления с полным выбором Рис. 3 Блок разветвления с неполным выбором

В блоке разветвления алгоритма с полным выбором в зависимости от результата проверки условия выполняются только действия ветви "да" (т. е., действия 1 и 2) или только действия ветви "нет" (действия 3 и 4). В блоке с неполным выбором в зависимости от результата проверки условия либо выполняются действия какой-либо ветви, либо они игнорируются.

Некоторой разновидностью блока разветвления алгоритма является **блок множественного выбора** (рис. 3). В нем, в зависимости от результатов выбора, выполняется одно из предусмотренных действий.

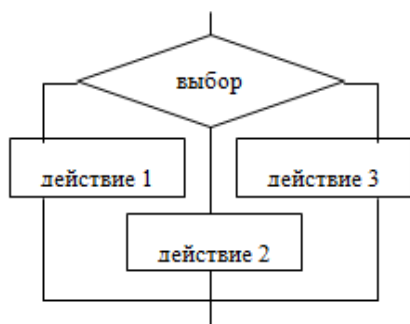


Рисунок 3— Блок множественного выбора

Рассмотрим пример задачи, алгоритм решения которой носит разветвляющийся характер.

Пример 2. Составить схему алгоритма для вычисления значений функции

$$y = \begin{cases} 2x + a, & \text{если } x < 0, \\ x + 3b^2, & \text{если } x \geq 0. \end{cases}$$

Поскольку значение функции вычисляется по разным формулам в зависимости от знака аргумента x , в алгоритме вычислительного процесса должна быть предусмотрена проверка знака названного аргумента. Блок-схема алгоритма решения задачи на ЭВМ представлена на рис. 4.

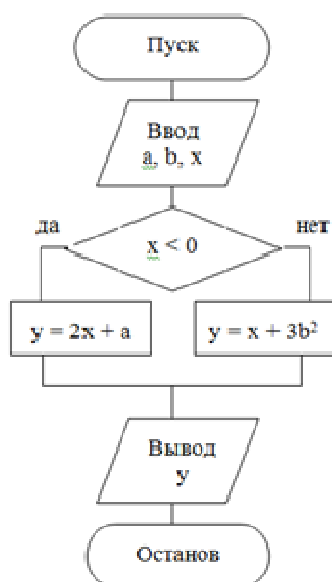


Рисунок 4 – Блок-схема алгоритма

В разветвляющихся вычислительных процессах могут иметь место так называемые "пустые" ветви. В направлении пустых ветвей задача, как правило, не имеет решения (вычислительный процесс не приводит к получению какого-либо промежуточного или конечного результата). Рассмотрим следующий пример.

Пример 3. Составить блок-схему алгоритма для вычисления значений функции

$$y = \begin{cases} x - a, & \text{если } x = 0, \\ x + b^2, & \text{если } 0 < x < 3, \\ x^2 + c, & \text{если } 3 < x < 5. \end{cases}$$

Как видно из условия задачи, переменная x может принимать только значения, находящиеся в диапазоне от 0 до 5. Следовательно, для остальных значений x ветви в алгоритме оказываются пустыми. В блок-схеме алгоритма должны быть предусмотрены действия ЭВМ на случай появления исходных данных (значений x), не описанных логическими условиями. На рис. 6 показан один из возможных способов решения проблемы.

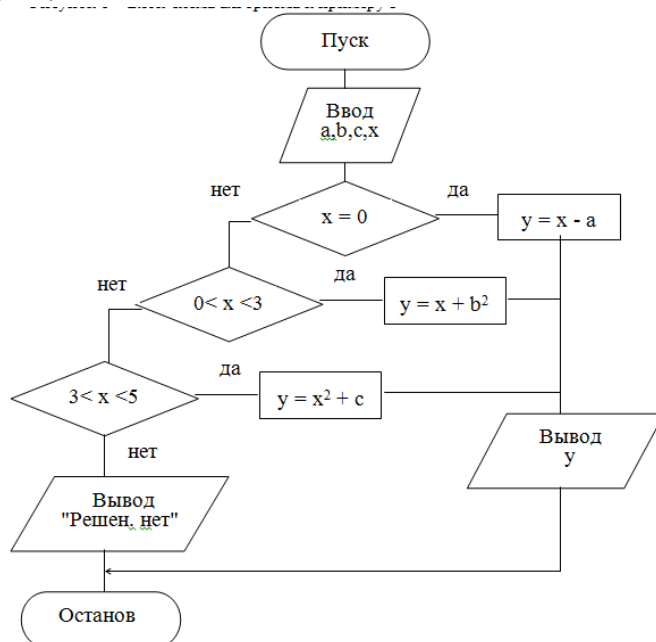


Рисунок 5 – Блок-схема алгоритма к примеру 3

В заключение кратко подведем итог:

1. Алгоритм разветвляющегося вычислительного процесса предусматривает выбор одной из нескольких возможных альтернатив в зависимости от значения исходных данных или промежуточных результатов.

2. В блоке разветвления алгоритма может быть предусмотрен полный или неполный выбор.

3. В разветвляющихся вычислительных процессах могут иметь место "пустые" ветви.

3.3. Алгоритм циклической структуры

Циклическим называется вычислительный процесс, в котором получение результата обеспечивается путем **многократного повторения некоторой последовательности действий**.

Графической интерпретацией алгоритма циклического вычислительного процесса является **блок цикла**. Различают несколько разновидностей блока цикла: **блок цикла с параметром**, **блок цикла с предварительным условием** и **блок цикла с последующим условием**.

Блок-схема блока цикла с параметром представлена на рис. 6.

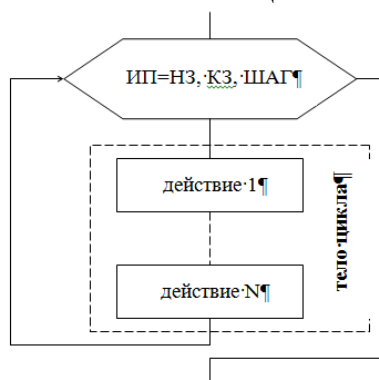


Рисунок 6 – Блок цикла с параметром

На рис. 6 приняты следующие сокращения:

ИП – имя ячейки памяти, в которую заносится значение параметра;

НЗ – начальное значение параметра;

КЗ – конечное значение параметра;

ШАГ – величина приращения параметра после каждого выполнения тела цикла.

Тело цикла представляет собой линейный вычислительный процесс и выполняется столько раз, сколько разных значений примет параметр в заданных пределах от НЗ до КЗ. Блок цикла с параметром относится к циклу **с явно выраженным числом повторений** (число повторений известно заранее). Для таких циклов характерным является то, что задаются:

- **начальное и конечное значения параметра** цикла;
- **закон изменения параметра цикла** при каждом повторном выполнении тела цикла;
- **количество повторных выполнений** тела цикла (вытекает из первых двух пунктов).

Блок цикла с предварительным условием и блок цикла с последующим условием относятся к так называемым **итерационным циклам**. В таких циклических вычислительных процессах число повторений тела цикла заранее не известно. Выход из цикла осуществляется не после того, как цикл повторится заданное число раз, а при выполнении определенного условия, связанного с проверкой значения монотонно изменяющейся в теле цикла величины. Блок-схема блока цикла с предварительным условием представлена на рис. 8, а блока цикла с последующим условием – на рис. 7.

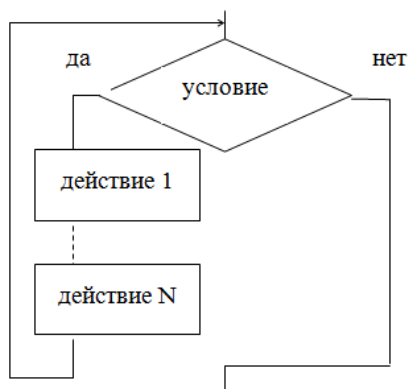


Рисунок 7 – Блок цикла с предварительным условием

Кратко суть алгоритма цикла с предварительным условием можно изложить следующим образом: **пока выполняется условие – повторять действия**. В таких циклах возможны ситуации, когда тело цикла не выполняется ни разу (например, если при первой же проверке не выполняется условие, то сразу происходит выход из цикла).

В цикле с последующим условием (рис. 8) тело цикла выполняется не менее одного раза. При этом **действия**, предусмотренные в теле цикла, **выполняются до тех пор, пока не выполнится заданное условие**.

Рассмотренные блоки циклов позволяют описать **простые** циклические вычислительные процессы. При решении сложных задач может возникнуть необходимость внутри одного цикла организовать дополнительно один или несколько циклов. Такие циклы называются **вложенными**. При этом цикл, внутри которого создается другой цикл, называется **внешним**, а цикл, создаваемый внутри другого – **внутренним**. Правила организации как внешнего, так и внутреннего циклов те же, что и для простых циклов. Параметры внешнего и внутреннего циклов должны быть разными.

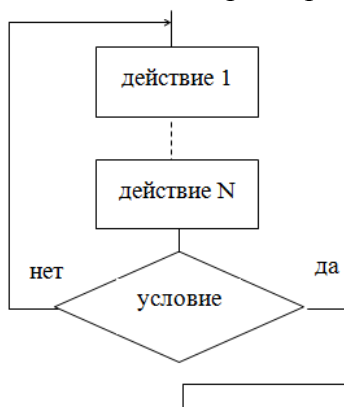


Рисунок 8 - Блок цикла с последующим условием

Рассмотрим пример алгоритмизации циклического вычислительного процесса.

Пример: Разработать алгоритм расчета значения функции по формуле $y = (a + b)^2$, при значениях a из интервала $[-5, 5]$ с шагом $+1$

Блок-схема алгоритма решения задачи на ЭВМ представлена на рис. 9.

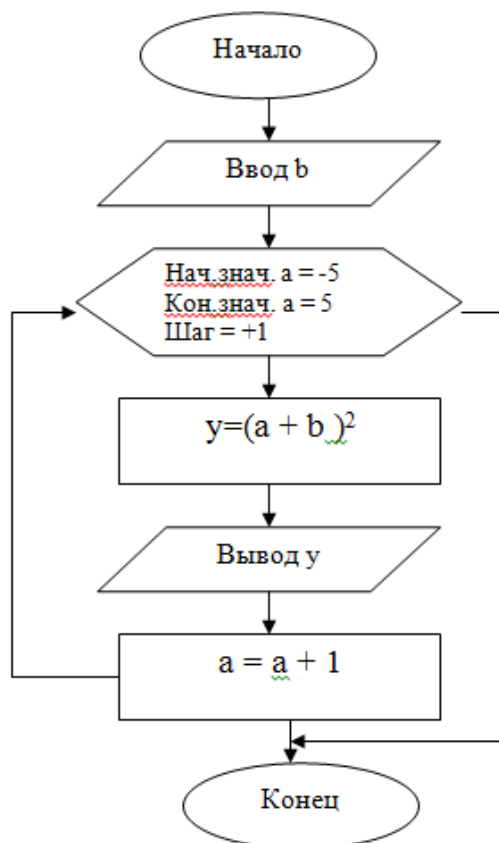


Рисунок 9 – Блок-схема алгоритма

Подведем итог:

1. В циклическом вычислительном процессе получение результата обеспечивается путем многократного повторения некоторой последовательности действий.
2. Для описания циклических вычислительных процессов в блок-схемах алгоритмов используют блоки циклов. Различают блок цикла с параметром, блок цикла с предварительным условием и блок цикла с последующим условием.
3. Циклические вычислительные процессы бывают простые и вложенные.

1.8 Лекция № 10, 11 (4 часа).

Тема: «Массивы. Обработка одномерных массивов»

1.8.1 Вопросы лекции:

1. Понятие массивов.
2. Стандартные алгоритмы обработки элементов одномерных массивов
 - 2.1. ввода-вывода элементов одномерного массива
 - 2.2. расчета суммы (произведения)
 - 2.3. организации различных счетчиков
 - 2.4. выбора максимального (минимального) элемента массива
 - 2.5. упорядочивания элементов массива по возрастанию (убыванию) их значений

1.8.2 Краткое содержание вопросов:

1. Понятие массивов.

Многие задачи, которые решаются с помощью ЭВМ, связаны с обработкой больших объемов информации, представляющей совокупность данных, объединенных единым математическим содержанием или связанных между собой по смыслу. Примером таких организованных совокупностей данных являются координаты, задающие положение точки в пространстве, матрица коэффициентов, определяющая систему линейных уравнений, значения некоторой функции в произвольных точках, коэффициенты многочлена и т. д. Такие данные удобно представлять в виде линейных или прямоугольных таблиц.

В линейной таблице каждому ее элементу соответствует порядковый номер. Для элемента прямоугольной таблицы должны быть указаны два номера: номер по вертикали (номер строки) и номер по горизонтали (номер столбца). В высшей математике табличные величины называют соответственно векторами и матрицами.

В программе для представления таких данных используются массивы.

Массив — это упорядоченная совокупность однотипных данных, с каждым из которых связан упорядоченный набор целых чисел, называемых индексами. Массив характеризуется именем, типом и размером.

Имя массива образуется по общему правилу образования имен, т. е. представляет собой идентификатор, например A, B1 C8 и т. д. Однако оно не должно совпадать с именем ни одной простой переменной, используемой в той же программе.

Переменная с индексами

Работа с массивом сводится к действиям над его элементами. Для того чтобы указать, какой элемент в данный момент используется, достаточно задать его порядковый номер, который приписывается к имени соответствующего массива. Таким образом, элементы массива обозначаются переменной с индексами. Запись переменной с индексами состоит из имени массива и следующего за ним в круглых скобках списка индексов, например A(1), A(I), B1(K), C8(I, J), C8(2, 1)

В качестве индексов можно использовать константы, переменные, а также арифметические выражения, называемые индексными выражениями, например A1(K+5), Q3(3*A+B, C(I)-2). Если в качестве индекса используется выражение, то перед обращением к элементу массива (его записью в программе) осуществляется вычисление индексного выражения и выделение целой части. Естественно, что все переменные, входящие в выражение, а также переменные, используемые в качестве индексов, должны быть заданы до обращения. Например, запись T(2*Y-3.2, Z+1.7) при $y = 1,8$ и $z = 0,9$ соответствует элементу T(0,2).

Индексы определяют положение элемента в массиве. Число индексов определяет размерность массива, т. е. форму его компоновки: одномерный, двумерный и т. д. Одномерный массив соответствует линейной таблице. Его элемент обозначается переменной с одним индексом: A(1), A(1) - соответственно первый и 2-й элементы одномерного массива A; , B1(K) — k-й элемент одномерного массива B1. Двумерный массив описывает в программе прямоугольную таблицу. Его элементы обозначаются переменной с двумя индексами: C8(I, J), C8(2, 1), где первый индекс обозначает номер строки, а второй — номер столбца. В языке Бейсик допускается работа с одномерными и двумерными массивами.

Таким образом, для обращения к конкретному элементу массива необходимо указать имя массива и значения индексов. Напомним, что если в качестве индексов используются переменные или выражения, то значения всех указанных переменных должны быть определены до обращения.

Описание массива

Для записи элементов массива в память ЭВМ нужно выделить для их хранения необходимое количество (массив) ячеек памяти, которое определяется размером массива.

Размеры массива задаются границами изменения индексов по каждому измерению. Минимальное значение индексов равно нулю. С учетом этого количество элементов в одномерном массиве будет равно значению верхней границы индекса плюс единица, а двумерного — произведению увеличенных на единицу значений верхних границ индексов.

В программе для каждого массива должны быть указаны его параметры: имя, размерность и размеры. Эта информация нужна для резервирования необходимого объема памяти для хранения числовых значений; она задается специальным оператором описания массивов DIM (сокращение от DIMENSION — размерность). Оператор DIM в общем случае имеет следующий вид:

<n> DIM V1($i_1[j_1]$), V2($i_2[j_2]$), ..., V $_k$ ($i_k[j_k]$)

где n — номер строки; DIM — имя оператора; V1, V2, ..., V $_k$ — имена массивов; i_1 , j_1 , i_2 , j_2 , ..., i_k , j_k — верхние границы индексов, представляющие собой положительные целые константы; [, j_k] — квадратные скобки здесь указывают на необязательность элемента. Например, если в программе используется одномерный массив A, содержащий 10 элементов, и двумерный массив B1, состоящий из трех строк и пяти столбцов, то оператор описания массивов будет иметь вид

10 DIM A(9), B1(2, 4)

Оператор DIM может быть записан в любом месте программы до первого оператора, использующего соответствующую индексную переменную. Однако наиболее употребительной является такая структура, при которой все описания объединяются в начале программы. Выполняя оператор DIM, машина выделяет для массива соответствующее количество ячеек памяти, причем имя массива определяет адрес первой ячейки участка памяти, хранящего данный массив. Элементы массивов располагаются в памяти ЭВМ один за другим в последовательных ячейках памяти. Строгая последовательность записи элементов исключает необходимость хранения порядковых номеров элементов.

2. Стандартные алгоритмы обработки элементов одномерных массивов

2.1. ввода-вывода элементов одномерного массива

При вводе массива необходимо последовательно вводить 1-й, 2-й, 3-й и т.д. элементы массива, аналогичным образом поступить и при выводе. Следовательно, необходимо организовать цикл.

Блок-схемы алгоритмов ввода элементов массива изображены на рис. 10-11.

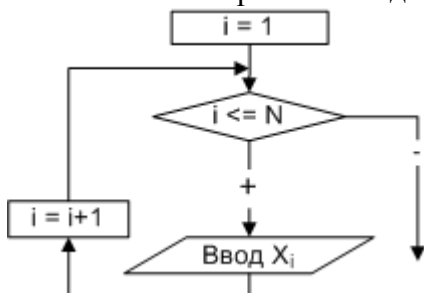


Рис 10 Алгоритм ввода массива с использованием цикла с предусловием

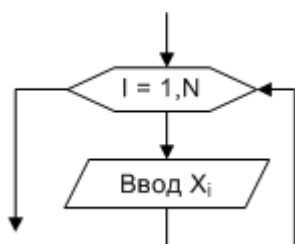


Рис. 11. Алгоритм ввода массива с использованием безусловного цикла

Как видно, безусловный цикл удобно использовать для обработки всего массива, и в дальнейшем при выполнении таких операций будем применять именно его. Вывод массива организуется аналогично вводу.

Рассмотрим несколько примеров обработки массивов. Алгоритмы, с помощью которых обрабатывают одномерные массивы, похожи на обработку последовательностей (вычисление суммы, произведения, поиск элементов по определенному признаку, выборки и т. д.). Отличие заключается в том, что в массиве одновременно доступны все его компоненты, поэтому становится возможной, например, сортировка его элементов и другие, более сложные преобразования.

2.2. расчета суммы (произведения)

Дан массив X , состоящий из n элементов. Найти сумму элементов этого массива. Процесс накапливания суммы элементов массива достаточно прост и практически ничем не отличается от суммирования значений некоторой числовой последовательности. Переменной S присваивается значение равное нулю, затем последовательно суммируются элементы массива X . Блок-схема алгоритма расчета суммы приведена на рис. 12.

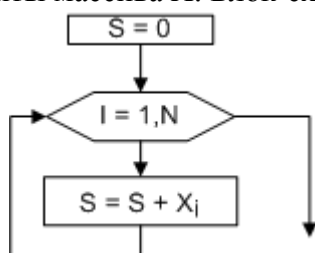


Рис. 12. Алгоритм вычисления суммы элементов массива

Дан массив X , состоящий из n элементов. Найти произведение элементов этого массива. Решение этой задачи сводится к тому, что значение переменной P , в которую предварительно была записана единица, последовательно умножается на значение i -го элемента массива. Блок-схема алгоритма приведена на рис. 13.

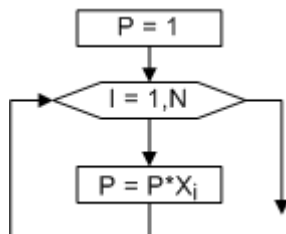
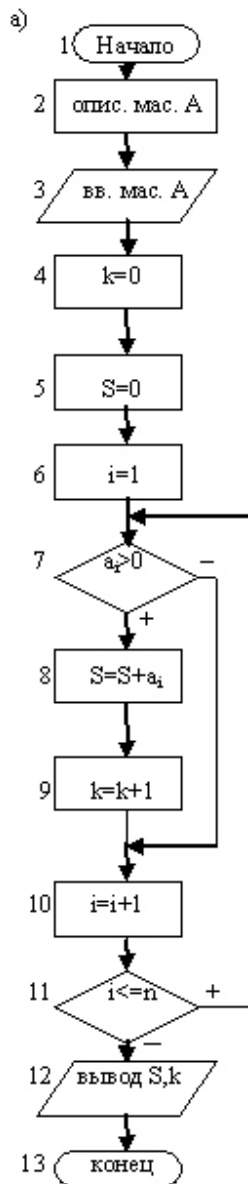


Рис. 13. Вычисление произведения элементов массива

2.3. организации различных счетчиков

Дан массив A , состоящий из n элементов. Найти сумму элементов этого массива и подсчитать количество положительных элементов. Для решения этой задачи надо ввести переменную k для получения количества положительных элементов и перед циклом присвоить ей значение 0. После блока проверки 7 по пути “+” нужно поставить блок, содержащий $k=k+1$, который ведет счет количества положительных элементов массива A .



2.4. выбора максимального (минимального) элемента массива

Поиск максимального элемента в массиве и его номера

Дан массив X , состоящий из n элементов. Найти максимальный элемент массива и номер, под которым он хранится в массиве.

Алгоритм решения задачи следующий. Пусть в переменной с именем Max хранится значение максимального элемента массива, а в переменной с именем $Nmax$ - его номер. Предположим, что первый элемент массива является максимальным, и запишем его в переменную Max , а в $Nmax$ занесем его номер, то есть - 1. Затем все элементы, начиная со второго, сравниваем в цикле с максимальным. Если текущий элемент массива оказывается больше максимального, то записываем его в переменную Max , а в переменную $Nmax$ - текущее значение индекса i . Процесс определения максимального элемента в массиве приведен в таблице 1 и изображен при помощи блок-схемы на рис. 14.

Таблица 1. Определение максимального элемента и его номера в массиве

Номера элементов	1	2	3	4	5	6	7
Исходный массив	4	7	3	8	9	2	5
Значение переменной Max	4	7	7	8	9	9	9
Значение переменной $Nmax$	1	2	2	4	5	5	5

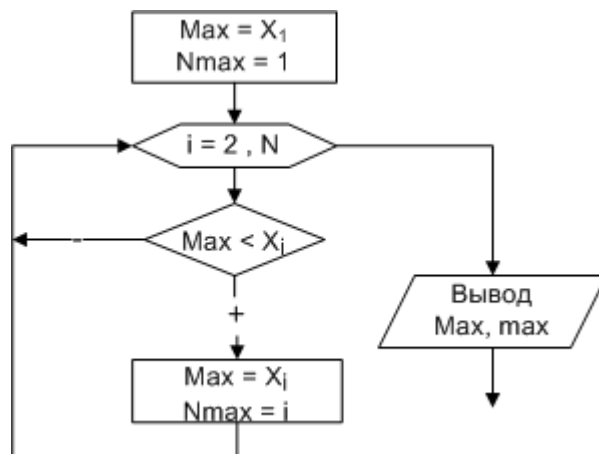


Рис. 14. Поиск максимального элемента и его номера в массиве

Совет. Алгоритм поиска минимального элемента в массиве будет отличаться от приведенного выше лишь тем, что в условном блоке знак поменяется с $>$ на $<$.

2.5. упорядочивания элементов массива по возрастанию (убыванию) их значений

Сортировка представляет собой процесс упорядочения элементов в массиве в порядке возрастания или убывания их значений. Например, массив X из n элементов будет отсортирован в порядке возрастания значений его элементов, если $X_1 \leq X_2 \leq \dots \leq X_n$, и в порядке убывания, если $X_1 \geq X_2 \geq \dots \geq X_n$.

Существует большое количество алгоритмов сортировки, но все они базируются на трех основных:

- сортировка обменом;
- сортировка выбором;
- сортировка вставкой.

Представим, что нам необходимо разложить по порядку карты в колоде. Для сортировки карт обменом можно разложить карты на столе лицевой стороной вверх и менять местами те карты, которые расположены в неправильном порядке, делая это до тех пор, пока колода карт не станет упорядоченной.

Для сортировки выбором из разложенных на столе карт выбирают самую младшую (старшую) карту и держат ее в руках. Затем из оставшихся карт вновь выбрать наименьшую (наибольшую) по значению карту и помещают ее позади той карты, которая была выбрана первой. Этот процесс повторяется до тех пор, пока вся колода не окажется в руках. Поскольку каждый раз выбирается наименьшая (наибольшая) по значению карта из оставшихся на столе карт, по завершению такого процесса карты будут отсортированы по возрастанию (убыванию).

Для сортировки вставкой из колоды берут две карты и располагают их в необходимом порядке по отношению друг к другу. Каждая следующая карта, взятая из колоды, должна быть установлена на соответствующее место по отношению к уже упорядоченным картам.

Итак, решим следующую задачу. Задан массив Y из n целых чисел. Расположить элементы массива в порядке возрастания их значений.

Сортировка методом "пузырька"

Сортировка пузырьковым методом является наиболее известной. Ее популярность объясняется запоминающимся названием, которое происходит из-за подобия процессу движения пузырьков в резервуаре с водой, когда каждый пузырек находит свой собственный уровень, и простотой алгоритма.

Сортировка методом "пузырька" использует метод обменной сортировки и основана на выполнении в цикле операций сравнения и при необходимости обмена соседних элементов. Рассмотрим алгоритм пузырьковой сортировки более подробно.

Сравним первый элемент массива со вторым, если первый окажется больше

второго, то поменяем их местами. Те же действия выполним для второго и третьего, третьего и четвертого, i -го и $(i+1)$ -го, $(n-1)$ -го и n -го элементов. В результате этих действий самый большой элемент станет на последнее n -е место. Теперь повторим данный алгоритм сначала, но последний n -й элемент, рассматривать не будем, так как он уже занял свое место. После проведения данной операции самый большой элемент оставшегося массива станет на $(n-1)$ -е место. Так повторяем до тех пор, пока не упорядочим весь массив.

В табл. 2 подробно расписан процесс упорядочивания элементов в массиве. Нетрудно заметить, что для преобразования массива, состоящего из n элементов, необходимо просмотреть его $n-1$ раз, каждый раз уменьшая диапазон просмотра на один элемент. Блок-схема описанного алгоритма приведена на рис. 15. Обратите внимание на то, что для перестановки элементов (блок 4) используется буферная переменная b , в которой временно хранится значение элемента, подлежащего замене.

Таблица 2. Процесс упорядочивания элементов в массиве по возрастанию

Номер элемента	1	2	3	4	5
Исходный массив	7	3	5	4	2
Первый просмотр	3	5	4	2	7
Второй просмотр	3	4	2	5	7
Третий просмотр	3	2	4	5	7
Четвертый просмотр	2	3	4	5	7

Совет. Для перестановки элементов в массиве по убыванию их значений необходимо при сравнении элементов массива заменить знак $>$ на $<$.

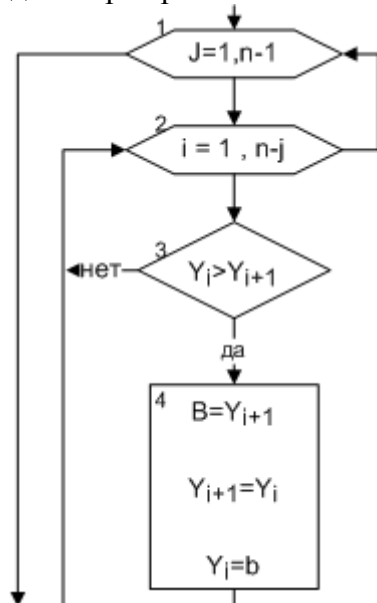


Рис. 15. Сортировка массива пузырьковым методом

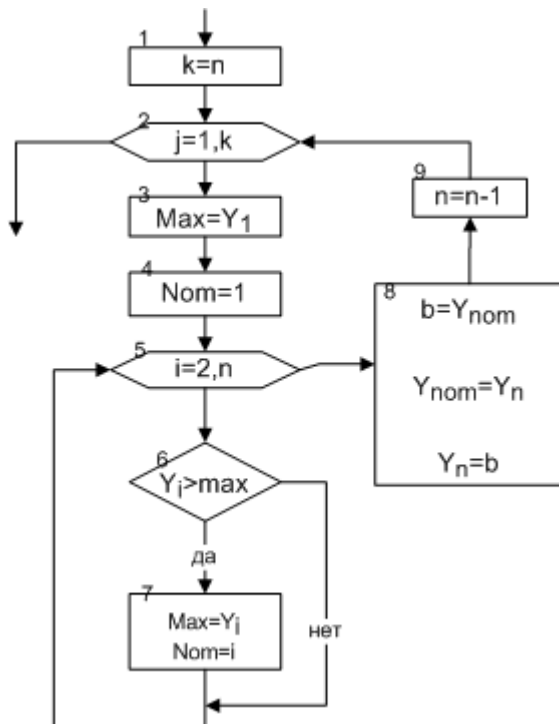


Рис. 16. Сортировка массива выбором наибольшего элемента

Сортировка выбором

Алгоритм сортировки выбором приведен в виде блок-схемы на рис. 16. Найдем в массиве самый большой элемент (блоки 3-7) и поменяем его местами с последним элементом (блок 8). Повторим алгоритм поиска максимального элемента, уменьшив количество просматриваемых элементов на единицу (блок 9), и поменяем его местами с предпоследним элементом (блоки 3-7). Описанную выше операцию поиска проводим до полного упорядочивания элементов в массиве. Так как в блоке 9 происходит изменение переменной n , то в начале алгоритма ее значение необходимо сохранить (блок 1).

Совет. Для упорядочивания массива по убыванию необходимо перемещать минимальный элемент.

2.6. удаления (добавления) элемента массива

Сортировка вставкой

Сортировка вставкой заключается в том, что сначала упорядочиваются два элемента массива. Затем делается вставка третьего элемента в соответствующее место по отношению к первым двум элементам. Четвертый элемент помещают в список из уже упорядоченных трех элементов. Этот процесс повторяется до тех пор, пока все элементы не будут упорядочены.

Прежде чем приступить к составлению блок-схемы рассмотрим следующий пример. Пусть известно, что в массиве из восьми элементов первые шесть уже упорядочены, а седьмой элемент нужно вставить между вторым и четвертым. Сохраним седьмой элемент во вспомогательной переменной, так как показано на рисунке 17, а на его место запишем шестой. Далее пятый переместим на место шестого, четвертый на место пятого, а третий на место четвертого, тем самым, выполнив сдвиг элементов массива на одну позицию вправо. Записав содержимое вспомогательной переменной в третью позицию, достигнем нужного результата.

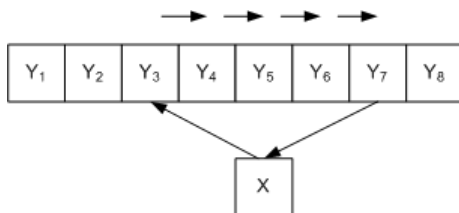


Рис. 17. Процесс вставки элемента в массив

Составим блок-схему алгоритма (рис. 18), учитывая, что возможно описанные выше действия придется выполнять неоднократно.

Организуем цикл для просмотра всех элементов массива, начиная со второго (блок 4). Сохраним значение текущего i -го элемента во вспомогательной переменной X , так как оно может быть потеряно при сдвиге элементов (блок 5) и присвоим переменной j значение индекса предыдущего $(i-1)$ -го элемента массива (блок 6). Далее движемся по массиву влево в поисках элемента меньшего, чем текущий и пока он не найден сдвигаем элементы вправо на одну позицию. Для этого организуем цикл (блок 7), который прекратиться, как только будет найден элемент меньше текущего. Если такого элемента в массиве не найдется и переменная j станет равной нулю, то это будет означать, что достигнута левая граница массива, и текущий элемент необходимо установить в первую позицию. Смещение элементов массива вправо на одну позицию выполняется в блоке 8, а изменение счетчика j в блоке 9. Блок 10 выполняет вставку текущего элемента в соответствующую позицию.

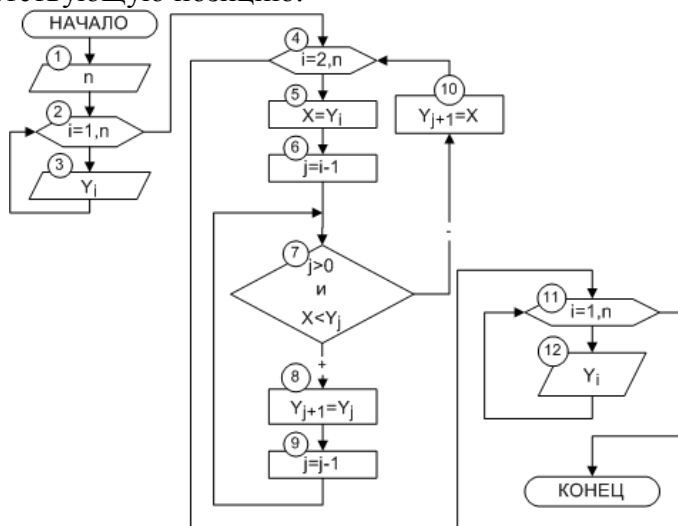


Рис. 18. Сортировка массива вставкой

Удаление элемента из массива

Необходимо удалить из массива X , состоящего из n элементов, m -й по номеру элемент. Для этого достаточно записать элемент $(m+1)$ на место элемента m , $(m+2)$ - на место $(m+1)$ и т.д., n - на место $(n-1)$ и при дальнейшей работе с этим массивом использовать $n-1$ элемент (рис. 19).

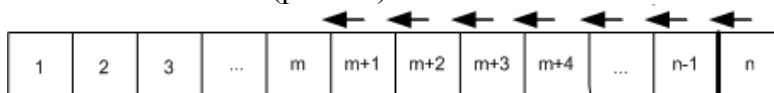


Рис. 19. Процесс удаления элемента из массива

Алгоритм удаления из массива X размером n элемента с номером m приведен на рис. 20.

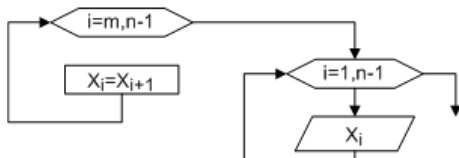


Рис. 20. Алгоритм удаления элемента из массива

Примеры алгоритмов обработки массивов

ПРИМЕР 3.1. Дан массив A состоящий из k целых положительных чисел. Записать все четные по значению элементы массива A в массив B .

Решение задачи заключается в следующем. Последовательно перебираются элементы массива A . Если среди них находятся четные, то они записываются в массив B . На рисунке 21 видно, что первый четный элемент хранится в массиве A под номером три, второй и третий под номерами пять и шесть соответственно, а четвертый под номером восемь. В массиве B этим элементам присваиваются совершенно иные номера. Поэтому для их формирования необходимо определить дополнительную переменную. В блок-схеме приведенной на рисунке 22 роль такой переменной выполняет переменная m . Операция, выполняемая в блоке 2, означает, что в массиве может не быть искоемых элементов. Если же условие в блоке 5 выполняется, то переменная m увеличивается на единицу, а значение элемента массива A записывается в массив B под номером m (блок 6). Условный блок 7 необходим для того, чтобы проверить выполнилось ли хотя бы раз условие поиска (блок 5).

i	1	2	3	4	5	6	7	8
A	3	7	4	9	2	8	1	6

B	4	2	8	6
m	1	2	3	4

Рис. 21 Процесс формирования массива B из элементов массива A

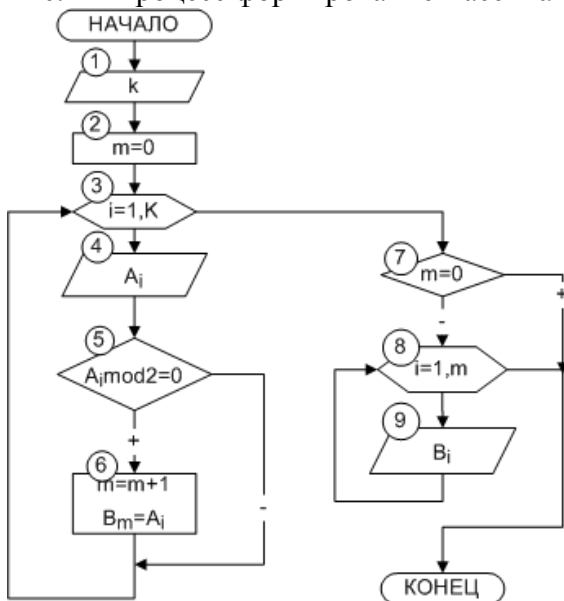


Рис. 22 Формирование массива B из соответствующих элементов массива A

1.9 Лекция № 12 (2 часа).

Тема: «Программные средства реализации информационных процессов. Инструментальное программное обеспечение»

1.9.1 Вопросы лекции:

1. Сущность, понятие, задачи и функции инструментального программного обеспечения
2. Системы программирования. Трансляторы и их виды
3. Понятие языка программирования, классификация

4. Обзор языков программирования
5. Методология разработки программных продуктов

1.9.2 Краткое содержание вопросов:

1. Сущность, понятие, задачи и функции инструментального программного обеспечения

Любая задача, подлежащая решению на ЭВМ, должна быть предварительно соответствующим образом подготовлена. Подготовка задачи к решению на ЭВМ включает несколько этапов. Один из основных этапов подготовки – создание алгоритма решения задачи. Алгоритм может быть представлен различными способами: для человека наиболее понятен алгоритм, изображенный в виде блок-схемы, а для ЭВМ подходит только представление алгоритма средствами алгоритмического языка программирования.

Алгоритм, записанный средствами языка программирования, называется программой. **Программа – это логически упорядоченная последовательность команд, реализующая алгоритм решения задачи.** ЭВМ понимает только язык двоичных кодов. Поэтому программа (алгоритм) для нее должна быть написана на **машинном языке** (языке низкого уровня). В то же время записать программу в машинных кодах – достаточно сложная задача, требующая соответствующего уровня подготовки программиста. В повседневной жизни при решении различных прикладных задач на ЭВМ не всегда имеется возможность привлечь специалиста по программированию, поэтому составлять программу вынужден тот, кто заинтересован в решении задачи. Для облегчения процесса программирования разработаны специальные **языки программирования**. Языки программирования – это искусственные языки. От естественных они отличаются ограниченным числом "слов", и очень строгими правилами записи команд (операторов). Тем не менее, эти языки понятны человеку (при соответствующей подготовке последнего).

С помощью языка программирования создается не готовая программа, а только ее текст, описывающий ранее разработанный алгоритм. Чтобы получить работающую программу, надо этот текст либо сначала автоматически перевести в машинный код (для этого служат программы - **компиляторы**) и затем использовать отдельно от исходного текста, либо переводить в машинный код параллельно с выполнением программы (этим занимаются программы **-интерпретаторы**).

В ходе лекции будет дано общее представление о языках программирования, пользующихся наибольшей популярностью, дана краткая характеристика этим языкам, а также изложено понятие о системе программирования.

Сущность, понятие инструментального программного обеспечения

Инструментальное программное обеспечение (ИПО) — программное обеспечение, предназначенное для использования в ходе проектирования, разработки и сопровождения программ.

Применяется инструментальное обеспечение в фазе разработки. Инструментальное программное обеспечение - это совокупность программ, используемых для помощи программистам в их работе, для помощи руководителям разработки программного обеспечения в их стремлении проконтролировать процесс разработки и получаемую продукцию. Наиболее известными представителями этой части программного обеспечения являются программы трансляторов с языков программирования, которые помогают программистам писать машинные команды. Инструментальными программами являются трансляторы с языков Фортран, Кобол, Джо-виал, Бейсик, АПЛ и Паскаль. Они облегчают процесс создания новых рабочих программ. Однако трансляторы с языков это

только наиболее известная часть инструментальных программ; существует же их великое множество.

Использование вычислительных машин для помощи в создании новых программ далеко не очевидно для людей, не являющихся профессиональными программистами. Часто же бывает так, что профессионалы рассказывают об инструментальном (фаза разработки) и системном (фаза использования) программном обеспечении на едином дыхании, предполагая, что не посвященному в тайны их мастерства известно об этой роли инструментального программного обеспечения. Так же как и в фазе использования (для прикладных программ), системное обеспечение работает и в фазе разработки, но только совместно с инструментальным обеспечением. Инструментальное ПО или системы программирования - это системы для автоматизации разработки новых программ на языке программирования.

В самом общем случае для создания программы на выбранном языке программирования (языке системного программирования) нужно иметь следующие компоненты:

1. Текстовый редактор для создания файла с исходным текстом программы.
2. Компилятор или интерпретатор. Исходный текст с помощью программы-компилятора переводится в промежуточный объектный код. Исходный текст большой программы состоит из нескольких модулей (файлов с исходными текстами). Каждый модуль компилируется в отдельный файл с объектным кодом, которые затем надо объединить в одно целое.
3. Редактор связей или сборщик, который выполняет связывание объектных модулей и формирует на выходе работоспособное приложение - исполнимый код.

Исполнимый код - это законченная программа, которую можно запустить на любом компьютере, где установлена операционная система, для которой эта программа создавалась. Как правило, итоговый файл имеет расширение .EXE или .COM.

В последнее время получили распространение визуальный методы программирования (с помощью языков описания сценариев), ориентированные на создание Windows-приложений. Этот процесс автоматизирован в средах быстрого проектирования. При этом используются готовые визуальные компоненты, которые настраиваются с помощью специальных редакторов.

Наиболее популярные редакторы (системы программирования программ с использованием визуальных средств) визуального проектирования:

Borland Delphi - предназначен для решения практически любых задачи прикладного программирования.

Borland C++ Builder - это отличное средство для разработки DOS и Windows приложений.

Microsoft Visual Basic - это популярный инструмент для создания Windows-программ.

Microsoft Visual C++ - это средство позволяет разрабатывать любые приложения, выполняющиеся в среде ОС типа Microsoft Windows

аким образом, сущность инструментального программного обеспечения заключается в создании любой исполняемой программы, путем преобразования формально логических выражений в исполняемый машинный код, а также его контроль и корректировка.

Задачи и функции инструментального программного обеспечения

Для инструментального программного обеспечения, как особой разновидности программного обеспечения, характерны общие и частные функции, как и для всего программного обеспечения в целом. Общие функции рассмотрены нами выше, а специализированными функциями, присущими только данному типу программ, являются:

1. Создание текста разрабатываемой программы с использованием специально установленных кодовых слов (языка программирования), а также определенного набора символов и их расположения в созданном файле - синтаксис программы.

2. Перевод текста создаваемой программы в машинно-ориентированный код, доступный для распознавания ЭВМ. В случае значительного объема создаваемой программы, она разбивается на отдельные модули и каждый из модулей переводится отдельно.

3. Соединение отдельных модулей в единый исполняемый код, с соблюдением необходимой структуры, обеспечение координации взаимодействия отдельных частей между собой.

4. Тестирование и контроль созданной программы, выявление и устранение формальных, логических и синтаксических ошибок, проверка программ на наличие запрещенных кодов, а также оценка работоспособности и потенциала созданной программы.

2. Системы программирования. Трансляторы и их виды

Для персональных компьютеров создано огромное количество программных средств, предназначенных для использования в различных областях деятельности людей. Тем не менее, в повседневной жизни иногда возникают задачи, для решения которых на ПЭВМ либо пока не разработано нужной программы, либо пользователь не знает, где ее можно найти. Как же быть в этой ситуации? Пользователь вынужден самостоятельно написать программу для решения возникшей задачи на ПЭВМ.

Что собой представляет программа для ЭВМ? Это последовательность команд, описывающих алгоритм функционирования ЭВМ в конкретной ситуации (то есть при решении конкретной задачи). Команды, поступающие в центральный процессор, на самом деле являются последовательностями электрических сигналов высокого или низкого уровней, что является интерпретацией последовательностей двоичных чисел, соответственно, единиц или нулей. Таким образом, реально **программа**, с которой работает процессор ЭВМ, **представляет собой последовательность двоичных чисел, называемую машинным кодом**.

Самому написать программу в машинном коде весьма сложно, причем эта сложность резко возрастает с увеличением трудоемкости решения нужной задачи и, соответственно, размера программы для ее решения. Поэтому на практике для разработки программ используют **системы программирования**.

Система программирования представляет собой **комплекс инструментальных программных средств, предназначенный для работы с программами на одном из языков программирования**. Языки программирования – искусственные языки. От естественных они отличаются ограниченным числом "слов" и очень строгими правилами записи команд (операторов).

С помощью языка программирования создается не готовая программа, а только ее текст, описывающий ранее разработанный алгоритм. Чтобы получить работающую программу, надо этот текст либо сначала автоматически перевести в машинный код (для этого служат программы - **компиляторы**) и затем использовать отдельно от исходного текста, либо переводить в машинный код параллельно с выполнением программы (этим занимаются программы -**интерпретаторы**).

Компиляторы полностью обрабатывают весь текст программы (он иногда называется **исходный код**). Они просматривают его в поисках синтаксических ошибок (иногда несколько раз), выполняют определенный смысловой анализ и затем автоматически переводят (**транслируют**) на машинный язык – генерируют машинный код. Нередко при этом выполняется оптимизация с помощью набора методов, позволяющих повысить быстродействие программы (например, с помощью инструкций, ориентированных на

конкретный процессор, путем исключения ненужных команд, промежуточных вычислений и т. д.). В результате законченная программа получается компактной и эффективной, работает в сотни раз быстрее программы, выполняемой с помощью интерпретатора, и может быть перенесена на другие компьютеры с процессором, поддерживающим соответствующий машинный код.

Основной **недостаток компиляторов** – трудоемкость трансляции языков программирования, ориентированных на обработку данных сложной структуры, часто заранее неизвестной или динамически меняющейся во время работы программы. Тогда в машинный код приходится вставлять множество дополнительных проверок, анализировать наличие ресурсов операционной системы, динамически их захватывать и освобождать, формировать и обрабатывать в памяти компьютера сложные объекты, что на уровне жестко заданных машинных инструкций осуществить довольно трудно, а для ряда задач – практически невозможно.

С помощью **интерпретатора**, наоборот, допустимо в любой момент остановить работу программы, исследовать содержимое памяти, организовать диалог с пользователем, выполнить сколь угодно сложные преобразования данных и при этом постоянно контролировать состояние окружающей программно-аппаратной среды, благодаря чему достигается высокая надежность работы. Интерпретатор при выполнении каждого оператора проверяет множество характеристик операционной системы и при необходимости максимально подробно информирует разработчика о возникающих проблемах. Кроме того, интерпретатор очень удобен для использования в качестве инструмента изучения программирования, так как позволяет понять принципы работы любого отдельного оператора языка.

В реальных системах программирования перемешаны технологии и компиляции, и интерпретации.

В **состав системы программирования**, как правило, входят следующие средства:

- трансляторы с языков высокого уровня;
- средства редактирования, компоновки и загрузки программ;
- макроассемблеры (машинно-ориентированные языки);
- отладчики машинных программ.

Таковыми средствами, в первую очередь, являются:

- **текстовый редактор (Edit)**, осуществляющий функции записи и редактирования **исходного текста программы**. В принципе, для написания текста программы может быть использован любой текстовый редактор, но лучше использовать **специализированные редакторы**, которые ориентированы на конкретный язык программирования и позволяют в процессе ввода текста выделять ключевые слова и идентификаторы разными цветами и шрифтами. Подобные редакторы созданы для всех популярных языков. Как правило, они дополнительно могут автоматически проверять правильность синтаксиса программы непосредственно во время ее ввода;

- **загрузчик программ (Load)**, позволяющий выбрать из директории нужный текстовый файл программы;

- **запускающий программ (Run)**, осуществляющий процесс выполнения программы;

- **компилятор (Compile)**, предназначенный для компиляции или интерпретации исходного текста программы в машинный код с диагностикой синтаксических и семантических (логических) ошибок;

- **отладчик (Debug)**, выполняющий сервисные функции по отладке и тестированию программы;

- **диспетчер файлов (File)**, предоставляющий возможность выполнять операции с файлами: сохранение, поиск, удаление и т. п.

3. Понятие языка программирования, классификация

Сегодня практически все программы для ПЭВМ создаются с помощью **языков программирования**. Теоретически программу можно записать и средствами обычного человеческого (естественного) языка – это называется программированием на **метаязыке** (подобный подход обычно используется на этапе составления алгоритма), но автоматически перевести такую программу в машинный код пока не представляется возможным из-за высокой неоднозначности естественного языка.

Языки программирования – это формальные языки специально созданные для общения человека с компьютером. Каждый язык программирования, равно как и "естественный" язык (русский, английский и т. д.), имеет **алфавит**, **словарный запас**, свои **грамматику** и **синтаксис**, а также **семантику**. **Алфавит** языка программирования представляет собой фиксированный для данного языка набор основных символов, допускаемых для составления текста программы на этом языке, **синтаксис** – это система правил, определяющих допустимые конструкции языка программирования из букв алфавита, **семантика** – система правил однозначного толкования отдельных языковых конструкций, позволяющих воспроизвести процесс обработки данных.

Взаимодействие синтаксических и семантических правил определяют те или иные понятия языка, например, операторы, идентификаторы, переменные, функции и процедуры, модули и т. д. В отличие от естественных языков **правила грамматики и семантики для языков программирования**, как и для всех формальных языков, должны быть **явно, однозначно и четко сформулированы**.

Рассмотрим некоторые основные понятия, общие для большинства языков программирования.

Оператор – одно из ведущих понятий всех языков программирования. Каждый оператор представляет собой законченную фразу языка и определяет однозначно трактуемый этап обработки данных.

Большая часть операторов ведет обработку **величин**. **Величины** могут быть **постоянными** и **переменными**. Значения **постоянных величин** не изменяются в ходе выполнения программы. **Переменная** – это программный объект, способный принимать некоторое значение с помощью оператора присваивания. В ходе выполнения программы значения переменной могут неоднократно изменяться. Каждая переменная после ее описания отождествляется с некоторой ячейкой памяти, содержимое которой является ее значением.

Любая величина характеризуется **типом**, **именем** и **значением**. Наиболее распространенные типы величин – числовые (целые и вещественные), символьные, логические.

Всем программным объектам в языках программирования даются индивидуальные **имена**. Имя программного объекта называют **идентификатором**. Чаще всего идентификатором является любая конечная последовательность букв и цифр, начинающаяся с буквы.

Функция – это программный объект, задающий вычислительную процедуру определения значения, зависящего от некоторых аргументов. Функции используются в программе в **выражениях**. **Выражения** строятся из величин – постоянных и переменных, функций, скобок, знаков операций и т. д. Выражение имеет определенный тип, определяемый типом принимаемых в итоге его вычисления значений. Возможны выражения **арифметические**, **логические**, **символьные**, **строковые** и т. д.

Модуль – это специальная программная единица, предназначенная для создания библиотек и разделения больших программ на логически связанные блоки.

Язык программирования составляет **ядро системы программирования**. Существующие языки программирования можно разделить на две группы: **процедурные** и **непроцедурные** (рис.1).

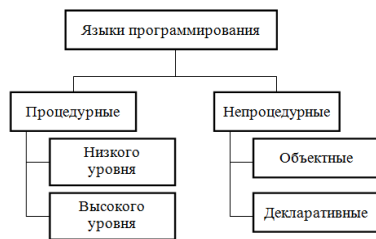


Рисунок 1 – Общая классификация языков программирования

Процедурные (алгоритмические) языки предназначены для написания процедурных программ, представляющих собой систему предписаний для решения конкретной задачи. Роль компьютера в этом случае сводится к механическому выполнению этих предписаний.

Процедурные языки программирования разделяют на языки **низкого** и **высокого** уровня.

Каждый процессор имеет **систему команд – совокупность машинных кодов**, которые он понимает и может исполнять. Если язык программирования ориентирован на конкретный тип процессора и учитывает его особенности, то он называется **языком программирования низкого уровня** (машинно-ориентированным). Это вовсе не означает, что он "плохой". Имеется в виду, что операторы языка близки к машинному коду и ориентированы на конкретные команды процессора. Языком программирования самого низкого уровня является **язык ассемблера**. Языки программирования низкого уровня позволяют создавать программы из машинных кодов, обычно в шестнадцатеричной форме. С ними трудно работать, но созданные с их помощью высококвалифицированным программистом программы занимают меньше места в памяти и работают быстрее. С помощью этих языков удобнее разрабатывать системные программы, драйверы, некоторые другие виды программ. Поскольку наборы инструкций для каждого типа процессора отличаются, конкретной компьютерной архитектуре соответствует свой язык программирования низкого уровня, и написанная на нем программа может быть использована только в этой среде.

Языки программирования, имитирующие естественные языки, обладающие укрупненными командами, ориентированными на решение содержательных прикладных задач, называют **языками программирования высокого уровня**. В настоящее время насчитывается несколько сотен таких языков, а если считать и их диалекты, то это число возрастет до нескольких тысяч. Языки программирования высокого уровня существенно отличаются от машинно-ориентированных языков.

Во-первых, машинная программа, в конечном счете, записывается с помощью лишь двух символов: 0 и 1. Во-вторых, каждая ЭВМ имеет ограниченный набор машинных операций, ориентированных на структуру процессора. Как правило, этот набор состоит из сравнительно небольшого числа простейших операций, типа: переслать число в ячейку; считать число из ячейки; увеличить содержимое ячейки на +1 и т. п. Команда на машинном языке содержит очень ограниченный объем информации, поэтому она обычно определяет простейший обмен содержимого ячеек памяти, элементарные арифметические и логические операции.

Языки программирования высокого уровня имеют, по сравнению с языками программирования низкого уровня, следующие достоинства:

- алфавит языка значительно шире машинного, что делает его гораздо более выразительным и существенно повышает наглядность и понятность текста;
- набор операций, допустимых для использования, не зависит от набора машинных операций, а выбирается из соображений удобства формулирования алгоритмов решения задач определенного класса;
- конструкции команд (операторов) отражают содержательные виды обработки данных и задаются в удобном для человека виде;

- используется аппарат переменных и действия с ними;
- поддерживается широкий набор типов данных.

Таким образом, языки программирования высокого уровня являются машинно-независимыми и требуют использования соответствующих программ-переводчиков (трансляторов) для представления программы на языке машины, на которой она будет исполняться.

Языки программирования высокого уровня значительно ближе и понятнее человеку, чем компьютеру. Особенности конкретных архитектур ПЭВМ в них не учитываются, поэтому создаваемые программы на уровне исходных текстов легко переносимы на другие платформы, для которых создан **транслятор** этого языка. Разрабатывать программы на языках высокого уровня с помощью понятных и мощных команд значительно проще, а ошибок при создании программ допускается значительно меньше.

Языки программирования высокого уровня являются **алгоритмическими языками**. **Алгоритмический язык** – это набор символов и терминов, которые в соответствии с правилами синтаксиса описывают алгоритм решения задачи. Своими конструкциями и правилами написания этот язык, с одной стороны, близок к математическому описанию задачи, а с другой стороны – содержит такие выражения, которые близки к естественному языку, чаще всего к английскому.

Программу на алгоритмическом языке записывают, как правило, в обычном текстовом редакторе. В результате получают текстовый файл, содержащий программный код, записанный на языке программирования высокого уровня. Такой текстовый файл называют **исходным модулем (исходным текстом)**. Исходный текст программы состоит из специальных команд (**операторов языка программирования**). Процессор исполнить эти команды не может, поэтому исходный текст с помощью **транслятора (компилятора или интерпретатора)** преобразуют в инструкции процессору. Назначение компиляторов и интерпретаторов, а также принципиальные отличия между ними будут рассмотрены в третьем вопросе.

4. Обзор языков программирования

Рассмотрим основные сведения об алгоритмических языках программирования, получивших наибольшее распространение среди программистов.

Fortran (Фортран). Это первый компилируемый язык, созданный Джимом Бэкусом в 50-е годы. Программисты, разрабатывавшие программы исключительно на ассемблере, выражали серьезное сомнение в возможности появления высокопроизводительного языка высокого уровня, поэтому основным критерием при разработке компиляторов Фортрана являлась эффективность исполняемого кода. Хотя в Фортране впервые был реализован ряд важнейших понятий программирования, удобство создания программ было принесено в жертву возможности получения эффективного машинного кода. Однако для этого языка было создано огромное количество библиотек, начиная от статистических комплексов и заканчивая пакетами управления спутниками, поэтому Фортран продолжает активно использоваться во многих организациях.

Cobol (Кобол). Это компилируемый язык для применения в экономической области и решения бизнес-задач, разработанный в начале 60-х годов. Он отличается большой "многословностью" – его операторы иногда выглядят как обычные английские фразы. В Коболе были реализованы очень мощные средства работы с большими объемами данных, хранящимися на различных внешних носителях. На этом языке создано очень много приложений, которые активно эксплуатируются и сегодня. Достаточно сказать, что наибольшую зарплату в США получают программисты на Коболе.

Algol (Алгол). Компилируемый язык, созданный в 1960 году. Он был призван заменить Фортран, но из-за более сложной структуры не получил широкого распростра-

нения. В 1968 году была создана версия **Алгол 68**, по своим возможностям и сегодня опережающая многие языки программирования, однако из-за отсутствия достаточно эффективных компьютеров для нее не удалось своевременно создать хорошие компиляторы. Язык Алгол сыграл большую роль в теории, но для практического программирования сейчас почти не используется.

Pascal (Паскаль). Язык Паскаль, созданный в конце 70-х годов основоположником множества идей современного программирования Никлаусом Виртом и названный в честь ученого Блеза Паскаля, во многом напоминает Алгол, но в нем ужесточен ряд требований к структуре программы и имеются возможности, позволяющие успешно применять его при создании крупных программных проектов. Изначально Паскаль был задуман как язык программирования для студентов. Он и сейчас является основным языком программирования, используемым в качестве учебного, во многих высших учебных заведениях. На базе языка Паскаль созданы несколько более мощных языков (Модула, Ада, Дельфи).

BASIC (Бейсик). Для этого языка имеются и компиляторы, и интерпретаторы, а по популярности он занимает первое место в мире. Бейсик был создан в 1964 г. Томасом Куртом и Джоном Кемени как язык для начинающих программистов, облегчающий написание простых программ. В настоящее время разработано несколько различных версий языка Бейсик, таких как GWBASIC, QBASIC, Pro-BASIC, Turbo-BASIC, Visual-BASIC и др.

C (Си). Данный язык был создан в лаборатории Bell и первоначально не рассматривался как массовый. Он планировался для замены ассемблера, чтобы иметь возможность создавать столь же эффективные и компактные программы, и в то же время не зависеть от конкретного типа процессора.

Си во многом похож на Паскаль и имеет дополнительные средства для прямой работы с памятью (**указатели**). На этом языке в 70-е годы написано множество прикладных и системных программ и ряд известных операционных систем (например, Unix). В настоящее время Си широко используется при создании системного программного обеспечения.

PL/1 (ПЛ/1). В середине 60-х годов компания **IBM** решила взять все лучшее из языков Фортран, Кобол и Алгол и воплотить в одном языке программирования. В результате в 1964 году на свет появился новый компилируемый язык программирования, который получил название **Programming Language One**. В этом языке было реализовано множество уникальных решений, полезность которых удастся оценить только спустя 33 года, в эпоху крупных программных систем. По своим возможностям **ПЛ/1** значительно мощнее многих других языков (Си, Паскаля). Например, в **ПЛ/1** присутствует уникальная возможность указания точности вычислений – ее нет даже у **Си++** и **Явы**. В настоящее время язык **ПЛ/1** почти не используется.

Ada (Ада). Назван по имени леди Огасты Ады Лавлейс, дочери английского поэта Байрона и его отдаленной родственницы Анабеллы Милбэнк. В 1979 году сотни экспертов Министерства обороны США отобрали из 17 вариантов именно этот язык, разработанный небольшой группой под руководством Жана Ихбиа. Он удовлетворил на то время все требования Пентагона, а к сегодняшнему дню в его развитие вложены десятки миллиардов долларов. Структура самого языка похожа на Паскаль. В нем имеются средства строгого разграничения доступа к различным уровням спецификаций, доведена до предела мощность управляющих конструкций.

Принципиально иное направление в программировании связано с методологиями **непроцедурного программирования**. Для этих целей используются **непроцедурные языки программирования**, которые можно разделить на **объектные** и **декларативные**. Объектно-ориентированный язык создает окружение в виде **множества независимых объектов**. Каждый объект ведет себя подобно отдельному компьютеру, их можно использовать для решения задач как "черные ящики", не вникая во внутренние механизмы

их функционирования. Объектно-ориентированный подход к проектированию программных продуктов основан на:

- выделении классов объектов;
- установлении характерных **свойств** объектов и **методов** их обработки;
- создании иерархии классов, наследовании свойств объектов и методов их обработки.

Каждый объект объединяет как **данные**, так и **программу обработки** этих данных и относится к определенному классу. С помощью класса один и тот же программный код можно использовать для относящихся к нему различных объектов.

Из языков объектного программирования, популярных среди профессионалов, следует назвать прежде всего:

C++ (Си++). Си++ - это объектно-ориентированное расширение языка Си, созданное Бьярном Страуструпом в 1980 году. Множество новых мощных возможностей, позволивших резко повысить производительность работы программистов, наложилось на унаследованную от языка Си определенную низкоуровневость, в результате чего создание сложных и надежных программ потребовало от разработчиков высокого уровня профессиональной подготовки;

Java (Джава). Этот язык был создан компанией Sun в начале 90-х годов на основе Си++. Он призван упростить разработку приложений на основе Си++ путем исключения из него всех низкоуровневых возможностей. Но главная особенность этого языка – компиляция не в машинный код, а в платформенно-независимый байт-код (каждая команда занимает один байт). Этот байт-код может выполняться с помощью интерпретатора – виртуальной Java-машины, версии которой созданы сегодня для любых платформ. Благодаря наличию множества Java-машин программы на Джава можно переносить не только на уровне исходных текстов, но и на уровне двоичного байт-кода, поэтому по популярности язык Джава сегодня занимает второе место в мире после Бейсика. Язык Джава чрезвычайно эффективен для создания интерактивных Web-страниц;

Smalltalk (Смолток). Работа над этим языком началась в 1970 году в исследовательской лаборатории корпорации XEROX, а закончилась спустя 10 лет, воплотившись в окончательном варианте интерпретатора **SMALLTALK-80**. Данный язык оригинален тем, что его синтаксис очень компактен и базируется исключительно на понятии объекта. В этом языке отсутствуют операторы или данные. Все, что входит в Смолток, является объектами, а сами объекты общаются друг с другом исключительно с помощью сообщений (например, появление выражения **I+1** вызывает посылку объекту **I** сообщения "+", то есть "прибавить", с параметром 1, который считается не числом-константой, а тоже объектом). Больше никаких управляющих структур, за исключением "оператора" ветвления (на самом деле функции, принадлежащей стандартному объекту), в языке нет, хотя их можно очень просто смоделировать. Сегодня версия **VisualAge for Smalltalk** активно развивается компанией IBM;

Delphi (Дельфи) – язык объектно-ориентированного "визуального" программирования. Явился развитием языка Паскаль.

При использовании **декларативного** языка программирования программист указывает исходные информационные структуры, взаимосвязи между ними и то, какими свойствами должен обладать результат. При этом процедуру его получения (то есть алгоритм) программист не строит. В этих языках отсутствует понятие "оператор".

Типичными представителями декларативных языков являются:

LISP (Лисп). Интерпретируемый язык программирования, созданный в 1960 году Джоном Маккарти. Ориентирован на структуру данных в форме списка и позволяет организовывать эффективную обработку больших объемов текстовой информации;

Prolog (Пролог). Создан в начале 70-х годов. Программа на этом языке, в основу которого положена математическая модель теории исчисления предикатов, строится из последовательности фактов и правил, а затем формулируется утверждение, которое

Пролог будет пытаться доказать с помощью введенных правил. Человек только описывает структуру задачи, а внутренний "мотор" Пролога сам ищет решение с помощью методов поиска и сопоставления.

5. Методология разработки программных продуктов

Классификация методов проектирования программных продуктов

Проектирование алгоритмов и программ - наиболее ответственный этап жизненного цикла программных продуктов, определяющий, насколько создаваемая программа соответствует спецификациям и требованиям со стороны конечных пользователей. Затраты на создание, сопровождение и эксплуатацию программных продуктов, научно-технический уровень разработки, время морального устаревания и многое другое- все это также зависит от проектных решений.

Пример: Переход к графической среде работы конечного пользователя типа Windows или Macintosh потребует создания пользовательского интерфейса с элементами управления в виде пиктограмм, кнопок, выпадающих меню, обязательного применения манипулятора мышь и др. Отсутствие в программном продукте уже ставших стандартом подобных элементов свидетельствует о том, что в будущем потребуются значительные затраты на модификацию этого продукта, иначе будет падать его конкурентоспособность и привлекательность для конечного пользователя.

Методы проектирования алгоритмов и программ очень разнообразны, их можно классифицировать по различным признакам, важнейшими из которых являются:

- степень автоматизации проектных работ;
- принятая методология процесса разработки.

По степени автоматизации проектирования алгоритмов и программ можно выделить:

- методы традиционного (неавтоматизированного) проектирования;
- методы автоматизированного проектирования (CASE-технология и ее элементы).

Неавтоматизированное проектирование алгоритмов и программ преимущественно используется при разработке небольших по трудоемкости и структурной сложности программных продуктов, не требующих участия большого числа разработчиков. Трудоемкость разрабатываемых программных продуктов, как правило, небольшая, а сами программные продукты имеют преимущественно прикладной характер.

При нарушении этих ограничений заметно снижается производительность труда разработчиков, падает качество разработки, и, как ни парадоксально, увеличиваются трудозатраты и стоимость программного продукта в целом.

Автоматизированное проектирование алгоритмов и программ возникло с необходимостью уменьшить затраты на проектные работы, сократить сроки их выполнения, создать типовые "заготовки" алгоритмов и программ, многократно тиражируемых для различных разработок, координации работ большого коллектива разработчиков, стандартизации алгоритмов и программ.

Автоматизация проектирования может охватывать все или отдельные лапы жизненного цикла программного продукта, при этом работы этапов могут быть изолированы друг от друга либо составлять единый комплекс, выполняемый последовательно во времени. Как правило, автоматизированный подход требует технического и программного "перевооружения" труда самих разработчиков (мощных компьютеров, дорогостоящего программного инструментария, а также повышения квалификации разработчиков и т.п.).

Автоматизированное проектирование алгоритмов и программ под силу лишь крупным фирмам, специализирующимся на разработке определенного класса программных продуктов, занимающих устойчивое положение на рынке программных средств.

Проектирование алгоритмов и программ может основываться на различных подходах, среди которых наиболее распространены:

- структурное проектирование программных продуктов;
- информационное моделирование предметной области и связанных с ней приложений;
- объектно-ориентированное проектирование программных продуктов.

В основе структурного проектирования лежит последовательная декомпозиция, целенаправленное структурирование на отдельные составляющие. Начало развития структурного проектирования алгоритмов и программ падает на 60-е гг. Методы структурного проектирования представляют собой комплекс технических и организационных принципов системного проектирования.

Типичными методами структурного проектирования являются:

- нисходящее проектирование, кодирование и тестирование программ;
- модульное программирование;
- структурное проектирование (программирование) и др.

В зависимости от объекта структурирования различают:

- функционально-ориентированные методы - последовательное разложение задачи или целостной проблемы на отдельные, достаточно простые составляющие, обладающие функциональной определенностью;
- методы структурирования данных.

Для функционально-ориентированных методов в первую очередь учитываются заданные функции обработки данных, в соответствии с которыми определяется состав и логика работы (алгоритмы) отдельных компонентов программного продукта. С изменением содержания функций обработки, их состава, соответствующего им информационного входа и выхода требуется перепроектирование программного продукта. Основной упор в структурном подходе делается на моделирование процессов обработки данных.

Для методов структурирования данных осуществляется анализ, структурирование и создание моделей данных, применительно к которым устанавливается необходимый состав функций и процедур обработки. Программные продукты тесно связаны со структурой обрабатываемых данных, изменение которой отражается на логике обработки (алгоритмах) и обязательно требует перепроектирования программного продукта.

Структурный подход использует:

- диаграммы потоков данных (информационно-технологические схемы) - показывают процессы и информационные потоки между ними с учетом "событий", инициирующих процессы обработки;
- интегрированную структуру данных предметной области (инфологическая модель, ER- диаграммы);
- диаграммы декомпозиции - структура и декомпозиция целей, функций управления, приложений;
- структурные схемы - архитектура программного продукта в виде иерархии взаимосвязанных программных модулей с идентификацией связей между ними, детальная логика обработки данных программных модулей (блок-схемы).

Для полного представления о программном продукте необходима также текстовая информация описательного характера.

Еще большую значимость информационные модели и структуры данных имеют для информационного моделирования предметной области, в основе которого положение об определяющей роли данных при проектировании алгоритмов и программ. Подход появился в условиях развития программных средств организации хранения и обработки данных - СУБД (см. гл. 15).

Один из основоположников информационной инженерии - Дж. Мартин - выделяет следующие составляющие данного подхода:

- информационный анализ предметных областей (бизнес - областей);
- информационное моделирование - построение комплекса взаимосвязанных моделей данных;

- системное проектирование функций обработки данных;
- детальное конструирование процедур обработки данных.

Первоначально строятся информационные модели различных уровней представления:

- информационно-логическая модель, не зависящая от средств программной реализации хранения и обработки данных, отражающая интегрированные структуры данных предметной области;

- даталогические модели, ориентированные на среду хранения и обработки данных.

Даталогические модели имеют логический и физический уровни представления. Физический уровень соответствует организации хранения данных в памяти компьютера. Логический уровень данных применительно к СУБД реализован в виде:

- концептуальной модели базы данных - интегрированные структуры данных под управлением СУБД;

- внешних моделей данных - подмножество структур данных для реализации приложений.

Средствами структур данных моделируются функции предметной области, прослеживается взаимосвязь функций обработки, уточняется состав входной и выходной информации, логика преобразования входных структур данных в выходные. Алгоритм обработки данных можно представить как совокупность процедур преобразований структур данных в соответствии с внешними моделями данных.

Выбор средств реализации базы данных определяет вид даталогических моделей и, следовательно, алгоритмы преобразования данных. В большинстве случаев используется реляционное представление данных базы данных и соответствующие реляционные языки для программирования (манипулирования) обработки данных СУБД и реализации алгоритмов обработки. Данный подход использован во многих CASE-технологиях.

Объектно-ориентированный подход к проектированию программных продуктов основан на:

- выделении классов объектов;
- установлении характерных свойств объектов и методов их обработки;
- создании иерархии классов, наследовании свойств объектов и методов их обработки.

Каждый объект объединяет как данные, так и программу обработки этих данных и относится к определенному классу. С помощью класса один и тот же программный код можно использовать для относящихся к нему различных объектов.

Объектный подход при разработке алгоритмов и программ предполагает:

- объектно-ориентированный анализ предметной области;
- объектно-ориентированное проектирование.

Объектно-ориентированный анализ - анализ предметной области и выделение объектов, определение свойств и методов обработки объектов, установление их взаимосвязей.

Объектно-ориентированное проектирование соединяет процесс объектной декомпозиции и представления с использованием моделей данных проектируемой системы на логическом и физическом уровнях, в статике и динамике.

Для проектирования программных продуктов разработаны объектно-ориентированные технологии, которые включают в себя специализированные языки программирования и инструментальные средства разработки пользовательского интерфейса.

Традиционные подходы к разработке программных продуктов всегда подчеркивали различия между данными и процессами их обработки. Так, технологии, ориентированные

на информационное моделирование, сначала специфицируют данные, а затем описывают процессы, использующие эти данные. Технологии структурного подхода ориентированы, в первую очередь, на процессы обработки данных с последующим установлением необходимых для этого данных и организации информационных потоков между связанными процессами.

Объектно-ориентированная технология разработки программных продуктов объединяет данные и процессы в логические сущности - объекты, которые имеют способность наследовать характеристики (методы и данные) одного или более объектов, обеспечивая тем самым повторное использование программного кода. Это приводит к значительному уменьшению затрат на создание программных продуктов, повышает эффективность жизненного цикла программных продуктов (сокращается длительность фазы разработки). При выполнении программы объекту посылается сообщение, которое инициирует обработку данных объекта.

Этапы создания программных продуктов

При традиционной неавтоматизированной разработке программ независимо от принятого метода проектирования и используемого инструментария выполняют следующие работы.

1. Составление технического задания на программирование

Данная работа соответствует этапу анализа и спецификации программ жизненного цикла программных продуктов.

При составлении технического задания требуется:

- определить платформу разрабатываемой программы - тип операционной системы (например, для IBM PC-совместимых машин делается выбор операционной среды: MS DOS, Windows, Windows NT либо Unix, OS/2);
- оценить необходимость сетевого варианта работы программы (определяется программное обеспечение (ПО) вычислительной сети - Windows NT, допустимая номенклатура программного обеспечения сетевой обработки);
- определить необходимость разработки программы, которую можно переносить на различные платформы;
- обосновать целесообразность работы с базами данных под управлением СУБД.

На этом же этапе выбирают методы решения задачи; разрабатывают обобщенный алгоритм решения комплекса задач, функциональную структуру алгоритма или состав объектов, определяют требования к комплексу технических средств системы обработки информации, интерфейсу конечного пользователя.

2. Технический проект

На данном этапе выполняется комплекс наиболее важных работ, а именно:

- с учетом принятого подхода к проектированию программного продукта разрабатывается детальный алгоритм обработки данных или уточняется состав объектов и их свойств, методов обработки, событий, запускающих методы обработки;
- определяется состав общесистемного программного обеспечения, включающий базовые средства (операционную систему, модель СУБД, электронные таблицы, методо-ориентированные и функциональные ППП промышленного назначения и т.п.);
- разрабатывается внутренняя структура программного продукта, образованная отдельными программными модулями;
- осуществляется выбор инструментальных средств разработки программных модулей.

Работы данного этапа в существенной степени зависят от принятых решений по технической части системы обработки данных и операционной среде, от выбранных инструментальных средств проектирования алгоритмов и программ, технологии работ.

Пример. Для создания MS DOS-приложений может быть использован язык программирования Visual Basic for DOS Standard, Fortran 5.1, Visual C++ for Windows. Если

необходима переносимость программ на другие ЭВМ или другие операционные платформы, выбирается среда Windows NT.

При разработке программ, работающих в среде Windows, возможно применение технологии OLE 2.0 для создания приложений, включающих объекты других приложений. Определяется способ использования объектов: внедрение (embedding) или связывание (linking).

Приложение может работать с базами данных различных СУБД, для этого служит стандартная технология интерфейса Open Database Connectivity (ODBC). Работа в режиме телекоммуникаций обеспечивается стандартной технологией Messaging Application Program Interface (MAPI).

3. Рабочая документация (рабочий проект)

На данном этапе осуществляется адаптация базовых средств программного обеспечения (операционной системы, СУБД, методо-ориентированных ППП, инструментальных сред конечного пользователя - текстовых редакторов, электронных таблиц и т.п.). Выполняется разработка программных модулей или методов обработки объектов - собственно программирование или создание программного кода. Проводятся автономная и комплексная отладка программного продукта, испытание работоспособности программных модулей и базовых программных средств. Для комплексной отладки готовится контрольный пример, который позволяет проверить соответствие возможностей программного продукта заданным спецификациям.

Основной результат работ этого этапа - также создание эксплуатационной документации на программный продукт:

- описание применения - дает общую характеристику программного изделия с указанием сферы его применения, требований к базовому программному обеспечению, комплексу технических средств;

- руководство пользователя - включает детальное описание функциональных возможностей и технологии работы с программным продуктом. Данный вид документации ориентирован на конечного пользователя и содержит необходимую информацию для самостоятельного освоения и нормальной работы пользователя (с учетом требуемой квалификации пользователя);

- руководство программиста (оператора) - указывает особенности установки (инсталляции) программного продукта и его внутренней структуры - состав и назначение модулей, правила эксплуатации и обеспечения надежной и качественной работы программного продукта.

В ряде случаев на данном этапе для программных продуктов массового применения создаются обучающие системы, демоверсии, гипертекстовые системы помощи.

4. Ввод в действие

Готовый программный продукт сначала проходит опытную эксплуатацию (пробный рынок продаж), а затем сдается в промышленную эксплуатацию (тиражирование и распространение программного продукта).

Структура программных продуктов

В большей степени программные продукты не являются монолитом и имеют конструкцию (архитектуру) построения - состав и взаимосвязь программных модулей.

Модуль - это самостоятельная часть программы, имеющая определенное назначение и обеспечивающая заданные функции обработки автономно от других программных модулей.

Таким образом, программный продукт обладает внутренней организацией, или внутренней структурой, образованной взаимосвязанными программными модулями. Это справедливо для сложных и многофункциональных программных продуктов, которые часто называются программными системами.

Структуризация программ выполняется в первую очередь для удобства разработки, программирования, отладки и внесения изменений в программный продукт. Как правило, программные комплексы большой алгоритмической сложности разрабатываются коллективом разработчиков (2 - 15 и более человек). Управлять разработкой программ в условиях применения промышленных технологий изготовления программ можно лишь на научной основе.

Таким образом, структуризация программных продуктов преследует основные цели:

- распределить работы по исполнителям, обеспечив приемлемую их загрузку и требуемые сроки разработки программных продуктов;
- построить календарные графики проектных работ и осуществлять их координацию в процессе создания программных изделий;
- контролировать трудозатраты и стоимость проектных работ и др.

Структурное "разбиение" программ на отдельные составляющие служит основой и для выбора инструментальных средств их создания, хотя имеет место и обратное влияние - выбор инструментальных средств разработчика программного обеспечения определяет типы программных модулей. При создании программных продуктов выделяются многократно используемые модули, проводится их типизация и унификация, за счет чего сокращаются сроки и трудозатраты на разработку программного продукта в целом.

Некоторые программные продукты используют модули из готовых библиотек стандартных подпрограмм, процедур, функций, объектов, методов обработки данных.

На рис. 1 приведена типовая структура программного продукта, состоящего из отдельных программных модулей и библиотек процедур, встроенных функций, объектов и т.п.

Среди множества модулей различают:

- головной модуль - управляет запуском программного продукта (существует в единственном числе);
- управляющий модуль - обеспечивает вызов других модулей на обработку;
- рабочие модули - выполняют функции обработки;
- сервисные модули и библиотеки, утилиты - осуществляют обслуживающие функции.

В работе программного продукта активизируются необходимые программные модули. Управляющие модули задают последовательность вызова на выполнение очередную модуля. Информационная связь модулей обеспечивается за счет использования общей базы данных либо межмодульной передачи данных через переменные обмена.

Каждый модуль может оформляться как самостоятельно хранимый файл; для функционирования программного продукта необходимо наличие программных модулей в полном составе.

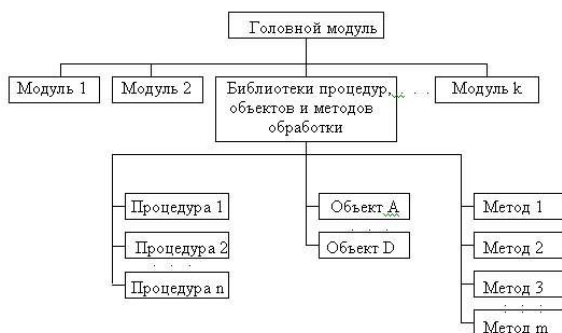


Рис 1. Структура программного продукта

Структурно-сложные программные продукты разрабатываются как пакеты программ, и чаще всего они имеют прикладной характер - пакеты прикладных программ, или ППП.

ППП (application program package) - это система программ, предназначенных для решения задач определенного класса.

Компоненты ППП объединены общими данными (базой данных), информационно и функционально связаны между собой и обладают свойством системности, т.е. объединению программ присуще новое качество, которое отсутствует для отдельного компонента ППП. Структура ППП, как правило, многомодульная.

Как показано в лекции, алгоритм функционирования ЭВМ при решении задач пользователя должен быть представлен в виде программы. Для того, чтобы ЭВМ понимала команды, записанные в программе, последняя должна представлять собой набор машинных кодов (двоичных слов). Написать программу в двоичных кодах очень трудоемкое занятие, требующее специальной подготовки программиста. Чтобы облегчить процесс программирования, разработаны языки программирования.

На сегодняшний день существует большое разнообразие языков программирования. Те, которые ближе по своей сущности к машинным кодам, называются языками программирования низкого уровня, а те, что более понятны человеку – языками высокого уровня. В частности, все алгоритмические языки программирования являются языками программирования высокого уровня.

Как правило, при решении различных прикладных задач используют разные языки программирования, наиболее подходящие в каждом конкретном случае.

1.10 Лекция № 13, 14 (4 часа).

Тема: «Основы программирования на алгоритмическом языке высокого уровня»

1.10.1 Вопросы лекции:

1. Средства разработки программ на языке Pascal
2. Тип. Иерархия типов
3. Структура программы
4. Выражения. Оператор присваивания
5. Операторы повтора

1.10.2 Краткое содержание вопросов:

1. Средства разработки программ на языке Pascal

Язык программирования Pascal был разработан в конце 60-х годов XX века Николасом Виртом. Он разрабатывал этот язык как учебный язык для своих студентов. С тех пор Pascal, сохранив простоту и структуру языка, разработанного Н. Виртом, превратился в мощное средство программирования. С помощью современного языка Pascal можно производить простые расчеты, разрабатывать программы для проведения сложных инженерных и экономических вычислений.

Разработку программы можно разбить на следующие этапы:

1. Составление алгоритма решения задачи. Алгоритм – это описание последовательности действий, которые необходимо выполнить для решения поставленной задачи.
2. Написание текста программы. Текст программы пишут на каком-либо языке программирования (например на Pascal) и вводят его в компьютер с помощью текстового редактора.

3. Отладка программы. Отладка программы – это процесс устранения ошибок из текста программы. Все ошибки делятся на синтаксические и логические. При наличии синтаксических ошибок (ошибок в написании операторов) программа не запускается. Подобные ошибки исправляются проще всего. Логические ошибки – это ошибки, при которых программа работает, но неправильно, выдавая не те результаты, которые ожидает разработчик или пользователь. Логические ошибки исправить сложнее, чем синтаксические, иногда для этого приходится переписывать отдельные участки программы, а иногда перерабатывать весь алгоритм.

4. Тестирование программы. Тестирование программы – процесс выявления ошибок в ее работе.

Процессы отладки и тестирования сопровождаются неоднократным запуском программы на выполнение. Процесс запуска может быть осуществлен только после того, как введенная в компьютер программа на алгоритмическом языке Pascal будет переведена в двоичный машинный код и создан исполняемый файл.

Процесс перевода текста программы в машинный код называют трансляцией. Все трансляторы делятся на два класса:

- интерпретаторы – трансляторы, которые переводят каждый оператор программы в машинный код и по мере перевода операторы выполняются процессором;
- компиляторы переводят всю программу целиком, и если этот перевод прошел без ошибок, то полученный двоичный код можно запускать на выполнение.

Если в качестве транслятора выступает компилятор, то процесс перевода текста программы в машинный код называют компиляцией.

При переводе программы с языка Pascal в машинный код используются именно компиляторы.

Рассмотрим основные этапы обработки компилятором программы на языке Pascal.

1. Компилятор анализирует, какие внешние библиотеки нужно подключить, разбирает текст программы на составляющие элементы, проверяет синтаксические ошибки и в случае их отсутствия формирует объектный код. Получаемый на этом этапе двоичный файл (объектный код) не включает в себя объектные коды подключаемых библиотек.

2. На втором этапе компоновщик подключает к объектному коду программы объектные коды библиотек и генерирует исполняемый код программы. Этот этап называется компоновкой, или сборкой, программы. Полученный на этом этапе исполняемый код можно запускать на выполнение.

2. Тип. Иерархия типов

Тип – это множество значений переменной вместе с множеством операций, которые можно выполнять над элементами этого множества.

Приписывая переменной некоторый тип, мы тем самым явно определяем множество значений, которые можно присвоить этой переменной, а также операции, с помощью которых можно манипулировать ее значениями.

Число отдельных значений, принадлежащих некоторому типу, называется мощностью типа. Так, например, мощность типа Boolean есть 2.

В языке программирования Pascal принято все типы данных разделять на три группы:

1. простейшие, или скалярные (к скалярным типам данных относятся типы данных таких величин, значения которых не содержат составных частей, т.е. простейшие типы: целочисленный, вещественный, логический, символьный, перечисляемый, интервальный (тип диапазон) и ссылочный);

2. структурированные (структурированный тип данных, или структуры данных, в своей основе имеет один или несколько скалярных типов данных, к ним относятся: строки, массивы, множества, записи, файлы);

3. процедурные и типа object. (этим типам данных трудно поставить в соответствие данные в обычном понимании этого слова. Их названия отражают их базовые признаки и используются, как правило, в объектно-ориентированном программировании, которое поддержано старшими версиями языка Pascal. Понимание работы с этими типами данных требует наличие определенного опыта и навыков программирования).

Рассмотрим некоторые скалярные (простейшие) типы данных.

1. Диапазон значений типа Byte: 0..255. Над данными типа Byte допустимы следующие операции:

сравнения (=, <>, <, <=, >, >=);

+ (сложение); + (одноместный (унарный) плюс);

- (вычитание); - (одноместный (унарный) минус);

* (умножение); DIV (деление (получение частного));

MOD (получение остатка (второй операнд должен быть положительным));

ShL (логический сдвиг влево); ShR (логический сдвиг вправо).

При логическом сдвиге биты, оказавшиеся за пределами разрядной сетки кода, отбрасываются, а вместо недостающих устанавливаются нули.

2. Диапазон значений типа Integer: -32768..32767. Над данными типа Integer допустимы те же операции, что и над данными типа Byte.

3. Диапазон значений типа Word: 0..65535. Над типом Word допустимы те же операции, что и над данными типа Integer.

4. Диапазон значений типа Real: 2.9E-39..1.7E+38 (по абсолютной величине). Данные типа Real имеют точность до одиннадцати значащих цифр. Над данными типа Real допустимы следующие операции:

сравнения (=, <>, <, <=, >, >=);

+ (сложение); + (одноместный (унарный) плюс);

- (вычитание); - (одноместный (унарный) минус);

* (умножение); DIV (деление (получение частного));

5. Тип данных Char определяет множество символов с кодами ASCII. Над данными типа Char допустимы операции сравнения (=, <>, <, <=, >, >=).

6. Данные типа Boolean могут принимать два значения: TRUE и FALSE. Над данными типа Boolean допустимы следующие операции:

сравнения (=, <>, <, <=, >, >=);

AND (логическое И); OR (логическое ИЛИ);

XOR (логическое исключающее ИЛИ); NOT (логическое отрицание).

3. Структура программы

Программа на языке Pascal состоит из заголовка, раздела описаний и раздела операторов.

PROGRAM Имя (Input,Output); Заголовок программы

label - раздел меток

const - раздел констант

type - раздел типов Раздел описаний

var - раздел переменных

procedure - раздел процедур

function - раздел функций

BEGIN

Оператор 1;

Оператор 2; Раздел операторов

... (тело программы)

Оператор n

END.

Заголовок содержит служебное слово PROGRAM, имя программы, задаваемое программистом, и в круглых скобках имена стандартных процедур Input, Output для связи программы с внешними устройствами ввода-вывода. Заканчивается заголовок символом ";" (точка с запятой).

Раздел описаний предназначен для объявления всех встречающихся в программе данных и их характеристик (имена данных, их тип, возможные значения). Этот раздел содержит следующие подразделы: объявление меток, констант, типов, переменных, объявление процедур и функций. Порядок расположения разделов не важен и описания могут повторяться.

Объявление процедур и функций является одним разделом. Следует заметить, что не все перечисленные разделы обязательны в программе. В простых программах могут потребоваться, например, только разделы описания констант и переменных.

После каждого описания ставится символ ";".

Раздел операторов ("тело" программы) заключается в операторные скобки вида: BEGIN ("начать") и END ("окончить"), при этом после служебного слова END обязательно ставится точка. В разделе операторов записывается последовательность исполняемых операторов и каждый выражает действие, которое необходимо выполнить. Исполняемые операторы отделяются друг от друга символом ";".

Хороший стиль программирования требует:

1. записывать слова PROGRAM, BEGIN, END с одинаковой позиции строки;
2. по отношению к ним описания и операторы принято сдвигать вправо на 3 символа. Желательно сдвиг делать на одинаковое количество позиций от края или по отношению к предыдущему сдвигу.

4.Выражения. Оператор присваивания

Выражения состоят из операций и операндов. Оператор присваивания-определяет процесс вычисления нового значения с помощью выражения и запоминание полученного значения переменной в ОП.

Большинство операций в языке Pascal являются бинарными, то есть содержат два операнда. Остальные операции являются унарными и содержат только один операнд. В бинарных операциях используется обычное алгебраическое представление, например: $a+b$. В унарных операциях операция всегда предшествует операнду, например: $-b$.

Операторы присваивания служат для вычисления значения выражения и присваивания его имени результату. Общий вид оператора присваивания:

$X:=a$; где X – имя переменной; a – выражение.

Например:

$X:=0$;

$P:=1$;

$N:=N+1$;

$X:=N * P$;

Основные свойства присваивания:

1. Значение переменной не определено, если ей не присвоено никакого значения;
2. Новое значение, присваиваемое переменной, заменяет ее старое значение;
3. Присвоенное переменной значение сохраняется в ней вплоть до нового присваивания.

Для ввода используются следующие процедуры:

Read (англ.) – читать

READ (a, b,...);

ReadLn (a, b,...);

ReadLn (a);

Для вывода используются следующие процедуры:

Write (англ.) – писать

WRITE(a, b,...);

WriteLn(a, b,...);

WriteLn ('a=',a);

Типы вводимых \ выводимых данных должны соответствовать типам переменных в списке процедуры ввода \ вывода.

В более сложных выражениях порядок, в котором выполняются операции, соответствует приоритету операций.

Старшинство операций

Операция	Приоритет	Вид операции
@, not	первый (высший)	унарная операция
*, /, div, mod, and, shl, shr	второй	операция умножения, деления, сдвига...
+, -, or, xor	третий	операция сложения
=, <, <=, >, >=, in	четвертый (низший)	операция отношения

Для определения старшинства операций имеется три основных правила:

1. Во-первых, операнд, находящийся между двумя операциями с различными приоритетами, связывается с операцией, имеющей более высокий приоритет.

2. Во-вторых, операция, находящаяся между двумя операциями с равными приоритетами, связывается с той операцией, которая находится слева от него.

3. В-третьих, выражение, заключенное в скобки, перед выполнением вычисляется, как отдельный операнд.

5. Операторы повтора

Если в программе возникает необходимость неоднократно выполнить некоторые операции, то используются операторы повтора (цикла). В языке Pascal различают три вида операторов цикла: While, repeat, for.

Они используются для организации циклов различных типов. Выражение, управляющее повторениями, должно иметь булевский тип («True(Истина)/False(Ложь)», «Да(1)/Нет(0)»).

Если число повторений заранее неизвестно, а задано лишь условие его повторения (или окончания), используются операторы while, repeat. Оператор for используется, если число повторений заранее известно.

Оператор повтора WHILE

Оператор while (пока) часто называют оператором цикла с предусловием за то, что проверка условия выполнения тела цикла производится в самом начале оператора. Формат записи:

While < условие продолжения повторений > do < тело цикла >;

Перед каждым выполнением тела цикла вычисляется значение выражения условия. Если результат равен True (Истина), тело цикла выполняется и снова вычисляется выражение условия. Если результат равен False (Ложь), происходит выход из цикла и переход к первому полю while оператору.

Оператор повтора REPEAT

Оператор повтора Repeat аналогичен оператору While, но отличается от него, во-первых, тем, что условие повторяется после очередного выполнения операторов тела цикла и таким образом гарантируется хотя бы однократное выполнение цикла, а во-вторых, что критерием прекращения цикла является равенство выражения константе True (Истина). За это цикл Repeat часто называют циклом с постусловием или циклом «До», та

как он прекращает выполняться, как только значение выражения условия, записанного после слова Until равно True (Истина).

Формат записи:

Repeat

< оператор; >

...

оператор >

Until < условие окончания цикла >;

Операторы, заключенные между словами Repeat и Until, являются телом цикла. Вначале выполняется тело цикла, затем проверяется условие выхода из цикла. Если результат булевского выражения равен False, то тело цикла активизируется еще раз; если результат True, происходит выход из цикла.

Оператор повтора FOR

В случаях, когда число повторений может быть за ранее известно, то для организации циклической обработки информации применяется оператор повтора For. Часто этот оператор повтора называют оператором цикла с параметром, так как число повторений задается переменной, называемой параметром цикла, или управляющей переменной.

Оператор For состоит из заголовка и тела цикла.

Формат записи:

For < параметр цикла >: = < S1> to < S2> do < оператор >;

или

For < параметр цикла >: = < S1> downto < S2> do < оператор >;

Где S1 и S2 – выражения, определяющие соответственно начальное и конечное значения параметра цикла;

а) For ... do – заголовок цикла, который определяет диапазон изменения значений управляющей переменной и одновременно число повторений оператора, содержащегося в теле цикла, направление изменения значений параметра цикла (возрастание – to, убывание – downto).

б) < оператор > - тело цикла, которое может быть простым или составным оператором. Оператор For обеспечивает выполнение тела цикла до тех пор, пока не будут перебраны все значения параметра цикла от начального до конечного.

1.11 Лекция № 15 (2 часа).

Тема: «Программные средства реализации информационных процессов. Прикладное программное обеспечение»

1.11.1 Вопросы лекции:

1. Классификация прикладного программного обеспечения
2. Понятие ППП
3. Структура и основные компоненты ППП
4. Этапы развития ППП

1.11.2 Краткое содержание вопросов:

1. Классификация прикладного программного обеспечения

Общее понятие ПО для ПК.

Персональный компьютер, как известно, является универсальным устройством для обработки информации. Персональные компьютеры могут выполнять любые действия по обработке информации. Для этого необходимо составить для компьютера на понятном ему языке точную и подробную последовательность инструкций – программу, как надо обрабатывать информацию.

Меняя программы для компьютера, можно превратить его в рабочее место бухгалтера или конструктора, дизайнера или ученого, писателя или агронома. Кроме того, тенденция понижения стоимости компьютерной техники при одновременном росте ее производительности привела к тому, что компьютеры становятся предметом домашнего обихода, как, например, телевизор или холодильник, что расширяет сферу применения ПК еще больше. Соответственно, требуется все более разнообразное программное обеспечение для решения задач в новых областях применения ПК. Непрерывное повышение мощности персональных компьютеров, периферийных устройств, а также развитие средств связи дает разработчикам программного обеспечения все больше возможностей для максимально полного удовлетворения запросов конечных потребителей. Это и ставший стандартом графический интерфейс для любого ПО, и внедренные возможности для отправки документов и данных с помощью Интернет непосредственно из прикладной программы (Microsoft Word , Excel , Access и др.), и возможность использования компьютера как хранилища информации благодаря появлению новых видов накопителей большой емкости и малым временем доступа к данным, а также многие другие возможности и сервисные функции.

При своем выполнении программы могут использовать различные устройства для ввода и вывода данных, подобно тому, как человеческий мозг пользуется органами чувств для получения и передачи информации. Сам по себе ПК не обладает знаниями ни в одной области своего применения, все эти знания сосредоточены в выполняемых на нем программах. Поэтому часто употребляемое выражение «компьютер сделал» означает ровно то, что на ПК была выполнена программа, которая позволила выполнить соответствующее действие. В настоящее время весь комплекс ПО делится на системные и пользовательские программы. Системное программное обеспечение выполняет функции «организатора» всех частей ПК, а также подключенных к нему внешних устройств. Программы для пользователей служат для выполнения каких – либо конкретных задач во всех сферах человеческой деятельности.

Прикладное программное обеспечение.

Данный класс программных средств наиболее разнообразен, что обусловлено, прежде всего, широким применением средств компьютерной техники во всех сферах деятельности человека, созданием автоматизированных информационных систем различных предметных областей.

Классификация прикладного программного обеспечения

Проблемно-ориентированные ППП

Программные продукты данного класса можно классифицировать по разным признакам:

- типам предметных областей;
- типам информационным системам;
- функциям и комплексам задач, реализуемых программным способом, и др.

Для некоторых предметных областей возможна типизация функций управления, структуры данных и алгоритмов обработки. Это вызвало разработку значительного числа ППП одинакового функционального назначения и, таким образом, создало рынок программных продуктов:

- ППП автоматизированного бухгалтерского учета;
- ППП финансовой деятельности;
- ППП управления персоналом (кадровый учет);
- ППП управления материальными запасами;

ППП управления производством;
банковские информационные системы и т. п.

Основные тенденции в области развития проблемно-ориентированных программных средств:

- создание программных комплексов в виде автоматизированных рабочих мест (АРМ) управленческого персонала;
- создание интегрированных систем управления предметной областью на базе вычислительных сетей, объединяющих АРМы в единый программный комплекс с архитектурой "клиент – сервер";
- организация данных больших информационных систем в виде распределенной базы данных в сети ЭВМ;
- наличие простых языковых средств конечного пользователя для запросов к базе данных;
- создание программного обеспечения, позволяющего настраивать функции обработки данных конечными пользователями (без участия программистов);
- защита программ и данных от несанкционированного доступа (парольная защита на уровне функций, режимов работы, данных).

Для подобного класса программ высоки требования к оперативности обработки данных (например, пропускная способность для банковских систем должна составлять несколько сот транзакций в секунду). Велики объемы хранимой информации, что обуславливает повышенные требования к средствам администрирования данных БД (актуализации, копирования, обеспечения производительности обработки данных).

Наиболее важно для данного класса программных продуктов создание дружественного интерфейса для конечных пользователей.

Данный класс программных продуктов развивается как в плане реализуемых ими функций, так и в плане используемого для их создания инструментария разработчика. Со временем границы компьютеризации информационных систем, как правило, расширяются, что приводит к изменению функций существующих ППП.

ППП автоматизированного проектирования

Программы этого класса предназначены для поддержания работы конструкторов и технологов, занимающихся построением чертежей, схем, диаграмм, графическим модулированием и конструированием, созданием библиотеки стандартных элементов чертежей и их многократным использованием, созданием демонстрационных иллюстраций и мультфильмов.

Отличительными особенностями этого класса программных продуктов являются высокие требования к технической части системы обработки данных, наличие библиотек встроенных функций, объектов, интерфейсов с графическими системами и базами данных.

ППП общего назначения

Данный класс содержит широкий перечень программных продуктов, поддерживающих преимущественно информационные технологии конечных пользователей. Кроме конечных пользователей, этими программными продуктами, благодаря встроенным средствам технологии программирования, могут пользоваться и программисты для создания усложненных программ обработки данных.

Представители данного класса программных продуктов – настольные системы управления базами данных (СУБД), обеспечивающие организацию и хранение локальных баз данных на автономно работающих компьютерах, либо централизованное хранение баз данных на файл-сервере и сетевой доступ к ним.

В настоящее время наиболее широко представлены реляционные СУБД для персональных компьютеров, осуществляющие:

- работу с базой данных через экранные формы;
- организацию запросов на поиск данных с помощью специальных языков запросов высокого уровня;

- генерацию отчётов различной структуры данных с подведением промежуточных и окончательных итогов;

- вычислительную обработку путём использования встроенных функций, программ, написанных с использованием языков программирования и макрокоманд.

Пользовательские приложения (прикладные программы), функционирующие в среде СУБД, создаются по типу меню работы конечного пользователя, каждая команда которого обеспечивает автоматизированное выполнение определенной функции. В современных СУБД (например, в СУБД Access 2.0) содержатся элементы CASE-технологии проектирования, в частности:

- визуализирована схема баз данных;
- осуществлена автоматическая поддержка целостности баз данных при различных видах обработки (включение, удаление или модификация данных баз данных);
- предоставляются так называемые "мастера", обеспечивающие поддержку процесса проектирования (режим "конструктор") – "мастер таблиц", "мастер форм", "мастер отчётов", "построитель меню" и т. п.;
- созданы для широкого использования прототипы (шаблоны) структур баз данных, форм, отчетов и т. д.

Всё это свидетельствует о расширении функциональных возможностей СУБД как инструментального средства для создания приложений.

Серверы баз данных – успешно развивающийся вид программного обеспечения, предназначенный для создания и использования при работе в сети интегрированных баз данных в архитектуре "клиент – сервер". Многопользовательские СУБД (типа Paradox, Access, FoxPro и др.) в сетевом варианте обработки данных хранят информацию на файл-сервере – специально выделенном компьютере в централизованном виде, но сама обработка данных ведется на рабочих станциях. Серверы баз данных, напротив, всю обработку данных (хранение, поиск, извлечение и передачу клиенту) выполняют самостоятельно, одновременно обеспечивая данными большое число пользователей сети. Общим для различных видов серверов баз данных является использование реляционного языка SQL (Structured Query Language) для реализации запросов к данным.

Большинство серверов баз данных может использовать одновременно несколько платформ (Windows NT, Unix, OS/2 и др.), поддерживает широкий спектр протоколов передачи данных (IPX, TCP/IP, X.25 и др.).

Некоторые серверы реализуют распределенное хранение информации в сети, поддерживают интерфейсы на уровне вызова типа:

ODBC – Open Data Base Connectivity – для доступа к разнородным базам данных;

DAL – Data Access Language – для создания запроса на выборку данных, распределённых в сети;

SAG/CLI – SQL Access Group/Call Level Interface – для распределённых запросов и др.

Самыми большими проблемами применения серверов баз данных являются обеспечение целостности (непротиворечивости) баз данных, решение вопроса, связанного с дублированием (тиражированием) данных по узлам сети и их синхронным обновлением.

Генераторы (серверы) отчётов – программные средства, имеющие самостоятельное направление развития, обеспечивающие реализацию запросов и формирование отчётов в печатном или экранном виде в условиях сети с архитектурой "клиент – сервер".

Сервер отчётов подключается к серверу баз данных, используя все уровни передач и драйверы сервера баз данных. Серверы отчётов включают:

- программы планирования используются для учёта времени для формирования отчётов по требованию пользователей, составления расписания выдачи и распространения отчётов по сети;
- программы управления очередью запросов на формирование отчётов;

- программы ведения словаря пользователей для разграничения доступа к сформированным отчётам;
- программы ведения архива отчётов и др.

Подготовленные отчёты рассылаются клиентам по электронной почте или с помощью другого транспортного агента. Серверы отчетов обычно поддерживают разнородные платформы, тем самым они эффективно работают в неоднородных вычислительных сетях.

Текстовые процессоры – программы, используемые для автоматического форматирования документов, вставки рисованных объектов и графики в текст, составления оглавлений и указателей, проверки орфографии, шрифтового оформления, подготовки шаблонов документов. Примером развития данного направления программных продуктов являются издательские системы.

Табличный процессор – программы для вычислений силами конечного пользователя; средства деловой графики, программы специализированной обработки (встроенные функции, работа с базами данных, статистическая обработка данных и др.).

Средства презентационной графики – специализированные программы, предназначенные для создания изображений и их показа на экране, подготовки слайд-фильмов, мультфильмов, видеофильмов, их редактирования, определения порядка следования изображений. Презентация может включать показ диаграмм и графиков. Все программы презентационной графики условно делятся на программы для подготовки слайд-шоу и программы для подготовки мультимедиа-презентации. Для работы этих программ необходимо также наличие специализированного оборудования – LCD (Liquid Crystal Desktop) – жидкокристаллической проекционной панели, которая просвечивается проектором для вывода изображения на экран, видеотехника.

Презентация требует предварительного составления плана показа. Для каждого слайда выполняется проектирование: определяются содержание слайда, размер, состав элементов, способы их оформления и т. п. Данные для использования в слайдах можно готовить как вручную, так и получать в результате обмена из других программных систем.

Интегрированные пакеты – набор нескольких программных продуктов, функционально дополняющих друг друга, поддерживающих единые информационные технологии, реализованные на общей вычислительной и операционной платформе.

Наиболее распространены интегрированные пакеты, компонентами которых являются:

- СУБД;
- текстовый редактор;
- табличный процессор;
- органайзер;
- средства поддержки электронной почты;
- программы создания презентаций;
- графический редактор.

Компоненты интегрированных пакетов могут работать изолированно друг от друга, но основные достоинства интегрированных пакетов проявляются при их разумном сочетании друг с другом. Пользователи интегрированных пакетов имеют унифицированный для различных компонентов интерфейс, тем самым обеспечивается относительная легкость процесса их освоения.

Отличительными особенностями данного класса программных средств являются:

- полнота информационных технологий для конечных пользователей;
- однотипный интерфейс конечного пользователя для всех программ, входящих в состав интегрированного пакета – общие команды в меню, стандартные пиктограммы одних и тех же функций (сохранение на диске, печать, проверка орфографии, шрифтовое оформление и т. п.), стандартное построение и работа с диалоговыми окнами и др.;

- общий сервис для программ интегрированного пакета (например, словарь и средства орфографии для проверки правописания, построитель диаграмм, конвертер данных и др.);

- легкость обмена и ссылок на объекты, созданные программами интегрированного пакета (применяется два метода: DDE – динамический обмен данными и OLE – динамическая компоновка объектами), единообразный перенос объектов (метод drag-and-drop);

- наличие единой языковой платформы для разбора макрокоманд, пользовательских программ;

- возможность создания документов, интегрирующих в себе возможности различных программ, входящих в состав интегрированного пакета.

Интегрированные пакеты эффективны и при групповой работе в сети многих пользователей. Так, из прикладной программы, в которой работает пользователь, можно отправить документы и файлы данных другому пользователю, при этом поддерживаются стандарты передачи данных в виде объектов по сети или через электронную почту.

Методо-ориентированные ППП

Данный класс включает программные продукты, обеспечивающие, независимо от предметной области и функции информационных систем, математические, статические и другие методы решения задач. Наиболее распространены методы математического программирования, решение дифференциальных уравнений, имитационного моделирования, исследования операций.

Методы статистической обработки и анализа данных (описательная статистика, регрессионный анализ, прогнозирование значений технико-экономических показателей и т. п.) имеют широкое применение. Так, современные табличные процессоры значительно расширили набор встроенных функций, реализующих статистическую обработку и информационные технологии статистического анализа. Вместе с тем необходимость в использовании специализированных программных средств статистической обработки, обеспечивающих высокую точность и многообразие статистических методов, также растёт. На базе методов сетевого планирования с экономическими показателями проекта, формированием отчётов различного вида оформилось новое направление программных средств – управление проектами, пользователями этих программ являются менеджеры проектов.

Офисные ППП

Данный класс программных продуктов включает программы, обеспечивающие организационное управление деятельностью офиса:

Организаторы (планировщики) – программное обеспечение для планирования рабочего времени, составления протоколов встреч, расписаний, ведения записной и телефонной книжки.

В состав программ-организаторов входят: калькулятор, записная книжка, часы, календарь и т. п. Наиболее часто подобное программное обеспечение разрабатывается для ноутбуков, персональных компьютеров блокнотного типа.

Программы-переводчики, средства проверки орфографии и распознавания текста включают:

- программы-переводчики, предназначенные для создания подстрочника исходного текста на указанном языке;

- словари орфографии, используемые при проверке текстов;

- словари синонимов, используемые для стилиевой правки текстов;

- программы для распознавания считанной сканерами информации и преобразования в текстовое представление. К ним относятся:

- ППП OCR Cunei Form 2.0 – обеспечивает распознавание смешанных русско-английских текстов, в формате RTF сохраняется как текст, так и иллюстрации;

- ППП OCR Tiger – шрифтовая обучаемая система распознавания русского языка с возможностью автоматического выбора шрифта из библиотеки, обеспечивает многостраничный ввод текстов;

- ППП Stylus Lingvo Office – реализует весь цикл «от листа до листа» – с помощью сканера осуществляется считывание текстового изображения, находящегося на печатном листе; FineReader осуществляет распознавание оптических образов и запись считанной информации в текстовом виде; Stylus for Windows выполняет перевод на указанный язык; корректор орфографии Lingvo Corrector и резидентный словарь Lingvo осуществляет проверку и правку. Результат перевода представляется в формате текстового редактора Word for Windows.

Коммуникационные ППП – предназначены для организации взаимодействия пользователя с удаленными абонентами информационной сети. В условиях развития глобальной информационной сети Internet появился новый класс программного обеспечения – браузеры, средства создания WWW-страниц. Они различаются возможностями поддержки языка HTML, использованием цвета при оформлении фона, текста, форматированием текста, использованием графических форматов изображений, таблиц, фонового звука, мультипликации и т. п. Большинство браузеров используют язык Java. Электронная почта также становится обязательным компонентом офисных ППП. Наиболее широко распространенные ППП электронной почты:

Eudora 2.1.2 (Qualcomm);

Pegasys Mail 2.4 (David Haris);

DML (DEMOS).

Они различаются платформами, на которых работают (DOS, Windows 3.x, Windows 95, Windows NT, Macintosh), ценой и условиями распространения, поддерживаемыми транспортными протоколами, интерфейсом с сетями. Электронная почта должна обеспечивать шифрование передаваемой информации, факсимиле подписи, проверку орфографии на любом из языков, управление сообщениями по электронной почте (оповещение о новой почте, организация почтовых ящиков, поиск, цитирование корреспонденции и т. д.).

Настольные издательские системы

Данный класс программ включает программы, обеспечивающие информационную технологию компьютерной издательской деятельности:

- форматирование и редактирование текстов;
- автоматическую разбивку текста на страницы;
- создание заголовков;
- компьютерную верстку печатной страницы;
- монтирование графики;
- подготовку иллюстраций и т. п.

ППП Adobe Page Maker 6.0 обеспечивает подготовку многостраничных цветных публикаций, гибкий дизайн страниц, высококачественную печать. Расширены возможности по верстке: неограниченное число страниц-шаблонов, которые могут использоваться в одной публикации; применение различных эффектов к цветным изображениям; настройка резкости и регулировка цветов в импортированных файлах; возможно закрепление расположения объектов на странице, автоматическое выравнивание объектов.

Разработаны и включены новые цветовые библиотеки, используются новые технологии HiFi Color и PANTONE Hexachrome, которые расширяют цветовую гамму традиционной офсетной печати.

Программные средства мультимедиа

Этот класс программных продуктов является относительно новым, он сформировался в связи с изменением среды обработки данных, появлением лазерных

дисков высокой плотности записи с хорошими техническими параметрами по доступным ценам, расширением состава периферийного оборудования, подключаемого к персональному компьютеру, развитием сетевой технологии обработки, появлением региональных и глобальных информационных сетей, располагающих мощными информационными ресурсами. Основное назначение программных продуктов мультимедиа – создание и использование аудио- и видеоинформации для расширения информационного пространства пользователя.

Программные продукты мультимедиа заняли лидирующее положение на рынке в сфере библиотечного информационного обслуживания, процесса обучения, организации досуга. Базы данных компьютерных изображений произведений искусства, библиотеки звуковых записей и будут составлять основу для прикладных обучающих систем, компьютерных игр, библиотечных каталогов и фондов.

Системы искусственного интеллекта

Данный класс программных продуктов реализует отдельные функции интеллекта человека. Основными компонентами систем искусственного интеллекта являются база знаний, интеллектуальный интерфейс с пользователем и программа формирования логических выводов. Их разработка идет по следующим направлениям:

- программы-оболочки для создания экспертных систем путем наполнения баз знаний и правил логического вывода;
- готовые экспертные системы для принятия решений в рамках определенных предметных областей;
- системы управления базами знаний для поддержания семантических моделей (процедуральной, семантической сети, фреймовой, продукционной и др.).

Как правило, интеллектуальный интерфейс включает:

- диалоговый процессор на естественном языке;
- планировщик, преобразующий описание задачи в программу решения на основе информации базы знаний;
- монитор, осуществляющий управление компонентами интерфейса.

2. Понятие ППП

Многочисленные программные средства для решения различных типов вычислительных задач можно разделить на 4 группы:

- отдельные прикладные программы;
- библиотеки прикладных программ;
- пакеты прикладных программ;
- интегрированные программные системы.

Рассмотрим по порядку каждую из этих групп. Отдельная прикладная программа пишется, как правило, на некотором универсальном языке программирования (Паскаль и т.п.) и предназначается для решения конкретной прикладной задачи. Примерами могут служить программа решения системы линейных алгебраических уравнений тем или иным численным методом, программа вычисления собственных значений матрицы и т. д. Авторами таких программ являются прикладные программисты, специализирующиеся в соответствующих предметных областях. Прикладная программа может быть реализована в виде набора модулей, каждый из которых выполняет некоторую самостоятельную функцию. Например, программа вычисления собственных значений матрицы может включать модули, - реализующие преобразования матрицы из одной, формы представления в другую ввод и вывод данных, обработку аварийных ситуаций с выдачей диагностических сообщений пользователю и другие действия.

Библиотека представляет собой набор отдельных программ, каждая из которых решает некоторую прикладную задачу или выполняет определенные вспомогательные функции (управление памятью, обмен с внешними устройствами и т.п.). Библиотеки

программ зарекомендовали себя эффективным средством решения вычислительных задач. Они интенсивно используются при решении научных и инженерных задач с помощью ЭВМ. Условно их можно разделить на библиотеки широкого применения и специализированные библиотеки.

Программы, входящие в состав библиотеки широкого применения, предназначены для решения задач из различных предметных областей.

Специализированные библиотеки ориентированы на решение отдельных, порой достаточно узких, классов задач.

Переход от разработки отдельных прикладных программ к созданию библиотек программ поставил перед разработчиками ряд проблем как системного, так и прикладного характера. К числу основных проблем, возникающих на различных этапах конструирования библиотеки, относятся проблемы систематизации, документирования, тестирования и переносимости.

Проблема систематизации состоит в разбиении библиотеки на разделы и подразделы в соответствии с классификацией задач предметной области и методов их решения. Сюда входит выбор единых правил наименования программ (с учетом модификации и развития библиотеки), единых форм представления и наименования математически сходных объектов, единой схемы контроля ошибок и т. д. Проблема документирования заключается в составлении единых правил описания программ библиотеки. Наличие качественной документации существенно упрощает доступ к отдельным программам, организацию взаимодействия между программами, включение новых программ. Важную роль при решении указанной проблемы играют средства автоматизации документирования, обеспечивающие широкое применение шаблонов (для титульных листов, фрагментов текста и т. п.), использование текстов программ библиотеки для автоматизированного составления документации и т. д.

Тестирование библиотеки состоит в проверке программ на специально подготовленных тестовых данных. Результаты тестирования в большой мере зависят от правильности и полноты набора тестов. Тестирование, в частности, включает проверку соответствия текста программы выбранному стандарту языка программирования, определение области применимости программы и качества диагностического аппарата, выявление разнообразных количественных характеристик, программы (скорость работы, точность получаемых результатов и т. п.) сравнение с другими программами для решения той же задачи.

Проблема переносимости состоит в разработке методов и средств, обеспечивающих возможность использования программ библиотеки в различных вычислительных условиях (на различных типах ЭВМ, в различных операционных системах и т. д.) Эта проблема включает в себя такие аспекты, как следование стандарту языка программирования, организация работы с машинно-зависимыми константами, создание инструментальных средств, позволяющих автоматизировать перенос программ из одной вычислительной среды в другую. Доступ к программам библиотеки осуществляется с использованием штатных средств того или иного языка программирования. Разработка библиотек программ обычно осуществляется силами прикладных программистов. При этом нередко в библиотеку включаются программы, написанные в разное время и разными авторами. Отсутствие в составе библиотеки специализированного системного обеспечения позволяет в большинстве случаев при ее конструировании обойтись без какой-либо существенной помощи системных программистов (они могут привлекаться, например, для написания лишь отдельных служебных программ).

Таким образом, характерной особенностью библиотек программ является отсутствие проблемно-ориентированного входного языка и достаточно развитого системного обеспечения. Как правило, библиотеки программ ориентированы на типовые задачи предметной области и не содержат средств решения специфических прикладных

задач (в первую очередь это относится к библиотекам широкого применения программы которых могут использоваться для решения задач из различных предметных областей).

Перейдем теперь к рассмотрению пакетов прикладных программ (ППП) как самостоятельной формы прикладного программного обеспечения. Для этого прежде всего необходимо уточнить само понятие пакета. В настоящее время не существует признанной всеми специалистами единой точки зрения по этому вопросу. Отсутствует также единая терминология в пакетной проблематике. Это объясняется прежде всего новизной данного научного направления, которое сложилось, в основном за последние 30 лет (приблизительно с начала 70-х годов). Кроме того, различные определения ППП рассматривают это понятие с разных точек зрения, выделяя те или иные функциональные или структурные особенности пакетов. ППП определяется и как совокупность программ для решения определенного класса задач, к которой обращаются при помощи простой символики (языка) и как совокупность программ, совместимых по структуре данных, способам управления, объединяемых общностью функционального назначениями представляющих собой средство решения класса задач определенным кругом пользователей. При этом под классом задач понимается множество прикладных проблем, обладающих общностью применяемых алгоритмов и информационных массивов, а также определение пакета как комплекса взаимосвязанных программ, обладающих специальной организацией, которая обеспечивает значительное повышение производительности труда программистов и пользователей пакета. В данном случае не делается попытки выделить ППП среди других форм программного обеспечения ЭВМ.

Будем считать пакетом программ любой комплекс, ориентированный на решение некоторого класса задач. Формально такое определение не исключает из числа пакетов и библиотеки программ. Однако сложившееся на сегодняшний день представление о ППП как о самостоятельной форме программного обеспечения, позволяет указать на ряд характерных отличительных особенностей пакетов. Одной из главных особенностей является ориентация ППП не на отдельную задачу, а на некоторый класс задач, включающий и специфические задачи предметной области. Отсюда следует необходимость, модульной организации ППП как основного технологического принципа его конструирования. Суть этого принципа состоит в оформлении общих фрагментов используемых алгоритмов в виде самостоятельных модулей. Решение сформулированной пользователем задачи осуществляется некоторой "цепочкой" таких модулей.

Другой особенностью ППП является наличие в его составе специализированных языковых средств, обеспечивающих удобную работу пользователя с пакетом. Как правило, развитый пакет обладает несколькими входными языками, ориентированными на выполнение различных функций и различные типы пользователей. Язык может предназначаться для формулировки исходной задачи, описания алгоритма решения и начальных данных, организации доступа и поддержания базы данных или информационной базы ППП, разработки программных модулей, описания модели предметной области, управления процессом решения в диалоговом режиме и других целей.

Еще одна особенность ППП состоит в наличии специальных системных средств, обеспечивавших принятую в предметной области дисциплину работы. К их числу относятся специализированные банки данных, средства информационного обеспечения, средства взаимодействия пакета с операционной системой и т. п.

Наконец, интегрированной программной системой назовем комплекс программ, элементами которого являются различные пакеты и библиотеки программ. Примером служат системы автоматизированного проектирования, имеющие в своем составе несколько ППП различного назначения. Часто в подобной системе решаются задачи, относящиеся к различным классам или даже к различным предметным областям.

Следует указать на отсутствие четких и однозначных границ между перечисленными формами прикладного программного обеспечения. Так, отдельная

прикладная программа, ориентированная на решение класса задач и оформленная в виде совокупности модулей может рассматриваться как библиотека или даже пакет программ несмотря на отсутствие специализированных языковых и системных средств. Переход от создания библиотек программ к разработке ППП был вызван целым рядом причин. К их числу прежде всего относится резкое увеличение возможностей ЭВМ. Это привело к значительному усложнению системного обеспечения вычислительных машин. Произошли существенные изменения в большинстве областей применения ЭВМ.

3. Структура и основные компоненты ППП

Несмотря на большое разнообразие конкретных пакетных разработок, можно выделить следующие основные компоненты ППП:

- входные языки;
- предметное обеспечение;
- системное обеспечение.

Важно отметить, что такое разбиение на составные элементы отражает в первую очередь функции, выполняемые программами ППП, а не структуру самих программ, которая зависит от индивидуальных особенностей конкретного пакета. В разных пакетах указанные компоненты могут быть развиты в различной степени или вовсе отсутствовать. Однако наиболее развитые ППП, как правило, обладают всеми этими компонентами, каждый из которых может иметь довольно сложную структуру. В многочисленных работах, посвященных пакетной проблематике, из-за не устоявшейся терминологии нередко используются другие названия составных элементов ППП. Например, входной язык называют также языком заданий или языком управления. Для обозначения предметного обеспечения применяются термины "функциональное наполнение", "функциональная подсистема" или "тело пакета". Системное обеспечение часто называют системным наполнением, организующей или управляющей программой, а также процессором пакета.

Рассмотрим функции каждого из компонентов ППП.

Входные языки представляют собой средство общения пользователя с пакетом. Как отмечалось раньше развитый пакет может обладать несколькими входными языками, предназначенными для выполнения различных функций и ориентированными на различные типы пользователей. Можно выделить следующие основные типы пользователей ППП:

Разработчик ППП, осуществляющий его модификацию и развитие с учетом изменения круга пользователей, класса решаемых задач (появление новых типов задач, развитие численных методов, модификация форм проведения работ и т. д.), а также состава аппаратного и программного обеспечения ЭВМ:

Ответственный за сопровождение, в функции которого входит поддержание пакета в работоспособном состоянии в условиях конкретной вычислительной системы (обеспечение сохранности программ и массивов данных, своевременное дублирование информационных файлов, выявление ошибок в программах пакета).

Администратор, отвечающий за организацию доступа пользователей к пакету, содержимое базы данных, защиту информации от несанкционированного доступа;

Конечный пользователь, применяющий пакет для решения конкретных прикладных задач.

Входные языки отражают объем и качество предоставляемых пакетом средств, а также удобство их использования. Таким образом, с точки зрения конечного пользователя именно входной язык является основным показателем возможностей ППП.

В качестве входных языков могут использоваться как универсальные, так и специализированные языки программирования. Например, в качестве входного языка разработчика ППП для написания прикладных и системных программ пакета обычно

используется тот или иной универсальный язык программирования (Фортран, Паскаль). В то же время входной язык конечного пользователя в развитом пакете, как правило, является языком качественно более высокого уровня по сравнению с универсальными языками. Изобразительные средства такого языка учитывают особенности задач предметной области и специфику пользователей. Подобные языки называют проблемно-ориентированными, или предметно-ориентированными.

Перейдем теперь к рассмотрению других компонентов ППП, конкретная прикладная деятельность характеризуется двумя факторами:

- классом решаемых задач и используемых для этих целей методов.
- дисциплиной работы, т.е. совокупностью правил, соглашений и технологических приемов, принятых при разработке, отладке, эксплуатации программ.

Предметное обеспечение представляет собой компонент пакета, отражающий особенности первого из этих факторов, т. е. особенности конкретной предметной области. Предметное обеспечение включает:

- программные модули, реализующие алгоритмы (или их отдельные фрагменты) решения прикладных задач;
- средства сборки программ из отдельных модулей;

Определение состава библиотеки модулей и форм их взаимодействия между собой является одной из наиболее трудоемких задач при построении ППП. Ее решение предполагает проведение тщательного и квалифицированного модульного анализа используемых алгоритмов. Удачно проведенный модульный анализ в значительной степени влияет на полноту охвата предметной области, а также на возможность расширения класса решаемых задач. Таким образом, выделение модулей существенно зависит от специфики задач и используемых алгоритмов и, по сути дела, отражает принятый в пакет способ сборки программ.

Наиболее распространено в настоящее время оформление каждого модуля в виде программной единицы на том или ином языке программирования. Такой модуль обеспечивает решение некоторой самостоятельной задачи и связан с другими модулями лишь входной и выходной информацией. Организация предметного обеспечения в виде библиотеки программ характерна для большинства существующих ППП.

Помимо рассмотренного подхода к оформлению модулей как программных единиц используются и другие способы.

Системное обеспечение представляет собой совокупность системных средств (программы, файлы, таблицы и т. д.), обеспечивающих определенную дисциплину работы пользователя при решении прикладных задач. По своей роли в составе ППП и выполняемым функциям системное обеспечение по существу является специализированной операционной системой, определяющей операционное окружение пакета. Несмотря на многообразие способов реализации системного обеспечения в рамках конкретных пакетных разработок, можно выделить его следующие основные компоненты:

- монитор, управляющий процессом решения и взаимодействием всех компонентов ППП;
- трансляторы с входных языков;
- средства работы с данными;
- средства информационного обеспечения, реализующие выдачу разнообразной справочной информации как по запросам пользователей (о структуре и возможностях ППП, о допущенных ошибках и т.д.), так и по запросам различных компонентов пакета (например, сведения о свойствах модулей предметного обеспечения, необходимые планировщику вычислений);
- различные служебные программы, в том числе реализующие взаимодействие пакета с операционной системой (работа с внешней памятью, средства ввода/вывода, драйверы специализированных, внешних устройств и др.).

В конкретном ППП, как правило, отсутствует четкое структурное разделение программ на предметное и системное обеспечение. Например, программа планирования вычислений может одновременно выполнять те или иные служебные функции {информационное обеспечение, связь с операционной системой и т. п.) и тем самым носиться как к предметному, так и к системному обеспечению. Кроме того, одни и те же программы в одном пакете могут относиться к предметному обеспечению, а в другом - к системному. Так, программы вывода графиков в рамках специализированного пакета машинной графики естественно отнести к предметному обеспечению. Однако те же программы следует считать служебными и относящимися к системному обеспечению, например, в пакете решения вычислительных задач).

4. Этапы развития ППП

Пакетная проблематика в качестве самостоятельного научно направления сложилась в основном за последние 15-20 лет. Первые ППП представляли собой простые тематические подборки программ для решения отдельных задач в той или иной прикладной области. Современный пакет является сложной программной системой, включающей специализированные системные и языковые средства. В относительно короткой истории развития вычислительных ППП можно выделить 4 основных поколения (класса) пакетов. Каждый из этих: классов характеризуется определенными особенностями входящих состав ППП компонентов - входных языков, предметного и системного обеспечения.

В качестве входных языков ППП первого поколения использовались универсальные языки программирования (Фортран, Алгол-60 и т. п.) или языки управления заданиями соответствующих операционных систем. Проблемная ориентация входных языков достигалась за счет соответствующей мнемоники в именах переменных, функций процедур, а также в текстовых константах. Составление заданий на таком языке практически не отличалось от написания программ на алгоритмическом языке.

Предметное обеспечение первых ППП, как правило, было организовано в форме библиотек программ, т.е. в виде наборов (пакетов) независимых программ на некотором базовом языке программирования (отсюда впервые возник и сам термин "пакет"). Такие ППП иногда называют пакетами библиотечного типа, или пакетами простой структуры.

В качестве системного обеспечения пакетов первого поколения обычно использовались штатные компоненты программного обеспечения ЭВМ: компиляторы с алгоритмических языков, редакторы текстов, средства организации библиотек программ, архивные системы и т. д. Эти пакеты не требовали сколько-нибудь развитой системной поддержки, и для их функционирования вполне хватало указанных системных средств общего назначения. В большинстве случаев разработчиками таких пакетов были прикладные программисты, которые пытались приспособить универсальные языки программирования к своим нуждам.

Разработка ППП второго поколения осуществлялась уже с участием системных программистов. Это привело к появлению специализированных входных языков (их называют встроенными языками) на базе универсальных языков программирования. Проблемная ориентация таких языков достигалась не только за счет использования определенной мнемоники, но также применением соответствующих языковых конструкций, которые упрощали формулировку задачи и делали ее более наглядной. Транслятор с такого языка представлял собой препроцессор (чаще всего макропроцессор) к транслятору соответствующего алгоритмического языка.

В качестве модулей в пакетах этого класса стали использоваться не только программные единицы (т.е. законченные программы на том или ином языке программирования), но и такие объекты, последовательность операторов языка программирования, совокупность данных, схема счета и др.

Существенные изменения претерпели также принципы организации системного обеспечения ППП. В достаточно развитых пакетах второго поколения уже можно выделить элементы системного обеспечения, характерные для современных пакетов: монитор, трансляторы с входных языков, специализированные банки данных, средства описания модели предметной области и планирования вычислений и др.

Третий этап развития ППП характеризуется появлением самостоятельных входных языков, ориентированных на пользователей-непрограммистов. Особое внимание в таких ППП уделяется системным компонентам обеспечивающим простоту и удобство. Это достигается главным образом за счет такой специализации входных языков и включения в состав пакета средств автоматизированного планирования вычислений.

Наконец, четвертый этап характеризуется созданием ППП, эксплуатируемых в диалоговом режиме работы. Основным преимуществом диалогового взаимодействия с ЭВМ является возможность активной обратной связи с пользователем в процессе постановки задачи, ее решения и анализа полученных результатов. Появление и интенсивное развитие различных форм диалогового общения обусловлено прежде всего прогрессом в области технических средств обеспечения диалога. Сюда относится создание разнообразной дисплейной техники (растровые дисплеи, средства реализации графических, цветовых и звуковых возможностей, различные технические устройства для ведения диалога и т. д.), а также надежных и скоростных линий связи. Развитие аппаратного обеспечения повлекло за собой создание разнообразных программных средств поддержки диалогового режима работы (диалоговые операционные системы, диалоговые пакеты программ различного назначения и т. д.). Во многих приложениях диалог уже полностью заменил пакетную обработку, а строчный режим диалога уступает место поэкранному режиму и многооконному графическому способу общения.

Прикладная система состоит из диалогового монитора-набора универсальных программ, обеспечивающих ведение диалога и обмен данными, и базы знаний об области. Информация о структуре, целях и форма диалога задает сценарий, в соответствии с которым монитор управляет ходом диалога. Носителями процедурных знаний о предметной области являются прикладные модули, реализующие функции собственной системы. Таким образом, создание прикладной системы сводится к настройке диалогового монитора на конкретный диалог, путем заполнения базы знаний. При этом программировать в традиционном смысле этого слова приходится лишь прикладные модули, знания о диалоге вводятся в систему с помощью набора соответствующих средств - редактора сценариев. Логично требовать, чтобы редактор сценариев также представлял собой диалоговую программу, отвечавшую рассмотренным выше требованиям. Благодаря готовому универсальному монитору программист может сосредоточиться на решении чисто прикладных задач, выделение же знаний о диалоге в сценарий обеспечивает в значительной степени необходимая гибкость программного продукта.

Большое внимание в настоящее время уделяется проблеме создания "интеллектуальных" ППП. Такой пакет позволяет конечному пользователю лишь сформулировать свою задачу в содержательных терминах, не указывая алгоритма ее решения. Синтез решения и сборка целевой программы производятся автоматически. При этом детали вычислений скрыты от пользователя, и компьютер становится интеллектуальным партнером человека, способным понимать его задачи. Предметное обеспечение подобного ППП представляет собой некоторую базу знаний, содержащую как, процедурные, так и описательные знания. Такой способ решения иногда называют концептуальным программированием, характерными особенностями которого является программирование в терминах предметной области использование ЭВМ уже на этапе постановки задач, автоматический синтез программ решения задачи, накопление знаний о решаемых задачах в базе знаний. В заключение данного раздела рассмотрим еще одну современную тенденцию разработки ППП. Она заключается в применении

специализированных инструментальных средств и систем, позволяющих ускорить и упростить процесс создания пакета, а также снизить стоимость разработки. При этом особое внимание уделяется созданию системных средств, позволяющих использовать в качестве предметного обеспечения ППП написанные ранее прикладные программы. Кроме того, инструментальные системы обычно реализуются таким образом, что их можно использовать в качестве базы (готовых компонентов) для системного обеспечения разрабатываемых пакетов (поэтому их иногда называют базовые инструментальными системами). Создание инструментальных средств, упрощающих разработку ППП в различных предметных областях, представляет собой одно из актуальных направлений системного программирования в пакетной проблематике.

При выборе метода реализации того или иного ППП следует учитывать особенности конкретной ситуации, в частности, имеющиеся в наличии людские и материальные ресурсы. Так пакет библиотечного типа, не являясь развитой системой с точки зрения рассмотренных требований, обладает, однако, тем преимуществом, что входной язык и системное обеспечение такого пакета могут быть достаточно легко реализованы силами прикладного программиста. Поэтому в случае, когда подобный пакет удовлетворяет конкретных пользователей, его разработка является вполне оправданной.

1.12 Лекция № 16, 17 (4 часа).

Тема: «Компьютерные сети»

1.12.1 Вопросы лекции:

1. Основные понятия. Классификация компьютерных сетей
2. Локальные вычислительные сети
3. Глобальная вычислительная сеть
4. Требования, предъявляемые к современным вычислительным сетям

1.12.2 Краткое содержание вопросов:

1. Основные понятия. Классификация компьютерных сетей

По мере того, как в информационные процессы вовлекалось все больше пользователей, как расширялся круг задач, решаемых с помощью ЭВМ, возникла необходимость в децентрализации процессов обработки данных. Принцип **централизованной** обработки данных (рисунок 1), когда к одной большой ЭВМ подключалось несколько терминалов (рабочих мест, состоящих из дисплея и клавиатуры), не отвечал высоким требованиям к надежности процесса обработки, затруднял развитие систем и не мог обеспечить необходимые временные параметры при диалоговой обработке данных в многопользовательском режиме. Кратковременный выход из строя центральной ЭВМ приводил к роковым последствиям для системы в целом. Поэтому приходилось дублировать функции центральной ЭВМ, значительно увеличивая затраты на создание и эксплуатацию систем обработки данных.

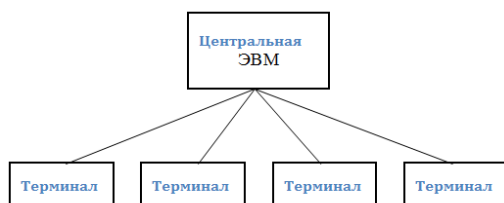


Рисунок 1 – Система централизованной обработки данных

Появление малых ЭВМ, микроЭВМ и, наконец, персональных компьютеров потребовало нового подхода к организации систем обработки данных, к созданию новых информационных технологий. Возникло логически обоснованное требование перехода от использования отдельных ЭВМ в системах централизованной обработки данных к **распределенной (децентрализованной) обработке данных**.

Распределенная обработка данных – обработка данных, выполняемая на независимых, но связанных между собой ЭВМ, представляющих распределенную систему.

Для реализации распределенной обработки данных были созданы **многомашинные ассоциации**, структура которых разрабатывается по одному из следующих направлений:

- многомашинные вычислительные комплексы (МВК);
- компьютерные (вычислительные) сети.

Многомашинный вычислительный комплекс – это группа установленных рядом вычислительных машин, объединенных с помощью специальных средств сопряжения и выполняющих совместно единый информационно-вычислительный процесс.

Многомашинные вычислительные комплексы могут быть:

- **локальными**, при условии установки ЭВМ в одном помещении;
- **дистанционными**, если некоторые ЭВМ комплекса установлены на значительном расстоянии от центральной ЭВМ и для передачи данных используются телефонные каналы связи.

С появлением персональных ЭВМ их также стали объединять в сети.

Компьютерная (вычислительная) сеть – это совокупность компьютеров, соединенных с помощью каналов связи в единую систему, удовлетворяющую требованиям распределенной обработки данных.

Несколько персональных ЭВМ, если они входят в состав вычислительной сети, могут выполнять значительно более широкий круг задач, чем каждая в отдельности. Использование сети позволяет ПЭВМ, подсоединенным к ней, обмениваться данными, организовывать коллективную обработку данных, а также совместно использовать программы, принтеры и прочие аппаратные средства.

В настоящее время **вычислительные (компьютерные) сети** являются высшей формой многомашинных ассоциаций.

Особенностью эксплуатации вычислительных сетей является не только приближение аппаратных средств непосредственно к местам возникновения и использования данных, но и разделение функций обработки и управления на отдельные составляющие с целью их эффективного распределения между несколькими ЭВМ, а также обеспечение надежного и быстрого доступа пользователей к вычислительным и информационным ресурсам и организация коллективного использования этих ресурсов.

В условиях вычислительной сети обеспечивается возможность:

- организовать параллельную обработку данных многими ЭВМ;
- создавать распределенные базы данных, размещаемые в памяти различных ЭВМ;
- специализировать отдельные ЭВМ (группы ЭВМ) для эффективного решения определенных классов задач;
- автоматизировать обмен информацией и программами между отдельными ЭВМ и пользователями сети;
- резервировать вычислительные мощности и средства передачи данных на случай выхода из строя отдельных из них с целью быстрого восстановления нормальной работы сети;
- перераспределять вычислительные мощности между пользователями сети в зависимости от изменения их потребностей и сложности решаемых задач;

– стабилизировать и повышать уровень загрузки ЭВМ и дорогостоящего периферийного оборудования.

Как показывает практика, за счет расширения возможностей обработки данных, лучшей загрузки ресурсов и повышения надежности функционирования системы в целом стоимость обработки данных в вычислительных сетях не менее чем в полтора раза ниже по сравнению с обработкой аналогичных данных на автономных ЭВМ.

Прогресс в развитии микропроцессорной техники сделал ее доступной массовому потребителю, а высокая надежность, относительно низкая стоимость, простота общения с пользователем – непрофессионалом в области вычислительной техники послужили основой для организации систем распределенной обработки данных, базирующихся на ПЭВМ. Такие вычислительные сети могут объединять десятки, сотни, тысячи и более ПЭВМ.

В зависимости от территориального расположения абонентских систем вычислительные сети можно разделить на три основных класса:

- глобальные сети (WAN – Wide Area Network);
- региональные сети (MAN – Metropolitan Area Network);
- локальные сети (LAN – Local Area Network).

Глобальная вычислительная сеть объединяет абонентов, расположенных в различных странах, на различных континентах. Взаимодействие между абонентами такой сети может осуществляться на базе телефонных линий связи, радиосвязи и систем спутниковой связи. Глобальные вычислительные сети позволяют решить проблему объединения информационных ресурсов всего человечества и организации доступа к этим ресурсам. Примером глобальной вычислительной сети служит **всемирная сеть Internet**.

Региональная вычислительная сеть связывает абонентов, расположенных на значительном расстоянии друг от друга. Она может включать абонентов внутри большого города, экономического региона, отдельной страны. Обычно в этом случае расстояние между абонентами вычислительной сети составляет десятки – сотни километров.

Локальная вычислительная сеть объединяет абонентов, расположенных в пределах небольшой территории. В настоящее время не существует четких ограничений на территориальный разброс абонентов локальной вычислительной сети. Обычно такая сеть привязана к конкретному месту. К классу локальных вычислительных сетей относятся сети отдельных предприятий, фирм, банков, офисов и т. д. Протяженность такой сети можно ограничить пределами 2 ... 2,5 км.

2. Локальные вычислительные сети

Назначение всех видов компьютерных ЛВС определяется, в основном, двумя функциями:

обеспечение совместного использования аппаратных и программных ресурсов сети;

обеспечение совместного доступа к ресурсам данных.

Локальные вычислительные сети объединяют относительно небольшое число ПЭВМ (обычно от 10 до 100) в пределах небольшой территории (одно помещение, здание или учреждение). Традиционное название – локальная вычислительная сеть – скорее дань тем временам, когда сети использовались в основном для решения вычислительных задач. Сегодня же в 99% случаев речь идет исключительно об обмене информацией в виде текстов, графических и видео-образов, числовых массивов.

Основными техническими средствами ЛВС являются **рабочие станции** (обычные ПЭВМ), которые могут быть подключены к более мощному компьютеру – **сетевому серверу**.

Сервер – это мощный компьютер, способный одновременно обрабатывать сотни (тысячи) запросов на обслуживание, поступающих от рабочих станций.

Рабочей станцией (или сетевым узлом) называется ПЭВМ, используемая в качестве клиента, то есть потребителя сетевых услуг, предоставляемых сервером.

Все рабочие станции в локальной сети могут использовать специализированные приложения, хранящиеся на сетевом сервере, и работать с общими устройствами: принтерами, факсами и другой периферией. ЛВС позволяют отдельным пользователям легко и быстро взаимодействовать друг с другом. Вот лишь некоторые задачи, которые позволяет решать ЛВС:

- совместная работа с документами;
- упрощение документооборота: руководитель получает возможность просматривать, корректировать и комментировать документы, не покидая своего рабочего места, не организовывая собраний и совещаний, отнимающих много времени;
- сохранение и архивирование созданных документов на сервере, чтобы не использовать ценное пространство на жестком диске ПК;
- простой доступ к приложениям на сервере;
- облегчение совместного использования в организациях дорогостоящих ресурсов, таких как принтеры, накопители CD-ROM, жесткие диски и приложений (например, текстовые процессоры или программное обеспечение баз данных).

Для организации взаимодействия между сервером и рабочими станциями используются специальные **аппаратные средства**, основными из которых являются:

- **адаптер** (сетевой адаптер) – техническое устройство, выполняющее функции сопряжения ПЭВМ с каналами связи. Один адаптер обеспечивает сопряжение ПЭВМ с одним каналом связи. Адаптер часто называют сетевой картой. Он устанавливается непосредственно в один из свободных слотов (ISA, PCI, EISA или другому) материнской платы ПЭВМ и к нему на задней панели системного блока подстыковывается линия связи.

В большинстве из адаптеров предусмотрено использование микросхемы ПЗУ (ROM) удаленной загрузки. Это нужно в том случае, если компьютер рабочей станции не имеет НГМД или НЖМД. Загрузка операционной системы в память таких компьютеров осуществляется через сеть из сервера.

- **концентратор (Hub)** – это устройство множественного доступа, выполняющее роль центральной точки соединения в топологии "физическая звезда". Концентратор является основным узлом сети Ethernet на витой паре. Каждый компьютер должен быть подключен к нему с помощью сегмента кабеля. Длина каждого сегмента не должна превышать 100 м. На концах кабельных сегментов устанавливается разъем RJ-45.

Концентраторы выпускают на разное количество портов: как правило, 8, 12 или 16. К каждому порту может быть подключен либо персональный компьютер, либо другой концентратор.

- **повторитель** – устройство, обеспечивающее сохранение формы и амплитуды сигнала при передаче его на большее, чем предусмотрено данным типом физической передающей среды, расстояние. Под **передающей средой** в данном случае понимается **канал связи**.

Существуют **локальные** и **дистанционные** повторители. Локальные повторители позволяют соединять фрагменты сетей, расположенные на расстоянии до 50 м, а дистанционные – до 2000 м;

- **канал связи** – это физическая среда, посредством которой происходит обмен данными между клиентами ЛВС. Канал связи может быть как проводным, так и беспроводным. Наиболее часто используются проводные линии связи. При этом обычно применяют четыре типа сетевых кабелей:

- незащищенная витая пара;
- защищенная витая пара;
- коаксиальный кабель;
- волоконно-оптический кабель.

Первые три типа кабелей передают электрический сигнал по медным проводникам. Волоконно-оптические кабели передают свет по стеклянному волокну. Большинство сетей допускает несколько вариантов кабельных соединений.

Витые пары представляют собой два провода, скрученных вместе шестью оборотами на дюйм для обеспечения защиты от электромагнитных помех и согласования импеданса или электрического сопротивления. Защищенные витые пары схожи с незащищенными, за исключением того, что они используют более толстые провода и защищены от внешнего воздействия слоем изолятора. Наиболее распространенный тип такого кабеля, применяемого в локальных сетях, представляет собой защищенный кабель с двумя витыми парами непрерывного провода.

Незащищенные витые пары чувствительны к электромагнитным помехам, например электрическим шумам, создаваемым люминесцентными светильниками и движущимися лифтами.

Защита и тщательное соблюдение числа повивов на дюйм делают защищенный кабель с витыми парами надежным альтернативным кабельным соединением. Однако эта надежность приводит к увеличению стоимости.

Коаксиальные кабели состоят из двух проводников, окруженных изолирующими слоями. Первый слой изоляции окружает центральный медный провод. Этот слой оплетен снаружи внешним экранирующим проводником. Наиболее распространенными коаксиальными кабелями являются **толстый** (около 10 мм в диаметре) и **тонкий** (около 4 мм) кабели "Ethernet". Такая конструкция обеспечивает хорошую помехозащищенность и малое затухание сигнала на расстоянии.

Обладая преимуществами по помехозащищенности, прочности, длине линий, толстый коаксиальный кабель дороже и сложнее в монтаже (его сложнее протягивать по кабельным каналам), чем тонкий.

Волоконно-оптические кабели передают данные в виде световых импульсов по стеклянным "проводам". Большинство систем локальных сетей в настоящее время поддерживает волоконно-оптическое кабельное соединение. Волоконно-оптический кабель обладает существенными преимуществами по сравнению с любыми вариантами медного кабеля. Волоконно-оптические кабели обеспечивают наивысшую скорость передачи; они более надежны, так как не подвержены потерям информационных пакетов из-за электромагнитных помех. Оптический кабель очень тонок и гибок, что делает его транспортировку более удобной по сравнению с более тяжелым медным кабелем. Однако наиболее важно то, что только оптический кабель имеет достаточную пропускную способность, которая в будущем потребуется для более быстрых сетей.

Пока еще цена волоконно-оптического кабеля значительно выше медного. По сравнению с медным кабелем монтаж оптического кабеля более трудоемок, поскольку концы его должны быть тщательно отполированы и выровнены для обеспечения надежного соединения. Однако ныне происходит переход на оптоволоконные линии, абсолютно неподверженные помехам и находящиеся вне конкуренции по пропускной способности. Стоимость таких линий неуклонно снижается, технологические трудности стыковки оптических волокон успешно преодолеваются.

В настоящее время наибольшее распространение получили две разновидности ЛВС – **одноранговая сеть** и **сеть типа клиент-сервер**.

В **одноранговой сети** все компьютеры полностью равноправны. Такая сеть не имеет сервера. Некоторые аппаратные средства (например, винчестеры, приводы CD-ROM, сканеры, принтеры и т. д.), подключенные к отдельным ПЭВМ, используются совместно на всех рабочих местах.

Каждый пользователь одноранговой сети может определить права доступа другим пользователям к информации на своем компьютере. Механизмом ограничения прав пользователей является возможность блокировки доступа к дискам и другим периферийным устройствам, подключенным к его компьютеру.

Под сетью типа **клиент-сервер** понимают сеть, в "центре" которой находится мощный персональный компьютер (сетевой сервер), соединенный с отдельными рабочими станциями (клиентами). Клиенты используют ресурсы сервера, поэтому могут быть укомплектованы относительно недорогими внешними устройствами. Управление сетью, в смысле управления отдельными рабочими станциями, а также контроль за периферийными устройствами сети (модемы, факсы и т. д.) осуществляется специальным мощным сетевым программным обеспечением.

ЛВС в зависимости от назначения и технических решений могут иметь различную **топологию** (конфигурацию). Топология ЛВС отражает структуру связей между ее основными функциональными элементами – сервером и рабочими станциями (другими словами, топология ЛВС – это способ соединения компьютеров в сети). Наибольшее распространение получили четыре топологии ЛВС:

- кольцевая;
- звездообразная (радиальная);
- шинная;
- древовидная.

В **кольцевой** ЛВС (рисунок 2, а) информация передается по замкнутому каналу. Каждый абонент непосредственно связан с двумя ближайшими соседями, хотя в принципе способен связаться с любым абонентом сети. В сети с кольцевой топологией данные передаются от одного компьютера к другому как бы по эстафете. Если компьютер получит данные, предназначенные для другого компьютера, он передает их дальше по кольцу. Если данные предназначены для получившего их компьютера, они дальше не передаются.

Достоинством кольцевой топологии является то, что нет ограничений на длину всей сети, то есть имеет значение только расстояние между отдельными компьютерами.

Недостатки:

- время передачи данных увеличивается пропорционально числу соединенных в кольцо компьютеров;
- выход из строя одной рабочей станции может парализовать работу всей сети.

В **звездообразной** (радиальной) ЛВС (рисунок 2, б) в центре находится центральный управляющий компьютер (сервер), последовательно связывающийся с абонентами и связывающий их друг с другом.

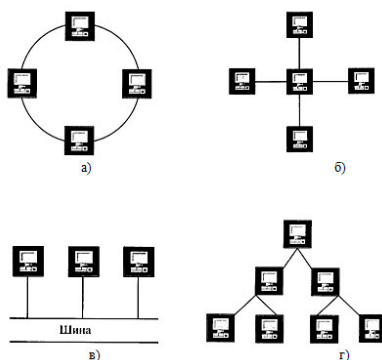


Рисунок 2 – Топологии ЛВС

Сеть с такой топологией имеет следующие достоинства:

- повреждение кабеля является проблемой для одного конкретного компьютера и в целом не сказывается на работе сети;
- надежный механизм защиты от несанкционированного доступа;
- высокая скорость передачи данных от рабочей станции к серверу.

Недостатками сети являются:

- мощность всей сети зависит, в первую очередь, от возможностей сервера;
- невозможна коммутация между отдельными рабочими станциями без сервера.

В **шинной** конфигурации (рисунок 2, в) компьютеры подключены к общему для них каналу (шине), через который могут обмениваться сообщениями. Аналогичным образом к шине подключаются и другие сетевые устройства.

В процессе работы сети информация от компьютера, передающего данные, поступает на все рабочие станции, однако воспринимается только той рабочей станцией, которой она адресована.

Достоинства сети с шинной топологией:

- высокое быстродействие
- относительно небольшие затраты на кабели;
- рабочие станции могут коммутироваться друг с другом без помощи сервера;
- повышенная устойчивость к возможным неисправностям отдельных узлов.

В качестве недостатков можно выделить следующее:

- при обрыве кабеля выходит из строя весь участок сети от места разрыва;
- возможность несанкционированного подключения к сети.

В **древовидной** конфигурации (рисунок 2, г) существует "главный" компьютер, находящийся на вершине иерархической структуры. Ему подчинены компьютеры следующего уровня, и т. д.

Выбор той или иной топологии определяется областью применения ЛВС, географическим расположением ее узлов и размерностью сети в целом.

3. Глобальная вычислительная сеть

Зарождение Интернета принято считать с момента появления первой компьютерной сети, родиной которой в середине 60-х годов двадцатого века стала Америка.

В то время еще не существовало персональных компьютеров, и крупные американские университеты могли себе позволить 1-2 больших компьютера. Компьютерное время было драгоценным ресурсом, и на него заранее записывались. Люди работали ночами, чтобы ни минуты этого времени не пропало даром.

Наконец появилась идея соединить между собой компьютеры разных университетов, чтобы сделать возможным удаленное использование любого свободного в данный момент компьютера. Этот проект получил название ARPANET. К концу 1969 года были соединены компьютеры четырех университетов и появилась первая компьютерная сеть.

Очень скоро обнаружилось, что сеть в основном используется не для вычислений на удаленном компьютере, а для обмена сообщениями между пользователями. В 1972 году, когда ARPANET уже соединял 23 компьютера, была написана первая программа для обмена электронной почтой по сети. Электронную почту оценили по достоинству, что побудило целый ряд государственных организаций и корпораций к созданию собственных компьютерных сетей. Эти сети обладали тем же недостатком, что и ARPANET: они могли соединять только ограниченное число однотипных компьютеров. Кроме того, они были не совместимы друг с другом.

В середине 70-х годов для ARPANET были разработаны новые стандарты передачи данных, которые позволяли объединять между собой сети произвольной архитектуры, тогда же было придумано слово "Интернет". Именно эти стандарты, впоследствии получившие название протокола TCP/IP, заложили основу для роста глобальной компьютерной сети путем объединения уже существующих сетей. Их важным достоинством было то, что сеть считалась в принципе не стопроцентно надежной, и предусматривались средства борьбы с ошибками при передаче данных. В 1983 году сеть ARPANET перешла на новый протокол и разделилась на две независимые сети - военную и образовательную. К этому времени сеть объединяла более тысячи компьютеров, в том

числе в Европе и на Гавайских островах. Последние использовали спутниковые каналы связи.

Развитие Интернета получило новый импульс благодаря инициативе Национального научного фонда США (NSF) по созданию глобальной сетевой инфраструктуры для системы высшего образования (1985-88). NSF создал сеть скоростных магистральных каналов связи и выделял средства на подключение к ней американских университетов, при условии, что университет обеспечивал доступ к сети для всех подготовленных пользователей. Интернет оставался преимущественно университетской сетью до начала 90-х годов, однако NSF сразу взял курс на то, чтобы сделать его в дальнейшем независимым от государственного финансирования. В частности, NSF поощрял университеты к поиску коммерческих клиентов. К 1988 году Интернет уже насчитывал около 56 тысяч соединенных компьютеров.

Настоящий расцвет Интернета начался в 1992 году, когда была изобретена новая служба, получившая странное название «Всемирная паутина» (World Wide Web, или WWW, или просто «веб»). WWW позволял любому пользователю Интернета публиковать свои текстовые и графические материалы в привлекательной форме, связывая их с публикациями других авторов и предоставляя удобную систему навигации. Постепенно Интернет начал выходить за рамки академических институтов и стал превращаться из средства переписки и обмена файлами в гигантское хранилище информации. К 1992 году Интернет насчитывал более миллиона соединенных компьютеров.

В настоящее время Интернет продолжает расти с прежней головокружительной скоростью. По оценке специалистов, количество передаваемой информации (трафик) в Интернете увеличивается на 30% ежемесячно. В 1999 году Интернет объединял около 60 миллионов компьютеров и более 275 миллионов пользователей, и каждый день в нем появлялось полтора миллиона новых веб-страниц. Эти оценки довольно приблизительны, потому что в Интернете нет центрального административного органа, который регистрировал бы новых пользователей и новые компьютеры.

В Россию Интернет впервые проник в начале 90-х годов. Ряд университетов и исследовательских институтов приступили в это время к построению своих компьютерных сетей и обзавелись зарубежными каналами связи. Особенно следует отметить Институт Атомной Энергии им. Курчатова. На базе ИАЭ сложились две крупнейшие коммерческие компании, предоставляющие услуги по подключению к Интернету – «Релком» и «Демос», а также Российский Институт Развития Общественных Сетей (РОСНИИРОС). Последний стал в дальнейшем головной организацией, координирующей развитие российской части Интернета.

По имеющимся оценкам, в 1999 году число российских пользователей Сети превысило 5 миллионов. Сейчас в Интернете есть уже много интересных материалов на русском языке, но знание английского языка желательно - пионерская роль стран английского языка в развитии Интернета закрепила за английским роль языка межнационального общения.

Принципы и организация сети Интернет. Серверы, клиенты и протоколы IP-адресация

Интернет обладает некоторыми чертами почты, некоторыми чертами телеграфа и некоторыми чертами телефона. Так же как в телеграфе, в Интернете используется цифровая передача информации. Как в телефонной сети каждому телефону присваивается телефонный номер, так и каждому компьютеру в Интернете присваивается свой номер, который называется IP-адресом. Только в Интернете, в отличие от телефона, нет путаницы с локальными номерами и междугородними кодами: каждый IP-адрес имеет длину ровно 32 бита и записывается обычно как четыре десятичных числа (от 0 до 255), - например, 62.76.161.102. Это глобальная нумерация - каждый компьютер, подключенный к Интернету, имеет уникальный IP-адрес.

Наиболее глубокая аналогия существует между Интернетом и обычной почтой. В данном случае речь идет о том, что информация по Интернету передается в виде отдельных пакетов. Если нужно передать длинное сообщение, оно разбивается на нужное число кусочков, и каждый из них снабжается адресом отправителя, адресом получателя и некоторой служебной информацией. Каждый пакет передается по Интернету независимо от всех остальных и, в принципе, они могут следовать разными маршрутами. По прибытии пакетов на место из них собирается исходное сообщение. Это называется коммутацией пакетов.

Одним из главных преимуществ режима коммутации пакетов это - эффективное использование общих коммуникационных ресурсов. В Интернете каждый компьютер может одновременно принимать пакеты от большого количества других компьютеров. При этом возможны перегрузки коммутационных узлов (серверов) в результате большого количества информации. Однако все пакеты, пусть с небольшой задержкой, все равно дойдут до адресата в порядке своей очереди. В то же время, если в данный момент вы никакой информации не посылаете, то вы не потребляете никаких ресурсов сети, и тем самым можете находиться «на линии» сколько угодно долго, не создавая помех другим.

Для обеспечения жизнеспособности всей сети необходимо, чтобы обмен информацией между различными ее блоками или отдельными компьютерами велся на основе общепринятых стандартов. Набор формальных правил о том, как и в каком виде следует передавать данные между различными устройствами и программами, называется протоколом. Протокол позволяет корректно взаимодействовать программам, написанным разными авторами для разных типов компьютеров и операционных систем.

Интернет использует протокол TCP/IP. Этот протокол регламентирует, как следует разбивать длинное сообщение на пакеты, как должны быть устроены пакеты, как контролировать прибытие пакетов к месту назначения, что делать в случае ошибок передачи данных, и другие детали.

Собственно, Интернет представляет собой объединение десятков тысяч отдельных сетей, которые используют протокол TCP/IP и единое пространство IP-адресов. В остальном эти сети административно и финансово независимы.

Еще одно важное достоинство коммутации пакетов - это легкость объединения в единую сеть разных по скорости каналов связи. В связи с этим скоростные характеристики подключения к Интернету могут варьироваться в очень широких пределах, и разница будет заключаться лишь в скорости получения информации. Никаких других качественных отличий или ограничений не будет.

Доменная система имен и указатели ресурсов

Хотя люди уже привыкли пользоваться цифровыми номерами абонентов (например, звоня по телефону), все-таки имена, которые можно произнести, легче запоминаются и более удобны для использования. У большинства компьютеров в Интернете есть собственное имя, а не только IP-адрес. Служба, которая обеспечивает перевод имен компьютеров в их IP-адреса, называется Доменной Службой Имен (DNS). Это что-то вроде гигантского, распределенного по многим компьютерам телефонного справочника, с IP-адресами вместо телефонов.

Имя компьютера записывается как несколько слов, разделенных точками, например: iomas.vsau.ru. Это отражает иерархическую, или доменную, структуру службы DNS. В нашем примере “iomas” - это имя компьютера в домене (второго уровня) “vsau”, который принадлежит домену (первого уровня) “ru”. Администратор, который отвечает за домен первого уровня “ru” (Россия), зарегистрировал домен второго уровня “vsau.ru” (Воронежский агроуниверситет) и передал туда все полномочия на регистрацию новых имен в пределах этого домена. В свою очередь администратор домена “vsau.ru” зарегистрировал имя “iomas.vsau.ru” за определенным IP-адресом. Такая структура службы DNS обеспечивает, с одной стороны, уникальность имен компьютеров в пределах

всего Интернета, а с другой стороны, четкое разделение административной ответственности.

DNS - это особая служба Интернета, потому что она используется всеми остальными службами, от telnet до www. Перевод имен DNS в IP-адреса происходит автоматически. Для этого надо только указать вашему компьютеру IP-адрес сервера DNS - того компьютера, которому будут направляться соответствующие запросы.

Хотя не существует особых правил, как следует называть домены, в применении к доменам первого, самого верхнего уровня сложилась определенная практика. Международные организации и США используют домены первого уровня com - для коммерческих, org и net - для некоммерческих организаций. В большинстве стран существует один домен первого уровня для страны: ru - для России, de - для Германии, uk - для Великобритании и т.д.

Так же, как каждый компьютер имеет свое уникальное имя, уникальное имя имеет и каждый документ в Интернете. Это уникальное имя называется URL - Универсальный Указатель Ресурса (Universal Resource Locator). URL имеет следующую форму:

служба://имя_компьютера/директория/поддиректория/.../имя_файла (например: <http://iomas.vsau.ru/people/peopl3.htm>).

Служба обозначается соответствующим протоколом, чаще всего вы встретите http:// для веб-страниц и ftp:// для файловых архивов. Обратите внимание, что используется "прямая" косая черта - "/", а не "обратная" - "\". Также нужно иметь ввиду, что в названиях директорий, поддиректорий и файлов большие и малые буквы различаются.

Серверы, клиенты и протоколы

Известно, что развитие общества связано с разделением труда. Разделение труда выгодно использовать и между людьми, и между компьютерами. Это очевидно: всегда лучше, когда каждый занимается своим делом.

В Интернете есть два «сорта» компьютеров - серверы и клиенты. Серверы - это серьезные, надежные машины. Они работают 24 часа в сутки, 7 дней в неделю. Они постоянно соединены с Интернетом и готовы предоставлять сервис - доставлять документы или пересылать почту, отвечая при этом на десятки и сотни запросов одновременно. От их работоспособности зависит работа того множества компьютеров и локальных сетей, которые подключены к ним. Поэтому серверы и сопутствующая узловая коммуникационная аппаратура защищены от сбоев электропитания источниками бесперебойного питания (UPS). Чаще всего они работают под управлением того или иного варианта операционной системы UNIX. Серверы также называют хостами или сайтами, особенно применительно к веб-сайтам.

Клиенты - это те персональные компьютеры, за которыми сидят пользователи, то есть мы с вами. Сейчас это чаще всего компьютеры под управлением операционной системы Microsoft Windows различных версий. Здесь нет таких требований к надежности - в случае сбоя на вашем компьютере никто кроме вас от этого не пострадает. Зато взаимодействие клиентского компьютера с пользователем отлажено и стандартизовано. Часто такой компьютер не соединен с Интернетом постоянно, а подключается к нему по мере необходимости.

Та же самая терминология относится и к программам - существует клиентское программное обеспечение, которое взаимодействует с пользователем и формирует запрос, и серверное программное обеспечение, которое отвечает на такие запросы. Эти запросы формируются в соответствии с определенным протоколом.

Виды сервиса глобальной сети Интернет и программное обеспечение

Способы использования Интернета определяются предоставляемыми посетителям сервисами, в число которых входят традиционные, специальные и новые виды сервиса.

К традиционным видам сервиса относятся:

- электронная почта (e-mail);

- телеконференции (news);
- живое общение (chat);
- использование вычислительных и информационных ресурсов компьютера в режиме удаленного терминала (telnet);
- служба хранения, поиска и пересылки файлов - FTP (File Transfer Protocol);
- WWW (World Wide Web) - всемирная паутина (в ее составе в июле 1993г. было 130 хост-компьютеров, а в июне 1996г. 230000).

К специальным сервисным средствам относятся:

- информационная система Gopher;
- информационная система WAIS, ведущая поиск по ключевым словам,
- поисковые системы WWW;
- библиографические (диспетчерские) системы;
- Archie - поисковая система FTP;
- и др.

Новые виды сервиса включают в себя:

- системы для заказа товаров по сети (цветы, пицца, авиабилеты, номер в гостинице и т.д.);
- библиотечные системы,
- электронные издания газет и журналов, блоги;
- сервисные представительства фирм,
- коммерческие информационные системы по производителям товаров и услуг, котировкам акций на фондовых биржах;
- и др.

Всех работающих в Интернете можно разделить на две группы: тех, кто предоставляет различные интернет-услуги (провайдеров Интернета), и тех, кто эти услуги использует (пользователей Интернета).

Обучение пользованию Интернетом в основном сводится к общему ознакомлению с устройством Интернета, с предоставляемыми сервисами, адресацией ресурсов, с поиском информации в сервисе WWW и основами работы с электронной почтой.

Провайдеры Интернета по составу предоставляемых услуг делятся на три группы:

- провайдеры, предоставляющие доступ в Интернет - ISP (Internet Service Providers). Примером могут служить www.rol.ru, www.mtu.ru, и др.;
- провайдеры, предоставляющие услуги присутствия в Интернет - IPP (Internet Presence Providers), примером которых могут служить www.mail.ru, www.narod.ru, и др. К предоставляемым ими услугам относятся электронный почтовый ящик и услуги почты, размещение сайта клиента на ЭВМ провайдера, различные поисковые системы, новости, и т.д.;
- издатели собственных материалов - PCP (Private Content Publisher), которые содержат (т.е. разрабатывают и эксплуатируют) свои информационные системы (например, www.garant.ru, www.rbc.ru и т.п.), электронные магазины, рекламные агентства, туристические фирмы, и т.д.

ISP (Internet Service Provider)

ISP - это поставщик услуг Internet, т.е. организации или частные лица, предоставляющие доступ в Internet (hosting). Источником доходов ISP являются владельцы локальных ЭВМ, которым предоставляется доступ к данной глобальной вычислительной сети. Через имеющиеся шлюзы локальные пользователи могут получить доступ к другим глобальным сетям и таким образом получить возможность работать в Internet.

ISP подключены к Internet постоянно и имеют постоянный IP-адрес (IP-адрес является частью URL). Остальные пользователи (клиенты) могут подключаться к ISP лишь на время работы. IP-адрес присваивается им ISP каждый раз при подключении, а

при отключении - отбирается и может быть отдан кому-нибудь другому. ISP часто предоставляет своим клиентам удаленный доступ по коммутируемым каналам телефонной связи (это называется "dual-up service"). Для этого ISP арендует у местной телефонной компании телефонные номера, по которым с ним можно связаться.

ISP иногда может предоставлять (делегировать) функции хостинга локальным ЭВМ (которые при этом получают свой постоянный IP-адрес), например, стоящим дома или на работе, и превращать таким образом локальную ЭВМ в ISP более низкого уровня. В свою очередь, локальная хост-ЭВМ может делегировать такие права другой локальной ЭВМ, которая становится хост-провайдером (т.е. ISP) еще более низкого уровня (если она, конечно, имеет доступ к каналам связи - например, через учрежденческую АТС). Образуется цепочка провайдеров, различающихся своими IP-адресами: mesi.ru; ex.mesi.ru; stud.ex.mesi.ru и т.д.

Широко известными интернет-сервис-провайдерами являются, например, mtu.ru и rol.ru.

IPP (Internet Presence Provider)

IPP - это провайдер, обеспечивающий своим клиентам присутствие в Internet. Он так же подключен к Internet постоянно и имеет постоянный IP-адрес. В отличие от ISP, он не предоставляет услуг типа dual-up service. Он может только размещать на своих серверах публикации других лиц, рекламу, веб-сайты, организовывать работу электронной почты и т.д.

IPP после регистрации на их сайте предоставляют имя, которое будет являться наименованием вашего почтового ящика (e-mail), и кроме того - именем вашего сайта, который размещается на ЭВМ провайдера.

Например, после регистрации на mail.ru предоставляется почтовый ящик имя@mail.ru. Адрес вашего сайта будет: <http://www.mail.ru/~имя>.

Такое имя сайта не удовлетворяет многих пользователей. У провайдера www.narod.ru имя сайта выглядит по-другому: <http://www.имя.narod.ru>. Получается, что ваш сайт имеет имя в домене III уровня, а не где-то среди каталогов IPP. Это выглядит солиднее, похоже на то, что у вас есть свой компьютер с постоянным IP-адресом.

Широко известными провайдерами присутствия в Интернете являются:

- <http://www.mail.ru>;
- <http://www.yandex.ru>;
- <http://www.narod.ru>;
- <http://www.rambler.ru> и др.

PCP (Private Content Publisher)

PCP - это издатель собственных материалов. Он является участником межсетевого обмена (провайдером услуг), который готовит информацию для размещения в Internet, размещает ее, как правило, на своих компьютерах и постоянно обновляет ее. На определенных условиях он разрешает пользоваться своими материалами клиентам, приходящим из Internet.

Он может содержать базу телефонов города, справочники различного назначения. Такой провайдер является квалифицированным источником информации. Обычно его базы данных относятся к области, в которой он является квалифицированным специалистом.

Этот тип провайдера очень бережно относится к своей информации, к исправности своего компьютера, и у него всегда можно получить самые достоверные, постоянно обновляемые данные по выбранной им специальности.

Пример этого типа провайдера - <http://www.garant.ru> (юридические документы).

В отличие от ISP, он не предоставляет услуг dual-up service. Подготовка провайдеров Интернета предусматривает изучение таких дисциплин, как:

- общий курс по работе в сети Интернет;
- техническое обеспечение глобальных вычислительных сетей;

- программирование на специальных алгоритмических языках;
- операционные системы для глобальных вычислительных сетей;
- программное обеспечение глобальных вычислительных сетей;
- интернет-протоколы;
- инструментальные средства для исследования глобальных вычислительных сетей;

- WWW-программирование на стороне клиента;
- WWW-программирование на стороне сервера;
- разработка веб-сайтов;
- веб-дизайн;
- администрирование узлов глобальных вычислительных сетей.

Характеристики хостинг-провайдеров

Размещение сайтов на чужой технической базе называется хостингом. Хостинг бывает платный и бесплатный.

При платном хостинге оговаривается состав услуг, предоставляемых провайдером:

- провайдер предоставляет все: канал, ЭВМ, URL-адрес, обслуживание сайта и т.д.;
- провайдер предоставляет площадь для размещения вашей ЭВМ, канал, URL, обслуживание, и т.д.;
- и т.д.

При бесплатном хостинге провайдер предоставляет URL, место на своих магнитных носителях, сервисные программы для создания и обслуживания сайта. Но взамен размещает свою баннерную рекламу на вашем сайте.

Хостинг-провайдеры характеризуются:

- техническими ресурсами, к которым относятся ЭВМ, каналы связи, маршрутизаторы, которые определяют такую характеристику, как скорость отклика.

Скорость отклика сайта - это характеристика провайдера, которая связана с загрузкой его каналов. Например, провайдер имеет выход в Интернет, соединяясь по радиоканалу с другим провайдером, имеющим спутниковую связь. Канал может быть перегружен, и для связи с Интернетом через такого провайдера может понадобиться большое время. Большое значение также имеет связь локальных ЭВМ с провайдером (телефонный канал, количество телефонов для подключения локальных ЭВМ и др.);

- предоставляемыми услугами, которые могут быть весьма специфичными. При их выборе необходимо понимать, что услуга вам дает и что от вас требуется для ее использования.

Например, предоставление вам ЭВМ провайдера - и предоставление вам площадки для установки вашей ЭВМ различаются тем, что если вы устанавливаете у провайдера свою ЭВМ, то ее никто из сотрудников провайдера не имеет права трогать (но в договоре можно оговорить, что ремонт отказавшей ЭВМ проводится сотрудниками провайдера) - это ваша ЭВМ, тогда как если вам предоставлена ЭВМ провайдера, то ее могут заменить на другую без согласования с вами;

- программными ресурсами провайдера: какая операционная система используется (80% провайдеров работает под Unix), какое программное обеспечение используется для сервиса WWW (чаще всего - сервер Apache), какие виды сервиса разрешены и обеспечиваются провайдером (например, часто запрещаются ftp и telnet).

Программные ресурсы провайдера характеризуются также программным обеспечением, предоставляемым клиентам (компиляторы, интерпретаторы, СУБД, предустановленные скрипты, управляющие интерфейсы (мастера и шаблоны), и т.д.). Этот состав очень важен для размещения созданного сайта у хостинг-провайдера, так как провайдер может плохо относиться к продукции фирмы и не поддерживать конструкции

FrontPage и другие расширения операционной системы Windows, а для клиента это программное обеспечение является основным;

- системой безопасности провайдера: для пользователя имеет значение наличие зеркальных серверов, резервных мощностей.

Зеркальный сервер - это дублирующий сервер, содержащий ту же самую информацию, что и основной. Такие зеркала нужны для увеличения надежности системы и ее пропускной способности. Зеркальные серверы могут устанавливаться в различных частях света для того, чтобы не загружать глобальные вычислительные сети. Они имеют одно и то же имя, но службы DNS отправляют посетителей на ближайший из них.

Резервная мощность - это запасные технические средства, которые находятся в резерве (холодном, теплом, горячем). При отказе какой-либо ЭВМ провайдер переключает работу на резервную ЭВМ;

- службой технической поддержки, обеспечивающей помощь клиентам и снабжающей клиентов технической документацией по предоставляемым сервисам.

Эта служба ведет разработку техдокументации для клиентов, проводит мониторинг сайта и оценку его эффективности, резервное копирование сайта (backup) и др.;

- сопутствующими услугами, предоставляемыми хостинг-провайдером, такими как регистрация доменных имен, бонусами (т.е. скидками или дополнительными услугами, предоставляемыми некоторым клиентам), возможным освобождением вашего сайта от размещения чужих баннеров, возможностью подключения до-полнительных услуг, таких как сервис Spylog, и др.; условиями и ограничениями: ограничениями трафика (установлением разной платы за работу в Рунете и в Интернете); установкой лимита на скачивание информации с локальной ЭВМ (например, 1 Гбайт в месяц) - и взиманием дополнительной платы за превышение трафика; ограничением объема предоставляемой памяти; запретом на хранение информации, не связанной гиперссылками с основным содержанием сайта; с условиями типа: "мы не несем ответственности за утерю информации клиента", - и т.д.

Программное обеспечение Интернета

Каждый сервис требует своего программного обеспечения (в общем виде структура программного обеспечения Internet приведена на рис. 11).

Интернет построен на основе архитектуры "клиент-сервер". В сетях этого типа выделяется мощный хост-компьютер (или даже несколько хост-компьютеров), на который ставится серверное программное обеспечение.

На клиентских ЭВМ устанавливается клиентское программное обеспечение.

Хост-ЭВМ постоянно включены, имеют постоянные IP-адреса. Клиентские ЭВМ включаются по мере необходимости, связываются с серверным программным обеспечением хост-ЭВМ, получают от него временный IP-адрес, действующий только в пределах данного сеанса связи.

Глобальные вычислительные сети имеют узлы (хосты), на которых устанавливается серверная часть программного обеспечения сервисов Интернета. Серверное и клиентское программное обеспечение взаимодействуют между собой.

Из рис.1 видно, что программное обеспечение Интернета состоит из трех видов программ: серверное ПО, клиентское ПО и ПО систем безопасности. Серверное ПО устанавливается на хост-компьютерах, клиентское - на локальных ЭВМ (т. е. на ЭВМ клиентов). ПО систем безопасности может не соответствовать архитектуре "клиент-сервер".

Системы безопасности либо устанавливаются на клиентской ЭВМ или только на хост-ЭВМ, либо для них выделяется отдельная ЭВМ, на которой устанавливаются специализированные программы, обеспечивающие безопасность, - такие программы называются "брандмауэрами", или

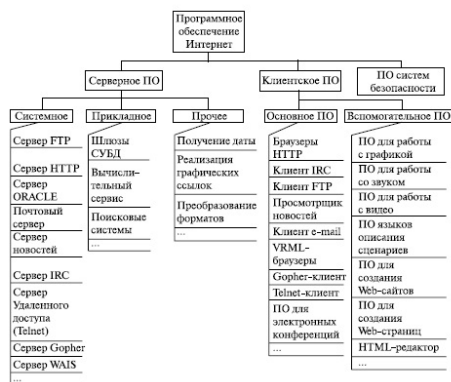


Рис. 11. Структура программного обеспечения Internet (общий вид)

"firewall" (эти названия заимствованы у пожарных, которые требуют, чтобы при строительстве длинных домов производилось их разделение на части и одна часть от другой отделялась каменной стеной, основное назначение которой - не допустить распространения огня на всю постройку при возгорании одной из ее частей). Программы безопасности являются аналогами такой стены между ЭВМ пользователей и Интернетом. Они могут быть настроены так, чтобы полностью разрывать связь между ЭВМ и Интернетом, могут допускать одностороннюю связь (например, разрешена только отправка почты с локальной ЭВМ) или разрешать двустороннюю связь только для определенных видов работ. Кроме того, может быть запрещено выполнение каких-либо действий.

Входная и выходная информация в таких программах проходит через фильтры, которые, например, могут быть настроены на выявление вирусов, на пропуск файлов, не превышающих заданных размеров, или файлов определенного типа, на запрет связи с определенными IP-адресами и т.д.

В клиентском программном обеспечении необходимо обратить внимание на VRML-браузеры, позволяющие просматривать виртуальные миры, работать в трех измерениях (в двух измерениях по экрану можно перемещаться вверх-вниз и вправо-влево; в трех измерениях, кроме этого, добавляется возможность приближаться к экрану или удаляться от него). VRML-браузеры - это клиентское программное обеспечение. В Интернете есть несколько сайтов с серверами VRML и своеобразными виртуальными мирами.

Как на клиентских ЭВМ, так и на хостах могут быть размещены программы, расширяющие возможности серверов и клиентов. Для написания таких программ применяются специальные алгоритмические языки: HTML, PHP, система программирования CGI, Java, Java-script, Perl, SSI и др.

При использовании таких программ на сервере должно быть установлено соответствующее программное обеспечение.

Программирование глобальных вычислительных сетей - сложное направление, в котором применяются приведенные выше алгоритмические языки на основе правил работы, определяемых протоколами TCP/IP и соответствующих сервисов Интернет

4. Требования, предъявляемые к современным вычислительным сетям

Производительность

Определяется такими показателями: время реакции системы - время между моментом возникновения запроса и моментом получения ответа. Пропускная способность сети определяется количеством информации, переданной через сеть или ее сегмент в единицу времени. Определяется в битах в секунду.

Надежность

Определяется надежностью работы всех ее компонентов. Для повышения надежности работы аппаратных компонентов обычно используют дублирование, когда при отказе одного из элементов функционирование сети обеспечат другие.

При работе компьютерной сети должна обеспечиваться *сохранность* информации и защита ее от искажений. Как правило, важная информация в сети хранится в нескольких экземплярах. В этом случае необходимо обеспечить согласованность данных (например, идентичность копий при изменении информации).

Одной из функций компьютерной сети является передача информации, во время которой возможны ее потери и искажения. Для оценки надежности исполнения этой функции используются показатели вероятности потери пакета при его передаче, либо вероятности доставки пакета (передача осуществляется порциями, которые называются пакетами).

В современных компьютерных сетях важное значение имеет другая сторона надежности - *безопасность*. Это способность сети обеспечить защиту информации от несанкционированного доступа. Задачи обеспечения безопасности решаются применением как специального программного обеспечения, так и соответствующих аппаратных средств.

Управляемость

При работе компьютерной сети, которая объединяет отдельные компьютеры в единое целое, необходимы средства не только для наблюдения за работой сети, сбора разнообразной информации о функционировании сети, но и средства управления сетью. В общем случае система управления сетью должна предоставлять возможность воздействовать на работу любого элемента сети. Должна быть обеспечена возможность осуществлять мероприятия по управлению с любого элемента сети. Управлением сетью занимается администратор сети или пользователь, которому поручены эти функции. Обычный пользователь, как правило, не имеет административных прав.

Другими характеристиками управляемости являются возможность определения проблем в работе компьютерной сети или отдельных ее сегментов, выработка управленческих действий для решения выявленных проблем и возможность автоматизации этих процессов при решении похожих проблем в будущем.

Расширяемость и масштабируемость

Любая компьютерная сеть является развивающимся объектом, и не только в плане модернизации ее элементов, но и в плане ее физического расширения, добавления новых элементов сети (пользователей, компьютеров, служб). Существование таких возможностей, трудоемкость их осуществления входят в понятие расширяемости. Другой похожей характеристикой является масштабируемость сети, которая определяет возможность расширения сети без существенного снижения ее производительности. Обычно одноранговые сети обладают хорошей расширяемостью, но плохой масштабируемостью. В таких сетях легко добавить новый компьютер, используя дополнительный кабель и сетевой адаптер, но существуют ограничения на количество подключаемых компьютеров в связи с существенным падением производительности сети. В многосегментных сетях используются специальные коммуникационные устройства, которые позволяют подключать к сети значительное количество дополнительных компьютеров без снижения общей производительности сети.

Прозрачность

Прозрачность компьютерной сети является ее характеристикой с точки зрения пользователя. Эта важная характеристика должна оцениваться с разных сторон.

Прозрачность сети предполагает скрытие (невидимость) особенностей сети от конечного пользователя. Пользователь обращается к ресурсам сети как к обычным локальным ресурсам компьютера, на котором он работает.

Компьютерная сеть объединяет компьютеры разных типов с разными операционными системами. Пользователю, у которого установлена, например, Windows,

прозрачная сеть должна обеспечивать доступ к необходимым ему при работе ресурсам компьютеров, на которых установлена, например, UNIX. Другой важной стороной прозрачности сети является возможность распараллеливания работы между разными элементами сети. Вопросы назначения отдельных параллельных заданий отдельным устройствам сети также должны быть скрытыми от пользователя и решаться в автоматическом режиме.

Интегрируемость

Интегрируемость означает возможность подключения к вычислительной сети разнообразного и разнотипного оборудования, программного обеспечения от разных производителей. Если такая неоднородная вычислительная сеть успешно выполняет свои функции, то можно говорить о том, что она обладает хорошей интегрируемостью.

Современная компьютерная сеть имеет дело с разнообразной информацией, процесс передачи которой сильно зависит от типа информации. Передача традиционных компьютерных данных характеризуется неравномерной интенсивностью. При этом нет жестких требований к синхронности передачи. При передаче мультимедийных данных качество передаваемой информации в существенной степени зависит от синхронизации передачи. Сосуществование двух типов данных с противоположными требованиями к процессу передачи является сложной задачей, решение которой является необходимым условием вычислительной сети с хорошей интегрируемостью.

Основным направлением развития интегрируемости вычислительных сетей является стандартизация сетей, их элементов и компонентов. Среди стандартов различных видов можно выделить стандарты отдельных фирм, стандарты специальных комитетов, создаваемых несколькими фирмами, стандарты национальных организаций по стандартизации, международные стандарты.

1.13 Лекция № 18 (2 часа).

Тема: «Защита информации»

1.13.1 Вопросы лекции:

1. Угрозы безопасности информации и их классификация
2. Принципы создания базовой системы защиты информации
3. Методы и средства защиты информации

1.13.2 Краткое содержание вопросов:

1. Угрозы безопасности информации и их классификация

Наряду с интенсивным развитием вычислительных средств и систем передачи информации все более актуальной становится проблема обеспечения ее безопасности. Меры безопасности направлены на предотвращение несанкционированного получения информации, физического уничтожения или модификации защищаемой информации.

Под *угрозой безопасности информации* понимается *действие или событие, которое может привести к разрушению, искажению или несанкционированному использованию информационных ресурсов, включая хранимую, передаваемую и обрабатываемую информацию, а также программные и аппаратные средства.*

Угрозы принято делить на случайные, или непреднамеренные, и умышленные. Источником первых могут быть ошибки в программном обеспечении, выходы из строя аппаратных средств, неправильные действия пользователей или администрации и т.д.

Умышленные угрозы, в отличие от случайных, преследуют цель нанесения ущерба пользователям АИТ и, в свою очередь, подразделяются на активные и пассивные.

Пассивные угрозы, как правило, направлены на несанкционированное использование информационных ресурсов, не оказывая при этом влияния на ее функционирование. Пассивной угрозой является, например, попытка получения информации, циркулирующей в каналах, посредством их прослушивания.

Активные угрозы имеют целью нарушение нормального процесса функционирования посредством целенаправленного воздействия на аппаратные, программные и информационные ресурсы. К активным угрозам относятся, например, разрушение или радиоэлектронное подавление линий связи, вывод из строя ПЭВМ или ее операционной системы, искажение сведений в базах данных или в системной информации в компьютерных технологиях и т.д. Источниками активных угроз могут быть непосредственные действия злоумышленников, программные вирусы и т.п.

К основным угрозам безопасности информации относят:

1) *Раскрытие конфиденциальной информации* – несанкционированный доступ к базам данных, прослушивание каналов и т.п. В любом случае получение информации, являющейся достоянием некоторого лица (группы лиц) другими лицами, наносит ее владельцам существенный ущерб.

2) *Компрометация информации* реализуется посредством внесения несанкционированных изменений в базы данных, в результате чего ее потребитель вынужден либо отказаться от нее, либо предпринимать дополнительные усилия для выявления изменений и восстановления истинных сведений. В случае использования скомпрометированной информации потребитель подвергается опасности принятия неверных решений со всеми вытекающими отсюда последствиями.

3) *Несанкционированное использование информационных ресурсов*, с одной стороны является средством раскрытия или компрометации информации, а с другой – имеет самостоятельное значение, поскольку, даже не касаясь пользовательской или системной информации, может нанести определенный ущерб абонентам и администрации. Этот ущерб может варьироваться в весьма широких пределах – от сокращения поступления финансовых средств до полного выхода АИТ из строя.

4) *Ошибочное использование информационных ресурсов* может привести к разрушению, раскрытию или компрометации указанных ресурсов. Данная угроза чаще всего является следствием ошибок, имеющих в программном обеспечении АИТ.

5) *Несанкционированный обмен информацией* может привести к получению одним из них сведений, доступ к которым ему запрещен, что по своим последствиям равносильно раскрытию содержания банковской информации.

6) *Отказ от информации* состоит в непризнании получателем или отправителем этой информации фактов ее получения или отправки. В условиях банковской деятельности это, в частности, позволяет одной из сторон расторгать заключенные финансовые соглашения «техническим» путем, формально не отказываясь от них и нанося тем самым второй стороне значительный ущерб.

7) *Отказ в обслуживании* представляет собой весьма существенную и распространенную угрозу, источником которой является сама АИТ. Подобный отказ особенно опасен в ситуациях, когда задержка с предоставлением ресурсов абоненту может привести к тяжелым для него последствиям. Так, отсутствие у пользователя данных, необходимых для принятия решения, в течении периода времени, когда это решение еще возможно эффективно реализовать, может стать причиной его нерациональных или даже антимонопольных действий.

Наиболее распространенными путями несанкционированного доступа к информации, сформулированными на основе анализа зарубежной печати, являются:

- перехват электронных излучений;
- принудительное электромагнитное облучение (подсветка) линий связи;

- применение подслушивающих устройств;
- дистанционное фотографирование;
- перехват акустических излучений и восстановление текста принтера;
- хищение носителей информации и документальных отходов;
- чтение остаточной информации в памяти системы после выполнения санкционированных запросов;
- копирование носителей информации с преодолением мер защиты;
- маскировка под зарегистрированного пользователя;
- мистификация (маскировка под запросы системы);
- использование программных ловушек;
- использование недостатков языков программирования и операционных систем;
- включение в библиотеки программ специальных блоков типа «Троянский конь»;
- незаконное подключение к аппаратуре и линиям связи;
- злоумышленный вывод из строя механизмов защиты;
- внедрение и использование компьютерных вирусов.

2. Принципы создания базовой системы защиты информации

При разработке АИТ возникает проблема по решению вопросов безопасности информации, составляющей коммерческую тайну, а также безопасности самих компьютерных информационных систем.

Современные АИТ обладают следующими основными признаками:

- наличием информации различной степени конфиденциальности;
- необходимостью криптографической защиты информации различной степени конфиденциальности при передаче данных;
- иерархичностью полномочий субъектов доступа и программ к АРМ, файл-серверам, каналам связи и информации системы;
- организацией обработки информации в диалоговом режиме, в режиме разделения времени между пользователями и в режиме реального времени;
- обязательным управлением потоками информации как в локальных сетях, так и при передаче по каналам связи на далекие расстояния;
- необходимостью регистрации и учета попыток несанкционированного доступа, событий в системе и документов, выводимых на печать;
- обязательным обеспечением целостности программного обеспечения и информации в АИТ;
- наличием средств восстановления системы защиты информации;
- обязательным учетом магнитных носителей;
- наличием физической охраны средств вычислительной техники и магнитных носителей.

Организационные мероприятия и процедуры, используемые для решения проблемы безопасности информации, решаются на всех этапах проектирования и в процессе эксплуатации АИТ.

Существенное значение при проектировании придается предпроектному обследованию объекта. На этой стадии выполняются следующие действия:

- устанавливается наличие секретной (конфиденциальной) информации в разрабатываемой АИТ и оценивается уровень конфиденциальности;
- определяются режимы обработки информации, состав комплекса технических средств, общесистемные программные средства и т.д.;

- анализируется возможность использования имеющихся на рынке сертифицированных средств защиты информации;
- определяется степень участия персонала, функциональных служб, специалистов и вспомогательных работников объекта автоматизации в обработке информации, характер взаимодействия между собой и со службой безопасности;
- определяются мероприятия по обеспечению режима секретности на стадии обработки.

Создание базовой системы защиты информации в АИТ основывается на следующих принципах:

- *Комплексный подход* к построению системы защиты при ведущей роли организационных мероприятий, означающий оптимальное сочетание программных аппаратных средств и организационных мер защиты и подтвержденный практикой создания отечественных и зарубежных систем защиты.
- *Разделение и минимизация полномочий* по доступу к обрабатываемой информации и процедурам обработки, т.е. предоставление пользователям минимума строго определенных полномочий, достаточных для успешного выполнения ими своих служебных обязанностей, с точки зрения автоматизированной обработки доступной им конфиденциальной информации.
- *Полнота контроля и регистрации попыток* несанкционированного доступа, т.е. необходимость точного установления идентичности каждого пользователя и протоколирования его действий для проведения возможного расследования, а также невозможность совершения любой операции обработки информации в АИТ без ее предварительной регистрации.
- *Обеспечение надежности системы защиты*, т.е. невозможность снижения уровня надежности при возникновении в системе сбоев, отказов, преднамеренных действий нарушителя или непреднамеренных ошибок пользователей и обслуживающего персонала.
- *Обеспечение контроля за функционированием системы защиты*, т.е. создание средств и методов контроля работоспособности механизмов защиты.
- *«Прозрачность» системы защиты* информации для общего и прикладного программного обеспечения и пользователей АИТ.
- *Экономическая целесообразность* использования системы защиты, т.е. стоимость разработки и эксплуатации систем защиты информации должна быть меньше стоимости возможного ущерба, наносимого объекту в случае разработки и эксплуатации АИТ без системы защиты информации.

3. Методы и средства защиты информации

Методы и средства обеспечения безопасности информации (на примере банковской системы)

МЕТОДЫ



Препятствие – метод физического преграждения пути злоумышленнику к защищаемой информации (к аппаратуре, носителям информации и т.д.).

Управление доступом – метод защиты информации регулированием использования всех ресурсов компьютерной информационной системы банковской деятельности (элементов баз данных, программных и технических средств). Управление доступом включает следующие функции защиты:

- идентификацию пользователей, персонала и ресурсов системы (присвоение каждому объекту персонального идентификатора);
- опознание объекта или субъекта по предъявленному им идентификатору;
- проверку полномочий (проверка соответствия дня недели, времени суток, запрашиваемых ресурсов и процедур установленному регламенту);
- разрешение и создание условий работы в пределах установленного регламента;
- регистрацию (протоколирование) обращений к защищаемым ресурсам;
- реагирование (сигнализация, отключение, задержка работ, отказ в запросе) при попытках несанкционированных действий.

Маскировка – метод защиты информации путем ее криптографического закрытия. Этот метод защиты широко применяется за рубежом как при обработке, так и при хранении информации, в том числе на дискетах. При передаче информации по каналам связи большой протяженности этот метод является единственно надежным.

Регламентация – метод защиты информации, создающий такие условия автоматизированной обработки, хранения и передачи защищаемой информации, при которых возможности несанкционированного доступа к ней сводились бы к минимуму.

Принуждение – такой метод защиты, при котором пользователи и персонал системы вынуждены соблюдать правила обработки, передачи и использования защищаемой информации под угрозой материальной, административной или уголовной ответственности.

Побуждение – такой метод защиты, который побуждает пользователя и персонал системы не разрушать установленные порядки за счет сложившихся моральных и этических норм (как регламентированных, так и неписаных).

Средства, с помощью которых реализуются рассмотренные методы:

Технические средства реализуются в виде электрических, электромеханических и электронных устройств. Вся совокупность технических средств делится на аппаратные и физические. Под аппаратными техническими средствами принято понимать устройства, встраиваемые непосредственно в вычислительную технику или устройства, которые сопрягаются с подобной аппаратурой по стандартному интерфейсу.

Физические средства реализуются в виде автономных устройств и систем. Например, замки на дверях, где размещена аппаратура, решетки на окнах, электронно-механическое оборудование охранной сигнализации.

Программные средства представляют из себя программное обеспечение, специально предназначенное для выполнения функций защиты информации.

Организационные средства защиты представляют собой организационно-технические и организационно-правовые мероприятия, осуществляемые в процессе создания и эксплуатации вычислительной техники и аппаратуры телекоммуникаций для обеспечения защиты информации.

Морально-этические средства защиты реализуются в виде всевозможных норм, которые сложились традиционно или складываются по мере распространения вычислительной техники и средств связи в обществе.

Законодательные средства защиты определяются законодательными актами страны, которыми регламентируются правила пользования, обработки и передачи информации ограниченного доступа и устанавливаются меры ответственности за нарушение этих правил.

Все перечисленные средства защиты разделены на *формальные* (выполняющие защитные функции строго по заранее предусмотренной процедуре без непосредственного участия человека) и *неформальные* (определяющиеся целенаправленной деятельностью человека либо регламентирующие эту деятельность).

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

2.1 Лабораторная работа № 1 (2 часа).

Тема: «Понятие информации»

Системы счисления. Позиционные системы счисления

2.1.1 Цель работы: получение навыков записи чисел в позиционных системах счисления; навыков перевода чисел из одной системы счисления с основанием p в систему счисления с основанием q . Выполнение арифметических операций с числами в позиционных системах счисления.

2.1.2 Задачи работы:

1. Изучить понятия: система счисления, алфавит системы счисления, основание системы счисления, способы представления чисел в позиционных системах счисления.
2. Изучить правила перевода чисел из одной системы счисления (с основанием p) в систему счисления с основанием q .
3. Изучить правила выполнения арифметических операций (сложение, умножение, вычитание, деление) с числами в позиционных системах счисления
4. Для закрепления и проверки полученных навыков, ответить на контрольные вопросы и выполнить задания. Результаты работы продемонстрировать преподавателю.

2.1.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Методические указания для выполнения лабораторной работы

2.1.4 Описание (ход) работы:

Перевод чисел из одной позиционной системы счисления в другую

1. Теоретическая часть

Системой счисления называется совокупность приемов наименования и записи чисел. В любой системе счисления для представления чисел выбираются некоторые символы (их называют *цифрами*), а остальные числа получаются в результате каких-либо операций над цифрами данной системы счисления.

Система называется *позиционной*, если значение каждой цифры (ее вес) изменяется в зависимости от ее положения (позиции) в последовательности цифр, изображающих число.

Число единиц какого-либо разряда, объединяемых в единицу более старшего разряда, называют *основанием позиционной системы счисления*. Если количество таких цифр равно P , то система счисления называется P -ичной. Основание системы счисления совпадает с количеством цифр, используемых для записи чисел в этой системе счисления.

Запись произвольного числа x в P -ичной позиционной системе счисления основывается на представлении этого числа в виде многочлена

$$x = a_n P^n + a_{n-1} P^{n-1} + \dots + a_1 P^1 + a_0 P^0 + a_{-1} P^{-1} + \dots + a_{-m} P^{-m}$$

Арифметические действия над числами в любой позиционной системе счисления производятся по тем же правилам, что и десятичной системе, так как все они основываются на правилах выполнения действий над соответствующими многочленами. При этом нужно только пользоваться теми таблицами сложения и умножения, которые соответствуют данному основанию P системы счисления.

При переводе чисел из десятичной системы счисления в систему с основанием $P > 1$ обычно используют следующий алгоритм:

1) если переводится целая часть числа, то она делится на P , после чего запоминается остаток от деления. Полученное частное вновь делится на P , остаток запоминается. Процедура продолжается до тех пор, пока частное не станет равным нулю. Остатки от деления на P выписываются в порядке, обратном их получению;

2) если переводится дробная часть числа, то она умножается на P , после чего целая часть запоминается и отбрасывается. Вновь полученная дробная часть умножается на P и т.д. Процедура продолжается до тех пор, пока дробная часть не станет равной нулю. Целые части выписываются после запятой в порядке их получения. Результатом может быть либо конечная, либо периодическая дробь в системе счисления с основанием P . Поэтому, когда дробь является периодической, приходится обрывать умножение на каком-либо шаге и довольствоваться приближенной записью исходного числа в системе с основанием P .

2. Порядок выполнения работы

1. Перевести данное число из десятичной системы счисления в двоичную: а) $464_{(10)}$; б) $380,1875_{(10)}$; в) $115,94_{(10)}$ (получить пять знаков после запятой в двоичном представлении).

Решение.

464 0	380 0	1875	115 1	94
232 0	190 0	0 375	57 1	1 88
116 0	95 1	0 75	28 0	1 76
58 0	47 1	1 5	14 0	1 52
а) 29 1	б) 23 1	1 0	в) 7 1	1 04
14 0	11 1		3 1	0 08
7 1	5 1		1 1	0 16
3 1	2 0			
1 1	1 1			

а) $464_{(10)} = 111010000_{(2)}$; б) $380,1875_{(10)} = 101111100,0011_{(2)}$; в) $115,94_{(10)} \square 1110011,11110_{(2)}$ (в настоящем случае было получено шесть знаков после запятой, после чего результат был округлен).

Если необходимо перевести число из двоичной системы счисления в систему счисления, основанием которой является степень двойки, достаточно объединить цифры двоичного числа в группы по столько цифр, каков показатель степени, и использовать приведенный ниже алгоритм. Например, если перевод осуществляется в восьмеричную систему, то группы будут содержать три цифры ($8 = 2^3$). Итак, в целой части будем производить группировку справа налево, в дробной — слева направо. Если в последней группе недостает цифр, дописываем нули: в целой части — слева, в дробной — справа. Затем каждая группа заменяется соответствующей цифрой новой системы. Соответствия приведены в таблицах.

P	2	00	01	10	11
	4	0	1	2	3

P	2	000	001	010	011	100	101	110	111
	8	0	1	2	3	4	5	6	7

<i>P</i>	2	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Переведем из двоичной системы в восьмеричную число 1111010101,11₍₂₎.

$$001\ 111\ 010\ 101,110_{(2)} = 1725,6_{(8)}.$$

Переведем из двоичной системы в шестнадцатеричную число 1111010101,11₍₂₎.

$$0011\ 1101\ 0101,1100_{(2)} = 3D5,C_{(16)}.$$

Соответствие между шестнадцатеричными цифрами и десятичными числами																	
16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
10	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

При переводе чисел из системы счисления с основанием *P* в десятичную систему счисления необходимо пронумеровать разряды целой части справа налево, начиная с нулевого, и в дробной части, начиная с разряда сразу после запятой слева направо (начальный номер -1). Затем вычислить сумму произведений соответствующих значений разрядов на основание системы счисления в степени, равной номеру разряда. Это и есть представление исходного числа в десятичной системе счисления.

2. Перевести данное число в десятичную систему счисления.

Некоторые неотрицательные степени числа 2 (в десятичной системе счисления)																		
Показатель	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
Степень	1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65536	

Некоторые отрицательные степени числа 2 (в десятичной системе счисления)

Показатель	-1	-2	-3	-4	-5	-6	-7
Степень	0,5	0,25	0,125	0,0625	0,03125	0,015625	0,0078125

а) 1000001₍₂₎.

$$1000001_{(2)} = 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 64 + 1 = 65_{(10)}.$$

Замечание. Очевидно, что если в каком-либо разряде стоит нуль, то соответствующее слагаемое можно опускать.

б) 1000011111,0101₍₂₎.

$$1000011111,0101_{(2)} = 1 \cdot 2^9 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-2} + 1 \cdot 2^{-4} = 512 + 16 + 8 + 4 + 2 + 1 + 0,25 + 0,0625 = 543,3125_{(10)}.$$

Некоторые неотрицательные степени числа 8 (в десятичной системе счисления)					
Показатель	0	1	2	3	4
Степень	1	8	64	512	4096

Некоторые отрицательные степени числа 8 (в десятичной системе счисления)

Показатель	-1	-2
Степень	0,125	0,015625

в) 1216,04₍₈₎.

$$1216,04_{(8)} = 1 \cdot 8^3 + 2 \cdot 8^2 + 1 \cdot 8^1 + 6 \cdot 8^0 + 4 \cdot 8^{-2} = 512 + 128 + 8 + 6 + 0,0625 = 654,0625_{(10)}.$$

Некоторые неотрицательные степени числа 16 (в десятичной системе счисления)					
Показатель	0	1	2	3	4

Степень	1	16	256	4096	65536
Некоторые отрицательные степени числа 16 (в десятичной системе счисления)					
Показатель	-1		-2		
Степень	0,0625		0,00390625		

г) $29A,5_{(16)}$.

$$29A,5_{(16)} = 2 \cdot 16^2 + 9 \cdot 16^1 + 10 \cdot 16^0 + 5 \cdot 16^{-1} = 512 + 144 + 10 + 0,3125 = 656,3125_{(10)}.$$

3. Контрольные вопросы и задания

1. Дать определение системы счисления. Назвать и охарактеризовать свойства системы счисления.
2. Какие символы используются для записи чисел в двоичной системе счисления, восьмеричной, шестнадцатеричной?
3. Чему равны веса разрядов слева от точки, разделяющей целую и дробную часть, в двоичной системе счисления (восьмеричной, шестнадцатеричной)?
4. Чему равны веса разрядов справа от точки, разделяющей целую и дробную часть, в двоичной системе счисления (восьмеричной, шестнадцатеричной)?
5. Преобразуйте следующие десятичные числа в двоичные (восьмеричные, шестнадцатеричные): 0, 1, 18, 25, 128.
6. Дешифруйте следующие двоичные числа, преобразовав их в десятичные: 0010, 1011, 11101, 0111, 0101.
7. Дешифруйте следующие восьмеричные числа, преобразовав их в десятичные: 777, 375, 111, 1015.
8. Дешифруйте следующие шестнадцатеричные числа, преобразовав их в десятичные: 15, A6, 1F5, 63.

Арифметические операции с числами в позиционных системах счисления

1. Порядок выполнения работы

Для выполнения арифметических операций в системе счисления с основанием P необходимо иметь соответствующие таблицы сложения и умножения. Для $P = 2, 8$ и 16 таблицы представлены ниже.

+	0	1
0	0	1
1	1	10

×	0	1
0	0	0
1	0	1

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25

4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

×□□	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4

D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

Сложить числа:

а) $10000000100_{(2)} + 111000010_{(2)} = 10111000110_{(2)}$.

б) $223,2_{(8)} + 427,54_{(8)} = 652,74_{(8)}$.

в) $3B3,6_{(16)} + 38B,4_{(16)} = 73E,A_{(16)}$.

10000000100	223,2	3B3,6
+ 111000010	+ 427,54	+ 38B,4
-----	-----	-----
10111000110	652,74	73E,A

Выполним проверку результатов расчетов переводом в десятичную систему счисления. Для этого переведем каждое слагаемое и сумму в десятичную систему счисления, выполним сложение слагаемых в десятичной системе счисления. Результат должен совпасть с суммой.

а) $10000000100_{(2)} = 1 \cdot 2^{10} + 1 \cdot 2^2 = 1024 + 4 = 1028_{(10)}$

$111000010_{(2)} = 1 \cdot 2^8 + 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^1 = 256 + 128 + 64 + 2 = 450_{(10)}$

$10111000110_{(2)} = 1 \cdot 2^{10} + 1 \cdot 2^8 + 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^2 + 1 \cdot 2^1 = 1024 + 256 + 128 + 64 + 4 + 2 = 1478_{(10)}$

$1028_{(10)} + 450_{(10)} = 1478_{(10)}$

Результаты совпадают, следовательно, вычисления в двоичной системе счисления выполнены верно!

б) $223,2_{(8)} = 2 \cdot 8^2 + 2 \cdot 8^1 + 3 \cdot 8^0 + 2 \cdot 8^{-1} = 128 + 16 + 3 + 0,25 = 147,25_{(10)}$

$427,54_{(8)} = 4 \cdot 8^2 + 2 \cdot 8^1 + 7 \cdot 8^0 + 5 \cdot 8^{-1} + 4 \cdot 8^{-2} = 256 + 16 + 7 + 0,625 + 0,0625 = 279,6875_{(10)}$

$652,74_{(8)} = 6 \cdot 8^2 + 5 \cdot 8^1 + 2 \cdot 8^0 + 7 \cdot 8^{-1} + 4 \cdot 8^{-2} = 384 + 40 + 2 + 0,875 + 0,0625 = 426,9375_{(10)}$

$147,25_{(10)} + 279,6875_{(10)} = 426,9375_{(10)}$

Результаты совпадают, следовательно, вычисления в восьмеричной системе счисления выполнены верно!

в) $3B3,6_{(16)} = 3 \cdot 16^2 + 11 \cdot 16^1 + 3 \cdot 16^0 + 6 \cdot 16^{-1} = 768 + 176 + 3 + 0,375 = 947,375_{(10)}$

$38B,4_{(16)} = 3 \cdot 16^2 + 8 \cdot 16^1 + 11 \cdot 16^0 + 4 \cdot 16^{-1} = 768 + 128 + 11 + 0,25 = 907,25_{(10)}$

$73E,A_{(16)} = 7 \cdot 8^2 + 3 \cdot 8^1 + 14 \cdot 8^0 + 10 \cdot 8^{-1} = 1792 + 48 + 14 + 0,625 = 1854,625_{(10)}$

$947,375_{(10)} + 907,25_{(10)} = 1854,625_{(10)}$

Результаты совпадают, следовательно, вычисления в шестнадцатеричной системе счисления выполнены верно!

Выполнить вычитание:

а) $1100000011,011_{(2)} - 101010111,1_{(2)} = 110101011,111_{(2)}$.

б) $1510,2_{(8)} - 1230,54_{(8)} = 257,44_{(8)}$.

в) $27D,D8_{(16)} - 191,2_{(16)} = EC,B8_{(16)}$.

1100000011,011	1510,2	27D,D8
- 101010111,1	-1230,54	-191,2
-----	-----	-----
110101011,111	257,44	EC,B8

Выполнить умножение:

а) $100111_{(2)} \times 1000111_{(2)} = 101011010001_{(2)}$.

б) $1170,64_{(8)} \times 46,3_{(8)} = 57334,134_{(8)}$.

в) $61,A_{(16)} \times 40,D_{(16)} = 18B7,52_{(16)}$.

100111	1170,64	61,A
* 1000111	* 46,3	* 40,D
-----	-----	-----
100111	355 234	4F 52
+ 100111	+ 7324 70	+ 1868

100111	47432 0	-----
100111	-----	18B7, 52
-----	57334, 134	
101011010001		

Выполнить деление:

а) $100110010011000_{(2)} : 101011_{(2)} = 111001000_{(2)}$;

б) $46230_{(8)} : 53_{(8)} = 710_{(8)}$;

в) $4C98_{(16)} : 2B_{(16)} = 1C8_{(16)}$.

$$\begin{array}{r}
 100110010011000 \overline{) 101011} \\
 \underline{101011} \\
 1000011 \\
 \underline{101011} \\
 110000 \\
 \underline{101011} \\
 101011 \\
 \underline{101011} \\
 0
 \end{array}$$

$$\begin{array}{r}
 46230 \overline{) 53} \\
 \underline{455} \\
 53 \\
 \underline{53} \\
 0
 \end{array}$$

$$\begin{array}{r}
 4C98 \overline{) 2B} \\
 \underline{2B} \\
 219 \\
 \underline{204} \\
 158 \\
 \underline{158} \\
 0
 \end{array}$$

2. Контрольные вопросы и задания

1) Каковы правила выполнения арифметических операций над числами в двоичном представлении?

2) Сложите, вычтите из большего меньшее, перемножьте и разделите первое на второе числа в двоичном представлении 1101001110011101 и 1001011010110111.

3) Выполнить указанные действия над числами в позиционных системах счисления:

а) $1110110011_{(2)} + 110001111_{(2)}$

б) $1034_{(8)} + 205_{(8)}$

в) $33C_{(16)} + 37D_{(16)}$

г) $110000101001_{(2)} - 111101_{(2)}$

д) $1733_{(8)} - 355_{(8)}$

е) $26F_{(16)} - D3_{(16)}$

ж) $1001111_{(2)} \cdot 1000100_{(2)}$

з) $1017_{(8)} \cdot 73_{(8)}$

и) $56_{(16)} \cdot 4A_{(16)}$

к) $11000001011_{(2)} : 10001_{(2)}$

л) $3440_{(8)} : 23_{(8)}$

м) $854_{(16)} : 1A_{(16)}$

2.2 Лабораторная работа № 2 (2 часа).

Тема: «Программные средства реализации информационных процессов. Системное программное обеспечение»

2.2.1 Цель работы: освоение работы в операционной системе Windows.

2.2.2 Задачи работы:

1. Изучить возможности операционной системы WINDOWS.
2. Освоить основные операции с объектами WINDOWS.
3. Освоить основные приемы работы со стандартными приложениями WINDOWS.
4. Для закрепления и проверки полученных навыков, необходимо ответить на контрольные вопросы. Результаты работы продемонстрировать преподавателю в электронном виде.

2.2.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Методические указания для выполнения лабораторной работы

2.2.4 Описание (ход) работы:

Операционная система WINDOWS. Операции с объектами WINDOWS.

1. Теоретическая часть

Обзор возможностей операционной системы WINDOWS

ОС Windows является 32-разрядной операционной системой. С ее появлением многие сложности и проблемы, связанные с 16-разрядными операционными системами остались позади. При этом ОС Windows обеспечивает совместимость с Windows 3.1 и DOS. т.е. в среде ОС Windows можно использовать множество уже существующих приложений, созданных для персональных компьютеров класса IBM PC.

ОС Windows имеет усовершенствованный интерфейс пользователя. Главная цель здесь предоставление любому пользователю, имеющему минимальные знания о компьютере, возможности сесть и запустить практически любое приложение без обучения. Главная особенность оконного интерфейса состоит в том, что он обеспечивает на экране модель «рабочего стола». Окна на экране являются эквивалентами разбросанных на столе бумаг. Выбирая окно, можно сделать его активным, т.е. поместить его поверх остальных, как бумаги на столе окна можно перемещать и перекладывать окна на экране. В продолжении аналогий с рабочим столом Кatalоги теперь называются Папками (Folder). Все файлы сосредоточены в Папках.

Для быстрого доступа к наиболее часто используемым файлам или устройствам в ОС Windows используются объекты, получившие названия Ярлыков (Shortcut). Ярлык к программе, документу, устройству может быть создан в любой папке или на Рабочем столе.

Рабочий стол ОС Windows снабжен кнопкой Пуск (Start) и имеет специальную область Панель задач (Taskbar). Кнопка Пуск предназначена для быстрого запуска программ. Панель задач позволяет очень просто переключаться между приложениями запущенными в среде ОС Windows.

ОС Windows позволяет присваивать длинные имена любому файлу. В именах файлов можно использовать пробелы, предельная длина имени файла составляет 255 символов. Имена файлов не должны содержать следующие символы : « \ », « / », « : », « * », « ? », « " », « < », « > ».

В ОС Windows реализован механизм вытесняющей многозадачности, который позволяет одновременно запускать несколько приложений. Эти приложения теперь выполняются быстрее.

ОС Windows снабжена встроенной системой Plug-and-Play. Технология Plug-and-Play предоставляет автоматизировать процесс установки дополнительного оборудования. Пользователь физически устанавливает оборудование, операционная система сама отыскивает необходимые драйвера и настраивает работу устройства в среде ОС Windows. Специализированные средства Мастера (Wizards) помогают выполнять операции по настройке оборудования.

Для удобства управления программами и файлами в ОС Windows появились системные программы Мой компьютер (My Computer), Сетевое окружение (Network Neighborhood), Корзина (Recycle Bin). Мой компьютер объект, который визуализирует структуру устройств, папок и файлов компьютера и позволяет проводить операции над ними. Для реализаций этих операций используется прием Drag-and-Drop (потащить и бросить). Перетаскивать можно любые объекты (окна, значки элементов, папки, файлы). Для этого мышкой выбирается объект, нажимается левая кнопка, удерживая кнопку, перемещаем указатель мыши вместе с выбранным объектом в нужное место и отпускаем клавишу мыши.

Программа Сетевое окружение обеспечивает доступ к сетевым ресурсам компьютера. Корзина - это место для временного помещения удаленных файлов. Через обращение к Корзине можно восстанавливать ошибочно удаленные файлы.

Рабочий стол Windows

Легко представить рабочий стол как чистый лист, находящийся за всеми остальными изображениями на дисплее, однако рабочий стол является гораздо большим, нежели просто фон. Помимо изменяемого фона рабочий стол Windows 95 содержит папки, значки и другие объекты (см. рис.1). Для удобства управления программами и файлами в ОС Windows появились системные программы Мой компьютер (My Computer), Сетевое окружение (Network Neighborhood), Корзина (Recycle Bin). Мой компьютер объект, который визуализирует структуру устройств, папок и файлов компьютера и позволяет проводить операции над ними. Для реализаций этих операций используется прием Drag-and-Drop (потащить и бросить). Перетаскивать можно любые объекты (окна, значки элементов, папки, файлы). Для этого мышкой выбирается объект, нажимается левая кнопка, удерживая кнопку, перемещаем указатель мыши вместе с выбранным объектом в нужное место и отпускаем клавишу мыши.

Программа Сетевое окружение обеспечивает доступ к сетевым ресурсам компьютера, если данный ПК подсоединен к сети. Эти сетевые ресурсы включают диски и принтеры, общие для всех компьютеров в сети. Корзина - это место для временного помещения удаленных файлов. Через обращение к Корзине можно восстанавливать ошибочно удаленные файлы.

Панель задач также является частью рабочего стола.

В следующих разделах содержится обзор объектов, являющихся составными частями рабочего стола.

Папки

Папки являются средством организации и представления системных ресурсов ПК (каталогов, файлов, программ и т.д.). Папка может содержать другие папки (вложенные папки), программы, а также такие объекты, как, например, принтеры и диски. Объекты в папке представляются значками и каждый значок имеет название, расположенное ниже него. Для того, чтобы открыть папку, запустить программу, открыть документ или активизировать и открыть объект любого другого типа в папке, достаточно дважды щелкнуть на соответствующем значке. Можно создавать свои собственные папки на рабочем столе для хранения объектов (программ и документов), которые вы используете наиболее часто. Важно понимать, что стандартная папка — это в действительности

графическое представление каталога. Если документ копируется в папку, то на самом деле он копируется в каталог, представленный этой папкой. Если вы создаете новую папку в папке, то вы тем самым создаете подкаталог в каталоге родительской папки. Некоторые папки, такие как Панель управления и Принтеры в действительности папками не являются. Они являются прикладными программами и специальными управляющими программами, которые Windows 98/2000 представляет как папки для того, чтобы обеспечить единообразный интерфейс со всеми ресурсами ПК.

Рабочий стол сам является в действительности скрытым каталогом под названием «\WINDOWS\Рабочий стол». Если вы копируете программный файл или файл документа в каталог «\WINDOWS\Рабочий стол», то его значок появляется в качестве объекта на рабочем столе. Аналогично, если вы создаете подкаталог в каталоге «\WINDOWS\Рабочий стол», то папка этого подкаталога появляется на рабочем столе. Это важно понимать, когда вы начинаете копировать или перемещать объекты на рабочем столе.

Панель задач

Панель задач обеспечивает другой способ открытия программ и документов, а также доступ к объектам, которые уже были открыты. Меню Пуск на панели задач содержит пункты, которые позволяют получить доступ к программам, документам и другим объектам. Оно содержит также команды для запуска программ из командной строки, установки системных параметров и завершения работы Windows 98/2000.

В нижеследующем списке приведены стандартные пункты меню Пуск:

Программы. Это каскадное меню дает быстрый доступ ко всем стандартным (в том числе игровым) программам, которые поставляются в составе Windows 98/2000. Оно включает также пункты для открытия Проводника и запуска сеанса MS-DOS. Документы. Это меню содержит 15 документов и папок, с которыми вы работали последними, включая те, с которыми вы работали в текущем сеансе.

Настройка. Это каскадное меню дает возможность доступа к Панели управления (используется для конфигурирования системы), к папке Принтеры (которая позволяет управлять всеми имеющимися принтерами и к листам свойств панели задач.

Поиск. Это меню позволяет найти файл или папку на компьютере или в сети, а также найти в сети определенный компьютер.

Справка. Этот пункт меню открывает справочную систему Windows 98/2000.

Выполнить. Эта команда обеспечивает быстрый запуск программ. Вы можете ввести команду непосредственно или просмотреть каталоги, чтобы найти нужную программу.

Остановка. Эта команда появляется в меню Пуск, если данный ПК поддерживает режим экономии питания. Она используется для перевода ПК в этот режим.

Завершение работы. Выберите этот пункт меню для завершения с работы ПК, перезагрузки Windows 98/2000 или выхода из сети (но с сохранением работы Windows 98/2000).

Помимо меню Пуск на панели задач располагаются кнопки всех папок и программ, открытых на данный момент. Для выбора объекта необходимо щелкнуть на его кнопке на панели задач. При этом объект перемещается на передний план, после чего с ним можно работать.

Создание новых папок

Вы можете создать папку на рабочем столе или внутри другой папки. Главное, что нужно помнить при создании папки, это то, что папка является ничем иным, как каталогом — когда вы создаете папку, вы на самом деле создаете каталог.

Для того, чтобы создать папку на рабочем столе, щелкните на нем правой кнопкой мыши. Выберите из появившегося меню пункт Создать и затем Папка. Windows 98/2000 создаст на экране новую папку и присвоит ей имя из восьми знаков «минус». Вначале название будет выделено. Наберите желаемое название и нажмите клавишу Enter. Для

изменения названия существующей папки щелкните на папке, чтобы выбрать ее, затем щелкните на имени папки и отредактируйте его.

После того, как вы создали новую папку, все готово для копирования и перемещения объектов на рабочий стол или в папку.

Копирование и перемещение объектов

Вы можете копировать или перемещать программы, документы или другие объекты на рабочий стол или в папки. В обоих случаях ваши действия очень похожи. Чтобы скопировать объект из папки на рабочий стол или в другую папку, нажмите клавишу Ctrl и, удерживая ее, перетащите объект из папки на экран или из одной папки в другую. Для того, чтобы переместить объект, его надо просто тащить, не нажимая при этом на клавиатуре никаких клавиш. Таким образом, можно переместить объект только в пределах диска. При перетаскивании объекта на другой диск он будет скопирован. Чтобы переместить объект на другой диск, при перетаскивании удерживайте нажатой клавишу Shift.

При копировании и перемещении объектов помните, что файлы, входящие в копируемый объект, также копируются или перемещаются в новое место.

Если вы предпочитаете не пользоваться клавиатурой в сочетании с мышью или испытываете трудности в запоминании того, какую клавишу следует нажимать в какой ситуации, попробуйте тащить объект правой кнопкой вместо левой. Когда вы тащите объект правой кнопкой и освобождаете его, Windows выводит на экран контекстное меню. При этом Windows предлагает вам переместить или скопировать объект, создать его ярлык или прервать операцию.

Поиск стандартными средствами Windows

Стандартный поиск в Windows предполагает поиск по некоторой части имени файла и/или части текста файла. Например, в поле “Папка” указано “C:”, в поле “Имя” указано “а”, в поле “Искать текст” указано “ля-ля”. Будут найдены все файлы и все папки на диске C:, в имени (или расширении) которых присутствует буква а, а в самом файле где-либо встречается последовательность “ля-ля”. Для более конкретного поиска существуют так называемые маски.

Маска

Для поиска файлов с заданными условиями на имя файла используется маска. Маска - это схематичная запись условий на имя файла. Для этого используются следующие символы: “*” и “?”. “*” означает произвольную (даже пустую) последовательность символов. “?” означает ровно один символ. Кроме того, в маске можно использовать все те же символы, что и в именах файлов (включая “.”), они будут означать сами себя.

Например, если мы хотим найти все файлы, начинающиеся на букву “а”, надо использовать маску “а*.”. (Расшифровка: сначала буква “а”, потом произвольная последовательность символов (м.б. пустая)).

Если мы хотим найти все файлы, в имени которых есть буква “а”, надо использовать маску “*а*.”. (Расшифровка: сначала произвольная последовательность символов, затем буква “а”, затем снова произвольная последовательность символов).

Если мы хотим найти файлы, вторая буква имени которых – “а”, надо использовать маску “?а*.”. (Расшифровка: сначала одна произвольная буква, затем буква “а”, потом произвольная последовательность символов).

Если мы хотим найти файл с заданным расширением “abc”, мы должны использовать маску “*.abc”

Ярлыки

В Windows существуют специальные файлы-ярлыки (линки, *.lnk). Программу (или др. файл) не всегда можно поместить в удобном для доступа месте. Поэтому в этом самом удобном месте (например, на рабочем столе или в главном меню) помещают ярлык для этой программы (файла).

2. Порядок выполнения работы

Задание 1

1. На рабочем столе создать папку с именем «Первая».
2. Открыть папку «Первая» и в ней создать папки «Вторая», «Третья» и «Четвертая».
3. С помощью команды Вид проверить включение следующих панелей (панель считается включенной, если она помечена галочкой):
 - Обычные кнопки;
 - Адресная строка;
 - Подписи к кнопкам.
4. С помощью команды Вид проверить содержание папки «Первая», последовательно устанавливая режимы: крупные значки, Мелкие значки, Таблица, Список.
5. В папке «Третья» создать текстовый документ с именем a.txt. следующего содержания:

«Организаторы стремятся нанимать победителей. Преподносите себя как подарок судьбы их фирме. Упаси Вас Бог выглядеть жалкой попрошайкой».
6. Выполнить следующие действия;
 - скопировать второе предложение в конец текста;
 - поменять местами первое и последнее предложения текста;
 - вставить в конец текста текущую дату.
7. Открыть окно приложения Калькулятор и выполнить следующие действия:
 $(344: 17 + 12.5) + 14 + 22^2 =$
8. Полученный результат, используя Буфер обмена, вставить в конец документа a.txt.
9. Свернуть окно «Калькулятора» до ярлыка на Панели задач, а затем закрыть окно из Панели задач с помощью контекстно-зависимого меню.
10. Закрыть окно текстового документа a.txt, сохранив его содержимое.
11. Удалить папку «Первая» с Рабочего стола вместе с ее содержимым в Корзину.
12. Открыть Корзину и убедиться, что удаленная папка «Первая» переместилась в Корзину.
13. Удалить папку «Первая» из Корзины.

Задание 2

1. На диске C: создать папку «Find».
2. В ней создать подпапки «a», «b», «c», «d».
3. Найти на диске C: все файлы, в имени которых встречается буква «a» (латинская).
4. Скопировать все эти файлы в папку C:\Find\A\.
5. Найти на диске C: все файлы, у которых «b» □ последняя буква имени (без учета расширения).
6. Скопировать все эти файлы в папку C:\Find\B\ (можно с клавиатуры: выделить несколько объектов – Ctrl; копировать объекты в буфер обмена – Ctrl+C).
7. В папке C:\Windows\ найти все файлы, название которых состоит из трех букв, а расширение содержит букву «c» (латинская). Вложенные папки не рассматривать.
8. Скопировать все эти файлы в папку C:\Find\C\.
9. Перенесите содержимое папки "a" в "d"(можно с клавиатуры: поместить объекты в буфер обмена – Ctrl+X; вставить объекты из буфера обмена - Ctrl+V).
10. В папке C:\Find\A\ создайте документ Microsoft Word и лист Microsoft Excel. В каждом файле впишите свою фамилию и дату.

11. При помощи стандартного средства поиска Windows посчитать, сколько файлов на диске C:.
12. Создать в папке C:\Find\ ярлыки для программ mspaint.exe, explorer.exe, которые находятся на одном из локальных дисков.
13. Изменить значок в свойствах ярлыка программы Проводник, подобрав подходящий из стандартных.
14. Сверните, разверните окно папки Find.
15. Восстановите нормальный размер окна папки Find и измените размеры его до максимума (на весь рабочий стол) и до минимума (чтобы были видны только созданные ярлыки).
16. Переместите окно папки Find сначала в верхний левый, затем в правый нижний угол рабочего стола.
17. Откройте в меню Пуск группу стандартных программ и отправьте на рабочий стол ярлык программы Калькулятор (используя контекстное меню).
18. Переместите ярлык программы Калькулятор с рабочего стола в папку Find.
19. В свойствах ярлыка Калькулятор установите для его быстрого вызова комбинацию клавиш, нажав ее (например: Ctrl+Alt+C).
20. Откройте программу Калькулятор установленной комбинацией клавиш.
21. Произведите любые арифметические и тригонометрические вычисления и закройте Калькулятор.
22. Загрузите графический редактор Paint через созданный ярлык.
23. Загрузите текстовый редактор WordPad через меню Пуск.
24. Переключитесь в окно Paint.
25. Вызовите диспетчер задач (Alt+Ctrl+Delete) и снимите задачу WordPad без сохранения данных (завершение выполнения "зависших" программ).
26. Закройте окно Paint, используя системное меню окна.
27. Сделайте копию ярлыков программы Проводник и Paint в папке Find.
28. Удалите копию ярлыка программы Проводник из папки Find перетаскиванием в Корзину.
29. Удалите копию ярлыка программы Paint из папки Find, минуя Корзину (Delete+Shift). В чем отличие этого способа удаления файлов от предыдущего?
30. Откройте свойства рабочего стола (экрана). Используя контекстную подсказку, изучите параметры настройки экрана.
31. Выберите любой фоновый рисунок из списка и нажмите кнопку Применить.
32. Измените оформление, выбрав любую цветовую схему из списка и нажмите ОК.
33. Верните прежнюю цветовую схему, выбрав Стандартная из списка и нажмите ОК.
34. Откройте свойства панели задач и изучите параметры (используя контекстную подсказку).
35. Выключите отображение часов в панели и включите "автоматически убирать панель с экрана". Нажмите кнопку ОК.
36. Откройте свойства панели задач, верните старые параметры, установите классическое меню "Пуск" и нажмите ОК.
37. Подготовьте компьютер к выключению и после появления надписи на экране, разрешающей выключение, выключите питание.

Проводник

1. Откройте программу Проводник, используя контекстное меню кнопки Пуск.
2. Создайте на диске C: папку с номером вашей группы.
3. Откройте ее и в ней создайте папку RAB.

4. Выполните поиск файла eula.txt на диске C:.
5. Скопируйте найденный файл в папку RAB, используя кнопки на панели инструментов.
6. Отключите в свойствах вида папки опцию скрывания расширений для зарегистрированных файлов (Сервис-Свойства папки).
7. Переименуйте файл eula.txt в новый.doc.
8. Откройте папку WINDOWS и измените поочередно вид на: список, эскизы, таблица.
9. Упорядочьте содержимое папки по времени изменения, а затем по размеру, отслеживая изменения в таблице ярлыков. Найдите самый старый и самый большой файл.
10. Выделите и скопируйте с помощью меню Правка 5 произвольных файлов, суммарный объем которых не более 35 Кбайт из папки C:\WINDOWS в папку RAB.
11. Используя комбинации клавиш, переместите (вырезать и вставить) файл новый.doc в корневую папку диска C:.
12. Упорядочьте ярлыки по типу.
13. Скопируйте перетаскиванием все текстовые документы (файлы с расширением txt) из папки C:\WINDOWS в папку RAB.
14. Установите в свойствах файла новый.doc параметр «скрытый».
15. Выключите режим отображения «скрытых» файлов (спрятать имеющие статус «скрытый» файлы).
16. Создайте текстовый документ в директории RAB.
17. Откройте созданный файл, наберите в нем несколько строк и сохраните.
18. Удалите созданный файл перетаскиванием мышью.
19. Включите режим отображения «скрытых» файлов.
20. Удалите тренировочные файлы, используя клавишу на клавиатуре.
21. Удалите папку RAB перетаскиванием в Корзину панели папок.
22. Выйдите из Проводника.

Стандартные приложения Windows

1. Запустите графический редактор Paint.
2. Сверните все окна.
3. Скопируйте в буфер обмена изображение рабочего стола Windows (клавиша Print/PrtScr).
4. Разверните Paint и вставьте содержимое буфера обмена в рабочую область программы.
5. Используя инструменты: линия и надпись, сделайте подписи с прозрачным фоном к основным элементам рабочего стола.
6. Сохраните рисунок в папку с номером вашей группы, введя имя "Рабочий стол" и установив тип файла JPG.
7. Очистите рабочую область.
8. Отмените очистку (последнюю операцию).
9. Создайте новый файл без сохранения последних изменений в файле Рабочий стол.jpg.
10. Растяните рабочую область на все окно.
11. Сохраните заготовку рисунка в папку Практика по информатике, введя имя файла Пейзаж и установив тип JPG.
12. Используя все инструменты рисования, создайте рисунок, например: домик на фоне голубого неба, белых облаков, желтого солнца, зеленой травки с цветочками и с серой дорожкой к порогу (можно нарисовать автомобиль, корабль, море или ночное небо). Периодически в процессе рисования сохраняйте внесенные изменения в рисунок.
13. Закройте графический редактор Paint с сохранением изменений в файле.
14. Сделайте рисунок Пейзаж.jpg фоном рабочего стола.

15. В соответствии с вариантом задания средствами графического редактора Paint подготовить рисунок. Высота рисунка - 6 см (Рисунок-Атрибуты).

16. Средствами текстового процессора WordPad подготовить отчет о работе, содержащий рисунок из Paint. При описании порядка создания рисунка следует выделить обычным курсивом названия инструментов, а полужирным курсивом – выбранные пункты меню.

17. Для защиты работы продемонстрировать отчет на компьютере в среде WordPad.

3. Контрольные вопросы

1. Что такое программное обеспечение компьютера?
2. Чем отличается системное программное обеспечение от прикладного? Приведите примеры системного и прикладного программного обеспечения.
3. Что такое операционная система? Какие операционные системы Вам известны?
4. Опишите основные элементы ОС Windows (рабочий стол, окно, значок, ярлык, кнопка, панель задач, меню, папка, меню «Пуск»).
5. Какие разновидности окон существуют в ОС Windows?
6. Опишите основные приемы работы с окнами в ОС Windows.
7. Что значит понятие «контекстное меню»? Как можно вызвать данное меню?
8. Опишите основные возможности программы «Проводник».
9. Как можно создать новую папку в ОС Windows?
10. Как можно переименовывать, копировать и удалять объекты в ОС Windows?

2.3 Лабораторная работа № 3 (2 часа).

Тема: «Программные средства реализации информационных процессов. Сервисное программное обеспечение»

2.3.1 Цель работы: приобретение навыков работы с сервисными программами.

2.3.2 Задачи работы:

1. Знакомство с архиватором WinRAR для ОС Windows и приобретение навыков архивирования, извлечения файлов из архивов, добавления файлов в архив.
2. Научиться повышать производительность работы жесткого диска путем буферизации данных, оптимизации размещения файлов и каталогов с помощью существующих программных средств.
3. Научиться восстанавливать данные на диске сразу после стирания, восстанавливать каталоги и частично восстанавливать файлы путем копирования отдельных кластеров после выполнения операций записи на диск.
4. Для закрепления и проверки полученных навыков, необходимо ответить на контрольные вопросы. Результаты работы продемонстрировать преподавателю в электронном виде.

2.3.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Методические указания для выполнения лабораторной работы

2.3.4 Описание (ход) работы:

Работа с сервисными программами. Работа с большими файлами. Архивация/разархивация файлов

1. Теоретические сведения

Архивация данных - это уменьшение физических размеров файлов, в которых хранятся данные, без значительных информационных потерь. Архиваторы (утилиты - упаковщики, программы - упаковщики) – программы (комплекс программ), выполняющие сжатие и восстановление сжатых файлов в первоначальном виде.

Для выполнения этой лабораторной работы необходимо ознакомиться с методами архивации данных и возможностями архиватора WinRAR, изложенными в лекционном курсе или других источниках информации.

2. Порядок выполнения работы

1. На своем электронном носителе: создайте структуру каталогов и файлов, представленную на Рис. 1.
2. В папке AUDIT выполните архивирование файла audit_1, используя различные способы сжатия.
3. В папке AUDIT выполните архивирование файла audit_1, создавая самораспаковывающиеся файлы (SFX-файл) с различными способами сжатия.
4. В папке ECONOMY выполните архивирование папки MANAGEMENT, размещая архив с именем management.rar в папке DATA.
5. Удалите из созданного архива файл ment_1.
6. Выполните извлечение файлов из архива management.rar в папку DATA.
7. Добавьте файлы group_1 и rate_29 в архив management.rar.
8. Удалите архив management.rar.

9. Добавьте в архив bas.rar две папки ECONOMY, BAS и файлы group_1, rate_29, поместив архив bas.rar в каталог BAS на диске A:.
10. Создайте непрерывный архив.
11. Протестируйте архив.
12. Просмотрите информацию об архиве.
13. Переименуйте архив.
14. Преобразуйте обычный архив в самораспаковывающийся архив.
15. Удалите архив.

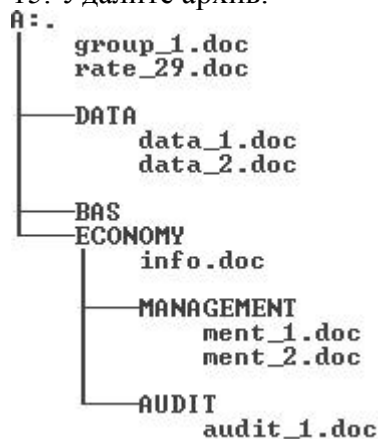


Рис. 1.

1. Включите ПК

Нажмите кнопку Power на системном блоке ПК.

2. Создание структуры каталогов и файлов

На своем электронном носителе: создайте любым способом структуру папок и файлов в соответствии с Рис. 1. В качестве исходных файлов используйте любые файлы с расширением .doc, которые размещены на ПК.

3. Ознакомьтесь с архиватором WinRAR

Запустите архиватор WinRAR из Главного меню и в открывшемся окне ознакомьтесь со строкой меню, панелью инструментами и другими элементами окна архиватора.

4. Создание архивных файлов

В WinRAR выберите каталог AUDIT и выполните архивирование файла audit_1, используя различные способы сжатия.

Для архивирования выполните следующее:

- выделите указанный файл;
- в строке инструментов щелкните на пиктограмме Добавить файлы в архив или выполните команду Команды/Добавить файл в архив;
- в открывшемся окне "Имя и параметры архива" выберите один из методов сжатия в раскрывающемся списке "Методы сжатия";
- переключатель "Формат архива" установите в положение RAR;
- в текстовое поле "Имя архива" введите новое имя архивного файла, соответствующее методу сжатия (например, обычный) и щелкните на кнопке ОК;
- повторите вышеуказанные действия, создав шесть архивных файлов с различными методами сжатия и разными именами соответственно;
- сравните размеры исходного файла и созданных архивных файлов с различными методами сжатия.

5. Просмотр содержимого архивных файлов

Для просмотра архивных файлов выполните следующее:

- в окне архиватора WinRAR выделите архивный файл;
- в строке инструментов щелкните на пиктограмме Просмотр или выполните команду "Команды"/"Просмотреть файл".

6. Создание самораспаковывающихся файлов

В каталоге AUDIT создайте группу самораспаковывающихся файлов, повторив пункт 8.4.4.

7. Создание архивной папки management.rar

Выполните архивирование папки MANAGEMENT, создав архив с именем management.rar в папке DATA:

- в архиваторе откройте папку EGONOMY и выделите папку MANAGEMENT;
- в строке инструментов щелкните на пиктограмме Добавить файлы в архив;
- для указания полного имени архивного файла в текстовой строке "Имя архива" воспользуйтесь кнопкой "Обзор". Выберите папку DATA, а имя архивного файла management.rar;

- в окне архиватора перейдите в папку DATA и просмотрите содержимое созданного архивного файла.

8. Удаление файла из архива management.rar

Удалите исходный файл ment_1 из созданного архива management.rar. Для удаления указанного файла в окне архиватора откройте архивный файл (дважды щелкните на архивном файле, затем дважды щелкните на открывшейся папке MANAGEMENT), выделите файл ment_1 и щелкните на пиктограмме Удалить на панели инструментов. В результате файл будет удален.

9. Извлечение файла из архива management.rar

Для извлечения файла из архива management.rar, откройте архивный файл, выделите файл ment_2 и щелкните на пиктограмме Извлечь на панели инструментов. В результате файл будет извлечена папка и требуемый файл. Убедитесь в том, что указанный файл извлечен, но не удален из архива.

10. Добавление файлов в архив management.rar

Для добавления файлов group_1, rate_29 в существующий архив management.rar выполните следующее:

- в окне архиватора откройте корневую папку своего электронного носителя;
- выделите указанные файлы, щелкните на пиктограмме Добавить файлы в архив;
- укажите имя файла A:\DATA\management.rar и щелкните на кнопке ОК;
- удалите архив management.rar.

11. Создание архивного файла bas.rar

Для добавления в архив bas.rar двух папок ECONOMY, BAS и файлов group_1, rate_29, и размещения архива bas.rar в каталоге BAS на диске A: выполните следующее:

- в окне архиватора откройте корневую папку своего электронного носителя;
- выделите указанные папки и файлы, щелкните на пиктограмме Добавить файлы в архив;
- укажите имя файла A:\BAS\bas.rar и щелкните на кнопке ОК.

12. Создание непрерывного архива

Для создания непрерывного архива повторите пункт 8.4.11 с дополнением пункта "Создать непрерывный архив" в разделе "Параметры архивации".

13. Выполнение некоторых операций над выделенным архивным файлом

13.1 Протестируйте архив, выполнив команду "Команды\Протестировать файлы в архиве".

13.2 Просмотрите информацию об архиве, выполнив команду "Операции\Показать информацию".

13.3 Преобразуйте обычный архив в самораспаковывающийся. Для выполнения преобразования выберите команду

"Операции\Преобразовать архив(ы) в SFX".

13.4 Переименуйте архив с помощью команды "Команды\Переименовать файл".

13.5 Удалите архив, применив команду "Команды\Удалить файл".

Оптимизация скорости и производительности диска

1. Порядок выполнения работы

1.1. Фиксация текущих характеристик диска

Установить в файле config.sys количество буферов DOS равным 1 и выполнить перезагрузку ПЭВМ.

Создать командные файлы измерения времени работы с диском. Например, с помощью команды "time%1" при выполнении операции копирования группы эталонных файлов с исследуемого диска на тот же самый диск с последующим удалением вновь созданных копий. Измерить с помощью данных командных файлов время работы с диском для двух групп эталонных файлов. Первая группа должна включать несколько (3-4) больших файлов суммарной длиной не менее 500 Кбайт, а вторая группа несколько (6-8) коротких файлов.

По указаниям преподавателя выбрать для работы жесткий диск. Отметить его емкостные и временные параметры. С этой целью с помощью пункта меню "Информация" утилиты sd.exe вывести на принтер информацию о статистике диска.

Примечание: Если фрагментированных файлов нет на диске, добиться их появления.

1.2. Команда Fastopen

Включить в файл autoexec.bat (после команды PATH) команду fastopen n:, где "n:"-имя оптимизируемого диска. Перезагрузить DOS или запустить на выполнение файл autoexec.bat. Выполнить три раза подряд измерение времени работы с диском. Проанализировать полученные результаты. Удалить из файла autoexec.bat команду fastopen. Перезагрузить DOS.

Команда Fastopen.

Дополнительно зафиксировать размер доступной оперативной памяти (команда mem).

Установить в файле config.sys количество буферов DOS равным 40 и вновь повторить команду mem.

Установить количество буферов DOS равным 80 и вновь повторить команду mem.

Кэширование диска. Организация кэша для уменьшения времени обращения к диску возможна только при наличии в структуре ПЭВМ дополнительной или расширенной оперативной памяти.

Включить в файл config.sys драйвер кэша:

DEVICE="путь\"smartdrv.sys"размер",

где "путь\"-путь доступа к драйверу;

"размер"-объем памяти, выделенный кэшу в килобайтах.

Если в файле config.sys определяется буфера DOS, -установить количество буферов равным 1.

Последовательно задать размеры кэша 128 и 256 Кбайт, каждый раз, выполняя перезагрузку ОС. Для каждого размера кэша измерить время работы диска с помощью эталонных файлов и размер доступной оперативной памяти. Проанализировать полученные результаты. Как правило, программа Smartdrv.exe задаётся только в DOS, в WINDOWS 95 не следует запускать её, так как WINDOWS сам кэширует запись и чтение на диске.

Проанализировать полученные результаты.

Удалить все файлы и каталоги, созданные для выполнения работы.

1.3. Утилита Speed Disk

Запустить утилиту sd.exe и выбрать диск для оптимизации. Рабочее окно Speed Disk показано на рисунке 4.

Через пункт меню «Properties» перейти в пункт меню «Оптимизация» и выбрать в качестве метода оптимизации дефрагментацию свободного дискового пространства. Выполнить оптимизацию. Вывести на принтер информацию о статистике диска и измерить время работы с диском.

Примечание. Дефрагментация возможна только при наличии на диске достаточно свободного пространства.

Выбрать в качестве метода оптимизации только дефрагментацию файлов выполнить ее и зафиксировать характеристики диска.

Выбрать в качестве метода оптимизации полную оптимизацию диска выполнить ее и зафиксировать характеристики диска.

С помощью пункта меню «Конфигурация» выполнить упорядочение каталогов на диске таким образом, чтобы на первых местах оказались каталоги, из которых и в которые производится копирование эталонных файлов. Сохранить эти установки на диске. Выбрать в качестве метода оптимизации оптимизацию каталогов, выполнить ее и зафиксировать время работы с эталонными файлами. Затем выполнить полную оптимизацию диска и вновь и вновь измерить время работы с эталонными файлами.

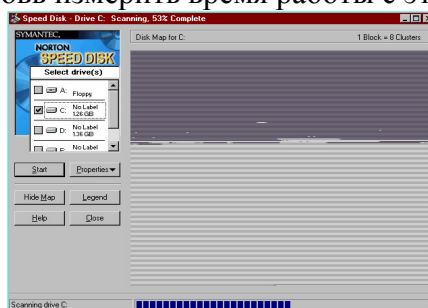


Рисунок 4 - Утилита Speeddisk

С помощью пункта меню «Конфигурация» переместить в начало диска файлы, используемые в качестве эталонных. Выполнить полную оптимизацию диска и вновь измерить время работы с эталонными файлами.

Проанализировать полученные результаты.

Восстановление данных на диске

1. Порядок выполнения работы

1.1. Быстрое восстановление данных сразу после стирания

При установленных Norton Utilities 3.0 в памяти находится программа, которая позволяет не удалять файлы физически, а помещать их корзину для возможного последующего восстановления. Файлы не удаляются физически не только из Windows 95 среды, но из DOS сессии.

Необходимо скопировать на диск в раздел пользователя несколько текстовых файлов, имеющих в распоряжении (например, read.mi из пакета NU, readme.txt из пакета DOS, autoexec.bat, config.sys), или создать их заново каким-либо текстовым редактором. С помощью утилиты de.exe проанализировать каталог пользователя, ТРФ и вывести их на печать. Удалить часть имеющихся текстовых файлов командой DOS del или средствами имеющейся программной оболочки. Вновь утилитой de.exe просмотреть каталог, ТРФ и скопировать их на принтер. Запустить утилиту unerase.exe и восстановить удаленные файлы. Вновь проанализировать каталог, ТРФ и вывести их печать. Либо запустить Norton Unease для восстановления файлов.

Примечание. Между процедурами удаления и восстановления никакие операции с диском выполняться не должны. Это касается стандартных методов восстановления, при использовании утилит это правило можно опустить, так как файл невозможно физически затереть, потому, что он находится в корзине, если удаление производилось из среды Windows 95. Если нет, то восстановление производится обычным образом.

1.2. Частичное восстановление данных после выполнения операций записи на диск

С помощью утилиты de.exe зафиксировать исходное состояние каталога пользователя, ТРФ и проанализировать их. Удалить часть файлов и записать в тот же каталог любой файл длиной больше каждого из удаленных, но меньше их суммарной длины. Вновь просмотреть каталог, ТРФ и вывести их на печать. Запустить утилиту unerase и попытаться восстановить с ее помощью часть удаленных файлов. Рабочее окно unerase показано на рисунке 2. Не восстановленный файл необходимо частично восстановить путем копирования принадлежащих ему кластеров в файл с другим именем. Просмотр и отбор кластеров можно выполнять с помощью утилиты de.exe. Зафиксировать длину вновь созданного файла. Частично восстановленный файл просмотреть текстовым редактором и удалить лишние записи. Конечное содержимое каталога и ТРФ также вывести на печать.

1.3. Восстановление данных из файлов с перекрестными связями при наличии “висячих” кластеров

Потеря данных может произойти при внезапном отключении питания или случайном запуске процедуры перезагрузки во время выполнения операций записи. Для выполнения данного пункта необходимо искусственно вызвать указанную ситуацию, нажав комбинацию клавиш Ctrl+alt+del (отключение питания при этом недопустимо). Потеря данных обнаруживается в этом случае двумя путями. Во-первых, файл записан в каталоге, но имеет нулевую длину. Во-вторых, имя файла в каталоге вообще отсутствует. Для восстановления данных необходимо, прежде всего, с помощью утилит chkdsk.exe DOS или Norton Disk Doctor (ndd.exe) выявить наличие файлов с перекрестными связями или “висячими” кластеров. Рабочее окно Norton Disk Doctor показано на рисунке 3. Если такие файлы обнаружены, то, прежде всего, необходимо утилитой de.exe просмотреть их, отыскав полностью сохраненные файлы, перезаписать их в тот же каталог с новым именем, а старые удалить. Частично искаженные файлы необходимо освободить от “чужих” записей, переписать их под новыми именами, а старые удалить. Затем из цепочек “висячих” кластеров (их может быть несколько) отыскать продолжение нужных файлов и попытаться с помощью какого-либо текстового редактора синтезировать нужный файл в целом.

Если файл с перекрестными связями или “висячие” кластеры не обнаружены, но файл в каталоге имеет нулевую длину, то необходимо просмотреть ТРФ и попытаться найти среди кластеров, помеченные как свободные, записи искомого файла. Если таковые обнаружены, то файл может быть полностью восстановлен (если он не фрагментирован) путем копирования соответствующих кластеров в новый файл.

По данному пункту необходимо представить все распечатки, поясняющие основные этапы восстановления данных.

Удалить все файлы и каталоги, созданные для выполнения работы.

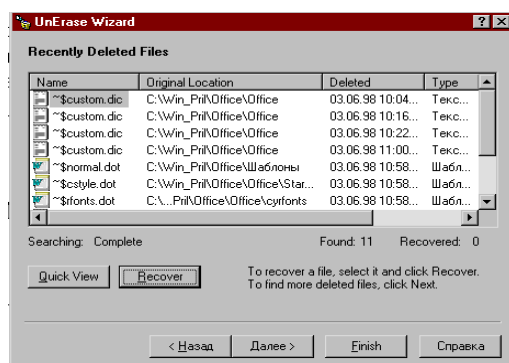


Рисунок 2 – Файлы, сохраненные в корзине

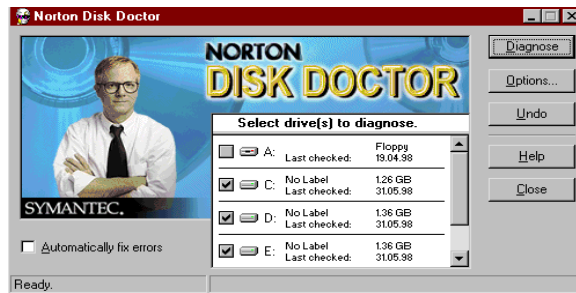


Рисунок 3 - Список логических дисков для проверки

2. Контрольные вопросы

1. Что такое сжатие данных и программ? На чем основаны методы сжатия?
2. Что такое саморазархивирующийся файл? Его назначение.
3. Что такое многотомный архив?
4. Каковы критерии выбора программы-архиватора?
5. Как выполняются операции чтения/записи файлов на диск в MS DOS?
6. Какой файл называется фрагментированным?
7. Каким образом возникает фрагментация файлов?
8. Как можно дефрагментировать файлы и что дает дефрагментация с точки зрения оптимизации работы с диском?
9. Как влияет команда fastopen на время работы с диском?
10. Что такое кэширование диска? Как влияет организация кэша на время работы с диском?
11. Что такое буфера DOS? Как влияет организация буферов DOS в оперативной памяти на время работы с диском?
12. Как взаимодействуют между собой буфера DOS и кэш при обращениях к диску?
13. Что такое коэффициент чередования секторов диска? Каким образом он может быть изменен?
14. Каким образом происходит удаление файлов в MS DOS?
15. Каким путем утилита unerase.exe восстанавливает файлы? Как используется ТРФ?
16. Возможно ли восстановление фрагментированных файлов?
17. Что такое “висячие” кластеры? В каких случаях они появляются на диске и как они могут быть обнаружены?
18. Что такое файлы с перекрестными связями? В каких случаях они появляются на диске и как они могут быть обнаружены?
19. Какие файлы могут быть восстановлены, а какие нет?
20. В каких случаях возможно частичное восстановление файлов?

2.4 Лабораторная работа № 4, 5, 6, 7 (8 часов)

Тема: «Алгоритмизация вычислительных процессов»

Лабораторная работа № 4 Разработка алгоритмов линейной структуры

2.4.1 Цель работы: выработать умение по созданию линейных алгоритмов и блок-схем к ним.

2.4.2 Задачи работы:

1. Решить задачи на линейный алгоритм разной степени сложности.
2. Для закрепления и проверки полученных навыков, необходимо ответить на контрольные вопросы и выполнить дополнительные задания. Результаты работы продемонстрировать преподавателю.

2.4.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Методические указания для выполнения лабораторной работы

2.4.4 Описание (ход) работы:

1. Теоретическая часть

Линейным называется **вычислительный процесс**, в котором предусматривается получение результата путем однократного выполнения последовательности действий при любых значениях исходных данных. Характерной особенностью линейного вычислительного процесса является то, что направление вычислений не зависит от исходных данных и промежуточных результатов.

Алгоритм линейного вычислительного процесса графически может быть представлен **блоком следования – композицией** (объединением) нескольких следующих друг за другом блоков ПРОЦЕСС (рис. 1).



Рисунок 1- Блок-схема линейного алгоритма

Линии, соединяющие отдельные блоки на блок-схеме алгоритма и указывающие на последовательность действий, называются **линиями потока**. Направление линии потока сверху вниз принимается за основное и стрелкой не обозначается.

Рассмотрим пример задачи, решение которой представляет собой линейный вычислительный процесс, и составим блок-схему алгоритма.

Пример: Найдите площадь круга S при заданном значении радиуса R . Для вычисления площади использовать формулу $S=3,14 \cdot R^2$.

Алг Решение (вещ R, S)

Арг R

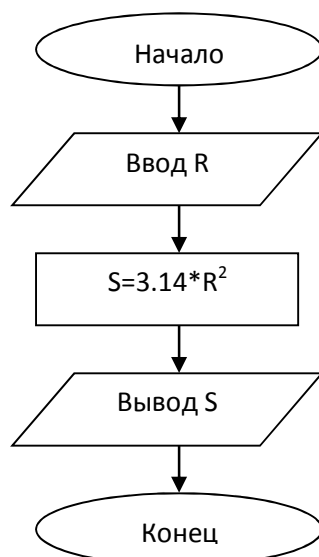
Рез S

Нач

$S=3.14 \cdot R^2$

Кон

Блок-схема



2. Порядок выполнения работы

1. Вспомните определение линейного алгоритма.
2. Вспомните, какие блоки Вам понадобятся для разработки блок-схемы для линейного алгоритма.

I уровень

Проанализируйте алгоритм решенной задачи и самостоятельно разработайте к нему блок-схему.

Задача 1. Найдите площадь и диагональ прямоугольника. Для вычисления диагонали использовать формулу $D=\text{SQR}(a^2+b^2)$.

Алг Решение (вещ a, b, P, S, D)

Арг a, b

Рез P, S, D

Нач

$P:=2 \cdot (a+b)$

$S:=a \cdot b$

$D:=\text{SQR}(a^2+b^2)$

Кон

II уровень

Самостоятельно разработайте алгоритм и блок-схему к задаче 2.

Задача 2. Стороны параллелограмма a и b , угол между ними x . Определите периметр, площадь, диагонали параллелограмма. Для определения диагоналей параллелограмма использовать формулу $D=\text{SQR}(a^2+b^2+(-)2ab\cos(x))$.

III уровень

Разработайте алгоритм и блок-схему к задаче 3.

Задача 3.

Длины сторон треугольника равны a, b, c . Найдите длины высот треугольника H_1, H_2, H_3 . Для определения высот треугольника использовать формулу $H_a = 2S/a$.

Разработайте алгоритм и блок-схему к задаче 4.

Задача 4.

Длины сторон треугольника a, b, c . Найдите площадь треугольника, радиус вписанной и радиус описанной окружности. Радиус вписанной окружности определяется по формуле $R = S/p$, а радиус описанной окружности - $R_o = abc/4S$.

3. Контрольные вопросы и задания

1. Что называют алгоритмом?
2. Что называют системой команд исполнителя алгоритма?
3. Какую форму записи алгоритма называют словесной?
4. Какой алгоритм называют линейным?

Задачи для самостоятельной работы

1. Даны два числа a и b . Получить их сумму, разность и произведение.
2. Даны действительные числа x и y . Получить $(|x| - |y|) / (1 + |x \cdot y|)$.
3. Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.
4. Даны два действительных положительных числа. Найти среднее арифметическое и среднее геометрическое этих чисел.

Лабораторная работа № 5 Разработка алгоритмов ветвящейся структуры

2.4.1 Цель работы: выработать умение по созданию разветвляющихся алгоритмов и блок-схем к ним.

2.4.2 Задачи работы:

1. Решить задачи на разветвляющийся алгоритм разной степени сложности.
2. Для закрепления и проверки полученных навыков, необходимо ответить на контрольные вопросы и выполнить дополнительные задания. Результаты работы продемонстрировать преподавателю.

2.4.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Методические указания для выполнения лабораторной работы

2.4.4 Описание (ход) работы:

1. Теоретическая часть

Разветвляющимся вычислительным процессом называется процесс, направление вычислений в котором зависит от результата проверки некоторого условия (условий).

Алгоритм разветвляющегося вычислительного процесса предусматривает выбор одной из нескольких возможных альтернатив (последовательностей действий) в зависимости от значения исходных данных или промежуточных результатов. Каждую из этих последовательностей называют ветвью алгоритма. Блок-схема алгоритма разветвляющегося вычислительного процесса содержит, по крайней мере, один блок РЕШЕНИЕ. Направления линий потока сверху вниз и слева направо на блок-схеме принимаются за основные и стрелками не обозначаются.

Графической интерпретацией алгоритма разветвляющегося вычислительного процесса является блок разветвления алгоритма, в котором может быть предусмотрен полный (рис. 1) или неполный (рис. 2) выбор.

В блоке разветвления алгоритма с полным выбором в зависимости от результата проверки условия выполняются только действия ветви "да" (т. е., действия 1 и 2) или только действия ветви "нет" (действия 3 и 4). В блоке с неполным выбором в зависимости от результата проверки условия либо выполняются действия какой-либо ветви, либо они игнорируются.

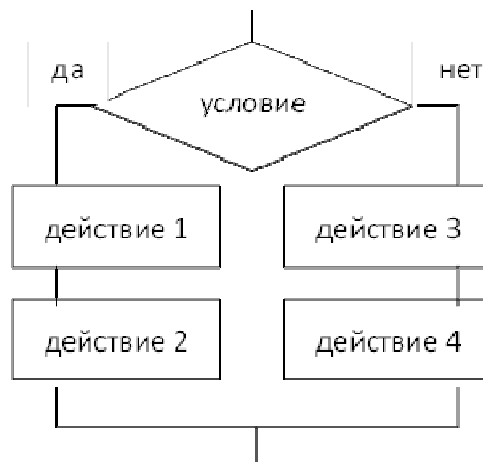


Рисунок 1 – Блок разветвления с полным выбором

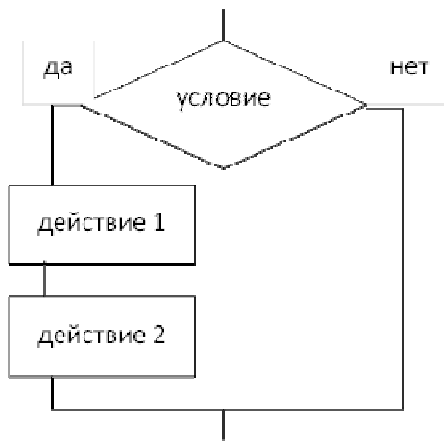


Рисунок 2 – Блок разветвления с неполным выбором

Некоторой разновидностью блока разветвления алгоритма является **блок множественного выбора** (рис. 3). В нем, в зависимости от результатов выбора, выполняется одно из предусмотренных действий.

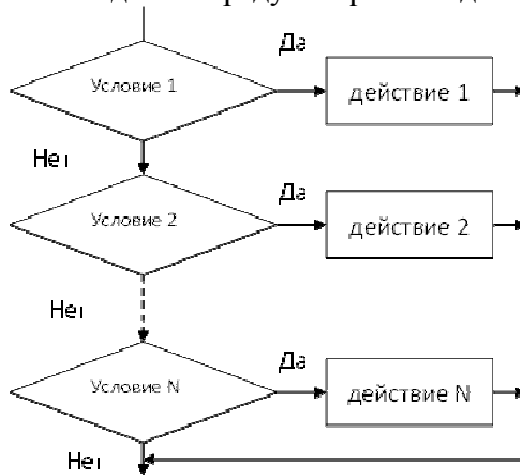


Рисунок 3 – Блок множественного выбора

Рассмотрим пример задачи, алгоритм решения которой носит разветвляющийся характер.

Пример 1. Разработать алгоритм выбирающий максимальное из двух чисел x и y . Присвоить его значение переменной **max**.

Алг Присваивание (**вещ** max)

Арг x, y

Рез max

Нач

Если $x > y$

То max = x

Иначе max = y

Все

Кон

Блок-схема алгоритма решения задачи на ЭВМ представлена на рис. 4.

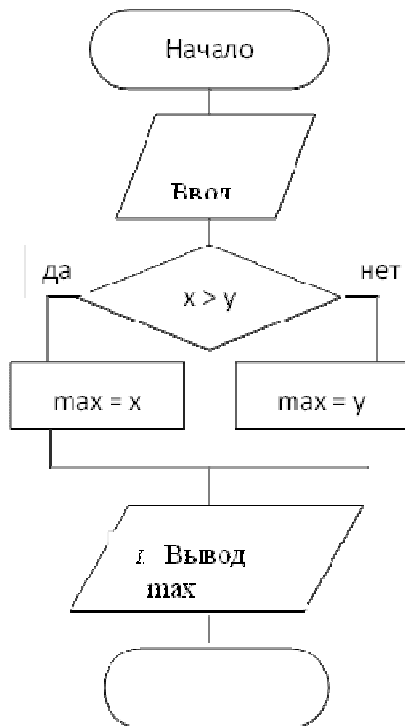


Рисунок 4 – Блок-схема алгоритма

В разветвляющихся вычислительных процессах могут иметь место так называемые **"пустые" ветви**. В направлении пустых ветвей задача, как правило, не имеет решения (вычислительный процесс не приводит к получению какого-либо промежуточного или конечного результата). Рассмотрим следующий пример.

Пример 2:

Алг Присваивание (**веш** max)

Арг x, y

Рез max

Нач

max = x

Если max < y

То max = y

Иначе «пусто»

Все

Кон

Блок-схема алгоритма решения задачи на ЭВМ представлена на рис. 5.

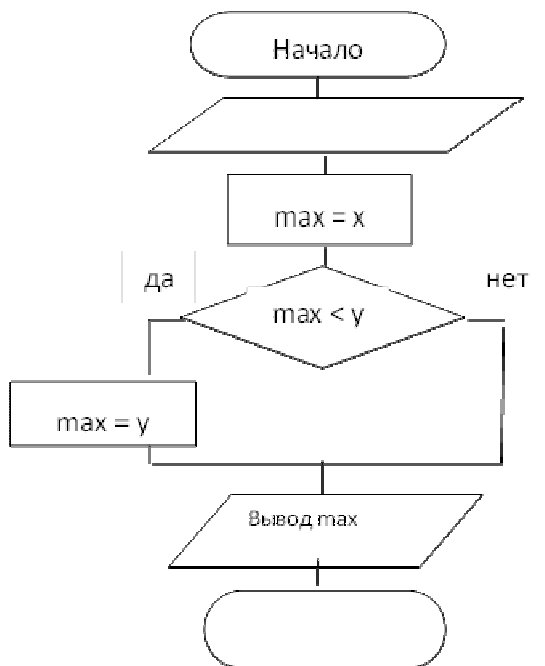


Рисунок 5 – Блок-схема алгоритма

2. Порядок выполнения работы

I уровень

4. Вспомните, как составляются линейные алгоритмы.

5. Подумайте, чем отличаются разветвляющиеся алгоритмы от линейных.

6. Вспомните, какие блоки Вам понадобятся для разработки блок-схемы для разветвляющегося алгоритма.

Проанализируйте алгоритм решенной задачи и самостоятельно разработайте к нему блок-схему.

Задача 1. Увеличьте число x на 1, если оно положительное, во всех остальных случаях уменьшите его на 1.

Алг Изменение (вещ x)

Арг x

Рез x

Нач

Если $x > 0$

То $x := x + 1$

Иначе $x := x - 1$

Все

Кон

II уровень

Самостоятельно разработайте алгоритм и блок-схему к задаче 2.

Задача 2. Вычислите функцию

$$Y = \begin{cases} x \sin(1/x), & \text{если } x \neq 0, \\ 0, & \text{если } x = 0 \end{cases}$$

III уровень

Разработайте алгоритм и блок-схему к задаче 3.

Задача 3.

Вычислите функцию

$$S = \begin{cases} 100+x, & \text{если } x < 1 \\ ax+50, & \text{если } x = 1 \\ x-50a, & \text{если } x > 1 \end{cases}$$

3. Контрольные вопросы и задания

1. Что называют алгоритмом?
2. Как выполняется команда присваивания?
3. Приведите общий вид команды присваивания?
4. Как в алгоритме осуществляется проверка условия?

Задачи для самостоятельной работы

1. Даны два действительных числа x и y . Если x и y отрицательные, то каждое значение заменить его модулем; если отрицательное только одно из них, то x и y увеличить на 0.5; если оба неотрицательные, то x и y увеличить в 10 раз.
2. Даны два действительных числа z и x . Возвести в квадрат те из них, значение которых отрицательны. Неотрицательные значения оставить без изменения.
3. Известны время старта и время финиша двух лыжников. Если один из лыжников выигрывает, то он получает поощрительный приз и слова поздравления. А проигравший лыжник ничего не выигрывает.
4. Даны два действительных числа a и b . Если числа не равны, то заменить каждое из них числами равными $(a+b)$ и $(a-b)$ соответственно, а если равны, то заменить их нулями.

Лабораторная работа № 6 Разработка алгоритмов циклической структуры

2.4.1 Цель работы: выработать умение по созданию циклических алгоритмов с и блок-схем к ним.

2.4.2 Задачи работы:

1. Решить задачи на циклический алгоритм разной степени сложности.
2. Для закрепления и проверки полученных навыков, необходимо ответить на контрольные вопросы и выполнить дополнительные задания. Результаты работы продемонстрировать преподавателю.

2.4.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Методические указания для выполнения лабораторной работы

2.4.4 Описание (ход) работы:

Разработка циклических алгоритмов с параметром

1. Теоретическая часть

Циклическим называется вычислительный процесс, в котором получение результата обеспечивается путем **многократного повторения некоторой последовательности действий**.

Графической интерпретацией алгоритма циклического вычислительного процесса является **блок цикла**. Различают несколько разновидностей блока цикла: **блок цикла с параметром**, **блок цикла с предварительным условием** и **блок цикла с последующим условием**.

Блок-схема **блока цикла с параметром** представлена на рис. 1.

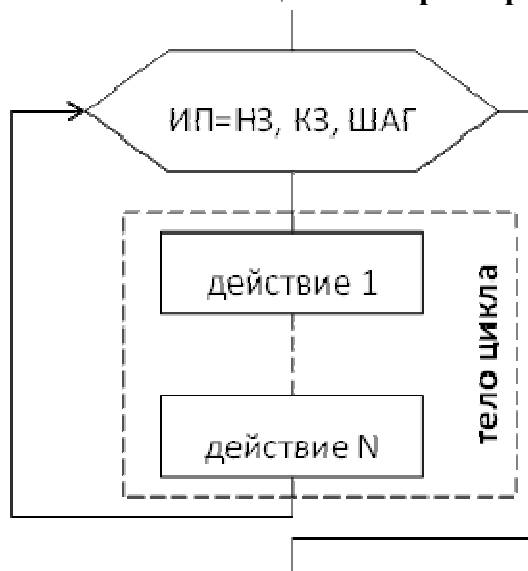


Рисунок 1 – Блок цикла с параметром

На рис. 1 приняты следующие сокращения:

ИП – имя ячейки памяти, в которую заносится значение параметра;

НЗ – начальное значение параметра;

КЗ – конечное значение параметра;

ШАГ – величина приращения параметра после каждого выполнения тела цикла.

Тело цикла представляет собой линейный вычислительный процесс и выполняется столько раз, сколько разных значений примет параметр в заданных пределах от НЗ до КЗ.

Блок цикла с параметром относится к циклу с **явно выраженным числом повторений** (число повторений известно заранее). Для таких циклов характерным является то, что задаются:

начальное и конечное значения параметра цикла;

закон изменения параметра цикла при каждом повторном выполнении тела цикла;

количество повторных **выполнений** тела цикла (вытекает из первых двух пунктов).

Рассмотрим пример алгоритмизации циклического вычислительного процесса.

Пример: Разработать алгоритм расчета значения функции по формуле $y = (a + b)^2$, при значениях **a** из интервала $[-5, 5]$ с шагом +1

Алг Решение(вещ y, a, b)

Арг a, b

Рез y

Нач нат **a**

Для a **от** -5 **до** 5

Нц

$y = (a + b)^2$

Кц

Кон

Блок-схема алгоритма решения задачи на ЭВМ представлена на рис. 2.

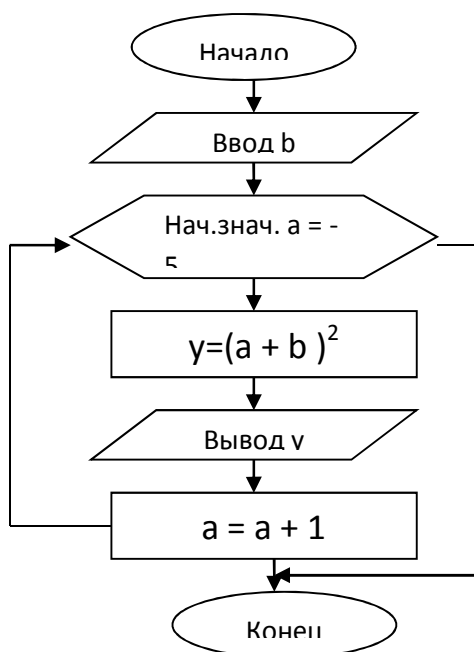


Рисунок 2 – Блок-схема алгоритма

2. Порядок выполнения работы

I уровень

1. Вспомните, что называется циклическим алгоритмом.
2. Подумайте, когда используется цикл с параметром.
3. Вспомните, какие блоки Вам понадобятся для разработки блок-схемы для составления блок-схемы к циклическому алгоритму с параметром.

Проанализируйте ход составления алгоритма задачи 1.

Задача 1. В 1626г индейцы продали остров Манхеттен за 20 долларов. Если бы эти деньги были помещены в банк на текущий счет, и ежегодный прирост составлял 4%, какова была бы стоимость капитала в 1988 г.?

Алг Капитал (вещ K)

Арг К

Рез К

Нач нат i

A:=20

Для i от 1 до 362

Нц

K:=K+0,04*K

Кц

Кон

Самостоятельно разработайте блок-схему к данному алгоритму.

II уровень

Самостоятельно разработайте алгоритм и блок-схему к задаче 2 и 3.

Задача 2. Вычислите сумму $S=1+1/2+1/3+\dots+1/n$.

В тексте алгоритма примите $n=10$.

Задача 3. Вычислите сумму $S=(n+1)^2+(n+2)^2+\dots+(n+n)^2$, где n задано равным 10 и может принимать целые значения из отрезка $[1,1000]$.

III уровень

Разработайте алгоритм и блок-схему к задаче 4.

Задача 4.

Вычислите функцию

Ax^2 , если $a<0$

$S=10x-b$ если $a\geq 0$

$x \in [-3;10]$ шаг 1

Разработка циклических алгоритмов с предусловием

1. Теоретическая часть

Блок цикла с предварительным условием и блок цикла с последующим условием относятся к так называемым **итерационным циклам**. В таких циклических вычислительных процессах число повторений тела цикла заранее не известно. Выход из цикла осуществляется не после того, как цикл повторится заданное число раз, а при выполнении определенного условия, связанного с проверкой значения монотонно изменяющейся в теле цикла величины. Блок-схема блока цикла с предварительным условием представлена на рис. 1, а блока цикла с последующим условием – на рис. 2.

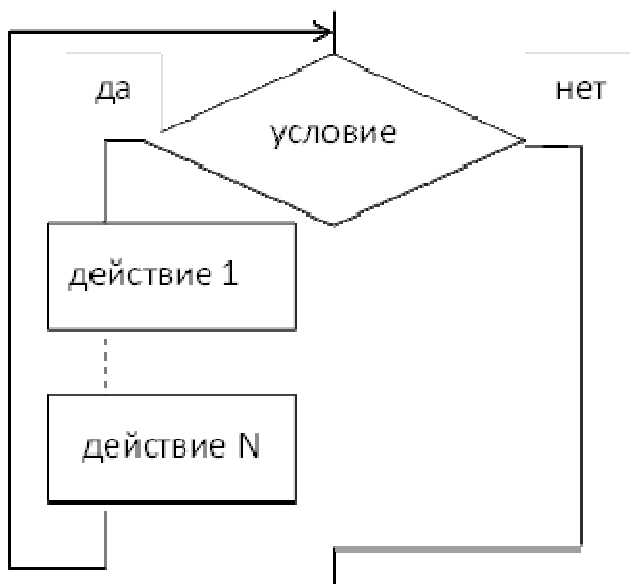


Рисунок 1 – Блок цикла с предварительным условием

Кратко суть алгоритма цикла с предварительным условием можно изложить следующим образом: **пока выполняется условие – повторять действия**. В таких циклах возможны ситуации, когда тело цикла не выполняется ни разу (например, если при первой же проверке не выполняется условие, то сразу происходит выход из цикла).

В цикле с последующим условием (рис. 2) тело цикла выполняется не менее одного раза. При этом **действия**, предусмотренные в теле цикла, **выполняются до тех пор, пока не выполнится заданное условие**.

Рассмотренные блоки циклов позволяют описать **простые** циклические вычислительные процессы. При решении сложных задач может возникнуть необходимость внутри одного цикла организовать дополнительно один или несколько циклов. Такие циклы называются **вложенными**. При этом цикл, внутри которого создается другой цикл, называется **внешним**, а цикл, создаваемый внутри другого – **внутренним**. Правила организации как внешнего, так и внутреннего циклов те же, что и для простых циклов. Параметры внешнего и внутреннего циклов должны быть разными.

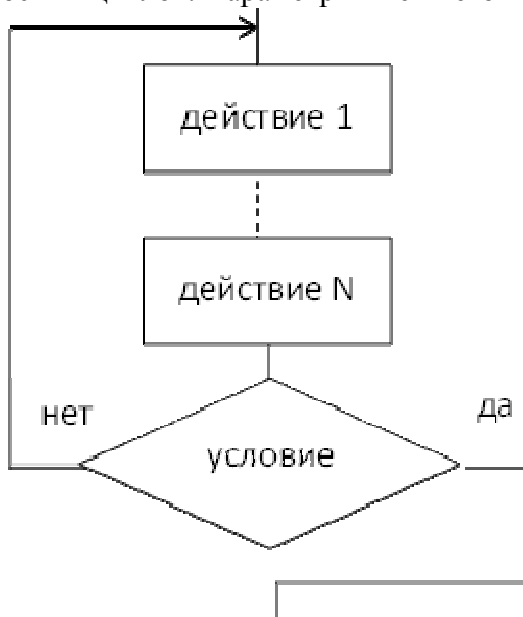


Рисунок 11 - Блок цикла с последующим условием

2. Порядок выполнения работы

I уровень

1. Вспомните, что собой представляют алгоритмы с предусловием.
2. Подумайте, какое условие является простым, а какое – составным.
3. Вспомните, какая геометрическая фигура блок-схемы определяет проверку условия.

Проанализируйте алгоритм решенной задачи и самостоятельно разработайте к нему блок-схему.

Задача 1. Сумма 10 000 рублей положена в банк, при этом прирост составляет 3% год и считается непрерывным. Определите, через какой промежуток времени первоначальная сумма увеличится в два раза.

Алг Прирост (вещ n,S)

Арг n

Рез S

Нач

n:=1

S:=0

Пока n<=10

Нц

S:=S+1/n

n:=n+1

Кц

Кон

II уровень

Разработайте алгоритм и блок-схему к задаче 2 и 3.

Задача 2.

Вычислите сумму $S=1+1/2+1/3+1/4+1/5+1/6+1/7+1/8+1/9+1/10$.

Задача 3.

$$X = \frac{(n^2 + n - 1)}{(n^2 + n + 1)}$$

Найдите номер первого члена последовательности, который отличается от 1 менее чем на i.

III уровень

Самостоятельно разработайте алгоритм и блок-схему к задаче 4.

Задача 4.

Сколько слагаемых должно быть в сумме $1+1/2+1/3+1/4+...+1/n$, чтобы эта сумма оказалась больше 5.

Разработка циклических алгоритмов с постусловием

1. Порядок выполнения работы

1 уровень

1. Вспомните, что представляет собой циклический алгоритм с постусловием.
2. Подумайте, в чем принципиальное различие циклических алгоритмов с постусловием от алгоритмов с предусловием.

Проанализируйте ход составления алгоритма задачи 1.

Задача 1 Найдите произведение чисел $1*3*5*7*\dots*21$

Алг Произведение (Цел x, П)

Арг x

Рез П

Нач

П:=1

x:=1

Нц

П:=П* x

x:= x+2

До x>21

Кц

Кон

Самостоятельно разработайте к составленному алгоритму блок-схему.

II уровень

Разработайте алгоритм и блок-схему к задаче 2.

Задача 2.

Среди первых членов последовательности a_n найдите номер первого отрицательного числа, если $a_n = -100 - n - n^3$.

III уровень

Самостоятельно разработайте алгоритм и блок-схему к задаче 3.

Задача 3.

Вычислите функцию $S = 100 + x^3 y - x a$, если $5 \leq a \leq 25$

$x \in [7; 70]$ $h=2$

$y \in [1; 20]$ $h=3$

2. Контрольные вопросы и задания

1. Что собой представляет циклический алгоритм?
2. В каких случаях используется команда повторения Для?
3. Что в команде цикла Для является параметром.
4. Что называется шагом изменения параметра и как происходит это изменение.
5. Что представляет собой тело цикла.
6. Что представляет собой алгоритм с предусловием?
7. Опишите технологию накопления суммы в алгоритме.
8. Опишите технологию применения цикла с «пока».
9. Опишите технологию составления циклического алгоритма с постусловием.
10. Что собой представляют вложенные циклы.
11. Изобразите схематически структуру алгоритма с параметром, с постусловием и предусловием.

Задания

1. Вывести на экран таблицу умножения на 5.
2. Дано количество студентов. Ввести имена студентов и их год рождения. Определить, через сколько лет они получают право голоса или сделать вывод, что они имеют право голоса.

3. Рассчитать $y = \frac{a * b + 2}{4 * b}$ для известного количества любых значений a и b.

4. Рассчитать $Z = \frac{a + b * x}{b} + 1$, где x изменяется на заданном интервале с заданным шагом, a,b – любые значения.

Лабораторная работа № 7 Разработка алгоритмов по вычислению суммы бесконечного ряда с заданной точностью

2.4.1 Цель работы: выработать умение по созданию алгоритмов по вычислению суммы бесконечного ряда.

2.4.2 Задачи работы:

1. Решить задачи по вычислению суммы бесконечного ряда с заданной точностью.
2. Для закрепления и проверки полученных навыков, необходимо ответить на контрольные вопросы. Результаты работы продемонстрировать преподавателю.

2.4.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Методические указания для выполнения лабораторной работы

2.4.4 Описание (ход) работы:

1. Порядок выполнения работы

I уровень

Проанализируйте ход составления блок-схемы к задаче 1 и самостоятельно разработайте алгоритм к данной задаче.

Задача 1. Вычислите значение данной функции, так чтобы соблюдалось условие $1/n < 0,0001$.

$$S = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \dots + \frac{1}{n}$$

II уровень

Самостоятельно разработайте алгоритм и блок-схему к задаче 2.

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!}$$

Задача 2. Вычислите значение функции $y = \cos(x)$ по нижеприведенной формуле, пока очередное слагаемое не станет меньше 0,00001.

III уровень

Самостоятельно организуйте в **задаче 2** при составлении алгоритма цикл «ДО» и разработайте новую блок схему.

2. Контрольные вопросы

1. Чем отличается цикл «ДО» от цикла «ПОКА»?
2. В каких случаях организуется цикл «ДО» и «ПОКА» и в каких случаях «ДЛЯ»? Приведите примеры.

2.5 Лабораторная работа № 8, 9 (4 часа)

Тема: «Массивы. Обработка одномерных массивов»

2.5.1 Цель работы: получить навыки разработки алгоритмов для обработки одномерных массивов.

2.5.2 Задачи работы:

1. Решить задачи на обработку одномерных массивов.
2. Для закрепления и проверки полученных навыков, необходимо выполнить дополнительные задания. Результаты работы продемонстрировать преподавателю.

2.5.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Методические указания для выполнения лабораторной работы

2.5.4 Описание (ход) работы:

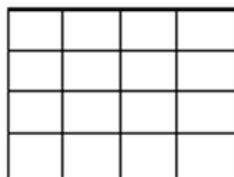
1. Теоретическая часть

Массивом называется упорядоченная совокупность однородных величин, обозначенных каждая одним и тем же именем с различными целочисленными индексами, изменяющимися по порядку.

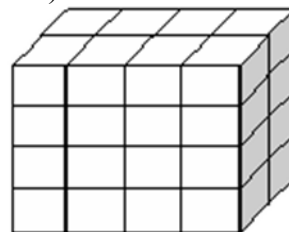
Массив - это набор однородных данных (чисел, символов, слов).



ОДНОМЕРНЫЙ МАССИВ



ДВУМЕРНЫЙ МАССИВ



ТРЕХМЕРНЫЙ МАССИВ

Рис.1 – Изображение одно-, двух- и трехмерных массивов

Одномерный массив

Одномерный массив можно рассматривать как список однотипных элементов. Например, список фамилий студентов группы – одномерный массив, численные данные о средней температуре за месяц - одномерный массив, буквы русского алфавита - одномерный массив.

2. Порядок выполнения работы

1. Изучить способы описания одномерных массивов.
2. Разработать схему алгоритма. Предусмотреть в алгоритме ввод значений элементов массива с клавиатуры. Исходный массив должен быть выведен на экран.

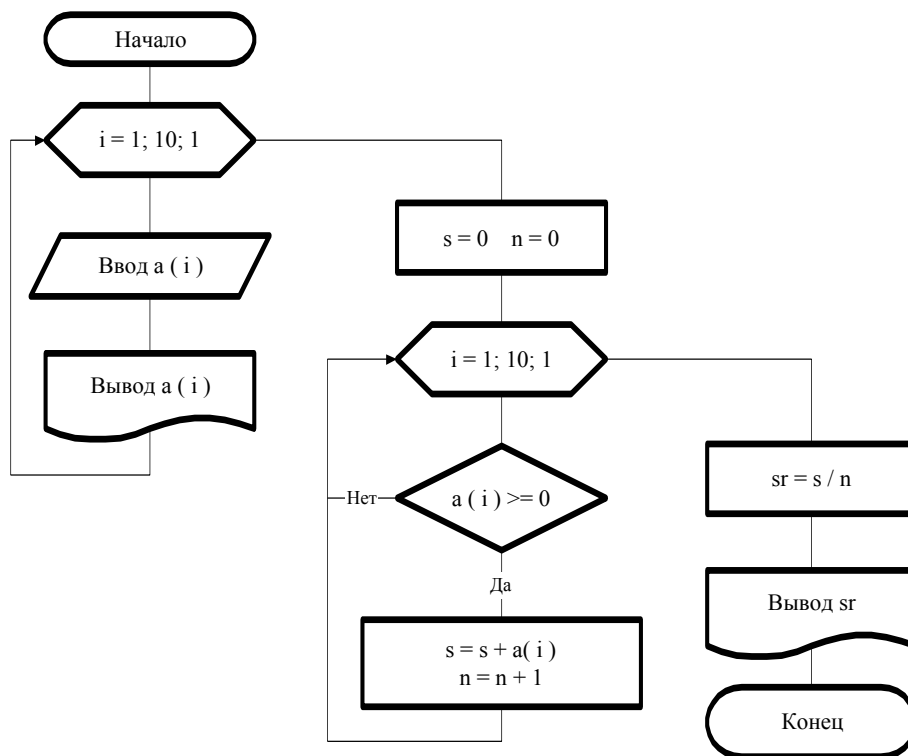
Отчет по лабораторной работе должен содержать:

1. Задание к лабораторной работе.
2. Схему алгоритма решения задачи.
3. Анализ структуры алгоритма.

Пример выполнения лабораторной работы

Задание. Дан массив A(10). Вычислить среднее значение неотрицательных элементов массива.

Схема алгоритма.



Анализ структуры алгоритма

$A(10) = 3, 5, 4, -7, -2, 4, -1; -10; -25; -3$

$i=1$, шаг = 1, конечное значение $i=10$

$a(1) = 3$

и т.д.

3. Контрольные вопросы и задания

1. Дан массив $A(10)$. Вычислить сумму элементов массива кратных трем и их количество.
2. Дан массив $A(10)$. Вычислить среднее значение положительных элементов массива и количество нулей.
3. Дан массив $A(10)$. Определить наибольший по модулю элемент массива и его номер.
4. Дан массив $A(10)$. Вычислить количество нулевых, отрицательных и положительных элементов массива отдельно.
5. Дан массив $A(10)$. Определить наименьший элемент массива среди положительных и его номер.
6. Дан массив $A(10)$. Вычислить количество нулевых элементов, сумму положительных элементов и среднее значение отрицательных элементов массива.
7. Дан массив $A(10)$. Вычислить среднее значение элементов массива, которые являются четными числами.
8. Дан массив $A(10)$. Вычислить суммы элементов массива, имеющих четные и нечетные индексы отдельно.
9. Дан массив $A(10)$. Вычислить наибольший элемент массива среди элементов с четными значениями и его номер.
10. Дана последовательность 10 чисел $a_1, \dots, a_{10}$. Вывести на экран символы данной последовательности в обратном порядке: $a_{10}, \dots, a_1$.

2.6 Лабораторная работа № 10, 11, 12 (6 часов)

Тема: «Основы программирования на алгоритмическом языке высокого уровня»

Лабораторная работа № 10 Создание программ на языке Turbo Pascal с использованием стандартных функций.

2.6.1 Цель работы: Знакомство со средой программирования Turbo Pascal. Изучение структуры программы, стандартных функций, оператора присваивания и процедур ввода-вывода.

2.6.2 Задачи работы:

1. Научиться создавать программы на языке Turbo Pascal с использованием стандартных функций.

2. Для закрепления и проверки полученных навыков, необходимо ответить на контрольные вопросы и выполнить дополнительные задания. Результаты работы продемонстрировать преподавателю.

Работа должна содержать:

- номер варианта для выполнения задания и условие своего варианта;
- блок-схемы решения задач;
- тексты программ;
- полученные при расчетах численные результаты;

2.6.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Методические указания для выполнения лабораторной работы

2.6.4 Описание (ход) работы:

1 Порядок выполнения работы

1. Загрузить среду программирования Turbo Pascal. Ознакомиться с назначением основных пунктов меню, содержанием строки состояния и работой в справочной системе.
2. Составить и отладить программу, включающую в себя:
 - сообщение о вводе двух чисел;
 - ввод значений двух вещественных чисел;
 - вывод в виде таблицы суммы, разности, произведения и среднеарифметического значения 1-го и 2-го числа с разным числом знаков в дробной части.
3. Составить и отладить программу вычисления функции $y=f(x)$ с использованием стандартных функций языка Turbo Pascal в соответствии с заданным преподавателем вариантом (Приложение А).
4. Провести расчеты для $x=0.1$, $x=0.2$, $x=0.3$, $x=0.4$ и $x=0.5$.
5. Записать полученные значения в виде таблицы.

Теоретические положения

Элементы языка

Алфавит языка

При записи программ разрешены символы:

- буквы латинского алфавита A-Z (в любом регистре), а также знак подчеркивания _;
- буквы русского алфавита А-Я;
- цифры 0-9;
- специальные символы $> < = + - / * [] () \{ \} . , : ; ^ @ ' \$ \#$
- пары символов (их нельзя разделять пробелами) $< > < = > = := (* *) (.)$
- пробелы (рассматриваются как ограничители идентификаторов, констант, чисел, зарезервированных слов).

Идентификаторы

Неделимые последовательности символов алфавита образуют слова -*идентификаторы*, используемые для обозначения констант, переменных, процедур, функций и т.д.

Идентификатор должен начинаться с буквы или символа подчеркивания, не должен содержать пробелов и специальных символов. Примеры идентификаторов:

name

WorkPhone

_SUM1

Константы

В качестве констант могут использоваться числа, логические константы, символы и строки символов.

Целые числа записываются со знаком или без него по обычным правилам и могут иметь значение от -2147483648 до +2147483647.

Вещественные числа записываются со знаком или без него с использованием десятичной точки и/или экспоненциальной части. Экспоненциальная часть начинается символом *e* или *E*, за которым могут следовать знаки «+» или «-» и десятичный порядок. Символ *e* (*E*) означает десятичный порядок и смысл «умножить на 10 в степени». Например, запись 3.14E5 означает $3,14 \cdot 10^5$, а запись -17e-2 – это $-17 \cdot 10^{-2}$.

Логическая константа – это либо слово FALSE (ложь) либо слово TRUE (истина).

Символьная константа – это любой символ, заключенный в апострофы:

'z' – символ z;

'ф' – символ ф.

Строковая константа – последовательность символов, заключенная в апострофы. Если в строке нужно указать сам символ апострофа, он удваивается, например:

'Это - строка символов ';

'That"s string'.

Строка символов может быть пустой, т.е. не иметь никаких символов в обрамляющих ее апострофах.

Выражения

Выражение задает порядок выполнения действий над элементами данных и состоит из операндов (констант, переменных, функций, круглых скобок и знаков операций).

Действия в выражении выполняются слева направо с соблюдением старшинства (в порядке убывания):

1) Not (логическое отрицание);

2) * (умножение), / (деление), div (целочисленное деление), mod (целочисленное деление с остатком по модулю), and (логическая операция “И”);

3) + (сложение), - (вычитание), or (логическая операция “ИЛИ”);

4) операции отношений: = (равно), <> (не равно), < (меньше), > (больше), < = (меньше или равно), > = (больше или равно).

Для изменения порядка выполнения действий используются круглые скобки. Число открывающихся скобок равно числу закрывающихся. Любое выражение в скобках вычисляется раньше, чем выполняется операция, предшествующая скобкам.

Все составные части выражения записываются в одну строку, например, выражение

$$\frac{a + bx}{c + d} \text{ запишется в виде } (a+b*x)/(c+d) .$$

В выражение могут входить функции. Наиболее часто употребляемые функции называют стандартными. Для работы с ними не надо ни заказывать библиотеку, ни описывать их предварительно в программе. Примеры стандартных математических функций:

ABS(x) – модуль x ($|x|$);

SQR(x) – квадрат числа x (x^2);

SQRT(x) – квадратный корень из x ($\sqrt{x}$);

$\text{LN}(x)$ – натуральный логарифм от x ($\ln x$);

$\text{EXP}(x)$ – e в степени x (e^x);

$\text{SIN}(x)$ – синус x ($\sin x$);

$\text{COS}(x)$ – косинус x ($\cos x$);

$\text{ARCTAN}(x)$ – арктангенс x ($\arctg x$).

Аргумент этих функций может быть как вещественным, так и целым. Результат – всегда вещественный.

Типы данных

Любые данные, т.е. константы, переменные, значения функций или выражения в Турбо Паскале характеризуются своими типами. Тип определяет множество допустимых значений, которые может иметь тот или иной объект, а также множество допустимых операций, которые применимы к нему. Все типы данных разделяются на две группы – простые и составные.

К простым (скалярным) типам относятся, например:

INTEGER - данные этого типа могут принимать только целые значения (положительные, отрицательные, 0) в диапазоне от -32768 до $+32767$;

REAL - величины этого типа могут принимать только вещественные значения (числа с дробной частью, целая часть от дробной отделяется точкой);

BOOLEAN - логический тип, принимает два значения TRUE (истина) и FALSE (ложь);

CHAR – символ.

Примером составного (структурированного типа) может служить тип STRING - строка символов. Этот тип широко используется для обработки текстов.

Структура программы

Структура программы должна быть такой:

<Заголовок программы>

{Блок описаний}

BEGIN

{Раздел исполняемых операторов}

END.

Заголовок программы

В заголовке указывается имя программы. Общий вид заголовка:

program n;

здесь n – имя программы.

Заголовок программы необязателен, его можно опускать без каких-либо последствий для программы.

Блок описаний

В блоке описаний объявляются идентификаторы типов, констант, переменных, а также метки, процедуры и функции. Блок описаний может состоять из пяти разделов, которые должны следовать в строго определенном порядке:

1) раздел меток (label);

2) раздел констант (const);

3) раздел типов (type);

4) раздел переменных (var);

5) раздел процедур и функций.

Раздел меток (label)

Любой выполняемый оператор может быть снабжен меткой – положительной константой, содержащей не более 4-х цифр. Метка отделяется от оператора двоеточием. Все метки, встречающиеся в программе, должны быть описаны в разделе label. Общий вид:

label l1, l2, l3...;

здесь $l1, l2, l3...$ - метки.

Пример.

label 20;

Пусть оператор $a:=b$; имеет метку 20. Тогда этот оператор выглядит так:

20: $a:=b$;

Раздел констант (const)

Если в программе используются константы, имеющие достаточно громоздкую запись (например, число π с 8-ю знаками), либо сменные константы (например, для задания варианта программы), то такие константы обычно обозначаются какими-либо именами и описываются в разделе const. Это делает программу более наглядной и удобной при отладке и внесении изменений.

Общий вид:

const a1 = c1; a2 = c2; ...

Здесь *a1, a2, ...* – имя константы, *c1, c2, ...* – значение константы.

Пример.

const pi=3.14; c=2.7531;

Раздел типов (type)

Если в программе вводится тип, отличный от стандартного, то этот тип описывается в разделе type:

type t1=<вид tuna>;

t2=<вид tuna>;

· · · · ·

где *t1* и *t2* – идентификаторы вводимых типов.

Пример.

Type color=(red, yellow, green, blue);

Здесь описан тип color, задаваемый перечислением значений.

Раздел переменных (var)

В разделе var вводится имя каждой переменной и указывается, к какому типу эта переменная принадлежит:

var v11, v12, ...: type1;

v21, v22, ...: type2; ...

Здесь *v11, v12, ...* - имена переменных; *type1* – тип переменных *v11, v12, ...*; *type2* - тип переменных *v21, v22, ...*

Пример.

var k,i,j:integer; a,b:real;

Раздел процедур и функций

Те алгоритмы, которые оформляются как подпрограммы (процедуры и функции) помещаются в главной программе после раздела var и перед begin программы.

Раздел действий (операторов).

Эта часть программы начинается с ключевого слова begin и заканчивается словом end, после которого должна стоять точка (end.). Раздел действий - это выполняемая часть программы, состоящая из операторов.

Комментарии

Комментарий – это произвольная последовательность любых символов, поясняющая текст программы. Комментарий разрешается вставлять в любое место программы, где по

смыслу может стоять пробел. В качестве ограничителей комментария используются фигурные скобки « { » и « } », а также пары символов: « (* » - слева от комментария и « *) » - справа от него:

```
{ Это комментарий }
(* Это тоже комментарий *)
```

Оператор присваивания

Под операторами в языке Паскаль подразумевают описание действий. Операторы отделяются друг от друга точкой с запятой. Если оператор стоит перед `end`, `until` или `else`, то в этом случае точка с запятой не ставится.

Общий вид оператора присваивания:

$v := a;$

здесь v – переменная, a – выражение, $:=$ - операция присваивания. Выражение a может содержать константы, переменные, названия функций, знаки операций и скобки. В операторе $v := a$ переменная v и выражение a должны иметь один и тот тип.

Примеры.

$f := 3 * c + 2 * \sin(x);$

$x := x + 1;$

Замечание. Разрешается присваивать переменной типа `real` выражение типа `integer`. Но нельзя присваивать переменной типа `integer` выражение типа `real`.

Процедура ввода информации

Общий вид:

Read ($v_1, v_2, \dots, v_n$);

или

Readln ($v_1, v_2, \dots, v_n$);

здесь $v_1, v_2, \dots, v_n$ – идентификаторы переменных.

Значения переменных вводятся с клавиатуры и должны соответствовать типам переменных. В случае использования процедуры *readln*, после ввода происходит переход на следующую строку.

Процедура вывода информации на печать

Общий вид оператора:

write ($p_1, p_2, \dots, p_n$);

или

writeln ($p_1, p_2, \dots, p_n$);

Здесь $p_1, p_2, \dots, p_n$ - список выражений, значения которых выводятся на печать.

Оператор *write* оставляет курсор в конце выведенной строки текста.

В случае использования процедуры *writeln*, после печати происходит переход на следующую строку.

Кроме значений выражений, на печать можно выводить и произвольный набор символов, заключенный в апострофы, например

writeln ('p=', p);

Этот оператор выполняется так: сначала выводятся символы, заключенные в апострофы. Затем выводится значение переменной p , например 13.5. На экране в результате работы оператора появится:

p=13.5

Пример 1. Вычислить длину окружности радиуса 5,785.

program t10;

(* Программа вычисления длины окружности *)

```

const r=5.785;
var l:real;
begin
l:=2*3.1416*r;
writeln(' l=',l);
end.

```

либо

```

program t11;
var r:real;
begin
readln(r);
writeln(' l=',2*3.1416*r);
end.

```

Имеется возможность задать ширину поля (число позиций) М для выводимой величины Р:
Write (P1:M1, P2:M2, ...PN:MN);

Для вещественных чисел можно задавать поля М и N, где М – общее число позиций, отводимых под все число, N –число позиций под его дробную часть.

Например,

Write (P:10:2);

Здесь под Р отводится 10 позиций, 2 из них под дробную часть.

Пример 2. Назначение следующей программы - ввести с клавиатуры два целых числа, найти результат деления первого числа на второе и вывести числа и полученный результат на экран в виде таблицы.

Program Input_Output;

{Программа вводит два целых числа
и выводит частное от деления 1-го на 2-е}

Var

n1,n2:Integer; {n1 и n2 - вводимые целые}

x:real; {x - результат}

Begin

Write(эn1=э); {Сообщение о вводе n1}

ReadLn(n1); {Ввод n1}

Write(эn2=э); {Сообщение о вводе n2}

ReadLn(n2); {Ввод n2}

X:=n1/n2; {Вычисление результата}

WriteLn(э-----э); {Печать таблицы}

WriteLn(э| n1 | n2 | Частное |э);

WriteLn(э-----э);

WriteLn(n1:8,n2:8,x:8:4);{Вывод n1, n2 и x}

WriteLn(э-----э);

End.

Методические рекомендации

1 Запуск программы Turbo Pascal выполняется любым из стандартных способов запуска, предусмотренных в ОС Windows (например, с помощью ярлыка на рабочем столе).

Для перехода к выбору команд главного меню используется клавиша **F10**. Для возврата в режим редактирования нужно нажать клавишу **ESC**.

Для получения справки используются клавиши:

F1 – получение контекстно-зависимой справки;

SHIFT+F1 – выбор справки из списка доступных справочных сообщений;

CTRL+F1 – получение справки о нужной стандартной процедуре, функции, о стандартной константе или переменной.

2 При решении задач можно воспользоваться примерами программ из п.5.7. Текст программы набирается в текстовом редакторе среды Turbo Pascal.

После заполнения очередной строки следует нажать клавишу **ENTER**, чтобы перевести курсор на следующую строку.

Наиболее часто используемые команды редактора Turbo Pascal:

- смещение курсора

Page Up – на страницу вверх;

Page Down – на страницу вниз;

Home – в начало текущей строки;

End – в конец текущей строки;

Ctrl+ Page Up – в начало текста;

Ctrl+ Page Down – в конец текста.

- команды редактирования

Backspace – стирает символ слева от курсора;

Delete - стирает символ, на который указывает курсор;

Ctrl+Y – стирает строку, в которой расположен курсор;

Enter – вставляет новую строку, разрезает старую;

Ctrl+Q L – восстанавливает измененную строку (действует, если курсор не покидал строку после ее изменения).

- работа с блоками

Ctrl+K B – начинает выделение блока;

Ctrl+K K – заканчивает выделение блока (кроме того, блок можно выделить с помощью мыши);

Ctrl+K Y – уничтожает выделенный блок;

Ctrl+K C – копирует блок;

Ctrl+K V – перемещает блок на новое место;

Ctrl+K W – записывает блок в файл;

Ctrl+K R – читает блок из файла;

Ctrl+K P – печатает блок.

Набранный текст программы запишите в файл. Клавишей **F2** вызывается окно диалога, в котором следует задать имя файла.

После подготовки текста программы нужно попытаться исполнить ее, т.е. откомпилировать программу, связать ее (если это необходимо) с библиотекой стандартных программ и функций, загрузить в оперативную память и передать ей управление. Эта последовательность действий – прогон программы – осуществляется после нажатия клавиш **CTRL+F9**.

Если в программе нет синтаксических ошибок, то все действия выполняются последовательно одно за другим, при этом в небольшом окне сообщается о количестве откомпилированных строк и объеме доступной памяти. Перед передачей управления загруженной программе среда выводит на экран окно прогона программы, а после завершения работы восстанавливает на экране окно редактора.

Если на каком-то этапе среда обнаружит ошибку, она прекращает дальнейшие действия, восстанавливает окно редактора и помещает курсор на ту строку программы, при компиляции или исполнении которой обнаружена ошибка. При этом в верхней строке редактора появляется диагностическое сообщение о причине ошибки. Все это позволяет отладить программу, т.е. устранить в ней синтаксические ошибки и убедиться в правильности ее работы. Некоторые сообщения об ошибках, их перевод и пояснения приведены в приложении Б.

По оператору **read** (или **readln**) вызывается встроенная процедура ввода данных и программа останавливается в ожидании ввода. Необходимо набрать на клавиатуре нужные данные и нажать клавишу **ENTER**.

С помощью клавиш **ALT+F5** в любой момент можно просмотреть данные, выданные на экран в результате прогона программы.

2. Контрольные вопросы

1. Как в программе на языке Turbo Pascal описываются переменные?
2. Какие бывают типы переменных?
3. Какой вид имеет оператор присваивания?
4. Каким символом отделяются друг от друга операторы в программе?
5. В каких случаях после оператора не ставятся точка с запятой?
6. Какая процедура служит для вывода информации на печать?
7. Какая процедура служит для ввода значений с клавиатуры?
8. Какие функции служат для вычисления квадрата, квадратного корня, модуля, экспоненты числа или числового выражения?
9. Какие стандартные тригонометрические функции существуют в языке Turbo Pascal?
10. Как в среде Turbo Pascal запустить программу на выполнение?

ПРИЛОЖЕНИЕ А

Номер варианта	Функция
1	$y = \sqrt{x^2 + 5x + 8}$
2	$y = \frac{3x^2 - 4}{x + 1}$
3	$y = x - \sqrt{x + 2}$
4	$y = \frac{7x + 4}{3x + 5} + \cos x$
5	$y = x^3 + 4x^2 + x - 2$
6	$y = x + \sin 3x$
7	$y = \cos x + \sqrt{x}$
8	$y = \frac{\sin x}{3} + \frac{\operatorname{tg} x}{6}$
9	$y = \sqrt{x + 2} - \sqrt{x + 3}$
10	$y = 2x + \sin 2x$
11	$y = 5 - \frac{\cos 3x}{3}$
12	$y = \sqrt{x} - \sqrt{x + 1}$
13	$y = x - 5 + x^2$
14	$y = e^x + 1$
15	$y = \sqrt{ \sin x }$

ПРИЛОЖЕНИЕ Б

Сообщения об ошибках.

- 2 Identifier expected (Не указан идентификатор).
- В этом месте должен находиться идентификатор.
- 3 Unknown identifier (Неизвестный идентификатор)

Этот идентификатор не был описан.

4 Duplicate identifier (Двойной идентификатор).

Попытка дважды описать один и тот же идентификатор.

5 Syntax error (Синтаксическая ошибка).

В исходном тексте найден недопустимый символ.

10 Unexpected end of file (Не найден конец файла).

Причины этого сообщения могут быть следующие:

- исходный файл закончился перед последним **END** основного раздела операторов; вероятно в программе неодинаковое количество операторов **BEGIN** и **END**;
- не закончен комментарий.

11 Line too long (Слишком длинная строка).

Максимальная длина строки, обрабатываемая компилятором, равна 126 символам.

12 Type identifier expected (Здесь нужен идентификатор типа).

Не указан тип идентификатора.

16 Disk full (Диск заполнен).

Нужно удалить некоторые файлы или воспользоваться новым диском..

20 Variable identifier expected (Отсутствует идентификатор переменной).

На этом месте должен быть идентификатор переменной.

21 Error in type (Ошибка в объявлении типа).

Объявление типа не может начинаться с этого символа.

26 Type mismatch (Несоответствие типа).

Это сообщение может быть вызвано следующими причинами:

- несовместимые типы переменной и выражения в операторе присваивания;
- тип выражения не совместим с типом индекса при объявлении массива;
- несовместимые типы операндов в выражении.

33 Label identifier expected (Нужен идентификатор метки)

Метка не обозначена с помощью идентификатора, как это требуется из контекста программы.

36 BEGIN expected (Нужен BEGIN)

37 END expected (Нужен END)

38 Integer expression expected (Нужно выражение типа Integer).

41 Operand types do not match operator (Типы операндов не соответствуют операции).

Данная операция не может быть применена к указанным операндам.

42 Error in expression (Ошибка в выражении)

Данный символ не может участвовать в выражении указанным образом. Возможно, не указана операция между двумя операндами.

50 DO expected (Нужен оператор DO)

57 THEN expected (Требуется THEN)

58 TO or DOWNT0 expected (Требуется TO или DOWNT0)

62 Division by zero (Деление на ноль)

Предшествующая операция пытается выполнить деление на ноль.

64 Cannot Read or Write variables of this type (Нет возможности считать или записать переменные данного типа).

Нарушены следующие ограничения:

- процедуры **READ** и **READLN** могут считывать переменные символьного, целого, действительного и строкового типов;
- процедуры **WRITE** и **Writeln** могут выводить переменные символьного, целого, действительного, логического и строкового типов.

76 Constant out of range (Константа нарушает границы).

Возможные причины сообщения:

- попытка указать индекс массива, выходящий за его границы;

- попытка присвоить переменной значение, выходящее за границы, допустимые для типа этой переменной.

79 Integer or real expression expected (Нужно выражение вещественного или целого типа).

81 Label already defined (Метка уже определена).

Данная метка уже помечает оператор.

85 «;» expected (Нужно указать «;»).

97 Invalid FOR control variable (Неправильный параметр цикла оператора FOR).

98 Integer variable expected (Нужна переменная целого типа).

Предшествующая переменная должна иметь целый тип.

103 Integer or real variable expected (Нужна переменная типа INTEGER or REAL).

113 Error in statement (Ошибка в операторе).

Данный символ не может быть первым символом в операторе.

207 Invalid floating point operation (Недопустимая операция с плавающей запятой) .

Возможные причины сообщения:

- отрицательный аргумент функции SQRT;

- аргумент функции LN равен нулю или имеет отрицательное значение.

Лабораторная работа № 11 Создание программ на языке Turbo Pascal линейной и разветвляющейся структуры.

2.6.1 Цель работы: Изучение основ построения алгоритмов и программ линейной и разветвляющейся структуры.

2.6.2 Задачи работы:

1. Ознакомиться с основными элементами блок-схем (приложение А), реализующими алгоритмы линейных и разветвляющихся структур и основными характеристиками алгоритмов (приложение Б).

2. Освоить приемы построения блок-схем алгоритмов линейных и разветвляющихся структур.

3. Разобрать приведенные в методических указаниях примеры.

4. Выполнить индивидуальные задания (по указанию преподавателя).

5. Для закрепления и проверки полученных навыков, необходимо выполнить дополнительные задания. Результаты работы продемонстрировать преподавателю.

2.6.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер

2. Методические указания для выполнения лабораторной работы

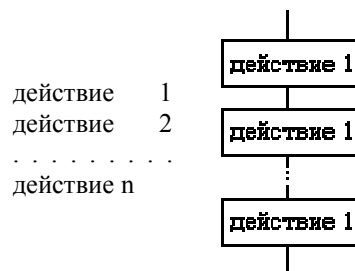
2.6.4 Описание (ход) работы:

Основные положения

Основным в процессе программирования является разработка алгоритма. Это один из наиболее сложных этапов решения задачи с использованием ЭВМ. Основными алгоритмическими структурами являются следование, развилка и цикл.

Базовая структура **следование** образуется из последовательности действий, следующих одно за другим:

Псевдокод Язык блок-схем



Простейшие задачи имеют линейный алгоритм решения. Это означает, что он не содержит проверок условий и повторений.

Пример 1. Пешеход шел по пересеченной местности. Его скорость движения по равнине v_1 км/ч, в гору — v_2 км/ч и под гору — v_3 км/ч. Время движения соответственно t_1 , t_2 и t_3 ч. Какой путь прошел пешеход?

1. Ввести v_1 , v_2 , v_3 , t_1 , t_2 , t_3 .
2. $S_1 := v_1 * t_1$.
3. $S_2 := v_2 * t_2$.
4. $S_3 := v_3 * t_3$.
5. $S := S_1 + S_2 + S_3$.
6. Вывести значение S .
7. Конец.

Пример 2. Дано натуральное трехзначное число n , в записи которого нет нулей. Составить алгоритм, который возвращает значение ИСТИНА, если верно утверждение: "число n кратно каждой своей цифре", и ЛОЖЬ — в противном случае.

1. Ввести число n
2. $A := n \bmod 10$ {разряд единиц}
3. $B := n \div 100$ {разряд сотен}
4. $C := n \div 10 \bmod 10$ {десятки}
5. $L := (n \bmod A = 0) \text{ and } (n \bmod B = 0) \text{ and } (n \bmod C = 0)$
6. Вывод L
7. Конец

На приведенной выше схеме DIV и MOD соответственно операции деления нацело и получения остатка от целочисленного деления. В фигурных скобках записаны пояснения (комментарии) к операторам.

Базовая структура **ветвление** обеспечивает в зависимости от результата проверки условия (да или нет) выбор одного из альтернативных путей работы алгоритма. Каждый из путей ведет к общему выходу, так что работа алгоритма будет продолжаться независимо от того, какой путь будет выбран.

Структура ветвление существует в четырех основных вариантах:

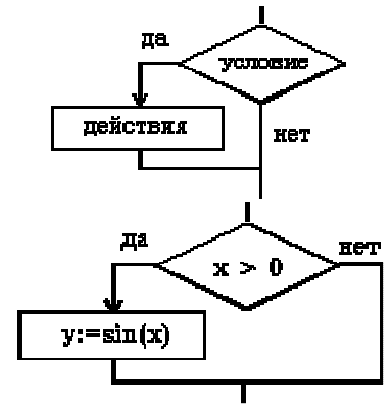
- если-то;
- если-то-иначе;
- выбор;
- выбор-иначе.

Псевдокод	Язык блок-схем
1. если-то	

если условие
то действия
все

Пример: Для положительных x вычислить значение функции $\sin(x)$

если $x > 0$
то $y := \sin(x)$
все

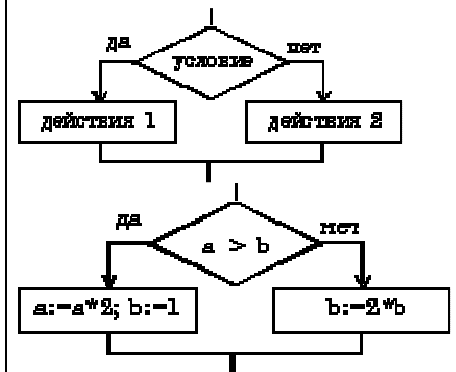


2. если-то-иначе

если условие
то действия 1
иначе действия 2
все

Пример: Даны два числа, если первое из них больше второго, то его необходимо удвоить, а второе число приравнять единице, в противном случае первое число оставить без изменения, а второе – удвоить.

если $a > b$
то $a := 2*a; b := 1$
иначе $b := 2*b$
все

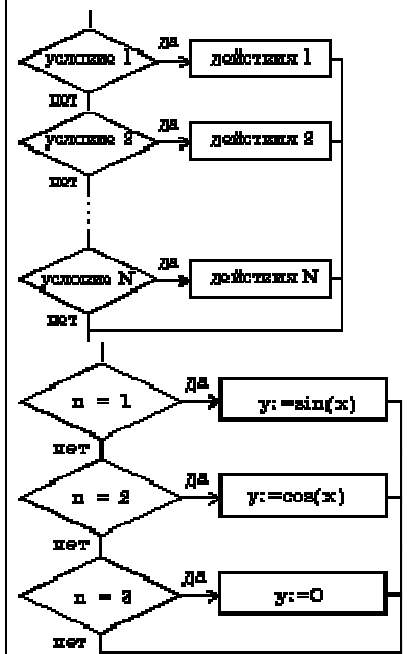


3. выбор

выбор
при условие 1: действия 1
при условие 2: действия 2
.....
при условие N: действия N
все

Пример: в зависимости от значения натурального числа n , могущего принимать значения 1, 2 или 3, присвоить функции y значения $\sin x$, $\cos x$ или нуля.

выбор
при $n = 1$: $y := \sin(x)$
при $n = 2$: $y := \cos(x)$
при $n = 3$: $y := 0$
все



4. выбор-иначе

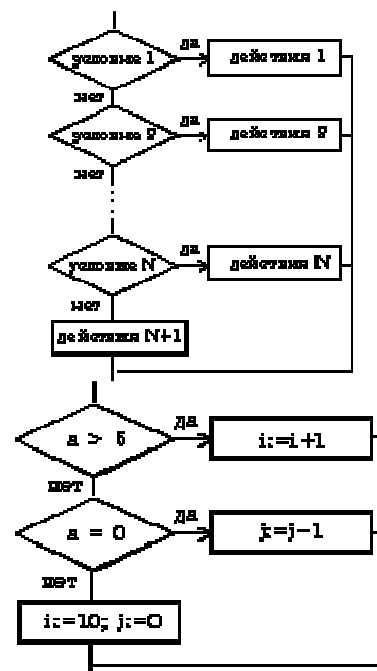
выбор
 при условии 1: действия 1
 при условии 2: действия 2

 при условии N: действия N
 иначе действия N+1

все
 Пример:
 выбор

при $a > 5$: $i := i + 1$
 при $a = 0$: $j := j + 1$
 иначе $i := 10$; $j := 0$

все



Пример 3. Дано натуральное число n . Если число нечётное и его удвоение не приведет к выходу за 32767 (двухбайтовое целое число со знаком), удвоить его, иначе — оставить без изменения.

Чтобы удовлетворить условию удвоения, число n должно быть нечетным и меньше 16384.

1. Ввести число n
2. Если число n нечетное и меньше 16384, то $n := n * 2$
3. Вывод n
4. Конец

Для проверки работоспособности алгоритма необходимо задать значения входных переменных, вычислить конечный результат по алгоритму и сравнить с результатом ручного счета.

Пример 4. Составить алгоритм и программу, запрашивающую имя и затем приветствующую его обладателя.

Тест

Данные	Результат
Имя = "Тимур"	"Привет, Тимур!"

Псевдокод

алг Знакомство (арг лит Имя, резлит t)

нач

вывод "Как тебя зовут ?"

ввод Имя

$t := \text{"Привет, " + Имя + "!"}$ | "+" - операция сцепки

вывод t

кон

Turbo Pascal

Program Hello;

Var Name: String; {Описание переменной Name строкового типа}

BEGIN

```

Write('Как тебя зовут ? '); {Вывод на экран текста вопроса}
ReadLn(Name);               {Ввод с клавиатуры имени}
WriteLn('Привет, ', Name, '!'); {Вывод на экран приветствия}
ReadLn
END.

```

Здесь последний оператор ReadLn позволяет видеть на экране результаты работы программы, пока не будет нажата клавиша <Enter>.

Результаты работы Pascal-программы

```

Как тебя зовут ? Тимур   <Enter>
Привет, Тимур !

```

Пример 5 . Определить объём и площадь боковой поверхности цилиндра с заданными радиусом основания R и высотой H.

Тест

Данные		Результат	
R = 1	H = 1	V = 3.14	S = 6.28

Псевдокод

алг Цилиндр (арг вещ R, H, рез вещ V, S)

нач вещ Pi

Pi := 3.14

V := Pi * R**2 * H

S := 2 * Pi * R * H

кон

Turbo Pascal

Program Cylinder;

Uses Crt; {Подключение библиотеки Crt}

Var

R, {радиус основания цилиндра}

H, {высота цилиндра }

V, {объем цилиндра }

S: Real; {площадь боковой поверхности цилиндра}

BEGIN

ClrScr; {Вызов из библиотеки Crt процедуры очистки экрана}

Write('Введите высоту цилиндра : '); ReadLn(H);

Write('Введите радиус основания : '); ReadLn(R);

V := Pi * R * R * H;

S := 2 * Pi * R * H; WriteLn;

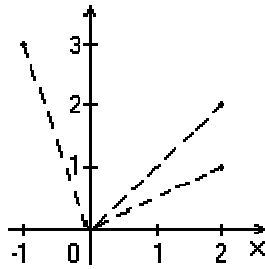
WriteLn('Объем цилиндра = ', V : 5 : 2); {Здесь 5 – общее количество позиций, занимаемых переменной V при выводе, а 2 - количество позиций в дробной части значения V}

WriteLn('Площадь боковой поверхности = ', S : 5 : 2);

ReadLn

END.

Пример 6. Даны три точки на плоскости. Определить, какая из них ближе к началу координат.



Система тестов

Номер теста	Данные						Результат
	xA	yA	xB	yB	xC	yC	Otvet
1	2	1	2	2	-1	3	"Это точка А"
2	2	2	2	1	-1	3	"Это точка В"
3	2	2	-1	3	2	1	"Это точка С"

Псевдокод

алг Точки(арг вещ xA,yA,xB,yB,xC,yC, рез лит Otvet)

```

нач вещ DistA,DistB,DistC
  ввод xA,yA,xB,yB,xC,yC
  DistA := sqrt(xA**2 + yA**2)
  DistB := sqrt(xB**2 + yB**2)
  DistC := sqrt(xC**2 + yC**2)
  если (DistA < DistB) и (DistA < DistC)
    то Otvet := "Это точка А"
  иначе если DistB < DistC
    то Otvet := "Это точка В"
    иначе Otvet := "Это точка С"
  все
все
вывод Otvet
кон
  
```

Turbo Pascal

```

Program Points;
Uses Crt;
Var xA, yA, xB, yB, xC, yC, DistA, DistB, DistC : Real;
BEGIN ClrScr;
  WriteLn('Введите координаты точки А:');
  Write('x = '); ReadLn(xA); Write('y = '); ReadLn(yA);
  WriteLn('Введите координаты точки В:');
  Write('x = '); ReadLn(xB); Write('y = '); ReadLn(yB);
  WriteLn('Введите координаты точки С:');
  Write('x = '); ReadLn(xC); Write('y = '); ReadLn(yC);
  DistA := sqrt(sqr(xA) + sqr(yA)); {расстояние до точки А}
  DistB := sqrt(sqr(xB) + sqr(yB)); {расстояние до точки В}
  DistC := sqrt(sqr(xC) + sqr(yC)); {расстояние до точки С}
  WriteLn; Write('Ответ : ');
  If (DistA < DistB) and (DistA < DistC) {сравнение расстояний}
    then WriteLn( 'Это точка А.')
    else If (DistB < DistC) {сравнение расстояний}
      then WriteLn('Это точка В.')
  
```

```

        else WriteLn('Это точка C. ');
    ReadLn
END.

```

Пример 7 Найти произведение цифр заданного целого четырехзначного числа.

Система тестов

Номер теста	Проверяемый случай	Число	Результат
1	Число положительное	2314	P = 24
2	Число отрицательное	-1245	P = 40

Псевдокод

алг Произведение цифр (арг цел Num, рез цел P)

нач цел i, j, k, l

Num := abs(Num) | abs - абсолютная величина

i := div(Num, 1000) | i - первая цифра

| div - частное от деления с остатком

j := mod(div(Num, 100), 10) | j - вторая цифра

| mod - остаток от деления с остатком

k := mod(div(Num, 10), 10) | k - третья цифра

l := mod(Num, 10) | l - четвертая цифра

P := i * j * k * l;

кон

Turbo Pascal

Program DigitsProduct;

Uses Crt;

Var Number, {заданное число}

i, j, k, l, {цифры числа}

P : Integer; {произведение цифр}

BEGIN ClrScr;

Write('Введите четырехзначное число : '); ReadLn(Number);

Number:=Abs(Number);

Write('Цифры числа ', Number, ' : ');

i := Number div 1000; Write(i:3); { выделение первой цифры}

j := Number div 100 mod 10; Write(j:3); { выделение второй цифры}

k := Number div 10 mod 10; Write(k:3); { выделение третьей цифры}

l := Number mod 10; WriteLn(l:3); { выделение четвертой цифры}

P := i * j * k * l;

WriteLn('О т в е т : произведение цифр равно ', P);

ReadLn

END.

Пример 8. Решить квадратное уравнение $ax^2 + bx + c = 0$.

Таблица

Система тестов

Номер теста	Проверяемый случай	Коэффициенты			Результаты
		a	b	c	

1	$d > 0$	1	1	-2	$x_1 = 1, x_2 = -2$
2	$d = 0$	1	2	1	Корни равны: $x_1 = -1, x_2 = -1$
3	$d < 0$	2	1	2	Действительных корней нет
4	$a=0, b=0, c=0$	0	0	0	Все коэффициенты равны нулю. x - любое число.
5	$a=0, b=0, c < 0$	0	0	2	Неправильное уравнение
6	$a=0, b < 0$	0	2	1	Линейное уравнение. Один корень: $x = -0,5$

Псевдокод (упрощенный алгоритм)

алг Квур (арг вещ a, b, c , рез вещ x_1, x_2 , рез лит t)

дано $a < 0$

нач вещ d

$d := b^2 - 4*a*c$ | d - дискриминант квадратного уравнения

если $d < 0$

то $t :=$ "Действительных корней нет"

иначе если $d = 0$

то $t :=$ "Корни равны"; $x_1 := -b/(2*a)$; $x_2 := x_1$

иначе $t :=$ "Два корня"

$x_1 := (-b + \text{sqrt}(d)) / (2*a)$

$x_2 := (-b - \text{sqrt}(d)) / (2*a)$

все

все

кон

Turbo Pascal

Program QuadraticEquation;

Uses Crt; { подключение библиотеки Crt }

Var a, b, c : Real; { a, b, c - коэффициенты уравнения }

Discr : Real;

x_1, x_2 : Real; { x_1, x_2 - корни }

Test, NTest : Integer; { Ntest - количество тестов }

BEGIN

ClrScr;

Write('Введите количество тестов : ');

ReadLn(NTest);

For Test := 1 to NTest do {цикл по всем тестам задачи }

begin

Write('Тест ', Test, '. Введите коэффициенты a, b, c : ');

ReadLn(a, b, c);

If ($a=0$) and ($b=0$) and ($c=0$) {проверка 4-го случая}

then begin Write('Все коэффициенты равны нулю.');

WriteLn('x - любое число ')

end

else

If ($a=0$) and ($b < 0$) {проверка 6-го случая}

then WriteLn('Линейное уравнение. Один корень: $x =$ ', $(-c/b):6:2$)

else

If ($a=0$) and ($b=0$) and ($c < 0$) {проверка 5-го случая}

then WriteLn('Неправильное уравнение.')

```

else
begin
  Discr := b*b - 4*a*c; {вычисление дискриминанта}
  If Discr > 0 {проверка 1-го случая}
  then begin {вычисление корней}
    x1:=(-b + Sqrt(Discr)) / (2*a);
    x2:=(-b - Sqrt(Discr)) / (2*a);
    WriteLn('x1=' , x1:6:2 , ' ; x2=' , x2:6:2)
  end
else
  If Discr = 0 {проверка 2-го случая}
  then begin
    x1 := -b/(2*a);
    WriteLn('Корни равны: x1=' , x1:6:2 , ' x2=' , x1:6:2)
  end
  else WriteLn('Действительных корней нет. '); {проверка 3-го случая}
end;
WriteLn
end;
ReadLn
END.

```

Пример 9 Контроль корректности данных. Составить программу расчета площади треугольника S по трем сторонам a, b, c, значения которых запрашиваются с клавиатуры, по формуле Герона

$$S = \sqrt{p(p-a)(p-b)(p-c)}, \text{ где } p = (a+b+c)/2.$$

```

Var a, b, c, p, S: real;
Begin
  Write('Ввести сторону a='); ReadLn(a);
  Write('Ввести сторону b='); ReadLn(b);
  Write('Ввести сторону c='); ReadLn(c);
  IF (a>0) and (b>0) and (c>0) {проверка положительности}
    and (a+b>c) and (a+c>b) and (c+b>a) {каждая сторона должна быть меньше
                                          суммы двух других сторон}
  THEN begin p:=(a+b+c)/2;
    S:=sqrt(p*(p-a)*(p-b)*(p-c));
    WriteLn('Площадь:', S:15:5) end
  ELSE Writeln("Неверные входные данные!");
ReadLn
END.

```

Пример 10 Программа случайного предсказания одного из десяти вариантов ближайшего будущего с вероятностью 1/20, в остальных случаях – Вы «неудачник».

```

Var N:word;
Begin
  WriteLn('Предсказание будущего!');
  Randomize; N:=Random(20)+1; {N-случайное число от 1 до 20}
  WriteLn; Write('Вас ожидает_');
  Case N of
    1: WriteLn('счастье');
    2: WriteLn('пятерка');
    3: WriteLn('дорога');
    4: WriteLn('двойка');
    5: WriteLn('здоровье');
    6: WriteLn('здоровье');
    7: WriteLn('деньги');
    8: WriteLn('любовь');
    9: WriteLn('встреча');

```

```

5: WriteLn('болезнь');      10: WriteLn('мама')
Else WriteLn('неудача')    end;
WriteLn('Нажми клавишу Enter');
ReadLn
END.

```

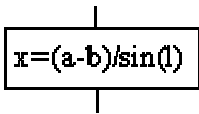
Здесь функция Random(x) генерирует случайное число, с равномерной плотностью распределения на заданном интервале. Для инициализации распределения в начале программы необходимо вызвать процедуру Randomize.

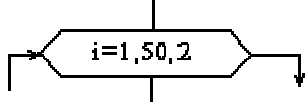
Задачи для самостоятельного решения

1. Вычислить длину окружности, площадь круга и объём шара одного и того же заданного радиуса.
2. Вычислить периметр и площадь прямоугольного треугольника по длинам двух его катетов.
3. По координатам трёх вершин некоторого треугольника найти его площадь и периметр.
4. Вычислить дробную часть среднего геометрического трёх заданных вещественных чисел.
5. Определить, является ли заданное целое число A нечётным двузначным числом.
6. Определить, имеется ли среди заданных целых чисел A, B, C хотя бы одно чётное.
7. Даны три числа. Выбрать те из них, которые принадлежат заданному отрезку $[e, f]$.
8. Выбрать наибольшее из трёх заданных чисел.
9. Два прямоугольника заданы длинами сторон. Определить, можно ли первый прямоугольник целиком разместить во втором.
10. Определить номер квадранта, в котором находится точка с заданными координатами (x, y) .
11. Определить, пройдет ли кирпич с рёбрами a, b, c в прямоугольное отверстие со сторонами x и y . Просовывать кирпич в отверстие разрешается только так, чтобы каждое из его рёбер было параллельно или перпендикулярно каждой из сторон отверстия.
12. Составить программу случайного выбора трех дисциплин, по которым придется сдавать экзамены, из предлагаемых на выбор четырех (всего возможно 4 варианта выбора);
13. Составить программу случайного выбора летнего отдыха из семи предлагаемых туристическим агентством курортов, причем с вероятностью $3/10$ придется отдыхать в деревне.
14. Вывести на экран сообщение в зависимости от полученного значения оценки (по десятибальной системе), например: 1..2: плохо; 3..5: удовлетворительно и т.д., иначе – неправильный ввод данных.
15. Вывести на экран сообщение в зависимости от значения температуры воздуха на улице (от -50 до $+50^{\circ}\text{C}$), например: $-50..-20$: очень холодно; $-19..-10$: холодно и т.д., иначе – неправильный ввод данных.

ПРИЛОЖЕНИЕ А

Основные функциональные элементы блок-схем алгоритмов в соответствии с ГОСТ 19002-89 ЕСПД (Единая система программной документации)

Название символа	Обозначение и пример заполнения	Пояснение
Процесс		Вычислительное действие или последовательность действий

Решение		Проверка условий
Модификация		Начало цикла
Предопределенный процесс		Вычисления по подпрограмме, стандартной подпрограмме
Ввод-вывод		Ввод-вывод в общем виде
Пуск-останов		Начало, конец алгоритма, вход и выход в подпрограмму
Документ		Вывод результатов на печать

Блок **процесс** применяется для обозначения действия или последовательности действий, изменяющих значение, форму представления или размещения данных. Для улучшения наглядности схемы несколько отдельных блоков обработки можно объединять в один блок. Представление отдельных операций достаточно свободно.

Блок **решение** используется для обозначения переходов управления по условию. В каждом блоке "решение" должны быть указаны вопрос, условие или сравнение, которые он определяет.

Блок **модификация** используется для организации циклических конструкций. (Слово модификация означает видоизменение, преобразование). Внутри блока записывается параметр цикла, для которого указываются его начальное значение, граничное условие и шаг изменения значения параметра для каждого повторения.

Блок **предопределенный процесс** используется для указания обращений к вспомогательным алгоритмам, существующим автономно в виде некоторых самостоятельных модулей, и для обращений к библиотечным подпрограммам

ПРИЛОЖЕНИЕ Б

Название "алгоритм" произошло от латинской формы имени среднеазиатского математика аль-Хорезми — Algorithmi. Алгоритм — одно из основных понятий информатики и математики.

Основные свойства алгоритмов следующие:

❶ **Понятность** для исполнителя — т.е. исполнитель алгоритма должен знать, как его выполнять.

❷ **Дискретность** (прерывность, раздельность) — т.е. алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов (этапов).

❸ **Определенность** — т.е. каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвола. Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.

❹ **Результативность** (или конечность). Это свойство состоит в том, что алгоритм должен приводить к решению задачи за конечное число шагов.

❶ **Массовость.** Это означает, что алгоритм решения задачи разрабатывается в общем виде, т.е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется областью применимости алгоритма.

Приложение В

Типы ошибок

Вид ошибки	Пример
Неправильная постановка задачи	Правильное решение неверно сформулированной задачи
Неверный алгоритм	Выбор алгоритма, приводящего к неточному или эффективному решению задачи
Ошибка анализа	Неполный учет ситуаций, которые могут возникнуть; логические ошибки
Семантические ошибки	Непонимание порядка выполнения оператора
Синтаксические ошибки	Нарушение правил, определяемых языком программирования
Ошибки при выполнении операций	Слишком большое число, деление на ноль, извлечение квадратного корня из отрицательного числа и т. п.
Ошибки в данных	Неудачное определение возможного диапазона изменения данных
Опечатки	Перепутаны близкие по написанию символы, например, цифра 1 и буквы <i>I, l</i>
Ошибки ввода-вывода	Неверное считывание входных данных, неверное задание форматов данных

Лабораторная работа № 12 Создание программ на языке Turbo Pascal циклической структуры.

2.6.1 Цель работы: Изучение основ построения алгоритмов и программ циклической структуры.

2.6.2 Задачи работы:

1. Ознакомиться с основными элементами блок-схем алгоритмов и программ циклической структуры .
2. Освоить приемы построения блок-схем алгоритмов циклической структуры.
3. Разобрать приведенные в методических указаниях примеры.
4. Выполнить индивидуальные задания (по указанию преподавателя).
2. Для закрепления и проверки полученных навыков, необходимо выполнить дополнительные задания. Результаты работы продемонстрировать преподавателю.

2.6.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Методические указания для выполнения лабораторной работы

2.6.4 Описание (ход) работы:

Основные положения

Командой повторения или *циклом* называется такая форма организации действий, при которой одна и та же последовательность действий повторяется до тех пор, пока сохраняется значение некоторого логического выражения. При изменении значения

логического выражения на противоположное повторения прекращаются (цикл завершается).

Для организации цикла необходимо выполнить следующие действия:

- перед началом цикла задать начальное значение параметра;
- внутри цикла изменять параметр цикла с помощью оператора присваивания;
- проверять условие повторения или окончания цикла;
- управлять циклом, т.е. переходить к его началу, если он не закончен, или выходить из цикла в противном случае.

Различают циклы с известным числом повторений (*цикл с параметром*) и *итерационные* (с пред- и постусловием).

В цикле с известным числом повторений параметр изменяется в заданном диапазоне.

Если в цикле изменяется простая переменная, то она является параметром цикла; если в цикле изменяется переменная с индексом, то индекс этой переменной является параметром цикла.

Ниже приведены графические обозначения (обозначения на блок-схемах) алгоритмов циклических структур (рисунок 1).

На схемах СЕРИЯ обозначает один или несколько любых операторов; УСЛОВИЕ есть логическое выражение (ЛВ) (если его значение ИСТИНА, переход происходит по ветви ДА, иначе — по НЕТ). На схеме цикла с параметром использованы обозначения: ПЦ — параметр цикла, НЗ — начальное значение параметра цикла, КЗ — конечное значение параметра цикла, Ш — шаг изменения параметра цикла.

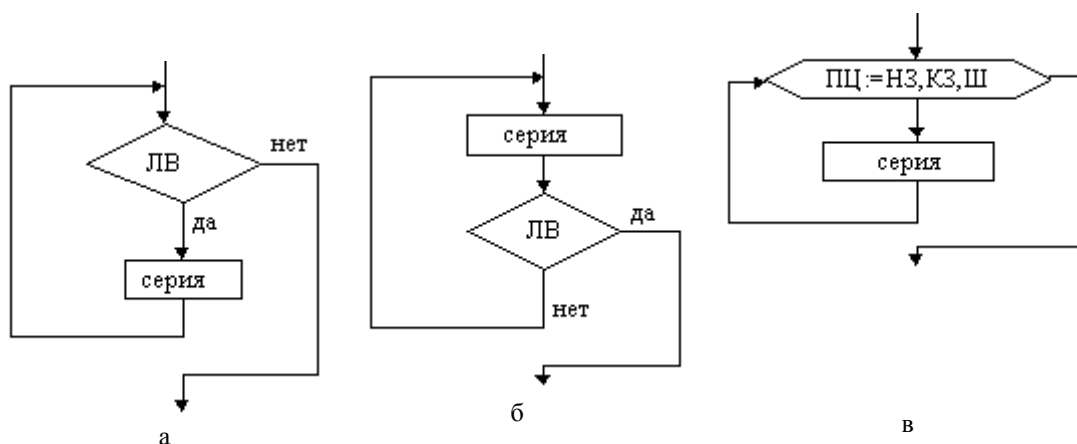


Рисунок 1

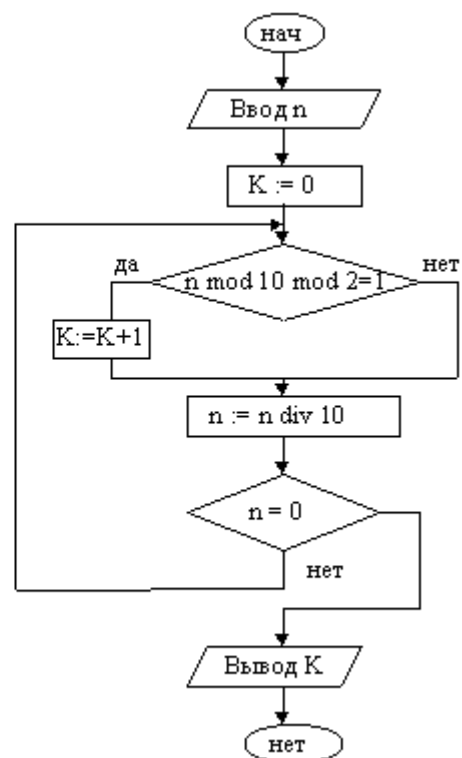
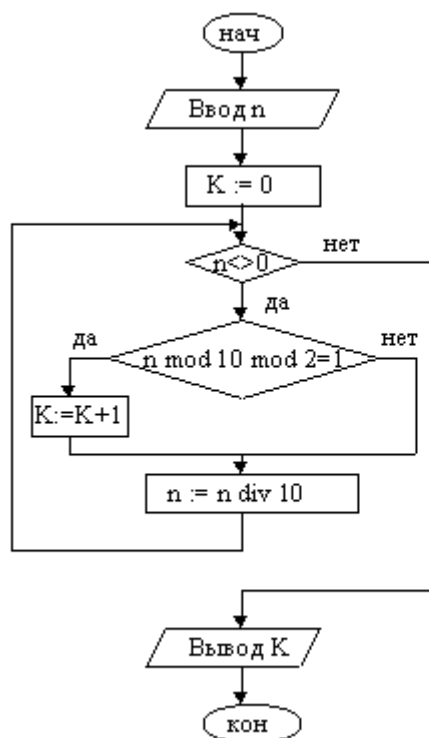
а-цикл с предусловием (цикл ПОКА), б-цикл с постусловием (цикл ДО), в-цикл с параметром

Начало и конец алгоритма на блок-схемах обозначают овалом, вводимые и выводимые переменные записываются в параллелограмме.

В примерах мы будем использовать запись алгоритмов с помощью блок-схем и словесное описание.

Пример 1. Подсчитать количество нечетных цифр в записи натурального числа n .

Идея решения. Из заданного числа выбирать из младшего разряда цифру за цифрой до тех пор, пока оно не исчерпается, т.е. станет равным нулю. Каждую нечётную цифру учитывать.



1. Ввести число n
2. $K := 0$ {подготавливаем счётчик}
3. Если $n = 0$, переход к п. 7
4. Если $n \bmod 10 \bmod 2 = 1$, то $K := K + 1$
5. $n := n \div 10$
6. Переход к п. 3
7. Вывод K
8. Конец

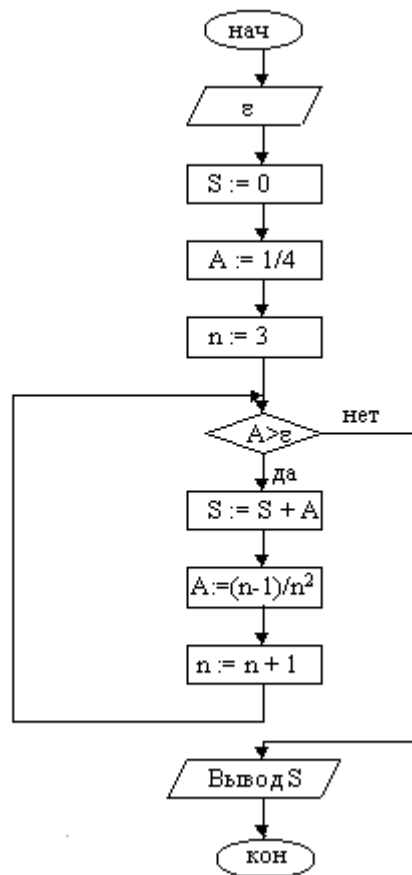
Задача решена двумя способами. Слева решение оформлено с использованием цикла с предусловием, справа — с постусловием.

Пример 2. Дана последовательность, общий член которой определяется формулой

$$a_n = \frac{n-1}{n^2}.$$

Вычислить при $n > 2$ сумму тех ее членов, которые больше заданного числа ε .

При решении задачи находится очередной член последовательно и, если он больше ε , добавляется к сумме.

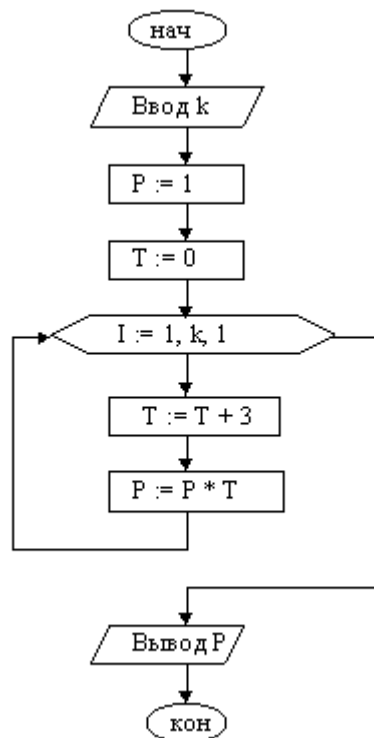


1. Ввести ε
2. $S := 0$
3. $A := 1/4$
4. $n := 3$
5. Сравнить A с ε . Если $A \geq \varepsilon$, переход к п. 10
6. $S := S + A$
7. $A := (n-1)/(n*n)$
8. $n := n + 1$
9. Переход к п. 5
10. Вывод S
11. Конец

В рассмотренных выше примерах количество повторений заранее неизвестно. В первом оно зависит от количества цифр в записи натурального числа, во втором — от числа ε . В тех же случаях, когда количество шагов известно из условия задачи, проще и предпочтительней использовать *цикл с параметром*.

Пример 3. Найти произведение первых k натуральных чисел, кратных трём.

При составлении алгоритма учтем, что первое натуральное число, кратное 3, есть тройка, а все последующие больше предыдущего на 3.



1. Ввод k
2. P := 1 {здесь накапливаем произведение}
3. T := 0 {здесь будут числа, кратные 3}
4. I := 1
5. Если I > k, переход к п. 10
6. T := T + 3
7. P := P * T
8. I := I + 1
9. Перейти к п. 5
10. Вывод P
11. Конец

Для проверки работоспособности алгоритма необходимо задать значения входных переменных, вычислить конечный результат по алгоритму и сравнить с результатом ручного счета.

Пример 4. Для заданного x и n вычислить

$$S = 1 - \frac{x}{1!} + \frac{x^2}{2!} - \frac{x^3}{3!} + \dots + (-1)^n \frac{x^n}{n!}$$

Здесь $n! = 1 \cdot 2 \cdot 3 \dots n$ (читается как "n-факториал").

Тест

Данные		Результат
X=1	n=3	$S = 1 - \frac{1}{1} + \frac{1^2}{1 \cdot 2} - \frac{1^3}{1 \cdot 2 \cdot 3} = 0.33$

Псевдокод

алг Сумма Ряда (арг вещ x , арг цел n , рез вещ S)

нач цел i , вещ P | P - очередное слагаемое

$S := 1$; $P := 1$

нц для i от 1 до n

$P := -P \cdot x / i$ | получение очередного слагаемого

$S := S + P$

кц

конец

Turbo Pascal

Program SumUp;

Uses Crt;

Var x, S, P : Real;

{ P - очередное слагаемое}

i, n : Integer;

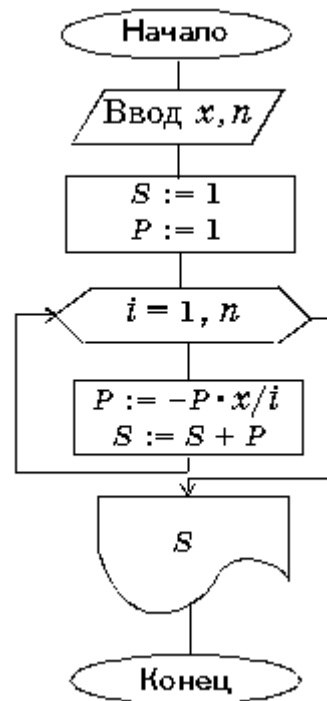
BEGIN ClrScr;

Блок-схема

```

Write('Введите n = '); ReadLn(n);
Write('Введите x = '); ReadLn(x); WriteLn;
S := 1; P := 1;
For i := 1 to n do
begin
  P := - P*x /i; {получение очередного слагаемого}
  S := S + P
end;
WriteLn('О т в е т : S = ', S : 7 : 3 ); ReadLn
END.

```



Пример 5 Фрагмент программы простой парольной защиты

```

For I:=1 to 3 do begin {дается три попытки ввести пароль}
Write ('Введите ПАРОЛЬ');
ReadLn(S); {переменные S и Parol – одного типа}
IF S=Parol
THEN BREAK {прерывание цикла защиты}
ELSE IF I<3 THEN CONTINUE; {даем еще попытки}
WriteLn('Не знаете Пароль!'); {сработает при I=3}
ReadLn;
HALT {прерывание программы}
End;
{ Защищаемый участок программы }

```

Пример 7. Числа Фибоначчи (F_i) определяются по формулам $F_0 = F_1 = 1$; $F_i = F_{i-1} + F_{i-2}$ при $i = 2, 3, \dots$ (каждое очередное число равно сумме двух предыдущих). Вычислить сумму всех чисел Фибоначчи, которые не превосходят заданного натурального числа M .

Тест

Номер теста	Данные	Результат
1	M=10	S=1+1+2+3+5+8=20
2	M=1	S=1+1=2

Псевдокод

алг Фибоначчи (арг цел M, рез цел S)

дано | $M > 0$

нач цел F0, F1, F2

F0:=1; F1:=1; F2:=2

S:=4 | 4 – сумма первых трех чисел Фибоначчи

нц пока $F2 \leq M$

F0:=F1; F1:=F2; F2:=F0+F1 | серия переприсваиваний

S:=S+F2;

кц

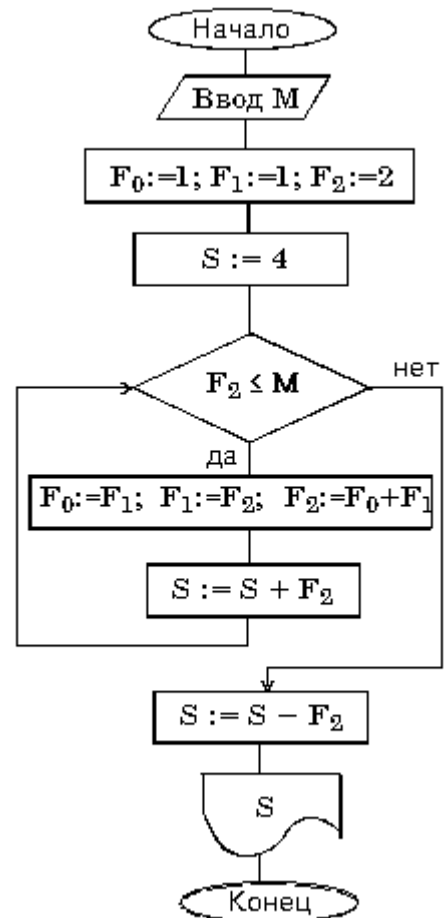
S:=S-F2 | из S вычитается последнее значение F2, превосходящее M

Кон

Исполнение алгоритма

F0	F1	F2	S	F2<M
1	1	2	4	+
1	2	3	4+3=7	+
2	3	5	7+5=12	+
3	5	8	12+8=20	+
5	8	13	20+13=33	- (кц)
			33-13=20	

Блок-схема



Turbo Pascal

```
Program SummaFib;
```

```
Uses Crt;
```

```
Var M,           {заданное число }
```

```
    F0, F1, F2,   {три последовательных числа Фибоначчи}
```

```
    S : Integer;  {сумма чисел Фибоначчи}
```

```
BEGIN
```

```
  ClrScr;
```

```
  Write('Введите натуральное M : ');
```

```
  ReadLn(M);
```

```
  F0:=1; F1:=1; F2:=2;
```

```
  S:=4;           {4 – сумма первых трех чисел Фибоначчи}
```

```
  Write('Числа Фибоначчи, не превосходящие ', M, ' : ', F0:4, F1:4);
```

```
  While F2<=M do
```

```
  begin
```

```
    F0:=F1; F1:=F2; Write(F1 : 4);
```

```
    F2:=F0+F1; S:=S+F2;
```

```
  end;
```

```
  S:=S-F2; {вычитание из суммы последнего числа, которое превосходит M}
```

```
  WriteLn; WriteLn;
```

```
  WriteLn('О т в е т : Сумма этих чисел равна ', S); ReadLn
```

```
END.
```

Результаты работы Pascal-программы

Введите натуральное M>0 : 10 <Enter>
 Числа Фибоначчи, не превосходящие 10 : 1 1 2 3 5 8
 О т в е т : Сумма этих чисел равна 20

Пример 8 Определим сумму ряда $S = 1 + \frac{1}{3^2} + \frac{1}{5^2} + \dots + \frac{1}{(2N-1)^2}$, пока очередной

член ряда не станет меньше 10^{-8} .

```
Var S,a,eps : real;
N : word;
Begin
  Eps:=1e-8; {точность вычислений}
  S:=0; N:=0; {начальные значения параметров}
REPEAT
  a:=1/(2*N+1)/(2*N+1); {текущий член ряда}
  inc(N); {счетчик членов ряда}
  S:=S+a; {частичная сумма ряда}
UNTIL a<eps;
WriteLn('S=',S:10:8);
ReadLn
END.
```

Пример 10 Предотвращение неверного ввода данных. Модернизируем программу определения площади треугольника по трем сторонам (из предыдущего практического занятия) так, чтобы при неверном вводе данных происходило не просто прерывание программы, а возникал запрос на их корректировку (с повтором до выполнения нужных условий).

```
Var a,b,c,p,S: real;
Begin
REPEAT {проверка корректности ввода данных}
  REPEAT
    Write('Ввести сторону a='); ReadLn(a);
    IF a<=0 THEN WriteLn('должно быть a>0!') Until a>0;
  REPEAT
    Write('Ввести сторону b='); ReadLn(b);
    IF b<=0 THEN WriteLn('должно быть b>0!') Until b>0;
  REPEAT
    Write('Ввести сторону c='); ReadLn(c);
    IF c<=0 THEN WriteLn('должно быть c>0!') Until c>0;
  IF (a+b<=c) and (a+c<=b) and (c+b<=a)
  THEN WriteLn('сумма двух сторон должна больше третьей!')
  UNTIL (a+b>c) and (a+c>b) and (c+b>a);
  {Расчет по проверенным данным}
  p:=(a+b+c)/2;
  S:=sqrt(p*(p-a)*(p-b)*(p-c));
  WriteLn('Площадь:', S:15:5);
ReadLn
END.
```

Цикл с предусловием в подобных случаях использовать менее удобно, в частности потому, что он может ни разу не выполниться, если переменные, которым задаются значения с клавиатуры, уже имели удовлетворяющие условиям «защиты» значения.

Пример 11 Рассчитать период (в месяцах) роста банковского вклада в m раз при известной ежемесячной процентной ставке pr (1% соответствует $pr=1$).

```
Var pr,m,s,s0: double;
```

N: word;
 Begin
 Repeat
 Write ('Введи проценты по вкладу:'); redLn(pr);
 Write ('Введи коэффициент роста вклада:'); redLn(m)
 Until (m>1) and (pr>0);
 N:=0; {счетчик месяцев}
 s0:=1000; {начальная сумма вклада – произвольно}
 s:=s0; {текущая сумма вклада}
 Repeat {перебор по месяцам}
 s:=s*(1+pr/100);
 inc(N)
 Until s/s0>=m;
 WriteLn ('количество месяцев вклада:',N);
 ReadLn
 End.

Задачи для самостоятельного решения

1 Даны действительные числа x , a , натуральное число n . Вычислить
 $((\dots ((x+a)^2+\dots a)^2+a)^2+a$
 n скобок

2 Дано действительное число x . Вычислить:

$$\frac{(x-2)(x-4)(x-8)\dots(x-64)}{(x-1)(x-3)(x-7)\dots(x-63)}$$

3 Даны натуральное n , действительное x . Вычислить:

$$x^{n^2} / 2^n$$

4 Даны натуральное n , действительное x . Вычислить:

$$x^{n^3} / 3^n$$

5 Даны натуральное n , действительное x . Вычислить:

$$\sum_{i=1}^n \frac{x^i}{i!}$$

6 Вычислить: $\sum_{i=1}^{100} \frac{1}{i^2}$

7 Вычислить: $\sum_{i=1}^{10} \frac{1}{i!}$

8 Вычислить: $\sum_{i=1}^{50} \frac{1}{i^3}$

9 Вычислить: $\sum \frac{1}{(2i)^2}$

10 Дано натуральное n . Вычислить: $\sum_{k=1}^n \frac{1}{K}$

11 Дано натуральное n . Вычислить:

12 Дано натуральное n . Вычислить:

$$\sum_{k=1}^n \frac{k!}{\frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{k+1}}$$

13. Известен начальный вклад клиента в банк и процент годового дохода. Определить, через сколько лет вклад превысит заданный размер и каков при этом будет размер вклада.

14. Торговая фирма в первый день работы реализовала товаров на P тыс. руб., а затем ежедневно увеличивала выручку на 3%. Какой будет выручка фирмы в тот день, когда она впервые превысит заданное значение Q ? Сколько дней придется торговать фирме для достижения этого результата?

15. Малое предприятие в первый день работы выпустило P единиц товарной продукции. Каждый последующий день оно выпускало продукции на Q единиц больше, чем в предыдущий. Сколько дней потребуется предприятию, чтобы общее количество выпущенной продукции за все время работы впервые превысило запланированный объем?

2.7 Лабораторная работа № 13, 14, 15 (6 часов)

Тема: «Программные средства реализации информационных процессов. Прикладное программное обеспечение»

Лабораторная работа № 13, 14 Создание комплексных документов в MS Word

2.7.1 Цель работы: изучение основных возможностей текстового процессора Microsoft WORD и получение практических навыков по созданию и редактированию текстовых документов.

2.7.2 Задачи работы:

1. Освоить основные приемы работы с документами и редактирование, форматирования текста в MS Word
2. Освоить основные приемы создания, редактирования и форматирования графических объектов в документах текстового процессора MS Word
3. Освоить основные приемы оформления структурированного документа в MS Word
4. Освоить основные приемы создания, редактирования и форматирования таблиц в документах текстового процессора MS Word
5. Освоить основные приемы создания и форматирования математических формул в текстовых документах MS Word
6. Для закрепления и проверки полученных навыков, необходимо ответить на контрольные вопросы и выполнить дополнительные задания. Результаты работы продемонстрировать преподавателю в электронном виде.

2.7.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Методические указания для выполнения лабораторной работы
3. Текстовый процессор Microsoft Word

2.7.4 Описание (ход) работы:

Для того, чтобы создавать, а также редактировать и форматировать (исправлять, изменять) тексты, существуют специальные программы, называемые текстовыми редакторами или текстовыми процессорами. Текстовые процессоры сложнее и обладают большими функциональными возможностями для оформления текстов. В последнее время все большую популярность среди широкого круга пользователей завоевывает текстовый процессор Word для Windows.

Word – многофункциональная программа обработки текстов, настольная издательская система. Ее предназначение:

- 1) набор, редактирование, верстка текста и таблиц;
- 2) управление всеми пунктами меню, опциями и командами с помощью мыши;
- 3) просмотр на дисплее готового к печати документа без затраты бумаги на дополнительные распечатки;
- 4) вставка рисунков и слайдов;
- 5) заготовка бланков, писем и других документов;
- 6) обмен информацией с другими программами;
- 7) проверка орфографии и поиск синонимов.

Запуск MS Word и создание нового документа

Запуск программы Word может осуществляться любым из стандартных способов

запуска приложений:

- Щелкните кнопку **Пуск** и выберите в меню пункт **Создать документ Microsoft Office**. В открывшемся окне дважды щелкните по значку **Новый документ**.
- Щелкните кнопку **Пуск**, выберите пункт **Программы, Microsoft Word**.
- Дважды щелкните левой клавишей мыши по значку **Microsoft Word** на *Рабочем столе*.

Как правило, при запуске программы MS Word можно сразу приступить к вводу текста. Создание нового документа в других случаях (уже в самом текстовом редакторе) осуществляется командами **Создать** в меню «Office».

Экранный интерфейс редактора Microsoft Word (Окно программы MS Word)

После запуска программы на экране появляется окно с открытым в нем пустым документом, которому по умолчанию присваивается имя **Документ Microsoft Office Word**. Окно содержит следующие элементы (Рис.1):

Строка названия документа (1) – слева в строке заголовка находится значок Word (кнопка «Office»), рядом имя загруженного файла документа, с которым вы работаете в текущий момент (в данном случае этот файл называется **Документ Microsoft Office Word**), затем – имя программы - **Microsoft Word** и справа 3 кнопки управления окном (свернуть, развернуть (свернуть в окно), закрыть документ).

Кнопка «Office» (2) – находится в верхнем левом углу документа. Некоторые пункты меню **Office** раскрываются и имеют соответствующее подменю, если навести на них указатель мыши.

Вкладка или панель инструментов (4) – находится ниже строки названия документа. Внутри каждой группы инструментов (5) собраны различные элементы управления, которые представляют наиболее часто выполняемые операции при работе с документами. Панели инструментов обеспечивают быстрый и более простой доступ к важнейшим функциям редактора.

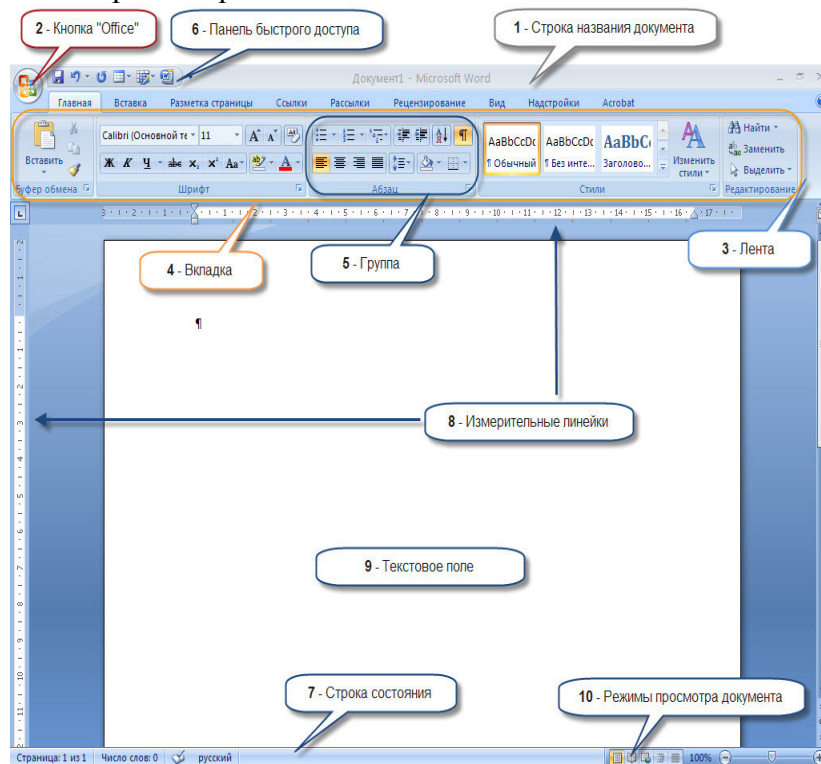


Рис.1 Окно программы Word

Ниже находится Рабочая область или Текстовое поле (9) – это область экрана, где отображается создаваемый документ и производится его редактирование. Сверху и слева документа находятся горизонтальная и вертикальная измерительные линейки (8).

Линейка – используется при формировании абзацев и таблиц. Для вывода на экран линейки, нужно выполнить команды: **Вид, Линейка**.

Справа и внизу документа – полосы вертикальной и горизонтальной прокрутки или лента (3) Они применяются в тех случаях, когда весь текст не помещается на экране и требует сдвига (прокрутки) вверх-вниз или влево-вправо, используя при этом *кнопки прокрутки* или *ползунок прокрутки*.

Окно документа Word можно разделить на две части (с помощью команды меню **Окно, Разделить**) и работать одновременно с двумя частями одного и того же документа. При этом каждая часть документа имеет свою собственную *полосу прокрутки*. Полосу разделения можно перемещать с помощью мыши.

Разделить экран на два подокна можно перетаскиванием вниз маленькой кнопки (вешки полосы разделения), которая находится над верхней стрелкой вертикальной полосы прокрутки.

Заккрыть второе подокно можно командой меню **Вид, Снять разделение** или перемещением разделителя.

Мигающая вертикальная черта (штрих) - текстовый курсор – указывает, куда в следующий момент времени будет вводиться знак текста, вставляться таблица или рисунок. Передвигать текстовый курсор можно с помощью клавиш управления курсором или с помощью мыши, произведя щелчок в нужном месте документа.

Нижняя строка окна программы называется информационной строкой или строкой состояния(7). В ней отображается различная информация о самом документе и о текущем состоянии текстового процессора. Первые три числа означают номер страницы в документе, общее количество страниц в документе, число слов в документе.

Завершение работы с MS WORD

Завершение работы Word можно осуществлять с помощью кнопки **Заккрыть** .

Если измененный документ не записан в файл, Word потребует записать его, либо подтвердить выход без записи.

Первоначальные сведения и правила работы с Word.

1. При работе с документом пользователь задает подходящий режим просмотра и редактирования текста. Режим устанавливается с помощью пункта меню **Вид**. В режиме **Обычный** достигается наибольшая скорость ввода и редактирования текста. Режим **Разметка страницы** более точно отображает окончательный внешний вид страниц документа при печати, кроме того, он обязателен при работе с графическими объектами, рамками и многоколоночным текстом. Режим **Структура документа** и **Режим электронного документа** полезны опытным пользователям при работе с большими документами. Также задать подходящий режим можно при помощи кнопок Режим просмотра документа (10) в рабочем окне Word.

2. Используя команды **Вид, Масштаб** можно изменять масштаб видимости документа.

3. Начинать работу с текстовым документом необходимо с настройки параметров страницы. Это осуществляется с помощью команды меню **Разметка страницы** на вкладке (4). В открывшейся ленте(4) **Разметка страницы** в группе(5) **Параметры страницы** выбрав вкладку **Поля**, можно установить отступы от всех краев листа бумаги. Вкладка **Размер бумаги** позволяет установить размер самого листа бумаги и его ориентацию при печати: **Книжная** или **Альбомная** и т.д.

4. Так же можно пронумеровать страницы. Для этого нужно:
 - выполнить команду меню **Вставка, Номера страниц**;
 - указать положение и выравнивание номера.

Номера страниц можно установить, используя колонтитулы. Колонтитулы представляют собой специальное поле, размещаемое, как правило, в верхнем и нижнем полях страницы, содержащее различные объекты, повторяющиеся на каждой странице, например, время, имя файла, имя автора и т.п. Устанавливаются колонтитулы через команду меню **Вставка, Верхний колонтитул** или **Нижний колонтитул** (все зависит от того в какое место документа – в верх или вниз – нужно вставить номер).

5. Для автоматического переноса слов в документе необходимо выполнить команды **Разметка страницы, Расстановка переносов** и в появившемся диалоговом окне включить переключатель *Авто* (переносы будут расставляться автоматически)

6. Для проверки правописания, необходимо щелкнуть кнопку **Рецензирование** на панели инструментов и выбрать команду **Правописание**.

Основные приемы редактирования документа

Под редактированием понимается изменение уже существующего документа. Начинают редактирование с загрузки (открытия) документа. Для этого используется команда **Открыть** в меню Office (2), по выполнению которой на экране появляется стандартное диалоговое окно **Открытие документа**. Если нужный документ находится в другой папке, его надо разыскать и открыть.

Процедура редактирования текста включает следующий комплекс операций:

1. **Ввод текста** – производится обычным способом при помощи клавиатуры. Следует отметить, что клавиша Enter служит для обозначения конца абзаца. Перенос текста на новую строку производится автоматически при достижении текстом правой границы. Если при вводе текста была допущена ошибка, она устраняется удалением символов при помощи клавиш Backspace (удаляет символ слева от курсора) или Delete (удаляет символ справа).

2. **Копирование.** Чтобы скопировать фрагмент текста, надо:

- выделить нужный фрагмент;
 - скопировать его в буфер обмена с помощью команды **Копировать**, находящийся в меню Главная в группе Буфер обмена
 - установить курсор в то место текста, куда нужно вставить фрагмент;
 - выполнить команду находящийся в меню Главная в группе Буфер обмена.
- Вставить фрагмент из буфера обмена можно столько раз, сколько нужно.

Прежде чем начинать редактировать или форматировать фрагмент текста, необходимо его выделить. Это можно сделать с помощью мыши или используя клавиатуру.

Чтобы выделить	Действие
Слово	Дважды щелкнуть слово.
Строку текста или несколько строк текста	Переместить указатель мыши в зону выделения (Зона выделения находится между левой границей текста и листа. В этой зоне указатель мыши принимает вид стрелки, направленной в правый верхний угол ↗) и щелкнуть левой кнопкой мыши. Для выделения более чем одной строки, необходимо удерживая левую кнопку мыши, осуществлять ее перемещение вверх или вниз.
Абзац	Переместить указатель мыши в зону выделения и сделать двойной щелчок левой кнопкой мыши.
Весь текст	Переместить указатель мыши в зону выделения и сделать тройной щелчок левой кнопкой мыши.

Часть строки	Переместите указатель мыши к первому выделяемому символу строки так, чтобы он принял вид I , удерживая нажатой левую кнопку мыши, протяните мышь вправо (влево) до нужного символа.
Блок текста	Щелкните начало фрагмента и удерживая клавишу Shift нажатой, щелкните конец фрагмента.

Чтобы снять выделение, щелкните по любому месту левой кнопкой мыши.

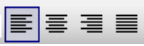
3. **Удаление текста.** Для удаления фрагмента текста необходимо его выделить, а затем выполнить команду **Вырезать**, находящийся в меню **Главная** в группе **Буфер обмена**

4. **Отмена** последнего действия осуществляется с помощью кнопок  на панели быстрого доступа (6). Вернуть отмененное действие можно кнопкой **Вернуть**  на той же панели (6).

Форматирование документа в MS Word

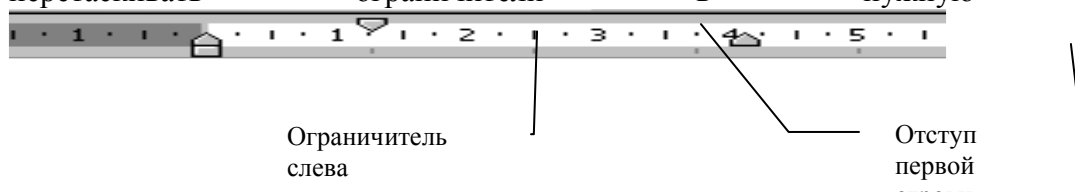
Целью обработки любого текста является придание ему некоторого вида облегчающего восприятие информации и подчиненного общепринятым правилам и стандартам. **Форматирование** – придание документу определенной формы. Форматирование текста осуществляется средствами меню **Главная**. Процедура форматирования текста включает следующие операции:

Форматирование абзацев. При вводе текста новый абзац начинается после очередного нажатия клавиши Enter. Форматирование абзаца подразумевает: выравнивание абзаца по заданным границам, установку межстрочных интервалов и отступов. Для форматирования одного абзаца его не обязательно выделять. Достаточно установить курсор в любое место внутри абзаца. Чтобы применить одинаковое форматирование к нескольким абзацам, их необходимо предварительно выделить.

Выравнивание. Это способ расположения строк между правой и левой границами текста. Существует четыре вида выравнивания абзаца – по центру, влево, вправо и по ширине. Выравнивание абзаца можно выполнить, используя команды **Главная**, группа **Абзац** кнопки .

Установка межстрочного интервала. Это расстояние между строками в абзаце. Интервалы устанавливаются с помощью команд **Главная**, группа **Абзац**, в открывшемся диалоговом окне **Отступы и интервалы** выбрать нужные параметры.

Установка абзацных отступов. Это отступы абзаца от левого и правого ограничителя и отступ первой строки абзаца (красная строка). Отступы устанавливаются с помощью команд **Главная**, группа **Абзац**, в открывшемся диалоговом окне **Отступы и интервалы** пункт **Отступ** или с помощью **Линейки**, на которой необходимо перетаскивать ограничители в нужную позицию.



Форматирование символов – подразумевает изменение типа, стиля, размера, начертания шрифта.

Установка шрифтов. Текст документа может быть оформлен разными шрифтами. Шрифтовое оформление текста производится либо по мере того, как он вводится, либо уже на введенном тексте. Прежде чем выполнять шрифтовое оформление существующего текста, необходимо произвести его выделение. Затем воспользоваться кнопками

Обычный + По л Times New Roman 12 Ж К Ч на панели инструментов Главная в группе Шрифт и выбрать нужные параметры.

Имеется ряд дополнительных возможностей при выборе шрифта: Подчеркивание, Цвет, Эффекты.

Сохранение документа

Для сохранения документа используются команды: **Файл, Сохранить** или **Сохранить как** в меню **Офиссе**. Команда **Сохранить** или кнопка  на панели быстрого доступа сохраняет документ с текущим именем, а команда **Сохранить как** позволяет сохранить документ под новым именем. Для этого в диалогом окне **Сохранение документа** необходимо выбрать папку, в которой нужно сохранить документ, ввести имя файла с клавиатуры и щелкнуть по кнопке «Сохранить»

Печать документа

Перед тем как печатать документ, его необходимо просмотреть. Для переключения в режим предварительного просмотра необходимо выбрать команды: **Предварительный просмотр**, кнопка  на панели быстрого доступа. Выход из режима просмотра документа осуществляется щелчком по кнопке «Заккрыть».

Печать документа производится командами: **Печать** в меню **Office**. В окне диалога **Печать** выбирают принтер (щелкнув по кнопке **Свойства**, можно изменить свойства печати, например, задать качество печати), указывают номера распечатываемых страниц и количество копий документа. Затем нажимают на кнопку «ОК».

1. Установка параметров окна пользователя

1. Загрузим программу **MS Word**:
 - выполним команды **Пуск, Программы, Microsoft Office, MS Word**.
2. Выберем команду **Вид**, щелкнув по ней левой кнопкой мыши.
3. Установим следующие режимы просмотра документов:
Выбор соответствующего режима осуществляется щелчком по нему левой кнопки мыши:
 - режим разметки страницы;
 - режим чтения;
 - режим электронного документа;
 - режим структуры.
4. Вернемся к режиму **Разметка страницы**.
5. Выведем на экран линейки, выбрав команды: **Вид, Линейка**.
Пункт меню Линейка должен быть помечен символом ✓
7. Установим масштаб – 150%, выполнив команду: **Вид, Масштаб**, а затем вернемся к исходному масштабу.
8. Завершим работу с редактором **MS Word**:
 - выполним команды **Заккрыть** в меню **Office**.

2. Сохранение, загрузка и печать документов

1. Загрузим редактор **MS Word**.
*Если после запуска редактора на экран не будет выведено автоматически окно документа, то создадим новый документ, выполнив команды: **Создать** в меню **Office** и подтвердив тип устанавливаемого по умолчанию документа – **новый документ**.*

В строке заголовка появилось предлагаемое по умолчанию имя – **Документ 1**.

2. Наберем текст в установленных по умолчанию границах и заданным по умолчанию шрифтом текста для набора.

Текст документа:

Находите время!

Находите время для размышлений – это источник силы!

Находите время для работы – это условие успеха!

Находите время для мечты – это путь к звездам!

Находите время для любви – это источник радости жизни!

Находите время для веселья – это музыка души!

Находите время для дружбы – это условие счастья!

Находите время для игры – это секрет молодости!

3. Сохраним документ под именем **«ww.doc»**:

– выполним команды: **Сохранить как** в меню Office или нажмем кнопку  на панели быстрого доступа;

– в окне диалога установим следующие параметры:

– в поле *Папка* выберем папку «Мои документы»;

– в поле *Имя файла* введем имя файла **ww.doc**;

– в поле *Тип файла* выберем тип – документ **Word**;

– щелкнем кнопку «Сохранить».

4. Просмотрим содержимое документа перед печатью:

– Нажмем кнопку  на панели быстрого доступа или же в меню Office, группа **Печать, Предварительный просмотр**

– для возврата в режим редактирования документа щелкнем левой кнопкой мыши кнопку «Заккрыть».

5. Сохраним документ под другим именем **«ww1.doc»**:

– выполним команды: **Сохранить как** в меню Office;

– в поле *Имя файла* введем имя файла - **ww1.doc**.

– щелкнем кнопку «Сохранить».

6. Распечатаем документ:

– выполним команды: **Печать** в меню Office;

– в диалоговом окне *Печать* установим параметры печати;

– щелкнем кнопку «ОК».

7. Закроем файл:

– выполним команды: **Заккрыть** в меню Office.

8. Завершим работу в программе **MS Word**:

– выполним команды: **Выход** 

3. Редактирование документа

1. Загрузим редактор **MS Word**.

2. Откроем документ с именем **«ww.doc»**:

– выполним команды: **Открыть** в меню Office ;

– выберем папку «Мои документы» и файл **ww.doc**;

– щелкнем кнопку «Открыть».

Открытый документ будет иметь вид:

Находите время!

Находите время для размышлений – это источник силы!

Находите время для работы – это условие успеха!
Находите время для мечты – это путь к звездам!
Находите время для любви – это источник радости жизни!
Находите время для веселья – это музыка души!
Находите время для дружбы – это условие счастья!
Находите время для игры – это секрет молодости!

3. Скопируем заголовок «Находите время!» в конец текста:
 - выделим заголовок текста;
 - выполним команды: **Копировать** в меню **Главная** в группе **Буфер обмена**;
 - переместим курсор в конец текста и нажмем клавишу **Enter**;
 - выполним команды: **Вставить** в меню **Главная** в группе **Буфер обмена**.
 4. Поменяем местами слова «Находите» и «Время» в скопированном тексте:
 - выделим слово «Находите»;
 - выполним команды: **Вырезать** в меню **Главная** в группе **Буфер обмена**;
 - установим курсор после слова «Время!»;
 - выполним команды: **Вставить** в меню **Главная** в группе **Буфер обмена**.
 5. Выполним замену слова «время» на слово «времечко»:
 - переместим курсор в начало текста;
 - выполним команды: **Заменить** в меню **Главная** в группе **Редактирование**;
 - в окне диалога установим следующие параметры:
 - в поле *поиск*: время
 - в поле *замена*: времечко;
 - щелкнем кнопку **Заменить все**.
- На экран будет выдано сообщение: «*Произведено 9 замен*», щелкнем кнопку «ОК».
- После выполнения операций документ примет вид:
- Находите времечко!
Находите времечко для размышлений – это источник силы!
Находите времечко для работы – это условие успеха!
Находите времечко для мечты – это путь к звездам!
Находите времечко для любви – это источник радости жизни!
Находите времечко для веселья – это музыка души!
Находите времечко для дружбы – это условие счастья!
Находите времечко для игры – это секрет молодости!
времечко! Находите

4. Форматирование документа

*Предполагается, что в окне редактора выведен текст документа **www.doc**.*

1. Выделим заголовок и выполним команды: **Главная**, в группе **Шрифт**:
 - в окне диалога установим следующие параметры:
 - шрифт – Comic Sans MS;
 - начертание – полужирный;
 - размер – 18;
 - цвет – коричневый;
 - щелкнем кнопку «ОК».

2. Выделим первое предложение текста и выполним операции:
 - выполним команды: щелкнув правой кнопкой мыши выбрать пункт **Абзац**;
 - в окне диалога установим следующие параметры:
 - выравнивание – по центру;
 - отступ слева – 2;
 - отступ справа – 4;
 - первая строка – отступ, на 2 см;
 - междустрочный интервал – двойной;
 - щелкнем кнопку «ОК».
 3. Выделим второе предложение текста и выполним операции:
 - выполним команды: щелкнув правой кнопкой мыши выбрать пункт **Шрифт**;
 - установим в окне диалога следующие параметры:
 - шрифт – Arial;
 - начертание на ваш выбор (желательно использовать все)
 - размер – 16;
 - цвет – синий;
 4. Выделим третье предложение текста и выполним операции:
 - выполним команды: **Главная, Стили** (вызов контекстного меню осуществляется путём нажатия левой кнопки мыши на значек, находящийся в нижнем правом углу **Группы**);
 - в окне диалога установим следующие параметры: выделенная цитата
 5. Выделим четвертое предложение текста и выполним операции:
 - выполним команды: **Главная, Стили** (вызов контекстного меню осуществляется путём нажатия левой кнопки мыши на значек, находящийся в нижнем правом углу **Группы**, изменение стиля происходит путем нажатия левой кнопкой мыши на стрелку находящуюся в строке названия стиля справа);
 - выберите стиль на свое усмотрение
 6. Выделим пятое предложение текста и выполним операции:
 - выполним команды: **Главная, Стили** (вызов контекстного меню осуществляется путём нажатия левой кнопки мыши на значек, находящийся в нижнем правом углу **Группы**) **Заголовок 3**;
 7. Выделим шестое предложение текста и выполним операции:
 - выполним команды: **Главная, Буквица, Положение – в тексте**;
 8. Выделим седьмое предложение текста и выполним операции:
 - выполним команды: **Главная, Стили** (вызов контекстного меню осуществляется путём нажатия левой кнопки мыши на значек, находящийся в нижнем правом углу **Группы**, изменение стиля происходит путем нажатия левой кнопкой мыши на стрелку находящуюся в строке названия стиля справа);
 - в диалогом окне стиль шапка;
- После выполнения операций документ примет вид:

Находите времечко!

Находите времечко для размышлений – это источник силы!

Находите времечко для работы – это условие успеха!

Находите время для мечты – это путь к звездам!

НАХОДИТЕ ВРЕМЕЧКО ДЛЯ ЛЮБВИ – ЭТО ИСТОЧНИК РАДОСТИ ЖИЗНИ!

Находите времечко для веселья – это музыка души!

Н аходите времечко для дружбы – это условие счастья!
Находите времечко для игры – это секрет молодости!
времечко! Находите

9. Создайте выражение **Маша** ^{Таня} **Маня** ^{Ваня} **Ляля**, содержащее верхний и нижний индексы, для этого:
- напишите выражение полностью одним шрифтом;
 - выделяйте поочередно части выражения, которые должны быть индексами (сначала Таня, затем Ваня);
 - выберите команду **Главная**, в группе **Шрифт**, кнопка **Надстрочный знак** или **Подстрочный знак**;
10. Сохраните документ под именем «ww2.doc».
11. Завершите работу в программе MS Word.

5. Вставка рисунков в текст документа

1. Загрузим редактор **MS Word**.
2. Откроем документ с именем «ww1.doc»:
- выполним команды: **Открыть** в меню Office;
 - выберем папку «Мои документы» и файл «ww1.doc»;
 - Щелкнем кнопку «Открыть»

Открытый документ будет иметь вид:

Находите время!

Находите время для размышлений – это источник силы!

Находите время для работы – это условие успеха!

Находите время для мечты – это путь к звездам!

Находите время для любви – это источник радости жизни!

Находите время для веселья – это музыка души!

Находите время для дружбы – это условие счастья!

Находите время для игры – это секрет молодости!

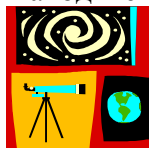
3. Вставим рисунок:
- переместим курсор в начало первой строки;
 - выполним команды: **Вставка, Рисунок**;
 - выберем картинку;
4. Уменьшим рисунок в размерах:
- щелкнем левой кнопки мыши по рисунку для его выделения;
Вокруг рисунка появится маркерная рамка.
 - установим указатель мыши на одном из маркеров;
Указатель примет вид двунаправленной стрелки ↔.
 - нажмем левую кнопку мыши и, не отпуская ее, будем перетаскивать маркер до тех пор, пока рисунок не станет нужных размеров;
 - отпустим кнопку мыши.
5. Аналогично вставим еще два рисунка.
- После выполнения операций документ примет вид:



Находите время!

Находите время для размышлений – это источник силы!

Находите время для работы – это условие успеха!



Находите время для мечты – это путь к звездам!

Находите время для любви – это источник радости жизни!



Находите время для веселья – это музыка души!

Находите время для дружбы – это условие счастья!

Находите время для игры – это секрет молодости!

6. Работа со списками

1. Выполним команды: **Создать** в меню Office
2. Создадим нумерованный список:
 - введем в заголовок списка текст «Список студентов», нажмем клавишу Enter;
 - выберем команды: **Главная, Нумерация** в группе Абзац;
 - выберем стиль списка, щелкнув соответствующий образец (1., 2., 3.);
 - введем элементы списка, нажимая клавишу Enter в конце абзаца:
 - Иванов И.И.;
 - Петров П.П.;
 - Сидоров С.С.;

На экране автоматически появляются номера списка.

После выполнения операций документ примет вид:

Список студентов

1. Иванов И.И.
2. Петров П.П.
3. Сидоров С.С.

3. Преобразуем нумерованный список в маркированный:
 - выделим нумерованный список (без заголовка);
 - выполним команды: **Главная, Маркеры**, выберем маркер на свой выбор;
- После выполнения операций документ примет вид:

Список студентов

- ◆ Иванов И.И.
- ◆ Петров П.П.
- ◆ Сидоров С.С.

4. Создадим многоуровневый список:
 - введем текст заголовка «Структура Университета»;
 - выполним команды: **Главная, Многоуровневый список**;
 - выберем образец многоуровневого списка (1., 1.1., 1.1.1.);
 - наберем текст пункта 1 «Факультет» и нажмем клавишу Enter;
Автоматически продолжится нумерация списка с пункта 2.
 - нажмем правую кнопку мыши для вызова контекстно-зависимого меню;
 - выполним команду **Увеличить отступ**;
 - Нумерация списка изменится с «2.» на «1.1.».*
 - наберем следующий пункт и нажмем клавишу Enter;
 - поступая аналогично и используя пункты контекстно-зависимого меню **Увеличить отступ** и **Уменьшить отступ**, получим многоуровневый список.
- После выполнения операций документ примет вид:

Структура Университета

1. Факультет А
 - 1.1. Специальность 1
 - 1.2. Специальность 2
2. Факультет Б
 - 2.1. Специальность 1
 - 2.2. Специальность 2

5. Сохраним документ под именем «Списки» в папке «Мои документы»:
 - выполним команды: **Сохранить как** в меню Office.

7. Создание оглавления

Для того чтобы сформировать оглавление документа, необходимо все заголовки, которые должны входить в оглавление, отформатировать стилем *Заголовок* (Заголовок 1, Заголовок 2 и т.д.). Для этого:

1. Перед набором заголовков выполним команды: **Главная, Стил**, **Заголовок 1**.
 - Наберем основной текст документа шрифтом Times New Roman размером 12.

Текст документа:

Microsoft Word

Microsoft Word – это текстовый редактор общего назначения. При работе с редактором пользователю предоставляются следующие возможности:

- набор и редактирование текста;
- форматирование текста;
- создание и редактирование таблиц;
- создание списков;
- работа в многооконном режиме.

Microsoft Excel

Microsoft Excel – это электронная таблица, которая используется для обработки табличных данных, построения диаграмм, проведения финансовых статистических расчетов, проведения математического моделирования.

Microsoft Access

Microsoft Access – это база данных. Она состоит из отдельных компонентов, которые используются для хранения и представления информации. Этими компонентами являются таблицы, формы, отчеты, запросы, макросы и модели.

2. Для автоматического формирования заголовка установим курсор в том месте документа, где будет находиться оглавление (в нашем примере перед набранным текстом):

- выполним команды: **Ссылки, Оглавления**, вкладка **Оглавление**.

3. В окне диалога установим следующие параметры:

- уровни – Уровень 1;
- параметры, доступные стили, Заголовок 1, Уровень 1;
- щелкнем кнопку «ОК».

После выполнения операций документ примет вид:

Microsoft Word	1
Microsoft Excel	206
Microsoft Access	206

Microsoft Word

Microsoft Word – это текстовый редактор общего назначения. При работе с редактором пользователю предоставляются следующие возможности:

- набор и редактирование текста;
- форматирование текста;
- создание и редактирование таблиц;
- создание списков;
- работа в многооконном режиме.

Microsoft Excel

Microsoft Excel – это электронная таблица, которая используется для обработки табличных данных, построения диаграмм, проведения финансовых статистических расчетов, проведения математического моделирования.

Microsoft Access

Microsoft Access – это база данных. Она состоит из отдельных компонентов, которые используются для хранения и представления информации. Этими компонентами являются таблицы, формы, отчеты, запросы, макросы и модели.

8. Работа с таблицами

Чтобы создать таблицу в том месте, где находится курсор, достаточно:

- Выбрать в меню **Вставка** опцию **Таблица**. В открывшемся диалоговом окне установить размерность таблицы (число столбцов и строк).

- Можно создавать таблицы, рисуя их «карандашом» с

помощью мыши при выбранном меню  **Нарисовать таблицу**.

Новая таблица состоит из строк и столбцов с пустыми ячейками. Чтобы ввести данные (текст, числа, графика, рисунки, формулы) в ячейку таблицы, щелкните на ячейке (в ней появится текстовый курсор) и наберите нужные данные.

Совет: Если в момент создания таблицы вы еще не знаете, сколько строк вам понадобится, просто создайте таблицу из одной строки. Можно очень легко добавить к концу таблицы новые строки во время ввода текста.

Меню **ТАБЛИЦА** открывает также альтернативные возможности для вставки и изменения таблиц, кроме того, меню позволяет выполнять некоторые дополнительные операции, которые невозможны при использовании интерактивных приемов.

Команды меню «Таблица» и их действия

Команды	Описание
Нарисовать таблицу	Создание новой таблицы
Добавить (Вставить)	Добавление таблицы, строк, столбцов или ячеек
Удалить	Удаление таблицы, выделенных строк, столбцов или ячеек
Выделить	Выделение строки, столбца, ячеек или таблицы целиком
Объединить ячейки	Объединение нескольких выделенных ячеек в одну
Разбить ячейки	Разбиение выделенных ячеек на заданное число строк или столбцов
Разбить таблицу	Разбиение таблицы на две
Автоформат	Автоматическое форматирование таблицы. В списке представлены различные стили оформления таблицы
Автоподбор	Автоматическая настройка ширины и высоты столбцов
Заголовки	Использование выделенной строки в качестве заголовка таблицы, который при печати будет выводиться на каждой странице, если таблица располагается на нескольких листах
Преобразовать	Преобразование таблицы в текст или наоборот
Сортировка	Упорядочивание данных в выделенных строках или списке по алфавиту, величине или дате
Формула	Выполнение математических операций в таблице
Отображать сетку	Команда позволяет включать и отключать отображение сетки из точечных линий
Свойства таблицы	Задание различных параметров таблицы

Вводить данные в таблицу, форматировать и редактировать их по своему усмотрению можно с помощью мыши. Перед тем, как вставлять, удалять или регулировать размеры таблицы, строки, столбца, ячейки необходимо предварительно их выделить с помощью мыши или команд меню **Таблица, Выделить строку** (столбец, ячейку, всю таблицу).

Способы выделения элементов таблицы с помощью мыши

Выделяемый элемент	Способ выделения
Ячейка	Подвести указатель мыши в левый угол ячейки, чтобы он принял вид направленной вправо стрелки  , и щелкнуть левой клавишей мыши
Строка	Щелкнуть левой клавишей мыши на полосе выделения – слева от выделяемой строки
Столбец	Установить указатель мыши в верхней части столбца так, чтобы он принял вид направленной вниз черной стрелки  , и щелкнуть левой клавишей мыши
Вся таблица	Нажать левую кнопку мыши в начале таблицы и протащить указатель мыши до конца таблицы

Чтобы поменять **ширину колонки**, надо взяться мышью за правую ее границу или за движок координатной линейки (указатель мыши примет вид ) и отодвинуть на нужное расстояние. Чтобы поменять **высоту строки**, надо взяться мышью за

горизонтальную линию, отмечающую нижнюю границу строки и подвинуть ее вверх или вниз. Чтобы поменять **ширину выделенной ячейки**, надо взяться мышью за правую или левую ее границу и подвинуть в нужную сторону. Она подвинет соседей и увеличится (уменьшится).

Для **выравнивания** текста в ячейке таблицы **по вертикали** следует установить курсор в нужную ячейку, нажать правую кнопку мыши и выбрать в контекстном меню ячейки команду **Выравнивание** и выбрать нужный шаблон.

Для изменения **направления текста** используется команда контекстного меню **Направление текста**.

Для получения **рамки** внутри и вокруг таблицы необходимо выделить таблицу и использовать:

- кнопки панели инструментов **Таблицы и границы** которая появляется при выделении таблицы упрощает работу с таблицами и позволяет реализовывать некоторые дополнительные возможности их форматирования.

ЗАДАНИЕ: Создать и отредактировать табличное представление информации в документе.

1. Загрузим редактор Microsoft Word.
2. Создадим таблицу. Для этого:
 - выполним команды: **Вставка, Таблица, Вставить таблицу**;
 - в окне диалога установить следующие параметры:
 - ◆ число столбцов – 4;
 - ◆ число строк – 4;
 - ◆ автоподбор ширины столбцов – авто;
 - ◆ щелкнем кнопку «ОК».
 - введем в ячейки таблицы исходные данные.

Таблица примет вид:

1	2	3	5
6	7	8	10
11	12	13	15
16	17	18	20

3. Вставим в таблицу перед последним четвертым столбцом дополнительный столбец. Для этого:
 - выделим последний столбец;
 - выполним команды: **Макет, Вставить слева** в группе **Строки и столбцы**;
 - введем исходные данные в дополнительный столбец.

Таблица примет вид:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

4. Добавим в таблицу дополнительную строку. Для этого:
 - выделим последнюю строку таблицы;

- выполним команды: **Макет, Вставить снизу** в группе Строки и столбцы;
- введем исходные данные в строку.

Таблица примет вид:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

5. Изменим ширину первого столбца. Для этого:

- выделим первый столбец;
- выполним команды: **Макет, в группе Размер ячейки, в строке Ширина столбца** таблицы ввести– 2 см;

Таблица примет вид:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

6. Объединим ячейки с исходными данными – 1, 2, 3, 4, 5. Для этого:

- выделим ячейки 1, 2, 3, 4, 5;
- выполним команды: **Макет, Объединить ячейки** в группе Объединить.

Таблица примет вид:

1				
2				
3				
4				
5				
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

7. В объединенную ячейку введите новый текст «Объединенные ячейки». Для этого:

- выделим объединенную ячейку;
- введем текст «Объединенные ячейки»;
- выполним выравнивание текста по центру .

8. Аналогично объедините ячейки 16 и 21.

9. Удалите информацию из объединенных ячеек 16 и 21.

Таблица примет вид:

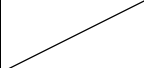
Объединенные ячейки				
6	7	8	9	10
11	12	13	14	15
	17	18	19	20
	22	23	24	25

10. Разобьем объединенную ячейку на 2 части по диагонали. Для этого:

- переместим курсор в объединенную ячейку 16, 21;

- выполним команды: **Конструктор, Нарисовать таблицу;**
- подвести указатель мыши к указанной клетке, при этом он примет форму карандаша .
- нарисуем в ячейке диагональ.

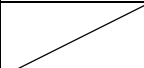
Таблица примет вид:

Объединенные ячейки				
6	7	8	9	10
11	12	13	14	15
	17	18	19	20
	22	23	24	25

11. Разобьем ячейку 24 на 3 части. Для этого:

- выделим ячейку 24
- выполним команды: **Макет, Разбить ячейки;**
- в окне диалога установим **Число столбцов –3, Число строк 1.**

Таблица примет вид:

Объединенные ячейки				
6	7	8	9	10
11	12	13	14	15
	17	18	19	20
	22	23	24	25

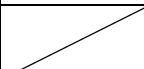
12. Объединим ячейки 13 и 18. Для этого:

- выделим ячейки 13, 18;
- выполним команды: **Макет, Объединить ячейки.**

13. Разобьем объединенные ячейки 13 и 18 на 2 части. Для этого:

- выделим объединенные ячейки 13 и 18;
- выполним команды: **Макет, Разбить ячейки;**
- в окне диалога установим: **Число столбцов –2, Число строк 1.**

Таблица примет вид:

Объединенные ячейки				
6	7	8	9	10
11	12	13	18	14
	17			19
	22	23	24	25

14. Отформатируем таблицу. Для этого:

- данные первого столбца выровняем по центру;
- данные пятого столбца выровняем по правому краю;
- выполним внешнее и внутреннее обрамление таблицы с помощью команд меню: **Конструктор, Границы, Границы и заливка** (таблица должна быть выделена);
- в диалоговом окне задать тип линий, цвет линий, ширину линий, задать границы.
- Нажать «ОК».

15. Оформим внешний вид таблицы. Для этого:

- выделим таблицу;
- выполним команды: **Конструктор, Стили таблицы, Создать стиль таблицы,** выберем в строке Основан на стиле – **Объемная таблица 3;**

- Щелчком по кнопке **Ок**.

Таблица примет вид:

Объединенные ячейки					
6	7	8		9	10
11	12	13	18	14	15
	17			19	20
	22	23		24	25

9. Создание и обработка графических объектов

- Загрузите текстовый процессор Word.
- Введите следующий текст:

Создание и обработка графических объектов

- Создайте овал в документе, для этого:
 - Выполните команды **Вставка, Фигуры, Овал**;
 - расположите курсор (в форме крестика **+**) в месте размещения одного из краев овала;
 - нажмите левую кнопку мыши и протащите указатель до получения овала требуемой формы и размера.
- Самостоятельно создайте прямоугольник и треугольник пользуясь теми же командами.
- Создайте прямоугольник и овал правильной формы (при рисовании нажата клавиша **Shift**).
- Отредактируйте созданные графические объекты, для этого:
 - подведите курсор к любому из объектов, и после того, как курсор примет форму стреловидного креста, щелкните левой клавишей мыши (объект выделен);
 - правой кнопкой мыши вызовите контекстное меню **Формат автофигуры**, измените **Цвет линии, Цвет заливки, Тип линии, Вид линии**;
 - снимите выделение с объекта – выполнив щелчок вне объекта.
- Переместите графические объекты, поменяв их взаимное расположение, для этого:
 - подведите курсор к границе объекта так, чтобы курсор принял форму стреловидного крестика и перетащите объект.
- Создайте схему, соединив между собой два созданных графических объекта стрелкой (с помощью команд **Вставка, Фигуры, Стрелка**).
- Сделайте надпись «нет» над стрелкой, для этого:
 - выполните команды **Вставка, Надпись, Нарисовать надпись**;
 - изобразите прямоугольник на месте будущего слова;
 - напечатайте слово;
 - вызовите контекстное меню правой клавишей мыши, выберите команду **Формат надписи** и во вкладке **Цвета и линии** сделайте установки: *Заливка, цвет - Нет заливки, Линии, цвет - Нет линий*; ОК.
- Сгруппируйте созданную графическую схему, для этого:
 - поочередно выделите эти объекты при нажатой клавише **Shift**;
 - правой кнопкой мыши вызвать контекстное меню;
 - в контекстном меню выберите команду **Группировка, Группировать**.
- Переместите сгруппированный объект на текст.
- Измените размещение объекта относительно текста, для этого:

- Вызовите контекстное меню;
- выберите **Порядок, Поместить за текстом**.
- 13. Разгруппируйте объекты, для этого:
 - выделите сгруппированные объекты;
 - вызовите контекстное меню;
 - выберите команду **Группировка, Разгруппировать**.
- 14. Измените размеры графических объектов, для этого:
 - выделите объект (или группу объектов);
 - установите курсор на один из маркеров выделяющей рамки (форма курсора - двухсторонняя стрелка $\leftrightarrow$) и переместите мышку;
 - выполните предыдущую операцию при нажатой клавише **Shift** или **Ctrl** и просмотрите результат.
- 15. Вставьте в документ и настройте автофигуру, для этого:
 - Выполните команды **Вставка, Фигуры**;
 - выберите любую автофигуру;
 - расположите курсор (в форме крестика **+**) в месте размещения одного из краев фигуры;
 - нажмите левую кнопку мыши и протащите указатель до получения фигуры требуемой формы и размера;
 - измените цвет, заливку, объем и другие параметры фигуры.
- 16. Установите надпись «Информатика» в автофигуру, для этого:
 - выделите фигуру;
 - вызовите контекстное меню (щелчок правой кнопкой мыши по фигуре) и выберите команду **Добавить текст**. Внутри фигуры появится курсор и можно вписывать текст или после выделения фигуры нажмите кнопку **Вставка, Надпись Нарисовать надпись**, изобразите прямоугольник на месте будущего слова, напечатайте слово.
 - вызовите контекстное меню правой клавишей мыши, выберите команду **Формат надписи** и во вкладке **Цвета и линии** сделайте установки: **Заливка, цвет - Нет заливки, Линии, цвет - Нет линий**; ОК
 - после ввода надписи щелкните левой клавишей мыши вне фигуры.

*Если текст не уместается в фигуре, то уменьшите размер выбранного шрифта. Если это не принесло нужного результата, то щелкните по фигуре дважды левой клавишей мыши – появится меню **Формат Автофигуры**. Выберите вкладку **Надпись** и уберите внутренние поля (просветы) в фигуре, возможно текст поместится.*

- 17. Измените направление надписи «Информатика»:
 - выделите фигуру;
 - в контекстном меню выполните команду **Формат Автофигуры, Надпись, Ок**;
 - в открывшейся на вкладке группе Работа с надписями, выбрать вкладку **Формат**, **Направление текста** (нажимать на кнопку направление текста следует до тех пор пока вы не получите нужного направления);
 - щелкните вне фигуры.
- 18. Создайте графический объект произвольной формы, для этого:
 - в меню **Вставка, Фигуры** выберите **Полилиния**;
 - нажав левую кнопку мыши, нарисуйте линию произвольной формы (закончить рисование – двойной щелчок).
- 19. Измените созданный объект:
 - выделите его и выберите в контекстном меню пункт - **Начать изменение узлов**;

- с помощью курсора перетащите точки перегиба в нужном направлении.
- 20. Измените параметры графических объектов, используя пункт **Формат автофигуры** в контекстном меню (самостоятельно).
- 21. Создайте блок-схему изображенную на рисунке 2:



Рис. 2 Учебная блок-схема

22. Сохраните документ под именем **Автофигура**.

10. Запись математических формул

1. Загрузите текстовый процессор Word.

$$\sum_{I=1}^K Z$$

2. Создайте следующий фрагмент формулы $\sum_{I=1}^K Z$. Для этого:

- поместите курсор в точку вставки формулы и выполните **Вставка, Формула** (нажать на стрелку находящуюся в строке названия пункта), выбрать пункт **Вставить новую формулу**;
- в появившейся панели **Работа с формулами** выберите **Конструктор, Крупный оператор**;
- выберите **Шаблон суммы с верхним и нижним пределами** (второй шаблон в первом ряду);
- введите нужный символ, число или выражение в каждый из слотов, предварительно помещая туда курсор;
- Завершите запись формулы (щелчок за пределами рамки).

3. Создайте формулу остатка ряда Тейлора, для этого:

- поместите курсор в точку вставки формулы;
- $$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!} (x-a)^{(n+1)}$$

- выполните команды **Вставка, Формула** (нажать на стрелку находящуюся в строке названия пункта), выбрать пункт **Вставить новую формулу**;
 - в появившейся панели **Работа с формулами** выберите **Конструктор, Индекс**, выберите нужный индекс
 - в слоте ввода формулы в индексе введите соответствующие буквы;
 - поместите курсор в конец введенной формулы и введите =;
 - выберите во вкладке **Работа с формулами** выберите **Конструктор, шаблоны Дробей**, выберите соответствующую дробь;
 - в слоте числителя вставьте шаблон нужного индекса и в каждом окошке наберите соответствующие символы;
 - установите курсор справа от индекса и введите оставшуюся часть числителя ;
 - в слот знаменателя введите **(n+1)!**;
 - установите курсор в конец введенной формулы и введите оставшуюся часть формулы аналогичным способом используя соответствующий индекс;
 - завершите запись формулы.
4. Создайте формулы с фигурными скобками по

образцу:

– Выполните команды **Вставка, Формула**, $y = \begin{cases} a + x, \text{при } x > 0 \\ a - x, \text{при } x \leq 0 \end{cases}$
Вставить новую формулу;

- в слоте ввода введите **y=**;
- выберите в группе **Конструктор** пункт **Скобки** и нажмите соответствующую скобку (первая в группе наборы условий и стопки);
- введите соответствующие выражения на места;
- завершите запись формул.

5. Создайте матричную формулу по образцу:

– выполните команды **Вставка, Формула**,
Вставить новую формулу ;

$$A = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_1 & b_1 & c_1 \\ a_1 & b_1 & c_1 \end{vmatrix}$$

- введите левую часть формулы **A=**;
- в пункте **Скобка** выбрать соответствующую скобку (третья во втором ряду);
- затем щелкнув в появившийся квадрат
- в пункте **Матрица** выбрать соответствующую матрицу;
- в первый слот первой строки вставьте соответствующий шаблон **Индекса**;
- в каждый квадрат записать соответствующие символы;
- аналогичным образом заполните остальные слоты, завершите запись формулы.

Вставка символа

Для того чтобы вставить символ в текст, необходимо дать команду **Вставка, Символ** и выбирается нужный символ.

7. Составьте математическое выражение $\lim f(x)=A \Leftrightarrow \forall \varepsilon > 0 \exists \delta > 0 \dots$, используя символьный набор формул, для этого:

- установите курсор в то место текста, куда нужно вставить символ;
- выберите в меню команду **Вставка, Символ, Другие символы**;
- в поле *Шрифт* установите тип шрифта **Symbol**, выберите нужный символ и нажмите кнопку «Вставить» затем «Заккрыть» и так с каждым символом.

8. Сохраните документ под именем **Формулы**.

Контрольные вопросы и задания

1. Как открыть уже существующий документ.
2. Как сохранить, созданный документ.
3. Каковы основные правила ввода текста.
4. Способы выделения текста. Для чего используется выделение текста.
5. Понятие абзаца.
6. Использование линейки для форматирования текста
7. Какие возможности предоставляет Word для преобразование внешнего вида текста.
8. Способы редактирования структуры таблицы:
 - объединение ячеек;
 - разбиение ячеек;
 - удаление строк и столбцов;
9. Понятие и назначение буфера обмена.
10. Как можно добавить столбцы в таблицу.
11. Как можно изменить ширину и высоту строк и столбцов.
12. Способы вставки рисунков.
13. Возможности WordArt для оформления документов
14. Какие Вы использовали специальные шрифты, и каким образом можно вставить в документ символ?
15. Нумерация страниц.
16. Сохранение документа.

Задание 1. Оформить докладную записку по образцу.

Краткая справка. Верхнюю часть докладной записки оформляйте в виде таблицы (2 столбца и 1 строка; тип линий – нет границ). Этот прием оформления позволит выполнить разное выравнивание в ячейках таблицы: в левой ячейке – по левому краю, в правой – по центру.

Образец задания

Сектор аналитики и экспертизы

Директору Центра ГАНЛ
Н.С.Петрову

ДОКЛАДНАЯ ЗАПИСКА

03.11.2002

Сектор не может завершить в установленные сроки экспертизу проекта маркетингового исследования фирмы «Астра-Н» в связи с отсутствием полных сведений о финансовом состоянии фирмы.

Прошу дать указания сектору технической документации предоставить полные сведения по данной фирме.

Приложение: протокол о некомплектности технической документации фирмы «Астра-Н».

Руководитель сектора
аналитики и экспертизы

(подпись)

М.П.Спелов

Задание 2. Оформить документ, содержащий таблицу, по образцу. Произвести расчет штатной численности по каждой группе оплаты труда.

Краткая справка. Верхнюю часть документа оформите с использованием таблицы (тип линий – нет границ). Произведите расчет суммы по столбцам. Дату вставьте командой *Вставка/Дата*.

ОАО «Прогресс»
03.02.2006 № 38
Москва

Утверждаю
Генеральный директор
_____ Б.Н.Добров
10.02.2006

Структура и штатная численность ОАО «Прогресс» на 2003 г.

Наименование должностей	Штатная численность и группы по оплате труда				
	Первая	Вторая	Третья	Четвертая	Пятая
Генеральный директор	1				
Главный бухгалтер	1				
Сотрудники бухгалтерии		2	2		
Старшие специалисты		3	7	1	
Специалисты			4	5	6
Итого	?	?	?	?	?

Задание 3. Оформить схему по образцу.



Задание 4. Набрать текст и формулу по образцу.

Образец задания

Точки $X_1 = -1$, $X_2 = 5/4$, $X_3 = 2$ делят числовую ось на четыре промежутка.

Найдем знаки произведения на каждом интервале и отметим их на схеме. Решением неравенства $(4X - 5)(X - 2)(X + 1) > 0$ является объединение двух промежутков $[-1; 5/4]$ и $[2; \infty)$.

Решением неравенства является объединение промежутков $[-1; 5/4]$ и $[2; 3]$. Серединами этих промежутков являются числа 0,125 и 2,5.

Ответ: 0,125; 2,5.

Пример.

$$(2X + 1) : (X^2 - Y^2 + 1) > 2/(X - 2),$$

где $Y = (-X)^{1/2}$.

Решение: Область допустимых значений (ОДЗ)

$$\begin{aligned} -X &\geq 0; \Leftrightarrow X \leq 0; \\ X - 2 &\neq 0 \Leftrightarrow X \neq 2; \\ X \leq 0 &\Rightarrow E = [-\infty; 0]. \end{aligned}$$

При $X \in E$ неравенство примет вид

$$\frac{2X + 1}{X^2 + X + 1} > \frac{2}{X - 2} \Leftrightarrow \frac{(2X + 1)(X - 2) - 2(X^2 + X + 1)}{(X^2 + X + 1)(X - 2)} > 0 \Leftrightarrow \frac{-5X - 4}{(X^2 + X + 1)(X - 2)} > 0.$$

Квадратный трехчлен $X^2 + X + 1$ положителен при всех X , так как его дискриминант отрицателен при $(X^2 + X + 1) > 0$, получим равносильное неравенство.

Лабораторная работа № 15 Система создания презентаций MS Power Point

2.7.1 Цель работы: освоить процесс создания презентации в Microsoft PowerPoint на основе пустой презентации.

2.7.2 Задачи работы:

1. При помощи приложения PowerPoint создать презентацию из нескольких слайдов, добавить эффекты мультимедиа.
2. Для закрепления и проверки полученных навыков, необходимо ответить на контрольные вопросы и выполнить дополнительное задание. Результаты работы продемонстрировать преподавателю в электронном виде.

2.7.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Методические указания для выполнения лабораторной работы
3. Microsoft Power Point

2.7.4 Описание (ход) работы:

1. Порядок выполнения работы

1. Запустить Microsoft PowerPoint.
2. В открывшемся окне PowerPoint выбрать ПУСТУЮ ПРЕЗЕНТАЦИЮ. Затем нажать на кнопку ОК.
3. На экране появится окно «Создать слайд» (см. рис. 1).

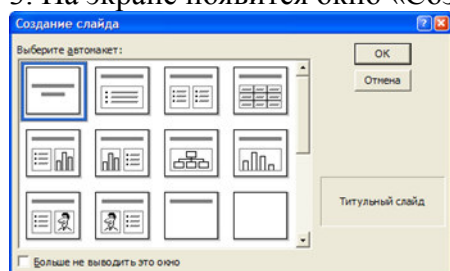


Рис. 1. Окно «Создать слайд»

Окно «Создать слайд» предоставляет пользователю возможность выбора автомакета создаваемого слайда. Новый слайд может содержать заголовок, список, таблицу, диаграмму, рисунок, а также различные сочетания перечисленных элементов.

Первый слайд создаваемой презентации – титульный, содержит заголовок и подзаголовок. Для выбора автомакета слайда необходимо выделить щелчком мышки нужный тип автомакета слайда и нажать кнопку ОК.

На экране появится первый слайд презентации.

Щелчком мышки введите заголовок – название фирмы и подзаголовок – специализацию фирмы. После ввода информации слайд примет вид (см. рис.2).



Рис. 2. Первый слайд презентации

4. Выполните команду ВСТАВКА – НОВЫЙ СЛАЙД. На экране появится окно «Создать слайд» (см. рис. 1). Выберите автомакет слайда – организационная диаграмма (см. рис. 3).



Рис. 3. Тип автомакета слайда «Организационная диаграмма»

5. Введите заголовок слайда – «Структура фирмы». Осуществите двойной щелчок по соответствующей области слайда для ввода организационной диаграммы. На экране появится окно Microsoft Organization Chart.

6. Заполните области диаграммы. В поле **Имя** введите ФИО сотрудника, в поле **Должность** введите должность сотрудника.

7. Выполните команду ФАЙЛ – ЗАКРЫТЬ И ВЕРНУТЬСЯ В ПРЕЗЕНТАЦИЮ, в появившемся диалоговом окне нажать кнопку ОК.

8. В результате выполненных действий второй слайд презентации примет вид (см. рис. 4):



Рис. 4. Второй слайд презентации

9. Выполните команду ВСТАВКА – НОВЫЙ СЛАЙД. На экране появится окно «Создать слайд» (см рис. 1). Выберите автомакет слайда – только заголовок (см. рис. 5).



Рис. 5. Тип автомакета слайда «Только заголовок»

10. Третий слайд презентации содержит заголовок и таблицу Microsoft Excel: прайс – лист компании. Не закрывайте файл презентации. Откройте ваш файл *Фирма_Сластена*. Вставьте новый лист, назовите его *Прайс*. Создайте на листе следующую таблицу:

Наименование	Цена
Конфеты «Белочка»	70р.
Конфеты «Маска»	68р.
Конфеты «Мелодия»	85р.
Конфеты «Костер»	92р.
Конфеты «Слива»	53р.
Конфеты «Василёк»	75р.
Конфеты «Ромашка»	55р.

Перейдите в вашу презентацию. На третий слайд Ведите заголовок слайда - «Прайс - Лист». Для вставки в слайд презентации таблицы Microsoft Excel, выполните следующие действия:

- В электронной таблице (Файл *Фирма_Сластена*, лист *Прайс*) выделите диапазон копируемых ячеек, затем нажмите кнопку **Копировать**;
- Переключитесь в презентацию PowerPoint;
- В меню **Правка** в PowerPoint выберите команду **Специальная вставка**;
- Чтобы вставить ячейки с расчетом последующего изменения их размеров и местоположения, как у рисунка, щелкните **Объект Лист Microsoft Excel**;
- Для хранения только текущего результата установите переключатель **Вставить** (во внедренном рисунке отобразятся только видимые данные ячеек);
- Для обновления результата при изменении исходных данных в Microsoft Excel установите переключатель **Связать** (связанный рисунок будет отражать ширину столбца и прочие форматы исходных ячеек так, как в данный момент в источнике);
- Нажмите кнопку ОК.

11. Выполните команду ВСТАВКА – НОВЫЙ СЛАЙД. На экране появится окно «Создать слайд» (см рис. 2). Выберите тип автомакета слайда – только заголовок (см. рис. 5).

Четвертый слайд презентации содержит заголовок и таблицу Microsoft Excel: движение товаров на складе. Ведите заголовок слайда - «Анализ движения товаров на складе». Для вставки в слайд презентации таблицы Microsoft Excel, выполните действия, указанные в пункте 10. Таблица из файла *Фирма_Сластена* с листа *Склад*.

12. Выполните команду ВСТАВКА – НОВЫЙ СЛАЙД. На экране появится окно «Создать слайд» (см рис. 2). Выберите тип автомакета слайда – только заголовок (см. рис. 5).

Пятый слайд презентации содержит заголовок и диаграмму Microsoft Excel: анализ движения товаров на складе. Ведите заголовок слайда - «Движение товаров на складе». Для вставки в слайд презентации диаграммы Microsoft Excel, выполните следующие действия:

- На листе Microsoft Excel выделите диаграмму, затем нажмите кнопку **Копировать**;
- Переключитесь в презентацию PowerPoint;
- В меню **Правка** в PowerPoint выберите команду **Специальная вставка**;
- Чтобы вставить диаграмму, щелкните **Объект Диаграмма Microsoft Excel**;
- Для хранения только текущего результата установите переключатель **Вставить**;
- Для обновления результата при изменении исходных данных в Microsoft Excel установите переключатель **Связать**;
- Нажмите кнопку ОК.

13. Для улучшения внешнего вида слайдов презентации можно задать красочное оформление слайдам – дизайн презентации из списка имеющихся в Microsoft PowerPoint.

Выполните команду ФОРМАТ – ПРИМЕНИТЬ ШАБЛОН ОФОРМЛЕНИЯ. В открывшемся окне выберите в списке оформление Вашей презентации и нажмите кнопку «Применить».

14. Настройте анимацию каждого слайда.

15. Настройте переход для слайдов презентации.

16. Сохраните презентацию под именем *Фирма_Сластена*.

17. Выполните команду ПОКАЗ СЛАЙДОВ – НАЧАТЬ ПОКАЗ.

СОЗДАНИЕ УПРАВЛЯЮЩИХ КНОПОК.

Для удобства работы некоторые виды презентации можно снабдить управляющими кнопками. В процессе демонстрации они помогут осуществить возврат к предыдущему слайду, переход к следующему слайду, вызов каких-то дополнительных функций.

Мы не будем рассматривать создание всех слайдов и переходов между ними. Обратим внимание только на создание управляющих кнопок. В данной работе 4 вида управляющих кнопок:

- переход к следующему слайду;
- переход к предыдущему слайду;
- выход из презентации;
- звуковое сопровождение.

Алгоритм создания управляющих кнопок.

1. Выбрать пункт *Показ слайдов* в главном меню.
2. Выбрать строчку *Управляющие кнопки*.
3. Выбрать нужную кнопку в предлагаемом наборе.
4. Указать размеры кнопки при помощи мыши.
5. В окне *Настройка действия* необходимо указать: По щелчку мыши или По указанию мыши.
6. Указать вид гиперссылки: следующий слайд, предыдущий слайд, слайд № и т.д.
7. Вместо перехода по гиперссылке можно вызывать любую программу, указав её в наборе.
8. Настроить звук, сопровождающий действие при переходе.

Практические советы.

1. При использовании кнопок необходимо отключить автоматический переход слайдов.
2. Кнопки с их функциями можно копировать.
3. В пункте *Запуск программы* можно указывать любую программу или файл, обрабатываемый приложениями.
18. Покажите результат преподавателю.

2. Контрольные вопросы и задания

1. Для чего предназначено приложение PowerPoint? Как его запустить?
2. Что такое презентация? Что такое слайд?
3. Назвать способы создания презентаций.
4. Что такое шаблон? Какие существуют виды шаблонов в PowerPoint?
5. Какое расширение имеет файл презентации, шаблон презентации?
6. Объясните назначение Режимы слайдов.
7. Назвать основные характеристики Режим сортировщика слайдов.
8. Как выполнить показ презентации? Назвать способы показа презентации.
9. Что такое анимация?
10. Что такое переход слайдов? Примеры перехода слайдов.

Задание:

Создайте презентацию в Power Point на тему «Я»

Содержание презентации: Вступление – Содержание – О себе – Моя семья – Мои друзья – Мое хобби – Заключение, (итого – 7 слайдов).

2.8 Лабораторная работа № 16, 17 (4 часа)

Тема: «Компьютерные сети»

Создание простейшего HTML-документа. Форматирование текста

2.8.1 Цель работы: познакомиться со структурой html-документа и на его основе создать свою страницу.

2.8.2 Задачи работы:

1. Изучить основные тэги форматирования текста и использовать их при создании страницы.
2. Изучить основные тэги для вставки изображений в html-документ и для их преобразования.
3. Результаты работы продемонстрировать преподавателю в электронном виде.

2.8.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер
2. Методические указания для выполнения лабораторной работы
3. Интернет-браузеры (Internet Explorer, Google Chrome и др.)

2.8.4 Описание (ход) работы:

1. Теоретическая часть

Структура html-документа:

HTML-код страницы помещается внутрь контейнера `<HTML> ... </HTML>`, заголовок страницы в контейнер `<HEAD>` Заголовок `</HEAD>`.

Контейнер обязательно содержит открывающий и закрывающий теги.

В разделе описания заголовка можно указать заглавие документа, для этого используется тэг `<TITLE> ... </TITLE>` (имя страницы).

Весь остальной HTML-документ, включая весь текст, содержится внутри тэга `<BODY>` содержание `</BODY>` (содержание страницы).

`<HTML>` {начало страницы}

`<HEAD>` {описание страницы, заголовка}

`<TITLE>` *название* `</TITLE>` {имя страницы}

`</HEAD>` {закрытый тэг описания заголовка}

`<BODY>` {содержание страницы}

текст

`</BODY>` {закрытый тэг описания страницы}

`</HTML>` {конец страницы}

Основные тэги форматирования html-документа:

Заголовок страницы задается с помощью тегов, где размер шрифта заголовка устанавливается с помощью H1 (самый крупный) и до H6 (самый мелкий)

Заголовок страницы целесообразно выделить самым крупным шрифтом: `<H1>` заголовок страницы `</H1>` (например, `<H1>` Все о компьютере `</H1>`)

`</H1>` заголовок страницы `</H6>` - в данном случае заголовок будет записан самым мелким шрифтом;

Если необходимо выровнять этот заголовок по правому краю, то это записывается следующим образом: `<H1 ALIGN = "RIGHT">` Все о компьютере `</H1>`

`<ALIGN = "RIGHT">` {выравнивание текста по правому краю}

`<ALIGN = "CENTER">` {выравнивание текста по центру}

<ALIGN = "LEFT"> {выравнивание текста по левому краю}

Для задания параметров форматирования используется контейнер ..., при этом используя различные атрибуты:

Атрибут FACE позволяет задать гарнитуру шрифта;

Атрибут SIZE – размер шрифта (может принимать иметь значение в диапазоне от 1 до 7);

Атрибут COLOR – цвет шрифта (можно задавать названием цвета (например, "red", "green", "blue" и так далее));

Например, ... (шрифт размера 7, тип шрифта – Arial, цвет - голубой);

Также можно задавать начертание текста или фрагмента текста:

 текст - полужирный шрифт;

<I> текст </I> - начало текста курсивом;

<U> текст </U> - подчёркивание;

Комбинированное выделение осуществляется следующим образом:

<Тэг-1><Тэг-2> фрагмент текста </Тэг-2></Тэг-1>

Например, <U> фрагмент текста </U> (текст полужирный и подчеркнутый);

При создании html-документов можно задавать и цвет самого фона страницы. Эта команда может задаваться только в начале HTML файла и не может быть изменена в дальнейшем. Ее параметры: BGCOLOR - определяет цвет фона документа, TEXT - задает цвет текста для всей страницы. Цвет также задается с помощью его названия ("red", "green", "blue");

Например, <BODY BGCOLOR = "blue" TEXT = "red"> - задание цвета фона и текста (фон голубого цвета, цвет шрифта - красный)

Списки:

Ненумерованный список располагается внутри контейнера ..., а каждый элемент списка определяется также тэгом . С помощью атрибута TYPE тэга можно задать вид маркера списка: "disc" (диск),

"square" (квадрат), "circle" (окружность). Например,

(маркированный список, маркер в виде окружности)

<UL TYPE = CIRCLE>

 текст

 текст

 текст

2. Порядок выполнения работы

1. Создать, используя только тэги структуры, текстовый html-документ. Сохранить этот документ под любым именем с расширением .htm в папке RABOCHLAB в директории HTMLLAB. Открыть документ в окне браузера и посмотреть, как он будет отформатирован.
2. Заголовок страницы должен быть выровнен по центру, начертание – полужирный шрифт, тип шрифта - Comic Sans MS, размер – 1, цвет шрифта – голубой, подчёркивание;
3. Задать цвет фона страницы – бирюзовый цвет;
4. Текст должен состоять из трех абзацев, один абзац должен быть выровнен по левому краю, второй – по центру, а третий – по правому краю;
5. Цвет текста в первом абзаце задать белым цветом, тип шрифта - Monotype Corsiva, размер – 7;

6. Цвет текста во втором абзаце задать розовым цветом, тип шрифта - Arial, размер – 6;
7. Цвет текста в первом абзаце задать желтым цветом, тип шрифта - Calibri, размер – 5;
8. В нижнем правом углу создать маркированный список, состоящий из трех компонентов, цвет текста задать зеленым цветом;

Листинг html-программы:

```
<HTML>
<HEAD>
<TITLE> Компьютер </TITLE>
</HEAD>
<BODY bgcolor='orange'>
  <font face='Comic Sans Ms' color='red'>
    <H1 ALIGN = 'CENTER'> <u> Все о компьютере </u> </H1>
  </font>
  <p align='left'> <font face='Monotype Corsiva' color='white' size=7> На этом сайте вы
  сможете получить различную информацию о компьютере, его программном обеспечении
  и ценах на компьютерные комплектующие. </font></p>
  <p align='center'> <font face='Arial' color='lime' size=6> Терминологический словарь
  познакомит вас с компьютерными терминами, а также вы сможете заполнить анкету.
  </font> </p>
  <p align='right'> <font face='Calibri' color='yellow' size=5> На этом сайте вы можете
  узнать много новой и полезной информации для вас.</p>
  <font color='aqua'>
    <ul type='circle'>
      <li> <b> Компьютер </b> </li>
      <li> <b> Терминологический словарь </b> </li>
      <li> <b> Цены на товары </b> </li>
    </ul>
  </font>
</BODY>
</HTML>
```

Вставка изображений в html-документ

1. Теоретическая часть

```
<IMG SRC = “имя файла.jpg (gif, png)”>
<IMG SRC = “C:\Documents and Settings\User\ Мои документы\...jpg”>
<IMG SRC = “ имя файла. jpg (gif, png)” ALIGN = “RIGHT”> выравнивание по правому
краю
<IMG SRC = “ имя файла. jpg (gif, png)” ALIGN = “LEFT”> выравнивание по левому краю
<IMG SRC = “ имя файла. jpg (gif, png)” ALIGN = “TOP”> выравнивание текста по
верхней кромке изображения
<IMG SRC = “ имя файла. jpg (gif, png)” ALIGN = “BOTTOM”> выравнивание текста по
левой кромке изображения
<IMG SRC = “ имя файла. jpg (gif, png)” ALIGN = “MIDDLE”> выравнивание текста по
средней кромке изображения
<IMG SRC = “ имя файла. jpg (gif, png)” BORDER = “3”> обрамление графического
объекта
<IMG SRC = “ имя файла. jpg (gif, png)” HSPACE = “30” VSPACE = “30”> вставка пустой
области вокруг изображения
```

 изменение размера изображения
 <EMBED SRC = “ имя файла. wav”> вставка звуковых объектов
 <EMBED SRC = “ имя файла. wav WIDTH = “ ” HEIGHT = “ ”> атрибуты для задания размеров экранных элементов

2. Порядок выполнения работы

1. Вставить изображение в текст:



выравнивание текста по центру рисунка



выравнивание текста по нижнему краю рисунка

2. Использовать рамки вокруг рисунка:



рисунок без рамки



рамка 2 пикселя

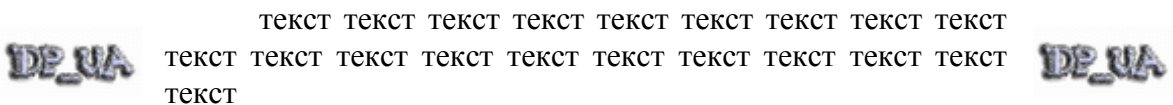


рамка 5 пикселей



рамка 10 пикселей

3. Создать плавающие рисунки с обтекающим текстом вокруг них:



4. Изменить размеры изображения:



- a. Без изменения размера



- b. Параметр height=60



- c. Параметр width=55



- d. Не пропорциональное изменения размера: height=40 width=80

5. Создание плавающих рисунков с обтеканием текста вокруг них и отступом в 10 и 20 пикселей вокруг них
6. Вставить звук на страницу.
7. Сохранить этот документ под любым именем с расширением .htm в папке RABOCHLAB в директории HTMLLAB.

Листинг html-программы:

```
<HTML>
<HEAD>
<TITLE> Компьютер </TITLE>
</HEAD>
<BODY bgcolor='«#008080'>
```

```

<font face='Comic Sans Ms' color='blue'>
<H1 ALIGN = 'CENTER'> <u> Все о компьютере </u> </H1>
</font>
<img src='D:\работа, учеба\для курсовика\dp_ua.gif' style='vertical-align:top'
border='1'>
<font face='Monotype Corsiva' color='white' size=2> На этом сайте вы сможете
получить различную информацию о компьютере, его программном обеспечении и ценах
на компьютерные комплектующие.</font> <br>
<img src='D:\работа, учеба\для курсовика\dp_ua.gif' style='vertical-align:middle'
border='1'>
<font face='Arial' color='pink' size=2> Терминологический словарь познакомит вас с
компьютерными терминами, а также вы сможете заполнить анкету. </font> <br>
<img src='D:\работа, учеба\для курсовика\dp_ua.gif' border='1'>
<font face='Calibri' color='yellow' size=2> На этом сайте вы можете узнать много
новой и полезной информации для вас.</font> <br>
<hr>
<img src='D:\работа, учеба\для курсовика\dp_ua.gif'> <br>
<img src='D:\работа, учеба\для курсовика\dp_ua.gif' border='2'> <br>
<img src='D:\работа, учеба\для курсовика\dp_ua.gif' border='5'> <br>
<img src='D:\работа, учеба\для курсовика\dp_ua.gif' border='10'> <br>
<hr>
<img src='D:\работа, учеба\для курсовика\dp_ua.gif' align='left'> <img src='D:\работа,
учеба\для курсовика\dp_ua.gif' align='right'> <font color='red' size=6><b> Процессор
</b></font> - это центральное устройство, производящее обработку информации в
двоичном коде.
<hr>
<img src='D:\работа, учеба\для курсовика\для лаб.jpg' width='39' height='53'><br>
<img src='D:\работа, учеба\для курсовика\для лаб.jpg' width='39' height='60'><br>
<img src='D:\работа, учеба\для курсовика\для лаб.jpg' width='55' height='53'><br>
<img src='D:\работа, учеба\для курсовика\для лаб.jpg' width='80' height='40'><br>
<hr>
<font face='Calibri' color='pink' size=5>
Публикации во Всемирной паутине реализуется с помощью web-сайтов.
<img src='D:\работа, учеба\для курсовика\dp_ua.gif' align='left' hspace='10'
vspace='10'>
<img src='D:\работа, учеба\для курсовика\dp_ua.gif' align='right' hspace='10'
vspace='10'>
Web-сайт по своей структуре напоминает журнал, который содержит информацию,
посвященную какой-либо теме или проблеме. Как журнал состоит из печатных страниц,
так и web-сайт состоит из компьютерных web-страниц. Сайт является интерактивным
средством представления информации. Интерактивность сайта обеспечивают различные
формы, с помощью которых посетитель сайта может
<img src='D:\работа, учеба\для курсовика\dp_ua.gif' align='left' hspace='20'
vspace='20'>
<img src='D:\работа, учеба\для курсовика\dp_ua.gif' align='right' hspace='20'
vspace='20'>
зарегистрироваться на сайте, заполнить анкету и так далее. Обычно сайт имеет
титильную страницу (страницу с оглавлением), на которой имеются гиперссылки на его
основные разделы. Гиперссылки также имеются на других web-страницах сайта, что
обеспечивает возможность пользователю свободно перемещаться по сайту. Web-сайты
являются мультимедийными, так как кроме текста могут содержать иллюстрации,
анимацию.</font>

```

```
<EMBED SRC = “ имя файла. wav WIDTH = “ ” HEIGHT = “ ”>  
</BODY>  
</HTML>
```

Разработал _____

Н.В. Андреева