МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования "Юго-Западный государственный университет" (ЮЗГУ)

Кафедра биомедицинской инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«31» 03 2023 г.

Медицинская информатика – лабораторный практикум

Методические рекомендации по выполнению практических работ для студентов специальности 30.05.03 «Медицинская кибернетика»

Курск 2023

УДК 004

Составители: М.В. Артеменко

Рецензент

Кандидат технических наук, доцент кафедры программной инженерии ЮЗГУ Белова Т.М.

Методические рекомендации по выполнению практических работ для студентов специальности 30.05.03 «Медицинская кибернетика» / Юго-Зап. гос. ун-т; сост. М.В. Артеменко − Курск, 2023. − 360 с.

Методические указания: содержат краткие теоретические сведения, порядок выполнения и содержание отчета по практическим работам по дисциплине «Медицинская информатика», касающиеся приобретение знаний и навыков студентами в части: программирования на языках высокого уровня, работы в медицинских информационных системах и базах данных, математического имитационного моделирования, обработки символьной информации, работе в электронных таблицах. Рассмотрены примеры решения различных задач на языках высокого уровня: Паскаль, Бейсик, Java, Pyton. Содержание методических указаний соответствуют требованиям Федерального государственного образовательного стандарта высшего образования подготовки специалистов 30.05.03 «Медицинская кибернетика», утвержденной учебно-методическим объединением.

Текст печатается в авторской редакции

Подписано в печать Формат 60х84х 1/16.
Усл.печ.л. . Уч.-изд.л. . Тираж _____ экз. Заказ. Бесплатно.
Юго-Западный государственный университет.
305040, г.Курск, ул. 50 лет Октября,9 4

Содержание

1. ОБЩЕЕ УСТРОЙСТВО ПЭВМ. АППАРАТНЫЙ СОСТАВ ПК. МОДУЛЬНЫЙ ПРИНЦИП ПОСТРОЕНИЯ ЭВМ.

Цель работы: ознакомление с базовыми принципами построения персональной вычислительной техники, применяемой для обработки информации (персональных компьютеров).

Краткие теоретические сведения.

Сегодня невозможно представить себе современную медицину без использования компьютеров, так как они являются неотъемлемым рабочим инструментом в различных сферах медицинской деятельности. Внедрение компьютерных технологий в медицину обеспечило высокую точность и скорость проведения различных исследований и медицинских осмотров.

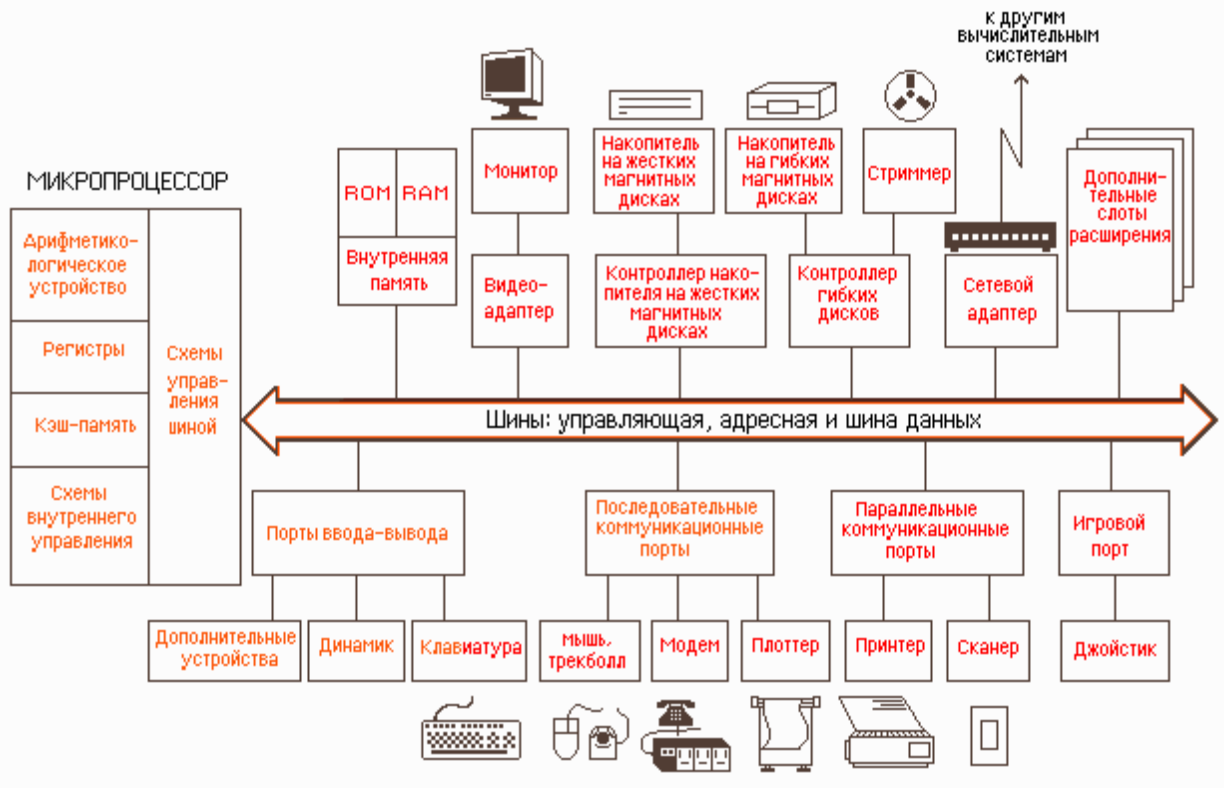
Медицина – одна из сложнейших наук, и в большинстве случаев даже самому лучшему специалисту бывает сложно поставить точный диагноз заболевания. В таких случаях компьютерная помощь существенно облегчает работу врача, так как результаты обследований пациента, переданные компьютеру, моментально обрабатываются с выявлением аномальных результатов анализа, и уже через несколько минут можно получить полные сведения о возможном диагнозе. Решающее слово всегда остается за врачом, но помощь компьютера значительно ускоряет процесс принятия правильного решения, от которого зачастую зависит здоровье, а иногда, и жизнь пациента. В современных медицинских учреждениях врачи давно перешли от бумажной работы к работе с компьютерами, в которых хранится необходимая информация об истории болезней всех пациентов, что позволяет медработникам уделять больше времени и внимания больным, а не «возне» с бумагами. Кроме того, современные компьютерные технологии помогают врачу эффективно и оперативно проводить профилактические осмотры. Так, например, медицинский прибор кошка-сканер является одним из наиболее безболезненных и точных методов изучения внутренних органов человека.

Это лишь несколько частных примеров использования компьютеров в медицине, а если копнуть глубже, можно увидеть, что использование компьютерной техники играет важнейшую роль в медицинских исследованиях. С помощью компьютеров можно изучать возможные последствия ударов для позвоночника и черепа человека при автомобильных катастрофах. Медицинские базы данных позволяют специалистам быть всегда в курсе современных научно-практических достижений. Компьютерные сети также широко используются для обмена информацией о донорских органах, в которых нуждаются критические пациенты, ожидающие трансплантации. Кроме того, компьютеры являются идеальным инструментом для обучения медработников. В таких случаях компьютеры «играют роль больного» и на основании выданных им симптомов, ассистент должен определить диагноз и назначить курс лечения. В случае ошибки обучающегося компьютер незамедлительно

отобразит ее и укажет на источник отклонения. Без компьютеров не обходятся и эпидемиологические службы, которые использует ЭВМ для создания эпидемиологических карт, позволяющих следить за скоростью и направлением распространения эпидемий. Говорить о пользе компьютеров в медицине можно долго, но никогда заключение без эмоционального компьютера не сможет сравниться с важным решением, которое должен принять человек.

Общая схема построения компьютера

Общая схема современного компьютера показана на следующем рисунке.



Любой компьютер построен на общих принципах, которые позволяют выделить следующие главные устройства:

- память (запоминающее устройство, ЗУ), состоящую из перенумерованных ячеек;
- процессор, включающий в себя устройство управления (У У) и арифметико-логическое устройство (АЛУ);
- устройства ввода-вывода;
— устройство вывода.

Эти устройства соединены каналами связи, по которым передается информация. В основу построения подавляющего большинства компьютеров положены следующие общие принципы, которые сформулировал в 1945 г. Джон фон Нейман.

Принцип программного управления. Из него следует, что программа состоит из набора команд, которые выполняются процессором автоматически друг за

другом в определенной последовательности. Процессор исполняет программу автоматически, без вмешательства человека.

Принцип однородности памяти. Программы и данные хранятся в одной и той же памяти. Компьютер не различает, что хранится в данной ячейке памяти — число, текст или команда. Над командами в памяти можно выполнять такие же действия, как и над данными. Таким образом, команды одной программы могут быть получены как результаты исполнения другой программы. На этом принципе основаны методы трансляции — перевода текста программы с языка программирования высокого уровня на язык конкретной машины.

Принцип адресности. Структурно основная память состоит из пронумерованных ячеек, процессору в произвольный момент времени доступна любая ячейка.

Основные устройства компьютера находятся в системном блоке. К ним относятся: материнская плата, процессор, видеокарта, оперативная память, жесткий диск.

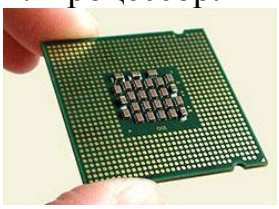
Системный блок.

К первой категории относятся устройства, или их еще называют комплектующие, которые «прячутся» в системной блоке. Они наиболее важны для его работы.

1. Материнская плата.

Материнская плата – это печатная плата, которая предназначена для подключения основных комплектующих компьютера. Часть из них, например, процессор или видеокарта устанавливается непосредственно на саму материнскую плату в предназначенный для этого разъем. А другая часть комплектующих, к примеру, жесткий диск или блок питания, подключается к материнской плате с помощью специальных кабелей.

2. Процессор.



Процессор – это микросхема и одновременно «мозг» компьютера. Чем лучше процессор тем быстрее он будет выполнять эти самые операции, соответственно компьютер будет работать быстрее.

3. Видеокарта.

Видеокарта или по-другому графический плата, предназначена для вывода картинки на экран монитора. Она также устанавливается в материнскую плату, в специальный разъем PSI-Express. Реже видеокарта может быть встроена в саму материнку, но её мощности чаще всего хватает только для офисных приложений и работы в интернете.

4. Оперативная память.

Оперативная память – это прямоугольная планка, похожа на картридж от старых игровых приставок. Она предназначена для временного хранения данных. Информация о запущенных программах, спящий режим компьютера и другие временные данные хранятся в оперативной памяти. Особенностью оперативки является то, что данные из неё после выключения компьютера полностью удаляются.

5. Жесткий диск.

Жесткий диск предназначен для длительного хранения файлов. Его называют винчестер. Он хранит данные на специальных пластинах. Также в последнее время распространились SSD диски.

6. Дисковод.

Дисковод осуществляет работу с дисками. Как минимум дисковод необходим для первоначальной установки системы.

7. Системы охлаждения.

Система охлаждения – это вентиляторы или кулеры, которые охлаждают комплектующие. Обычно установлено три и более кулеров. Обязательно один на процессоре, один на видеокарте, и один на блоке питания, а далее уже по желанию. Устанавливаются также вентиляторы на жесткие диски и в самом корпусе. Если кулер в корпусе установлен на передней панели, то он забирает тепло, а кулеры установленные на заднем отсеке подают в «системник» холодный воздух.

8. Звуковая карта.

Звуковая карта выводит звук на колонки. Обычно она встроена в материнскую плату. Но бывает, что она либо ломается, и поэтому покупается отдельно, либо же изначально качество стандартной владельца ПК не устраивает и он покупает другую звуковуху. В общем звуковая карта также имеет право быть в этом списке устройств для ПК.

9. Блок питания.

Блок питания нужен для того, чтобы все вышеописанные устройства компьютера заработали. Он обеспечивает все комплектующие электроэнергией необходимой мощности.

10. Корпус

Чтобы материнскую плату, процессор, видеокарту, оперативную память, жесткий диск, дисковод, звуковую карту, блок питания и возможно какие-то дополнительные комплектующие можно было оптимальным образом разместить необходим корпус, в который все это аккуратно устанавливается, закручивается, подключается и начинает ежедневную жизнь, от включения до выключения. В корпусе поддерживается необходимая температура, и все защищено от повреждений.

В итоге получается полноценный системный блок, со всеми важнейшими устройствами компьютера, которые нужны для его работы.

Периферийные устройства.

Чтобы полноценно использовать возможности компьютера применяются внешние или периферийные устройства. К ним относятся те компоненты компьютера, которые находятся за пределами «системника».

1. Монитор.

2. Клавиатура.

Клавиатура предназначена для ввода информации.

3. Мышь.

Мышь необходима для управления курсором на экране и выполнения ряда операций.

4. Ауди - колонки.

Колонки необходимы для воспроизведения звука и звуковых эффектов.

5. Принтер и сканер или МФУ.

Принтер и сканер предназначены для печати и сканирования документов. МФУ – многофункциональное устройство объединяет в себе функции принтера и сканера.

Персональная ЭВМ является основной составляющей автоматизированного рабочего места врача - кибернетика.

Состав вычислительной системы (конфигурация) включает в себя аппаратные и программные средства. Ее функционирование определяется определенной архитектурой.

Архитектура ПК - это совокупность аппаратных и программных средств ПК, а также система взаимодействия их, обеспечивающая функционирование ПК. Основные особенности архитектуры ПК - открытость и модульность.

Открытость означает возможность замены отдельных компонентов ПК их более совершенными версиями, а также возможность подключения новых устройств к компьютеру с целью расширения его возможностей. Все компоненты компьютера оформлены в виде законченных конструкций – модулей, имеющих стандартные размеры и стандартные средства соединения с ЭВМ. Они не связаны жестко, в единое неразъемное устройство: предусмотрена возможность быстрого подсоединения и отсоединения любого из них к ПК. Кроме того, в любой ЭВМ подобного типа используется стандартный набор основных модулей, при любой её модификации.

Архитектура компьютера обычно определяется совокупностью ее свойств, существенных для пользователя. Основное внимание при этом уделяется структуре и функциональным возможностям машины, которые можно разделить на основные и дополнительные.

Основные функции определяют назначение ЭВМ: обработка и хранение информации, обмен информацией с внешними объектами. Дополнительные функции повышают эффективность выполнения

основных функций: обеспечивают эффективные режимы ее работы, диалог с пользователем, высокую надежность и др. Названные функции ЭВМ реализуются с помощью ее компонентов: аппаратных и программных средств.

Структура компьютера – это некоторая модель, устанавливающая состав, порядок и принципы взаимодействия входящих в нее компонентов.

Персональный компьютер - это настольная или переносная ЭВМ, удовлетворяющая требованиям общедоступности и универсальности применения.

Достоинствами ПК являются:

• малая стоимость, находящаяся в пределах доступности для индивидуального покупателя;

• автономность эксплуатации без специальных требований к условиям окружающей среды;

• гибкость архитектуры, обеспечивающая ее адаптивность к разнообразным применениям в сфере управления, науки, образования, в быту;

• "дружественность" операционной системы и прочего программного обеспечения, обусловливающая возможность работы с ней пользователя без специальной профессиональной подготовки;

• высокая надежность работы (более 5 тыс. ч наработки на отказ).

Структура персонального компьютера

Микропроцессор (МП). Это центральный блок ПК, предназначенный для управления работой всех блоков машины и для выполнения арифметических и логических операций над информацией. В состав микропроцессора входят:

• устройство управления (УУ) – формирует и подает во все блок машины в нужные моменты времени определенные сигналы управления (управляющие импульсы), обусловленные спецификой выполняемой операции и результатами предыдущих операций; формирует адреса ячеек памяти, используемых выполняемой операцией, и передает эти адреса в соответствующие блоки ЭВМ; опорную последовательность импульсов устройство управления получает от генератора тактовых импульсов;

• арифметико-логическое устройство (АЛУ) - предназначено для выполнения всех арифметических и логических операций над числовой и символьной информацией (в некоторых моделях ПК для ускорения выполнения операций к АЛУ подключается дополнительный математический сопроцессор)',

• микропроцессорная память (МПП) - служит для кратковременного хранения, записи и выдачи информации, непосредственно используемой в вычислениях в ближайшие такты работы машины.

МПП строится на регистрах и используется для обеспечения высокого быстродействия машины, ибо основная память (ОП) не всегда обеспечивает скорость записи, поиска и считывания информации, необходимую для эффективной работы быстродействующего микропроцессора.

Регистры - быстродействующие ячейки памяти различной длины (в отличие от ячеек ОП, имеющих стандартную длину 1 байт и более низкое быстродействие);

• интерфейсная система микропроцессора – реализует сопряжение и связь с другими устройствами ПК; включает в себя внутренний интерфейс МП, буферные запоминающие регистры и схемы управления портами ввода-вывода (ПВВ) и системной шиной. Интерфейс (interface) – совокупность средств сопряжения и связи устройств компьютера, обеспечивающая их эффективное взаимодействие.

Порт ввода-вывода (I/O – Input/Output port) – аппаратура сопряжения, позволяющая подключить к микропроцессору другое устройство ПК.

Генератор тактовых импульсов. Он генерирует последовательность электрических импульсов; частота генерируемых импульсов определяет тактовую частоту машины. Промежуток времени между соседними импульсами определяет время одного такта работы машины или просто такт работы машины. Частота генератора тактовых импульсов является одной из основных характеристик персонального компьютера и во многом определяет скорость его работы, ибо каждая операция в машине выполняется за определенное количество тактов.

Системная шина. Это основная интерфейсная система компьютера, обеспечивающая сопряжение и связь всех его устройств между собой. Системная шина включает в себя:

• кодовую шину данных (КШД), содержащую провода и схемы сопряжения для параллельной передачи всех разрядов числового кода (машинного слова) операнда;

• кодовую шину адреса (КША), включающую провода и схемы сопряжения для параллельной передачи всех разрядов кода адреса ячейки основной памяти или порта ввода-вывода внеш н его устройства;

• кодовую шину инструк ц ий (КШИ), содержащую провода и схемы сопряжения для передачи инструкций (управляющих сигналов, импульсов) во все блоки машины;

• шину питания, имеющую провода и схемы сопряжения для подключения блоков ПК к системе энергопитания. Системная шина обеспечивает три направления передачи информации:

1) между микропроцессором и основной памятью;

2) между микропроцессором и портами ввода-вывода внешних устройств;

3) между основной памятью и портами ввода-вывода внешних устройств (в режиме прямого доступа к памяти).

Все блоки, а точнее их порты ввода-вывода, через соответствующие унифицированные разъемы (стыки) подключаются к шине единообразно: непосредственно или через контроллеры (адаптеры). Управление системной шиной осуществляется микропроцессором либо непосредственно, либо, что чаще, через дополнительную микросхему – контроллер шины, формирующий основные сигналы управления. Основная память (ОП). Она предназначена для хранения и оперативного обмена информацией с прочими блоками машины. ОП

содержит два вида запоминающих устройств: постоянное запоминающее устройство (ПЗУ) и оперативное запоминающее устройство (ОЗУ).

ПЗУ служит для хранения неизменяемой (постоянной) программной и справочной информации, позволяет оперативно только считывать хранящуюся в нем информацию (изменить информацию в ПЗУ нельзя).

ОЗУ предназначено для оперативной записи, хранения и считывания информации (программ и данных), непосредственно участвующей в информационно-вычислительном процессе, выполняемом ПК в текущий период времени. Главными достоинствами оперативной памяти являются ее высокое быстродействие и возможность обращения к каждой ячейке памяти отдельно (прямой адресный доступ к ячейке). В качестве недостатка ОЗУ следует отметить невозможность сохранения информации в ней после выключения питания машины (энергозависимость).

Внешняя память. Она относится к внешним устройствам ПК и используется для долговременного хранения любой информации, которая может когда-либо потребоваться для решения задач. В частности, во внешней памяти хранится все программное обеспечение компьютера. Внешняя память содержит разнообразные виды запоминающих устройств, но наиболее распространенными, имеющимися практически на любом компьютере, являются накопители на жестких (НЖМД) и гибких (НГМД) магнитных дисках. Назначение этих накопителей – хранение больших объемов информации, запись и выдача хранимой информации по запросу в оперативное запоминающее устройство. Различаются НЖМД и НГМД лишь конструктивно, объемами хранимой информации и временем поиска, записи и считывания информации.

В качестве устройств внешней памяти используются также запоминающие устройства на кассетной магнитной ленте (стриммеры), накопители на оптических дисках (CD - ROM – Compact Disk Read Only Memory – компакт-диск с памятью, только читаемой) и др.

Источник питания. Это блок, содержащий системы автономного и сетевого энергопитания ПК.

Таймер. Это внутримашинные электронные часы, обеспечивающие при необходимости автоматический съем текущего момента времени (год, месяц, часы, минуты, секунды и доли секунд). Таймер подключается к автономному источнику питания – аккумулятору и при отключении машины от сети продолжает работать.

Внешние устройства (ВУ). Это важнейшая составная часть любого вычислительного комплекса. Достаточно сказать, что по стоимости ВУ иногда составляют 50 – 80 % всего ПК. От состава и характеристик ВУ во многом зависят возможность и эффективность применения ПК в системах управления и в народном хозяйстве в целом. ВУ ПК обеспечивают взаимодействие машины с окружающей средой: пользователями, объектами управления и другими ЭВМ. ВУ весьма разнообразны и могут быть классифицированы по ряду признаков. Так, по назначению можно выделить следующие виды ВУ:

• внешние запоминающие устройства (ВЗУ) или внешняя память ПК;
• диалоговые средства пользователя;
• устройства ввода информации;
• устройства вывода информации;
• средства связи и телекоммуникации.

Диалоговые средства пользователя включают в свой состав видеомониторы (дисплеи), реже пультовые пишущие машинки (принтеры с клавиатурой) и устройства речевого ввода-вывода информации.

Видеомонитор (дисплей)– устройство для отображения вводимой и выводимой из ПК информации.

Устройства речевого ввода-вывода относятся к быстроразвивающимся средствам мультимедиа. Устройства речевого ввода – это различные микрофонные акустические системы, "звуковые мыши", например, со сложным программным обеспечением, позволяющим распознавать произносимые человеком буквы и слова, идентифицировать их и закодировать.

Устройства речевого вывода – это различные синтезаторы звука, выполняющие преобразование цифровых кодов в буквы и слова, воспроизводимые через громкоговорители (динамики) или звуковые колонки, подсоединенные к компьютеру.

К устройствам ввода информации относятся:

• клавиатура – устройство для ручного ввода числовой, текстовой и управляющей информации в ПК;
• графические планшеты (диджитайзеры) – для ручного ввода графической информации, изображений путем перемещения по планшету специального указателя (пера); при перемещении пера автоматически выполняются считывание координат его местоположения и ввод этих координат в ПК;
• сканеры (читающие автоматы) – для автоматического считывания с бумажных носителей и ввода в ПК машинописных текстов, графиков, рисунков, чертежей; в устройстве кодирования сканера в текстовом режиме считанные символы после сравнения с эталонными контурами специальными программами преобразуются в коды ASCII , а в графическом режиме считанные графики и чертежи преобразуются в последовательности двухмерных координат;
• манипуляторы (устройства указания): джойстик –рычаг, мышь, трекбол – шар в оправе, световое перо и др. – для ввода графической информации на экран дисплея путем управления движением курсора по экрану с последующим кодированием координат курсора и вводом их в ПК;

К устройствам вывода информации относятся:

• принтеры – печатающие устройства для регистрации информации на бумажный носитель;
• графопостроители (плоттеры)– для вывода графической информации (графиков, чертежей, рисунков) из ПК на бумажный носитель; плоттеры бывают векторные с вычерчиванием изображения с помощью пера и растровые:

термографические, электростатические, струйные и лазерные. По конструкции плоттеры подразделяются на планшетные и барабанные.

Устройства связи и телекоммуникации используются для связи с приборами и другими средствами автоматизации (согласователи интерфейсов, адаптеры, цифро- аналоговые и аналого-цифровые преобразователи и т.п.) и для подключения ПК к каналам связи, к другим ЭВМ и вычислительным сетям (сетевые интерфейсные платы, "стыки", мультиплексоры передачи данных, модемы).

Средства мультимедиа (multimedia – многосредовость) - это комплекс аппаратных и программных средств, позволяющих человеку общаться с компьютером, используя самые разные, естественные для себя среды: звук, видео, графику, тексты, анимацию и др.

Дополнительные схемы. К системной шине и к МП ПК наряду с типовыми внешними устройствами могут быть подключены и некоторые дополнительные платы с интегральными микросхемами, расширяющие и улучшающие функциональные возможности микропроцессора: математический сопроцессор, контроллер прямого доступа к памяти, сопроцессор ввода-вывода, контроллер прерываний и др.

Математический сопроцессор широко используется для ускоренного выполнения операций над двоичными числами с плавающей запятой, над двоично-кодированными десятичными числами, для вычисления некоторых трансцендентных, в том числе тригонометрических, функций. Математический сопроцессор имеет свою систему команд и работает параллельно (совмещено во времени) с основным МП, но под управлением последнего. Ускорение операций происходит в десятки раз. Последние модели МП, начиная с МП 80486 DX , включают сопроцессор в свою структуру.

Основные периферийные устройства (ПУ) подключаются к СШ не непосредственно, а через цепочку адаптер ПУ (АПУ) – порт ввода вывода (ПВВ). АПУ выполняет две основные функции:

- осуществляет непосредственное управление ПУ по запросам от МП, освобождая тем самым последний от выполнения рутинных операций;
- обеспечивает согласование интерфейса ПУ с СШ.

Остановимся несколько подробнее только на второй функции. Периферийное оборудование выпускается многими производителями, зачастую без ориентации на то или иное семейство ПЭВМ. Кроме того каждый тип ПУ обладает своей спецификой. Другими словами, ПУ имеют различные интерфейсы (соглашения о связях с другими устройствами). Поэтому очевидно то, что для подключения таких ПУ к общей СШ ПЭВМ нужны согласующие устройства, роль которых и выполняют АПУ. Для подсоединения нового ПУ достаточно разработать соответствующие адаптер, совместимый с той или иной СШ. Иногда в таких случаях можно обойтись даже без какой-либо модернизации системного ПО, если ПУ данного типа уже предусмотрено в системе.

Понятие "*адаптер периферийного устройства*" можно считать синонимом термина "*контроллер*" однако последний употребляется чаще для устройств, реализующих более сложные функции по управлению ПУ. Развитые АПУ включают в свой состав специализированные МП и память. Это же относится и к ПУ со сложными алгоритмами работы, требующими наличия совершенных блоков управления.

ПВВ обеспечивает непосредственное подключение АПУ к СШ (т.е. является по сути "точкой" такого подключения). Каждый ПВВ имеет свой адрес, аналогичный адресу в ОП, но содержащийся в другом адресном пространстве. Одному ПУ может быть приписано несколько ПВВ. Упрощено ПВВ можно считать регистром, в который записывается информация для передачи в ПУ или с которого считывается полученная из ПУ информация. Каждое стандартное ПУ для унификации ПО закреплено за ПВВ с определенными адресами. Можно провести аналогию ПВВ с морским портом, объясняющую происхождение этого термина.

Достаточно считать, что грузы, привозимые в морской порт и отправляемые на судах в различные концы света, а также грузы, разгружаемые с пришедших судов и доставляемые получателям. – это информация, передаваемая через ПВВ. Однако последние, в основном, являются однонаправленными. Совокупность ПВВ образует систему портов ввода-вывода.

Контроллер прямого доступа к памяти освобождает МП от прямого управления накопителями на магнитных дисках, что существенно повышает эффективное быстродействие ПК. Без этого контроллера обмен данными между внешним ЗУ и ОЗУ осуществля ется через регистр МП, а при его наличии данные непосредственно передаются между внешним ЗУ и ОЗУ, минуя МП.

Сопроцессор ввода-вывода за счет параллельной работы с МП значительно ускоряет выполнение процедур ввода-вывода при обслуживании нескольких внешних устройств (дисплей, принтер, НЖМД, НГМД и др.); освобождает МП от обработки процедур ввода-вывода, в том числе реализует и режим прямого доступа к памяти.

МП должен оперативно реагировать на различные события, происходящие в ПЭВМ в результате действий пользователя или без его ведома. В качестве примеров таких событий можно привести нажатие клавиши на клавиатуре (и другие события в ПУ), попытка деления на нуль, переполнение разрядной сетки, сбой питания (а также иные нарушения в работе оборудования), запланированные в программе обращения к ядру ОС и т.п. Необходимую реакцию на события обеспечивает система прерываний.

Прерыванием называется ситуация, требующая каких-либо действий (реакции) МП при возникновении определенного события. Под системой прерыванийпонимают комплекс аппаратных и программных средств, обеспечивающих выявление и обработку прерываний.

Обработка прерываний сводится к приостановке исполнения текущей последовательности команд (программы), вместо которой начинает

интерпретироваться другая последовательность инструкции, соответствующая данному типу прерывания и называемая обработчиком прерываний. После ее реализации исполнение прерванной программы может быть продолжено, если это возможно и/или целесообразно, что зависит от типа прерывания. Реакция на прерывание может состоять, например, в обработке введенного с клавиатуры символа.

Контроллер прерываний обслуживает процедуры прерывания, принимает запрос на прерывание от внешних устройств, определяет уровень приоритета этого запроса и выдает сигнал прерывания в МП. МП, получив этот сигнал, приостанавливает выполнение текущей программы и переходит к выполнению специальной программы обслуживания того прерывания, которое запросило внешнее устройство. После завершения программы обслуживания восстанавливается выполнение прерванной программы. Контроллер прерываний является программируемым.

Вся компьютерная система подразделяет виды архитектуры ЭВМ на три группы, обусловленные числом потоков команд и данных, рассмотрим их: Основоположником классической архитектуры ЭВМ 1-го и 2-го поколения был Джон фон Нейман, который и сформулировал основные принципы последовательности. К такой группе относятся однопроцессорные системы, в одном случае имеющие одиночный поток данных (SISD), а во втором - множественный поток данных (SIMD). Эти виды архитектуры обусловлены одним векторным потоком команд, при том что самих потоков данных множество. Следующая группа, включающая в себя виды архитектуры — MIMD. Представляет собой многопроцессорную систему, имеющую множественный поток команд и такой же поток данных. Данная архитектурная система в основном используется в современных супер-ЭВМ. И последние, третьи виды архитектуры - MISD, представляющие одну программу со множеством данных. К сожалению, MISD не имеет практической значимости. Данный вид причисляют не к компьютерной архитектуре, а к форме распараллеливания программ. Он обозначает одновременное исполнение двух и более копий одной программы в различных процессорных модулях с разными данными. Стоит рассмотреть такое немаловажное направление развития компьютерной архитектуры, как машины потоков данных. В 80-х годах предполагалось, что перспектива высокой производительности ЭВМ напрямую связана с управляемым потоком данных компьютера, в котором эти потоки способны исполнять несколько команд, притом, что рассматриваемые выше виды архитектуры ЭВМ имеют вычислительные системы, управляющиеся поками команд. В современном производстве прижились лишь немногие элементы этого подхода, применяемых в микропроцессорах, содержащих

множество синхронно действующих функциональных устройств, ожидающих готовности операндов.

В основу архитектуры современных персональных компьютеров положен магистрально-модульный принцип. Модульный принцип позволяет потребителю самому комплектовать нужную ему конфигурацию компьютера и производить при необходимости ее модернизацию. Модульная организация компьютера опирается на магистральный (шинный) принцип обмена информацией между устройствами.

Магистраль. Магистраль (системная шина) включает в себя три многоразрядные шины: шину данных, шину адреса и шину управления, которые представляют собой многопроводные линии (рис. 4.1). К магистрали подключаются процессор и оперативная память, а также периферийные устройства ввода, вывода и хранения информации, которые обмениваются информацией на машинном языке (последовательностями нулей и единиц в форме электрических импульсов).

Шина данных. По этой шине данные передаются между различными устройствами. Например, считанные из оперативной памяти данные могут быть переданы процессору для обработки, а затем полученные данные могут быть отправлены обратно в оперативную память для хранения. Таким образом, данные по шине данных могут передаваться от устройства к устройству в любом направлении.

Разрядность шины данных определяется разрядностью процессора, то есть количеством двоичных разрядов, которые могут обрабатываться или передаваться процессором одновременно. Разрядность процессоров постоянно увеличивается по мере развития компьютерной техники.

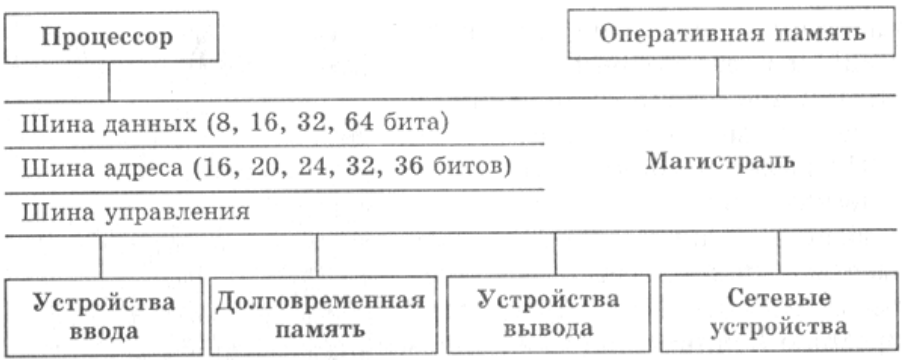


Рисунок 1. Магистрально-модульное устройство компьютера

Шина адреса. Выбор устройства или ячейки памяти, куда пересылаются или откуда считываются данные по шине данных, производит процессор. Каждое устройство или ячейка оперативной памяти имеет свой адрес. Адрес передается по адресной шине, причем сигналы по ней передаются в одном направлении - от процессора к оперативной памяти и устройствам (однонаправленная шина). Разрядность шины адреса определяет объем адресуемой памяти (адресное пространство), то есть количество однобайтовых ячеек оперативной памяти, которые могут иметь уникальные адреса. Количество адресуемых ячеек памяти

можно рассчитать по формуле: $N = 2^I$, где I - разрядность шины адреса. Разрядность шины адреса постоянно увеличивалась и в современных персональных компьютерах составляет 36 бит. Таким образом, максимально возможное количество адресуемых ячеек памяти равно: N = 236 = 68 719 476 736.

Шина управления. По шине управления передаются сигналы, определяющие характер обмена информацией по магистрали. Сигналы управления показывают, какую операцию - считывание или запись информации из памяти - нужно производить, синхронизируют обмен информацией между устройствами и так далее.

Порядок выполнения работы.

1. Изучите теоретический материал.
2. Рассчитайте необходимую минимальную конфигурацию ПЭВМ для рабочего места сотрудника регистрационного отделения поликлиники, если известно, что:

- в поликлинике обслуживается К*500 человек (где К – остаток от деления даты Вашего рождения на 6 плюс единица);
- в поликлинике находится К*2 кабинетов, в каждом из которых работает одна медсестра и один врач);
- для хранения медицинской карты требуется не более 20 листов (один лист 1500 символов);
- дубль карты может находится в любом из кабинетов;
- для связи с управляющими органами используется сеть интернет;
- в поликлинике имеется локальная сеть;
- документация хранится на дискетах и в бумажном виде глубиной 10 лет;
- в среднем за день обслуживается 60 новых пациентов;
- не менее чем раз в квартал готовится отчет в графической форме;
- доступ к информации ограничен;
- финансовые средства лимитированы годовым бюджетом;
- в регистратуре работает 3 человека.

3. Оформите отчет, включающий выполнение п.2 (с объяснениями), ответы на контрольные вопросы (не менее 5).

Примечание. По результатам п.2 в ходе практического занятия рекомендуется организовать дискуссионное обсуждение обоснования выбранного решения.

Контрольные вопросы

1. Как применяется вычислительная техника в хирургии?
2. Как применяется вычислительная техника в кардиологии?
3. Как применяется вычислительная техника в диагностике заболеваний?
4. Как применяется вычислительная техника для контроля состояния здоровья?

5. Основные принципы телемедицины.
6. Как применяется вычислительная техника в палатах интенсивной терапии и реанимации?
7. Как применяется вычислительная техника в здравоохранении?
8. Что входит в состав автоматизированного рабочего места врача?
9. На каких принципах основывается работа персонального компьютера?
10. Что входи в минимальный состав персонального компьютера?
11. Что такое архитектура ПЭВМ?
12. Перечислите особенности поколений ЭВМ (в том числе, применительно к медицине, биологии, экологии, здравоохранения).
13. В чем заключается принцип модульности построения архитектуры?
14. Поясните отличий одно, двух и трех шинной организации ПЭВМ.
15. Как программируются параллельный компьютеры?
16. Что такое нейрокомпьютеры и как они используются в медицине?
17. Что такое квантовые компьютеры?

2. ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ

Цель работы: овладение навыками имитационного моделирования, включая анимацию, для решения медико-биологических задач.

Краткие теоретические сведения.

Модель — это создаваемое человеком подобие изучаемого объекта (макет, изображение, схема, карта, словесное описание, математическое представление и т.п.). Метод моделирования состоит в исследовании объекта, явления или процесса путем построения моделей и их изучения. Модель всегда проще реального объекта, но она позволяет выделить главное, не отвлекаясь на детали. Необходимость моделирования объясняется принципиальной невозможностью исследования многих объектов или большой ресурсоемкостью их изучения.

Различают биофизические, физические, электрические, ситуационные, информационные, математические и другие модели.

Информационная модель — модель объекта, процесса или явления, в которой представлены информационные аспекты моделируемого объекта, процесса или явления. Среди информационных моделей особое место занимают модели представления знаний. Математическая модель — приближенное описание объекта, явления или процесса с помощью математической символики. Эта модель представляет собой систему математических соотношений: формул, функций, уравнений, систем уравнений, описывающих те или иные стороны изучаемого объекта, явления или процесса. Математическое моделирование — мощное средство познания, прогнозирования и управления. Анализ математической модели помогает проникнуть в суть изучаемого объекта или явления.

Математические модели строятся на основе данных эксперимента или умозрительно, описывают гипотезу, теорию или закономерность того или иного феномена и требуют дальнейшей проверки на практике. Различные варианты проводимых экспериментов выявляют границы применения математической модели и создают условия для ее дальнейшей коррекции. Математическое моделирование часто позволяет предвидеть характер изменения исследуемого процесса в условиях, трудно воспроизводимых в эксперименте, а в отдельных случаях позволяет предсказать ранее неизвестные явления и процессы.

Процесс математического моделирования принято делить на несколько этапов.

1. *Постановка задачи.* Необходимо отметить, что построение модели подразумевает наличие у специалиста хорошего уровня знаний предметной области, в рамках которой осуществляется моделирование. В постановку задачи входят определение цели исследования, выделение объекта исследования, определение параметров исследуемого объекта, выявление взаимосвязей между параметрами. Этап завершается записью модели в математическом виде.

2. Проведение модельных экспериментов. На этом этапе осуществляется решение прямой задачи, для которой предназначена математическая модель, т. е. получение выходных данных для дальнейшего сопоставления с результатами

наблюдений изучаемых явлений. Исследователь сознательно изменяет условия функционирования модели, регистрирует ее «поведение» в разных условиях. Важная роль при проведении модельных экспериментов принадлежит вычислительной технике. Именно она обеспечивает возможность обсчета многочисленных модельных экспериментов. Итогом второго этапа моделирования является множество результатов модельных экспериментов. При математическом моделировании разных процессов и явлений может использоваться один и тот же математический аппарат. Это упрощает задачу моделирования, дает возможность выбора из полученных вариантов.

3. *Оценка реализованной модели.* Выясняют, удовлетворяет ли созданная математическая модель критерию практики, т.е. согласуются ли результаты наблюдений с теоретическими (гипотетическими, модельными) данными в пределах заданной точности. Достижение такого результата означает, что положения, лежащие в основе модели, правильны и модель пригодна для исследования выбранного объекта или явления.

4. *Анализ модели на основе накопленных данных об изучаемом объекте, модернизация первоначально построенной модели.* С получением новых научных данных знания об исследуемом объекте уточняются, и наступает момент, когда результаты, получаемые на основании существующей модели, перестают им соответствовать. Возникает необходимость уточнения данной модели или построения новой. Между моментами построения исходной и последующей моделей проходят разные промежутки времени в зависимости от сути изучаемого явления, уровня и скорости исследования данной предметной области, характера полученных новых знаний и данных.

В медицине модели применяются для исследования структур, функций и процессов на разных уровнях организации живого организма: атомарно-молекулярном, субклеточном, клеточно-тканевом, органно-системном, организменном, биоценотическом.

В медицине, как и в биологи, используются в большинстве случаев биологические, физико-химические, математические модели. Исторически сложилось, что в медицине до сих пор широко распространены словесные описания объектов и процессов (например, заболеваний), а в последние десятилетия все чаще применяются информационные модели.

Биологические модели в медицине применяются для воспроизводства на лабораторных животных заболеваний или состояний, встречающихся у человека. Таким образом, в эксперименте исследуются механизмы возникновения заболевания, его этиология, патогенез, течение, изучаются варианты воздействия на протекание болезни, сравнивается эффективность применения различных лечебных пособий. В эксперименте, например, моделируются ишемические нарушения и гипертоническая болезнь, злокачественные новообразования и генетические заболевания, инфекционные процессы и др.

Для реализации биологических моделей экспериментальным животным вводят токсины, заражают их микробами, перевязывают сосуды, исключают из пищи определенные вещества, помещают в искусственно создаваемую среду обитания и др. Подобные экспериментальные модели применяются в нормальной и патологической физиологии, генетике, фармакологии, хирургии, реаниматологии. Физико-химические модели имитируют сложные акты поведения, например формирование условного рефлекса.

Удачным следует признать опыт построения электронных схем, моделирующих биоэлектрические потенциалы в нервной клетке и синапсе на основе данных электрофизиологических исследований.

В настоящее время в медицине самое широкое распространение получили математические модели. Они используются практически во всех ее областях. Математические модели применяются для изучения сложных физиологических процессов, диагностики патологических состояний, исследования взаимодействия систем организма в норме и патологии, при изучении эпидемических процессов, в клинической иммунологии, фармакокинетике.

Из математических моделей, известных в физиологии, следует упомянуть модель возбуждения нервного волокна, предложенную А.Ходжкином и А.Хаксли.

Модель сердечной деятельности Ван дер Пола и Ван дер Марка, основанная на теории релаксационных колебаний, позволила предсказать возможность особого нарушения сердечного ритма, впоследствии обнаруженного у человека.

Ярким примером использования математической модели для обобщения накопленных экспериментальных знаний является модель кровообращения Ф. Гродинза. Построением и исследованием моделей кровообращения, применяющихся в практике российской сердечно-сосудистой хирургии, занимается В.А.Лищук.

В медицинской информатике широко используется моделирование, особенно часто математическое и информационное. Математические модели используются для расчета клинически значимых показателей при обработке сигналов и изображений, для описания заболеваний и состояний при вычислительной диагностике и прогнозировании. Информационное моделирование все чаще применяется при описании деятельности ЛПУ и их подразделений.

Когда осуществляют процессный анализм в ЛПУ с последующим выходом на планирование эксперимента и принятие управленческих решений, то всегда наталкиваемся на одну и ту же проблему разработки и апробации модели управления процессов обеспечения КМП в условиях реального функционирования ЛПУ. Одна из важных особенностей управления качеством производства медицинских услуг — принципиальная невозможность проведения реальных работ по управлению КМП до завершения эксперимента. Возможным выходом является использование имитационных моделей. Сущность метода имитационного моделирования состоит в построении так

называемой имитационной модели исследуемого ЛПУ или его подразделения и целенаправленном экспериментировании с разрабатываемой моделью управления для получения ответов на те или иные вопросы. Говоря о методе имитационного моделирования, как правило, имеют в виду метод, ориентированный на применение вычислительной техники, хотя в принципе могут использоваться любые средства, включая лист бумаги и карандаш.

Другой важный аспект - использование имитационных моделей в процессе эксплуатации информационных технологий управления для принятия управленческих решений по качеству. Такие модели создаются в процессе проектирования, чтобы их можно было непрерывно модернизировать и корректировать в соответствии с изменяющимися условиями работы пользователей. Эти же модели могут быть использованы для обучения персонала ЛПУ перед вводом в действие разработанных технологий в эксплуатацию или для проведения деловых игр.

Принципиальные возможности метода имитационного моделирования весьма велики, он позволяет при необходимости исследовать системы любой сложности и назначения с любой степенью детализации. Ограничениями являются лишь мощность используемой ПЭВМ и трудоемкость подготовки сложного комплекса программ. Методы имитационного моделирования развиваются в основном в направлении исследования степени подобия имитационных моделей реальным системам и разработки типовых методов и приемов создания имитационных моделей.

Имитационное моделирование в медицине используется в основном по следующим направлениям:

1. при исследовании сложных внутренних и внешних взаимодействий ЛПУ с целью оптимизации их функционирования. Для этого на модели ЛПУ изучают закономерности взаимосвязи переменных, вносят в модель ЛПУ изменения и наблюдают их влияние на поведение системы производства медицинских услуг;
2. для прогнозирования поведения ЛПУ в будущем на основе моделирования развития самого ЛПУ и его внешней среды;
3. в целях обучения персонала ЛПУ, которое может быть двух типов:

- первый тип - индивидуальное обучение оператора, управляющего неким технологическим процессом или медицинской аппаратурой;
- второй тип - обучение группы персонала, осуществляющей коллективное управление сложным объектом по производству медицинских услуг.

Рассмотрим простейшие примеры математического имитационного моделирования.

1. Вычисление площади неизвестной фигуры на примере определения числа Пи.

Исходя из предположений полагаем, что площадь круга известного радиуса равна квадрату радиуса умноженного на некоторую величину. Эта величина экспериментально определяется следующим образом. На листе бумаги чертится круг радиусом R (чем больше, тем лучше), а около него описывается квадрат со

стороной 2R . Далее на рисунок нангосится случайным образом несколько сот точек в случайных местах. На следующем этапе подсчитывается количество течек: общее – N и попавших в круг – N0. Тогда отношение N0/N – позволяет определить площаджь круга в зависимости от площади квадрата, которая известна.

Заметим, что таким образом можно определить площадь любой фигуры, находящейся в квадрате с известными сторонами –например, площадь клетки, листа, капли крови, отпечатка пальцев, пятна лакмусовой бумажки на определенную концентрации химического вещества. Такой способ называется методом Монте-Карло.

Имитационная модель описанного процесса легко формализуется и следовательно может быть реализована с помощью ПЭВМ, имеющей фон Нейманскую архитектуру и последовательную организации выполнения программного кода. Алгоритм в этом случае выглядит следующим образом:

1. Задаемся определенным радиусом R.
2. Задаемся количеством «точек» N. Количество точек, попавших в круг считаем равным 0.
3. С помощью датчика случайных чисел генерируем координаты точки x и y [0,2R].
4. Проверяем: если координаты точки попали в круг с координатами центра (R,R) и радиусом R, то увеличиваем количество точек, попавших в круг на 1.
5. Процедуры 3,4 повторяем, пока общее количество случайным образом сгерерированных точек не превысит N.
6. Находим число Пи.

2. Игра «Жизнь».

Многие процессы, интересующих медиков (например, динамика возрастного состава в регионе) может быть с определенной точностью описано математической моделью. Самая простая модель известна под назнванием «Игра Жизнь».

Предполагается наличие прямоугольного клетчатого поля, в каждой клетке которого может «жить» существо. Если клетка пустая – то в ней никто не живет. Модель задается двумя параметрами: начальной конфигурацией (размером поля и расположением живых существ) и определенными «биологическими законами», регулирующими жизнь популяции существ. На каждой итерации осуществляется последовательный просмотр всех клеток с некоторой начальной (координаты клетки выбираются исследователем или случайным обрзом) с применением к ним биологических законов.

В качестве типовых законов, предлагаются, например, следующие:

1. Если выбранная клетка пуста, а в соседней с ней клетках находится более двух существ, то внутри клетки появляется существо («размножение»).

2. Если выбранная клетка непуста, а в соседних с ней клетках живет меньше трех или больше четырех существ, то клетка очищается (существо в ней погибает от одиночества или перенаселения).
3. Если правила 1 и 2 не выполняются, то ничего с клеткой не происходит.
(Под соседними полразумеваются восемь окружающих клеток, за грницей ореала – прямоугольного поля – существ нет).

3. Модель «хищник» - «жертва».
В системе хищник-жертва ситуация моделируется следующим образом. В случае конкурирующих популяций исчезновение одной означает выигрыш для другой в борьбе за дополнительные ресурсы. Обозначим через С численность популяции хищника, N – популяцию жертвы. Наиболее популярная модель, отражающая колебания численности имеет вид:

$$N_{i+1} = N_i + (r \cdot N_i - a \cdot N_i \cdot C_i) \cdot \Delta t,$$
$$C_{i+1} = C_i + (-q \cdot C_i - a \cdot f \cdot N_i \cdot C_i) \cdot \Delta t.$$

Согласно первому уравнению при С=0 численность жертв быстро растет со скоростью r , поскольку модель не учитывает внутривидовой конкуренции. Скорость роста числа жертв ($\frac{\Delta N}{\Delta t}$) уменьшается тем больше, чем чаще происходят встречи особей видов (тогда а - коэффициент эффективности поиска).

Второе уравнение показывает, что в отсутствии жертв численность хищников быстро убывает со скоростью q: положительное слагаемое в правой части уравнения компенсирует эту убыль, f – коэффициент эффективности перехода пищи к потомству хищников.

4. Внутривидовая конкуренция в популяции с дискретным размножением.
Для популяций с дискретным размножением (некоторые виды растений, насекомые) поколения дифференцирпованно разнесены во времени и особи разныз покалений вместе не сосуществуют. Численность подобной популяции характеризуется числом N_t , а время t- дискретная величина – можно, в первом приближении, считать номером популяции. Тогда одна из моделей межвидовой конкуренции может быть описана уравнением:

$$N_{t+1} = \frac{N_t \cdot R}{1 + (a * N_t)^b}$$

где R – скорость воспроизводства попляции в отсутствии внутривидовой конкуренции (математически это соответствует a=0); a – параметр, характеризующий интенсивностьвнутривидовой конкуренции, при b=1 осуществляется выход численности популяции на стационарное значение при любых значениях других параметров модели.

Знаменатель в уравнении отражает наличие конкуренции, делающей скорость роста тем меньше, чем больше численность популяции. Ланная модель описывает четыре вида эволюции:
1. мнотонное установление стационарной численности популяции;
2. колебательное установление стационарной численности популяции;
3. устойчивые предельные циклы изменения численности популяций;
4. случайные изменения численности популяции без наличия явных закономерностей.

5. Внутривидовая конкуренция в популяции с непрерывным размножением.
В данном случае численность популяции N(t) является непрерывной функцией во времени. В начале эволюционного процесса численность популяции невилика, а ее удельная скорость не зависит от численности: $\frac{1}{N} \cdot \frac{\Delta N}{\Delta t} = r$ - скорость роста численности популяции в отсутствии конкуренции. Далее, по мере роста численности, скорость роста начинает уменьшаться и при достижении определенного критического значения К обращаетя в ноль. Таким образом, в первом приближении, математическая модель имеет вид:
$N_{i+1} = N_i + r \cdot N_i \cdot (\frac{K-N_i}{K}) \cdot \Delta t$.

Порядок выполнения работы.
1. Изучите теоретические сведения.
2. Составьте блок-схему алгоритма и отладьте программу, определяющую число Пи. Постройте график зависимости числа Пи от количества точек N в двух вариантах: учитывающеьм и неучитывающем попадание координат случайной точки на окружность. Сделайте выводы.
3. Разработайте последовательность действий (и апробируйте ее) для экспериментального определения числа Пи в электронной таблице Excel.
4. Составьте и отладьте программу «игра Жизнь» на языке высокого уровня и предложите ее реализацию в Excel. Сравните полученные Вами результаты с программным продуктом, предложенном на сайте http://www.nature.air.ru/models/models.htm.
5а. Проведите с помощью MatCad и Exel моделирование динамики в системе хищник-жертва при значениях параметров модели: r=8, a=0.15, q=2, f=0.5, N0=175, C0=45. Сделайте выводы.
6а. Изучите функционирование аналогичной программы, представленной на сайте http://www.nature.air.ru/models/models.htm.
7а. Исследуйте зависимость амплитуд колебаний численности хищников от амплитуд колебаний численности жертв в зависимости от параметров a и f. Значения остальных параметров фиксируйте по своему усмотрению.
5б. Составьте и отладьте программу моделирующую внутривидовую конкуренция в популяции с дискретным размножением на языке высокого уровня и предложите ее реализацию в Excel.

6б. Проведите с помощью MatCad и Exel моделирование в четырех видах эволюции с R=2 (1 и 2 режимы) и R=2 (3 и 4 режимы). На фазовой плоскости (b, R) найдите границы зон, разделяющих разные режимы эволюции изучаемой системы. Сделайте выводы.

7в. Составьте и отладьте программу моделирующую внутривидовую конкуренцию в популяции с непрерывным размножением при нескольких значениях входящих в модель параметров на языке высокого уровня и предложите ее реализацию в Excel. Сделайте выводы.

7в. Постройте (или найдите в информационных источниках) альтернативную модель внутривидовой конкуренции и исследуйте предсказываемые режимы.

Примечание: обучающийся самостоятельно выбирает подпункты «а», «б» и «в», исходя из номера зачетной книжки (определяется отстаток от деления последние цифры на 3 и прибавляется 1 – полученное число пределяет соответствующую букву).

8. Изучите интерфейс и функциональные возможности программ, представленных на сайте http://www.nature.air.ru/models/models.htm. Результаты изучения представьте в табличном виде.

9. Оформите отчет, включающий результаты выполненных действий и презентации к минидокладу на тему «Роль имитационного моделирования в медико-биологических исследованиях».

10. Придумайте контрольные вопросы (не менее 5) в виде мини тестов по указанной в п.9 тематике. (Конкретное гносеологическое насыщение вопросов – на Ваше усмотрение).

Литература

1. Имитационное моделирование в биологии. Презентация. http://900igr.net/prezentatsii/biologija/Modelirovanie-v-biologii/Modelirovanie-v-biologii.html

3. ОСНОВЫ РАБОТЫ В ЭЛЕКТРОННЫХ ТАБЛИЦАХ (НА ПРИМЕРЕ EXCEL)

Цель работы: Освоение базовыми навыками работы с электронными таблицами на примере Excel.

Краткие теоретические сведения.

Электронные таблицы позволяют обрабатывать большие массивы числовых данных. В отличие от таблиц на бумаге электронные таблицы обеспечивают проведение динамических вычислений, т. е. пересчет по формулам при введении новых чисел. С помощью электронных таблиц можно представить функцию в числовой форме и построить ее график, в физике - обработать результаты лабораторной работы, в географии или истории - представить статистические данные в форме диаграммы.

Электронная таблица состоит из столбцов и строк. Заголовки **столбцов** обозначаются буквами или сочетаниями букв (A, C, AB и т. п.), заголовки **строк** - числами (1, 2, 3 и далее).

К обработке данных в электронных таблицах относятся:

- проведение различных вычислений с помощью формул и функций, встроенных в редактор;
- построение диаграмм;
- обработка данных в списках (Сортировка, Автофильтр, Расширенный фильтр, Форма, Итоги, Сводная таблица);
- решение задач оптимизации (Подбор параметра, Поиск решения, Сценарии "что - если" и другие задачи);
- статистическая обработка данных, анализ и прогнозирование (инструменты анализа из надстройки "Пакет анализа").

Таким образом, Excel являются не только средством автоматизации расчетов, но и средством моделирования различных ситуаций.

Область применения Excel: статистические расчеты, математическое моделирование, обработка табличных данных, визуализация результатов исследований в форме графиков и диаграмм, автоматизация расчетной деятельности, подготовка различных документов, простейшие логические вычисления.

При запуске Excel открывается окно приложения, в котором отображается новая рабочая книга – Книга 1. Окно приложения Excel имеет пять основных областей: строка меню; панели инструментов; строка состояния; строка ввода; область окна рабочей книги (см. рисунок 1).

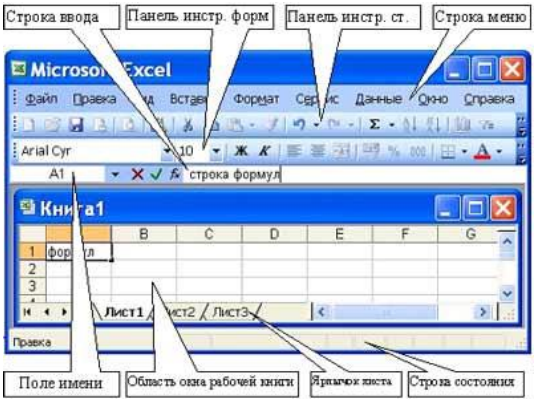


Рис. 1.

Основная обработка данных в Excel осуществляется при помощи команд из строки меню. Панели инструментов *Стандартная* и *Форматирование* являются встроенными панелями MS Excel, которые располагаются под строкой меню и содержат определенные наборы пиктограмм (кнопок). Основная часть пиктограмм предназначена для выполнения наиболее часто используемых команд из строки меню. *Строка формул* в Excel используется для ввода и редактирования значений, формул в ячейках или диаграммах. *Поле имени* – это окно слева от строки формул, в котором выводится имя активной ячейки. *Пиктограммы:* **X, V, fx**, расположенные слева от строки формул - это кнопки отмены, ввода и вставка функции соответственно.

Строка состояния окна приложения Excel расположена в нижней части экрана. Левая часть строки состояния указывает сведения о состоянии рабочей области электронной таблицы (Готово, Ввод, Правка, Укажите). Кроме того, в левой части строки состояния кратко описываются результаты выполненной команды. В правой части строки состояния выводятся результаты вычислений (при выполнении автоматических вычислений с помощью контекстного меню строки состояния) и отображаются нажатые клавиш Ins, Caps Lock, Num Lock, Scroll Lock.

Рабочая книга (документ Excel) состоит из рабочих листов, каждый из которых является электронной таблицей. По умолчанию открывается три рабочих листа или три электронных таблицы, переход к которым можно осуществить, щелкая на ярлычках, расположенных внизу книги. При необходимости в книгу можно добавить рабочие листы (электронные таблицы) или удалить их из книги. *Кнопки прокрутки* ярлычков осуществляют прокрутку ярлычков рабочей книги. Крайние кнопки осуществляют прокрутку к первому и последнему ярлычку рабочей книги. Внутренние кнопки осуществляют прокрутку к предыдущему и следующему ярлычку рабочей книги.

Основные понятия электронной таблицы: заголовок столбца, заголовок строки, ячейка, имя ячейки, маркер выделения, маркер заполнения, активная ячейка, строка формул, поле имени, активная область листа.

Рабочая область электронной таблицы состоит из строк и столбцов, имеющих свои имена. Имена строк – это их номера. Нумерация строк начинается с 1 и

заканчивается максимальным числом, установленным для данной программы. Имена столбцов – это буквы латинского алфавита сначала от A до Z , затем от AA до AZ, BA до BZ и т.д. Максимальное количество строк и столбцов электронной таблицы определяется особенностями используемой программы и объемом памяти компьютера, например, в табличном процессоре Excel 256 столбцов и более 16 тысяч строк. Пересечение строки и столбца образует ячейку электронной таблицы, имеющую свой уникальный адрес. Для указания адресов ячеек в формулах используются ссылки (например, A6 или D8).

Ячейка – область, определяемая пересечением столбца и строки электронной таблицы, имеющая свой уникальный адрес.

Адрес ячейки определяется именем (номером) столбца и именем (номером) строки, на пересечении которых находится ячейка, например A10. Ссылка – указание адреса ячейки. *Активной ячейка* - это выделенная ячейка, имя которой отображается в поле имени. Маркером выделения называется полужирная рамка вокруг выделенной ячейки. Маркер заполнения - это черный квадрат в правом нижнем углу выделенной ячейки.

Активная область листа - это область, которая содержит введенные данные. В электронных таблицах можно работать как с отдельными ячейками, так и с группами ячеек, которые образуют блок. Блок ячеек – группа смежных ячеек, определяемая с помощью адреса. Адрес блока ячеек задается указанием ссылок первой и последней его ячеек, между которыми ставится разделительный символ – двоеточие. Если блок имеет вид прямоугольника, то его адрес задается адресами левой верхней и правой нижней ячеек, входящих в блок. Блок используемых ячеек может быть указан двумя путями: либо заданием с клавиатуры начального и конечного адресов ячеек блока, либо выделением соответствующей части таблицы при помощи левой клавиши мыши. Пример задания адресов ячейки и блоков в электронной таблице:

- адрес ячейки, находящейся на пересечении столбца F и строки 9, выражается ссылкой F9;
- адрес блока, образованного в виде части строки 1 - B1:E1;
- адрес блока, образованного в виде столбца C - C1:C21;
- адрес блока, образованного в виде прямоугольника - A3:G10.

Работа с файлами в Excel

Сохранение и присвоение имени рабочей книге.

При сохранении рабочей книги в Excel открывается окно диалога "Сохранение документа". В этом окне необходимо указать: имя файла, тип файла, выбрать диск и папку, в которой будет храниться рабочая книга. Таким образом, книга с входящими в нее рабочими листами сохраняется в папке на диске в виде отдельного файла с уникальным именем. Файлы книг имеет расширение xls.

Открытие рабочей книги в Excel

Для открытия рабочей книги в Excel, надо выбрать команду Файл / Открыть или щелкнуть на кнопке Открыть на стандартной панели инструментов. Excel

выведет окно диалога "Открытие документа" в нем можно выделить требуемый файл и щелкнуть на кнопке Открыть.

Закрытие рабочей книги и выход из Excel

Для того чтобы закрыть рабочую книгу в Excel выберите команду Файл / Закрыть, в результате чего закроется рабочая книга. Для выхода из Excel необходимо выбрать команду Файл / Выход или щелкнуть на кнопку закрыть в правой части строки заголовка окна приложения.

Любая обработка информации начинается с ее ввода в компьютер. В электронные таблицы MS Excel можно вводить текст, числа, даты, время, последовательные ряды данных и формулы.

Ввод данных осуществляется в три этапа: выделение ячейки; ввод данных; подтверждение ввода (нажать клавишу Enter).

После того как данные введены, их нужно представить на экране в определенном формате. Для представления данных в MS Excel существуют различные категории форматных кодов.

Для редактирования данных в ячейке необходимо дважды щелкнуть на ячейке и произвести редактирование или исправление данных.

К операциям редактирования относятся:

- удаление и вставка строк, столбцов, ячеек и листов;
- копирование и перемещение ячеек и блоков ячеек;
- редактирование текста и чисел в ячейках.

К операциям форматирования относятся:

- изменение числовых форматов или формы представления чисел;
- изменение ширины столбцов;
- выравнивание текста и чисел в ячейках;
- изменение шрифта и цвета;
- Выбор типа и цвета границы;
- Заливка ячеек.

Ввод чисел и текста. Любую информацию, которая обрабатывается на компьютере, можно представить в виде чисел или текста. Числа и текст по умолчанию Excel вводит в формате Общий.

Ввод текста. Текст - это любая последовательность введенных в ячейку символов, которая не может быть интерпретирована Excel как число, формула, дата, время суток. Введенный текст выравнивается в ячейке по левому краю. Чтобы ввести текст, выделите ячейку и наберите текст с клавиатуры. Ячейка может вмещать до 255 символов. Если требуется ввести некоторые числа как текст, то для этого выделите ячейки, а затем выберите команду Формат / Ячейки. Далее выберите вкладку “Число” и в появившемся списке форматов выберите Текстовый. Еще один способ ввода числа как текста – это ввести перед числом символа апострофа. Если текст не помещается в ячейку, то необходимо увеличить ширину столбца или разрешить перенос по словам (Формат / Ячейки, вкладка Выравнивание).

Ввод чисел

Числовые данные – это числовые константы: 0 - 9, +, -, /,*, E, %, точка и запятая. При работе с числами необходимо уметь изменять вид вводимых чисел: число знаков после запятой, вид целой части, порядок и знак числа.

Excel самостоятельно определяет относится ли введенная информация к числу. Если введенные в ячейку символы относятся к тексту, то после подтверждения ввода в ячейку они выравниваются по левому краю ячейки, а если символы образуют число – то по правому краю ячейки.

Числа в Excel отображаются в категориях Числовой, Экспоненциальный, Финансовый, Денежный, Процентный, Дробный.

Ввод последовательных рядов данных

Под рядами данных подразумеваются данные, отличающиеся друг от друга на фиксированный шаг. При этом данные не обязательно должны быть числовыми. Для создания рядов данных необходимо выполнить следующее:

1. Ввести в ячейку первый член ряда.
2. Выделить область, где будет расположен ряд. Для этого нужно подвести указатель мыши к маркеру заполнения, и в этот момент, когда белый крестик переходит в черный, нажать левую кнопку мыши. Далее, удерживая нажатой кнопку мыши, надо выделить нужную часть строки или столбца. После того как вы отпустите кнопку мыши, выделенная область заполнится данными.

Формат данных

Данные в MS Excel выводятся на экран в определенном формате. По умолчанию информация выводится в формате Общий. Можно изменить формат представления информации в выделенных ячейках. Для этого выполните команду Формат / Ячейки.

Появится окно диалога “Формат ячеек”,в котором нужно выбрать вкладку “Число”. В левой части окна диалога “Формат ячеек” в списке “Числовые форматы” приведены названия всех используемых в Excel форматов.

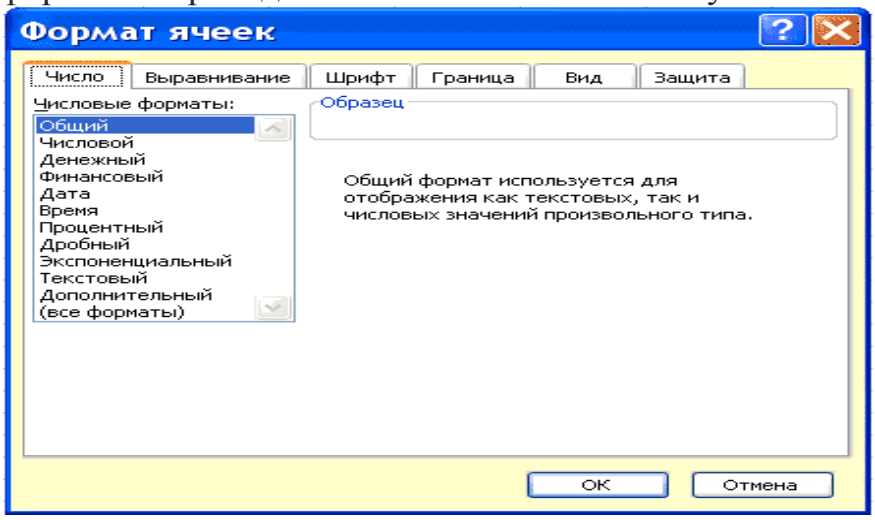


Рисунок 2

Для формата каждой категории приводится список его кодов. В правом окне “Тип” вы можете просмотреть все форматные коды, которые используются для

представления на экране информации. Для представления данных вы можете использовать встроенные форматные коды MS Excel или ввести свой (пользовательский) код формата. Для ввода форматного кода выберите строку (все форматы) и введите символы форматного кода в поле ввода “Тип”.

Стиль представления данных

Одним из способов упорядочения данных в Excel является введение стиля. Для создания стиля используется команда Формат / Стиль. Выполнение этой команды открывает окно диалога “Стиль”.

Рассмотрим технологию создания электронной таблицы на примере проектирования таблицы Учет товаров на складе:

1. Для создания таблицы надо выполнить команду Файл / Создать и щелкнуть в области задач на пиктограмме Чистая книга.
2. Сначала необходимо осуществить разметку таблицы (Рис. 1). Например, таблица Учет товаров имеет семь колонок, которые закрепим за столбцами от A до G. Далее надо сформировать заголовки таблицы. Затем нужно ввести общий заголовок таблицы, а потом названия полей. Они должны находиться в одной строке и следовать друг за другом. Заголовок можно расположить в одну или две строки, выровнять по центру, правому, левому, нижнему или верхнему краю ячейки.
3. Для ввода заголовка таблицы необходимо установить курсор в ячейку A2 и ввести название таблицы «Остатки товаров на складе».
4. Выделить ячейки A2:G2 и выполнить команду Формат/Ячейки, на вкладке Выравнивание выбрать способ выравнивания по центру и установить флажок объединение ячеек. Нажать ОК (Рис. 3.).
5. Создание «шапки» таблицы. Ввести названия полей, например, № склада, Поставщик и т.д.
6. Для расположения текста в ячейках "шапки" в две строки необходимо выделить эту ячейку и выполнить команду Формат/Ячейки, на вкладке Выравнивание установить флажок переносить по словам.
7. Вставка различных шрифтов. Выделить текст и выбрать команду Формат/Ячейки, вкладка Шрифт. Установить гарнитуру шрифта, например, Times New Roman, его размер (кегль) и начертание.
8. Осуществить выравнивание текста в «шапке» таблицы (выделить текст и щелкнуть на кнопке По центру на панели инструментов форматирования).
9. При необходимости изменить ширину столбцов с помощью команды Формат / Столбец / Ширина.
10. Изменить высоты строки можно командой Формат / Строка / Высота.
11. Добавление рамки и заливки ячеек можно осуществить командой Формат / Ячейка на вкладках Граница и Вид соответственно. Выделите ячейку или ячейки и на вкладке Граница выберите тип линии и с помощью мыши укажите, к какой части выделенного диапазона он относится. На вкладке Вид выберите цвет заливки выделенных ячеек.

12. Перед вводом данных в таблицу можно осуществить форматирование ячеек столбцов под «шапкой» таблицы при помощи команды Формат/Ячейки, вкладка Число (Рис. 4.). Например, выделите вертикальный блок ячеек под ячейкой "№ склада" и выберите команду Формат/Ячейки на вкладке Число выделите Числовой и щелкните ОК.

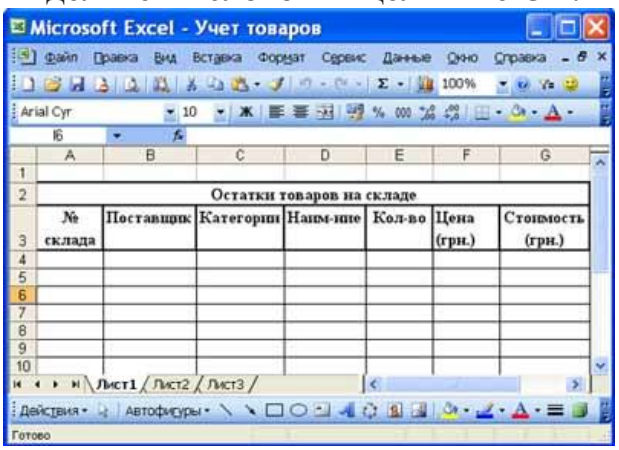


Рисунок 2.

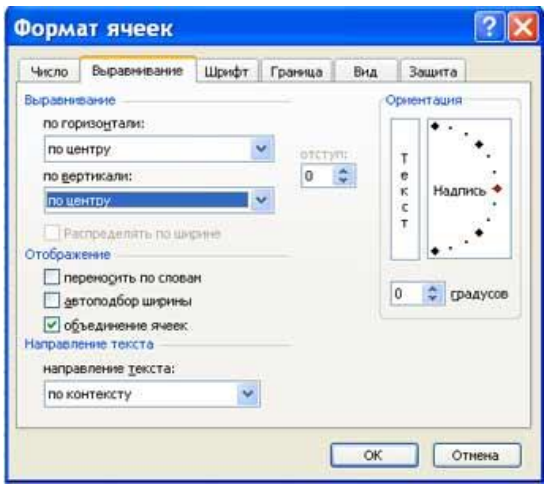


Рисунок 3.

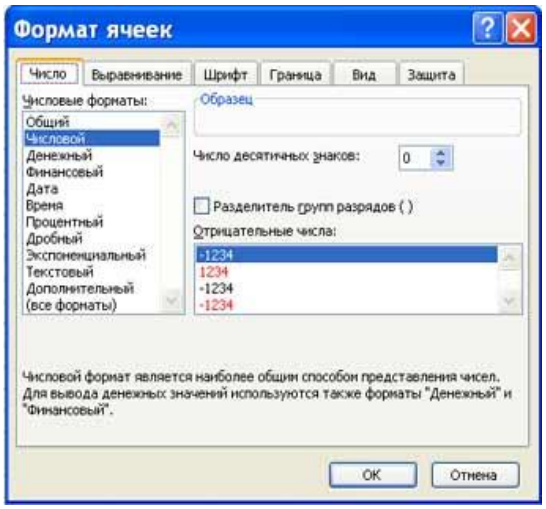


Рисунок 4.

Формулы

Формулы – это выражение, начинающееся со знака равенства и состоящее из числовых величин, адресов ячеек, функций, имен, которые соединены знаками арифметических операций. К знакам арифметических операций, которые используются в Excel относятся: сложение; вычитание; умножение; деление; возведение в степень.

Некоторые операции в формуле имеют более высокий приоритет и выполняются в такой последовательности:

- возведение в степень и выражения в скобках;
- умножение и деление;
- сложение и вычитание.

Результатом выполнения формулы является значение, которое выводится в ячейке, а сама формула отображается в строке формул. Если значения в ячейках, на которые есть ссылки в формулах, изменяются, то результат изменится автоматически.

Внесение изменений в формулу

Для внесения изменений в формулу щелкните мышью на строке формул или клавишу F2. Затем внесите изменения и нажмите кнопку Ввода в строке формул или клавишу Enter. Если вы хотите внести изменения в формулу непосредственно в ячейке, где она записана, то дважды щелкните мышью на ячейке с этой формулой. Для отмены изменений нажмите кнопку Отмена в строке формул или клавишу Esc.

Использование ссылок

Ссылка однозначно определяет ячейку или группу ячеек рабочего листа. С помощью ссылок можно использовать в формуле данные, находящиеся в различных местах рабочего листа, а также значение одной и той же ячейки в нескольких формулах. Можно также ссылаться на ячейки, находящиеся на других листах рабочей книги, в другой рабочей книге, или даже на данные другого приложения. Ссылки на ячейки других рабочих книг называются внешними. Ссылки на данные в других приложениях называются удаленными.

Перемещение и копирование формул

После того как формула введена в ячейку, вы можете ее перенести, скопировать или распространить на блок ячеек. При перемещении формулы в новое место таблицы ссылки в формуле не изменяются, а ячейка, где раньше была формула, становится свободной. При копировании формула перемещается в другое место таблицы, при этом абсолютные ссылки не изменяются, а относительные ссылки изменяются.

При копировании формул можно управлять изменением адресов ячеек или ссылок. Если перед всеми атрибутами адреса ячейки поставить символ “$” (например, $A$1), то это будет абсолютная ссылка, которая при копировании формулы не изменится. Изменятся только те атрибуты адреса ячейки, перед которыми не стоит символ “$”, т.е. относительные ссылки. Для быстрой

установки символов “$” в ссылке ее необходимо выделить в формуле и нажать клавишу F4.

Для перемещения формулы подведите указатель мыши к тому месту границы ячейки, где изображение указателя мыши изменяется с белого крестика на белую стрелку. Затем нажмите левую кнопку мыши и, удерживая ее, перемещайте ячейку в нужное место таблицы. Завершив перемещение, отпустите кнопку мыши. Если в записи формулы есть адреса ячеек, они при перемещении формулы не изменяются.

Для копирования формулы подведите указатель мыши к тому месту границы ячейки или блока, где изображение указателя изменяется с белого крестика на белую стрелку. Затем нажмите клавишу Ctrl и левую кнопку мыши и перемещайте ячейку в нужное место таблицы. Для завершения копирования отпустите кнопку мыши и клавишу Ctrl. Если в записи формулы есть относительные адреса ячеек, при копировании формулы они изменятся.

Распространение формул

Помимо копирования и перемещения формулу можно распространить на часть строки или столбца. При этом происходит изменение относительных ссылок. Для распространения формулы необходимо выполнить следующие действия:

1. Установите курсор в ячейку с формулой.
2. Подведите указатель мыши к маркеру заполнения. Изображение указателя изменяется на черный крестик.
3. Нажмите левую кнопку мыши и, удерживая ее нажатой, перемещайте курсор до нужного места. Для завершения распространения формулы отпустите кнопку.

Необходимо отметить, что Excel выводит в ячейку значение ошибки, когда формула для этой ячейки не может быть правильно вычислена. Если формула содержит ссылку на ячейку, которая содержит значение ошибки, то эта формула также будет выводить значение ошибки.

Функции Excel

Функции Excel — это специальные, заранее созданные формулы для сложных вычислений, в которые пользователь должен ввести только аргументы.

Функции состоят из двух частей: имени функции и одного или нескольких аргументов. Имя функции описывает операцию, которую эта функция выполняет, например, СУММ.

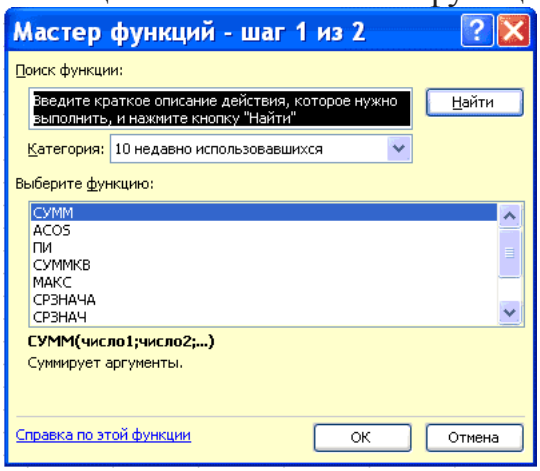
Аргументы функции Excel - задают значения или ячейки, используемые функцией, они всегда заключены в круглые скобки. Открывающая скобка ставится без пробела сразу после имени функции. Например, в формуле «=СУММ(A2;A9)», СУММ — это имя функции, а A2 и A9 — ее аргументы.

Эта формула суммирует числа в ячейках A2, и A9. Даже если функция не имеет аргументов, она все равно должна содержать круглые скобки, например функция ПИ(). При использовании в функции нескольких аргументов они отделяются один от другого точкой с запятой. В функции можно использовать до 30 аргументов.

Ввод функций в рабочем листе

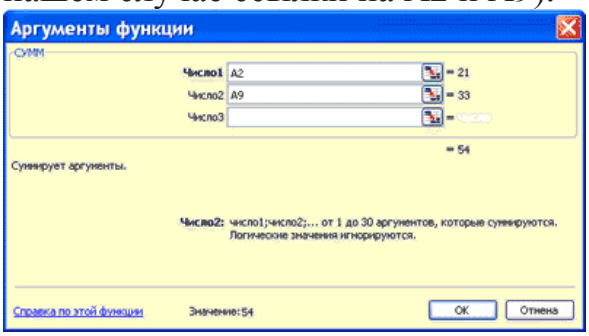
Вы можете вводить функции в рабочем листе прямо с клавиатуры или с помощью команды Функция меню Вставка.

Если вы выделите ячейку и выберете команду Вставка/Функция, Excel выведет окно диалога Мастер функций – шаг 1 из 2. Открыть это окно можно также с помощью кнопки Вставка функции на строке ввода формул.



В этом окне сначала выберите категорию в списке Категория и затем в алфавитном списке Функция укажите нужную функцию.

Excel введет знак равенства (если вы вставляете функцию в начале формулы), имя функции и круглые скобки. Затем Excel откроет второе окно диалога мастера функций, в котором необходимо установить аргументы функции (в нашем случае ссылки на A2 и A9).



Второе окно диалога Мастера функций содержит по одному полю для каждого аргумента выбранной функции. Справа от каждого поля аргумента отображается его текущее значение (21 и 33). Текущее значение функции отображается внизу окна диалога (54). Нажмите кнопку ОК или клавишу Enter, и созданная функция появится в строке формул.

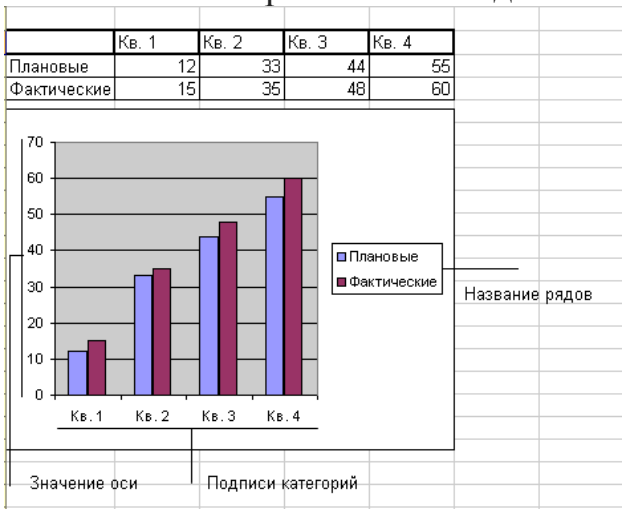
Диаграммы в Excel

С помощью Microsoft 2003 можно создавать сложные диаграммы для данных рабочего листа. Диаграмма – графическое изображение зависимости между величинами. Диаграммы являются наглядным средством представления данных

рабочего листа. Диаграмму можно создать на отдельном листе или поместить в качестве внедренного объекта на лист с данными.

Представление данных на диаграмме

Диаграмма связана с данными, на основе которых она создана, и обновляется автоматически при изменении данных.



Ось значений. Excel создает ось значений на основе указанных данных. В данном случае значения оси изменяются от 0 до 70, что соответствует значениям ячеек диапазона на листе.

Ось категорий (имена категорий). В качестве имен оси категорий Excel использует заголовки столбцов или строк данных. В приведенном примере в качестве имен оси категорий отображаются заголовки столбцов, соответствующие первому кварталу, второму кварталу и т.д.

Имена рядов данных диаграммы. Excel также использует заголовки столбцов или строк данных в качестве имен рядов данных. Имена рядов отображаются в легенде диаграммы. В приведенном примере в качестве имен рядов выступают заголовки рядов планируемых и фактических значений.

Маркеры данных. Маркеры данных одного цвета представляют один ряд данных. Каждый маркер соответствует одному значению данных листа. В приведенном примере самый правый маркер данных соответствует фактическому значению за четвертый квартал, равному 60.

Подсказки. При остановке указателя над каким-либо элементом диаграммы появляется подсказка с названием элемента. Например, при остановке указателя над легендой появляется подсказка «Легенда».

Создание диаграммы

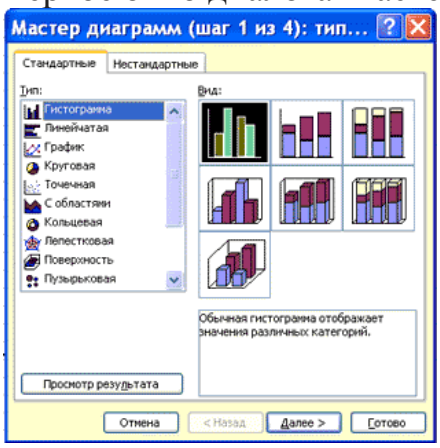
Чтобы создать диаграмму, необходимо, прежде всего, ввести данные для диаграммы на лист. Затем выделите любые ячейки, которые содержат исходные данные диаграммы. Далее в меню Вставка выберите команду Диаграмма или нажмите кнопку Мастер диаграмм на стандартной панели инструментов. В любом случае Excel выведет на экран первое окно мастера диаграмм. С помощью четырех окон диалога мастер диаграмм соберет всю информацию,

необходимую Excel для построения диаграммы. Кроме того, можно создать диаграмму за один шаг без использования мастера диаграмм. При создании таким способом диаграммы используются стандартные тип и параметры форматирования, которые позже можно изменить.

Создание диаграммы за один шаг. Самый быстрый способ для создания листа диаграммы, использующего стандартный тип диаграммы, выделите необходимые данные и нажмите клавишу F11. В этом случае лист диаграммы – это лист книги, содержащий только диаграмму.

Шаг 1. Выбор типа диаграммы.

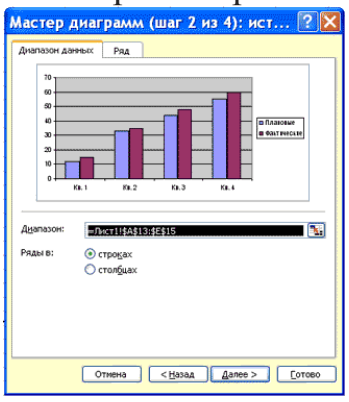
Первое окно диалога Мастера диаграмм, предлагает выбрать тип диаграммы.



Это окно диалога содержит две вкладки: одну для стандартных и другую для нестандартных типов диаграмм.

Шаг 2. Задание исходных данных диаграммы.

Во втором окне диалога мастера диаграмм можно задать данные, используемые Excel при построении диаграммы.



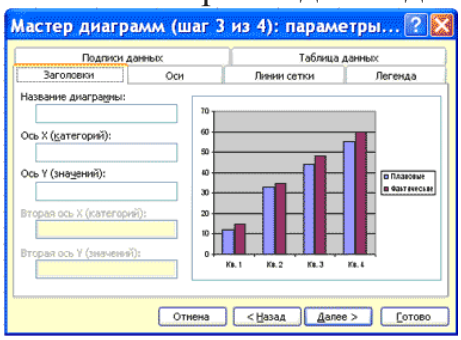
Второе окно диалога Мастера диаграмм позволяет задать исходный диапазон и расположение в нем рядов данных. Если перед запуском Мастера был выделен диапазон с исходными данными, то это поле будет содержать ссылку на выделенный диапазон. Excel выводит подвижную рамку вокруг исходного диапазона. Если по каким-то причинам исходный диапазон указан неправильно,

выделите нужный диапазон и введите его прямо в окне диалога Мастера диаграмм. Excel обычно выбирает ориентацию рядов, предполагая, что диаграмма должна содержать меньше рядов, чем точек. Просматривая образец при разной ориентации рядов, можно выбрать наиболее эффективный способ отображения данных в создаваемой диаграмме.

Второе окно диалога Мастера диаграмм, как и первое, содержит две вкладки. Чтобы убедиться, что Excel использует правильные имена и диапазоны ячеек, для каждого ряда данных, можно перейти на вкладку Ряд. Ннажмите кнопку. Далее, чтобы перейти к следующему шагу.

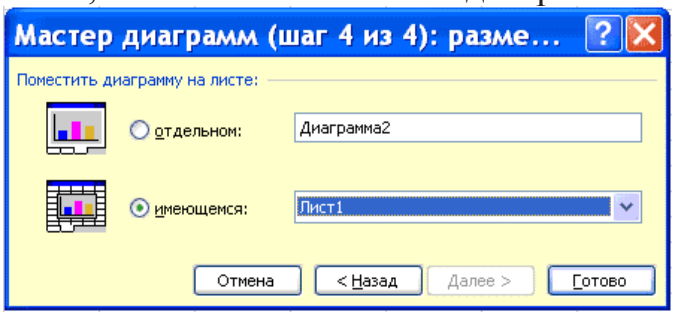
Шаг 3. Задание параметров диаграммы.

Третье окно диалога Мастера диаграмм содержит шесть вкладок. Они позволяют задать характеристики осей, название диаграммы и заголовки для ее осей, легенду, подписи значений в рядах данных и т.д. Все это можно выполнить при создании диаграммы или после ее построения.



Шаг 4. Размещение диаграммы.

Excel может внедрить диаграмму в рабочий лист или помесить ее на отдельном листе, так называемом листе диаграммы.



После построения диаграммы ее можно отредактировать в режиме редактирования диаграммы. Для этого нужно дважды щелкнуть кнопку мыши на диаграмме или воспользоваться контекстным меню.

Математические приложения Excel

К типичным математическим приложениям Excel относятся:

- структуризация и первичная логическая обработка данных;
- статистическая обработка данных, анализ и прогнозирование;

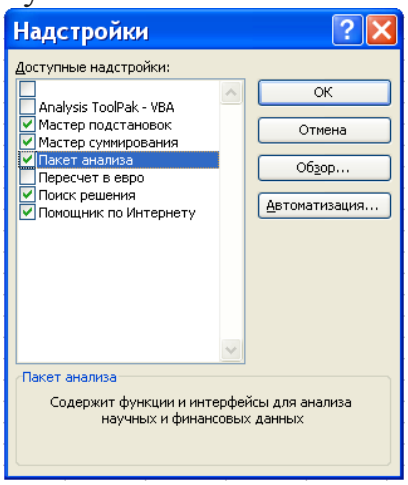
- проведение финансово-экономических расчетов;
- решение уравнений и оптимизационных задач.

Структуризация и первичная логическая обработка данных

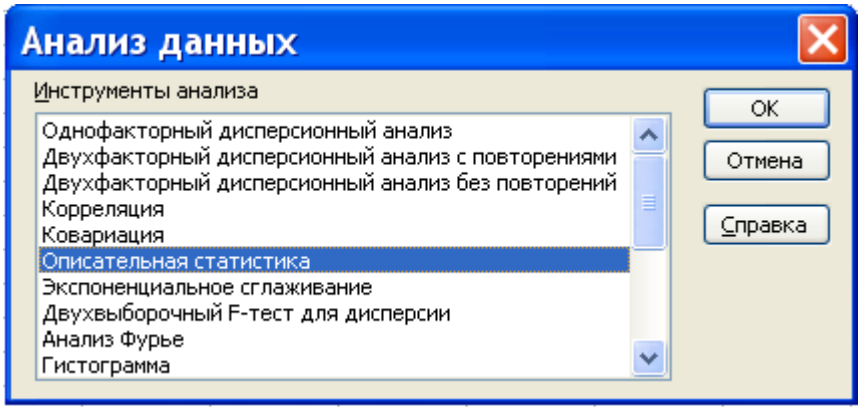
Списки в Excel являются примером формализованной структуры у исходной информации. Список - это содержащая в рабочем листе Excel таблица, данные в строках которой имеют однородную структуру или данные одного типа. К традиционным задачам первичной или предварительной логической обработки данных относятся сортировка и выборка (фильтрация) по заданному критерию.

Статистическая обработка данных, анализ и прогнозирование

Функции, реализующие статистические методы обработки и анализа данных, в Excel реализованы в виде специальных программных средств - надстройки Пакета анализа, которая входит в поставку Microsoft Office и может устанавливаться по желанию пользователей. Установка надстройки Пакет анализа осуществляется так же, как и установка других надстроек с помощью команды Сервис/Надстройка. Далее необходимо установить флажок перед пунктом Пакет анализа и нажать ОК.



После успешной установки надстройки в меню Сервис появится пункт: Анализ данных, а в окне мастера функций становится доступной категория функций - Статистические.

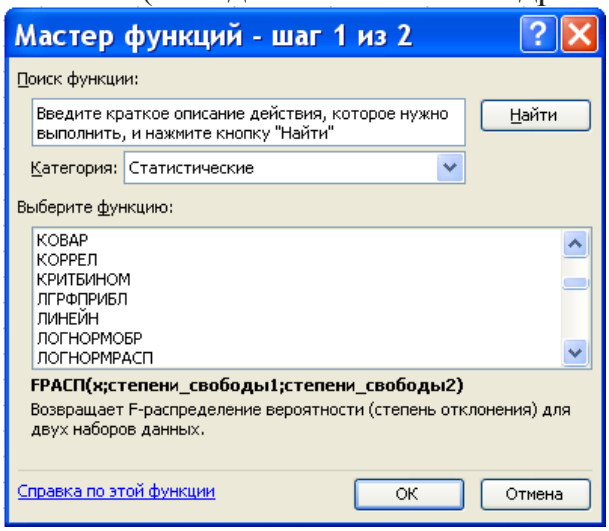


Целью статистического исследования является обнаружение и исследование соотношений между статистическими данными и их использование для изучения, прогнозирования и принятия решений. Фундаментальным понятием статистического анализа являются понятия вероятности и случайной величины. Excel не предназначен для комплексного статистического анализа и обработки данных, но с помощью команд, доступных из окна Анализ данных можно провести:

- описательный стстистический анализ (описательная статистика);
- ранжирование данных (Ранг и персентиль);
- графический анализ (Гистограмма);
- прогнозирование данных (Скользящее среднее. Экспоненциальное сглаживание);
- регрессионный анализ (Регрессия) и т.д.

Статистические функции для регрессионного анализа из категории Статистические в окне мастера функций:

- ЛИНЕЙН(знач.У; знач.Х; константа;стат.) - Определяет параметры линейного тренда для заданного массива;
- ТЕНДЕНЦИЯ(знач.У;знач.Х; новые знач.Х; константа;) - Определяет предсказанные значения в соответствии с линейным трендомдля заданного массива (метод наименьших квадратов) и многие другие.



Проведение финансово-экономических расчетов

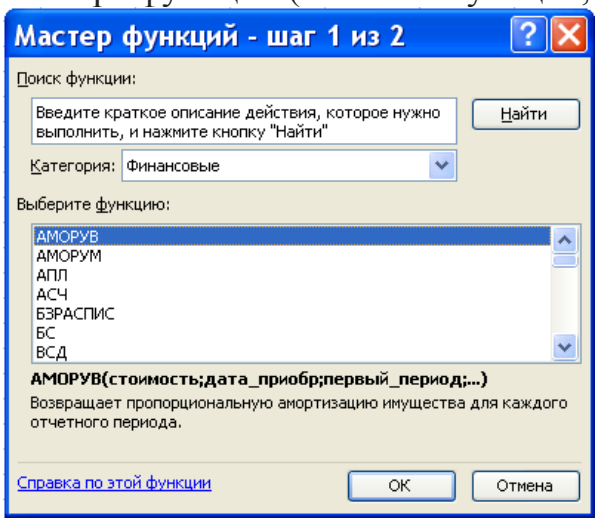
Одной из важнейших сфер приложения Excel - это осуществление финансовых рачетов. Финансовые вычисления включают в себя всю совокупность методов и расчетов, используемых при принятии управленческих решений (от элементарных арифметических операций до сложных алгоритмов построения многокритериальных моделей, позволяющих получить оптимальные характеристики коммерческих сделок и т.д.).

В Excel реализованы встроенные и дополнительные финансовые функции. Для применения дополнительных функций необходимо установить надстройку Пакет анализа.

По типу решаемых задач все финансовые функции Excel можно разделить на условные группы:

- функции для анализа инвестиционных проектов;
- функции для анализа ценных бумаг;
- функции для расчета амортизационных платежей;
- вспомогательные функции.

Применение функций для финансовых расчетов осуществляется с помощью мастера функций (Вставка/Функция, выбрать категорию Финансовые)



Порядок выполнения работы.

1. Изучите теоретический материал.

Внимание!: пп.2-12 предназначены для выполнения студентами, ранее не изучавшими и не работавшими в Excel при изучения курса «Информатика» в рамках образовательного стандарта основного общего образования, и выполняются в рамках самостоятельной работы, под консультационным наблюдением и контролем преподавателя.

2. **Практическое задание № 1.** Первое знакомство с Excel.

Задание 1. *Запустите Excel.*

Для вызова Excel необходимо дважды щелкнуть кнопкой мыши на пиктограмме Microsoft Excel, которая обычно располагается в панели Microsoft Office.

Задание 2. *Разверните окно Excel на весь экран и внимательно рассмотрите его.*

Верхняя строка — строка заголовка с кнопками управления. *Вторая строка* — меню Excel.

Третья и четвертая строки — панели инструментов Стандартная и Форматирование.

Прочитайте назначение кнопок панели инструментов **Стандартная, Форматирование** медленно перемещая курсор мыши по кнопкам. *Пятая строка* — строка ввода и редактирования или строка формул. *Строки между пятой и последней* заняты рабочим листом электронной таблицы. Строки и столбцы таблицы имеют определенные обозначения. *Нижняя строка* — строка состояния. В крайней левой позиции нижней строки отображается индикатор режима работы Excel. Например, когда Excel ожидает ввода данных, то находится в режиме «готов» и индикатор режима показывает «Готов».

3. **Практическое задание №2.** *Освойте работу с меню Excel.*

Чтобы войти в меню, необходимо подвести курсор к нужному пункту меню и нажать левую клавишу мыши.

Аналогично выбираются необходимые команды подменю и раскрываются вкладки, а также устанавливаются флажки.

>В меню **Сервис** выберите команду **Параметры** и раскройте вкладку **Правка.**

> Проверьте, установлен ли флажок «Разрешить перетаскивание ячеек». Если нет, то установите его и нажмите кнопку **ОК.**

(Далее в тексте подобные действия по работе с меню будут описываться в краткой форме: **Сервис, Параметры, Правка, Разрешить перетаскивание ячеек [X], ОК.)**

Щелчок мыши вне меню приводит к выходу из него и закрытию подменю.

4. **Практическое задание №3.** *Познакомьтесь с основными приемами редактирования таблиц.*

При использовании меню необходимо выделить строки или столбцы и выполнить команды **Формат, Строка, Размер или Формат, Столбец, Размер.** При помощи мыши измените ширину столбца А так, чтобы текст был виден полностью, а ширину столбцов В, С, D сделайте минимальной. При помощи меню измените высоту строки номер 1 и сделайте ее равной 30.> 6.3. Сделайте высоту строки номер 1 первоначальной (12, 75). Редактирование данных может осуществляться как в процессе ввода в ячейку, так и после ввода.

5. Практическое задание №4. *Освойте действия с таблицей в целом: Сохранить, Закрыть, Создать, Открыть.*

Сохраните таблицу на рабочем диске в личном каталоге под именем work1. Уберите документ с экрана. Вернитесь к своему документу workl. Закройте файл.

Завершите работу с Excel. Для выхода из Excel можно воспользоваться одним из следующих способов: 1) командой меню **Файл, Выход.;** 2) из системного меню — команда Закрыть., 3) с клавиатуры — AII+F4.

6. Практическое задание №5. *Табулирование функций*

Для дальнейшего знакомства с Excel рассмотрим **задачу табулирования функции.**

Постановка задачи: вычислить значения функции y=k* (x^2-l)/(x^2+l) для всех x на интервале [-2;2] с шагом 0,2 при k=10.
Решение должно быть получено в виде таблицы:

таб.1

№	x	k	$y_1=x^2-1$	$y_2=x^2+1$	$y=k*(y_1/y_2)$

Прежде чем перейти к выполнению задачи, познакомьтесь со способами адресации в Excel.

Абсолютная, относительная и смешанная адресации ячеек и блоков. При обращении к ячейке можно использовать описанные ранее способы: B3, A1 : G9 и т.д. Такая адресация называется относительной. При ее использовании в формулах Excel запоминает расположение относительно текущей ячейки. Так, например, когда вы вводите формулу =B1+B2 в ячейку B4, то Excel интерпретирует формулу как «прибавить содержимое ячейки, расположенной тремя рядами выше, к содержимому ячейки, расположенной двумя рядами выше». Если вы скопировали формулу =B1+B2 из ячейки B4 в C4, Excel также интерпретирует формулу как «прибавить содержимое ячейки, расположенной тремя рядами выше, к содержимому ячейки двумя рядами выше». Таким образом, формула в ячейке C4 примет вид =C1+C2.Если при копировании формул вы пожелаете сохранить ссылку на конкретную ячейку или область, то вам необходимо воспользоваться абсолютной адресацией. Для ее задания необходимо перед именем столбца и перед номером строки ввести символ $. Например: $B$4 или $C$2:$E$48 и т. д.

Смешанная адресация. Символ $ ставится только там, где он необходим. Например: B$4 или $C2. Тогда при копировании один параметр адреса изменяется, а другой — нет.

7. Практическое задание №6. Шрифтовое оформление текста

Символы любой ячейки или блока можно оформить разными шрифтами Для этого необходимо выделить ячейку или блок, а затем воспользоваться кнопками из панели **Форматирование** или командой меню **Формат, Ячейки, Шрифт**. Для заголовка «Таблицы» задайте шрифт Courier New Cyr, размер шрифта 14, полужирный. Используйте кнопки панели инструментов **Форматирование,** для заголовков «основная» и «вспомогательная» задайте шрифт Courer New Cyr, размер шрифта 12, полужирный. Используйте команды меню **Формат, Ячейки, Шрифт**; для шапок таблиц установите шрифт Courier New Cyr, размер шрифта 12, курсив. Любым способом.. Подгоните ширину столбцов так, чтобы текст помещался полностью. Произведите выравнивание надписей шапок по центру.

Выравнивание

Содержимое любой ячейки можно выровнять по левому или правому краю. по центру (по горизонтали и вертикали), а также можно задать необходимую ориентацию текста (снизу вверх, сверху вниз и т. д.). Для задания необходимой

ориентации кнопки в панели **Форматирование** или команда меню **Формат, Ячейки, Выравнивание.** Задайте рамки для основной и вспомогательной таблиц, используя кнопки панели инструментов **Форматирование.**

Для задания рамки используется кнопка в панели форматирование или меню **Формат, Ячейки, Рамка.** Задайте фон заполнения внутри таблиц - желтый, фон заполнения шапок таблиц - малиновый. **Фон** Содержимое любой ячейки или блока может иметь необходимый ф о н (тип штриховки, цвет штриховки, цвет фона). Для задания фона используется кнопка в панели **Форматирование** или команда меню **Формат, Ячейки, Вид.**

8. Практическое задание №7. В рецепт входят лекарства шести видов: «Л», «Б», «М», «В», «Р» и «Ю». Известен общий вес и процентный состав, а также запас каждой компонеты. Необходимо рассчитать: состав рецепта в граммах (сколько грамм каждого вида содержится в итоге), количество рецептов, которое можно реализовать из каждого вида препаратов, максимальное количество рецептов, которые можно составить из имеющегося запаса лекарств, количество оставшихся лекарств каждого из видов (в граммах) после изготовления максимального числа рецептов.

Виды лекарств	Запас каждого вида (гр)	Количество из каждого вида	Остаток
«Л»	56784300		
«Б»	4632100		
«М»	85632018		
«В»	57641900		
«Р»	67849038		
«Ю»	5879032		
Общий вес рецепта (в граммах)		650	
	Состав рецепта		
	В процентах	*В граммах*	
«Л»	15		
«Б»	10		
«М»	25		
«В»	30		
«Р»	25		
«Ю»	5		
	Максимальное количество выписанных рецептов		

9. Практическое задание №8. Защита ячеек

В Excel можно защитить от изменения всю рабочую книгу, лист или некоторые ячейки. Защита делает невозможным изменение информации, до тех пор пока она не отключена. Обычно защищают данные, которые не должны изменяться (расчетные формулы, заголовки, шапки таблиц).

Установка защиты выполняется в два действия:

1) отключают защиту (блокировку) с ячеек, подлежащих последующей корректировке;

2) включают защиту листа или книги.

После этих действий можно корректировать и заполнять только те ячейки, которые ранее были разблокированы.

Разблокировка (блокировка) ячеек. Выделите блок. Выполните команду **Формат, Ячейки, Защита**, а затем в диалоговом окне выключите (включите) параметр **Заблокировать.** Включение (снятие) защиты с листа или книги.

Выполните команду **Сервис, Защита, Защитить лист (книгу)** (для отключения: **Сервис, Защита, Снять защиту листа (книги)).** Выделите блок H4:J4 и снимите блокировку.

Выполните команду **Формат. Ячейки, Защита**, убрать знак [x] в окне **Блокировка.** Защитите лист. Выполните команду **Сервис, Защита, Защитить лист. Ок.** В результате действий заблокируется вся основная таблица и шапка вспомогательной. Попробуйте изменить значения в ячейках: в ячейке A4 с 1 на 10.

Это невозможно; значение шага во вспомогательной таблице с 0,2 на 0,5. Это возможно. В основной таблице произошел пересчет; измените текст «step» в ячейке 13 на текст "шаг". Каков результат? Почему? > верните начальное значение шага 0,2. *Сохраните файл под старым именем.* Воспользуйтесь кнопкой **Сохранить** на панели инструментов **Стандартная.** *Снимите защиту с листа*

Выполните команду **Сервис, Защита, Снять защиту листа.**

10. Практическое задание №9. Составление штатного расписания хозрасчетной больницы.

*Постановка задачи***:** заведующий хозрасчетной больницей должен составить штатное расписание, т. е. определить, сколько сотрудников, на каких должностях и каким окладом он должен принять на работу. Общий месячный фонд зарплат составляет 10 000 тысяч. Построим **модель решения** этой задачи. Поясним, что является исходными данными. Казалось бы, ничего не дано, кроме общего фонда заработной платы. Однако заведующему больницей известно больше: он знает, что для нормальной работы больницы нужно 5—7 санитарок, 8—10 медсестер, 10—12 врачей, 1 заведующий аптекой, 3 заведующих отделениями. 1 главный врач, 1 заведующий хозяйством, 1 заведующий больницей. На некоторых должностях число людей может меняться. Например, зная, что найти санитарок трудно, руководитель может принять решение о сокращении числа санитарок, чтобы увеличить оклад каждой из них.

Итак, заведующий принимает следующую модель задачи. За основу берется оклад санитарки, а все остальные вычисляются исходя из него: во столько-то раз или столько-то больше. Говоря математическим языком, каждый оклад является линейной функцией от оклада санитарки: A*C+B, где C — оклад санитарки; A и B—коэффициенты, которые для каждой должности определяются решением совета трудового коллектива. Допустим, совет решил, что **медсестра** должна получать в 1,5 раза больше санитарки (A=1,5, B=0):**врач**

— в 3 раза больше санитарки (B=0, A=3); **заведующий отделением** — на 30 тысяч больше, чем врач (A=3, B=30);**заведующий аптекой** — в 2 раза больше санитарки (A=2, B=0): **заведующий хозяйством —** на 40 тысч больше медсестры (A=1.5, B=40); **главный врач** — в 4 раза больше санитарки (A=4. B=0);**заведующий больницей** —на 20 тысяч больше главного врача (A=4. B=20). Задав количество человек на каждой должности, можно составить уравнение:

Nl*(Al*C+Bl) +N2*(A2*C+B2)+...+N8*(A8*C+B8) = l0000,

где NI — количество санитарок; N2 — количество медсестер и т. д. В этом уравнении нам известны A1...A8 и B1...B8. а неизвестны C и NI...NI.

Ясно, что решить такое уравнение известными методами не удастся, да и единственно верного решения нет. Остается решать уравнение путем подбора. Взяв первоначально какие-либо приемлемые значения неизвестных, подсчитаем сумму. Если эта сумма равна фонду заработной платы, то нам повезло. Если фонд зарплаты превышен, то можно снизить оклад санитарки либо отказаться от услуг какого-либо работника **и** т. д. Проделать такую работу в ручную трудно. Но нам поможет электронная таблица.

Ход работы

1. Отведите для каждой должности одну строку и запишите названия должностей в столбец A
2. В столбцах B и C укажите соответственно коэффициенты A и B.
3. В ячейку H5 занесите заработную плату санитарки (в формате с фиксированной точкой и двумя знаками после нее).
4. В столбце D вычислите заработную плату для каждой должности по формуле A*C+B.

Обратите внимание! Этот столбец должен заполняться формулами с использованием абсолютной ссылки на ячейку H5, в которой указана зарплата санитарки. Изменение содержимого этой ячейки должно приводить к изменению содержимого всего столбца D и пересчету всей таблицы.

5. В столбце E укажите количество сотрудников на соответствующих должностях в соответствии со штатным расписанием.
6. В столбце F вычислите заработную плату всех рабочих данной должности. Тогда сумма элементов столбца F даст суммарный фонд заработной платы.

Данные в столбцах D, *F* должны быть представлены в формате с фиксированной точкой и двумя знаками после нее.

7. Если расчетный фонд заработной платы не равен заданному, то внесите изменения в зарплату санитарки или меняйте количество сотрудников в пределах штатного расписания, затем осуществляйте перерасчет - до тех пор, пока сумма не будет равна заданному фонду.

8. Сохраните таблицу в личном каталоге под именем work3.
9. После получения удовлетворительного результата отредактируйте таблицу. Дайте заголовок таблице «Штатное расписание хозрасчетной больницы. Оформите таблицу, используя автоформатирование. Для этого: выделите всю

таблицу, включая заголовки; выберите пункт меню **Формат, Автоформат**; выберите удовлетворяющий вас формат.
10. Сохраните отредактированную таблицу в личном каталоге под именем hospital.
11. Распечатайте отредактированную таблицу hospital. Воспользуйтесь режимом предварительного просмотра. При просмотре выберите ландшафтное расположение и подберите оптимальную ширину полей.

11. Практическое задание №10.

Задача Для некоторой аптеки известна заработная плата сотрудников и их стаж работы. Необходимо рассчитать: величину премии, которая составляет 75% от заработной платы для сотрудников со стажем работы более 5 лет. итоговую сумму, которую надо выделить для выплаты премии.

№	Ф.И.О.	Зарплата	Стаж	Премия
1	Иванов И.Б.	1590	2	
2	Петров А.П.	3625	4	
3	Сидоров Р.Н.	2000	8	
4	Козлов Н.Н.	3126	7	
5	Орлова О.М.	3570	6	
6	Медведева А.П.	2245	4	
7	Петухова Е.Н.	5734	10	
8	Коровина Т.Н.	3651	7	
9	Чуйкова И.П.	1327	6	
10	Сидорова В.М.	2438	4	
	ИТОГО:			

Решение
Для этого выполните следующие действия:
Запустите табличный редактор Excel, выполнив команду: **Пуск → Программы → Microsoft Excel**
Заполните таблицу как показано в образце:
Рассчитайте величину премии. Для этого выбелите ячейку **E2** и вызовите диалоговое окно Вставка функции, щелкнув по пиктограмме: f_x или выполнив команду: **Вставка → Функция.**
Выберите категорию **Логические** и в списке выберите функцию **ЕСЛИ.**
В поле ввода **Логическое выражение** наберите условие **D2 > 5** (стаж работы больше 5 лет)

В поле ввода **Значение _если_истина** наберите формулу **C2*0,75** (значение премии =75% от заработной платы)
В поле ввода **Значение_если_ложь** наберите **0** (премия не начисляется тем, у кого стаж меньше или равен 5)
Щелкните по кнопке **ОК**.
Скопируйте формулу, протянув ее за правый нижний маркер, в ячейки **E3:E11.**
Рассчитайте итоговую сумму премий. Для этого: выделите ячейку **E12** и нажмите пиктограмму Автосуммы  на панели инструментов Стандартная. При этом произойдет выделение диапазона ячеек **E2:E11**, которые находятся выше ячейки **E12**. Нажмите клавишу **Enter,** чтобы подтвердить суммирование.
Сохраните таблицу в папке с названием вашей группы, имя файла **Ведомость 1.**

12. Практическое задание №11. Знакомство с графическими возможностями Excel. Построение диаграмм.
Диаграммы — это удобное средство графического представления данных. Они позволяют оценить имеющиеся величины лучше, чем самое внимательное изучение каждой ячейки рабочего листа. Диаграмма помогает обнаружить ошибку в данных, закравшуюся в какую-нибудь ячейку. Excel поддерживает 14 типов различных двух- и трехмерных диаграмм. Создать диаграмму или график легче всего с помощью Мастера диаграмм.
Это функция Excel, которая с помощью пяти диалоговых окон позволяет получить всю необходимую информацию для построения диаграммы или графика и внедрения его в рабочий лист.
Щелкните по кнопке **Мастер диаграмм в** панели инструментов **Стандартная.**
Excel просит указать, где в обрабатываемой таблице вы хотите разместить диаграмму.
> Укажите курсором на ячейку G18.
> Нажмите и удерживайте нажатой левую кнопку мыши.
> Выделите ячейки G18: M35.
> Отпустите левую кнопку мыши.
Шаг 1
Здесь Excel выводит первое диалоговое окно Мастера диаграмм «Шаг 1 из 5». С помощью этого окна выделите ячейки, содержимое которых вы хотите представить в диаграмме. Если выделяемую область закрывает диалоговое окно, то, щелкнув по зоне заголовка и не отпуская кнопки мыши, передвиньте окно.
> С помощью мыши выделите группу ячеек G16:M17.
> Щелкните по кнопке <Шаг> в диалоговом окне Мастер Диаграмм.
Если окажется , что вы выделили не те или не все необходимые ячейки, то можно на этом месте еще раз произвести выбор.
Шаг 2

В следующем диалоговом окне показаны различные типы диаграмм, которые умеет строить Excel. Из них нужно выбрать, диаграмму какого типа вы хотите создать.

> Выберите тип **Гистограмма** и щелкните по кнопке Шаг>.

Шаг 3

Содержимое третьего диалогового окна зависит от того, какой тип диаграммы вы выбрали.

>Выберите формат 7 и щелкните по кнопке Шаг>

Шаг 4

Четвертое диалоговое окно - это догадка Excel о том, как она должна использовать выделенные данные.

На этом шаге следует указать, где находятся данные — в столбцах или в строках, и указать интервал, содержащий названия рядов данных для легенды. Легенда показывает названия и маркеры данных справа от диаграммы.

Если полученный результат вас не устраивает, то можно вернуться к предыдущему диалогу, нажав кнопку **<Шаг>.**

Например, в нашем образце диаграммы Мастер диаграмм правильно определил, что данные представлены в виде строк, и менять ничего не надо.

Для легенды в нашей таблице не было введено никаких названий, поэтому рядом с меткой стоит название «Ряд 1», принятое по умолчанию. Менять количество столбцов для текста легенды здесь тоже не нужно.

> Щелкните по кнопке **<Шаг>.**

Шаг 5

Это последнее диалоговое окно Excel. В нем необходимо указать, следует ли добавлять к тексту легенду с названиями и маркерами данных, а также ввести названия диаграммы, осей X и Y.

> Щелкните по переключателю **Нет** при вопросе **Добавить легенду.**
> В окно **Название диаграммы** введите «Население Москвы (в тыс. чел.)».
> Щелкните по кнопке **Закончить.**

Вы получили диаграмму, внедренную в ваш рабочий лист. Если вас что-то не устраивает в построенной диаграмме, то ее можно отредактировать.

Изменение размеров диаграммы

Часто бывает весьма затруднительно определить наилучшие размеры внедренной диаграммы до того, как вы увидите представленные на ней данные. Поэтому часто приходится изменять размеры и пропорции внедренной диаграммы для того, чтобы придать ей хороший вид или облегчить ее редактирование.

> Сделайте одиночный щелчок по диаграмме. На рамке диаграммы появятся маркеры выделения - маленькие черные квадратики в углах и на сторонах рамки.

> «Зацепившись» курсором мыши за рамку, сдвиньте диаграмму в столбец F.

> Установите указатель мыши на маркер справа (указатель мыши при этом изменяет свою форму на двунаправленную стрелку) и растяните ее до столбца О.

«Протаскивание» маркера, расположенного на середине стороны, позволяет изменять вертикальные или горизонтальные размеры диаграммы. «Протаскивание» углового маркера позволяет изменять вертикальные и горизонтальные размеры диаграммы одновременно.

После того как выбран тип диаграммы с помощью Мастера диаграмм, Excel предоставляет большие возможности для изменения ее содержимого и вида.

Изменение типа диаграммы

Сначала необходимо двойным щелчком выбрать диаграмму для редактирования. Вокруг диаграммы появится серая штриховая рамка. Затем с помощью панели инструментов **Диаграмма** можно изменить тип диаграммы.

Укажите на любое место в диаграмме и дважды щелкните мышью.

Выведите на экран панель инструментов **Диаграмма: Вид, Панели инструментов, Диаграмма.**

Щелкните по кнопке **Тип диаграммы,** которая содержит список различных видов диаграмм.

Вид и названия диаграмм те же, что на шаге 3. Для того чтобы построить новую диаграмму по имеющимся данным, просто выберите желаемый тип.

Щелкните по кнопке объемной гистограммы.

Наша плоскостная гистограмма преобразуется в объемную Не все типы диаграмм подходят для наглядного представления данных, а некоторые невозможно построить. Например, по нашим данным диаграмму типа X-Y точечная построить нельзя. Некоторые типы объемных диаграмм тоже могут плохо отразить данные и привести к неразберихе.

Попробуйте различные типы диаграмм и подберите наиболее наглядный из них.

Внимание! Если в результате экспериментов вы испортите диаграмму, то удалите ее и начните построение сначала. Для удаления следует один раз щелкнуть на диаграмме мышью, а затем нажать клавишу Del.

Щелкните по кнопке **Страница** и выберите вкладку **Колонтитулы.** Снимите верхний и нижний колонтитулы, выбрав слово **Нет** в списке.

Выберите ландшафтную ориентацию страницы.

Щелкните по кнопке **Поля** и убедитесь) что диаграмма помещается на странице.

Убедитесь, что принтер подключен, и нажмите на кнопку **Печать.**

Редактирование диаграмм

Для редактирования диаграмму надо *выбрать* (двойным щелчком).

Диаграмма состоит из нескольких частей, называемых элементами. К ним относятся:

• область построения диаграммы; • область диаграммы;

• легенда; • заголовок;

• метки данных; • ряды данных.

Для редактирования элемента его прежде всего необходимо *выбрать*. Выбрать элемент можно при помощи мыши или нажатием клавиш управления курсором.

Выбранный элемент отмечается маленькими черными квадратиками. После выбора элемента при нажатии правой кнопки мыши появляется контекстно-зависимое меню — индивидуальное для каждого элемента. С его помощью можно производить редактирование.

Изменение размеров и перемещение элементов диаграммы

Изменять размеры элементов диаграммы можно так же, **как и** размер самой диаграммы (см. задачу 1 работы 4). Также можно перемещать элементы внутри области построения диаграммы (предварительно их выбрав). Кроме того, всю диаграмму можно перемещать по рабочему листу.

Ознакомьтесь с элементами диаграммы.

Выберите элементы диаграммы, нажимая клавиши управления курсором, а затем правую кнопку мыши.

Обратите внимание, что контекстно-зависимое меню элемента появляется только тогда, когда курсор указывает на этот элемент.

Ознакомьтесь с элементами диаграммы и главным меню Excel при помощи мыши.

В режиме редактирования диаграммы оно изменяется.

Попробуйте поперемещать элементы диаграммы.

Вырежьте кусочки из диаграммы.

Выделите область диаграммы.

Щелкните внутри любого сектора.

Вокруг сектора появились квадратики, которые обозначают границу выделенного.

Удерживая нажатой левую кнопку мыши, "отбуксируйте» сектор в сторону на 1 см.

Вырежьте еще 2 сектора.

Добавьте заголовок к диаграмме.

Если во время построения название не было указано, то его можно добавить потом.

Вставка названий

Для этого необходимо вызвать контекстно-зависимое меню форматирования области диаграммы или пункт основного меню Excel Вставка и выбрать Вставить названия. Присоединить текст.

Здесь название можно ввести двумя способами: непосредственно ввести в текстовое поле или сослаться на ячейку, содержащую его.

Выберите в меню **Вставить названия, Присоединить текст.**

Перейдите в строку формул.

Введите знак «°».

Установите курсор на ячейку А1 (в ней содержится наш заголовок) и нажмите Enter.

В контекстно-зависимом меню редактирования названия выберите Форматировать название диаграммы, Шрифт, Arial Cyr, полужирный, 12.

Измените цвет секторов на «узоры».

Так как при печати на черно-белом принтере цвет секторов не будет виден, то лучше использовать «узоры».

Выделите сектор диаграммы.

Вызовите контекстно-зависимое меню и выберите **Форматировать элемент данных, Вид, Закраска области. Узор.**

Подберите узоры всем секторам.

Отформатируйте легенду.

Выделите легенду и вызовите контекстно-зависимое меню.

Подберите шрифт, вид, размещение таким образом, чтобы легенда красиво выглядела на графике.

Измените размер области диаграммы.

Выделите область диаграммы с помощью клавиш управления курсором. Рамка охватывает рисунок диаграммы. Размер ее можно изменить, «буксируя» черные квадратики.

Подберите оптимальный размер области диаграммы.

Подготовьте диаграмму к печати.

Снимите режим редактирования диаграммы и перейдите в режим предварительного просмотра.

Здесь вы должны увидеть и таблицу и диаграмму.

Выберите ландшафтное расположение.

Уберите колонтитулы.

Уберите сетку: **Страница, Лист, Снять флажок [x], Печатать сетку.**

Построение графиков

Построить графики функций $Y_1 = x^2 - I$, $Y_1 = x^2 + l$, $Y=10x(Y_1/Y_2)$

Для построения обыкновенных графиков функций y = f(x) используется тип диаграммы XY-точечная. Этот тип диаграммы требует два ряда значений: X-значения должны быть расположены в левом столбце, а Y-значения — в правом. На одной диаграмме можно построить несколько графиков функций. Эта возможность используется для проведения сравнительного анализа значений Y при одних и тех же значениях X, а также для графического решения систем уравнений с двумя переменными.

На одной диаграмме постройте три совмещенных графика: $Y_1= x^2-1$. $Y_2 = x^2+1$, $Y = 10 x (Y_{I /} Y_2)$.

Отформатируйте область диаграммы.

Подберите оптимальный размер области диаграммы так, чтобы таблица значений и график размещались на одном листе в ландшафтной ориентации.

Вызовите контекстно-зависимое меню и выберите Форматировать область диаграммы. Рамка пользовательская. Закраска области. Узоры.

Выберите среднюю толщину рамки и узор (см. рис. 4.12).

Установите маркеры на графиках. Выделите линию графика и вызовите контекстно-зависимое меню Форматировать ряд, Вид, Маркер пользовательский, Стиль. Выберите необходимый маркер. Обратите внимание, что маркеры в легенде автоматически изменяются.
Отредактируйте названия осей X и Y.

13. **Практическое задание №12.** *Использование электронной таблицы для численного моделирования.*

Электронная таблица выполняет не только функцию автоматизации расчетов. Она является очень эффективным средством численного моделирования ситуации или объекта, для математического описания которых, т. е. построения математической модели, используется ряд параметров. Часть этих параметров известна, а часть рассчитывается по формулам. Меняя в различных сочетаниях значения исходных параметров, можно наблюдать за изменением расчетных параметров и анализировать получаемые результаты. Электронная таблица производит такие расчеты быстро и без ошибок, предоставляя в считанные минуты множество вариантов решения поставленной задачи, на основании которых вы выберете наиболее приемлемое.

Задача 1

Какова будет численность населения России в начале третьего тысячелетия?

Сразу ясно, что задачу не решить, если не знать, как со временем будет меняться численность населения России, т. е. необходимо иметь функцию, выражающую зависимость численности населения от времени. Обозначим эту функцию f(t). Но такая функция неизвестна, так как народонаселение зависит от многих факторов: экологии, состояния медицинского обслуживания, морали, права и даже от политической обстановки. Но, обобщив демографические данные, можно указать общий вид функции f(t): $f(t)=a*e^{b*t}$

где коэффициенты a, b - коэффициенты каждого государства, e - основание натурального логарифма.

Эта формула лишь приближенно отражает реальность. Однако слишком большая точность и не нужна. Хорошо, если численность населения будет спрогнозирована с точностью до десятков миллионов.

Как же определить a и b? Идея состоит в следующем: хотя a и b неизвестны, значение функции f(t) можно получить из статистического справочника. Зная t и f(t). можно приближенно подобрать a и b так, чтобы теоретические значения f(t), вычисленные по формуле (1), не сильно отличались от данных справочника (т. е. максимальное отклонение теоретических результатов от фактических данных не должно быть слишком большим). Каждое из отклонений — это модуль разности двух чисел: фактического и соответствующего теоретического значений f(t). Максимальное отклонение называют погрешностью. Необходимо найти такие a и b, чтобы погрешность была наименьшей.

Итак, математическая модель процесса изменения численности населения такова - предполагается, что:

1) зависимость численности населения от времени выражается формулой $f(t)=a*e^{b*t}$;
2) a = const и b = const следует считать справедливым лишь для не очень большого промежутка времени (например, 40 лет);
3) значения a и b можно найти с достаточной точностью, минимизировав погрешность.
Исходные данные: сведения из статистического справочника за период с 1960 по 1995 г. ($60 \le t \le 95$).
Результаты:
1) значения a **и** b;
2) численность населения России в 2000 г. (t = 100).
Кроме того, установлена связь между исходными данными и результатами: сначала надо найти a и b, минимизируя погрешность, а затем при этом a и b вычислить значение f(100).
Итак, математическая модель составлена. Использование электронной таблицы освобождает от составления программы. Нужно только определенным образом записать в таблицу исходные данные и математические соотношения, входящие в модель. После этого можно начать процесс численного моделирования исследуемой ситуации, т. е. подбор коэффициентов a и b в формуле (1), а затем определение численности населения.

Ход работы

Заполните таблицу

Сделайте заголовок. Заполните шапку таблицы. Столбцы A и B отведите под коэффициенты a и b соответственно. Коэффициенты a и b не изменяются стечением времени, значит, во всех ячейках столбцов A и B должны быть записаны одни и те же числа.

	Файл Правка Вид Вставка Формат Сервис Данные Окно ?					
	A	B	C	D	E	F
1	Таблица эксперимента					
2 3	a	b	Год.	Численнность статистическ.	Численность теоретическ.	Погрешность.
4 5 6 7 8 9 1 1 *1* *1*			60 70 80 90 91 92 93 94 95 100	117,5 130,1 137,6 147,4 148,5 147,7 148,7 148,4 148,3		

В столбец C занесите значения t с 1960 г.

В столбец D занесите взятые из справочника значения численности населения России (с 1960 г.).
В столбец E занесите формулу =A*EXP(B*C).

В столбец F занесите формулу =ABS(E-D).
Это модуль разности теоретического и фактического значений функции f.
Для подсчета максимальной погрешности по столбцу F в любую свободную ячейку этого столбца введите функцию max.
Всякий раз после пересчета чисел в столбце F в этой клетке будет записано максимальное из чисел этого столбца.
Задание 2. *Подберите коэффициенты a и b.*
Поместите в столбцы A и B какие-нибудь числа. При этом заполнятся столбцы E и F, причем погрешность, появившаяся в ячейках столбца F, скорее всего очень велика. Попробуйте поменять коэффициенты a и b так, чтобы погрешность уменьшилась. Для этого постарайтесь составить алгоритм поиска a и b. В составлении этого алгоритма вам поможет анализ приведенной выше формулы.
Задание 3. *Определите численность населения России в 2000 г.*
Сделав погрешность малой (на ваш взгляд), подставьте в свободную ячейку столбца C число 100. В соответствующей ячейке столбца E появится искомое число.
Задание 4. *Постройте графики.*
Постройте на одной диаграмме два графика роста численности населения: на основе статистических и теоретических данных (по формуле (1) с подобранными вами коэффициентами a и b).
Разместите диаграмму на одном листе с таблицей.
Задание 5. *Оформите таблицу на свой вкус (обрамление, заполнение, шрифты).*
Задание 6. *Воспользуйтесь режимом предварительного просмотра для печати.*
Добейтесь хорошего расположения таблицы и диаграммы на листе. Установите верхний колонтитул:
Численное моделирование. Работу выполнил <фамилия и имя>.
Задание 7. *Сохраните файл в личном каталоге.*
Задание 8. *Проанализировав данные таблицы и графики, сделайте вывод об адекватности предложенной математической модели реальному процессу, т. е. вывод о правильности описания роста населения формулой (1).*
Задание 9. *Распечатайте результаты работы на принтере.*
Задача 2
Несколько человек решили организовать видеокафе на 6 столиков по 4 места. С каждого посетителя будет взиматься плата за сеанс видеофильма и ужин (всем посетителям будет предлагаться один и тот же набор блюд). Администрация города постановила, что плата за вход не должна превышать 5$. Требуется определить такую входную плату, при которой будет получена наибольшая выручка.

Казалось бы, здесь и решать нечего. Разве не ясно, что чем больше входная плата, тем больше выручка. Вот и ответ: входная плата должна быть 50 рублей. Очень часто планирующие органы подобным образом и поступают, а в результате люди перестают посещать кафе, так как входная плата сильно завышена.

Начать надо, как всегда, с построения математической модели. В чем причины неудачи при введении максимальной входной платы? Мы предположили, что посещаемость не зависит от входной платы, и получили модель задачи, не соответствующую действительности. Значит, надо предполагать, что посещаемость зависит от платы за вход.

Обозначим входную плату через x. Тогда среднее число посетителей видеосалона является функцией от x. Обозначим эту функцию P(x). В задаче требуется найти такое значение x, при котором выручка, равная произведению входной платы на количество посетителей x *(P(x), достигает максимума. Если бы функция P(x) была известна, то найти требуемый максимум не составило бы особого труда. Но эта функция неизвестна, поэтому попробуем найти хотя бы общий ее вид. Его можно указать, обобщив опыт работы подобных кафе: $P(x)=ax^2-bx+c$.

Коэффициенты a, b и c для каждого кафе свои. Как же их определить? Проще всего найти значение c. Представьте себе невообразимое — в видеокафе пускают бесплатно (т. е. x = *O)*. Ясно, что свободных мест не будет. Следовательно, P(0) равно числу мест в кафе. С другой стороны, подставив 0 вместо x, получим P(0) = c. Значит, c равно количеству мест. В нашем случае c = 24 (6 столиков по 4 места).

Определить a и b так же просто не удается. Справочников по посещаемости видеокафе еще нет. Поэтому здесь требуется эксперимент.

Достаточно открыть кафе и установить на некоторый срок (дней на десять) определенную плату за вход. Среднее число посетителей и даст нам значение функции (приближенное). Установив другую плату за вход, найдем приближенное значение P(x) при новом x, и так несколько раз. Зависимость посещаемости от входной платы (на основе экспериментальных данных для конкретного кафе):

Входная плата x (рубли)	Среднее число посетителей сеанса P(x)
10	20
15	17,5
20	16
25	14
30	12,4
35	11
40	9,2
50	7

Пользуясь электронной таблицей, можно подобрать значения a и b. Вам останется определить, при какой входной плате выручка будет наибольшей.

14. **Практическое задание №13.** *Работа с электронной таблицей как с базой данных. Сортировка и фильтрация данных.*

Задание 1. *На листе 2 заполните таблицу, содержащую информацию об учащихся вашей группы и сохраните ее в личной папке.*

	A	B	C	D	E
1	Сведения об учащихся				
2	№ п/п	ФИО	Дата рождения	Дом. Адрес	Телефон
3					

Основные понятия баз данных

Область таблицы, содержащая заголовки столбцов и данные, можно рассматривать как базу данных. Столбцы A, B, C, D, E называются полями, а строки, начиная 3, - записями. Область A2:E2 содержит имена полей. Существуют ограничения, накладываемые на структуру базы данных:

- первый ряд базы должен содержать неповторяющиеся имена полей;
- остальные ряды базы данных должны содержать записи, которые не являются пустыми рядами:
- информация по полям (столбцам) должна быть однородной, т. е. только цифры или только текст.

Основная работа с любой базы данных заключается в поиске информации по определенным критериям. С увеличением количества записей поиск информации затрудняется. Excel позволяет упростить этот процесс путем фильтрации данных.

Фильтрация данных.

Команды **Данные, фильтр** позволяют выделять (фильтровать) нужные записи. Фильтрация возможна как через автоматический фильтр — **Автофильтр,** так и через ручной —**Усиленный.**

Автофильтр.

При использовании Автофильтра необходимо переместить курсор в область, содержащую базу данных, или выделить ее. Затем нужно выполнить команды **Данные, Фильтр, Автофильтр**. На именах полей появятся кнопки с изображением стрелок вниз. Нажимая на кнопки, можно задавать критерии фильтрации. В появляющемся подменю пункт **Все** отключает фильтрацию, а пункт **Настройка** вызывает диалоговое окно, в котором можно установить параметры фильтрации. Для одного поля могут быть заданы два условия одновременно, связанные логическим И или ИЛИ.

Задание 2. *С использованием Автофильтра осуществить поиск студентов, которые живут на указанной улице и имеют телефон.*

Выполните команды **Данные, Фильтр, Автофильтр.** На полях появились кнопки.

Нажмите на кнопку в поле **Дом. адрес.** Выберите пункт **Настройка.** В диалоговом окне задайте критерии и нажмите на кнопку ОК.
Нажмите на кнопку на поле **Телефон**. Выберите пункт **Настройка**.
В диалоговом окне задайте критерий:
Выполните команды меню **Данные, Фильтр, Показать все**.
Задание 3. *С использованием Автофильтра самостоятельно:*
осуществите поиск учеников, родившихся в указанном месяце.
Усиленный фильтр. При использовании **Усиленного фильтра** необходимо сначала определить (создать) три области: интервал списка—область базы данных; интервал критериев — область, где задаются критерии фильтрации и интервал извлечения— область, в которой будут появляться результаты фильтрации. Имена полей во всех интервалах должны точно совпадать.
Для выполнения действий по фильтрации необходимо воспользоваться командами меню **Данные, фильтр, Усиленный фильтр.** В диалоговом окне надо указать координаты интервалов.
Если необходимо получать результаты фильтрации в интервале извлечения, нужно выбрать **Копировать** на другое место.
Задание 4. *С использованием Усиленного фильтра осуществить поиск учащихся, проживающих на указанной улице и имеющих телефон, начинающийся на указанную цифру.*
>Создайте интервал критериев и интервал извлечения
>Запишите критерии поиска в интервал критериев.
>Поместите курсор в область базы данных.
>Выполните команды **Данные, Фильтр, Усиленный фильтр.**
>Установите **Копировать на другое место.**
>Проверьте правильность задания интервалов. Нажмите кнопку ОК.
Задание 5. *Сохраните результаты в файле журнал .*
Задание 6. *Распечатайте результаты работы, предварительно введя в строку 13 заголовок "Критерии поиска", а в строку 16 заголовок "Результаты поиска".*
Сортировка данных. Команды **Данные, Сортировка** позволяют упорядочивать (сортировать) базу данных. Для выполнения сортировки необходимо выделить область базы данных или поместить в нее курсор, а затем выполнить команды **Данные, Сортировка.** При этом появится диалоговое окно. Нужно установить **Есть** для **строки меток столбцов** и выбрать название поля, по которому нужно производить сортировку.
Кроме того, необходимо указать метод сортировки: по возрастанию или по убыванию и нажать кнопку ОК.
После указанных действий база будет упорядочена. Символьные поля упорядочиваются в алфавитном порядке.
Задание 8. *Отсортируйте данные в таблице в алфавитном порядке фамилий учеников.*
Задание 9. *Отсортируйте данные в таблице в порядке возрастания даты рождения.*

умеете ли вы: производить поиск информации в базе по различным критериям; производить сортировку информации.

15. **Практическое задание №14.** *Рабочая книга Excel. Связь таблиц.*

До сих пор вы работали только с одним листом рабочей книги. Часто бывает полезно использовать несколько рабочих листов. В нижней части экрана видны ярлычки листов. Если щелкнуть на ярлычке левой клавишей мыши, то указанный лист становится активным и перемещается наверх. Щелчок правой кнопкой на ярлычке вызовет меню для таких действий с листом, как перемещение, удаление, переименование и т. д. В левом нижнем углу окна рабочей книги находятся кнопки прокрутки ярлычков, с помощью которых можно переходить от одного рабочего листа к другому. Щелкнув правой кнопкой мыши на кнопках прокрутки ярлычков, можно открыть контекстно-зависимое меню для выбора нужного рабочего листа.

Постановка задачи. Необходимо создать журнал студенческой группы. Для каждого предмета отводится отдельный лист рабочей книги, который содержит список класса, текущие оценки и итоговую оценку за четверть. На отдельном листе должна быть представлена ведомость итоговых оценок за семетр, заполненная оценками с использованием ссылок на соответствующие листы по предметам. Групповой журнал кроме оценок содержит также сведения об учащихся.

Ход работы

Задание 1. *На <Листе 1> заполните и оформите таблицу*

	A	B	C	D	E	F	G	H	I	J	K	L
1		*Медицинская информатика*										
2	*№ п/п*	*ФИО студента*	*2.09*	*3.09*								*1 семестр*
3												
4												
5												
6												
7												

Для чисел в ячейках, содержащих даты проведения занятий, задайте формат Дата. Оценки за 1 -ю четверть вычислите по формуле, как среднее арифметическое текущих оценок.

Задание 2. *Сохраните таблицу в личном каталоге рабочего диска под именем jurnal.*

Задание 3. *Создайте аналогичные листы для других предметов, для чего:*

Скопируйте таблицу **Русский язык** на следующий лист, используя команды меню **Правка, Переместить/Скопировать..., Перед листом <Лист2>, Создавать копию [X].**

После выполнения команды появится лист <Лист 1 [2] >. Скопируйте таблицу еще раз, используя те же команды меню. После выполнения этой команды появится лист < Лист 1 [3] >.

Задание 4. *Переименуйте листы: <Лист 1 [1]>, <Лист 1 [2]>, <Лист 1 [3] > в названия предметов начальной школы.*

Для этого дважды щелкните на ярлычке соответствующего листа и задайте в диалоговом окне новое имя. Можно один раз щелкнуть на ярлычке правой кнопкой мыши и открыть контекстно-зависимое меню, в котором выбрать пункт **Переименовать.**

Задание 5. *На листах предметов в таблицах соответственно измените названия предметов, текущие оценки, даты.*

Связь рабочих листов

Excel позволяет использовать в таблице данные с других листов и из других таблиц. Связь между двумя листами достигается за счет введения в один лист формулы связи со ссылкой на ячейку в другом листе.

Задание 6. *На <Листе 2> создайте таблицу - ведомость итоговых оценок за 1-ый семестр, для чего:*

Переименуйте <Лист2> в лист <1четверть>

Заполните таблицу ссылками на соответствующие ячейки других листов:

В ячейку А2 занесите формулу =Медицинская информатика!А2.

Здесь: Русский язык! — ссылка на другой лист, символ «!» обязателен; А2 — адрес ячейки на листе <Медицинская информатика>, используется относительная адресация.

Размножьте формулу на последующие ячейки столбца А и соответствующие ячейки столбца В.

В ведомости заполнятся колонки «№» и «Фамилия студента».

В ячейку С3 занесите формулу =Медицинская информатика!L3.

Размножьте формулу на последующие ячейки столбца.

Столбец заполнится оценками за 1-ый семестр по медицинской информатике.

Таким образом будет установлена связь между листом <1 семестр) и листом <Медицинская информатика >.

Задание 7. *Удалите листы, которые не будут использоваться в рабочей книге (с 3 по 16).*

Для удаления листа выберите команду **Удалить** из контекстно-зависимого меню для ярлычков. Для удаления сразу нескольких рабочих листов предварительно выделите их с помощью клавиши Ctrl.

Работа с несколькими окнами

Пока информация рабочего листа занимает один экран, достаточно одного листа. Если это не так, то можно открыть несколько окон и одновременно отслеживать на экране разные области рабочего файла.

Задание 8. *Проверьте правильность заполнения таблицы.*

Откройте для просмотра еще одно окно. Выполните команды меню **Окно, Новое окно.** В новом окне выберите рабочий лист <Русский язык>.

Выполните команды меню **Окно, Упорядочить окна, Упорядочить мозаикой.** Активным всегда является только одно из окон. Для активизации другого окна нужно щелкнуть по нему мышью.

Задание 9. *Проверьте, как работает связь таблиц.*

На листе “Медицинская информатика” исправьте одному из учеников текущие оценки с 3 на 4.

Обратите внимание! Изменилась итоговая оценка за первую четверть как на листе “Медицинская информатика”, так и на листе “1-ый семестр”.

Исправьте текущие оценки выбранному ученику опять на 3.

Связь между файлами

Связь между двумя файлами достигается за счет введения в один файл формулы связи со ссылкой на ячейку в другом файле. Файл, который получает данные из другого, называется файлом назначения, а файл, который предоставляет данные, - файлом-источником.

Как только связь устанавливается, Excel копирует величину из ячейки в файле-источнике в ячейку файла-назначения. Величина в ячейке назначения автоматически обновляется.

Задание 10. *Осуществите связь между листами разных рабочих книг. Заполните столбец в ведомости оценками по русскому языку, взяв их из файла jurnal, записанного на диске a:*

Очистите от оценок столбец Русский язык, используя команды меню **Правка, Очистить,Все.**

В ячейку C3 занесите формулу: =’a:\[jurna.xlsl]Медицинская информатика’!L3.

‘A:\[jurnal.xls]Медицинская информатика’! - путь к файлу jurnal.xls и листу «Медицинская информатика». Этот путь обязательно должен быть заключен в одинарные кавычки. Имя файла должно быть заключено в квадратные скобки.

Размножьте формулу на последующие 4 ячейки столбца.

Столбец заполнился оценками по медицинской информатике языку, т.е. связь установлена.

Задание 11. *Самостоятельно заполните ведомость оценок за 1-ый семестр по другой дисциплине.*

Задание 12. *Самостоятельно найдите среднюю оценку по каждому предмету.*

16. Оформите отчет, включающий скрин-шоты конечных результатов выполненых заданий, пояснений к ним и краткие ответы не менее, чем на 5 контрольных вопросов.

Контрольные вопросы

1.Назовите основные элементы окна Excel.

2. Как вводится текст, формулы, числа в ячейки Excel?

3. Как происходит изменение размеров строк и столбцов в Excel?

4. Какие действия необходимо выполнить для выполнения операций: перемещение, копирование, удаление информации в Excel?

5. Что такое относительная, абсолютная и смешанная адресация?

6. Как осуществляется защита (и снятие) информации в ячейках Excel?
7. Что такое функции Excel, для каких целей они предназначены?
8. Какие существуют форматы чисел в Excel?
9. Как осуществляется печать таблицы на экране и принтере?
10. Что такое внедренная диаграмма в электронной таблице?
11. Какие существуют типы диаграмм в Excel?
12. Как происходит редактирование диаграмм в Excel?
13. Назовите этапы построение графиков в Excel.
14. Какие операции выполняются для построения одиночного графика?
15. Как в Excel строить и редактировать совмещенные графики?
16. Как происходит выбор оптимального решения задачи в Excel?
17. Как осуществляется проверка правильности математической модели?
18. Что такое база данных в электронной таблице?
19. Как происходит сортировка и фильтрация данных в электронной таблице?
20. Как осуществляется связь между листами одной рабочей книги?
21. Назовите основные операции для осуществления связи между файлами?

Информационные источники

1. Компьютерные видиоуроки /URL: http://compteacher.ru/microsoft-office/1131-samouchitel-excel-2010-chast-1-onlayn-obuchenie.html
2. Самоучитель работы с Excel //URL: http://www.teachvideo.ru/course/380
3. Электронные таблицы. Презентация. /URL: http://prezentacii.com/informatike/4808-elektronnye-tablicy.html

4. ТЕКСТОВЫЕ РЕДАКТОРЫ (НА ПРИМЕРЕ WORD)

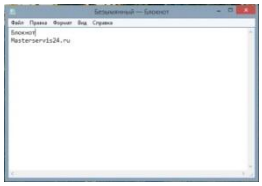
Цель работы: закрепление знаний работы с текстовыми редакторами, полученными в процессе освоения образовательного стандарта основного общего образования, на примере редактора Word, достаточными для подготовки документаций медицинского характера.

Краткие теоретические сведения.

Обзор текстовых редакторов для компьютера.

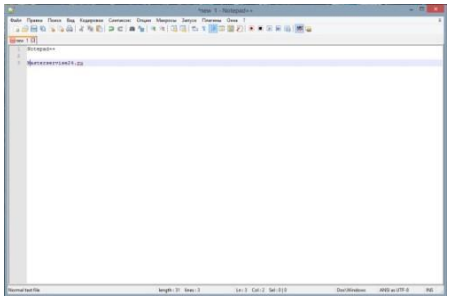
Неотъемлемой частью работы компьютера является текстовый редактор.

Блокнот. Это самый простой и незамысловатый текстовый редактор в Windows. Этот текстовый редактор используется для каких-либо заметок, небольших фраз и прочих пометок. Многие программисты в блокнот копируют различные коды, которые в Блокноте остаются в первозданном виде, так как более продвинутые текстовые редакторы могут их распознавать и преобразовывать визуально, в итоге теряя часть кода. Также в Блокнот удобно копировать пароли, ссылки и консольные команды. Блокнот поставляется наряду со стандартным предустановленным пакетом программ операционной системы, по сути, является бесплатным.



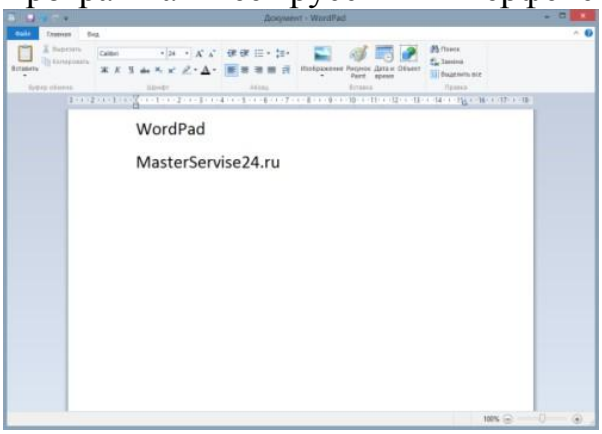
В операционной системе Linux есть свой аналог блокнота – gedit. Программа по функциям и назначению абсолютно идентична с аналогом из Windows.

Преимущество данных текстовых редакторов в том, что они просты и компактны, это идеальный вариант для пометок и хранения элементов кода. Недостатком данного редактора текста является его преимущество - излишняя простата, которая не даёт возможность производить оформление теста.

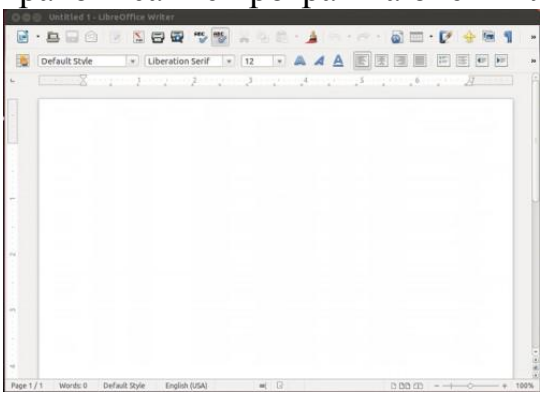


Notepad++. Для более продвинутых компьютерных пользователей лучше всего подойдёт расширенная версия блокнота – Notepad++, которая имеет большое количество функций, но при этом остаётся всё тем же блокнотом. Загрузить

блокнот можно с официального сайта программы Notepad-plus-plus.org. Программа имеет русский интерфейс и распространяется бесплатно.

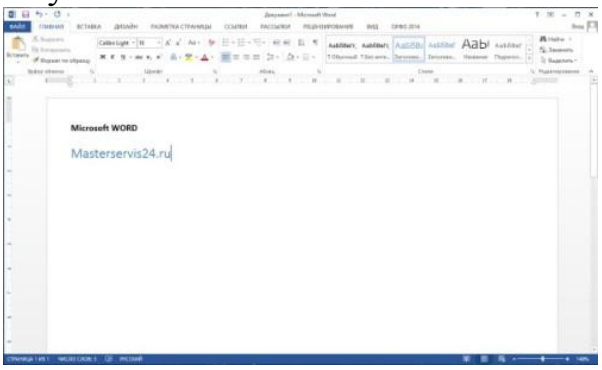


WordPad. WordPad – это ещё один стандартный текстовый редактор, который поставляется в перечне предустановленных в Windows программ. Забегая наперёд, чтобы более точно описать данную программу, стоит сказать, что WordPad это что-то среднее между блокнотом и Microsoft Word. То есть WordPad имеет простую основу, как и Блокнот, но в него включены некоторые элементы оформления текста из Word. Такое сочетание: простота + минимальный набор функций для оформления, делают его довольно привлекательным для набора простого текста, не требующего особого оформления. WordPad – идеальный вариант для тех, кому нужно набирать простые текстовые документы, используя данную программу, вы сможете сэкономить на покупке Microsoft Word, только учтите, что проверяет правописание программа очень плохо.

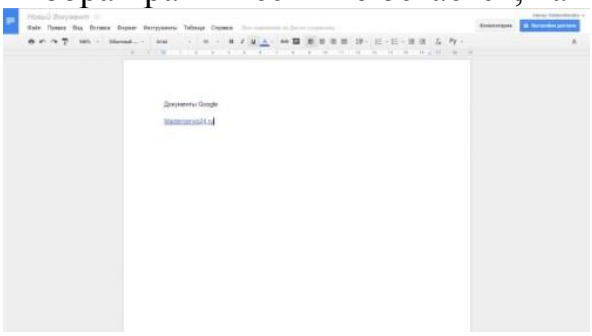


LibreOffice. Изначально данный офисный пакет был создан для операционной системы Linux, сменив устоявшийся на то время OpenOfficeOrg, который существует и поныне. Затем была выпущена версия LibreOffice под Windows. Если вкратце описать данный офисный пакет, то это Word 2003 года с не очень хорошим качеством проверки правописания. Но в то же время, что можно хотеть от бесплатного текстового редактора? Для бытовых нужд в Windows и Linux, чтобы сохранить на компьютере и распечатать красиво обрамлённый текст – это оптимальный вариант, так как не нужно тратить лишние средства на покупку офисного пакета от Microsoft. Загрузить офисный пакет для Windows

можно со странички LibereOffice, как было сказано ранее, в Linux он установлен по умолчанию.



Microsoft Word. Данное приложение неспроста является флагманом, так как даже по сей день, у него нет аналогов, которые бы смогли воплотить красивый и удобный интерфейс, с большим количеством функций, для редактирования и оформления текстов. Он по праву считается самым лучшим текстовым редактором в Windows 7 и Windows 8 и более поздние версии. Word – это незаменимый редактор текстов для любого пользователя, которому необходимо часто заниматься набором текстов и документов. Помимо этого стоит отметить хорошее качество проверки орфографии в текстах, чем не может похвастаться ни один текстовый редактор для компьютера. Единственным недостатком Word является его цена, так как за самый бюджетный офисный пакет вам необходимо будет выложить порядка 3 000 рублей, что достаточно дорого. Тем не менее, если вы часто сталкиваетесь с необходимостью набора текста, то тут у вас выбора практически не остаётся, как приобрести офисный пакет от Microsoft.



Текстовый редактор онлайн. Данный вид редакторов текста позволяет набирать тексты непосредственно на удалённом сервере в Интернете – онлайн. Преимущества очевидны – набранный текст сразу же сохраняется в облачном хранилище и поэтому доступ к тексту можно открыть сразу же нескольким пользователям (и даже редактировать вместе), а также самому иметь доступ к данным документам со всех своих устройств, которые подключены к Интернету. Благодаря этому вам не страшна потеря данных или отключение света – документ всегда доступен в Интернете.

Наилучшим онлайн редактором текста является сервис GoogleДиск. Зарегистрировав себе облачный диск, вы сможете создавать на нём текстовые

документы. Дизайн, конечно, таких документов особой красотой не блещет, но она тут особо ни к чему, так как это лишняя нагрузка на Интернет-канал, да и главное это набор текста. Проверка орфографии производится браузером, что конечно не идеальный вариант, но и не самый плохой. К преимуществам можно отнести и бесплатность сервиса, что немаловажно. Документ можно сразу же сохранить на компьютере, если в этом будет необходимость.

К бесплатным текстовым редакторам относятся

 OpenOffice.org 4.1.2 Бесплатная и качественная альтернатива офисным пакетам, в том числе и MS Office.

 Notepad++ 6.9.2 Редактор текстовых файлов с подсветкой синтаксиса огромного количества языков программирования.

 Kingsoft Writer 9.1.0.4550 Бесплатный текстовый редактор уровня MS Word.

 Notepad2 4.2.25 Быстрый и маленький по размеру текстовый редактор с подсветкой синтаксиса.

 Document.Editor 2016.4 Небольшой бесплатный редактор текстовых файлов, который отличается простотой в использовании.

 AbiWord 2.9.4 Бесплатный кроссплатформенный текстовый процессор для всех основных операционных систем.

 Spell-Checker 2.1.0.115 Spell Checker – это программа, предназначенная для проверки орфографии введенного с клавиатуры текста, а также текста находящегося в буфере обмена.

 Vim 7.4 Vim - это один из наиболее старых и мощных текстовых редакторов в мире.

 Microsoft Office Compatibility Pack 2007 File Formats 4 Приложение, которое делает возможным открытие файлов DOCX в более ранних версиях MS Office.

 Bred3 3.0.3 Bred 3.0.3 - простой и функциональный текстовый редактор.

 AkelPad 4.9.8 Простой текстовый редактор с открытым исходным кодом.

 IBM Lotus Symphony 3.0.1 Отличный редактор текстов, с помощью которого набор или редактирование уже созданного материала будет самым любимым занятием.

 SciTE 3.6.5 Полнофункциональный текстовый редактор с возможностью подсветки кода.

 TextMaker 7.0 TextMaker - текстовый редактор для замены стандартного блокнота Windows.

 TextEdit 3.0.0.483 TextEdit - текстовый редактор, поддерживающий подсветку синтаксиса и вкладки.

 Texmaker 4.5 Текстовый редактор, ориентированный на работу с документами в формате LaTeX.

 XP Text Editor 1.06.2

 ConTEXT 0.98.6 ConTEXT – это маленький, быстрый и мощный текстовый редактор.

 Bluefish 2.2.9 Продвинутый текстовый редактор, который пригодится программистам и веб-разработчикам.

 Translit Coder 2.1.77 Translit Coder – очень удобная программа, основное предназначение которой – это перевод текста,

 Atom 1.8.0 Удобный текстовый редактор с прагматичным интерфейсом и подсветкой синтаксиса.

 Иероглиф 3.7 Build 037 Иероглиф – многофункциональный текстовый редактор, ориентированный на работу с русскими текстами.

 TestReader 5.03.02 Программа является оболочкой для проведения тестирования. Основным плюсом которой является то, что текст можно набирать абсолютно в любом редакторе.

 Текстовик 1.10.1 Текстовик - небольшой текстовый редактор для замены стандартного блокнота Windows.

 Word List Creator 1.0 Create Word Lists from text Files.

 Texter 2.16 build 524 Texter - функциональный текстовый редактор, предназначенный для замены Блокнота.

 WriteMonkey 2.7.0.3 WriteMonkey - настоящий дзен-софт, текстовый редактор с урезанным пользовательским интерфейсом, который оставляет вас наедине со своими мыслями.

 NotepadTabs 4.0.0 Простой и удобный блокнот, имеющий многовкладочный интерфейс и позволяющий открыть одновременно несколько текстовых файлов в разных вкладках.

 Zim 0.65 Удобный текстовый редактор, рассчитанный на создание страниц для Wiki.

 MEdit 1.2.0 Простенький, текстовый редактор с подсветкой синтаксиса.

 LidaEd 1.2 LidaEd - текстовый редактор, поддерживающий работу с изображениями, таблицами и документами Word.

 GridinSoft Notepad LITE 3.3.1

 Notepad X 2.1.2

 Angel Writer 3.2 Build 400 Данная программа облегчит вам чтение, просмотр или редактирование документов и файлов.

 Win32Pad 1.5.10.4 Win32Pad - текстовый редактор для замены стандартного блокнота Windows.

 AleX.1 2.0 RC5 Удобный текстовый редактор.

 TED Notepad 6.0.2 Beta Функциональный текстовый редактор, поддерживающий множество горячих клавиш, позволяющий управлять буфером обмена и др.

 EditPad 7.3.8 Lite Многофункциональный текстовый редактор с поддержкой вкладок.

 Simple Text Editor (Sted) 1.3.1 Небольшой текстовый редактор призванный заменить стандартный системный Блокнот.

 Uniqway Poetic 1.4 Uniqway Poetic - текстовый редактор с возможностью подсветки синтаксиса.

 Virtual Keyboard 4.0 Небольшая, но многофункциональная программа для набора текста на различных языках при помощи компьютерной мыши.

 ВерсткаТекстаКнижкой 5.11 ВерсткаТекстаКнижкой - макрос для текстового редактора Microsoft Word, позволяющий верстать текст в виде книжки, когда на одной странице размещаются сразу два текстовых блока.

 metapad 3.6 Быстрый, небольшой по размеру и абсолютно бесплатный текстовый редактор, созданный для полной замены стандартного Блокнота Windows.

 EControl Syntax Editor 3.00 EControl Syntax Editor - функциональный текстовый редактор, ориентированный на работу с кодом.

 Rusred 11 Rusred - текстовый редактор, отличающийся от других подобных программ наличием собственной графической русифицированной клавиатуры.

 Эверест WYSIWYG 1.0.0.88 Эверест WYSIWYG - текстовый редактор, поддерживающий RTF, бинарный и собственный теговый форматы документов.

Порядок выполнения работы.

1. Изучите теоретический материал.
2. Составьте в текстовом редакторе следующий лист:

ТЕКСТ на русском языке	Объект-рисунок (блок-схема)	Объект - Excel
Формула с латинскими символами	Перевод текста на английском и китайском языках	Формула со смешанными символами
Объект-Word-art	диаграмма	Объект – рисунок – собственный рисунок средствами Word

В каждом элементе таблицы необходимо применять разное выравнивание текста и разные шрифты.

3. Скопируйте любой текст из медицинского справочника (объемом не менее 3 страниц). Выделите в нем ключевые слова (не менее трех) и оформите их в виде ссылок.
4. В тексте из.п.3 выделите название глав и организуйте автоматическое составление содержания по ним.
5. Выполните автоматическое составление реферата текста.
6. Выполните п.2 в любом текстовом редакторе, приведенном в теоретических сведениях к работе (отличном от Word).
7. Оформите отчет, включающий описание выполненных действий и скрин-шоты результатов, ответы на контрольные вопросы (не менее 3).

Контрольные вопросы.

1. Функциональное назначение графических редакторов.
2. Каковы назначение и основные функции текстовых редакторов?
3. Каковы основные методы запуска на выполнение программы текстового редактора Word?
4. Поясните структуру окна текстового редактора Word и назначение его элементов.
5. Каковы режимы отображения документа в окне редактора Word и в чем разница между ними?
6. Какие непечатаемые знаки может отображать на экране текстовый редактор Word?

7. В чем суть операций открытия и сохранения документа Word? Какова разница между командами сохранения?
8. Каковы основные методы передвижения текстового курсора по документу?
9. Какие основные методы выделения различных фрагментов текста предоставляет текстовый редактор Word? Поясните их.
10. Поясните стандартный алгоритм копирования и перемещения фрагментов текста. Каковы различия в этом алгоритме при выполнении операций копирования и перемещения?
11. В чем суть операций копирования и перемещения фрагментов текста методом перетаскивания ("Drag and Drop" – "Перетащить и Уронить")? Чем отличается процесс выполнения этим методом операций копирования и перемещения?
12. Что такое Буфер обмена, и какими особенностями он обладает?
13. В чем суть форматирования символов, абзацев и страниц?
14. В каких единицах измеряется размер шрифта?
15. Какие начертания символов возможны в документах Word?
16. Какие виды выравнивания абзацев текста допустимы в документах Word?
17. Каковы назначение и особенности использования основных элементов масштабной
линейки?
18. Какие виды списков используются в документах Word, и каковы их особенности?
19. Поясните смысл и назначение полей страницы (верхнего, нижнего, левого, правого, переплета, колонтитулов).
20. Каковы основные требования к оформлению официальных текстовых документов.

Информационные источники.

1. Как работать с редактором Word /URLL: http://www.neumeka.ru/microsoft_word.html
2. Работа с офисными программами /URLL: http://www.autoaf.ru/docs_office.htm
3. Текстовые редакторы /URLL: http://shkolo.ru/tekstovyie-redaktoryi/

5. ГРАФИЧЕСКИЕ РЕДАКТОРЫ. СОЗДАНИЕ ПРЕЗЕНТАЦИЙ

Цель работы: овладение практическими навыками работы в графическом редакторах и создании презентаций.

Краткие теоретические сведения.

Представление данных на мониторе компьютера в графическом виде впервые было реализовано в середине 50-х годов для больших ЭВМ, применявшихся в научных и военных исследованиях. С тех пор графический способ отображения данных стал неотъемлемой частью подавляющего числа компьютерных систем, в особенности персональных. Компьютерная графика-это специальная область информатики, изучающая методы и средства создания и обработки изображений с помощью программно-аппаратных вычислительных комплексов. Она охватывает все виды и формы представления изображений, доступных для восприятия человеком либо на экране монитора, либо в виде копии на внешнем носителе (бумага, киноплёнка, ткань и прочее).

Без компьютерной графики невозможно представить себе не только компьютерный, но и обычный, вполне материальный мир. На сегодняшний день компьютеры и компьютерная графика неотъемлемая часть жизни современного общества. Для примера назовём медицину (компьютерная томография), научные исследования (визуализация строения вещества, векторных полей и других данных), моделирование тканей и одежды, опытно-конструкторские разработки, рекламные щиты, цветные журналы, спецэффекты в фильмах – всё это в той или иной мере имеет отношение к компьютерной графике. Поэтому созданы программы для создания и редактирования изображений, то есть графические редакторы.

Для обработки изображений на компьютере используются специальные программы - графические редакторы. Графический редактор - это программа создания, редактирования и просмотра графических изображений.

Рассмотрим некоторые из графических редакторов:

1)*Графический редактор Paint* - простой однооконный графический редактор, который позволяет создавать и редактировать достаточно сложные рисунки. Окно графического редактора Paint имеет стандартный вид (рис.1).

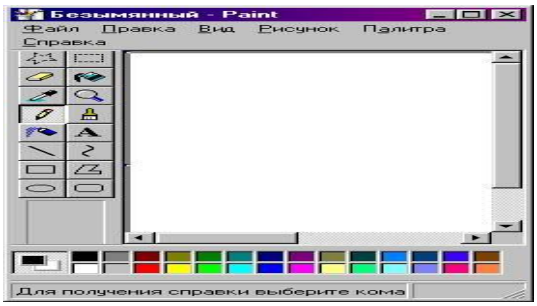


Рисунок 1

2) *Photoshop* фирмы Adobe многооконный графический редактор позволяет создавать и редактировать сложные рисунки, а также обрабатывать графические изображения (фотографии). Содержит множество фильтров для обработки фотографий (изменение яркости, контрастности и т.д.).
3) *Программа Microsoft Draw* - входящая в комплект MS Office. Эта программа служит для создания различных рисунков, схем. Обычно вызывается из MS Word.
4) *Adobe Illustrator, Corel Draw* - программы используются в издательском деле, позволяет создавать сложные векторные изображения.
Изображения в графических редакторах хранятся по-разному.
Растровое изображение хранится с помощью точек различного цвета (пикселей), которые образуют строки и столбцы. Любой пиксель имеет фиксированное положение и цвет. Хранение каждого пикселя требует некоторого количества бит информации, которое зависит от количества цветов в изображении.
Векторные изображения формируются из объектов (точка, линия, окружность и т. Д.), которые хранятся в памяти компьютера в виде графических примитивов и описывающих их математических формул.
Например, графический примитив точка задается своими координатами (X, У), линия – координатами начала (XI, Y1) и конца (X2, Y2), окружность – координатами центра (X, У) и радиусом ®, прямоугольник – величиной сторон и координатами левого верхнего угла (XI, У1) и правого нижнего угла (X2, У2) и т. Д. Для каждого примитива назначается также цвет.
Таким образом, различают растровые и векторные графические редакторы.
Векторные графические редакторы
Векторные графические изображения являются оптимальным средством для хранения высокоточных графических объектов (чертежи, схемы и т. Д.), для которых имеет значение наличие четких и ясных контуров. С векторной графикой вы сталкиваетесь, когда работаете с системами компьютерного черчения и автоматизированного проектирования, с программами обработки трехмерной графики. Все компоненты векторного изображения описываются математически, а значит – абсолютно точно. Векторные изображения, как правило, строятся вручную, однако в некоторых случаях они могут быть также получены из растровых с помощью программ трассировки. Векторные изображения не в состоянии обеспечить близкую к оригиналу реалистичность, но достоинством векторной графики является то, что файлы, хранящие векторные графические изображения, имеют сравнительно небольшой объем. Важно также, что векторные графические изображения могут быть увеличены или уменьшены без потери качества.
Растровые графические редакторы
Растровые графические редакторы являются наилучшим средством обработки фотографий и рисунков, поскольку растровые изображения обеспечивают высокую точность передачи градаций цветов и полутонов. Способ

представления растровых изображений совершенно отличен от векторных. Растровые изображения состоят из отдельных точек, называемых растром.

Такое представление изображений существует не только в цифровом виде. Растровые изображения обеспечивают максимальную реалистичность, поскольку в цифровую форму переводится каждый мельчайший фрагмент оригинала. Такие изображения сохраняются в файлах гораздо большего объёма, чем векторные, поскольку в них запоминается информация о каждом пикселе изображения. Таким образом, качество растровых изображений зависит от их размера (числа пикселей по горизонтали и вертикали) и количества цветов, которые могут принимать пиксели. Как следствие того, что они состоят из пикселей фиксированного размера, свободное масштабирование без потери качества к ним не применимо. Эта особенность, а также сама структура растровых изображений несколько затрудняет их редактирование и обработку.

Кроме создания изображений графические редакторы позволяют хранить полученные изображения. Для этого существуют файлы, которые различны для векторных и растровых графических редакторов.

Как правило, файлы для хранения растровых графических изображений логически состоят из двух частей: заголовка и области данных. В заголовке указаны данные о формате файла, изображения по горизонтали, по вертикали: количество цветов, палитра и т.д. В области данных закладываются цвета пикселов.

В настоящее время наиболее распространенные следующие форматы файлов.

- bmp (bit map) — битовая карта. Формат распространен в Windows (Paint). В этом формате файл состоит из двух частей. 1- заголовок, в котором указывается разрешение изображения и количество бит, которыми кодируется цвет пикселя. 2- область данных (битовая карта) в которой хранятся в виде последовательности бит цвета пикселов изображений.

-pcx. Формат pcx использует простейший способ сжатия изображений, позволяющий выполнять быструю перезапись изображения из файла в видеопамять и обратно. Данный формат использует в своей работе многие графические редакторы, в частности Paint. Вместе с форматом tif формат pcx является одним из наиболее распространённых форматов, которые используют сканеры.

В заголовке файлов этого формата указывается информация о версии формата pcx, информация о том — используется сжатие информации или нет, информация о цветах изображения, размерах изображения, разрешения сканера, разрешение дисплея. Для сжатия в файле изображения формата pcx используется метод группового кодирования, в котором группа повторяющихся байт заменяется двумя байтовыми: байтом повторителем и повторяющимся байтом.

Байт повторитель имеет уникальный код и содержит в себе число повторяющихся байт.

- Формат GIF, при достаточно простой структуре файла и наличии наибольшего числа атрибутов изображения используют более эффективный, чем в pcx алгоритм сжатия. Этот формат в настоящее время используется при размещении графической информации в гипертекстовых документах Internet.

- TIF (Tiff – Tag Image File Format). Основной областью применения данного формата является настольная издательская деятельность и связанные с ней приложения. Этот формат имеет множество атрибутов, позволяющих точно описать сложение изображения. Часто этот формат используется, для хранения отсканированных изображений. Форматы GIF и TIF в основном используют lzw сжатие. Название этого алгоритма произошло от фамилии его разработчиков Lampel, Ziv и Welch.

Jpg – формат, который использует специальный алгоритм сжатия изображения, позволяющее сжать изображение до требуемого размера и качества. При этом качество изображения теряется. Формат распространен для размещения графической информации в гипертекстовых документах Internet.

Для хранения векторных изображений в графических редакторах используются свои форматы.

Пользовательский интерфейс большинства графических редакторов организуется следующим образом. С левой стороны экрана располагается набор пиктограмм (условных рисунков) с изображением инструментов, которыми можно пользоваться в

процессе редактирования изображений. В нижней части экрана – палитра, из которой художник выбирает краски требуемого цвета. Оставшаяся часть экрана представляет собой пустой холст (рабочее поле). Над рабочим полем находится меню, позволяющее изменять режимы работы ГР. На левом краю палитры выводится квадрат, окрашенный в фоновый цвет. В нем помещаются еще два квадрата, верхний из которых окрашен в первый рабочий цвет, а нижний – во второй рабочий цвет. В левом нижнем углу экрана выводится калибровочная шкала, которая позволяет устанавливать ширину рабочего инструмента (кисти, резинки и т. Д.). Рассмотрим подробнее панель инструментов, инструменты, а также режимы работы графических редакторов.

Графические редакторы имеют набор инструментов для создания или рисования простейших графических объектов: прямой линии, кривой, прямоугольника, эллипса, многоугольника и т. д. (рис.4). После выбора объекта на панели инструментов его можно нарисовать в любом месте окна редактора.

Рисунок 4

Графические редакторы предоставляют пользователю различные инструменты для работы с объектами.

Выделяющие инструменты. В графических редакторах над элементами изображения возможны различные операции: копирование, перемещение, удаление, поворот, изменение размеров и т. д. Чтобы выполнить какую-либо операцию над объектом, его сначала необходимо выделить. Для выделения объектов в растровом графическом редакторе обычно имеются два инструмента: выделение прямоугольной области и выделение произвольной области. Процедура выделения аналогична процедуре рисования. Выделение объектов в векторном редакторе осуществляется с помощью инструмента выделение объекта (на панели инструментов изображается стрелкой). Для выделения объекта достаточно выбрать инструмент выделения и щелкнуть по любому объекту на рисунке.

Операции над выделенным фрагментом. Работа с фрагментами и буфером позволяет переносить фрагмент рисунка на другое место, создавать несколько копий фрагмента или передавать его в другое приложение. Выделенную область можно перетащить на другое место. Для этого нажимают левую кнопку на области, затем, не отпуская ее, перетаскивают мышь на другое место. Также можно поместить фрагмент в файл. Над фрагментом рисунка можно производить и другие операции – изменять размеры, растягивать, поворачивать, наклонять и отражать.

Инструменты редактирования рисунка позволяют вносить в рисунок изменения: стирать его части, изменять цвета и т. Д. Для стирания изображения в растровых графических редакторах используется инструмент Ластик, который убирает фрагменты изображения (пиксели), при этом размер Ластика можно менять.

В векторных редакторах редактирование изображения возможно только путем удаления объектов, входящих в изображение, целиком. Для этого сначала необходимо выделить объект, а затем выполнить операцию Вырезать.

Операцию изменения цвета можно осуществить с помощью меню Палитра, содержащего набор цветов, используемых при создании или рисовании объектов (рис.5).

Рисунок 5.

Текстовые инструменты позволяют добавлять рисунок текст и форматировать его.

Изменение шрифта текста на рисунке. Для набора текста можно использовать различные шрифты. Шрифт представляет собой набор букв, цифр, символов и знаков пунктуации определенного внешнего вида.

Характеристики шрифта – название (Times New Roman, Arial, Courier New и др.), размер и начертание (обычное, полужирное, курсив, подчеркнутый). В растровых редакторах инструментом Надпись (буква А на панели инструментов) создаются текстовые области на рисунках. Установив курсор в любом месте текстовой области, можно ввести текст. Форматирование текста производится с помощью панели *Атрибуты текста*. В векторных редакторах тоже можно создавать текстовые области для ввода и форматирования текста. Кроме того, надписи к рисункам вводятся посредством так называемых выносок различных форм. Масштабирующие инструменты в растровых графических редакторах дают возможность увеличивать или уменьшать масштаб представления объекта на экране, не влияя при этом на его реальные размеры. В увеличенном масштабе можно работать с отдельными пикселями, составляющими изображение рисунка. Обычно такой инструмент называется *Лупа*.

В векторных графических редакторах легко изменять реальные размеры объекта с помощью мыши. Режимы Графических Редакторов определяют возможные действия художника, а также команды, которые художник может отдавать редактору в данном режиме.

1. Режим работы с рисунком (рисование). В этом режиме на рабочем поле находится изображение инструмента. Художник наносит рисунок, редактирует его, манипулирует его фрагментами.
2. Режим выбора и настройки инструмента. Курсор-указатель находится в поле экрана с изображениями инструментов (меню инструментов). Кроме того, с помощью меню можно настроить инструмент на определенный тип и ширину линии, орнамент закраски.
3. Режим выбора рабочих цветов. Курсор находится в поле экрана с изображением цветовой палитры. В этом режиме можно установить цвет фона, цвет рисунка. Некоторые Графические редакторы дают возможность пользователю изменять палитру.
4. Режим работы с внешними устройствами. В этом режиме можно выполнять команды записи рисунка на диск, считывания рисунка с диска, вывода рисунка на печать. Графические редакторы на профессиональных ПК могут работать со сканером, используя его для ввода изображения с репродукций.

Обработку графической информации выхода сканера производят программы PhotoShop, PhtoWorks, PhotoPlus. Они преобразуют информацию в графические файлы формата jpg, gif. При использовании графического редактора PhotoShop при сканировании графических изображений, необходимо включить сканер и вложить в него картинку, фотографию и т.д., запустить Photo Shop ,взять в меню Файл пункт Получить, TWAIN_32. Будет запущен сканер. В режиме Preview надо произвести сканирование с низким разрешением и выбрать с помощью установления рамки область сканирования, а затем в режиме Scan произвести сканирование выбранной области с высоким разрешением (>=300 dpi).

Затем этого закрывают программу управления сканером, и изображение считывается программой Photo Shop. Далее указанное изображение обрабатывают в Photo Shop ,например, устанавливают размер изображения в пикселях, меняют яркость, контрастность, устраняют мелкие дефекты изображения, так, чтобы получить желаемое качество изображения. После этого его сохраняют на диске в виде файла с расширением jpg, чтобы он имел наименьший размер и занимал меньше места. Для обработки текстовой информации, её распознавания и преобразования её в текстовый файл используется программа Fine Reader. Полученный в этой программе текст копируют в буфер обмена и затем сохраняют в редакторе Word в виде doc файла.

Основные функции графического редактора.

Работа в графическом редакторе относится к технологии обработки графики. Для некоторого обобщённого графического редактора характерно выполнение следующих функций:

1. Создание рисунка

а) В режиме ручной прорисовки;

б) С использованием панели инструментов (штампов, примитивов).

2. Манипулирование рисунком

а) Выделение фрагментов рисунка;

б) Проработка мелких деталей рисунка (увеличение фрагментов картины);

в) Копирование фрагмента рисунка на новое место экрана (а также возможность вырезать, склеивать, удалять фрагменты изображения);

г) Закраска отдельных частей рисунка ровным слоем или узором, возможность применять для рисования произвольные «краски», «кисти» и «напыление».

Д) Масштабирование изображения;

е) Перемещение изображения;

ж) Поворот изображения;

3. Ввод в изображение текста

а) Выбор шрифта;

б) Выбор символов (курсив, подчёркивание, оттенение);

4. Работа с цветами

а) Создание своей палитры цветов;

б) Создание своего узора (штампа) для закраски;

5. Работа с внешними устройствами (диски, принтер, сканер и др.)

а) Запись рисунка на диск (дискету) в виде файла стандартного формата (pcx, bmp, tif, gif, jpg, png и др.);

б) Чтение файла с диска (дискеты);

в) Печать рисунка;

г) Сканирование рисунка.

В качестве «кисти» чаще всего используется мышь, реже курсор при управлении клавиатурой. Панель инструментов используется для рисования

прямых и кривых линий, окружностей (овалов, эллипсов), прямоугольников (квадратов).

К типовым инструментариям графических редакторов относятся:

 Google SketchUp 2016 2016 16.0.19911 Легкая в освоении программа для 3D моделирования.

 GIMP 2.8.18 Современный бесплатный графический редактор со всеми необходимыми для профессионального пользователя функциями.

 Picasa 3.9.141 Build 255 Программа для просмотра фотографий от Google с возможностью создания альбомов с последующей их загрузкой в интернет.

 PhotoScape 3.7 PhotoScape - это отличный, многофункциональный и бесплатный графический редактор, с помощью которого можно просматривать и обрабатывать картинки и фотографии.

 Paint.NET 4.10 Бесплатная программа для редактирования изображений и фотографий, позиционирующаяся как замена стандартного MS Paint.

 Paint XP 1.5 MS Paint с интерфейсом Windows XP и Windows 98. Два отдельных приложения.

 RAWTherapee 4.2.1148 Мощный бесплатный инструмент для качественной и эффективной обработки цифровых фотографий в RAW формате.

 Inkscape 0.91 Inkscape - функциональное приложение для редактирования векторной графики.

 Pencil2D 0.5.4b Программа для создания векторных и растровых изображений и 2D анимации.

 Pixia 6.0.2s Очень популярный графический редактор, предназначенный для рисования и ретуширования фотографий.

 Perfect365 1.8.0.3 Программа для нанесения макияжа на фото.

 MyPaint 1.2.0 Программа для рисования. Имеет все необходимые инструменты для создания художественных произведений.

 Скан Корректор А4 2.01 Скан Корректор А4 - приложение для сканирования и последующей корректировки полученных изображений.

 Life Photo Maker 1.60 Простая программа для редактирования изображений.

 Fotor 3.0 Современный мультиплатформенный графический редактор, с необычным интерфейсом.

 Pixel Art 11.0.1 Программа для создания по-пиксельных рисунков.

 Wings 3D 1.5.4 Используйте данное программное обеспечение для создания и конфигурации 3D моделей.

 ViewNX 2.10.3 Программа, предназначенная для просмотра и редактирования изображений и цифровых фотографий.

 Beauty Guide Lite 2.2.7 Приложение для коррекции косметических дефектов и накладывания макияжа непосредственно на фотографию.

 PhotoME 0.8 Beta 2 Программа для просмотра и редактирования метаданных файлов изображений.

 PIXresizer 2.0.8 Программа для удобного и быстрого изменения размера изображения.

 PhotoEditor 1.0 Маленький, но очень функциональный графический редактор.

 VicMan Photo Editor 8.0 VicMan Photo Editor – это программа, представляющая собой очень удобный графический редактор и являющаяся аналогом знаменитого Photoshop, вмещающая в себе множество разнообразных функций.

 StereoPhoto Maker 5.10 Специализированный графический редактор, позволяющий из обычной фотографии создать стереофото (включая, анаглиф).

 Яндекс.Краски 1.1 Программа для создания красивых композиций и коллажей.

 Picture Cutout Guide Lite 3.2.10 Приложение для отделения изображений от окружающего их фона с возможностями последующего редактирования.

 Fotobook 3.1.6 Средство создания альбомов под распечатку из своих собственных фотографий.

 WinTopo 1.753 WinTopo - удобный, продуманный и довольно быстрый графический редактор, рассчитанный на все категории пользователей.

 Adobe Photoshop CS3 Extended 10.0 Extended-версия одного из самых популярных графических редакторов на планете.

 Falco GIF Animator 4.3 Программа для создания GIF анимаций.

 Photo Art Studio 3.45 Программа для создания поздравительных открыток и коллажей.

 Photobie 7.2.1 Компактный и бесплатный графический редактор с неплохим набором возможностей и поддержкой фильтров и плагинов Фотошопа.

 BImageStudio 1.2.1 Небольшой бесплатный программный инструмент, который был создан для быстрого редактирования одного или нескольких изображений.

 Altarsoft Photo Editor 1.51 Altarsoft Photo Editor - приложение, предназначенное для обработки и редактирования изображений.

 Image Analyzer 1.37 Небольшой, но функциональный пакет для редактирования и коррекции изображений.

 ImageJ 1.50i Средство для массового анализа и обработки графических файлов.

 ZModeler 3.1.3 Build 1137 Программа для создания 3D моделей для игр.

 Pixlr-o-matic 2.1 Редактор фотографий, позволяющий накладывать красивые эффекты.

 yEd 3.16 Графический редактор, ориентированный на создание диаграмм, графиков и простых схем.

 RAW Border 1.0 RAW Border - это программа, которая предназначена для оформления любых графических файлов, а чаще всего - фотографий, в рамку. Рамки можно выбирать как из уже существующих в программе, так и из дополнительно загруженных с сайта разработчика.

 APM Graph Lite APM Graph Lite – это программа для работы с конструкторской документацией, помогающая выполнять графическую часть работы.

 MeshLab 1.3.3 Программа для редактирования неструктурированных трехмерных моделей с открытым исходным кодом

 Gimpshop 2.2.8 Fix 1 Графический редактор с мощными функциями, представленными в дружественном виде.

 FotoMix 9.2.7 Программа для редактирования фотографий, применения всевозможных эффектов и базового фотомонтажа.

 Alivecolors Freeware 1.1.1 Alivecolors Freeware - приложение для цветокоррекции фотографий.

Порядок выполнения работы.

1. Изучите теоретический материал.
2. Выполните в графических редакторах PAINT и Photoshop редактирование фотографии (цвет, тень, объем, котрасность, масштаб).
3. Нарисуйте схему обслуживания пациента в поликлинике средствами Microsoft Draw (или Corel Draw).
4. Составьте презентацию, поясняющую работу одной из физиологических систем или органов человека
5. Оформите отчет, включающий описание выполненных действий и скрин-шоты результатов, ответы на контрольные вопросы (не менее 3).

Контрольные вопросы:

1. Что такое графический редактор?
2. Какие виды графических редакторов Вы знаете?
3. Что собой представляет растровый графический редактор, его предназначение?
4. Что собой представляет векторный графический редактор, его предназначение?
5. Приведите примеры графических редакторов, которые относятся к растровым, а какие к векторным?
6. Перечислите форматы файлов для хранения графических изображений?
7. Для чего предназначена Панель инструментов в графическом редакторе?
8. С помощью чего осуществляется выделение объекта?
9. Какие операции можно производить над выделенным объектом?
10. Какие инструменты графического редактора Вы знаете?
11. Перечислите режимы работы графического редактора?
12. Какие команды входят в систему команд графических редакторов?
13. Перечислите основные функции графического редактора?

Информационные источники:

1. Алексеев А.Г., Евсеев Г.А., Симонович С.В. Специальная информатика /URL: http://www.twirpx.com/file/423283/
2. Графические редакторы. Презентация. /URL: http://www.myshared.ru/slide/365021/
3. Работа в CorelDRAW X3 /URL: http://www.intuit.ru/studies/courses /2311/611/info
4.Самоучители по графическим редакторам. /URL: http://nnm.me/blogs/Dealing24/ amouchitel_po_graficheskim_redaktoram/
5. Самоучитель компьютерной графики и звука /URL: http://softsearch.ru/books/0-304-read.shtml
6. Обучающие видиокурсы /URL: http://grinikkos.com/view_post.php?id=60

6. Математические модели. Информационная модель лечебно-диагностического процесса

Цель работы: ознакомление с основами математического моделирования, синтеза концептуальных и информационно-аналитических моделей лечебно-диагностического процесса.

Краткие теоретические сведения.

Диагностические задачи включают распознавание текущего состояния организма пациента, постановку развернутого нозологического диагноза, оценку тяжести состояния больного. Кроме того, в процессе наблюдения за больным врач решает задачи оценки динамики состояния пациента и прогнозирования развития патологического процесса, включая возможность и характер осложнений, исход заболевания. Знания модели построения лечебно-диагностического процесса и процесса работы врача необходимы как при дальнейшем обучении на клинических кафедрах, так и при работе в больнице.

Порядок выполнения работы.

Оформите отчет, включающий результаты выполнения работы и краткие ответы не менее чем на пять контрольных вопросов.

Задание №1. Составление схемы модели лечебно-диагностического процесса

АЛГОРИТМ ВЫПОЛНЕНИЯ

На основе лекционного материала и материала учебника составьте в рабочих тетрадях схему этапов диагностического процесса при постановке диагноза в лечебном учреждении с указанием сроков нахождения в лечебном учреждении.

Задание №2. Составление схемы функции медицинского персонала в зависимости от должности.

АЛГОРИТМ ВЫПОЛНЕНИЯ

На основе лекционного материала и материала учебника составьте в рабочих тетрадях схему функций медицинского персонала высшего и среднего звена в течении всего периода времени нахождения пациента в стационаре лечебного учреждения.

	Врач	Медицинская сестра
Приемное отделение		
Лечебное отделение		
Диагностическое отделение		

Задание №3. Составление схемы этапов процесса математического моделирования. **АЛГОРИТМ ВЫПОЛНЕНИЯ**

На основе лекционного материала и материала учебника составьте в рабочих тетрадях схему этапов процесса математического моделирования с указанием их основной характеристики.

Контрольные вопросы.

1. Дайте определение медицинскому технологическому процессу.
2. Кто является объектом и субъектом управления в медицинском технологическом процессе?
3. Назовите этапы управления состоянием пациента в лечебно-диагностическом процессе.
4. Дайте определение информатизации медицинского технологического процесса.
5. Какие элементы деятельности врача подлежат информатизации?
6. Опишите уровни информатизации врачебной деятельности.
7. Что представляют собой модель и моделирование?
8. Дайте характеристику информационной и математической моделям.
9. Назовите этапы процесса математического моделирования.
10. Какие модели используются в медицине?
11. Какие модели и с какой целью применяются в медицинской информатике?
12. Что такое концептуальная модель?
13. Что такое информационно-аналитическая модель?
14. Что такое кибернетическая модель?
15. Как проверить качество диагностической модели?
16. Для чего предназначены системы поддержки лечебно-диагностического процесса?
17. Какие технологии относятся к медицинским информациионым?
18. В чем сущность (и как представляется) модель дифференциальной диагностики?
19. Дайте определение моделям: математическая, информационная, энергетическая, компартментальная, физическая, химическая, биологическая, механическая, гидравлическая, аналоговая, дискретная, символическая, семантическая, интеллектуальная.

7. ВЕРИФИКАЦИЯ, ИСПЫТАНИЯ И ТРАССИРОВКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Цель работы: овладение навыками тестирования программного и алгоритмического обеспечений модулей автоматизированных рабочих мест в здравоохранении.

Краткие теоретические сведения.

В различных сферах деятельности человека под верификацией подразумеваются понятия:

- проверка, подтверждение, способ подтверждения с помощью доказательств каких-либо теоретических положений, алгоритмов, программ и процедур путём их сопоставления с опытными (эталонными или эмпирическими) данными, алгоритмами и программами. Принцип верификации был выдвинут Венским кружком;
- подтверждение соответствия конечного продукта предопределённым эталонным требованиям;
- методика распознавания лжи (укрывательства, искажения).

Формальная верификация - доказательство с помощью формальных методов правильности или неправильности программы (системы) в соответствии с формальным описанием свойств программы (системы).

К методам верификации относятся:

- Метод аксиоматической семантики Хоара
- метод индуктивных утверждений Флойда
- доказательное программирование (proofing programming)
- автоматическое доказательство теорем (Theorem proving)
-проверка моделей (Model checking)
-символьное выполнение (Symbolic execution)
- абстрактная интерпретация (Abstract Interpretation)

Верификация или эмпирическое подтверждение является основным критерием научности знания. В ГОСТ Р ИСО 9000-2008 (аналоге ISO 9000:2000) «верификация» определена следующим образом: «Подтверждение на основе представления объективных свидетельств того, что установленные требования были выполнены».

В медицинской информатике различают следующие виды верификации:

- верификация медицинских изделий;
- верификация методов диагностики;
- верификация программного обеспечения (включая универсальные инструментарии);
- верификация алгоритмического обеспечения;
- статистическое тестирование программного обеспечения.

Классическая верификация в указанных случаях основывается на идеологии доказательной медицины, сущность которой заключается в применении статистических критериев, отражающих адекватность или приемлемость

верифицируемого продукта требуемым критериям качества, полученных на репрезентативном экспериментальном (наблюдаемом или имитируемом) материале.

Центр доказательной медицины в Оксфорде, предлагает следующие критерии достоверности медицинской информации:

-Высокая достоверность – информация основана на результатах нескольких независимых клинических испытаний с совпадением результатов, обобщенных в систематических обзорах.

-Умеренная достоверность – информация основана на результатах по меньшей мере нескольких независимых, близких по целям клинических испытаний.

-Ограниченная достоверность – информация основана на результатах одного клинического испытания.

-Строгие научные доказательства отсутствуют (клинические испытания не проводились) – некое утверждение основано на мнении экспертов.

Представленные классификации могут быть последовательными этапами в формировании массива данных с определенной степенью достоверности.

На первом этапе оценке подвергаются данные отдельных клинических исследований, а на втором этапе анализируются научные обзоры и данные мета-анализа.

Далее - исследователем должно быть четко определено отношение к информации с низкой степенью достоверности. Она может исключаться или не исключаться из исследования, но в любом случае это должно быть особо оговорено.

На последнем этапе верификации осуществляется анализ содержания отобранных статей, чтобы отсеять те, которые не соответствуют сформулированной в начале исследования гипотезе.

На каждом из этих этапов имеется ряд факторов, оказывающих влияние на ход верификации, и в конечном итоге способных вызвать искажение результатов. Например, на этапе поиска информации могут быть найдены не все значимые источники по интересующей теме из-за плохого индексирования или нежелания фирм-спонсоров публиковать отрицательные результаты клинических испытаний. Сложности корректной трактовки, связанные с различиями в подходах, стандартах и способах оценки также могут оказывать влияние на результат.

Ограниченность метода верификации особенно заметна при попытке анализа воздействий внешней среды на здоровье человека, где многие факторы находятся в тесной взаимосвязи друг с другом, а степень риска других факторов не установлена. В данном случае результаты верификации могут быть представлены в виде «есть эффект» - «эффект отсутствует» на различных уровнях воздействия в рамках сформулированной теории о факторах риска.

Формирование базы эмпирических данных в контексте социально-гигиенического мониторинга является основой для процесса принятия решений в управлении рисками. Основной же проблемой при верификации степени

риска факторов внешней среды будет поиск и оценка достоверности данных с позиций медицинской статистики.

Таким образом, принцип верификации в доказательной медицине является мощным инструментом установления научной значимости выдвигаемых гипотез.

Процесс верификации требований к ПО является неотъемлемой частью всего процесса разработки. Верификация тесно связана проектированием, разработкой и сопровождением программной системы.

Верификация, в данном случае, - это процесс удостоверения, что программы и их компоненты выполняют предъявляемые им требования. Целью верификации является удостоверение в том, что программное обеспечение соответствует предъявляемым требованиям. Параллельно с этим фиксируются новые дефекты, добавленные в процессе разработки. Процесс верификации является составляющей частью более общего процесс обеспечения условленного уровня качества разрабатываемой системы.

Верификация - это контролируемый извне процесс, демонстрирующий наличие в системе багов и условия их проявления, в то время как отладка — это практически стихийный процесс локализации и устранения ошибок в системе.

В процессе тестирования обнаруживаются ошибки кода разрабатываемой программной системы. В результате этого вида деятельности устанавливается, соответствует ли текущая реализация системы предъявляемым к ней требованиям, отрабатывает ли система корректно или существую несоответствия в процессе управляемого исполнения кода системы.

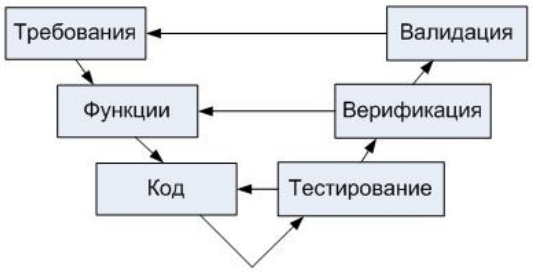


Рисунок 1 - Верификация, валидация и тестирование в связи с объектами анализа

Тестирование — составная часть деятельности по верификации, как и инспекции системы, анализ результатов тестирования, составление, анализ и отчетов о проблемах. Верификация позволяет гарантировать, что программная система реализована без непредусмотренной функциональности, соответствует предъявляемым требованиям, спецификациям и стандартам. Валидация предназначена для доказательства того, что в результате применения разработанной программной системы достижимы определенные перед началом разработки цели (бизнес-цели). Процесс валидации так же управляем.

На рисунке 1 показана схема взаимосвязи деятельности по верификации, тестированию и валидации и анализируемых в процессе объектов. По этой схеме очень легко запомнить, что тестируется код, верифицируется функциональность системы, и валидируются высокоуровневые требования.
Т.е. эти три вида деятельности имею различные объекты анализа, тем и отличаются. Даже простые системы ПО обладают высокой степенью сложности, поэтому при их разработке приходится применять весь арсенал технических и инженерных методов. Следовательно, инженерия программного обеспечения – это инженерная дисциплина, где специалисты используют теорию и методы компьютерных наук для успешного решения разного рода нестандартных задач.
При создании программного продукта перед инженером встает множество вопросов различного рода, таких как, например, требования к ПО, модели систем, спецификации ПО, надежность создаваемого продукта, и т.д.
Несмотря на кажущуюся схожесть, термины «верификация» и «аттестация» означают разные уровни проверки корректности работы программной системы. Дабы избежать дальнейшей путаницы, четко определим эти понятия.
Верификация – это проверка соответствия программного обеспечения проектной спецификации и стандартам, технической документации, представленной технико-медицинским заданием, архитектурой или моделью предметной области. Так же целью верификации является достижение гарантии того, что верифицируемый объект (требование или программный код) реализован без непредусмотренных функций.
Аттестация программной системы – более общий процесс, целью которого является доказательство того, что в результате разработки системы мы достигли тех целей, которые планировали достичь благодаря ее использованию. Иными словами, аттестация - это проверка соответствия системы ожиданиям заказчика, поэтому она проводится после верификации.

Поскольку основной задачей верификации, как и аттестации, является контроль качества программного обеспечения, необходимо обратиться к этому понятию. Стандарт ISO 9126 предлагает учитывать три разных точки зрения при рассмотрении качества ПО: точку зрения разработчиков, которые воспринимают внутреннее качество ПО, точку зрения руководства и аттестации ПО на соответствие сформулированным к нему требованиям, в ходе которой определяется внешнее качество ПО, и точку зрения пользователей, ощущающих качество ПО при использовании.

Во всех трех случаях для описания качества используется предложенная МакКолом многоуровневая модель, состоящая из целей или факторов, атрибутов или критериев и метрик качества. Цели (факторы) позволяют на верхнем уровне определять основные характеристики, которые ПО должно иметь или уже имеет. Каждый фактор состоит из набора атрибутов (критериев), позволяющих качественно описать желаемые или полученные характеристики более детально. Каждый атрибут поддерживается набором метрик, которые

позволяют количественно оценивать наличие соответствующей характеристики. Для двух точек зрения — внешнего качества и внутреннего качества — в рамках ISO 9126 предложена модель качества, схематически представленная на Рисунке 2.

Рисунок 2. Факторы и атрибуты внешнего и внутреннего качества ПО по ISO 9126

Существуют две основные методики проверки и анализа систем в процессах верификации и аттестации: инспектирование и автоматический анализ – это статические методы, которые могут выполняться на всех этапах процесса разработки системы, а так же тестирование – это динамический метод, который выполняется если уже создана исполняемая программа, то есть на этапе реализации системы и после завершения ее реализации.

Тестирование (testing)—это процесс выполнения программы (или части программы) с намерением (или целью) найти ошибки. Отладка (debugging) не является разновидностью тестирования. Хотя слова «отладка» и «тестирование» часто используются как синонимы, под ними подразумеваются разные виды деятельности. Тестирование — деятельность, направленная на обнаружение ошибок; отладка направлена на установление точной природы известной ошибки, а затем — на исправление этой ошибки. Эти два вида деятельности связаны — результаты тестирования являются исходными данными для отладки. Отладка – сложный процесс и это обусловлено следующими причинами: от программиста требуются глубокие знания специфики управления используемыми техническими средствами, операционной системы, среды и языка программирования, реализуемых процессов, природы и специфики различных ошибок, методик отладки и соответствующих программных средств, так же сложность отладки может быть обусловлена взаимовлиянием ошибок в разных частях программы. Найденная программистом ошибка исправляется, после чего проводится повторное

тестирование, так как в процессе исправления не исключается возможность появления новых ошибок. Полное повторное тестирование занимает много времени и как следствие является дорогостоящим, поэтому система разбивается на отдельные части и повторно тестируется только та часть, а так же связанные с нею другие части, где обнаружена ошибка.

Верификация и аттестация программного обеспечения представляют собой дорогостоящую, ресурсоемкую работу, на выполнение которой расходуется до половины сметной стоимости проекта. Высокая эффективность планирования испытаний и правильная оценка трудозатрат в значительной мере содействует успеху всего процесса тестирования, в то время как неудачи на этой стадии могут привести к превышению сметной стоимости проекта и нарушению графика работ. Поэтому процесс планирования аттестации и верификации должен начаться как можно раньше. Диаграмма видов деятельности, связанных с планированием испытаний, показана на рисунке 3.

Рисунок 3. Виды деятельности, осуществляемые при составлении плана испытаний

Входными для процесса планирования будут документы, которые содержат требования к ПО, без которых невозможно провести необходимое тестирование. Выходным результатом действий исполнителей, осуществляющих планирование тестирования, является документ или набор документов, который должен быть проверен тестовой группой, группой разработчиков и персоналом, осуществляющим управление разработкой и сопровождением программ. В плане проведения испытаний указываются ресурсы, необходимые для тестирования программного продукта, и определяется, что подлежит тестированию, как должно проводиться тестирование и какие выходы или выходные результаты будут получены по итогам тестирования.

Планирование начинается с определения стратегии тестирования на концептуальном уровне, после чего добавляются уровни дальнейшей детализации, которые описывают архитектуру тестирования, условия испытаний, а также тестовые случаи до тех пор, пока не будет составлен план проведения испытаний в его окончательном виде. После того, как план проведения испытаний будет оформлен в виде документа и утвержден, процесс планирования не прекращается, поскольку возможны изменения требований, изменения в графике работ и другие виды изменений, которые влекут за собой коррекцию плана проведения испытаний.

План проведения испытаний должен поддерживаться на всем протяжении процесса разработки программного продукта как динамически развивающийся документ.

Инспектирование обычно используется на ранних этапах разработки. Все проектные решения, принятые на том или ином этапе, должны анализироваться с точки зрении их правильности и целесообразности как можно раньше, пока их можно легко пересмотреть. Поскольку возможность практической проверки подобных решений на ранних этапах разработки отсутствует, большое значение имеет их обсуждение, которое проводится в разных формах. Инспектирование способствует существенному увеличению производительности и надежности программ и с его помощью можно находить от 30% до 70% ошибок логического проектирования и кодирования.

Рассмотрим инспектирование программы, т.е. проверку исходного текста (кода). Инспекции исходного текста представляют собой набор процедур и приемов обнаружения ошибок при изучении текста группой специалистов. В эту группу входят: автор программы, проектировщик, специалист по тестированию и координатор – компетентный программист, который управляет и организует процесс инспектирования. Общая процедура инспекции предполагает следующие операции:

- участника группы заранее выдается листинг программы и спецификация на нее;
- программист рассказывает о логике работы программы и отвечает на вопросы инспекторов;
- программа анализируется про списку вопросов для выявления исторически сложившихся ошибок программирования.

Список вопросов для инспекции исходного текста зависит, как от используемого языка программирования, так и от специфики разрабатываемого ПО. В качестве примера ниже приведен список вопросов, который можно использовать при анализе правильности программ, написанных на языке Pascal.

1. *Контроль обращений к данным:* Все ли переменные инициализированы? Не превышены ли максимальные (или реальные) размеры массивов и строк? Не перепутаны ли строки со столбцами при работе с матрицами? Присутствуют ли переменные со сходными именами? Используются ли файлы? Если да, то при вводе из файла проверяется ли завершение файла? Соответствуют ли типы

записываемых и читаемых значений? Использованы ли нетипизированные переменные, открытые массивы, динамическая память? Если да, то соответствуют ли типы переменных при «наложении» формата? Не выходят ли индексы за границы массивов?

2. *Контроль вычислений:* Правильно ли записаны выражения (порядок следования операторов)? Корректно ли выполнены вычисления над неарифметическими переменными? Корректно ли выполнены вычисления с переменными различных типов (в том числе с использованием целочисленной арифметики)? Возможно ли переполнение разрядной сетки или ситуация машинного нуля? Соответствуют ли вычисления заданным требованиям точности? Присутствуют ли сравнения переменных различных типов?

3. *Контроль передачи управления*: Будут ли корректно завершены циклы? Будет ли завершена программа? Существуют ли циклы, которые не будут выполняться из-за нарушения условия входа? Корректно ли продолжатся вычисления? Существуют ли поисковые циклы? Корректно ли отрабатываются ситуации «элемент найден» и «элемент не найден»?

4. *Контроль межмодульных интерфейсов:* Соответствуют ли списки параметров и аргументов по порядку, типу, единицам измерения? Не изменяет ли подпрограмма аргументов, которые не должны изменяться? Не происходит ли нарушения области действия глобальных и локальных переменных с одинаковыми именами? Кроме непосредственного обнаружения ошибок, результаты инспекции позволяют программисту увидеть другие сделанные им ошибки, получить возможность оценить свой стиль программирования, выбор алгоритмов и методов тестирования.

Инспекция является главным и наиболее эффективным способом раннего выявления частей программы, с большей вероятностью содержащих ошибки, что позволяет при тестировании уделить внимание именно этим частям. Для проведения инспектирования участники группы должны пройти тщательную подготовку. По времени процесс инспектирования должен занимать не более двух часов и сосредоточиваться только на выявлении дефектов и несоответствий требованиям.

Для сканирования исходного текста программы и выявления в нем возможных ошибок и противоречий используются такие инструментальные программные средства как статические анализаторы программ. Целью автоматического статического анализа является привлечение внимания проверяющего к аномалиям в программе, например к переменным, которые используются без инициализации или совсем не используются, или к данным, значения которых превышают заданное, и т.п.

Статический анализ кода (англ. static code analysis) — анализ программного обеспечения, производимый (в отличие от динамического анализа) без реального выполнения исследуемых программ. В большинстве случаев анализ производится над какой-либо версией исходного кода, хотя иногда анализу подвергается какой-нибудь вид объектного кода. При статическом анализе

осуществляется проверка синтаксиса, статической семантики и стиля кодирования программ, а также некоторые другие сопутствующие анализу действия. В зависимости от используемого инструмента глубина анализа может варьироваться от определения поведения отдельных операторов до анализа, включающего весь имеющийся исходный код. Способы использования полученной в ходе анализа информации также различны — от выявления мест, возможно содержащих ошибки (утилиты типа lint), до формальных методов, позволяющих математически доказать какие-либо свойства программы (например, соответствие поведения спецификации).

Используются следующие принципы статического анализа.

Большинство компиляторов (например, GNU C Compiler) выводят на экран «предупреждения» (англ. warnings) — сообщения о том, что код, будучи синтаксически правильным, скорее всего, содержит ошибку. Это простейший статический анализ. У компилятора есть много других немаловажных характеристик — в первую очередь скорость работы и качество машинного кода, поэтому компиляторы проверяют код лишь на очевидные ошибки. Статические анализаторы предназначены для более детального исследования кода.

Типы ошибок, обнаруживаемых статическими анализаторами.

-Неопределенное поведение — неинициализированные переменные, обращение к NULL-указателям. О простейших случаях сигнализируют и компиляторы.

-Нарушение алгоритма пользования библиотекой. Например, для каждого fopen нужен fclose. И если файловая переменная теряется раньше, чем файл закрывается, анализатор может сообщить об ошибке.

-Типичные сценарии, приводящие к недокументированному поведению. Стандартная библиотека Си известна большим количеством неудачных технических решений. Некоторые функции, например, gets, в принципе небезопасны.

-Переполнение буфера — когда компьютерная программа записывает данные за пределами выделенного в памяти буфера.

-Ошибки в повторяющемся коде. Многие программы исполняют несколько раз одно и то же с разными аргументами. Обычно повторяющиеся фрагменты не пишут с нуля, а размножают и исправляют.

-Ошибки форматных строк — в функциях наподобие printf могут быть ошибки с несоответствием форматной строки реальному типу параметров.

-Прочие ошибки — многие функции из стандартных библиотек не имеют побочного эффекта, и вызов их как процедур не имеет смысла.

Статический анализ состоит из нескольких этапов.

1. *Анализ потока управления*. Идентификация и выделение циклов, их точек входа и выхода, выявление неиспользуемого кода.

2. *Анализ использования данных*. Проверка переменных в программе. На этом этапе также можно выявить условные операторы с избыточными условиями.

3. *Анализ интерфейса.* Проверка согласованности различных частей программы, правильности объявления процедур и их использования. Данный этап является лишним, если используется язык со строгим контролем типов.
4. *Анализ потоков данных.* Определяются зависимости между входными и выходными переменными. Такой анализ не позволяет выявить конкретных ошибок, он дает полный список значений, используемых в программе. Следовательно, легко обнаруживается ошибочный вывод данных.
5. *Анализ ветвей программы.* На этом этапе семантического анализа определяются все ветви программы и выделяются операторы, исполняемые в каждой ветви. Такой анализ помогает разобраться в управлении программой и позволяет проанализировать каждую ветвь отдельно.

Следует отметить, что анализ потока данных и анализ ветвей программы генерируют огромное количество информации, которая не выявляет конкретных ошибок, а представляет программу в разных аспектах. Из-за большого объема генерируемой информации эти этапы зачастую исключают из процесса анализа и используют только на ранних стадиях для обнаружения аномалий в разрабатываемой программе. Статические анализаторы полезны в тех случаях, когда используются языки программирования, подобные С, так как в них нет строгого контроля типов, и потому проверка, осуществляемая компилятором языка С, ограничена. В этом случае средствами статического анализа выявляется широкий спектр ошибок, что особенно важно при разработке критических систем.

Анализ с помощью инструментальных средств не может заменить инспектирования, так как существуют такие типы ошибок, которые не выявляются с помощью статического анализа. В частности, анализаторы могут обнаружить необъявленные переменные, но неправильное присвоение они обнаружить не смогут. В итоге статический анализ является наиболее эффективным при обнаружении ошибок в языках высокого уровня.

Метод «чистая комната»

Cleanroom Software Engineering (методология «чистой комнаты») - процесс разработки программного обеспечения, предназначенный для создания программного обеспечения с сертифицируемым уровнем надёжности. Основной принцип cleanroom состоит в том, что предупреждение дефектов лучше, чем их устранение. Название Cleanroom («чистая комната») взято из электронной промышленности — так называются помещения с высокой степенью защиты от загрязнений, позволяющие предотвратить появление дефектов в процессе производства полупроводников. Впервые процесс был применён в середине-конце 80-х годов.

Основные принципы метода.

- Разработка программного обеспечения основывается на формальных методах
- Инкрементальная реализации в рамках статистического контроля качества
- Статистическое тестирование

- Формальная верификация

Наиболее важные для заказчика системные функции проверяются наибольшее количество раз. Метод «чистой комнаты» позволяет выявить дефекты еще до исполнения программы. Эти дефекты исправляются в процессе разработки ПО. Такой метод наиболее эффективен при разработке систем с наивысшим уровнем надежности.

Верификация и аттестация критических систем

Верификация и аттестация систем, критических по обеспечению безопасности, имеет много общего с тестированием любых систем с высокими требованиями надежности. Чтобы обнаружить наибольшее количество ошибок, следует применять всестороннее тестирование, а при оценке безопасности использовать статические методы тестирования. Однако вследствие чрезвычайно низкой частоты отказов, присущих многим КС, с помощью статического тестирования не всегда удается количественно оценить безотказность, так как для этого требуется очень большое число тестов. Эти тесты лишь дают основание считать ту или иную КС безопасной.

При создании КС, важен всесторонний анализ разрабатываемой системы. Выделяют пять типов анализа системы, обязательных для КС:

1. Анализ правильности функционирования системы
2. Анализ возможности изменения и понятности системной архитектуры
3. Анализ соответствия алгоритма обработки и структуры данных определенному в спецификации поведению системы
4. Анализ согласованности программного кода, алгоритмов и структур данных.
5. Анализ адекватности тестовых сценариев системным требованиям.

Все доказательства безопасности системы строятся на следующем предположении: количество ошибок в системе, которые приводят к аварийным ситуациям, намного меньше общего числа ошибок в системе. Обеспечение безопасности должно сосредоточиться на выявлении потенциально опасных ошибок. Если оказывается, что эти ошибки не проявляются или проявляются, но не приводят к серьезным последствиям, то система считается надежной. Доказательства правильности программ были предложены в качестве методов верификации ПО более 25 лет назад. Однако эти методы в основном используются только в лабораториях. Практические проблемы построения доказательства правильности ПО настолько сложны, что некоторые организации считают использование данных методов в процессе разработки обычных систем неоправданно дорогим. Но, как отмечалось ранее, для ряда КС экономически выгодно использовать доказательства правильности системы, чем ликвидировать последствия отказов. Несмотря на то, что для большинства систем разрабатывать доказательства правильности нерентабельно, иногда возникает необходимость разработать доказательства безопасности, демонстрирующие соответствие данной программы требованиям по обеспечению безопасности. При доказательстве безопасности необязательно

доказывать соответствие программы спецификации. Необходимо только показать, что выполнение программы не приводит к сбоям с опасными последствиями.

В процессах верификации и аттестации используются две основные методики проверки и анализа систем.

1. Инспектирование ПО. Анализ и проверка различных представлений системы, например документации спецификации требований, архитектурных схем или исходного кода программ. Инспектирование выполняется на всех этапах процесса разработки программной системы. Параллельно с инспектированием может выполняться автоматический анализ исходного кода программ и соответствующих документов. Инспектирование и автоматический анализ – это статические методы верификации и аттестации, поскольку им не требуется исполняемая система.

2. Тестирование ПО. Запуск исполняемого кода с тестовыми данными и исследование выходных данных и рабочих характеристик программного продукта для проверки правильности работы системы. Тестирование – это динамический метод верификации и аттестации, так как применяется к исполняемой системе.

На рисунке 3 показано место инспектирования и тестирования в процессе разработки ПО. Стрелки указывают на те этапы процесса разработки, на которых можно применять данные методы. Согласно этой схеме, инспектирование можно выполнять на всех этапах процесса разработки системы, а тестирование – в тех случаях, когда создан прототип или исполняемая программа.

К методам инспектирования относятся: инспектирование программ, автоматический анализ исходного кода и формальная верификация. Но статические методы могут проверить только соответствие программ спецификации, с их помощью невозможно проверить правильность функционирования системы. Кроме того, статическими методами нельзя проверить такие нефункциональные характеристики, как производительность и надежность. Поэтому для оценивания нефункциональных характеристик проводится тестирование системы.

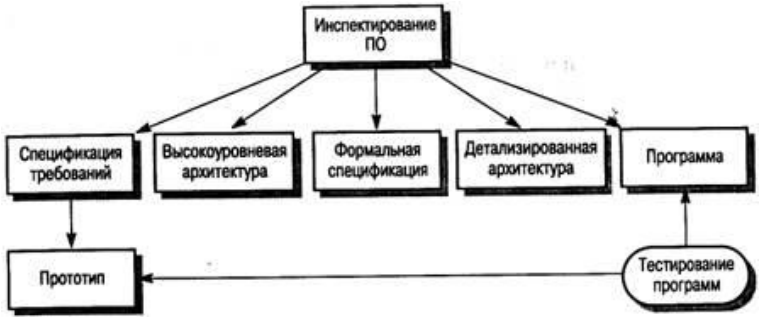


Рисунок 3. Статическая и динамическая верификация и аттестация

Несмотря на широкое применение инспектирования ПО, преобладающим методом верификации и аттестации все еще остается тестирование. Тестирование – это проверка работы программ с данными, подобными реальным, которые будут обрабатываться в процессе эксплуатации системы. Наличие в программе дефектов и несоответствий требованиям обнаруживается путем исследования выходных данных и выявления среди них аномальных. Тестирование выполняется на этапе реализации системы (для проверки соответствия системы ожиданиям разработчиков) и после завершения ее реализации.

На разных этапах процесса разработки ПО применяют *различные виды тестирования*.

1. Тестирование дефектов проводится для обнаружения несоответствий между программой и ее спецификацией, которые обусловлены ошибками или дефектами в программах. Такие тесты разрабатываются для выявления ошибок в системе, а не для имитации ее работы.
2. Статистическое тестирование оценивает производительность и надежность программ, а также работу системы в различных режимах эксплуатации. Тесты разрабатываются так, чтобы имитировать реальную работу системы с реальными входными данными. Надежность функционирования системы оценивается по количеству сбоев, отмеченных в работе программ. Производительность оценивается по результатам измерения полного времени выполнения операций и времени отклика системы при обработке тестовых данных.

Главная цель верификации и аттестации – удостовериться в том, что система "соответствует своему назначению". Соответствие программной системы своему назначению отнюдь не предполагает, что в ней совершенно не должно быть ошибок. Скорее, система должна достаточно хорошо соответствовать тем целям, для которых планировалась. Уровень необходимой достоверности соответствия зависит от назначения системы, ожиданий пользователей и условий на рынке программных продуктов.

1. *Назначение ПО*. Уровень достоверности соответствия зависит от того, насколько критическим является разрабатываемое программное обеспечение по тем или иным критериям. Например, уровень достоверности для систем, критическим по обеспечению безопасности, должен быть значительно выше аналогичного уровня достоверности для опытных образцов программных систем, разрабатываемых для демонстрации некоторых новых идей.
2. *Ожидания пользователей*. Следует с грустью отметить, что в настоящее время у большинства пользователей невысокие требования к программному обеспечению. Пользователи настолько привыкли к отказам, происходящим во время работы программ, что не удивляются этому. Они согласны терпеть сбои в

работе системы, если преимущества ее использования компенсируют недостатки. Вместе с тем с начала 1990-х годов терпимость пользователей к отказам в работе программных систем постепенно снижается. В последнее время создание ненадежных систем стало практически неприемлемым, поэтому компаниям, занимающимся разработкой программных продуктов, необходимо все больше внимания уделять верификации и аттестации программного обеспечения.

3. *Условия рынка программных продуктов.* При оценке программной системы продавец должен знать конкурирующие системы, цену, которую покупатель согласен заплатить за систему, и назначенный срок выхода этой системы на рынок. Если у компании-разработчика несколько конкурентов, необходимо определить дату выхода системы на рынок до окончания полного тестирования и отладки, иначе первыми на рынке могут оказаться конкуренты. Если покупатели не желают приобретать ПО по высокой цене, возможно, они согласны терпеть большее количество отказов в работе системы. При определении расходов на процесс верификации и аттестации необходимо учитывать все эти факторы.

Как правило, в ходе верификации и аттестации в системе обнаруживаются ошибки. Для исправления ошибок в систему вносятся изменения. Этот процесс отладки обычно интегрирован с другими процессами верификации и аттестации. Вместе с тем тестирование (или более обобщенно – верификация и аттестация) и отладка являются разными процессами, которые имеют различные цели.

В то время как, верификация и аттестация – процесс обнаружения дефектов в программной системе, то отладка – это процесс локализации дефектов (ошибок) и их исправления (рисунок 4).

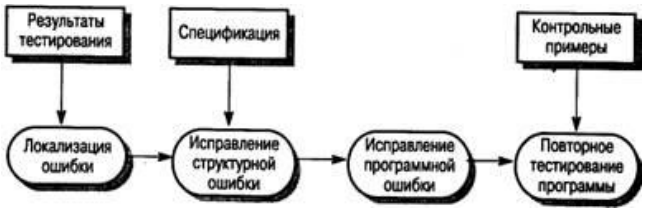


Рисунок 4.- Процесс отладки

Простых методов отладки программ не существует. Опытные отладчики обнаруживают ошибки путем сравнения шаблонов тестовых выходных данных с выходными данными тестируемых систем. Чтобы определить местоположение ошибки, необходимы знания о типах ошибок, шаблонах выходных данных, языке программирования и процессе программирования. Очень важны знания о процессе разработке ПО. Отладчикам известны наиболее распространенные ошибки программистов (например, связанные с пошаговым увеличением значения счетчика). Также учитываются ошибки, типичные для определенных

языков программирования, например связанные с использованием указателей в языке С.

Определение местонахождения ошибок в программном коде не всегда простой процесс, поскольку ошибка необязательно находится возле того места в коде программы, где произошел сбой. Чтобы локализовать ошибки, программист-отладчик разрабатывает дополнительные программные тесты, которые помогают выявить источник ошибки в программе. Может возникнуть необходимость в ручной трассировке выполнения программы.

Интерактивные средства отладки являются частью набора средств поддержки языка, интегрированных с системой компиляции программного кода. Они обеспечивают специальную среду выполнения программ, посредством которой можно получить доступ к таблице идентификаторов, а оттуда к значениям переменных. Пользователи часто контролируют выполнение программы пошаговым способом, последовательно переходя от оператора к оператору. После выполнения каждого оператора проверяются значения переменных и выявляются возможные ошибки.

Обнаруженная в программе ошибка исправляется, после чего необходимо снова проверить программу. Для этого можно еще раз выполнить инспектирование программы или повторить предыдущее тестирование. Повторное тестирование используется для того, чтобы убедиться, что сделанные в программе изменения не внесли в систему новых ошибок, поскольку на практике высокий процент "исправления ошибок" либо не завершается полностью, либо вносит новые ошибки в программу.

В принципе, во время повторного тестирования после каждого исправления необходимо еще раз запускать все тесты, однако на практике такой подход оказывается слишком дорогостоящим. Поэтому при планировании процесса тестирования определяются зависимости между частями системы и назначаются тесты для каждой части. Тогда можно трассировать программные элементы с помощью специальных контрольных примеров (контрольных данных), подобранных для этих элементов. Если результаты трассировки задокументированы, то для проверки измененного программного элемента и зависимых от него компонентов можно использовать только некоторое подмножество всего множества тестовых данных.

Планирование верификации и аттестации

Верификация и аттестация – дорогостоящий процесс. Для больших систем, например систем реального времени со сложными нефункциональными ограничениями, половина бюджета, выделенного на разработку системы, тратится на процесс верификации и аттестации. Поэтому очевидна необходимость тщательного планирования данного процесса.

Планирование верификации и аттестации, как один из этапов разработки программных систем, должно начинаться как можно раньше. На рисунке 5 показана модель разработки ПО, учитывающая процесс планирования испытаний. Здесь планирование начинается еще на этапах создания

спецификации и проектирования системы. На этой схеме также показано разделение процесса верификации и аттестации на несколько этапов, причем на каждом этапе выполняются соответствующие тесты.

Рисунок 5. Планирование испытаний в процессе разработки и тестирования

В процессе планирования верификации и аттестации необходимо определить соотношение между статическими и динамическими методами проверки системы, определить стандарты и процедуры инспектирования и тестирования ПО, утвердить технологическую карту проверок программ и составить план тестирования программ. Чему уделить больше внимания – инспектированию или тестированию, зависит от типа разрабатываемой системы и опыта организации. Чем более критична система, тем больше внимания необходимо уделить статическим методам верификации.

В плане верификации и аттестации основное внимание уделяется стандартам процесса тестирования, а не описанию конкретных тестов. Этот план предназначен не только для руководства, он в основном предназначен специалистам, занимающимся разработкой и тестированием систем. План дает возможность техническому персоналу получить полную картину испытаний системы и в этом контексте спланировать свою работу. Кроме того, план предоставляет информацию менеджерам, отвечающим за то, чтобы у группы тестирования были все необходимые аппаратные и программные средства.

План испытаний не является неизменным документом. Его следует регулярно пересматривать, так как тестирование зависит от процесса реализации системы. Например, если реализация какой-либо части системы не завершена, то невозможно провести тестирование сборки системы. Поэтому план необходимо периодически пересматривать, чтобы сотрудников, занятых тестированием, можно было использовать на других работах.

Системное тестирование программ требует разработки огромного количества тестов, их выполнения и проверки. Это значит, что данный процесс достаточно трудоемкий и дорогостоящий. Каждый тест позволяет обнаруживать одну, а в лучшем случае несколько ошибок в программе. Причина такого положения заключается в том, что сбои в работе, происходящие из-за ошибок в системе, часто приводят к разрушению данных. Поэтому трудно сказать, какое количество ошибок "ответственно" за сбой в системе.

Инспектирование программ не требует их исполнения, поэтому данный метод можно использовать до завершения полной реализации программ. Во время инспектирования проверяется исходное представление системы. Это может быть модель системы, спецификация или программа, написанная на языке высокого уровня. Для обнаружения ошибок используется знание разрабатываемой системы и семантика ее исходного представления. Каждую ошибку можно рассматривать отдельно, не обращая внимания на то, как она влияет на поведение системы.

Доказано, что инспектирование является эффективным методом обнаружения ошибок. Также немаловажно, что инспектирование значительно дешевле экстенсивного тестирования программ. В экспериментах, сравнивалась эффективность инспектирования и тестирования. Инспектирование программного кода оказалось более эффективным и менее дорогостоящим, чем тестирование. Более 60% ошибок в программах можно обнаружить с помощью неформального исследования (инспектирования) программ. При более формальном подходе, использующем математические методы, в программе можно обнаружить более 90% всех ошибок. Такая проверка используется в процессе разработки систем методом "чистая комната". Процесс инспектирования также может оценить другие качественные характеристики систем: соответствие стандартам, переносимость и удобство сопровождения.

В системных компонентах и подсистемах выявление ошибок путем просмотра и инспектирования обычно более эффективно, чем с помощью тестирования, по двум причинам.

1. За один сеанс инспектирования можно выявить множество разнообразных программных дефектов. Недостатком тестирования является то, что обычно за один сеанс тестирования можно обнаружить только одну ошибку, поскольку ошибки могут привести к отказу системы или их эффекты могут накладываться друг на друга.
2. Инспектирование использует знания о предметной области и языке программирования. Специалист, проводящий инспектирование, должен знать типы ошибок, присущие конкретным языкам программирования и приложениям определенного типа. Поэтому в ходе анализа программ есть возможность сосредоточиться только на конкретных типах ошибок.

Инспектирование, безусловно, не может полностью заменить тестирование. Инспектирование лучше использовать как начальный процесс верификации для обнаружения большей части программных дефектов. Путем инспектирования проверяют соответствие ПО ее спецификации, однако таким способом нельзя проверить динамическое поведение системы. Более того, нерационально инспектировать законченные системы, собранные из нескольких подсистем. На этом уровне возможно только тестирование. Тестирование также необходимо для оценки надежности и производительности, проверки пользовательского интерфейса и соответствия системы требованиям заказчика.

Инспектирование и тестирование не являются конкурирующими методами верификации и аттестации. Каждому из них присущи свои преимущества и недостатки, поэтому в процессе верификации и аттестации их следует использовать совместно. Одним из наиболее эффективных методов инспектирования является применение контрольных примеров. В этом случае можно обнаружить программные дефекты и разработать более эффективные методы тестирования системы.

Целью тестирования дефектов является выявление в программной системе скрытых дефектов до того, как она будет сдана заказчику. Тестирование дефектов противоположно аттестации, в ходе которой проверяется соответствие системы своей спецификации. Во время аттестации система должна корректно работать со всеми заданными тестовыми данными. При тестировании дефектов запускается такой тест, который вызывает некорректную работу программы и, следовательно, выявляет дефект. Обратите внимание на эту важную особенность: тестирование дефектов демонстрирует наличие, а не отсутствие дефектов в программе.

Общая модель процесса тестирования дефектов показана на рисунке 6. Тестовые сценарии – это спецификации входных тестовых данных и ожидаемых выходных данных плюс описание процедуры тестирования. Тестовые данные иногда генерируются автоматически. Автоматическая генерация тестовых сценариев невозможна, поскольку результаты проведения теста не всегда можно предсказать заранее.

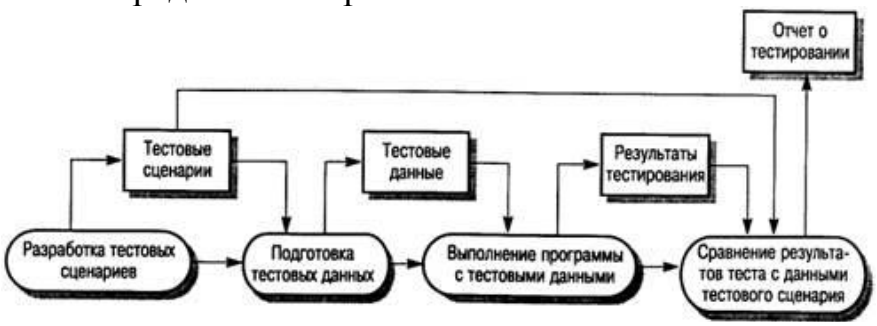


Рисунок 6. Процесс тестирования дефектов

Полное тестирование, когда проверяются все возможные последовательности выполнения программы, нереально. Поэтому тестирование должно базироваться на некотором подмножестве всевозможных тестовых сценариев. Существуют различные методики выбора этого подмножества. Например, тестовые сценарии могут предусмотреть выполнение всех операторов в программе по меньшей мере один раз. Альтернативная методика отбора тестовых сценариев базируется на опыте использования подобных систем, в этом случае тестированию подвергаются только определенные средства и функции работающей системы, например следующие.

1. Все системные функции, доступные через меню.

2. Комбинации функций, доступные через меню (например, сложное форматирование текста).
3. Если в системе предполагается ввод пользователем каких-либо входных данных, тестируются функции с правильным и неправильным вводом данных.

Из опыта тестирования (и эксплуатации) больших программных продуктов, таких, как текстовые процессоры или электронные таблицы, вытекает, что необычные комбинации функций иногда могут вызывать ошибки, но наиболее часто используемые функции всегда работают правильно.
Функциональное тестирование, или тестирование методом черного ящика базируется на том, что все тесты основываются на спецификации системы или ее компонентов. Система представляется как "черный ящик", поведение которого можно определить только посредством изучения ее входных и соответствующих выходных данных. Другое название этого метода – функциональное тестирование – связано с тем, что испытатель проверяет не реализацию ПО, а только его выполняемые функции.

Рисунок 7. Тестирование методом черного ящика

На рисунке 7 показана модель системы, тестируемая методом черного ящика. Этот метод также применим к системам, организованным в виде набора функций или объектов. Испытатель подставляет в компонент или систему входные данные и исследует соответствующие выходные данные. Если выходные данные не совпадают с предсказанными, значит, во время тестирования ПО успешно обнаружена ошибка (дефект).
. Во многих случаях выбор тестовых данных основывается на предварительном опыте испытателя. Однако дополнительно к этим эвристическим знаниям можно также использовать систематический метод выбора входных данных, обсуждаемый в следующем разделе.

Входные данные программ часто можно разбить на несколько классов. Входные данные, принадлежащие одному классу, имеют общие свойства, например это положительные числа, отрицательные числа, строки без пробелов и т.п. Обычно для всех данных из какого-либо класса поведение программы одинаково (эквивалентно). Из-за этого такие классы данных иногда называют

областями эквивалентности. Один из систематических методов обнаружения дефектов состоит в определении всех областей эквивалентности, обрабатываемых программой. Контрольные тесты разрабатываются так, чтобы входные и выходные данные лежали в пределах этих областей.
На рисунке 8 каждая область эквивалентности изображена в виде эллипса. Области эквивалентности входных данных – это множества данных, все элементы которых обрабатываются одинаково. Области эквивалентности выходных данных – это данные на выходе программы, имеющие общие свойства, которые позволяют считать их отдельным классом. Корректные и некорректные входные данные также образуют две области эквивалентности.

Рисунок 8. Области эквивалентности

После определения областей эквивалентности для каждой из них подбираются тестовые данные. При выборе тестовых данных можно руководствоваться следующим полезным правилом: для тестов выбираются данные, расположенные на границе области эквивалентности, и отдельно данные, лежащие внутри этой области. Основная причина такого выбора данных заключается в следующем. В процессе разработки системы разработчики и программисты используют для тестов типичные значения входных данных, находящиеся внутри области эквивалентности. Граничные значения часто нетипичны (например, нулевое значение обрабатывается не так, как неотрицательные числа) и потому игнорируются программистами. Хотя чаще всего ошибки в программе возникают именно при обработке подобных нетипичных значений.
Области эквивалентности определяются на основании программной спецификации или документации пользователя и опыта испытателя, выбирающего классы значений входных данных, пригодные для обнаружения дефектов. Пусть, например, в спецификации программы указано, что в программу могут вводиться от 4 до 10 целых пятизначных чисел. Области эквивалентности и возможные значения тестовых входных данных для этого примера показаны на рисунке 9.

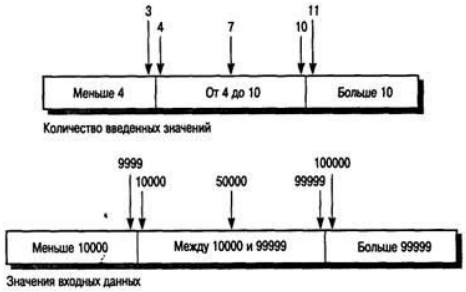


Рисунок 9. Области эквивалентности

При определении областей эквивалентности руководствуются различными правилами. Вот несколько правил выбора тестирующих последовательностей.

1. Тестирующая последовательность может состоять из одного элемента. Обычно считается, что последовательности состоят из нескольких элементов и программисты иногда закладывают такое представление в свои программы. Следовательно если ввести последовательность из одного элемента, программа может сработать неправильно.
2. Следует использовать в разных тестах различные последовательности, содержащие разное количество элементов. Это уменьшает вероятность того, что программа имеющая дефекты, случайно выдаст правильные результаты в силу некоторых случайных свойств входных данных.
3. Следует использовать тестирующие последовательности, в которых ключевой элемент является первым, средним и последним элементом последовательности. Такой метод помогает выявить проблемы на границах областей эквивалентности.

Метод структурного тестирования (рисунок 10) предполагает создание тестов на основе структуры системы и ее реализации. Такой подход иногда называют тестированием методом "белого ящика", "стеклянного ящика" или "прозрачного ящика", чтобы отличать его от тестирования методом черного ящика.

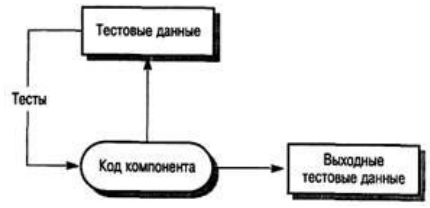


Рисунок 10. Структурное тестирование

Как правило, структурное тестирование применяется к относительно небольшим программным элементам, например к подпрограммам или методам, ассоциированным с объектами. При таком подходе испытатель анализирует программный код и для получения тестовых данных использует знания о

структуре компонента. Например, из анализа кода можно определить, сколько контрольных тестов нужно выполнить для того, чтобы в процессе тестирования все операторы выполнились по крайней мере один раз.

Знание алгоритма, используемого при реализации некоторой функции, можно применять для определения областей эквивалентности.

Метод тестирования ветвей. Это метод структурного тестирования, при котором проверяются все независимо выполняемые ветви компонента или программы. Если выполняются все независимые ветви, то и все операторы должны выполняться по крайней мере один раз. Более того, все условные операторы тестируются как с истинными, так и с ложными значениями условий. В объектно-ориентированных системах тестирование ветвей используется для тестирования методов, ассоциированных с объектами. Количество ветвей в программе обычно пропорционально ее размеру. После интеграции программных модулей в систему, методы структурного тестирования оказываются невыполнимыми. Поэтому методы тестирования ветвей, как правило, используются при тестировании отдельных программных элементов и модулей. При тестировании ветвей не проверяются все возможные комбинации ветвей программы. Не считая самых тривиальных программных компонентов без циклов, подобная полная проверка компонента оказывается нереальной, так как в программах с циклами существует бесконечное число возможных комбинаций ветвей. В программе могут быть дефекты, которые проявляются только при определенных комбинациях ветвей, даже если все операторы программы протестированы (т.е. выполнились) хотя бы один раз.

Метод тестирования ветвей основывается на анализе графа потоков управления программы. Этот граф представляет собой скелетную модель всех ветвей программы. Граф потоков управления состоит из узлов, соответствующих ветвлениям решений, и дуг, показывающих поток управления. Если в программе нет операторов безусловного перехода, то создание графа – достаточно простой процесс. При построении графа потоков все последовательные операторы (операторы присвоения, вызова процедур и ввода-вывода) можно проигнорировать. Каждое ветвление операторов условного перехода (if-then-else или case) представлено отдельной ветвью, а циклы обозначаются стрелками, концы которых замкнуты на узле с условием цикла. На рисунке 11 показаны циклы и ветвления в графе потоков управления программы бинарного поиска.

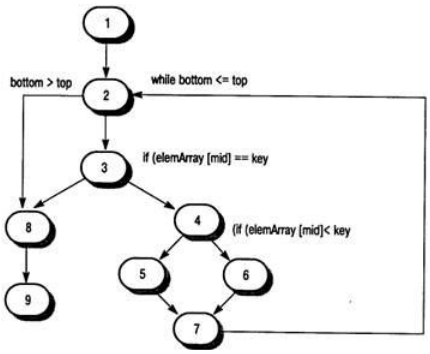


Рисунок 11 Граф потоков управления программы бинарного поиска

Цель структурного тестирования – удостовериться, что каждая независимая ветвь программы выполняется хотя бы один раз. Независимая ветвь программы – это ветвь, которая проходит по крайней мере по одной новой дуге графа потоков. В терминах программы это означает ее выполнение при новых условиях. С помощью трассировки в графе потоков управления программы бинарного поиска можно выделить следующие независимых ветвей.

1, 2, 3, 8, 9
1, 2, 3, 4, 6, 7, 2
1, 2, 3, 4, 5, 7, 2
1, 2, 3, 4, 6, 7, 2, 8, 9

Если все эти ветви выполняются, можно быть уверенным в том, что, во-первых, каждый оператор выполняется по крайней мере один раз и, во-вторых, каждая ветвь выполняется при условиях, принимающих как истинные, так и ложные значения.

Количество независимых ветвей в программе можно определить, вычислив цикломатическое число графа потоков управления программы.

Для программ, не содержащих операторов безусловного перехода, значение цикломатического числа всегда больше количества проверяемых условий. В составных условиях, содержащих более одного логического оператора, следует учитывать каждый логический оператор. Например, если в программе шесть операторов if и один цикл while, то цикломатическое число равно 8. Если одно условное выражение является составным выражением с двумя логическими операторами (объединенными операторами and или or), то цикломатическое число будет равно 10. Цикломатическое число программы бинарного поиска равно 4. После определения количества независимых ветвей в программе путем вычисления цикломатического числа разрабатываются контрольные тесты для проверки каждой ветви. Минимальное количество тестов, требующееся для проверки всех ветвей программы, равно цикломатическому числу.Проектирование контрольных тестов для программы бинарного поиска не вызывает затруднений. Однако, если программы имеют сложную структуру ветвлений, трудно предсказать, как будет выполняться какой-либо отдельный

контрольный тест. В таких случаях используется динамический анализатор программ для составления рабочего профиля программы.

Динамические анализаторы программ – это инструментальные средства, которые работают совместно с компиляторами. Во время компилирования в сгенерированный код добавляются дополнительные инструкции, подсчитывающие, сколько раз выполняется каждый оператор программы. Чтобы при выполнении отдельных контрольных тестов увидеть, какие ветви в программе выполнялись, а какие нет, распечатывается рабочий профиль программы, где видны непроверенные участки.

Тестирование сборки. После того как протестированы все отдельные программные компоненты, выполняется сборка системы, в результате чего создается частичная или полная система. Процесс интеграции системы включает сборку и тестирования полученной системы, в ходе которого выявляются проблемы, возникающие при взаимодействии компонентов. Тесты, проверяющие сборку системы, должны разрабатываться на основе системной спецификации, причем тестирование сборки следует начинать сразу после создания работоспособных версий компонентов системы.

Во время тестирования сборки возникает проблема локализации выявленных ошибок. Между компонентами системы существуют сложные взаимоотношения, и при обнаружении аномальных выходных данных бывает трудно установить источник ошибки. Чтобы облегчить локализацию ошибок, следует использовать пошаговый метод сборки и тестирования системы. Сначала следует создать минимальную конфигурацию системы и ее протестировать. Затем в минимальную конфигурацию нужно добавить новые компоненты и снова протестировать, и так далее до полной сборки системы.

В примере на рисунка 12 последовательность тестов Tl, T2 и T3 сначала выполняется в системе, состоящей из модулей A и B (минимальная конфигурация системы). Если во время тестирования обнаружены дефекты, они исправляются. Затем в систему добавляется модуль C. Тесты Tl, T2 и T3 повторяются, чтобы убедиться, что в новой системе нет никаких неожиданных взаимодействий между модулями A и B. Если в ходе тестирования появились какие-то проблемы, то, вероятно, они возникли во взаимодействиях с новым модулем C. Источник проблемы локализован, таким образом упрощается определение дефекта и его исправление. Затем система запускается с тестами T4. На последнем шаге добавляется модуль D и система тестируется еще раз выполняемыми ранее тестами, а затем новыми тестами T5.

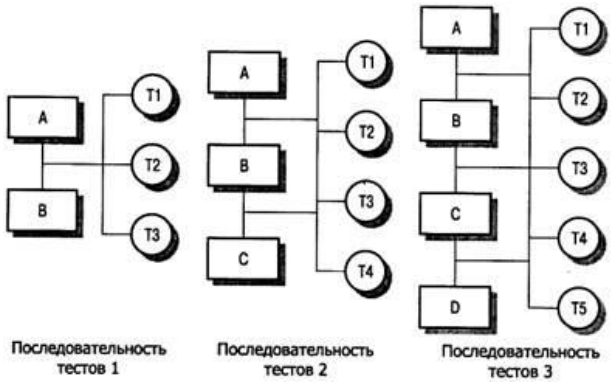


Рисунок 12.- Тестирование сборки

Конечно, на практике редко встречаются такие простые модели. Функции системы могут быть реализованы в нескольких компонентах. Тестирование новой функции, таким образом, требует интеграции сразу нескольких компонентов. В этом случае тестирование может выявить ошибки во взаимодействиях между этими компонентами и другими частями системы. Исправление ошибок может оказаться сложным, так как в данном случае ошибки влияют на целую группу компонентов, реализующих конкретную функцию. Более того, при интеграции нового компонента может измениться структура взаимосвязей между уже протестированными компонентами. Вследствие этого могут выявиться ошибки, которые не были выявлены при тестировании более простой конфигурации.

Нисходящее и восходящее тестирование. Методики нисходящего и восходящего тестирования (рисунок 13) отражают разные подходы к системной интеграции. При нисходящей интеграции компоненты высокого уровня интегрируются и тестируются еще до окончания их проектирования и реализации. При восходящей интеграции перед разработкой компонентов более высокого уровня сначала интегрируются и тестируются компоненты нижнего уровня.

Нисходящее тестирование является неотъемлемой частью процесса нисходящей разработки систем, при котором сначала разрабатываются компоненты верхнего уровня, а затем компоненты, находящиеся на нижних уровнях иерархии. Программу можно представить в виде одного абстрактного компонента с субкомпонентами, являющимися заглушками. Заглушки имеют такой же интерфейс, что и компонент, но с ограниченной функциональностью. После того как компонент верхнего уровня запрограммирован и протестирован, таким же образом реализуются и тестируются его субкомпоненты. Процесс продолжается до тех пор, пока не будут реализованы компоненты самого нижнего уровня. Затем вся система тестируется целиком.

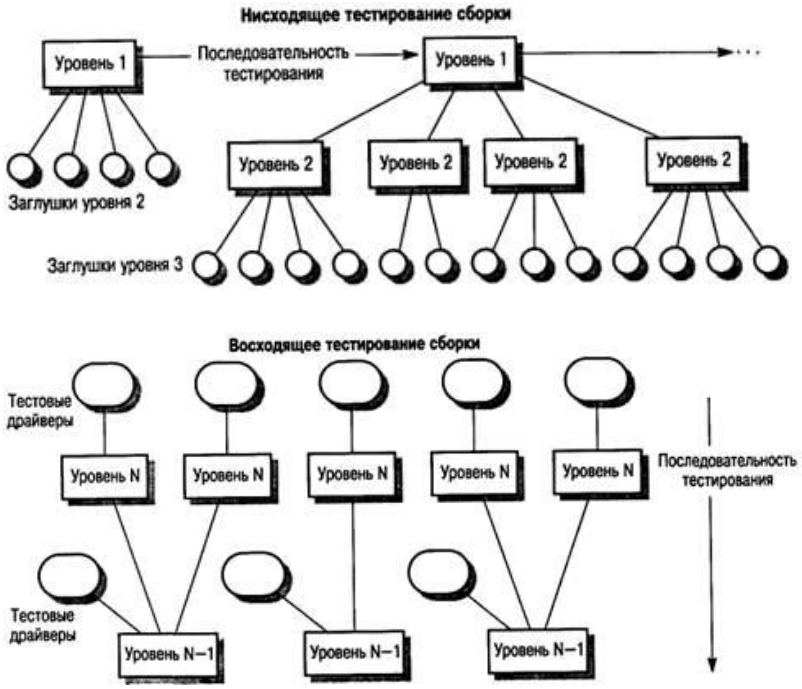


Рисунок 13. Нисходящее и восходящее тестирования сборки

При восходящем тестировании, наоборот, сначала интегрируются и тестируются модули, расположенные на более низких уровнях иерархии. Затем выполняется сборка и тестирование модулей, расположенных на верхнем уровне иерархии, и так до тех пор, пока не будет протестирован последний модуль. При таком подходе не требуется наличие законченного архитектурного проекта системы, и поэтому он может начинаться на раннем этапе процесса разработки. Обычно такой подход применяется тогда, когда в системе есть повторно используемые компоненты или модифицированные компоненты из других систем.

Нисходящее и восходящее тестирование можно сравнить по четырем направлениям.

1. *Верификация и аттестация системной архитектуры.* При нисходящем тестировании больше возможностей выявить ошибки в архитектуре системы на раннем этапе процесса разработки. Обычно это структурные ошибки, раннее выявление которых предполагает их исправление без дополнительных затрат. При восходящем тестировании структура высокого уровня не утверждается вплоть до последнего этапа разработки системы.

2. *Демонстрация системы.* При нисходящей разработке незаконченная система вполне пригодна для работы уже на ранних этапах разработки. Этот факт является важным психологическим стимулом использования нисходящей модели разработки систем, поскольку демонстрирует осуществимость управления системой. Аттестация проводится в начале процесса тестирования путем создания демонстрационной версии системы. Но если система создается

из повторно используемых компонентов, то и при восходящей разработке также можно создать ее демонстрационную версию.

3. *Реализация тестов.* Нисходящее тестирование сложно реализовать, так как необходимо моделировать программы-заглушки нижних уровней. Программы-заглушки могут быть упрощенными версиями представляемых компонентов. При восходящем тестировании для того, чтобы использовать компоненты нижних уровней, необходимо разработать тестовые драйверы, которые эмулируют окружение компонента в процессе тестирования.

4. *Наблюдение за ходом испытаний.* При нисходящем и восходящем тестировании могут возникать проблемы, связанные с наблюдениями за ходом тестирования. В большинстве систем, разрабатываемых сверху вниз, более верхние уровни системы, которые реализованы первыми, не генерируют выходные данные, однако для проверки этих уровней нужны какие-либо выходные результаты. Испытатель должен создать искусственную среду для генерации результатов теста. При восходящем тестировании также может возникнуть необходимость в создании искусственной среды (тестовых драйверов) для исследования компонентов нижних уровней.

Тестирование интерфейсов. Как правило, тестирование интерфейса выполняется в тех случаях, когда модули или подсистемы интегрируются в большие системы. Каждый модуль или подсистема имеет заданный интерфейс, который вызывается другими компонентами системы. Цель тестирования интерфейса – выявить ошибки, возникающие в системе вследствие ошибок в интерфейсах или неправильных предположений об интерфейсах.

Схема тестирования интерфейса показана на рисунке 14. Стрелки в верхней части схемы означают, что контрольные тесты применяются не к отдельным компонентам, а к подсистемам, полученным в результате комбинирования этих компонентов.

Данный тип тестирования особенно важен в объектно-ориентированном проектировании, в частности при повторном использовании объектов и классов объектов. Объекты в значительной степени определяются с помощью интерфейсов и могут повторно использоваться в различных комбинациях с разными объектами и в разных системах. Во время тестирования отдельных объектов невозможно выявить ошибки интерфейса, так как они являются скорее результатом взаимодействия между объектами, чем изолированного поведения одного объекта.

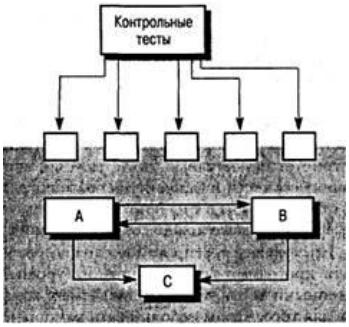


Рисунок 14. Тестирование интерфейсов

Между компонентами программы могут быть разные типы интерфейсов и соответственно разные типы ошибок интерфейсов.

1. Параметрические интерфейсы. Интерфейсы, в которых ссылки на данные и иногда функции передаются в виде параметров от одного компонента к другому.
2. Интерфейсы разделяемой памяти. Интерфейсы, в которых какой-либо блок памяти совместно используется разными подсистемами. Одна подсистема помещает данные в память, а другие подсистемы используют эти данные.
3. Процедурные интерфейсы. Интерфейсы, в которых одна подсистема инкапсулирует набор процедур, вызываемых из других подсистем. Такой тип интерфейса имеют объекты и абстрактные типы данных.
4. Интерфейсы передачи сообщений. Интерфейсы, в которых одна подсистема запрашивает сервис у другой подсистемы посредством передачи ей сообщения. Ответное сообщение содержит, результаты выполнения сервиса. Некоторые объектно-ориентированные системы имеют такой тип интерфейсов; например, так работают системы клиент/сервер.

Ошибки в интерфейсах являются наиболее распространенными типами ошибок в сложных системах и делятся на *три класса*.

Неправильное использование интерфейсов. Компонент вызывает другой компонент и совершает ошибку при использовании его интерфейса. Данный тип ошибки особенно распространен в параметрических интерфейсах; например, параметры могут иметь неправильный тип, следовать в неправильном порядке или же иметь неверное количество параметров.

Неправильное понимание интерфейсов. Вызывающий компонент, в который заложена неправильная интерпретация спецификации интерфейса вызываемого компонента, предполагает определенное поведение этого компонента. Если поведение вызываемого компонента не совпадает с ожидаемым, поведение вызывающего компонента становится непредсказуемым. Например, если программа бинарного поиска вызывается для поиска заданного элемента в неупорядоченном массиве, то в работе программы произойдет сбой.

Ошибки синхронизации. Такие ошибки встречаются в системах реального времени, где используются интерфейсы разделяемой памяти или передачи

сообщений. Подсистема – производитель данных и подсистема – потребитель данных могут работать с разной скоростью. Если при проектировании интерфейса не учитывать этот фактор, потребитель может, например, получить доступ к устаревшим данным, потому что производитель к тому моменту еще не успел обновить совместно используемые данные.

Тестирование дефектов интерфейсов сложно, поскольку некоторые ошибки могут проявиться только в необычных условиях. Например, пусть некий объект реализует очередь в виде структуры списка фиксированного размера. Вызывающий его объект при вводе очередного элемента не проверяет переполнение очереди, так как предполагает, что очередь реализована как структура неограниченного размера. Такую ситуацию можно обнаружить только во время выполнения специальных тестов: специально вызывается переполнение очереди, которое приводит к непредсказуемому поведению объекта. Другая проблема может возникнуть из-за взаимодействий между ошибками в разных программных модулях или объектах. Ошибки в одном объекте можно выявить только тогда, когда поведение другого объекта становится непредсказуемым. Например, для получения сервиса один объект вызывает другой объект и полагает, что полученный ответ правильный. Если объект неправильно понимает вычисленные значения, возвращаемое значение может быть достоверным, но неправильным. Такие ошибки можно выявить только тогда, когда оказываются неправильными дальнейшие вычисления.

К основным *правилам тестирования интерфейсов* относятся.

1. Просмотрите тестируемый код и составьте список всех вызовов, направленных к внешним компонентам. Разработайте такие наборы тестовых данных, при которых параметры, передаваемые внешним компонентам, принимают крайние значения из диапазонов их допустимых значений. Использование экстремальных значений параметров с высокой вероятностью обнаруживает несоответствия в интерфейсах.
2. Если между интерфейсами передаются указатели, всегда тестируйте интерфейс с нулевыми параметрами указателя.
3. При вызове компонента через процедурный интерфейс используйте тесты, вызывающие сбой в работе компонента. Одна из наиболее распространенных причин ошибок в интерфейсе – неправильное понимание спецификации компонентов.
4. В системах передачи сообщений используйте тесты с нагрузкой, которые рассматриваются в следующем разделе. Разрабатывайте тесты, генерирующие в несколько раз большее количество сообщений, чем будет в обычной работе системы. Эти же тесты позволяют обнаружить проблемы синхронизации.
5. При взаимодействии нескольких компонентов через разделяемую память разрабатывайте тесты, которые изменяют порядок активизации компонентов. С помощью таких тестов можно выявить сделанные программистом неявные предположения о порядке использования компонентами разделяемых данных.

Тестирование с нагрузкой. После полной интеграции системы можно оценить такие интеграционные свойства системы, как производительность и надежность. Чтобы убедиться, что система может работать с заданной нагрузкой, разрабатываются тесты для измерения производительности. Обычно планируются серии тестов с постоянным увеличением нагрузки, пока производительность системы не начнет снижаться.

Некоторые классы систем проектируются с учетом работы под определенной нагрузкой. Например, система обработки транзакций проектируется так, чтобы обрабатывать 100 транзакций в секунду; сетевая операционная система – чтобы обрабатывать информацию от 200 отдельных терминалов. При тестировании с нагрузкой выполнение тестов начинается с максимальной нагрузки, указанной в проекте системы, и продолжается до тех пор, пока не произойдет сбой в работе системы. Данный тип тестирования выполняет две функции.

1. Тестируется поведение системы во время сбоя. В процессе эксплуатации могут возникать ситуации, при которых нагрузка в системе превышает максимально допустимую. В таких ситуациях очень важно, чтобы сбой в системе не приводил к нарушению целостности данных или к потере сервисных возможностей.
2. Чтобы выявить дефекты, которые не проявляются в обычных режимах работы, система подвергается тестированию с нагрузкой. Хотя подобные дефекты не приводят к ошибкам при обычном использовании системы, на практике могут возникнуть необычные комбинации стандартных условий; именно они воспроизводятся во время тестирования с нагрузкой.

Тестирование с нагрузкой чаще всего применяется в распределенных системах. В таких системах при большой нагрузке сеть порой "забивается" данными, которыми обмениваются разные процессы. Постепенно процессы все больше замедляются, поскольку они ожидают данные запросов от других процессов.

Тестирование объектно-ориентированных систем. Существует два основных подхода к тестированию программного обеспечения – компонентное тестирование, при котором компоненты системы тестируются независимо друг от друга, и тестирование сборки, когда компоненты интегрированы в подсистемы и тестируется конечная система. Эти подходы в равной мере применимы и к объектно-ориентированным системам. Однако системы, разработанные по функциональной модели, и объектно-ориентированные системы имеют существенные отличия.

1. Объекты, как отдельные программные компоненты, представляют собой нечто большее, чем отдельные подпрограммы или функции.
2. Объекты, интегрированные в подсистемы, обычно слабо связаны между собой и поэтому сложно определить "самый верхний уровень" системы.
3. При анализе повторно используемых объектов их исходный код может быть недоступным для испытателей.

Эти отличия означают, что при проверке объектов можно применять тестирование методом белого ящика, основанное на анализе кода, а при тестировании сборки следует использовать другие подходы. Применительно к объектно-ориентированным системам можно определить четыре уровня тестирования.

1. Тестирование отдельных методов (операций), ассоциированных с объектами. Обычно методы представляют собой функции или процедуры. Поэтому здесь можно использовать тестирование методами черного и белого ящиков.
2. Тестирование отдельных классов объектов. Принцип тестирования методом черного ящика остается без изменений, однако, понятие "класса эквивалентности" необходимо расширить.
3. Тестирование кластеров объектов. Нисходящая и восходящая сборки оказываются не пригодными для создания групп связанных объектов. Поэтому здесь следует применять другие методы тестирования, например основанные на сценариях.
4. Тестирование системы. Верификация и аттестация объектно-ориентированной системы выполняется точно так же, как и для любых других типов систем.

При разработке объектно-ориентированных систем различия между уровнями интеграции менее заметны, поскольку методы и данные компонуются (интегрируются) в виде объектов и классов объектов. Тестирование классов объектов соответствует тестированию отдельных элементов. В объектно-ориентированных системах нет непосредственного эквивалента тестированию модулей. Однако считается, что группы классов, которые совместно предоставляют набор сервисов, следует тестировать вместе. Такой вид тестирования называется тестированием кластеров.

Для объектно-ориентированных систем не подходит ни восходящая, ни нисходящая интеграция системы, поскольку здесь нет строгой иерархии объектов. Поэтому создание кластеров основывается на выделении методов и сервисов, реализуемых посредством этих кластеров.

При тестировании сборки объектно-ориентированных систем используется три подхода:

1. *Тестирование сценариев и вариантов использования*. Варианты использования или сценарии описывают какой-либо один режим работы системы. Тестирование может базироваться на описании этих сценариев и кластеров объектов, реализующих данный вариант использования.
2. *Тестирование потоков.* Этот подход основывается на проверке системных откликов на ввод данных или группу входных событий. Объектно-ориентированные системы, как правило, событийно-управляемые, поэтому для них особенно подходит данный вид тестирования. При использовании этого подхода необходимо знать, как в системе проходит обработка потоков событий.

3. *Тестирование взаимодействий между объектами.* Этот промежуточный уровень тестирования сборки системы основан на определении путей "метод-сообщение", отслеживающих последовательности взаимодействий между объектами.

Тестирование сценариев часто оказывается более эффективным, чем другие методы тестирования. Сам процесс тестирования можно спланировать так, чтобы в первую очередь проверялись наиболее вероятные сценарии и только затем исключительные сценарии. Поэтому тестирование сценариев удовлетворяет основному принципу, согласно которому при тестировании больше внимания необходимо уделять наиболее часто используемым частям системы. После выбора сценариев для тестирования системы важно убедиться, что все методы каждого класса будут выполняться хотя бы один раз. Для этого можно составить технологическую карту проверок классов объектов и методов и при выборе сценария отмечать выполняемый метод.

Конечно, все комбинации методов выполнить невозможно, но по крайней мере можно удостовериться, что все методы протестированы как часть какой-либо последовательности выполняемых методов.

Тестирование – дорогой и трудоемкий этап разработки программных систем. Поэтому разработан широкий спектр инструментальных средств для поддержки процесса тестирования, которые значительно сокращают расходы на него.

На рисунке 15 показаны возможные инструментальные средства тестирования и отношения между ними. Перечислим их.

1. *Организатор тестов.* Управляет выполнением тестов. Он отслеживает тестовые данные, ожидаемые результаты и тестируемые функции программы.
2. *Генератор тестовых данных.* Генерирует тестовые данные для тестируемой программы. Он может выбирать тестовые данные из базы данных или использовать специальные шаблоны для генерации случайных данных необходимого вида.
3. *Оракул.* Генерирует ожидаемые результаты тестов. В качестве оракулов могут выступать предыдущие версии программы или исследуемого объекта. При тестировании параллельно запускаются оракул и тестируемая программа и сравниваются результаты их выполнения.
4. *Компаратор файлов.* Сравнивает результаты тестирования с результатами предыдущего тестирования и составляет отчет об обнаруженных различиях. Компараторы особенно важны при сравнении различных версий программы. Различия в результатах указывают на возможные проблемы, существующие в новой версии системы.
5. *Генератор отчетов.* Формирует отчеты по результатам проведения тестов.
6. *Динамический анализатор.* Добавляет в программу код, который подсчитывает, сколько раз выполняется каждый оператор. После запуска теста создает исполняемый профиль, в котором показано, сколько раз в программе выполняется каждый оператор.

7. *Имитатор*. Существует несколько типов имитаторов. Целевые имитаторы моделируют машину, на которой будет выполняться программа. Имитатор пользовательского интерфейса – это программа, управляемая сценариями, которая моделирует взаимодействия с интерфейсом пользователя. Имитатор ввода-вывода генерирует последовательности повторяющихся транзакций.

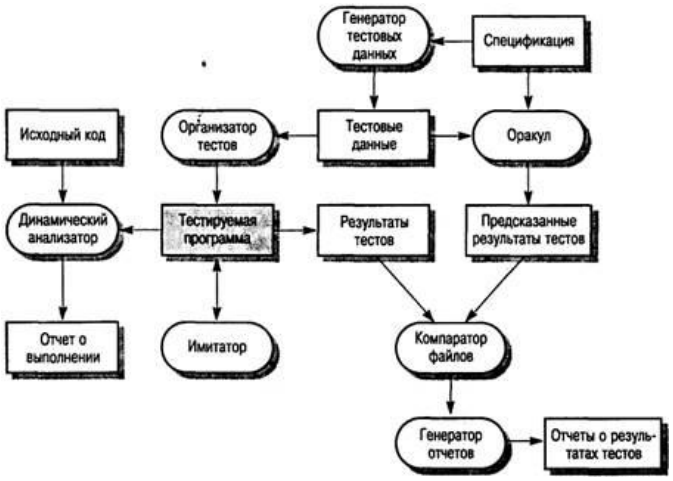


Рисунок 15. Инструментальные средства тестирования

Требования, предъявляемые к процессу тестирования больших систем, зависят от типа разрабатываемого приложения. Поэтому инструментальные средства тестирования неизменно приходится адаптировать к процессу тестирования конкретной системы.

Для создания полного комплекса инструментального средства тестирования, как правило, требуется много сил и времени. Весь набор инструментальных средств, показанных на рисунке, используется только при тестировании больших систем. Для таких систем полная стоимость тестирования может достигать 50% от всей стоимости разработки системы. Вот почему выгодно инвестировать разработку высококачественных и производительных CASE-средств тестирования.

Руководители программных проектов выполняют такую же работу, что и руководители технических проектов. Вместе с тем процесс разработки ПО существенно отличается от процессов реализации технических проектов, что порождает определенные сложности в управлении программными проектами. Приведем небольшой список этих отличий.

1. *Программный продукт нематериален.* Менеджер судостроительного проекта или проекта постройки здания видит результаты выполнения своего проекта. Если реализация проекта отстает от графика, это также видно воочию, так как часть конструкции не завершена. В противоположность этому программное обеспечение нематериально. Его нельзя увидеть или потрогать. Менеджер программного проекта не видит процесс "роста" разрабатываемого ПО. Он может полагаться только на документацию, которая фиксирует процесс разработки программного продукта.

2. *Не существует стандартных процессов разработки ПО.* На сегодняшний день не существует четкой зависимости между процессом создания ПО и типом создаваемого программного продукта. Другие технические дисциплины имеют длительную историю, процессы разработки технических изделий многократно опробованы и проверены. Процессы создания большинства технических систем хорошо изучены. Изучением же процессов создания ПО специалисты занимаются только несколько последних лет. Поэтому пока нельзя точно предсказать, на каком этапе процесса разработки ПО могут возникнуть проблемы, угрожающие всему программному проекту.
3. *Большие программные проекты - это* часто «одноразовые», *специфичные проекты*. Большие программные проекты, как правило, значительно отличаются от проектов, реализованных ранее. Поэтому, чтобы уменьшить неопределенность в планировании проекта, руководители проектов должны обладать очень большим практическим опытом. Но постоянные технологические изменения в компьютерной технике и коммуникационном оборудовании обесценивают предыдущий опыт. Знания и навыки, накопленные опытом, могут не востребоваться в новом проекте.

Одним из способов тестирования программных средств или алгоритмов является трассировка, заключающаяся в поэлементном выполнении системы программного кода или алгоритма с фиксацией и-или контролем значений переменных в него входящих. Трассировка может выполняться как автоматически (с помощью специальных средств) так и в «ручную».

Основные правила культурного программирования

1. Тщательно продумайте алгоритм решения задачи. Порой выбор лучшего алгоритма позволяет кардинально повысить скорость вычислений и упростить программу (впрочем, одновременно это достигается далеко не всегда).
2. Используйте прежде всего возможности функционального программирования .
3. Разделяйте задачу на малые части и оформляйте их в виде законченных программных модулей — прежде всего функций.
4. Не скупитесь на программные комментарии — чем их больше, тем понятнее программа и тем больше шансов, что она заинтересует пользователей и будет долго жить. Учтите, что ясность программы в большинстве случаев важнее скорости ее работы.
5. Тщательно готовьте сообщения об ошибках и диагностические сообщения, а также наименования программных модулей и описания их назначения.
6. Тщательно производите диагностику программных модулей, в том числе с самыми безумными значениями и типами параметров - хорошо спроектированный модуль должен диагностировать любые виды ошибочных ситуаций и реагировать на них адекватным образом.

7. Используйте имена переменных и констант в стиле, принятом в Mathematica, и обязательно с использованием понятных по смыслу обозначений. По мере возможности не используйте в именах зарегистрированные идентификаторы команд и функций.
8. Заменяйте циклы функциями обработки списков, например функциями суммирования и произведения. Применяйте эффективные варианты упрощенных операторов и функций.
9. В максимальной степени используйте функции ядра системы. Обращайтесь к пакетам расширений только в том случае, когда это действительно необходимо.
10. Проводите тщательное тестирование своих модулей, в том числе с выполнением их трассировки. Помните, что нет программы, которую нельзя хоть чуть-чуть, но улучшить и сократить. Однако при этом цените затраченное на это время!
11. По мере возможности используйте готовые апробированные программные модули — изобретать велосипед и делать то, что уже сделано, неразумно.
12. Обращайте особое внимание на реализацию механизма контекстов, позволяющего избежать грубых ошибок при модернизации различных объектов программ, прежде всего наборов функций.
13. Не слишком оригинальничайте! Не применяйте программные трюки и недокументированные приемы программирования. Такие программы в момент создания могут выглядеть удивительно эффектными и потрясающе оригинальными, но вполне возможно, что в следующей версии системы они перестанут работать вообще, поскольку разработчики обычно стараются исключить любые недокументированные трюки в своих программах.

Порядок выполнения работы.

1. Изучите теоретический материал.
2. Выполните верификацию (тестирование, испытание) следующих материалов:

2.1 Инспектирование программы «Домашний доктор» (загрузите программу из сети – или из кафедральной базы данных, поставьте диагноз согласно медицинским справочникам или рекомендациям медиков, поставьте диагноз по программе «Домашний доктор», сравните результаты – собственные и сокурсников);

2.2 Осуществите тестирование электронной таблицы Excel (найдите значение некоторого случайного числа – больше нуля и меньше 1 – в степени, равной количеству букв в Вашем имени, выполните обратное действие, сравните результаты с точностью до 32 знака);

2.3 Составьте тестовые примеры для программы:

```
var x, a, b: integer;
begin readln(x);
 a := 2; b := 1;
```

```
while x>0 do begin
a : = a + 1; b : = b*(x mod a); x : = x div 10;
end; writeln(a); write(b).
```

2.4 Выполните трассировку программы п.2.3
Выполните трассировку следующей программы:

```
var a, b, t, M, R: integer;
function F(x: integer) : integer; begin F := 19*(x-19)*(x-17)+17; end;
 begin
a := 20; b := 20; M := a; R := F(a); for t := a to b do
 begin if (F(t)<R) then begin M:-t; R:=F(t); end; end;
write (M);
end.
```

2.5 Выполните трассировку и верификацию по ресурсу http://learningapps.org/236204

2.6 Выполните верификацию прибора – медицинского тонометра: измерьте пульс и давление с помощью тонометра; подсчитайте пульс с помощью секундомера (или смартфона); считая последний результат действительным, оцените относительную ошибку (погрешность) прибора и, принимая ее постоянной, пересчитайте давление. В этом опыте условно считается, что второе измерение выполняется с помощью эталонного прибора.

2.7 Выполните верификацию метода определения площади фигура методом Монте - Карло (точность в зависимости от количества испытаний): начертите прямоугольник определенной площади, вокруг него начертите любую фигуру известной площади, превышающей площадь прямоугольника в два раза, случайным образом нанесите N точек внутри фигуры, подсчитайте количество вошедших точек в прямоугольник, путем соотношений оцените площадь прямоугольника и ошибку измерений. Считая N количеством испытаний, повторите опыт с несколькими значениями N и постройте график, отражающий точность измерений от количества испытаний и сделайте вывод о приемлемости данного метода.

2.8 Разработайте методику оценки артериального давления в левой руке по давлению в правой – для чего: с помощью тонометра проведите не менее 6 измерений (интервал – не менее 10 минут, давление в левой руке и правой измеряются как можно с меньшим интервалом времени). Получите различные функциональные связи – математические модели (с помощью Excel) и проведите их тестирование на сокурснике.

2.9 ЗАДАНИЕ ПОВЫШЕННОЙ СЛОЖНОСТИ. Выполните тестирование любого программного модуля средствами представленными на ресурсе http://www.loadtesting.ru/download.shtml

3. Оформите отчет, включающий результаты выполнения заданий, выводы по полученным результатам и краткие ответы на контрольные вопросы (не менее 5, вопрос №6 - обязательный)

Контрольные вопросы.

1. В чем заключается идеология доказательной медицины?
2. Что такое верификация?
3. В чем состоят отличия процессов верификации и тестирования?
4. Охарактеризуйте основные этапы планирования стратегии тестирования?
5. В чем заключается инспектирование программных средств?
6. Приведите краткую характеристику содержания библиографических источников?
7. Сформулируйте типовое множество вопросов инспекции программного кода.
8. Для чего предназначен статический анализатор программ?
9. Для чего предназначен динамический анализатор программ?
10. Охарактеризуйте основные составляющие (этапы) статического анализа программ?
11. В чем заключается метод «чистой комнаты»?
12. Каким образом осуществляется верификация и аттестация критических систем?
13. Что такое «отладка программ»?
14. В чем проявляется эффективность инспектирования программного обеспечения по сравнению с тестированием?
15. В чем заключается тестирование методом «черного ящика»?
16. Что такое «области эквивалентности ПО» (назначение, определение)?
17. В чем заключается метод структурного тестирования?
18. В чем заключается метод тестирования ветвей?
19. Как осуществляется тестирование программной сборки?
20. Поясните основные принципы нисходящего и восходящего тестирований?
21. Каким образом осуществляется тестирование интерфейсов?
22. Охарактеризуйте основные правила тестирования интерфейса?
23. Как осуществляется тестирование ПО с нагрузкой?
24. Как осуществляется тестирование объектно-ориентированных систем?
25. В чем сущность тестирования кластеров?
26. Опишите типовые инструментарии средств тестирования.
27. Как осуществляется трассировка программных продуктов?
28. Перечислите и охарактеризуйте базовые правила культуры программирования.

Информационные источники

1. Верификация и аттестация программного обеспечения /URL: http://helpiks.org/4-83891.html

2. Программа Wapt Pro для нагрузочного тестирования ПО /URL: http://www.loadtesting.ru/download.shtml
3. Проблема верификации в доказательной медицине /URL: http://www.rusmedserv.com/expo/articles/articl4.html

8. Обработка информационных массивов

Краткие теоретические сведения

А. Обработка массивов с помощью языков высокого уровня на примере языка Паскаль.

Массивом называют упорядоченный набор однотипных переменных (элементов). Каждый элемент имеет целочисленный порядковый номер, называемый индексом. Число элементов в массиве называют его размерностью. Массивы используются там, где нужно обработать сразу несколько переменных одного типа: например, оценки студентов группы или координаты точек на плоскости. Строку текста можно рассматривать как массив символов, а текст на странице как массив строк.

Массив описывается в разделе var оператором следующего вида:

var ИмяМассива: array [НИ .. ВИ] of Тип;

Здесь

НИ (нижний индекс) - целочисленный номер 1-го элемента массива; .. - оператор задания диапазона в Паскале; ВИ (верхний индекс) -- целочисленный номер последнего элемента;

Тип - любой из известных типов данных Паскаля. Каждый элемент массива будет рассматриваться как переменная соответствующего типа.

Опишем несколько массивов разного назначения.

var a: array [1..20] of integer;

Здесь описан массив с именем А, состоящий из 20 целочисленных элементов;

var x,y : array [1..10] of real;

Описаны 2 массива с именами x и y, содержащие по 10 вещественных элементов;

var t : array [0..9] of string;

Массив t состоит из 10 строк, которые занумерованы с нуля.

Размерность (число элементов) массива вычисляется как ВИ - НИ + 1.

Для обращения к отдельному элементу массива используется оператор вида ИмяМассива [Индекс].

Здесь Индекс - целочисленный номер элемента (может быть целочисленным выражением или константой). Индекс не должен быть меньше значения нижнего или больше верхнего индекса массива, иначе возникнет ошибка "Constant out of range". Отдельный элемент массива можно использовать так же, как переменную соответствующего типа, например:

A[1]:=1; x[1]:=1.5; y[1]:=x[1]+1; t[0]:='Hello';

В одномерных массивах каждый элемент имеет один номер (индекс), характеризующий его положение в массиве. В математике понятию одномерного массива из n элементов соответствует понятие вектора из n компонент: A = {Ai}, i=1, 2 ,..., n.

Как правило, ввод, обработка и вывод массива осуществляются поэлементно, с использованием цикла for.

Простейший способ ввода – это ввод массива с клавиатуры, например:

```
const n = 10;
var a: array [1..n] of real;  i:integer;
begin writeln ('Введите элементы массива');
for i:=1 to n do read (A[i]);
```

Размерность массива определена константой n, элементы вводятся по одному в цикле for - при запуске этой программы пользователю придется ввести 10 числовых значений. При решении ряда задач вводить массивы "вручную", особенно если их размерность велика, не всегда удобно. Существуют, как минимум, два альтернативных решения.

Описание массива констант удобно, если элементы массива не должны изменяться в процессе выполнения программы. Как и другие константы, массивы констант описываются в разделе const. Приведем пример такого описания:

```
const a:array [1..5] of real=( 3.5, 2, -5, 4, 11.7 );
```

Как видно из примера, элементы массива перечисляются в круглых скобках через запятую, их количество должно соответствовать его размерности.

Формирование массива из случайных значений уместно, если при решении задачи массив служит лишь для иллюстрации того или иного алгоритма, а конкретные значения элементов несущественны. Для того чтобы получить очередное случайное значение, используется стандартная функция random(N), где параметром N передается значение порядкового типа. Она вернет случайное число того же типа, что тип аргумента и лежащее в диапазоне от 0 до N-1 включительно. Например, оператор вида a[1]:=random(100); запишет в a[1] случайное число из диапазона [0,99].

Для того чтобы при каждом запуске программы цепочка случайных чисел была новой, перед первым вызовом random следует вызвать стандартную процедуру randomize;, запускающую генератор случайных чисел. Приведем пример заполнения массива из 20 элементов случайными числами, лежащими в диапазоне от -10 до 10:

```
var a:array [1..20] of integer;
 i:integer;
begin
 randomize;  for i:=1 to 20 do begin   a[i]:=random(21)-10;   write (a[i]:4);
 end;
end.
```

Еще более удобный путь - чтение элементов массива из текстового или численногоо файла. К массивам применимы все типовые алгоритмы, изученные в теме "Циклы". Приведем один пример, в котором вычисляется сумма s положительных элементов массива.

```
var b:array [1..5] of real;
```

```
s:real; i:integer;
begin
writeln ('Введите 5 элементов массива'); for i:=1 to 5 do read (b[i]);
s:=0; for i:=1 to 5 do if b[i]>0 then s:=s+b[i];
```

Вывод массива на экран также делается с помощью цикла for.

```
for i:=1 to 5 do write (b[i]:6:2);
```

Здесь 5 элементов массива b напечатаны в одну строку, каждый элемент состоит из 6 символов и имеет 2 десятичных разряда после запятой. Для вывода одного элемента на одной строке можно было бы использовать оператор writeln вместо write.

Существенно то, что если обработка массива осуществляется последовательно, по 1 элементу, циклы ввода и обработки зачастую можно объединить, как в следующем примере.

Найти арифметическое среднее элементов вещественного массива t размерностью 6 и значение его минимального элемента.

```
var b: array [1..6] of real;  s, min:real;  i:integer;
begin
s:=0; min:=1e+30; writeln ('Ввод B[6]');
for i:=1 to 6 do begin  read (b[i]);  s:=s+b[i];  if b[i]<min then min := b[i]; end;
writeln ('min=',min,' s=', s/6);
end.
```

Теоретически в этой программе можно было бы обойтись и без массива - ведь элементы b[i] используются только для накопления суммы и поиска максимума, так что описание массива вполне можно было заменить описанием вещественной переменной b. Однако, в реальных задачах данные, как правило, обрабатываются неоднократно и без массивов обойтись трудно. Приведем пример учебной задачи, где использование массива дает выигрыш за счет уменьшения объема вычислений, выполняемых программой.

Задана последовательность Ti = max {sin(i), cos(i)}, i= -5, -4, ..., 5. Найти элемент последовательности, имеющий минимальное отклонение от арифметического среднего положительных элементов.

Здесь в первом цикле можно сформировать массив по заданному правилу и найти арифметическое среднее положительных элементов. Во втором цикле, когда среднее известно, можно искать отклонение. Без использования массива необходимо было бы считать элементы последовательности дважды.

```
var t : array [-5..5] of real;  i,k:integer;  s, ot : real;
begin
 s:=0; k:=0;
 for i:=-5 to 5 do begin
  t[i]:=sin(i);   if t[i]<cos(i) then t[i]:=cos(i);
  if t[i]>0 then begin    k:=k+1; s:=s+t[i];   end;
 end;
 s:=s/k;  ot:=1e30;
 for i:=-5 to 5 do begin
```

```
 if abs(t[i]-s)<ot then ot:= abs(t[i]-s);
end;
writeln ('Ot=', ot:8:2);
end.
```

Распространена обработка в одной задаче сразу нескольких массивов. Приведем пример.

Координаты 10 точек на плоскости заданы массивами x={xi}, y={yi}, i=1, 2, ..., 10. Найти длину ломаной, проходящей через точки (x1, y1), (x2, y2), ..., (x10, y10), а также номер точки, лежащей дальше всего от начала координат.

При решении задачи используем формулу для нахождения расстояния между 2 точками на плоскости, заданными координатами (x1,y1) и (x2,y2): $r = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Обозначим через r расстояние между текущей точкой и следующей, Len искомую длину ломаной, Dist -- расстояние от текущей точки до начала координат, max -- максимальное из этих расстояний, Num -- искомый номер точки.

```
var x,y : array [1..10] of real;
I, num:integer;
r, Len, Dist, max : real;
begin  {найдем max при вводе данных}
 max:=0; {т.к. расстояние не м.б. <0 }
 num:=1; {на случай, если все точки (0,0)}
 writeln ('Введите координаты 10 точек');
 for i:=1 to 10 do begin
  read (x[i], y[i]);   Dist:=sqrt (sqr(x[i]) + sqr (y[i]));
  if dist > max then begin     max:=dist; {запомнили новое расстояние}
    Num:=i;    {и номер точки}
  end;
 end;
 writeln ('Номер точки=',num, ' расстояние=',dist:8:2);
 Len:=0;{длина ломаной - сумма длин сторон}
 for i:=1 to 9 do begin
        {у 10-й точки нет следующей!}
   r:= sqrt (sqr(x[i]-x[i+1])+ sqr(y[i]-y[i+1]));    Len:=Len+r;
 end;
 writeln ('Длина ломаной=',len:8:2);
end.
```

Приведем пример задачи формирования массива по правилу.

Задан массив x из 8 элементов. Сформировать массив y по правилу

$$y_i = \begin{cases} 4 x_i, & \text{если } i - \text{четное} \\ \cos(2 x_i), & \text{если } i - \text{нечетное} \end{cases}$$

и найти количество его положительных элементов.

```
var x,y: array [1..8] of real;  i,k:integer;
begin
 writeln ('Введите массив x из 8 эл.');
for i:=1 to 8 do begin   read (x[i]);
if  i mod 2 =0 then y[i]:=4*x[i]
                    else y[i]:=cos(2*x[i]);
end;
K:=0; writeln ('Массив y');
 for i:=1 to 8 do begin     if y[i]>0 then k:=k+1;    write (y[i]:8:2); end;
 writeln;
 writeln ('K=',k);
end.
```

Пример обработки многомерного массива

Как правило, при обработке многомерных массивов используются вложенные циклы, т.е. цикл по столбцам располагается внутри цикла по строкам. Дана матрица A(3,4), и вектор B (4), состоящие из целых чисел . Умножить матрицу A на вектор B .

```
program  pr4-2 ;
const m=3;   n=4;
var
a  : array [ 1 .. m, 1 .. n ] of    integer; (* описание матрицы *)
b  : array [ 1 .. n ] of   integer;                (* описание вектора *)
c  : array [ 1 .. m ]  of  integer;                ( * описание  C  *)
i,  j: integer;
begin
for i:=1 to m do                (* ввод матрицы  *)
begin writeln (‘введите элементы ‘, i , ‘-той строки’);
for j:=1 to n do read (a [i, j] ); writeln;
end;
writeln (‘введите элементы вектора’);
for j:=1 to n  do                          (* ввод  вектора *)
read (b[ j]);   writeln;
for i:=1 to m do begin
c [ i ]:=0;       for j:=1 to n do       c[i] := c[ i ]+ a[i , j]* b[j];
end;
for i:=1 to m do      (*форматный вывод матрицы  *) begin
for j:=1 to n do write (a [i, j]: 4 );  writeln;
end;
for j:=1 to n do  write (b [ j ] :4);    (* вывод  массива B *)
writeln ;
for i:=1 to m do  write (c [ i ] :4);   (* вывод  массива C  *)
readln;
end.
```

В программе элементы матрицы вводятся по строкам по одному с подтверждением клавишей Enter. А выводятся в общепринятом виде: каждая строка матрицы с новой строки экрана (цикл i по строкам внешний, а цикл j – внутренний)

Б. Обработка массивов с помощью языков высокого уровня на примере языка Java.

Язык Java является на сегодняшний момент самым распространенным языком в мире, поскольку на нем создаются как программные продукты всемирной сети Интернет, так и мобильные приложения самой распространенной операционной системы Android. Язык Java создана на основе C++, к небольшим отличиям между этими языками следует отнести работу с массивами и отсутствие указателей в языке Java.

В отличае от языка Паскаль, массив Java создается в любом месте программы с использованием следующей конструкции:

Тип[] ИмяМассива=new Тип [количество элементов];

Здесь

Тип - любой из известных типов данных языка Java. Каждый элемент массива будет рассматриваться как переменная соответствующего типа.

new - оператор создания блока памяти. Так как массивы в языке Java динамические, при создании массива выделяется область памяти за пределами основной программы. Это позволяет создавать массивы любой длинны, даже размером сопоставимым с физической памятью компьютера.

Количество элементов - описывает общее количество элементов, которые будут созданны. При этом, как и в языке C++ нижний индекс массива всегда равен нулю, а верхний всегда на единицу меньше количества элементов в массиве. То есть, при создании массива, содержащего n элементов, индексы этого массива всегда будут в диапазоне 0…n-1.

Опишем несколько массивов разного назначения.

```
var a: array [1..20] of integer;
int[] a=new int[20];
```

Здесь описан массив с именем A, состоящий из 20 целочисленных элементов; Индексы этих элементов начинаются с 0 и заканчиваются числом 19.

```
var x,y : array [1..10] of real;
float[]  x=new float[10];
double[] y=new double[4];
```

Описаны 2 массива с именами x и y. Массив x содержит 10 вещественных элементов, массив y содержит 4 вещественных числа с двойной точностью.

```
String[] t=new String[10];
```

Массив t состоит из 10 строк, которые занумерованны с нуля.

Для обращения к отдельному элементу массива используется оператор вида ИмяМассива [Индекс].

Здесь Индекс - целочисленный номер элемента (может быть целочисленным выражением или константой). Индекс в языке Java не должен быть меньше нуля и не должен быть больше значения n-1, где n является количеством элементов в массиве, иначе возникнет ошибка времени выполнения. Отдельный элемент

массива можно использовать так же, как переменную соответствующего типа, например:

a[1] =1; x[1]=1.5; y[1]=x[1]+1; t[0] ="Hello";

Помните, что во всех современных языках программирования (C++, C#, Java) имена чувствительны к регистру. Это означает, что a[1] и A[1] являются разными массивами.

Как и в языке Паскаль, ввод, обработка и вывод массива осуществляются поэлементно, с использованием любого цикла: for, while, do…while. При этом, так как массивы в языке Java динамические, можно задать размер массива в процессе выполнения программы. Приведем пример ввода массива произвольной длинны с клавиатуры и печать его элементов:

```
Scanner input=new Scanner(System.in);
int n;
System.out.println("Введите количество элементов");
n=input.nextInt ();
int[] a=new int[n];
for (int i=0;i<n;i++)
{System.out.println ("Введите элемент номер "+(i+1));
a[i]=input.nextInt();
}
```

В данном примере в первой строке создается объект с именем input для чтения данных с клавиатуры компьютера. Далее создается целочисленная переменная n. У пользователя спрашивается о требуемом количестве элементов в массиве и после ввода числа n создается массив требуемой длинны. Далее, с использованием цикла for поэлементно заполняется массив, при этом ввод элементов выполняется с использованием созданного выше объекта input в строчке input.nextInt().

При решении ряда задач вводить массивы "вручную", особенно если их размерность велика, не всегда удобно. Существуют, как минимум, два альтернативных решения.

Описание массива с заданными начальными элементами производится следующим образом:

float[] m={ 3.5, 2, -5, 4, 11.7 };

Как видно из примера, элементы массива перечисляются в фигурных скобках через запятую, их количество определяется общим количеством чисел в ряду (в данном случае 4), а индексы массива находятся в диапазоне 0…3.

Формирование массива из случайных значений уместно, если при решении задачи массив служит лишь для иллюстрации того или иного алгоритма, а конкретные значения элементов несущественны. Для того чтобы получить очередное случайное значение, используется стандартная функция random() класса Math вызов которой осуществляется следующим образом: Math.random(). Данная функция возвращает число типа double в диапазоне 0…1. Например, оператор вида a[1]=Math.random(); запишет в a[1] случайное число из диапазона [0,1]. Если вам понадобится числов диапазоне 0..100 целого типа,

то его можно получить вот так: a[1]=(int)(Math.random()*100); В этом примере иллюстрируется преобразование типов, при котором новый, требуемый тип заключается в круглые скобки, как в примере (int). Генерируемое функцией random() число преобразуется в целый тип и будет присвоено ячейке массива a[1].

Следует добавить, что генератор псевдослучайных чисел языка Java привязан к счетчику реального времени (микросхеме таймера компютера), таким образом смена основания генератора не требуется.

Приведем пример заполнения массива из 20 элементов случайными числами, лежащими в диапазоне от -10 до 10:

```
int[] a=new int[20];
        for (int i=0;i<20;i++)
        {
                a[i]=(int)(Math.random()*21-10);
                System.out.print(a[i]+" ");
        }
```

Еще более удобный путь - чтение элементов массива из текстового или численногоо файла. К массивам применимы все типовые алгоритмы, изученные в теме "Циклы". Приведем один пример, в котором вычисляется сумма s положительных элементов массива.

```
        Scanner input=new Scanner(System.in);
        int[] b=new int[5];
        float s;
        System.out.println("Введите 5 элементов массива"); for (int i=0;i<5;i++)
b[i]=input.nextInt();
         s=0; for (int i=0;i<5;i++) if (b[i]>0) s=s+b[i];
         //Вывод массива на экран также делается с помощью цикла for.
        for (int i=0;i<5;i++) System.out.println(b[i]+" ");
        System.out.println("Сумма положительных элементов: "+s);
```

Здесь 5 элементов массива b напечатаны в одну строку с разделителем в виде пробела. Для вывода одного элемента на одной строке можно было бы использовать функцию println() вместо print().

Существенно то, что если обработка массива осуществляется последовательно, по 1 элементу, циклы ввода и обработки зачастую можно объединить, как в следующем примере.

Найти арифметическое среднее элементов вещественного массива t размерностью 6 и значение его минимального элемента.

```
        float[] b=new float[6];
        float s=0,min=999999;
        System.out.println("Ввод B[6]");
        for (int i=0;i<6;i++)
                {b[i]=input.nextInt();
                s=s+b[i]; if( b[i]<min) min=b[i];
                }
        System.out.println("min= "+min+" s="+s/6.0);
```

Обратите внимание на выражение s/6.0. Язык Java, так же как и языки C++ и C# являются "более низкоуровневыми", в сравнении с языком Паскаль. В результате, при вычислении частного, если оба аргумента являются целыми числами, микропроцессор при делении использует целочисленную арифметику, потеряв дробную часть. Чтобы этого не происходило деление производят не нацелое число, а на дробное, которым, в частности, является число 6.0.

Теоретически в этой программе можно было бы обойтись и без массива - ведь элементы b[i] используются только для накопления суммы и поиска максимума, так что описание массива вполне можно было заменить описанием вещественной переменной b. Однако, в реальных задачах данные, как правило, обрабатываются неоднократно и без массивов обойтись трудно. Приведем пример учебной задачи, где использование массива дает выигрыш за счет уменьшения объема вычислений, выполняемых программой.

Задана последовательность Ti = max {sin(i), cos(i)}, i= -5, -4, ..., 5. Найти элемент последовательности, имеющий минимальное отклонение от арифметического среднего положительных элементов.

Здесь в первом цикле можно сформировать массив по заданному правилу и найти арифметическое среднее положительных элементов. Во втором цикле, когда среднее известно, можно искать отклонение. Без использования массива необходимо было бы считать элементы последовательности дважды.

```
float[] t=new float[10];
int k=0;
float s=0;
int count=0;
for (int i=-5;i<5;i++)
{t[count]=(float)Math.sin(i);
   if (t[count]<Math.cos(i)) t[count]=(float)Math.cos(i);
   if (t[count]>0){k++;s=s+t[count];};
    count++;
}
s=s/k;count=0;
float ot=1e30f;
for (int i=-5;i<5;i++) if (Math.abs(t[count]-s)<ot) {ot=Math.abs(t[count]-s);count++;};
System.out.println("ot="+ot);
```

Распространена обработка в одной задаче сразу нескольких массивов. Приведем пример.

Координаты 10 точек на плоскости заданы массивами x={xi}, y={yi}, i=1, 2, ..., 10. Найти длину ломаной, проходящей через точки (x1, y1), (x2, y2), ..., (x10, y10), а также номер точки, лежащей дальше всего от начала координат.

При решении задачи используем формулу для нахождения расстояния между 2 точками на плоскости, заданными координатами (x1,y1) и (x2,y2): $r = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Обозначим через r расстояние между текущей точкой и следующей, Len искомую длину ломаной, Dist -- расстояние от текущей точки до начала координат, max -- максимальное из этих расстояний, Num -- искомый номер точки.

```
float[] x=new float[10],y=new float[10];
int i,num;
float r,len,dist=0,max;
//найдем max при вводе данных
max=0;//т.к. расстояние не м.б. <0
num=1; //на случай, если все точки (0,0)
System.out.println("Введите координаты 10 точек");
for (i=0;i<10;i++)
 {x[i]=input.nextFloat();y[i]=input.nextFloat();
 dist=(float)Math.sqrt (x[i]*x[i] + y[i]*y[i]);
 if (dist > max){  max=dist; //запомнили новое расстояние
                num=i;    //и номер точки
              }
}
System.out.println("Номер точки="+num+" расстояние="+dist);
len=0;//длина ломаной - сумма длин сторон
for(i=0;i<9;i++)
{//у 10-й точки нет следующей!}
r= (float)Math.sqrt (Math.pow(x[i]-x[i+1],2)+ Math.pow(y[i]-y[i+1],2));len=len+r;
}
System.out.println("Длина ломаной="+len);
```

Приведем пример задачи формирования массива по правилу.
Задан массив x из 8 элементов. Сформировать массив y по правилу

$$y_i = \begin{cases} 4x_i, & \text{если } i - \text{четное} \\ \cos(2x_i), & \text{если } i - \text{нечетное} \end{cases}$$

и найти количество его положительных элементов.

```
Scanner input=new Scanner(System.in);
float[] x=new float[8];
float[] y=new float[8];
int k;
System.out.println("Введите массив x из 8 эл.");
for (int i=0;i<8;i++)
{ x[i]=input.nextInt();
if  (i%2==0) y[i]=4*x[i];
   else y[i]=(float)Math.cos(2*x[i]);
}
k=0; System.out.println("Массив y");
for (int i=0;i<8;i++){    if (y[i]>0)  k=k+1;System.out.println(y[i]); }
System.out.println("K="+k);
```

Пример обработки многомерного массива

Как правило, при обработке многомерных массивов используются вложенные циклы, т.е. цикл по столбцам располагается внутри цикла по строкам. Дана матрица A(3,4), и вектор B (4), состоящие из целых чисел . Умножить матрицу A на вектор B .

```
Scanner input=new Scanner(System.in);
int m=3, n=4;
int[][] a =new int[m][n];// описание матрицы
int[] b=new int[n];//описание вектора
int[] c=new int[m];//описание  C
int i,  j;

for (i=0;i<m;i++)//ввод матрицы
{System.out.println("введите элементы "+ i+" -той строки");
for (j=0;j<n;j++) a[i][j]=input.nextInt();
}
System.out.println("введите элементы вектора");
for (j=0;j<n;j++) b[j]=input.nextInt(); // ввод  вектора

for(i=0;i<m;i++)
{c[i]=0;for(j=0;j<n;j++) c[i]=c[i]+a[i][j]*b[j];
}
for(i=0;i<m;i++)
{for (j=0;j<n;j++)System.out.println(a[i][j]);  System.out.println();
}
for (j=0; j<n;j++) System.out.println (b[j]+"    ");//вывод  массива B
System.out.println();
for(i=0;i<m;i++)System.out.println(c[i]+"    ");// вывод  массива C
```

В программе элементы матрицы вводятся по строкам по одному с подтверждением клавишей Enter. А выводятся в общепринятом виде: каждая строка матрицы с новой строки экрана (цикл i по строкам внешний, а цикл j – внутренний)

Пример обработки массивов на языке Pyton

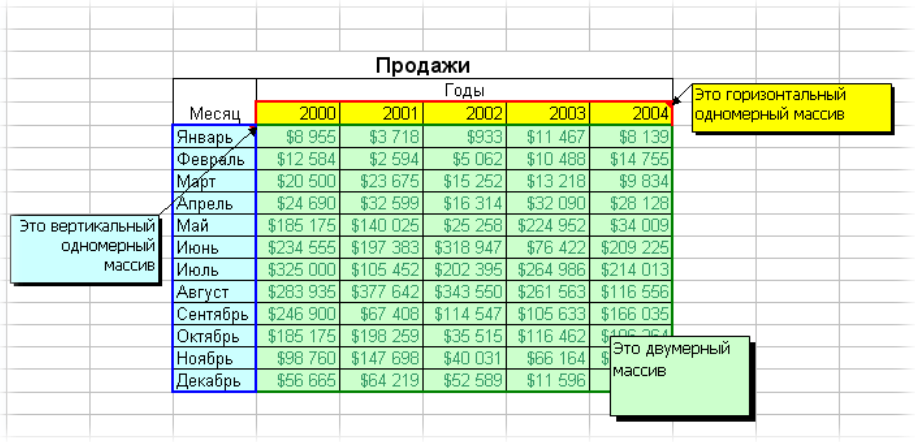
На числовом массиве среди целых чисел, принадлежащих числовому отрезку [174457; 174505], требуется еайти числа, имеющие ровно два различных натуральных делителя, не считая единицы и самого числа. Для каждого найденного числа запишите эти два делителя в два соседних столбца на экране с новой строки в порядке возрастания произведения этих двух делителей. Делители в строке также должны следовать в порядке возрастания.

```
def f(x):
    k=2
    deliteli=set()
    while k * k <= x:
        if x % k==0:
            deliteli.add(k)
            if x // k < x:
                deliteli.add(x // k)
        k = k + 1
```

```
    return sorted(deliteli)
start = 174457
end = 174505
for i in range(start, end + 1):
    if len(f(i)) == 2:
        print(f(i))
```

Б. Обработка массивов в Excel.

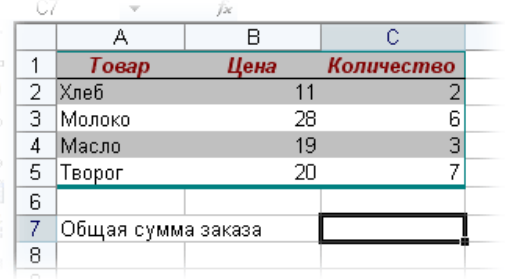
В любой таблице Excel при желании можно найти один или несколько одномерных и многомерных массивов:



Месяц	2000	2001	2002	2003	2004
Январь	$8 955	$3 718	$933	$11 467	$8 139
Февраль	$12 584	$2 594	$5 062	$10 488	$14 755
Март	$20 500	$23 675	$15 252	$13 218	$9 834
Апрель	$24 690	$32 599	$16 314	$32 090	$28 128
Май	$185 175	$140 025	$25 258	$224 952	$34 009
Июнь	$234 555	$197 383	$318 947	$76 422	$209 225
Июль	$325 000	$105 452	$202 395	$264 986	$214 013
Август	$283 935	$377 642	$343 550	$261 563	$116 556
Сентябрь	$246 900	$67 408	$114 547	$105 633	$166 035
Октябрь	$185 175	$198 259	$35 515	$116 462	$
Ноябрь	$98 760	$147 698	$40 031	$66 164	$
Декабрь	$56 665	$64 219	$52 589	$11 596	

Формулы массива в Excel - это специальные формулы для обработки данных из таких массивов. Формулы массива делятся на две категории - те, что возвращают одно значение и те, что дают на выходе целый набор (массив) значений. Рассмотрим их на простых примерах...

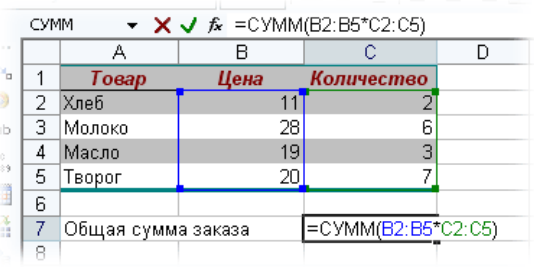
Пример 1. Товарный чек



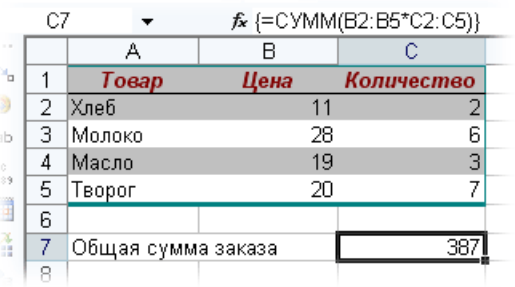
	A	B	C
1	*Товар*	*Цена*	*Количество*
2	Хлеб	11	2
3	Молоко	28	6
4	Масло	19	3
5	Творог	20	7
6			
7	Общая сумма заказа		
8			

Задача: рассчитать общую сумму заказа. Если идти классическим путем, то нужно будет добавить столбец, где перемножить цену и количество, а потом взять сумму по этому столбцу. Если же применить формулу массива, то все будет гораздо красивее:

1. выделяем ячейку **C7**
2. вводим с клавиатуры **=СУММ(**
3. выделяем диапазон **B2:B5**
4. вводим знак умножения (**звездочка**)
5. выделяем диапазон **C2:C5** и закрываем скобку функции СУММ - в итоге должно получиться так:



6. чтобы Excel воспринял формулу как формулу массива гажимаем не Enter, как обычно, а **Ctrl + Shift + Enter.**



Т.е., Excel произвел попарное умножение элементов массивов B2:B5 и C2:C5 и образовал новый массив стоимостей (в памяти компьютера), а затем сложил все элементы этого нового массива.

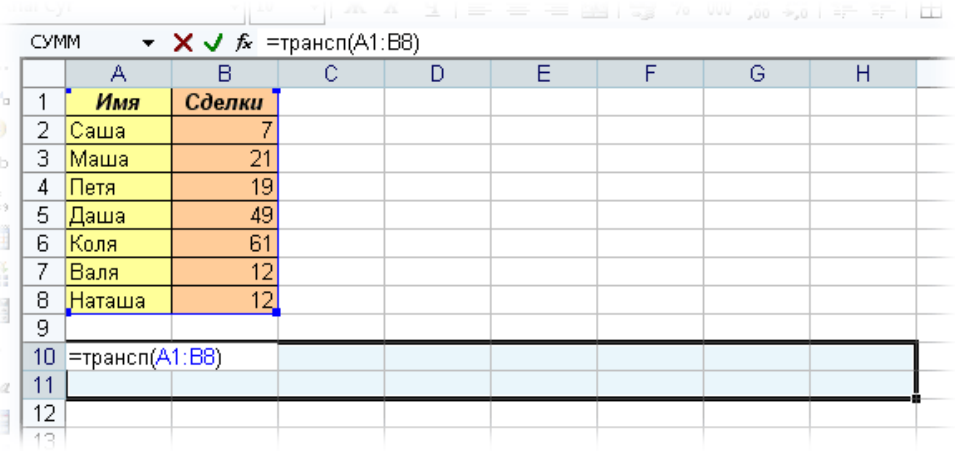
Обратите внимание на фигурные скобки, появившиеся в формуле - отличительный признак *формулы массива*. Вводить их вручную с клавиатуры бесполезно - они автоматически появляются при нажатии **Ctrl + Shift + Enter.**

Пример 2. Транспонирование.

При работе с таблицами часто возникает необходимость поменять местами строки и столбцы, т.е. развернуть таблицу на бок, чтобы данные, которые раньше шли по строке, теперь располагались в столбцах и наоборот. В математике такая операция называется транспонированием. При помощи формулы массива и функции **ТРАНСП (TRANSPOSE)** это делается на раз.

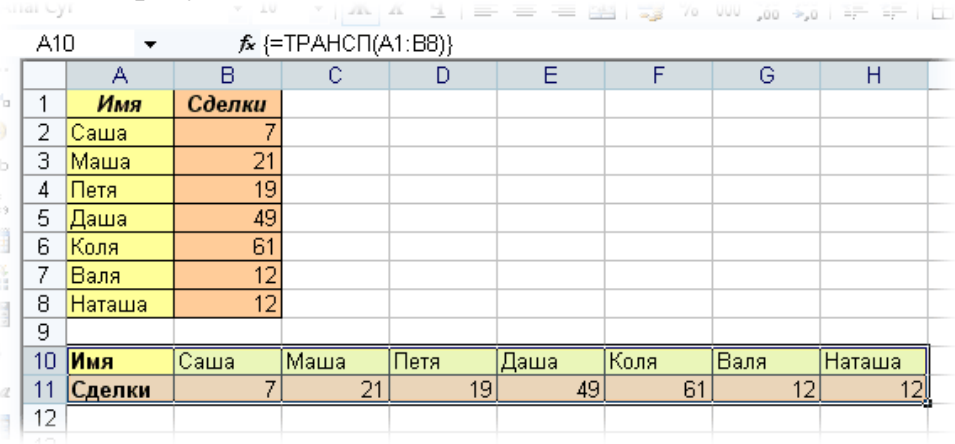
Допустим, имеем двумерный массив ячеек, который хотим транспонировать.

- Выделяем диапазон ячеек для размещения транспонированной таблицы. Поскольку исходный массив ячеек был 8 строк на 2 столбца, то надо выделить диапазон пустых ячеек размером 2 строки на 8 столбцов.
- вводим функцию транспонирования =ТРАНСП(
- в качестве аргумента функции выделяем наш массив ячеек A1:B8



	A	B
1	**Имя**	**Сделки**
2	Саша	7
3	Маша	21
4	Петя	19
5	Даша	49
6	Коля	61
7	Валя	12
8	Наташа	12

Нажимаем клавиши **Ctrl + Shift + Enter** и получаем "перевернутый массив" в качестве результата:



	A	B	C	D	E	F	G	H
1	**Имя**	**Сделки**						
2	Саша	7						
3	Маша	21						
4	Петя	19						
5	Даша	49						
6	Коля	61						
7	Валя	12						
8	Наташа	12						
9								
10	**Имя**	Саша	Маша	Петя	Даша	Коля	Валя	Наташа
11	**Сделки**	7	21	19	49	61	12	12
12								

Редактирование формулы массива

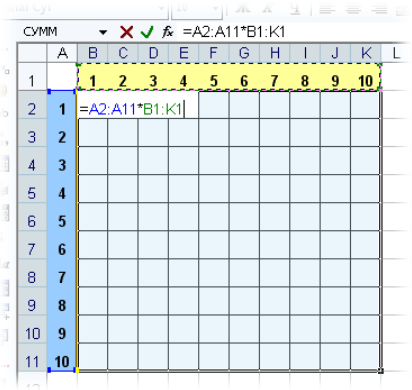
Если формула массива расположена не в одной ячейке, а в нескольких ячейках, то Excel не позволит редактировать или удалить одну отдельно взятую формулу (например в ячейке D10) и выдаст предупреждающее сообщение **Невозможно изменить часть массива.**

Для редактирования формулы массива необходимо выделить весь диапазон (A10:H11 в данном случае) и изменить формулу в строке формул (или нажав **F2**). Затем необходимо повторить ввод измененной формулы массива, нажав сочетание клавиш **Ctrl + Shift + Enter.**

Excel также не позволит свободно перемещать ячейки, входящие в формулу массива или добавлять новые строки-столбцы-ячейки в диапазон формулы массива (т.е. в диапазон A10:H11 в нашем случае)

Пример 3. Таблица умножения

При помощи формул массива она вся делается в одно движение:



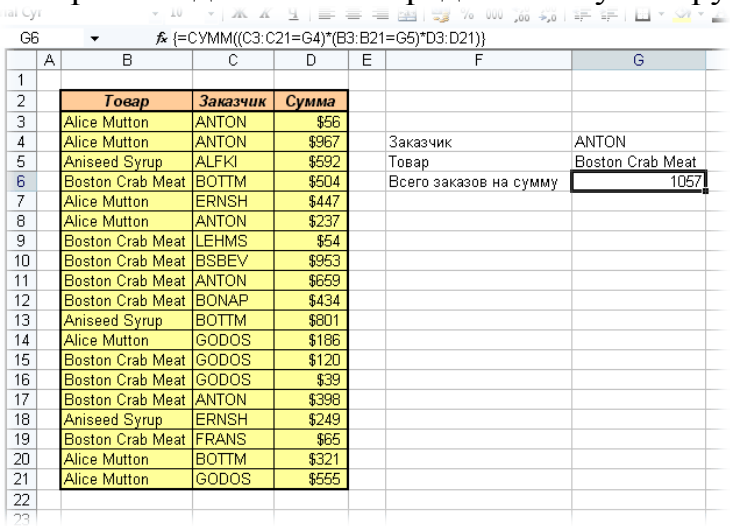
1. выделяем диапазон B2:K11
2. вводим формулу =A2:A11*B1:K1
3. нажимаем **Ctrl + Shift + Enter**, чтобы Excel воспринял ее как формулу массива

и получаем результат:

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

Пример 4. Выборочное суммирование

Посмотрите как при помощи одной формулы массива красиво и легко выбираются данные по определенному товару и заказчику:



Товар	Заказчик	Сумма
Alice Mutton	ANTON	$56
Alice Mutton	ANTON	$967
Aniseed Syrup	ALFKI	$592
Boston Crab Meat	BOTTM	$504
Alice Mutton	ERNSH	$447
Alice Mutton	ANTON	$237
Boston Crab Meat	LEHMS	$54
Boston Crab Meat	BSBEV	$953
Boston Crab Meat	ANTON	$659
Boston Crab Meat	BONAP	$434
Aniseed Syrup	BOTTM	$801
Alice Mutton	GODOS	$186
Boston Crab Meat	GODOS	$120
Boston Crab Meat	GODOS	$39
Boston Crab Meat	ANTON	$398
Aniseed Syrup	ERNSH	$249
Boston Crab Meat	FRANS	$65
Alice Mutton	BOTTM	$321
Alice Mutton	GODOS	$555

Заказчик	ANTON
Товар	Boston Crab Meat
Всего заказов на сумму	1057

В данном случае формула массива синхронно пробегает по всем элементам диапазонов C3:C21 и B3:B21, проверяя, совпадают ли они с заданными значениями из ячеек G4 и G5. Если совпадения нет, то результат равенства ноль, если совпадение есть, то единица. Таким образом суммы всех сделок, где

заказчик не ANTON и товар не Boston Crab Meat умножаются на ноль и суммируются только нужные заказы.

Простейшие операции с массивами

Часто при работе с таблицами возникает необходимость применить одну и ту же операцию к целому диапазону ячеек или произвести расчеты по формулам, зависящим от большого объема данных. **Массив** – это прямоугольный диапазон ячеек с однородными (однотипными) данными. Использование массивов дает возможность ввести одну формулу для выполнения вычислений сразу во многих ячейках. Формула массива может выполнить несколько вычислений, а затем вернуть одно или группу значений. Для работы с массивами есть специальные функции, кроме того, обычные функции для обработки массивов применяются особым образом. Рассмотрим **операцию умножения массива на число** . В качестве примера введите данные в диапазон A1:B2. Далее выделите на рабочем листе область такого же размера, как массив-множимое (например, D 1: E 2). В строке формул введите формулу = A 1: B 2*5 и закончите ввод нажатием сочетания клавиш < Ctrl >+< Shift >+< Enter >. Таким образом вы сообщите программе, что необходимо выполнить операцию над массивом. При этом Excel заключит формулу в строке формул в фигурные скобки {= A 1: B 2*5} **Эти скобки нельзя вводить вручную**, при этом возникнет сообщение об ошибке. При работе с массивами формула действует на все ячейки диапазона. Нельзя изменять отдельные ячейки в операндах формулы. Аналогично можно вычислить:

- сумму (разность) массивов {= A 1: B 2+ D 1: E 2};
- поэлементное умножение(деление) массивов {= A 1: B 2* D 1: E 2};
- массив, каждый элемент которого связан посредством некоторой функции с соответствующим элементом первоначального массива {= sin (A 1: B 2)}.

Примером **специальной функции** для работы с массивом может служить функция транспонирования массива ТРАНСП. Введите в таблицу значения массива. Выделите целевой диапазон ячеек (т.е. диапазон, в который будет вставлен результат транспонирования). Целевой диапазон обязательно должен соответствовать по количеству строк и столбцов ожидаемому результату. Если выделите иное количество строк или столбцов, получите сообщение об ошибке. Затем выполните вставку функции ТРАНСП из категории **Ссылки и массивы.** Во втором окне введите вручную или укажите с помощью мыши исходный диапазон с данными. Завершите ввод нажатием комбинации клавиш < Ctrl >+< Shift >+< Enter >. В Excel могут быть созданы формулы массивов, в которых сами формулы содержат массивы значений, используемых в расчетах. Например, в ячейках B2:F2 находятся данные 1, 2, 3, 4, 5; и 2, 4, 6, 8, 10 – данные, находящиеся в ячейках B3:F3. Тогда результаты попарного сложения могут быть помещены в диапазон B4:F4 с помощью формулы: = {1;2;3;4;5} +{2;4;6;8;10}. После ввода формулы и нажатия сочетания клавиш <CTRL+SHIFT+ENTER> Excel поместит в каждую ячейку выделенного

диапазона формулу: {={1;2;3;4;5}+{2;4;6;8;10}}. **Формулы массива** редактируются так же, как и другие формулы. Нужно дважды щелкнуть по содержащей формулу ячейке, которую необходимо редактировать. В поле ввода можно внести в формулу необходимые изменения. По окончании редактирования обязательно нажать сочетание клавиш <Ctrl+Shift+Enter>. Только в этом случае все изменения в формуле будут внесены во все ячейки, содержащие данную формулу. Действия, выполняемые Excel при обработке формул массивов, могут рассматриваться как математические операции над матрицами. Результат вычислений с матрицами приводит к созданию новых матриц. Если правильно составить формулу, то Excel позволяет производить вычисление и обработку диапазонов разных размеров.

В Excel имеются следующие специальные функции для работы с матрицами: МОБР – обратная матрица; МОПРЕД – определитель матрицы; МУМНОЖ – матричное произведение двух матриц; и уже знакомая функция ТРАНСП – транспонированная матрица.

Во всех случаях при работе с матрицами перед вводом формулы надо выделить область на рабочем листе, куда будет выведен результат вычислений. В качестве примера решим систему линейных уравнений с двумя неизвестными, матрица коэффициентов которой записана в ячейки D 1: E 2 , а свободные члены – в ячейки G1:G2 . Напомним, что решение системы линейных уравнений, записанной в матричном виде **AX=B** , где **A** – матрица коэффициентов, **B** – столбец (вектор) свободных членов, **X** – столбец (вектор) неизвестных, имеет вид **X=A -1 B** , где **A -1** - матрица, обратная по отношению к матрице **A** . Выделим под вектор решений диапазон H1:H2 и введем формулу **{=МУМНОЖ(МОБР(D 1: E 2);G1:G2)}.**

Другой пример - решение системы линейных уравнений ***A 2 X=B*** , где $A = \begin{pmatrix} 7 & 2 \\ 1 & 4 \end{pmatrix}, \quad B = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$

С учетом, что элементы матрицы **A** введены в ячейки A1:B2 , элементы матрицы свободных членов B – в диапазон D 1: D 2 , а в диапазон F 1: F 2 поместим элементы вектора решения, формула будет иметь вид **{=МУМНОЖ(МОБР(МУМНОЖ(A1:B2;A1:B2)); D 1: D 2)}** Результатом вычисления приведенных примеров является массив. При вычислении следующих выражений результатом является одно число. Пример – вычислить квадратичную форму $z = X^T AX$, где символ (T) – операция транспонирования матрицы. Матрица **A** введена в диапазон A1:B2 , вектор **X** – в диапазон D1:D2 . Для вычисления в ячейку F1 введем формулу **{=МУМНОЖ(МУМНОЖ(ТРАНСП(D 1: D 2);A1:B2); D 1: D 2)}** Хотя результатом является число, обязательно нажмите комбинацию клавиш <Ctrl+Shift+Enter>. Решение квадратичной формы $z = Y^T A^T AY$, где

$A = \begin{pmatrix} 7 & 2 \\ 1 & 4 \end{pmatrix}, \quad Y = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$Матрица **A** введена в диапазон A1:B2 , вектор **Y** – в диапазон D1:D2 . Для вычисления в ячейку F1 введем формулу нужно ввести формулу:**{=МУМНОЖ(ТРАНСП(D 1: D 2);МУМНОЖ(ТРАНСП(A1:B2); МУМНОЖ(A1:B2; D 1: D 2)))} .**

ЗАДАНИЕ:

А. Составить и отладить программу на языке высокого уровня.

Перед выполнением задания изучите теоретический материал.

1. Напишите фрагмент программы для заполнения элементов массива датчиком случайных чисел в диапазоне (sin(N), cos(N)*2), где N –порядковый номер в студенческой группе.
2. Создайте и распечатайте в виде таблицы квадратную матрицу размерностью k=mod(N,4)+3.
3. Создайте массив A(3), сформированный датчиком случайных чисел на интервале (-50,113). Из элементов массива A кратных N сформируйте массив B.
4. Составьте по следующей программе блок-схему и поясните ее функциональное назначение.

```
For i:=1 to 40 do begin
A[i]:=random(100);
Writeln('number', i, ' element array', a[i]
End;
For i:=1 to 20 do begin
P:=a[i]; a[i]:=a[40-i+1]; a[40-i+1]:=p
End;
```

Выполните трассировку программы.

5. Напишите и отладьте программу вычисления средних значений столбцов положительных элементов в квадратной матрице.
6. Напишите и отладьте программу преобразования матрицы в вектор.
7. Напишите и отладьте программу преобразования вектора в матрицу.
8. Составьте алгоритм замены в матрице строк – содержащей элемент с максимальным значением и содержащей минимальное значение (предполагается, что эти элементы единственные).
9. Составьте алгоритм и программу преобразования предложения в квадратную символьную матрицу, каждым элементом которой является два символа из предложения, если они являются буквами, три – если среди них встречается хотя бы одна цифра, четыре – если нет цифр и есть знак препинания.

Б. **В электронной таблице EXCEL**.

Перед выполнением задания изучите теоретический материал и просмотрите видиоурок в Интернет по адресу:

http://www.titorov.ru/index.php/distant/inform-tecnology/535-excel

1. Создать массив целых чисел с помощью датчика случайных чисел в заданном интервале – массив X (10 элементов).
2. Создать матрицу действительных чисел в диапазоне [-1,1] – матрица H (10*10 элементов).
3. С помощью командной строки: определить минимальный четный элемент в массиве X, определить сумму элементов массива H, отсортировать каждый столбец матрицы H по возрастанию, отобразить каждую строчку массива H в виде различных диаграмм, построить точечные графики зависимости массива X от каждого столбца матрицы H и определить соответствующие функциональные зависимости линий тренда, транспонируйте матрицу H, перемножьте массивы X и H, найдите определитель матрицы H.

Составьте отчет, включающий в себя результаты выполненной работы и краткие ответы на контрольные вопросы.

Контрольные вопросы.

1. Опишите каким образом можно осуществить ввод линейного массива в «строчку» и-или в «столбец»?
2. Каким образом задать трехмерный массив?
3. Опишите способы заполнения массивов.
4. Перечислите основные алгоритмы обработки массивов.
5. Приведите алгоритм в виде блок-схемы, словесного описания, графа, языке высокого уровня, выполняющий формирование из одномерного массива двух одномерных – в один входят отрицательные элементы, в другой – положительные.
6. Опишите способы выводы матрицы на печать.
7. В чем особенности организации программ для обработки массивов произвольных размеров?
8. Составьте алгоритм и программу формирования квадратной матрицы по спирали:

1	2	3	4	5	6
20	21	22	23	24	7
19	28	27	26	25	8
18	29	30	31	32	9
17	36	35	34	33	10
16	15	14	13	12	11

9. Обработка символьной информации ная языках высоеого уровня и в электронных таблицах.

Краткие теоретические сведения.

С помощью компьютера можно решать весьма разнообразные задачи обработки текстов: от составления платежных ведомостей до автоматической верстки газет. Для того, чтобы компьютер мог обрабатывать тексты, он должна уметь оперировать не только с числами, но и со словами.

Будем полагать, что текст- это произвольная последовательность символов некоторого алфавита.*Алфавитом может служить любое множество символов, например (0, 1,2, ...), (А, Б, В,...), (A, B, C,...).*

Строкой символов, или символьной (строковой, текстовой) константой, будем называть последовательность символов, заключенных в апострофах.

Строка символов может состоять из одного или нескольких символов, а также не содержать ни одного символа (пустая строка, или строка нулевой длины). Максимальная длина текстовой строки 255 символов.

В Turbo Pascal 7.0 для работы с символами используются два типа переменных: ***символьный тип и строковый тип данных.***

1. Символьный тип данных (Char).

Описание: идентификатор char, (var x: char).

Диапазон значений: значением переменной этого типа может быть любой символ – это буквы, цифры, знаки препинания и специальные символы. Каждому символу алфавита соответствует индивидуальный числовой код от 0 до 255.

В Turbo Pascal 7.0 значения для переменных типа char задаются в апострофах: sh := ''; a := '3'; summa :='G'.*

2. Строковый тип данных (string).

Как правило, одно целое число или один символ занимают в памяти ЭВМ два байта. В то же время для изображения символа достаточно одного байта. С целью экономии памяти машины при использовании символьных данных в языке Паскаль введено понятие строки. ***Строкой называется последовательность символов определенной длины.*** *Элементы строки хранятся по два в двух байтах памяти ЭВМ.*

Переменные типа ***string*** *могут быть объявлены следующим образом:*

var *s1: string[30]; s2: string.*

Число 30 означает максимально возможное количество символов строки s1.

В языке программирования Java для работы с символами используется тип данных char и два класса, с именами String и StringBuilder. Класс String позволяет отображать строки на экране, но не имеет механизма манипулирования символами внутри строки. Класс StringBuilder представляет

собой миощный инструмент обработки строк, но не имеет средств отображения на экране.

1. Символьный тип данных (char).

Описание: идентификатор char, (char x;).

Диапазон значений: значением переменной этого типа может быть любой символ – это буквы, цифры, знаки препинания и специальные символы. Данный тип данных занимает 2 байта в памяти, таким образом каждому символу алфавита соответствует индивидуальный числовой код от 0 до 65535. Это позволяет хранить символы всех алфавитов на земле и все символы, которые только необходимы. В связи с этим кодировку символьного типа в языке Java называют юникодом, от английского слова Unicode, что означает "универсальная кодировка".

В Java значения для переменных типа char задаются в апострофах: sh = ''; a = '3';*

summa ='G'.

2. Класс String

Объекты класса ***String*** *могут быть объявлены следующим образом:*

String s1,s2;

Или так: String s1="Привет студентам";

Как видно из примера, количество символов, хранимых строкой не указывается, поскольку строка в языке Java представляет собой динамический объект переменной длинны с диапазоном памяти от нуля до физической памяти ЭВМ. Иными словами, вновь созданная строка по умолчанию является пустой, но в процессе работы ее длинна может увеличиваться до любых значений. Следует помнить, что как и в стандартных массивах, номер первого элемента строки начинается всегда с нуля.щ

2. Класс StringBuilder

Объекты класса StringBuilder объявляются следующим образом:

StringBuilder sb=new StringBuilder("Привет");

Эта структура данных может хранить неограниченное количество символов, позволяет их обрабатывать и в любой момент массив хранимых символов может быть передан в класс String:

String st=sb.toString();

В данном примере создается строка st, которой передается содержимое объекта sb.

II. Стандартные функции для работы с символьными величинами

1. ***Операция сложения символьных величин.***

Операция сложения позволяет строить из двух символьных строк третью, состоящих из символов первой, за которой следуют символы второй. Обозначается эта операция знаком "+". Описываем строковые переменные. *var s1, s2, s3: string;*

Присваиваемое значение строки заключается в апострофы. Присвоим первым двум следующие значения, а третья будет равна их склеиванию:
s1: = ' Тише воды, '; s2:= 'ниже травы'; s3:=s1+''+s2;
Строка *s3* имеет значение 'Тише воды, ниже травы'.
В языке Java используется сходный подход. Операция сложения строк понимается как их объединение, что иллюстрирует следующий пример:
String s1="Иванов",s2="Вася",s3;
s3=s1+' '+s2.
Строка s3 имеет значение "Иванов Вася". Обратите внимание, что объявления строк в Java могут производиться в любом месте программы и в процессе объявления им можно присваивать начальные значения. Вместо единичного символа пробела, заключенного в одинарные кавычки, можно было использовать строку с единичным символом, в этом случае пример выглядел бы так: s3=s1+" "+s2.

2. Длина строки
Под длиной строки понимается количество введенных символов, но она не может превышать максимально возможной длины (в описательной части). Это значение можно определить при помощи функции, результат которой целое число, равное количеству символов.
s1:='12345';
s2:= 'Семеро одного не ждут';
k1:=Length(s1); k2:=Length(s2).
В результате значения целых переменных будут равны: kl=5, k2=21.
Данный пример в языке Java будет представлен следующим образом:
String s1="12345";
String s2= "Семеро одного не ждут";
int k1=s1.length(); **int** k2=s2.length();
3. Копирование
Функция copy(str,n,m*) в Turbo Pascal 7.0* – копируют ***m*** символов строки **str**, начиная с **n**-го символа, при этом исходная строка не меняется. Можно результат этой функции присваивать другой строке или сразу выводить его на экран.
s1:='паровоз'; s2:='123456'; s3:=copy(s1, 5, 3);
writeln(s3); writeln(copy(s2, 3, 2));
Значения переменной s1='воз'. А на экране будут выведены следующие строки: воз и 34.
В языке Java для копирования части строк используется метод replace(начало, конец). При этом "начало" определяет начальный индекс копирования, "конец" - конечный индекс копирования минус один. Например, в этом примере:
String s1;
String s2= "Семеро одного не ждут";
s1=s2.substring(1, 3);

System.***out***.println(s1);
на экране будет напечатано слово "ем", поскольку первый символ копирования равен единице (а строки начинаются с нуля), а последний 3-1=2, скопированы будут символы с индексами 1 и 2.

4. Удаление

В Turbo Pascal 7.0 для этого используется процедура **Delete(str, n,m)**, которая вырезает из строки **str m**символов, начиная с **n**-го. таким образом сама строка изменяется.

Дан фрагмент программы:

s:='123456'; delete(s, 3, 2); writeln(s);

После выполнения этих операторов из строки будут удалены два символа, начиная с третьего, то есть строка будет такой: *s = '1256'*.

В языке Java удаление символов в строке производится с использованием вспомогательного класса StringBuilder, с последующей передачей результата в String. Для примера, вышеописанный пример будет иметь следующий вид:

```
String s="123456";
StringBuilder sb=new StringBuilder(s);
sb.delete(2, 4);
s=sb.toString();
System.out.println(s);
```

Во второй строке программы создается объект sb класса StringBuilder, которому в качестве начального з-начения передается строка. Далее, используется метод delete(2, 4), который удаляет символы, начиная с позиции 2 (не забывайте про нумерацию с нуля), по позицию 4, но не включая ее. Порсле изменения содержимого строки результат передается в переменную s и печатается на экране. Текст результата будет такой: *1256*

5. Замена (Вставка)

*В Turbo Pascal 7.0 это можно сделать, применяя процедуру I**nsert(s1,s2,n)** – вставка строки s1 в строку s2, начиная с **n**-го символа, при этом первая строка остается такой же, как и была, а вторая получает новое значение.*

s1;='34': s2:='1256'; insert (s1, s2, 3);

В результате выполнения данной процедуры строка будет такой *s2='123456'*.

На языке Java вставка символов используется через вспомогательный класс StringBuilder. Вышеописанный пример примет следующий вид:

```
String s1="34";
String s2="1256";
StringBuilder sb=new StringBuilder(s2);
sb.insert(2, s1);
s2=sb.toString();
System.out.println(sb);
```

6. Числа и строки

Надо заметить, что число 25 и строка 25 – это не одно и то же. Для работы с числами и строками в Turbo Pascal 7.0 применяются две процедуры.

Str(n,s1) – переводит числовое значение n в строковое и присваивает результат строке s1, причем можно переводить как целые числа, так и вещественные.

n:=12;

str(n,s1);

- после выполнения s1 ='12';

Существует обратная операция, переводящая строковое значение в числовое.

Функция val(s, n, k) – переводит строковое значение в числовое, если данная строка действительно является записью числа (целого или вещественного), то значение k=0, а n – это число, иначе k будет равно номеру символа, в котором встречается первое нарушение записи числа n.

val('1234',n,k) ***n=1234****, k=0;*

В языке Java также существует перевод строковой информации в численную и наоборот. Для перевода чиленных данных в строковые можно использовать операцию соединения пустой строки и числа, компилятор при этом правильно преобразует типы:

```
int n=25;
String s=""+n;
System.out.println(s);
```

Во втором случае можно использовать метод valueOf класса String:

```
int n=25;
String s=String.valueOf(n);
System.out.println(s);
```

И в первом и во втором случаях на экран будет выведена строка "25"

Обратное преобразование возможно с использованием соответствующих численному типу классов. Следующий пример иллюстрирует перевод строки в численный тип с использованием класса Integer:

```
String s="25";
int n=Integer.valueOf(s);
System.out.println(n);
```

7. Функции преобразования типов

Иногда в программах возникает необходимость по коду определить символ и, наоборот, по символу определить его код. Для этого используют функцию: ***CHR(x)***.

Эта функция возвращает символ, соответствующий ASCII-коду числа *x*.

```
for i = 0 to 255 do
writeln( i,' ', chr(i));
```

Для определения кода по символу используют функцию **ORD**.

```
readln(s);  writeln(ord(s));
```

В языке Java преобразование типа char в численный тип производится с использованием оператора преобразования типа, который выглядит следующим образом:

переменная1=(новый тип)переменная2;
Следующий пример иллюстрирует сказанное:

```
for (int i=0;i<65535;i++)
{
 char c=(char)i;//символ с принял значение кода i
 int n=c;//число n приняло значение символа c
 System.out.println("Код "+i+" имеет символ "+(char)c);
}
```

III. Решение задач на обработку текстовой информации

Пример 1.

Составить программу, определяющую по введенному с клавиатуры символу его код.

```
Programm prim1;
Var s: char;
Begin
Writeln('введите символ с клавиатуры'); Readln(s);
Writeln('код символа ',s,'=',ord(s));
Readln;
End.
```

Пример 2.

В три символьные переменные F, I, O ввести свои фамилию, имя, отчество. Сформировать из этих данных строку S, содержащую ваши фамилию и инициалы.

```
Program prim2
Var F, I, O, S : string;
Begin
Writeln('введите вашу фамилию');
Readln(F);
Writeln('введите ваше имя');
Readln(I);
Writeln('введите ваше отчество');
Readln(O);
S:=F+' '+copy(I,1,1)+'.'+copy(O,1,1)+'.';
Writeln('ваши реквизиты: ', S);
Readln;
End.
```

Пример 3.

Определить сколько цифр содержится в записи произвольного натурального числа.

```
Program prim3;
Var s: string;
x, k: integer;
```

```
Begin
Writeln('введите число');
Readln(x);
Str( x, s);
k:=length(s);
Writeln('в числе ',k,' цифр');
Readln;
End.
```

Пример 4.

Переменные A и B содержат строки цифр. Найти сумму соответствующих чисел.

```
Program prim4;
Var A, B: string;
S, x, y, n, k: integer;
Begin
Writeln('введите первое число');
Readln(A);
Writeln('введите второе число');
Readln(B);
Val(A, x, n);
Val(B, y, k);
S:=x+y;
Writeln('сумма чисел равна ',S);
Readln;
End.
```

Пример 5.

Распечатать заданное слово в одной строке с разрядкой (пробел после каждой буквы).

```
Program prim5;
Var s, x: string;
i: integer;
Begin
Writeln('введите слово');Readln(s);
x:='';
For i:=1 to length(s) do begin  x;=x+copy(s,i,1)+' '; End;
Writeln('получилось слово ', x);
Readln;
End.
```

Пример 6.

Составить программу подсчета количества вхождений буквы “a” в заданном тексте.

```
Program prim6;
Var s: string;
i, k: integer;
Begin
Writeln('введите текст');
Readln(s);
k:=0;
for i:=1 to length(s) do begin
if copy(s, i, 1)='a' then k:=k+1
end;
Writeln('количество букв "a" в тексте равно ', k);
Readln;
End.
```

Пример 7.

Определить, какое из двух исходных слов длиннее и насколько.

```
Program prim7;
Var s1, s2: string;
L1, l2: integer;
Begin
Writeln('введите первое слово'); Readln(s1);
Writeln('введите второе слово'); Readln(s2);
l1:=length(s1);
l2:=length(s2);
if l1>l2 then writeln('первое слово длиннее второго на ',l1-l2,' символов')
else if l1=l2 then writeln('слова одинаковой длины')
else writeln('первое слово длиннее второго на ',l2-l1,' символов');
Readln;
End.
```

Переменная типа char может принимать значения из определенной упорядоченной последовательности символов. Переменная этого типа занимает 1 байт и принимает одно из 256 значений кода ASCII (американский стандартный код для обмена информацией). Символы упорядочены в соответствии с их кодом, поэтому к данным символьного типа применимы операции отношения.

В программе вместо символа можно использовать его код, состоящий из # и номера кодируемого символа (например, #51). Обычно символы, имеющие экранное представление, записывают в явном виде, заключив в апострофы (например, 'A', 'b', '*').

Две стандартные функции позволяют поставить в соответствие данную последовательность символов множеству целых неотрицательных чисел (порядковым номерам символов последовательности).

Эти функции называются функциями преобразования:

ord(ch) – выдает номер символа (нумерация с нуля),
chr(i) – выдает i-ый символ из таблицы символов.
Пример. ord(H) выдает номер символа H в последовательности всех символов, используемых транслятором. chr(15) выдает 15-ый символ этой последовательности.
Кроме того, для символьных переменных применяются такие функции:
pred(ch) – возвращает предыдущий символ;
succ(ch) – возвращает следующий символ;
upcase(ch) – преобразует строчную букву в заглавную. Обрабатывает буквы только латинского алфавита.
В языке Java для вывода кода символа достаточно использовать операцию преобразования типа: char c=(char)65; или обратное преобразование: int n=(int)'A';
Также можно использовать процедуры inc и dec.

Строковый тип данных

Для обработки строковой информации в Турбо Паскаль введен строковый тип данных. Строкой в Паскале называется последовательность из определенного количества символов. Количество символов последовательности называется длиной строки. Синтаксис:
var s: string[n];
var s: string;
n - максимально возможная длина строки - целое число в диапазоне 1..255. Если этот параметр опущен, то по умолчанию он принимается равным 255.
Строковые константы записываются как последовательности символов, ограниченные апострофами. Допускается формирование строк с использованием записи символов по десятичному коду (в виде комбинации # и кода символа) и управляющих символов (комбинации ^ и некоторых заглавных латинских букв).
Примеры:
'Текстовая строка'
#54#32#61
'abcde'^A^M
Пустой символ обозначается двумя подряд стоящими апострофами. Если апостроф входит в строку как литера, то при записи он удваивается.
Переменные, описанные как строковые с разными максимальными длинами, можно присваивать друг другу, хотя при попытке присвоить короткой переменной длинную лишние символы будут отброшены.
Выражения типа char можно присваивать любым строковым переменным.
В Турбо Паскаль имеется простой доступ к отдельным символам строковой переменной: i-й символ переменной st записывается как st[i]. Например, если st - это 'Строка', то st[1] - это 'С', st[2] - это 'т', st[3] - 'р' и так далее.
Над строковыми данными определена операция слияния (конкантенации), обозначаемая знаком +. Например:

a := 'Turbo'; b := 'Pascal'; c := a + b;
В этом примере переменная с приобретет значение 'TurboPascal'.
Кроме слияния над строками определены операции сравнения <,>,=,<>,<=,>=. Две строки сравниваются посимвольно, слева направо, по кодам символов. Если одна строка меньше другой по длине, недостающие символы короткой строки заменяются символом с кодом 0.
В системе Turbo Pascal имеется несколько стандартных процедур и функций, ориентированных на работу со строками. Например:
Length(s:string):integer
Функция возвращает в качестве результата значение текущей длины строки-параметра
n := length('Pascal'); {n будет равно 6}
Concat(s1,[s2,...,sn]:string):string
Функция выполняет слияние строк-параметров, которых может быть произвольное количество. Каждый параметр является выражением строкового типа. Если длина строки-результата превышает 255 символов, то она усекается до 255 символов. Данная функция эквивалентна операции конкатенации "+" и работает немного менее эффективно, чем эта операция.
Copy(s:string; index:integer; count:integer):string
Функция возвращает подстроку, выделенную из исходной строки s, длиной count символов, начиная с символа под номером index.
s := 'Система Turbo Pascal';
s2 := copy(s, 1, 7); {s2 будет равно 'Система'}
s3 := copy(s, 9, 5); {s3 будет равно 'Turbo'}
s4 := copy(s, 15, 6); {s4 будет равно 'Pascal'}
Delete(var s:string; index,count:integer)
Процедура удаляет из строки-параметра s подстроку длиной count символов, начиная с символа под номером index.
s := 'Система Turbo Pascal'; delete(s,8,6); {s будет равно 'Система Pascal'}
Insert(source:string; var s:string;index:integer)
Процедура предназначена для вставки строки source в строку s, начиная с символа index этой строки.
s := 'Система Pascal'; insert('Turbo ',s,9); {s будет равно 'Система Turbo Pascal'}
Pos(substr,s:string):byte
Функция производит поиск в строке s подстроки substr. Результатом функции является номер первой позиции подстроки в исходной строке. Если подстрока не найдена, то функция возвращает 0.
s := 'Система Turbo Pascal'; x1 := pos('Pascal', s); {x1 будет равно 15}
x2 := pos('Basic', s); {x2 будет равно 0}
Str(X: *арифметическое выражение*; var st: string)
Процедура преобразует численное выражение X в его строковое представление и помещает результат в st.
Val(st: string; x: *числовая переменная*; var code: integer)

Процедура преобразует строковую запись числа, содержащуюся в st, в числовое представление, помещая результат в x. x - может быть как целой, так и действительной переменной. Если в st встречается недопустимый (с точки зрения правил записи чисел) символ, то преобразование не происходит, а в code записывается позиция первого недопустимого символа. Выполнение программы при этом не прерывается, диагностика не выдается. Если после выполнения процедуры code равно 0, то это свидетельствует об успешно произошедшем преобразовании.

В дополнение приведем некоторые функции, связанные с типом char, но которые тем не менее часто используются при работе со строками.

Chr(n: byte): char

Функция возвращает символ по коду, равному значению выражения n. Если n можно представить как числовую константу, то можно также пользоваться записью #n.

Ord(ch: char): byte;

В данном случае функция возвращает код символа ch.

UpCase(c: char): char;

Если c - строчная латинская буква, то функция возвращает соответствующую прописную латинскую букву, в противном случае символ c возвращается без изменения.

Понятие множества в языке Паскаль основывается на математическом представлении о конечных множествах: это ограниченная совокупность различных элементов. При этом понятие множества встречается исключительнов языке Паскаль, подобная реализация полностью отсутствует в языках программирования Java, C# и C++. Для построения конкретного множественного типа используется перечисляемый или интервальный тип данных. Тип элементов, составляющих множество, называется базовым типом. Множественный тип описывается с помощью служебных слов Set of, например:

type M = Set of B; Здесь M - множественный тип, B - базовый тип.

Пример описания переменной множественного типа:

```
Type    M = Set of 'A'..'D';
Var    MS: M;
```

Принадлежность переменных к множественному типу может быть определена прямо в разделе описания переменных:

var C: Set of 0..7;

Константы множественного типа записываются в виде заключенной в квадратные скобки последовательности элементов или интервалов базового типа, разделенных запятыми, например: ['A', 'C'] [0, 2, 7] [3, 7, 11..14]

Константа вида [ ] означает пустое подмножество. Количество базовых элементов не должно превышать 256. Инициализация величин множественного типа может производиться с помощью типизированных констант: const seLit: Set of 'A'..'D'= [];

Порядок перечисления элементов базового типа в константах безразличен.

Значение переменной множественного типа может быть задано конструкцией вида [T], где T - переменная базового типа. Например, вполне допустима конструкция: type T = set of char;
Множество включает в себя набор элементов базового типа, все подмножества данного множества, а также пустое подмножество. Так, переменная T множественного типа
var T: Set of 1..3;
может принимать восемь различных значений:
[ ] [1] [2] [3] [1,2] [1,3] [2,3] [1,2,3]
К переменным и константам множественного типа применимы операции присваивания(:=), объединения(+), пересечения(*) и вычитания(-):
['A','B'] + ['A','D'] даст ['A','B','D']
['A','D'] * ['A','B','C'] даст ['A']
['A','B','C'] - ['A','B'] даст ['C'].
Результат выполнения этих операций есть величина множественного типа.
К множественным величинам применимы операции: тождественность (=), нетождественность (<>), содержится в (<=), содержит (>=). Результат выполнения этих операций имеет логический тип, например:
['A','B'] = ['A','C'] даст FALSE
['A','B'] <> ['A','C'] даст TRUE
['B'] <= ['B','C'] даст TRUE
['C','D'] >= ['A'] даст FALSE.
Кроме этих операций для работы с величинами множественного типа в языке ПАСКАЛЬ используется операция in, проверяющая принадлежность элемента базового типа, стоящего слева от знака операции, множеству, стоящему справа от знака операции. Результат выполнения этой операции - булевский. Операция проверки принадлежности элемента множеству часто используется вместо операций отношения, например:
'A' in ['A', 'B'] даст TRUE,
2 in [1, 3, 6] даст FALSE.
IBM-совместимые компьютеры обрабатывают 256 различных символов, каждый из которых кодируется одним байтом. Соответствие символов и байтов задается ***таблицей кодировки***, в которой для каждого символа указывается соответствующий байт. Символы с кодами от 0 до 127 построены по стандарту ASCII (American Standard Code for Information Interchange — Американский стандартный код обмена информацией, читается "аски"). Вторая половина таблицы (коды 128 ... 255) в нашей стране содержит русские буквы (кириллицу) и символы псевдографики. Для того, чтобы определить по этим таблицам код того или иного символа, нужно сложить номер строки с номером столбца, в которых он расположен. Так, код цифры 5 равен 05+048 = 053.
Символьная информация в алгоритмах и программах описывается данными двух типов: *символьным* и *литерным*. Они отличаются друг от друга тем,

что значением символьной переменной является один символ, а литерной — строка символов.

Коды (кодировка ASCII) 0...127

	000	016	032	048	064	080	096	112	
00		◆		0	@	P	`	p	00
01		◆	!	1	A	Q	a	q	01
02		↕	"	2	B	R	b	r	02
03	♥	‖	#	3	C	S	c	s	03
04	◆	¶	$	4	D	T	d	t	04
05	♣	§	%	5	E	U	e	u	05
06	♠		&	6	F	V	f	v	06
07	•		'	7	G	W	g	w	07
08		↑	(	8	H	X	h	x	08
09	°	↓	)	9	I	Y	i	y	09
10		→	*	:	J	Z	j	z	10
11		←	+	;	K	[	k	{	11
12		∟	,	<	L	\	l	|	12
13		•	-	=	M	]	m	}	13
14		•	.	>	N	^	n	~	14
15	Ц	◆	/	?	O	_	o	⌂	15

Коды (модифицированный альтернативный вариант) 128...255

	128	144	160	176	192	208	224	240	
00	А	Р	а	░	└	╨	р	≡	00
01	Б	С	б	▒	┴	╤	с	±	01
02	В	Т	в	▓	┬	╒	т	≥	02
03	Г	У	г	|	├	╓	у	≤	03
04	Д	Ф	д	┤	─	╙	ф	⌠	04
05	Е	Х	е	╡	┼	╔	х	⌡	05
06	Ж	Ц	ж	╢	╞	╫	ц	÷	06
07	З	Ч	з	╖	╟	╪	ч	≈	07
08	И	Ш	и	╕	╚	┘	ш	°	08
09	Й	Щ	й	╣	╔	┌	щ	∙	09
10	К	Ъ	к	╤	╩	┌	ъ	·	10
11	Л	Ы	л	╗	╦	■	ы	√	11
12	М	Ь	м	╝	╠	▄	ь	n	12
13	Н	Э	н	╜	¥	▌	э	²	13
14	О	Ю	о	╛	╪	▐	ю	■	14
15	П	Я	п	┐	╧	▀	я		15

Типы данных, используемые для обработки символьной информации

Язык	Тип, ключевое слово	Примеры использования
	Литерный **лит**	t := "Литерная величина" s := "" (пустая строка)
Turbo Pascal	Символьный **Char**	a := ' f ' ; b := '+' ; c := '5' ; If a = ' ' then k := k + 1
	Литерный **String**	t := 'Литерная величина' ; f := ' ' ; (пустая строка)
QBasic	Литерный	t$:= "Литерная величина" f$:= "" (пустая строка)

Для данных символьного и литерного типов применимы ***операции сцепки*** (соединения, конкатенации) и ***сравнения*** (<, >, <=, >=, =, <>). Сравнивать можно строки разной длины. Сравнение осуществляется слева направо в соответствии с ASCII-кодами соответствующих символов. Так, строка "*стол*" меньше строки "*стул*", строка "*teacher*" больше строки "*pupil*" , а строка "*пар*" меньше строки "*парад*".

Функции и команды обработки строк QBasic

Функции

ASC (X$) Возвращает порядковый номер символа X$ в таблице кодов символов.

CHR$ (N) Возвращает символ с заданным порядковым номером N.

INSTR ([N ,] X$, Y$) Возвращает номер позиции строки X$, начиная с которой в ней размещается подстрока Y$. Если подстрока не найдена, то значение функции равно нулю. Поиск подстроки ведется с позиции N, а если N не задано, то с начала строки.
LEFT$ (X$, N) Возвращает подстроку, составленную из первых N символов строки X$.
LEN (X$) Возвращает количество символов в строке X$.
MID$ (X$, N [, M]) Возвращает подстроку, составленную из M символов строки X$, начиная с позиции N (если параметр M опущен, то возвращаются все символы, начиная с позиции N).
RIGHT$ (X$, N) Возвращает подстроку, составленную из последних N символов строки X$.
STR$ (N) Возвращает представление числа N в символьной форме.
VAL (X$) Возвращает представление символов строки X$ в числовой форме.
Операторы (функции)
**MID$ (X$, N , M) = Y$** Часть строки X$, начиная с позиции N, длиной M позиций заменяется на строку Y$. Длина X$ не изменяется.
**SWAP X$, Y$** Строки X$ и Y$ обмениваются своими значениями.

Пример 1. ***Определить количество слов в заданном тексте.***
Если слова в тексте разделены **одним пробелом**, то задача сводится к подсчету числа пробелов. Количество слов при этом равно числу пробелов плюс 1. Если же **число пробелов** между соседними словами **произвольное**, как обычно и бывает, то алгоритм усложняется. Рассмотрим оба варианта решения этой задачи.
Вариант 1. Слова в тексте разделены одним пробелом.
Тест

Данные	Результат
"Кот на крыше"	**N=3**

Turbo Pascal

```
Program Probel;
  Uses Crt;
  Var Text      : String;  {заданный непустой текст}
      i, Number : Integer; {Number — количество слов в тексте}
      Letter    : Char;    {текущая буква }
BEGIN ClrScr;
  WriteLn('Введите текст :'); ReadLn(Text);
  Number:=1;
  For i:=1 to Length(Text) do {цикл по буквам текста}
   begin
     Letter:=Text[i];
     If (Letter = ' ' ) then Number:=Number+1;
```

```
   end;
  WriteLn('О т в е т : количество слов в тексте равно ', Number);
END.
```

Java

```
String text;
Scanner input=new Scanner(System.in);
System.out.println("Введите текст");text=input.nextLine();
int number=1;
for (int i=0;i<text.length();i++)
{
        char letter=text.charAt(i);
        if (letter==' ') number++;
}
System.out.println("Ответ: количество слов в предложении "+number);
```

Вариант 2. Слова в тексте разделены произвольным количеством пробелов.

Тест

Данные	Результат
"Кот на крыше"	**N=3**

```
  Program KolSlov;
  Uses Crt;
  Var Text      : String;  {заданный текст}
      i, Number : Integer; {Number - количество слов в тексте}
      Flag      : Boolean;
      Letter    : Char;    {текущая буква }
BEGIN
  ClrScr;
  WriteLn('Введите текст :');
  ReadLn(Text);
  Number := 0; Flag := TRUE;
  For i := 1 to Length(Text) do {цикл по буквам текста}
    begin
      Letter := Text[i];        {текущая буква текста }
      If (Letter <> ' ') and Flag
        then Number := Number+1;
      Flag := (Letter=' ')     {(Letter=' ') — логическое выражение,}
    end;                       {принимающее значения TRUE или FALSE }
  WriteLn;
  WriteLn('О т в е т : количество слов в тексте равно ', Number); ReadLn
END.
```

QBasic

```
CLS
```

```
PRINT "Введите текст, отделяя слова пробелами."
PRINT "Если в тексте есть запятые, заключите его в кавычки."
INPUT Text$ : PRINT
Number = 0 : Flag = 0
FOR i = 1 TO LEN(Text$)              'цикл по буквам текста
   Letter$ = MID$(Text$, i, 1)       'текущая буква текста
   IF (Letter$ <> " ") AND (Flag = 0) THEN Number=Number+1
   IF (Letter$ = " ") THEN Flag = 0 ELSE Flag = 1
NEXT i
PRINT "О т в е т : количество слов в тексте равно "; Number
END
```

Пример 7.2. *Определить, является ли заданное слово "перевёртышем" (слово называется "перевёртышем", если совпадает с собой после переворачивания).*

Система тестов

N теста	Данные	Результат
1	Slovo = "казак"	Otvet = "Перевертыш"
2	Slovo = "коза"	Otvet = "Не перевертыш"

Turbo Pascal

```
Program TurnOver;
  Uses Crt;
  Var Slovo    : String;
      Dlina, i : Integer;
      Flag     : Boolean;
BEGIN
  ClrScr;
  Write('Введите слово : ');  ReadLn(Slovo);
  Dlina:= Length(Slovo);
     {Сравниваются пары букв: первая буква с последней, }
     {вторая буква с предпоследней и т.д. }
  i:=1;  Flag := TRUE;
  While (i <= Dlina/2) and Flag do       {цикл до первой несовпавшей }
    begin                                {пары букв (если такая есть)}
      Flag := (Slovo[i]=Slovo[Dlina-i+1]);
      i := i+1
    end;
  WriteLn; Write( 'О т в е т : слово ', Slovo);
  If Flag then WriteLn(' — перевертыш. ')
          else WriteLn(' — не перевертыш');
  ReadLn
END.
```

QBasic

```
CLS : INPUT "Введите слово : ", SLOVO$
Dlina = LEN(SLOVO$)
  ' Сравниваются пары букв: первая буква с последней,
  ' вторая буква с предпоследней и т.д.
i = 1 : Flag = 0
WHILE (i<=Dlina/2) AND (Flag=0) 'цикл до первой несовпавшей пары букв
  Letter1$ = MID$(SLOVO$, i, 1)          'первая буква пары
  Letter2$ = MID$(SLOVO$, Dlina-i+1, 1)  'вторая буква пары
  IF Letter1$ = Letter2$ THEN i=i+1 ELSE Flag=1
WEND
PRINT : PRINT "О т в е т : слово "; SLOVO$;
IF Flag = 0 THEN PRINT " — перевертыш." ELSE PRINT " — не перевертыш."
END
```

Пример 7.3. *В заданном тексте одно заданное слово везде заменить на другое заданное слово такой же длины.*

Тест

Данные			Результат
Текст	Слово1	Слово2	
"2sinx+siny"	"sin"	"cos"	"2cosx+cosy"

Turbo Pascal

(эта программа, использующая стандартную функцию Pos , не требует, чтобы длины заменяемого и вставляемого слов были одинаковыми)

```
Program Replace;
  Uses Crt;
  Var Text, Slovo1, Slovo2 : String;
      i, DlinaSlova, P     : Integer;
BEGIN ClrScr;
  Write('Введите строку : '); ReadLn(Text);
  Write('Какое слово заменить ? '); ReadLn(Slovo1);
  Write('На какое слово заменить ? '); ReadLn(Slovo2);
  WriteLn; WriteLn('О т в е т : ');
  WriteLn('Исходный текст: ', Text); DlinaSlova:=Length(Slovo1);
  DlinaSlova:=Length(Slovo1);
  P:=Pos(Slovo1,Text); {номер позиции, с которой в строке Text     }
                       {в первый раз встречается подстрока Slovo1  }
  While P>0 do         {цикл продолжается до тех пор,пока подстрока}
                       {Slovo1 встречается в строке Text           }
   begin
     Delete(Text, P, DlinaSlova); {удаление подстроки Slovo1, начинаю-}
                                  {щейся с позиции P, из строки Text  }
     Insert(Slovo2, Text, P);  {вставка подстроки Slovo2  }
```

```
                    { в строку Text с позиции P}
    P:=Pos(Slovo1, Text); {номер позиции, с которой подстрока Slovo1}
                  {встречается в строке Text в очередной раз}
   end;
  WriteLn('Новый текст: ', Text);
  ReadLn
END.
```

QBasic

```
CLS : INPUT "Введите текст : " , Text$
INPUT "Какое слово заменить ? " , Slovo1$
INPUT "На какое слово заменить ? " , Slovo2$
PRINT : PRINT "О т в е т"
PRINT "Исходный текст : " ; Text$
DlinaText = LEN(Text$) : DlinaSlova = LEN(Slovo1$)
FOR i = 1 TO DlinaText-DlinaSlova+1
  IF MID$(Text$, i, DlinaSlova) = Slovo1$ THEN
      MID$(Text$, i) = Slovo2$ : i=i+DlinaSlova
  END IF
NEXT i
PRINT "Новый текст : " ; Text$
END
```

Пример 7.4. *Заданную последовательность слов переупорядочить в алфавитном порядке (то есть выполнить лексикографическое упорядочение).*

Тест

Данные	Результат
Words=("стул", "гора", "яма", "стол")	Words=("гора", "стол", "стул", "яма")

Turbo Pascal

```
Program LexOrder;
  Uses Crt;
  Var Words       : Array[1..10] of String; {массив слов}
      Tmp         : String;       {Tmp — вспомогательная переменная}
      i, j, NWords : Integer;     {NWords — количество слов}
BEGIN
  ClrScr;
  Write('Количество слов в тексте — ');
  ReadLn(NWords);
  For i := 1 to NWords do
    begin Write(i, '-ое слово : ');
          ReadLn(Words[i])
    end;
  For i := 1 to NWords-1 do {лексикографическое упорядочение слов}
    For j := i+1 to NWords do
```

```
    If Words[i]>Words[j] then
      begin
        Tmp := Words[i]; Words[i]:=Words[j]; Words[j]:=Tmp
      end;
  WriteLn;  WriteLn('О т в е т');
  WriteLn('Лексикографически упорядоченный массив слов:');
  For i := 1 to NWords do Write(Words[i], ' ');
  WriteLn; ReadLn
END.
```

QBasic

```
CLS : INPUT "Количество слов в тексте — ", NWords
DIM Words(NWords) AS STRING
FOR i = 1 TO NWords
  PRINT i; "-ое слово " ; : INPUT Words(i)
NEXT i
FOR i = 1 TO NWords - 1     'лексикографическое упорядочение слов
  FOR j = i + 1 TO NWords
    IF Words(i) > Words(j) THEN SWAP Words(i), Words(j)
  NEXT j
NEXT i
PRINT : PRINT "О т в е т"
PRINT "Лексикографически упорядоченный массив слов:"
FOR i = 1 TO NWords
  PRINT Words(i); " " ;
NEXT i
PRINT
END
```

Пример 7.5. Проверить, имеется ли в линейной записи заданной математической формулы баланс открывающих и закрывающих скобок.

Система тестов

Номер теста	Проверяемый случай	Данные	Результат
1	При просмотре линейной записи слева направо первой встречается закрывающая скобка	"a)b+1("	"Нет баланса"
2	Первой встречается открывающая скобка, но число открывающих и закрывающих скобок не совпадает	"(a+b))"	"Нет баланса"
3	Есть баланс скобок	"(a+b/(c*d))"	"Есть баланс"

Turbo Pascal

```
Program Balance;
```

```
  Uses Crt;
  Var S            : String;
     Dlina, Flag, i : Integer;
BEGIN ClrScr;
  GotoXY(15, 5);
  Write('Введите линейную запись математической формулы :');
  GotoXY(32,7); ReadLn(S);
  i:=1; Flag:=0; Dlina:=Length(S);
  While (Flag>=0) and (i<=Dlina) do
    begin
      If S[i] = '(' then Flag:=Flag + 1;
      If S[i] = ')' then Flag:=Flag - 1;
      i:=i+1
    end;
  GotoXY(32, 9); WriteLn('О т в е т');
  GotoXY(15,11);
  If Flag=0 then Write('Есть баланс ') else Write('Нет баланса ');
  WriteLn('открывающих и закрывающих скобок');
  ReadLn
END.
```

QBasic

```
CLS
INPUT "Введите линейную запись математической формулы :", S$
i = 1 : Flag = 0 : Dlina = LEN(S$)
WHILE Flag >= 0 AND i <= Dlina
  IF MID$(S$, i, 1) = "(" THEN Flag = Flag + 1
  IF MID$(S$, i, 1) = ")" THEN Flag = Flag - 1
  i = i + 1
WEND
PRINT : PRINT "О т в е т"
IF Flag = 0 THEN PRINT "Есть баланс "; ELSE PRINT "Нет баланса ";
PRINT "открывающих и закрывающих скобок"
END.
```

Пример 8. Обработка символьной информации ная языке Pyton/

Текстовый файл состоит не более чем из 10^6 символов *X*, *Y* и *Z*. Определите длину самой длинной последовательности, состоящей из символов *X*.

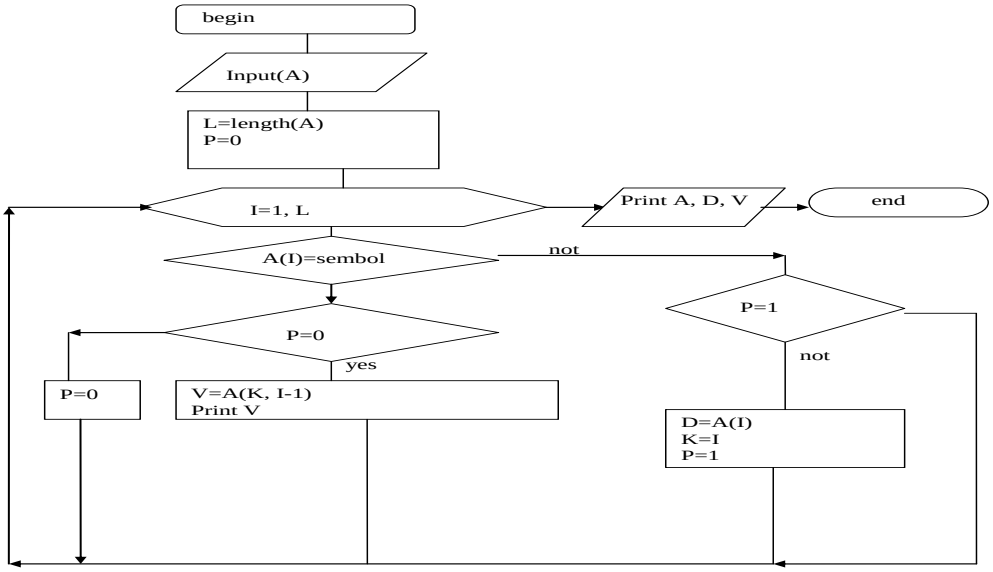
```
s=’
XXXYYXXYZXXYXZYXXXYYXXXXXXXXXYZXYXXXYXYZYYZXYXXXXXXXXXYXZ
XYYXYYYXXXYXYZYYXYXYZXYXXYYXZXXXXYXZXYXYXXXYXXZXXYYZYXXX
YYYYZ’
mx=1
cnt=1
for i in range(len(s)-1):
```

```
    if s[i]=='X' and s[i+1]=='X':
        cnt+=1
        if cnt>mx:
            mx=cnt
    else:
        cnt=1
mx = max(cnt, mx)
print(mx)
```

Порядок выполнения

Изучите теоретический материал и презентации - видио уроки (файл «Символьная информация в ТР», сайты http://excelvideo.ru/, http://book.kbsu.ru/kbrinfo/kabardinfo/2_7/2_7.html;http://www.excel-eto-prosto.ru/?s=yc2&yclid=5959032698412038227).

1. Используя функции символьных переменных написать программу, которая из слова ИНФОРМАТИКА составит слово КИНО.
2. Используя функции символьных переменных, написать программу составления из слов ТЕРРИКОН, ОПЕРА, МЕДИЦИНА слово ТРИОД.
3. Написать и отладить программу, проверяющую – является ли это слово полидромом (читается одинаково «слева - направо» и «справа – налево)?
4. Выполнить пункт 3 в электронной таблице Excel, если каждая буква слова записана в отдельной ячейки.
5. Составить программу замены первого и последнего слова местами, если количество символов в слове максимальной длины больше 4.
6. Ввести текст в электронную таблицу. Замените каждое второе слово, если его длина меньше 3 символов, на мнимое число, действительная часть которого равна количеству символов в предложении без этого слова, мнимая – количеству символов в данном слове.
7. После выполнения п.6 сделайте все символы в предложении прописными.
8. В электронной таблице определите количество совпадений пар символов в двух различных предложениях.
9. Занесите в ячейку электронной таблице любое слово. Измените формат его представления на все имеющиеся модификации. Сравните результаты.
10. Занесите в ячейку целое число меньше 255. Переведите его в символьные форматы (символ, текст, денежный, время, дата).
11. Занесите в ячейку целое число меньше 255. Переведите его в символьные форматы (символ, текст, денежный, время, дата).
12. Занесите в ячейку целое число большее 300. Переведите его в символьные форматы (символ, текст, денежный, время, дата).
13. Выполните трассировку (проверка верификации) следующего алгоритма:



14. Оформите отчет: результаты выполнения заданий и ответы на контрольные вопросы.

Контрольные вопросы:

1. Что такое символьные и текстовые переменные? Как они задаются определяются на языках высокого уровня программирования, Ассемблере, электронных таблицах?
2. Какие стандартные функции определены для символьных переменных?
3. Что напечатается в результате выполнения программы: a:=’aunkciy’; m:=legth(a)-2; writeln(m)?
4. Что напечатается в результате выполнения программы: a:=’aunkciy’; b:=’signal’; c:=pos(b,a); m:=legth(a)+2; writeln(c,m-1)?
5. Перечислите и опишите не менее 5 функций, предназначенных для обработки символьной информации в электронной таблице.
6. Перечислите и опишите не менее 5 функций, предназначенных для обработки символьной информации в Паскале.
7. Перечислите и опишите не менее 5 функций, предназначенных для обработки символьной информации в Бейсике.
8. Перечислите и опишите не менее 5 функций, предназначенных для обработки символьной информации .

Какие режимы форматирования текстовой информации имеются в электронной таблице?

10. Отображение графической информации (программно и инструментарием EXCEL и MATHCAD)

Цель работы: овладение практическими навыками отображения графической информации программными средствами и универсальными пакетами (на примере Excel Mat и MathCad.

Краткие теоретические сведения.

При обработке биомедицинской информации часто возникает задача отображения графической информации, которую можно условно разделить на статическую и динамическую. К статической информации относятся различные графики и диаграммы, к динамической – движущиеся объекты. В последнем случае используются различные мультимедийные средства.

Поскольку размеры экрана монитора и изображаемого объекта в реальности не идентичны, то на первом этапе с помощью линейных преобразований необходимо рассчитать коэффициенты масштабирования по осям ординат и абсцисс.

Графическая информация отображается как программным путем, так и с помощью инструментария универсальных программных модулей типа MathCad, MatLab, Excel и т.п. В случае разработки программ используются языки программирования высокого уровня или специализированные языки и макросы универсальных программных модулей.

Построение графических объектов в Делфи.

1) //Очистка обозначеного сектора

```
procedure TForm1.Button1Click(Sender: TObject);
var x0,x1,y0,y1:integer;
begin
x0:=strtoint(Edit1.text); x1:=strtoint(Edit2.text); y0:=strtoint(Edit3.text); y1:=strtoint(Edit4.text);
//Image1.Canvas.FillRect(Rect(0,0,305,21
//Image1.Canvas.Brush.Style:=BsClear;
 Image1.Canvas.Pen.Color:=clWhite;//Цвет контура
Image1.Canvas.Brush.Color:=clWhite;//Цвет Заливки
Image1.Canvas.rectangle(x0,y0,x1,y1);//Цвет линии
end;
```

//**Начало**

```
procedure TForm1.Button2Click(Sender: TObject);
begin
```

2) //Определение размера окна Image1 в пикселах X:(Image1.ClientWidth), Y:(Image1.ClientHeight)

```
label1.caption:='Размер окна: Высота '+inttostr(Image1.ClientHeight)+'   Ширина '+inttostr(Image
1.ClientWidth);
//
```

3) //Пример построения прямоугольника

```
Image1.Canvas.Pen.Color:=clRed;//Цвет линии
Image1.Canvas.rectangle(0,0,305,217);//Цвет линии
```

//

4**) //Пример построения прямоугольника с заккругленными углами (Roundrect)**

Image1.Canvas.Pen.Color:=clblue;//Цвет линии
Image1.Canvas.Roundrect(20,20,285,197,10,10);// Roundrect последние цифры 10,10 определяют степень закргглленности прямоугольника
//

5**) //квадрат сиреневого цвета, с зеленым обводом (свойства: Pen.Width, Pen.Color, Brush.Color)**

Image1.Canvas.Pen.Width:=3;//толщина линии контура
Image1.Canvas.Pen.Color:=clGreen;//Цвет контура линии
Image1.Canvas.Brush.Color:=clFuchsia;//Цвет Заливки
Image1.Canvas.rectangle(30,160,60,190);//Цвет линии
//

6**) //квадрат сиреневого цвета, с зеленым обводом с штриховкой (свойства: Pen.Width, Pen.Color, Brush.Color)**

Image1.Canvas.Pen.Width:=3;//толщина линии контура
Image1.Canvas.Pen.Color:=clGreen;//Цвет контура линии
Image1.Canvas.Brush.Style:=bsFDiagonal;//Тип штриховки
Image1.Canvas.Brush.Color:=clFuchsia;//Цвет Заливки
Image1.Canvas.rectangle(220,140,270,190);//Цвет линии
//

7**) //Построение треугольника из линий**

Image1.Canvas.Pen.Width:=3;//толщина линии контура
Image1.Canvas.Pen.Color:=clyellow;//Цвет контура линии
Image1.Canvas.Moveto(60,160);// Установка пера на начальную точку
Image1.Canvas.LineTo(160,20);// Линия Снизу вверх
Image1.Canvas.LineTo (230,160);//Линия Сверху вниз
Image1.Canvas.LineTo(60,160);//Горизонтальная нижняя линия
//

8**) //Построение окружности или элипса**

Image1.Canvas.Pen.Color:=clBlue;//Цвет контура
Image1.Canvas.Brush.Color:=clLime;//Цвет Заливки контура
Image1.Canvas.Brush.Style:=bsSolid;//Тип штриховки
Image1.Canvas.Pen.Width:=2;//толщина линии контура
Image1.Canvas.Ellipse(120,70,190,140);// Координаты прямоугольника в котором строится окружность

9**) //Построение коричневой точки в центре окружности**

Image1.Canvas.Pen.Color:=clMaroon;//Цвет контура линии
Image1.Canvas.Pixels[153,105]:=clRed;//Установить красную точку в координатах x=150,y=110
//

10**) //Созданиен надписи около окружности (стр 261)**

Image1.Canvas.Brush.Style:=BsClear;//Установить бесцветный цвет фона
Image1.Canvas.Font.Size:=9;//Установка размера шрифта
Image1.Canvas.Font.Style:=[fsBold,fsUnderline];//установка стиля шрифтов (Жирный, подчеркнутый)
Image1.Canvas.Font.Color:=clRed;//Установка цвета шрифта (Красный)
Image1.Canvas.TextOut(155,105,'Центр окружности');//Установка текста надписи

//

Построение графических объектов на языке Java

На языке Java создание графических изображений возможно в программах с графическим интерфейсом, у которых главной действующей единицей является форма (объект класса JFrame). Этот язык позволяет получить прямой доступ к форме и использовать богатую библиотеку 2D и 3D графики для создания приложений и игр. Для простоты рассмотрим простейший примепр программы с графическим интерфейсом, в произвольной части которой рисуется красная линия:

```
import javax.swing.*;
import java.awt.*;
import java.awt.geom.*;
public class Example1 extends JFrame{
   public void paint(Graphics g) {
   super.paint(g);  // вызов метода рисования предка
   Graphics2D g2 = (Graphics2D) g;
   Line2D lin = new Line2D.Float(100, 100, 250, 260);
   g2.setColor(Color.RED);
   g2.draw(lin);
   }
   public static void main(String []args){
      Example1 s=new Example1();
      s.setVisible(true);
      s.setSize(400,400);
   }
         }
```

Как видно из представленного примера, основной действующей единицей программы является класс Example1, наследуемый от стандартного класса формы JFrame. В главной программе (main) создается объект этого класса и задаются начальные размеры формы 400X400. Рисование происходит в методе paint(), в котором создается объект g2 класса Graphics2D, который позволяет отображать двумерные объекты. Далее, создается объект lin с координатами концов 100,100 и 250, 260, который и отображается на форме

1) //Очистка обозначеного сектора

Осуществляется вызовом метода clearRect с координатами прямоугольной зоны, которая подлежит очистке. Сказанное иллюстрирует пример:

```
g2.clearRect(0, 0, 400, 400);
```

2) //Определение размера окна в пикселах:

Так как рисование производится на форме, то внутри класса формы доступ к размерам окна можно получить через указатель на форму (this) и вызовом метода getHeight() и getWidth().

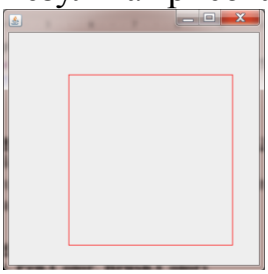
```
int h=this.getHeight();
int w=this.getWidth();
```

3) //Пример построения прямоугольника

```
public void paint(Graphics g) {
```

```
super.paint(g);  // Вызов метода рисования предка
Graphics2D g2 = (Graphics2D) g;
Rectangle2D lin = new Rectangle2D.Float(100, 100, 250, 260);
g2.setColor(Color.RED); // Задать красный цвет
g2.draw(lin);
```

Результат рисования представлен ниже



4) //Пример построения прямоугольника с заккругленными углами (Roundrect)

```
Graphics2D g2 = (Graphics2D) g;
RoundRectangle2D lin = new RoundRectangle2D.Float(100, 100, 250, 260, 40, 40);
g2.setColor(Color.RED);//Цвет линии
g2.draw(lin);//Зарисовка
```

Roundrect последние цифры 40,40 определяют размеры дуги, которая скругляет стороны прямоугольника

5) //закрашенный квадрат синего цвета

```
Graphics g2 = g;
g2.setColor(Color.BLUE);//Цвет квадрата
g2.fillRect(100, 100, 250, 260);
```

7) //Построение треугольника используя полигон

```
    g2.setColor(Color.BLUE);//Цвет линии
    int[] x={200,100,300};//создать массив x точек
    int[] y={50,300,300};//создать массив y точек
    g.fillPolygon(x,y,3); //рисовать полигон
```

8) //Построение окружности или элипса

```
 g2.setColor(Color.BLUE);//Цвет эллипса
 g.fillOval(150, 150, 90, 150);//рисовать эллипс, центр в точке 150, 150, ширина 90 пикселей, высота 150
```

Построение графических движущих объектов на Паскале

Очень часто хочется "оживить" картинку, "заставить" что-нибудь двигаться. Как сделать это, не используя анимационные программы средствами модуля Graph? Движение простых объектов может быть имитировано с помощью некоторых не очень сложных приемов.

I способ. Имитация движения объекта на экране за счет многократного выполнения программой набора действий: нарисовать – пауза – стереть (нарисовать в том же месте цветом фона) – изменить координаты положения рисунка.

Перед началом составления программы надо продумать описание «двигающегося» объекта, характер изменения координат, определяющих текущее положение объекта, диапазон изменения и шаг.

II способ. Иллюзия движения создается при помощи специальных процедур и функций.

Функция ImageSize(x1, y1, x2, y2: integer) возвращает размер памяти в байтах, необходимый для размещения прямоугольного фрагмента изображения, где x1,y1 – координаты левого верхнего и x2, y2 – правого нижнего углов фрагмента изображения.

Процедура GetImage(x1, y1, x2, y2:integer, var Buf) помещает в память копию прямоугольного фрагмента изображения, где x1, ..,y2 – координаты углов фрагмента изображения, Buf – специальная переменная, куда будет помещена копия видеопамяти с фрагментом изображения. Buf должна быть не меньше значения, возвращаемого функцией ImageSize с теми же координатами.

Процедура PutImage(x1, y1, x2, y2:integer, var Buf, Mode:word) выводит в заданное место экрана копию фрагмента изображения, ранее помещенную в память процедурой GetImage. X, Y – координаты левого верхнего угла того места на экране, куда будет скопирован фрагмент изображения; Buf – специальная переменная, откуда берется изображение, Mode – способ копирования. Координаты правого нижнего угла не указываются, так как они полностью определяются размерами выводимой на экран копии изображения. Координаты левого верхнего угла могут быть любыми, лишь бы только копия уместилась в пределах экрана (если копия не размещается на экране, то она не выводится, и экран остается без изменений). Параметр Mode определяет способ взаимодействия размещаемой с уже имеющимся на экране изображением

Примеры

1 Разработка программы, реализующей движение по траектории графического объекта.

Исходные данные:

x,y – координаты стартовой точки, тип целый.

d,t – переменные для инициализации графического режима

Промежуточные данные:

a,b – переменные для построения линии моря, тип целый

y0 – координата для высоты положения линии моря, тип целый

Использование модулей:

1. *crt* включает в себя процедуры очистки (clrscr) и задержки экрана(readkey);
2. *graph* позволяет провести инициализация графического режима с помощью процедуры InitGraph; включает в себя процедуры и функции, позволяющие вырисовывать графические объекты и применять к ним различные типы, стили и цвета оформления.

Алгоритмическая структура:

1. цикл прямого пересчета *for… to… do;*
2. цикл с предусловием *while … do*

Алгоритм программы:

1. Задание имени программы
2. Открытие модулей
3. Написание процедуры «море» с использованием цикла прямого пересчета
4. Инициализация графического режима
5. В цикле с предусловием произвести рисование волны и задание движения объекта по траектории
6. Рисование графического объекта
7. Задержка выполнения программы
8. Установка шага движения
9. Закрытие графического режима

Листинг программы:

```
program corablik;
uses Graph, Crt;
var  d,t,x,y,y0,a,b: integer
procedure more(a,b:integer);
begin
  moveto(0,y0);   setcolor(blue);
  for a:=0 to 680 do    begin
    b:=y0-round(sin(a*pi/180)*30;      lineto(a,b);
                              end;
end;
Begin
  d := Detect;  t:=2;   InitGraph(d, t, '');
  y0 := 250;     x:=600;
    while x>=0 do
     begin
      cleardevice;
      more(a,b);       setcolor(white);
      y:=y0-40-round(sin(x*pi/180)*30);
      MoveTo(x - 60, y + 40);       LineTo(x - 40, y + 60);
      LineTo(x + 40, y + 60);       LineTo(x + 60, y + 40);
      LineTo(x - 60, y + 40);
      MoveTo(x + 35, y + 40);       LineTo(x + 35, y - 60);
      LineTo(x - 40, y + 40);       LineTo(x + 35, y + 40);
      delay(2500);
      x:=x-2;{шаг движения}
    end;
 CloseGraph;
end.
```

2 Разработка программы, реализующей перемещение по экрану окружности

Исходные данные:

x,y – начальные координаты центра окружности, тип целый.

r – радиус окружности, тип целый

d,t – переменные для инициализации графического режима
Промежуточные данные:
dx – величина перемещения по оси X, тип целый
dy – величина перемещения по оси Y, тип целый
Использование модулей:
1. *crt* включает в себя процедуры очистки (clrscr) и задержки экрана(readkey);
2. *graph* позволяет провести инициализация графического режима с помощью процедуры InitGraph; включает в себя процедуры и функции, позволяющие вырисовывать графические объекты и применять к ним различные типы, стили и цвета оформления.

Алгоритмическая структура:
1. цикл с постусловием *repeat … until*
2. условный оператор *if… then…[else]*

Алгоритм программы:
1. Задание имени программы
2. Открытие модулей
3. Инициализация графического режима
4. Рисование рамки вокруг экрана
5. Рисование окружности белого цвета
6. С помощью условного оператора If указывается смена направления движения при достижении края экрана и включение звукового экрана
7. Задержка выполнения программы
8. Рисование черной окружности
9. Расчет новых координат
10. Закрытие графического режима

Листинг программы

```
Program Multik;
Uuses Graph, Crt;
Var x, y, dy, dx, r, d, : integer;
Begin
 d :=detect; t:=2;
 Initgraph(d,t,");
 Rectangle(0,0,GetMaxX,GetMaxY);
 x:=100; y:=100;      dx:=10;          dy:=10;          r:=15 ;
Repeat
  SetColor(15);        Circle(x,y,r);
  if y>=GetMaxY-radius then
     begin dy:=-10; Sound(2000); end;
  if y<=radius         the     begin dy:= 10; Sound(3000); end;
  if x>=GetMaxX-radius the     begin dx:=-10; Sound(5000); end;
  if x<=radius         then         begin dx:= 10; Sound(4000); end;
  Delay(1000);
  NoSound;
  SetColor(0);
 Circle(x,y,r);
```

```
 x:=x+dx; y:=y+dy;
Until KeyPressed;
CloseGraph;              End.
```

Использование графического режима на Бейсике

Базовые понятия работы в графическом режиме на языке высокого уровня Бейсик представлены в презентации на сайте http://festival.1september.ru/articles/603216/.

Монитор может работать в нескольких режимах, которые отличаются друг от друга разрешающей способностью (т. е. количеством точек по горизонтали и вертикали), и также количеством различных цветов. Существует много различных графических режимов, но для рисования картинок в Basic наиболее употребительны следующие: 320x200 4 CGA,EGA,VG 2 640x200 2 ACGA.EGA.VG 7 320x200 16 EGA, VGA 8 640x200 16 EGA.VGA 9 640x350 16 EGA, VGA 12 640x480 16 VGA 13* 320x200 256 VGA.

Для установки нужного графического режима в программе надо вначале написать инструкцию SCREEN, без нее рисовать будет нельзя. В простейшем случае она выглядит так: SCREEN <режим> - например, SCREEN 12.

Каждая точка экрана имеет свои координаты. Эти координаты измеряются от левого верхнего угла экрана (точка (0, 0)) вправо по горизонтали (по X) и вниз по вертикали (по Y). Большинство графических инструкций требует задания координат отдельных точек экрана и цвета рисования. Цвет рисования закодирован целыми числами от 0 до 15 следующим образом: 0 — черный (цвет фона), 8 — темно-серый, 1 — синий, 9 — голубой, 2 — темно-зеленый, 10 — ярко-зеленый, 3 — салатовый, 11 — ярко-салатовый, 4 — темно-красный 12 — ярко-красный, 5 — темно-вишневый, 13 — ярко-вишневый, 6 — коричневый, 14 — желтый, 7 — светло-серый, 15 — белый. Коды темного и яркого вариантов одного и того же цвета отличаются на 8.

Рассмотрим простейшие инструкции рисования.

PSET (X,Y) [,цвет] — рисуется одна точка в заданной позиции экрана (x,y). Если параметр цвет задан, то точка имеет этот цвет, иначе она будет белой.

CIRCLE (X,У),радиус [,цвет] — рисуется окружность с центром в точке экрана с координатами (X,Y), с заданным радиусом и цветом. Если цвет не задан, то окружность будет белой.

LINE (X1,Y1)—(X2,Y2) [.цвет] — рисуется прямая линия из точки с координатами (X1,Y1) в точку с координатами (X2,Y2). Если цвет не задан, то линия будет белой.

LINE (X1,Y1)—(X2,Y2) [,цвет] ,B — рисуется прямоугольник заданного цвета. Точки (XI,Y1) и (X2,Y2) задают две любые противоположные вершины этого прямоугольника.

LINE (X1,Y1)—(X2,Y2) [,цвет] ,BF — рисуется прямоугольник, закрашенный заданным цветом. Точки (XI,Y1) и (X2,Y2) определяют две любые

противоположные вершины этого прямоугольника. Там, где в этих инструкциях стоят X, Y, XI, X2, Y1, Y2, цвет, радиус, могут быть записаны любые арифметические выражения, но буквы B и BF являются составной частью инструкции LINE и должны быть записаны именно так.

PAINT (X,У), цвет_закрашивания, цвет_границы эта инструкция закрашивает область экрана, ограниченную линиями заданного цвета, точка (X,Y) должна быть внутри этой области. При использовании этой инструкции возможны следующие три распространенные ошибки.

1. Точка (X,Y) находится не внутри, а вне требуемой области. Тогда будет закрашена внешняя по отношению к данной области часть экрана.
2. Закрашиваемая область не ограничена сплошной линией одного цвета или в инструкции PAINT неверно указан цвет границы. Тогда цвет «вырвется наружу» и закрасит весь экран.
3. Точка (X,Y) попадает на место экрана, имеющее цвет границы. Тогда закрашивание сразу прекратится.

Пример использования инструкций.

```
SCREEN 9
поставим несколько точек разного цвета
PSET (20,20), 12 PSET (30,20), 14 PSET (25,30), 11
нарисуем две окружности
CIRCLE (200,100), 50 ,1 CIRCLE (300,100), 70 ,5 REM
Нарисуем треугольник из трех линий
LINE (200,200)—(250,200), 10 LINE (250,200)—(225,250), 10 LINE (200,200)—(225,250), 10
нарисуем простой и закрашенный прямоугольники
LINE (20,200)—(70,250), 6, B LINE (20,270)—(70,300), 9, BF
закрасим нарисованный ранее треугольник синим цветом PAINT (225,220), 1, 10
```

Графика в Excel.

Выделяем ячейки с данными , которые надо разместить на диаграмме. Активизируется инструмент **Мастер диаграмм** (или «Вставка», «Диаграммы» на панели инструментов «Стандартная». Внимательно просматриваем все закладки каждого открывающиеся окна мастера создания диаграмм и выполняем необходимые для реализации цели функции.

Графика в MathCad.

Для построения графиков в Mathcad можно воспользоваться функцией **Вставка > График > Тип графика** или панелью инструментов **График**. Поддерживаются следующие типы графиков: двумерный ("X-Y график"); в полярных координатах ("Полярный график"); линии уровня ("Контурный график"); столбчатая диаграмма ("3D панели"); поверхность ("Поверхностный график"); векторный ("Векторное поле").

Анимация

С помощью анимации пользователь может сделать любую скучную презентацию Power Point более привлекательной, интересной и динамичной. Анимация - это визуальный или звуковой эффект, который может быть добавлен к тексту или графическому объекту слайда (текст или изображение в движении, выделение цветом, изменение шрифта и т.д.). Правильно используя анимацию Power Point пользователь может быть уверен, что информация будет лучше запоминаться. Самые распространенные такие эффекты: -Вход и выход текста; -Изменение глубина текста; -Добавление различных звуков.

Анимацию пользователь может добавить к отдельным объектам слайда, к самим страницам слайда или к группе слайдов. В презентации Power Point объекты могут появляться в определенном, заданном порядке, меняться во время демонстрации слайдов, а по окончанию анимации - исчезать. Еще можно указывать время показа для каждого объекта отдельно.

К каждому элементу презентации Power Point пользователь может установить несколько эффектов одновременно. Анимацию можно задавать как в режиме сортировки слайдов, так и в обычном режиме. Еще существуют схемы анимации - это последовательность уже готовых эффектов в Power Point, которые облегчают работу пользователя с презентацией.

Анимация создана для того, чтобы акцентировать внимание человека на конкретных элементах, но если ее очень много, то это отвлекает и попросту раздражает. Цель презентации - донести определенную информацию слушателю, а звук или анимация не должны отвлекать от самого доклада.

Коллекция анимаций находится на различных сайтах. Например, http://prezentacii.com/animacii-dlya-prezentaciy.html

Графика в Pyton

```
from turtle import * # Подключим модуль черепашка
color('black','red') # устанавливаем цвет пера и цвет заливки
speed(100)
lt(90)
k = 100 # коэффициент для настраивания более удобного масштаба
begin_fill()
for i in range(12): #указываем число циклов необходимое до полного завершения фигуры
    rt(60)
    fd(2*k)
    rt(60)
    fd(2*k)
    rt(270)
end_fill()
cnt = 0
canvas = getcanvas()
for x in range(-100*k,100*k,k):
    for y in range(-100*k,100*k,k):
```

```
        s = canvas.find_overlapping(x,y,x,y)
        if len(s) == 1 and s[0] == 5:
            cnt+=1
print(cnt)
done()
exit()
```

Порядок выполнения работы.

1. Написать и отладить программу на языке высокого уровня, строящегося на экране объекта образа согласно одного из вариантов: 1 - пятиконечная звезда, 2 -усеченная призма, 3 - ряд окружностей при изменениях диаметра по геометрическому закону, 4 -ряд прямоугольных треугольников с изменяющимся по гармоническому закону периметром, 5 -лесенка, 6 - ступенчатая диаграмма (задается шаг и площадь прямоугольных элементов диаграммы).
2. Написать и отладить программу на языке высокого уровня построения движения окружности по фигуре Лиссажу с изменением цвета окружности по гармоническому закону (задаются: амплитуды, частоты и фазы изменения координат по гармоническому закону).
3. Написать и отладить программу отображения на экране параболы и касательной к ней в точке с заданными координатами.
4. Выполнить пункт 2 с выводом символьной информации на изображение (наименование осей, название графика, текущих координат движения шарика.
5. Выполнить п.2 (без движения – отображается только график движения - и изменения в цвете) и п.3 в Excel . Отобразить диаграммы заданного участка координат движения.
6. Выполнить п.2 (без движения – отображается только график движения - и изменения в цвете) п.3 в MathCad. Отобразить трехмерный график фигуры Лиссажу (третья координата изменяется по гармоническому закону).
7. В Excel в одном из столбцов значения любой гармонической функции, во втором столбце – производной данной функции, в третьем – производной данной функции полученных с помощью численных методов, в третьем – относительную разницу значений второго и третьего столбцов. Отобразить на одном листе три графика одинакового размера: функция, производные функции, разница значений. Сделать выводы.
8. Отобразить с помощью мультимедийных средств возникновение экстрасистолы.
9. Оформить отчет: описание программ (и алгоритмов к ним), скрин-шоты выполненных пунктов, краткое описание выполненных действий, выводы и краткие ответы на контрольные вопросы.

Примечания: п.8 – является заданием дополнительной сложности (за его качественное выполнение на экзамене (зачете) прибавляется до 6 баллов); вместо фигуры Лиссажу может быть предложена функция, отображающая движение точки по любой параметрически заданной кривой с изменяющимися

во времени координатами или спирали; описание алгоритмов привести в словесной и графической формах, в случае последнего допускается «ручной», а не печатный вариант.

Контрольные вопросы.

1. Каким образом отображается символьная информация в графическом режиме на языках высокого уровня?
2. Каким образом задаются цвета в графическом и символьном режимах на языке высокого уровня?
3. Как вычисляются коэффициенты масштабирования (подобия) ?
4. Сколько и какие фигуры будут выведены на экран в результате выполнения программы: «for i:=1 to 150 do begin x:=random(640); y:=random(480); setcolor(random(15+i); Dec(i); end.»?
5. Какие формы диаграмм могут быть отображены в Excel?
6. Какие формы графиков могут быть отображены в MathCad?
7. Какие инструментальные средства используются для отображения динамических графических объектов? (Привести наименования и перечень базовых возможностей)

11 . Работа с файловыми структурами

Цель работы: овладение практическими работы с файловыми структурами на языках высокого уровня.

Краткие теоретические сведения.

Информация на внешних носителях хранится в виде файлов. Работа с файлами является очень важным видом работы на компьютере. В файлах хранится все: и программное обеспечение, и информация, необходимая для пользователя. С файлами, как с деловыми бумагами, постоянно приходится что-то делать: переписывать их с одного носителя на другой, уничтожать ненужные, создавать новые, разыскивать, переименовывать, раскладывать в том или другом порядке и пр.

Файл - это информация, хранящаяся на внешнем носителе и объединенная общим именем.

Для прояснения смысла этого понятия удобно воспользоваться следующей аналогией: сам носитель информации (диск) подобен книге. Мы говорили о том, что книга - это внешняя память человека, а магнитный диск - внешняя память компьютера. Книга состоит из глав (рассказов, разделов), каждый из которых имеет название. Также и файлы имеют свои названия. Их называют именами файлов. В начале или в конце книги обычно присутствует оглавление - список названий глав. На диске тоже есть такой список-каталог, содержащий имена хранимых файлов.

Каталог можно вывести на экран, чтобы узнать, есть ли на данном диске нужный файл.

В каждом файле хранится отдельный информационный объект: документ, статья, числовой массив, программа и пр. Заключенная в файле информация становится активной, т. е. может быть обработана компьютером, только после того, как она будет загружена в оперативную память.

Любому пользователю, работающему на компьютере, приходится иметь дело с файлами. Даже для того, чтобы поиграть в компьютерную игру, нужно узнать, в каком файле хранится ее программа, суметь отыскать этот файл и инициализировать работу программы.

Работа с файлами на компьютере производится с помощью файловой системы.

Файловая система - это функциональная часть ОС, обеспечивающая выполнение операций над файлами.

Чтобы найти нужный файл, пользователю должно быть известно: а) какое имя у файла; б) где хранится файл .

Имя файла

Практически во всех операционных системах имя файла составляется из двух частей, разделенных точкой. Например:

myprog.pas

Слева от точки находится собственно имя файла (ту-prog). Следующая за точкой часть имени называется расширением файла (pas). Обычно в именах

файлов употребляются латинские буквы и цифры. В большинстве ОС максимальная длина расширения - 3 символа. Кроме того, имя файла может и не иметь расширения. В операционной системе Windows в именах файлов допускается использование русских букв; максимальная длина имени - 255 символов.

Расширение указывает, какого рода информация хранится в данном файле. Например, расширение txt обычно обозначает текстовый файл (содержит текст); расширение pcx - графический файл (содержит рисунок), zip или гаг - архивный файл (содержит архив - сжатую информацию), pas - программу на языке Паскаль.

Файлы, содержащие выполнимые компьютерные программы, имеют расширения exe или com. Например, программа популярной игры "Тетрис" хранится в файле tetris.exe. Инициализация программы происходит путем записи ее в оперативную память и перехода работы процессора к ее исполнению.

Логические диски

На одном компьютере может быть несколько дисководов - устройств работы с дисками. Каждому дисководу присваивается однобуквенное имя (после которого ставится двоеточие), например A:, B:, C:. Часто на персональных компьютерах диск большой емкости, встроенный в системный блок (его называют жестким диском), делят на разделы. Каждый из таких разделов называется логическим диском, и ему присваивается имя C:, D:, E: и т. д. Имена A: и B: обычно относятся к сменным дискам малого объема - гибким дискам (дискетам). Их тоже можно рассматривать как имена дисков, только логических, каждый из которых полностью занимает реальный (физический) диск. Следовательно, A:, B:, C:, D: - это всё имена логических дисков.

Имя логического диска, содержащего файл, является первой "координатой", определяющей место расположения файла.

Файловая структура диска

Вся совокупность файлов на диске и взаимосвязей между ними называется **файловой структурой**. Различные ОС могут поддерживать разные организации файловых структур. Существуют две разновидности файловых структур: простая, или одноуровневая, и иерархическая - многоуровневая.

Одноуровневая файловая структура - это простая последовательность файлов. Для отыскания файла на диске достаточно указать лишь имя файла. Например, если файл tetris.exe находится на диске A:, то его "полный адрес" выглядит так:

A:\tetris.exe

Операционные системы с одноуровневой файловой структурой используются на простейших учебных компьютерах, оснащенных только гибкими дисками.

Многоуровневая файловая структура - древовидный (иерархический) способ организации файлов на диске. Для облегчения понимания этого вопроса воспользуемся аналогией с традиционным "бумажным" способом хранения информации. В такой аналогии файл представляется как некоторый

озаглавленный документ (текст, рисунок) на бумажных листах. Следующий по величине элемент файловой структуры называется **каталогом**. Продолжая "бумажную" аналогию, каталог будем представлять как папку, в которую можно вложить множество документов, т. е. файлов. Каталог также получает собственное имя (представьте, что оно написано на обложке папки).

Каталог сам может входить в состав другого, внешнего по отношению к нему каталога. Это аналогично тому, как папка вкладывается в другую папку большего размера. Таким образом, каждый каталог может содержать внутри себя множество файлов и вложенных каталогов (их называют подкаталогами). Каталог самого верхнего уровня, который не вложен ни в какой другой каталог, называется корневым каталогом.

В операционной системе Windows для обозначения понятия "каталог" используется термин "папка".

Графическое изображение иерархической файловой структуры называется деревом.

На рис. 1 имена каталогов записаны прописными буквами, а файлов - строчными. Здесь в корневом каталоге имеются две папки: IVANOV и PETROV и один файл fin.com. Папка IVANOV содержит в себе две вложенные папки PROGS и DATA. Папка DATA - пустая; в папке PROGS имеются три файла и т. д. На дереве корневой каталог обычно изображается символом \.

Путь к файлу

А теперь представьте, что вам нужно найти определенный документ. Для этого надо знать ящик, в котором он находится, а также "путь" к документу внутри ящика: всю последовательность папок, которые нужно открыть, чтобы добраться до искомых бумаг.

Второй координатой, определяющей место положения файла, является **путь к файлу на диске**. Путь к файлу - это последовательность, состоящая из имен каталогов, начиная от корневого и заканчивая тем, в котором непосредственно хранится файл.

Вот всем знакомая сказочная аналогия понятия "путь к файлу": "На дубе висит сундук, в сундуке - заяц, в зайце - утка, в утке - яйцо, в яйце - игла, на конце которой смерть Кощеева".

Последовательно записанные имя логического диска, путь к файлу и имя файла составляют **полное имя файла.**

Если представленная на рис. 1 файловая структура хранится на диске C:, то полные имена некоторых входящих в нее файлов в символике операционных систем MS-DOS и Windows выглядят так:

C:\fin.com
C:\IVANOV\PROGS\progl.pas
C:\PETROV\DATA\task.dat

Таблица размещения файлов

Сведения о файловой структуре Диска содержатся на этом же диске в виде таблицы размещения файлов. Используя файловую систему ОС, пользователь

может последовательно просматривать на экране содержимое каталогов (папок), продвигаясь по дереву файловой структуры вниз или вверх.
На рис. 2 показан пример отображения на экране компьютера дерева каталогов на логическом диске E: (левое окно).
В правом окне представлено содержимое папки ARCON. ")то множество файлов различных типов. Отсюда, например, понятно, что полное имя первого в списке файла следующее:
E:\GAME\GAMES\ARCON\dos4gw.exe
Из таблицы можно получить дополнительную информацию о файлах. Например, файл dos4gw.exe имеет размер 254 556 байтов и был создан 31 мая 1994 года в 2 часа 00 мин.
Найдя в таком списке запись о нужном файле, применяя команды ОС, пользователь может выполнить с ним различные действия: инициализировать программу, содержащуюся В файле; удалить, переименовать, скопировать файл. Выполнять все эти операции вы научитесь на практическом занятии.

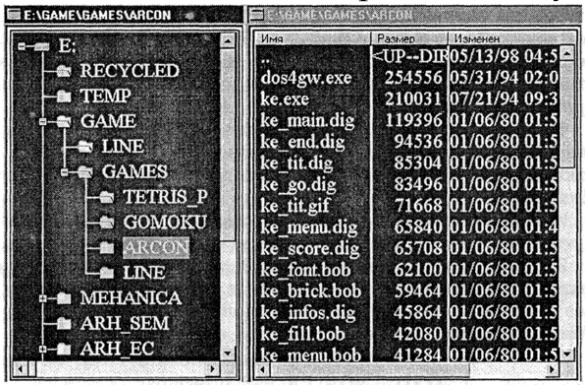


Рисунок 2. Дерево каталогов

Пользовательский интерфейс
Дружественный пользовательский интерфейс
А теперь познакомьтесь с новым для вас понятием "пользовательский интерфейс".
Разработчики современного программного обеспечения стараются сделать работу пользователя за компьютером удобной, простой, наглядной. Потребительские качества любой программы во многом определяются удобством ее взаимодействия с пользователем.
Форму взаимодействия программы с пользователем называют **пользовательским интерфейсом**. Удобная для пользователя форма взаимодействия называется дружественным пользовательским интерфейсом.
Объектно-ориентированный интерфейс
Интерфейс современных системных и прикладных программ носит название объектно-ориентированного интерфейса. Примером операционной системы, в которой реализован объектно-ориентированный подход, является Windows.
Операционная система работает с множеством объектов, к числу которых относятся: документы, программы, дисководы, принтеры и другие объекты, с

которыми мы имеем дело, работая в операционной системе.

Документы содержат некоторую информацию: текст, звук, картинки и т. д. Программы используются для обработки документов. Отдельные программы и документы неразрывно связаны между собой: текстовый редактор работает с текстовыми документами, графический редактор - с фотографиями и иллюстрациями, программа обработки звука позволяет записывать, исправлять и прослушивать звуковые файлы.

Документы и программы - это информационные объекты. А такие объекты, как дисководы и принтеры, являются аппаратными (физическими) объектами. С объектом операционная система связывает:

имя;

графическое обозначение;

свойства;

поведение.

В интерфейсе операционной системы для обозначения документов, программ, устройств используются значки (их еще называют пиктограммами, иконками) и имена. Имя и значок дают возможность легко отличить один объект от другого. С каждым объектом связан определенный набор свойств и множество действий, которые могут быть выполнены над объектом.

Например, свойствами документа являются его местоположение в файловой структуре и размер. Действия над документом: открыть (просмотреть или прослушать), переименовать, напечатать, скопировать, сохранить, удалить и др.

Контекстное меню

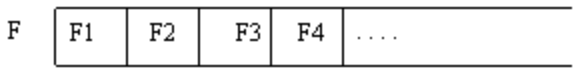
Операционная система обеспечивает одинаковый пользовательский интерфейс при работе с разными объектами. В операционной системе Windows для знакомства со свойствами объекта и возможными над ним действиями используется контекстное меню (рис. 4) (для вызова контекстного меню следует выделить значок объекта и щелкнуть правой кнопкой мыши).

Меню - это выводимый на экран список, из которого пользователь может выбирать нужный ему элемент.

Выбор нужного пункта меню производится с помощью клавиш управления курсором или манипулятора (например, мыши). Если выбрать пункт меню "Свойства", то на экран будет выведен список свойств данного объекта.

Работа с файлами на языке высокого уровня (на примере Pascal-Паскаль)

Существенной особенностью всех рассмотренных до сих пор значений производных типов является наличие в них конечного, наперед заданного числа компонент. Так, в значении многомерного массива это число можно определить, зная количество компонент по каждому измерению, а в значении записи это число определяется количеством и типом полей. Таким образом, заранее, еще до выполнения программы, по этому описанию можно выделить необходимый объем памяти машины для хранения значений переменных этих типов. Но существует определенный класс задач и определенные ситуации, когда количество компонент (пусть даже одного и того же из известных уже типов) заранее определить невозможно, оно выясняется только в процессе решения задачи. Поэтому возникает необходимость в специальном типе значений, которые представляют собой произвольные последовательности элементов одного и того же типа, причем длина этих последовательностей заранее не определяется, а конкретизируется в процессе выполнения программы. Этот тип значений получил название **файлового типа**. Условно **файл в Паскале** можно изобразить как некоторую ленту, у которой есть начало, а конец не фиксируется. Элементы файла записываются на эту ленту последовательно друг за другом:



где F – имя файла, а F1, F2, F3, F4 – его элементы. Файл во многом напоминает магнитную ленту, начало которой заполнено записями, а конец пока свободен. В программировании существует несколько разновидностей файлов, отличающихся методом доступа к его компонентам: **файлы последовательного доступа** и **файлы произвольного доступа.**

Простейший метод доступа состоит в том, что по файлу можно двигаться только последовательно, начиная с первого его элемента, и, кроме этого, всегда существует возможность начать просмотр файла с его начала. Таким образом,

чтобы добраться до пятого элемента файла, необходимо, начав с первого элемента, пройти через предыдущие четыре. Такие файлы называют **файлами последовательного доступа**. У последовательного файла доступен всегда лишь очередной элемент. Если в процессе решения задачи необходим какой-либо из предыдущих элементов, то необходимо вернуться в начало файла и последовательно пройти все его элементы до нужного.

Файлы произвольного доступа Паскаля позволяют вызывать компоненты в любом порядке по их номеру. Важной особенностью файлов является то, что данные, содержащиеся в файле, переносятся на внешние носители. Файловый тип Паскаля – это единственный тип значений, посредством которого данные, обрабатываемые программой, могут быть получены извне, а результаты могут быть переданы во внешний мир. Это единственный тип значений, который связывает программу с внешними устройствами ЭВМ.

Любой файл имеет три характерные особенности. Во-первых, у него есть имя, что дает возможность программе работать одновременно с несколькими файлами. Во-вторых, он содержит компоненты одного типа. Типом компонентов может быть любой тип Паскаля, кроме файлов. Иными словами, нельзя создать «файл файлов». В-третьих, длина вновь создаваемого файла никак не оговаривается при его объявлении и ограничивается только емкостью устройств внешней памяти.

Файловый тип или переменную файлового типа в Паскале можно задать одним из трех способов:

Type <имя_ф_типа>=file of<тип_элементов>;
<имя_ф_типа>=text;
<имя_ф_типа>=file;

Здесь <имя_ф_типа> – имя файлового типа (правильный идентификатор); File, of – зарезервированные слова (файл, из); <тип_элементов> – любой тип Паскаля, кроме файлов.

Пример описания файлового типа в Паскале

```
Type
  Product= record
    Name: string;
    Code: word;
  End;
  Text80= file of string[80];
Var
  F1: file of char;
  F2: text;
  F3: file;
  F4: Text80;
  F5: file of Product;
```

В зависимости от способа объявления можно выделить три вида файлов Паскаля:

- типизированные файлы Паскаля(задаются предложением file of..);

- текстовые файлы Паскаля(определяются типом text);
- нетипизированные файлы Паскаля(определяются типом file).

Следует помнить, что физические файлы на магнитных дисках и переменные файлового типа в программе на Паскале – объекты различные. Переменные файлового типа в Паскале могут соответствовать не только физическим файлам, но и логическим устройствам, связанным с вводом/выводом информации. Например, клавиатуре и экрану соответствуют файлы со стандартными именами Input, Output.

Как известно, каждый тип данных в Паскале, вообще говоря, определяет множество значений и множество операций над значениями этого типа. Однако над значениями файлового типа Паскаля не определены какие-либо операции, в том числе операции отношения и присваивания, так что даже такое простое действие, как присваивание значения одной файловой переменной другой файловой переменной, имеющей тот же самый тип, запрещено. Все операции могут производиться лишь с элементами (компонентами) файлов. Естественно, что множество операций над компонентами файла определяется типом компонент.

Переменные файлового типа используются в программе только в качестве параметров собственных и стандартных процедур и функций.

Основные процедуры и функции для работы с файлами:

1.До начала работы с файлами в Паскале необходимо установить связь между файловой переменной и именем физического дискового файла:

Assign(<файловая_переменная>, <имя_дискового_файла>)

Следует помнить, что имя дискового файла при необходимости должно содержать путь доступа к этому файлу, включая имя дисковода. При этом имя дискового файла – строковая величина, т.е. должна быть заключена в апострофы. Например:

Assign (chf, 'G:\Home\ Student\ Lang\ Pascal\ primer.dat').

Если путь не указан, то программа будет искать файл в своем рабочем каталоге и по указанным путям в autoexec.bat.

Вместо имени дискового файла можно указать имя логического устройства, каждое из которых имеет стандартное имя:

CON – консоль, т.е. клавиатура-дисплей;

PRN – принтер. Если к компьютеру подключено несколько принтеров, доступ к ним осуществляется по именам LPT1, LPT2, LPT3.

Не разрешается связывать с одним физическим файлом более одной файловой переменной.

2.После окончания работы с файлами на Паскале, они должны быть закрыты.

Close(<список файловых переменных>);

При выполнении этой процедуры закрываются соответствующие физические файлы и фиксируются сделанные изменения. Следует иметь в виду, что при выполнении процедуры close связь файловой переменной с именем дискового

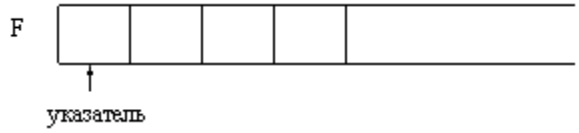
файла, установленная ранее процедурой assign, сохраняется, следовательно, файл можно повторно открыть без дополнительного использования процедуры assign.

Работа с файлами заключается, в основном, в записи элементов в файл и считывании их из файла. Для удобства описания этих процедур введем понятие указателя, который определяет позицию доступа, т.е. ту позицию файла, которая доступна для чтения (в режиме чтения), либо для записи (в режиме записи). Позиция файла, следующая за последней компонентой файла (или первая позиция пустого файла) помечается специальным маркером, который отличается от любых компонент файла. Благодаря этому маркеру определяется конец файла.

3.Подготовка к записи в файл Паскаля

Rewrite(<имя_ф_переменной>);

Процедура Rewrite(f) (где f – имя файловой переменной) устанавливает файл с именем f в начальное состояние режима записи, в результате чего указатель устанавливается на первую позицию файла. Если ранее в этот файл были записаны какие-либо элементы, то они становятся недоступными. Результат выполнения процедуры rewrite(f); выглядит следующим образом:



4.Запись в файл Паскаля

Write(<имя_ф_переменной>, <список записи>);

При выполнении процедуры write(f, x) в ту позицию, на которую показывает указатель, записывается очередная компонента, после чего указатель смещается на следующую позицию. Естественно, тип выражения x должен совпадать с типом компонент файла. Результат действия процедуры write(f, x) можно изобразить так:

Состояние файла f до выполнения процедуры

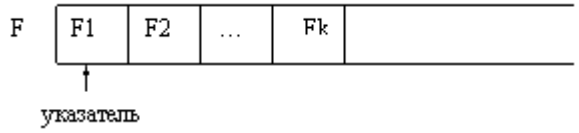
Состояние файла f после выполнения процедуры

Для типизированных файлов выполняется следующее утверждение: если в списке записи перечислено несколько выражений, то они записываются в файл, начиная с первой доступной позиции, а указатель смещается на число позиций, равное числу записываемых выражений.

5.Подготовка файла к чтению Паскаля

Reset(<имя_ф_переменной>);

Эта процедура ищет на диске уже существующий файл и переводит его в режим чтения, устанавливая указатель на первую позицию файла. Результат выполнения этой процедуры можно изобразить следующим образом:



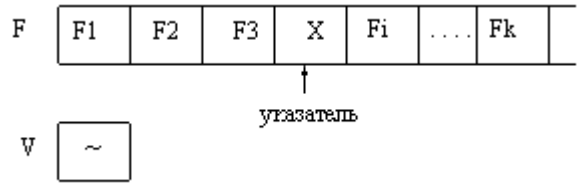
Если происходит попытка открыть для чтения не существующий еще на диске файл, то возникает ошибка ввода/вывода, и выполнение программы будет прервано.

6.Чтение из файла в Паскале

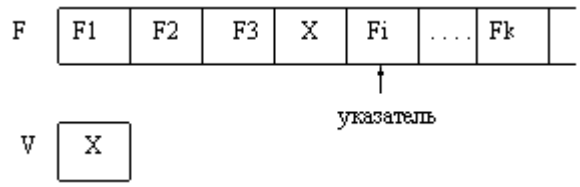
Read(<имя_ф_переменной>,<список переменных>);

Рассмотрим результат действия процедуры read(f, v):

Состояние файла f и переменной v до выполнения процедуры:



Состояние файла f и переменной v после выполнения процедуры:



Для типизированных файлов при выполнении процедуры read() последовательно считывается, начиная с текущей позиции указателя, число компонент файла, соответствующее числу переменных в списке, а указатель смещается на это число позиций.

В большинстве задач, в которых используются файлы, необходимо последовательно перебрать компоненты и произвести их обработку. В таком случае необходимо иметь возможность определять, указывает ли указатель на какую-то компоненту файла, или он уже вышел за пределы файла и указывает на маркер конца файла.

7.Функция определения достижения конца файла в Паскале

Eof(<имя_ф_переменной>);

Название этой функции является сложносокращенным словом от end of file. Значение этой функции имеет значение true, если конец файла уже достигнут, т.е. указатель стоит на позиции, следующей за последней компонентой файла. В противном случае значение функции – false.

8.Изменение имени файла в Паскале

Rename(<имя_ф_переменной>, <новое_имя_файла>);

Здесь новое_ имя_ файла – строковое выражение, содержащее новое имя файла, возможно с указанием пути доступа к нему.

Перед выполнением этой процедуры необходимо закрыть файл, если он ранее был открыт.

9.Уничтожение файла в Паскале

Erase(<имя_ф_переменной>);

Перед выполнением этой процедуры необходимо закрыть файл, если он ранее был открыт.

10.Уничтожение части файла от текущей позиции указателя до конца в Паскале

Truncate(<имя_ф_переменной>);

11.Файл Паскаля может быть открыт для добавления записей в конец файла

Append(<имя_ф_переменной>);

Типизированные файлы Паскаля. Длина любого компонента типизированного файла строго постоянна, т.к. тип компонент определяется при описании, а, следовательно, определяется объем памяти, отводимый под каждую компоненту. Это дает возможность организовать прямой доступ к каждой компоненте (т.е. доступ по порядковому номеру).

Перед первым обращением к процедурам ввода/вывода указатель файла стоит в его начале и указывает на его первый компонент с номером 0. После каждого чтения или записи указатель сдвигается к следующему компоненту файла. Переменные и выражения в списках ввода и вывода в процедурах **read() и write()** должны иметь тот же тип, что и компоненты файла Паскаля. Если этих переменных или выражений в списке несколько, то указатель будет смещаться после каждой операции обмена данными на соответствующее число позиций.

Для облегчения перемещения указателя по файлу и доступа к компонентам типизированного файла существуют специальные процедуры и функции:

fileSize(<имя_ф_переменной>) – функция Паскаля, определяющая число компонентов в файле;

filePos(<имя_ф_переменной>) – функция Паскаля, значением которой является текущая позиция указателя;

seek(<имя_ф_переменной>,n) – процедура Паскаля, смещающая указатель на компоненту файла с номером n. Так, процедура seek(<имя_ф_переменной>,0) установит указатель в начало файла, а процедура seek(<имя_ф_переменной>, FileSize(<имя_ф_переменной>)) установит указатель на признак конца файла.

Текстовые файлы Паскаля. Текстовые файлы предназначены для хранения текстовой информации. Именно в таких файлах хранятся, например, исходные тексты программ. Компоненты текстовых файлов могут иметь переменную длину, что существенно влияет на характер работы с ними. Доступ к каждой строке текстового файла Паскаля возможен лишь последовательно, начиная с первой. К текстовым файлам применимы процедуры assign, reset, rewrite, read, write и функция eof. Процедуры и функции seek, filepos, filesize к ним не применяются. При создании текстового файла в конце каждой записи (строки) ставится специальный признак EOLN(end of line – конец строки). Для

определения достижения конца строки существует одноименная логическая функция EOLN(<имя_ф_переменной>), которая принимает значение true, если конец строки достигнут.

Форма обращения к процедурам write и read для текстовых и типизированных файлов одинакова, но их использование принципиально различается.

В списке записываемых в текстовый файл элементов могут чередоваться в произвольном порядке числовые, символьные, строковые выражения. При этом строковые и символьные элементы записываются непосредственно, а числовые из машинной формы автоматически преобразуются в строку символов.

- текстовые файлы удобнее для восприятия человеком, а типизированные соответствуют машинному представлению объектов;
- текстовые файлы, как правило, длиннее типизированных;
- длина текстовых файлов зависит не только от количества записей, но и от величины переменных.

Так, в типизированном файле числа 6, 65 и 165 как целые будут представлены одним и тем же числом байт. А в текстовых файлах, после преобразования в строку, они будут иметь разную длину. Это вызывает проблемы при расшифровке текстовых файлов. Пусть в текстовый файл пишутся подряд целые числа (типа byte): 2, 12, 2, 128. Тогда в файле образуется запись 2122128. При попытке прочитать из такого файла переменную типа byte программа прочитает всю строку и выдаст сообщение об ошибке, связанной с переполнением диапазона.

Но, вообще-то, такой файл не понимает не только машина, а и человек.

Чтобы избежать этой ошибки, достаточно вставить при записи в файл после каждой переменной пробел. Тогда программа при каждом чтении берет символы от пробела до пробела и правильно преобразует текстовое представление в число.

Кроме процедур read и write при работе с текстовыми файлами используются их разновидности readln и writeln. Отличие заключается в том, что процедура writeln после записи заданного списка записывает в файл специальный маркер конца строки. Этот признак воспринимается как переход к новой строке. Процедура readln после считывания заданного списка ищет в файле следующий признак конца строки и подготавливается к чтению с начала следующей строки.

Пример решения задачи с файлами Паскаля

Пусть нам необходимо сформировать текстовый файл с помощью Паскаля, а затем переписать из данного файла во второй только те строки, которые начинаются с буквы «А» или «а».

Пояснения: нам понадобятся две файловые переменные f1 и f2, поскольку оба файла текстовые, то тип переменных будет text. Задача разбивается на два этапа: первый – формирование первого файла; второй – чтение первого файла и формирование второго.

Для завершенности решения задачи есть смысл добавить еще одну часть, которая в задаче явно не указана – вывод на экран содержимого второго файла.

Пример процедуры Assign Паскаля

```
Program primer;
Var f1,f2:text;
   I,n: integer;
   S: string;
Begin
{формируем первый файл}
   Assign(f1, ‘file1.txt’); {устанавливаем связь файловой переменной с физическим файлом на диске}
   Rewrite(f1);  {открываем файл для записи}
   Readln(n) {определим количество вводимых строк}
   for i:=1 to n do
   begin        readln(s); {вводим с клавиатуры строки}
      writeln(f1,s); {записываем последовательно строки в файл}
   end;
   close(f1); {заканчиваем работу с первым файлом, теперь на диске существует файл с именем file1.txt, содержащий введенные нами строки. На этом программу можно закончить, работу с файлом можно продолжить в другой программе, в другое время, но мы продолжим}
{часть вторая: чтение из первого файла и формирование второго}
   Reset(f1); {открываем первый файл для чтения}
   Assign(f2, ‘file2.txt’); {устанавливаем связь второй файловой переменной с физическим файлом}
   Rewrite(f2); {открываем второй файл для записи}
 {Дальше необходимо последовательно считывать строки из первого файла, проверять выполнение условия и записывать нужные строки во второй файл. Для чтения из текстового файла рекомендуется использовать цикл по условию «пока не конец файла»}
   While not eof(f1) do
   Begin
      Readln(f1,s);{считываем очередную строку из первого файла}
      If (s[1]=’A’) or (s[1]=’a’) then
      Writeln(f2,s); {записываем во второй файл строки, удовлетворяющие условию}
   End;
   Close(f1,f2); {заканчиваем работу с файлами}
 {часть третья: выводим на экран второй файл}
   Writeln;
   Writeln(‘Второй файл содержит строки:’);
   Reset(f2); {открываем второй файл для чтения}
   While not eof(f2) do {пока не конец второго файла}
   Begin
```

```
    Readln(f2,s);{считываем очередную строку из второго файла}
    Writeln(s); {выводим строку на экран}
  End;
End.
```

Работа с файлами на языке Java

Язык программирования Java представляет обширный механизм для доступа и манипуляцией файлами и элементами директорий (папок). Несмотря на огромное обилие операционных систем, данный язык представляет универсальный механизм работы с файлами для любых типов файловых систем: FAT16, FAT32, NTFS, EXT3 и.т.д. Программы на языке Java для доступа к файлам обычно используют классы, описанные в библиотеке java.io. В этом разделе мы рассмотрим толькор самые необходимые классы: File, FileReader, FileWriter, FileInputStream, FileOutputStream.

Класс "File" является полезным инструментом для получения информации о файлах и директориях компьютера. Объекты этого класса сами по себе не открывают файлы и не извлекают из них информацию, в тоже время взаимодействуя с операционной системой они позволяют извлечь необходимую для пользователя информацию

Методы класса "File"	**Описание**
boolean canRead()	Возвращает true, если из файла может производиться чтение и false в противном случае
boolean canWrite()	Возвращает true, если имеется возможность производить запись в файл и false в противном случае
boolean exists()	Возвращает true, если указанный файл существует и false в противном случае
boolean isFile()	Возвращает true, если по указанному пути находится файл и false, если это директория
boolean isDirectory()	Возвращает true, если по указанному пути находится директория и false, если это файл
boolean isAbsolute()	Returns true if the arguments specified to the File constructor indicate an absolute path to a file or directory; false otherwise.
String getName()	Возвращает имя указанного файла
String getPath()	Возвращает путь (место расположения) файла
String getParent()	Возвращает имя директории, в которой файл находится
long length()	Возвращает размер (количество байт) указанного файла
long lastModified()	Возвращает дату и время последнего изменения файла
String[] list()	Возвращает список файлов, которые находятся в указанной директории

Example 1 illustrates how to use the class File to get information about the file. New componets used in the example are the class JFileChooser that displays a dialog (known as the JFileChooser dialog) that enables the user to easily select files or

directories and the class JTextArea that allows to represent many linesa of text inside a frame.

Example 1. Test of "File" class using GUI application

```
import java.awt.*;
import java.io.File;
import javax.swing.*;

public class Example1 extends JFrame
{
  JTextArea outputArea = new JTextArea();//1
  JScrollPane scrollPane=new JScrollPane( outputArea );//2
  JFileChooser fileChooser = new JFileChooser();//3

  public Example1()
  {  this.setLayout(new BorderLayout());
    this.add( scrollPane, BorderLayout.CENTER ); //4
    this.setSize( 400, 400 ); // 5 set GUI size
    this.setVisible( true ); //6 display GUI
    fileChooser.setFileSelectionMode(//7
              JFileChooser.FILES_AND_DIRECTORIES );
    int result = fileChooser.showOpenDialog( this );//8
    if ( result == JFileChooser.CANCEL_OPTION )//9
    System.exit( 1 );//10
    File name = fileChooser.getSelectedFile(); // получить список файлов
    if ( name.exists() ) ;// Если файл существует, то вывести о нем информацию
    String text="Информация о файле:\n";
      text=text+name.getName()+" существует\n";
      text=text+"Это файл? "+name.isFile()+"\n";
      text=text+"Это директория? "+name.isDirectory()+"\n";
      text=text+"Длинна файла: "+name.length()+"\n";
      outputArea.setText(text);
      if ( name.isDirectory() ) // output directory listing
       {  String[] directory = name.list();
         outputArea.append( "\n\nСодержимое директории:\n" );
         for ( String directoryName : directory )
         outputArea.append( directoryName + "\n" );
       } // end else
    } // end outer if
   // end FileDemonstration constructor

    public static void main( String[] args )
  {
    Example1 application = new Example1();

  } // конец главной программы
} // конец класса
```

При выборе папки C:/Windows программа выводит следующую информацию:

Информация о файле:
Windows существует
Это файл? false
Это директория? true
Длинна файла: 0
Содержимое директории:
activator.exe
addins
AppCompat
AppPatch
ARJ.PIF
assembly
atiogl.xml
atisamu32.dll
ativpsrm.bin
bfsvc.exe
и еще сотни файлов

Классы **FileInputStream and FileOutputStream** предназначены для работы с бинарными файлами (файлы, представленные как массив байт), классы **FileReader and FileWriter** позволяют работать с текстовыми файлами. Класс **FileReader** позволяет читать символы из текстового файла. Данный класс содержит следующие методы
read() - вернуть один символ из файла
ready() - возвращает true, если конец файла еще не достигнут
read(char[] cbuf) - читать все символы текстового файла в массив "cbuf"
skip(long n) - пропустить n символов
close() - закрыть файл

Класс **FileWriter** предназначен для записи в текстовые файлы. Содержит следующие методы:
append (char c) - добавить символ "c" в конец текстового файла
write(char c) - записать в файл символ "c"
write(char[] cbuf,int off, int len); записать в файл "len" символов из массива cbuf, используя смещение от его начала заданное параметром "off" следующие методы:
available(); Возвращает число байт информации, которые можно прочитать из файла. При открытии файла возвращает длинну файла, в процессе чтения происходит сдвиг невидимого указателя и количество доступных байт уменьшается. Если этот параметр равен нулю, это значит, что достигнут конец файла.
read(); прочитать 1 байт информации из файла

read(byte[] b); Заполнить массив b данными из файла. Количество прочитанных байт будет равно длинне массива.
read(byte[] b, int off, int len); Прочитать "len" байт из файла и внести данные в массив "b", начиная с индекса, заданного числом "off".
skip(long n); Пропустить "n" байт, сместив указатель

Класс **FileOutputStream** записывает последовательность байт в файл. Данный класс содержит следующие методы:
close(); Закрыть файл.
flush(); Произвести запись накопленной информации в файл, но не закрывать его.
write(int b); Записать число b в файл.
write(byte[] b); Записать массив b в файл. Количество записанных байт будет равно длинне массива b.
write(byte[] b, int off, int len); Данный метод записывает "len" байт в файл, используя данные массива b, начиная с индекса off.

Пример программы, которая читает текст с клавиатуры и записывает его в файл

Порядок выполнения работы.

1. Изучите теоретический материал.
2. Создайте в текстовом редакторе таблицу. Скопируйте ее в Excel. Сделайте вывод.
3. Создайте текстовый файл в блокноте. Откройте его в Word. Сделайте вывод.
4. Создайте формулы и рисунок как объект в Word версии 10 и выше. Откройте его в версии ниже 10. Сделайте вывод.
5. Создайте рисунок в текстовом файле как объект. Откройте его в Paint. Сделайте вывод.
6. Напишите и отладьте программу, выполняющую следующие действия: осуществляется расчет значений заданной функции в определенных интервале и шагом изменения аргумента, результат запоминается в текстовом файле. Подумайте, как прочитать этот файл в Блокноте и Word.
7. Напишите программу на языке высокого уровня импортирования и экспортирования файла из электронной таблице в массив, определенном в оболочке языка высокого уровня.
8. Создайте и отладьте на языке высокого уровня программу, формирующую простейший текстовый файл и осуществляет его последовательное чтение, запись в котором состоит из переменных разных типов (например, ФИО, год рождения, номер медицинской карты, дата заполнения).
9. Научитесь выполнять основные файловые операции в используемой ОС (копирование, перемещение, удаление, переименование файлов).
9. Оформите отчет, включающий описание действий, полученных результатов, кратких ответов на не менее чем трех контрольных вопросов.

Контрольные вопросы.

1. Что называется файлом? Как задается структура файла? Как осуществляется поиск информации в файле?
2. Как осуществляется поиск файла в каталоге и на языке высокого уровня?
3. Почему файлы с разным расширением в имени несовместимы? Как можно решить эту проблему?
4. Что выполняют процедуры открытия и закрытия файла?
5. В чем заключается назначение файловых менеджеров?
6. Как можно определить наличие вируса в файле?
7. Каким образом можно защитить файл от несанкционированного доступа?
8. Что такое типизированный и нетипизированный файл?
9. Как можно разместить один файл на нескольких носителях информации (если он не помещается на один)?
10. Каким образом можно прочесть содержимое файла по байта?
11. Как можно найти файл по контексту, если имя файла неизвестно (в менеджере, операционной системе и на языке высокого уровня)?
12. Каким образом можно создать семантическую структуру файлов, связав их по определенным элементам?
13. Какую файловую структуру использует операционная система на ваших компьютерах (простую, многоуровневую)? Что это такое?
14 Что такое пользовательский интерфейс операционной системы?
15. Чем характеризуется объект-файл (с точки зрения объектно-ориентированного подхода)?
16. Каким образом можно узнать свойства объекта или выполнить действие над ним?

12. Итерация и рекурсия: необходимость и организация

Цель работы: овладение навыками организации итерационных вычислительных процессов с использованием электронных таблиц и языков программирования высокого уровня.

Краткие теоретические сведения.

Сущность метода итераций заключается в построении рекуррентной последовательности чисел, сходящейся к решению, по формуле $x_{к+1} = u(x_к)$, к=0,1,2,..., где x_0=[a,b] -произвольная точка.

Итерация - это цикл, участок программного кода, который выполняется несколько раз подряд до тех пор, пока не выполнится определённое условие, или же до бесконечности.

Итерации используются для того, чтобы повторить то или иное действие несколько раз, как для разных объектов, так и для одного и того же. Счётчик цикла - это переменная, которая указывает на то, сколько раз надо повторить данные действия, и позволяет подсчитать, какой по счёту заход (т. е. итерация) выполняется в данный момент.

Итерация (часто называема «простая итерация») часто используется при решении алгебраических уравнений. Методы решения систем линейных алгебраических уравнений классифицируют на прямые (точные) и итерационные. Прямые методы основаны на выполнении конечного числа арифметических операций, это, например, метод обратной матрицы, метод Гаусса, метод Гаусса-Жордана, метод прогонки для трехдиагональных матриц и т.д. Суть итерационных методов, в свою очередь, заключаются в том, чтобы за счет последовательных приближений получить решение системы, определяемое необходимой точностью.

Для итерационных методов можно выделить три последовательных этапа:

1. Приведение исходной системы вида $\bar{A} \times \bar{x} = \bar{b}$ к итерационной форме $\bar{x} = \bar{C} * \overline{x^0} + \bar{d}$.
2. Проверка условия сходимости.
3. Решение системы одним из методов.

Метод простых итераций

Условимся, что для общего вида систем выполняется тождество $m=n$, где m - количество уравнений в системе, n - количество неизвестных. Т.е. не имеет смысла решать недоопределенные ($m<n$) и переопределенные ($m>n$) системы, т.к. они могут быть сведены путем элементарных алгебраических преобразований к нормальным ($m=n$) системам линейных уравнений. Другими словами, если имеется «ненормальная» система, то прежде, чем использовать метод простых итераций, преобразуйте ее к нормальной.

Система линейных уравнений может быть записана в матричной форме, где A – матрица коэффициентов, b – вектор свободных членов, x – вектор неизвестных. Возьмем систему:

$$\begin{cases} 5x_1-4x_2-x_3=-2 \\ 4x_1+x_2-2x_3=8 \\ 3x_1+x_2-5x_3=10 \end{cases}$$

Ее матричная форма:

$$\begin{pmatrix} 5 & -4 & -1 \\ 4 & 1 & -2 \\ 3 & 1 & -5 \end{pmatrix} * \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} -2 \\ 8 \\ 10 \end{pmatrix}$$

Переход к итерационному виду осуществляется по следующим формулам:
$c_{ij} = -a_{ij}/a_{ii}$ $d_i = b_i/a_{ii}$, где i, j = 1,2,3…
Также следует отметить, что, несмотря на эти формулы, диагональные элементы новой матрицы обнуляются, хотя должны быть равны -1.
В итоге:

$$\begin{pmatrix} 0 & -0.25 & 0.5 \\ 0.2 & 0 & 0.2 \\ 0.6 & 0.2 & 0 \end{pmatrix} * \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ -2 \end{pmatrix}$$

Некоторые элементы матрицы C будут больше единицы. А глядя на условие сходимости, становится понятно, что, если хотя бы один будет больше единицы, то условие не выполнится, и решение системы путем простых итераций не будет найдено.
Поэтому открываем учебник по линейной алгебре и вспоминаем элементарные матричные преобразования! Прежде чем следовать этапам итерационных методов, нужно привести исходную систему к виду, в котором все диагональные элементы были бы максимальными по модулю в своих строках. Только при таком виде матрицы коэффициентов можно надеется на выполнение условия сходимости.
Смотрим начальную систему. Видим, что третий элемент третьей строки по модулю больше других. Оставим его неизменным. Меняем местами первую и вторую строки. Теперь умножаем строку, ставшую первой, на -1 и складываем с новой второй. В итоге получаем:

$$\begin{pmatrix} 4 & 1 & -2 \\ 1 & -5 & 1 \\ 3 & 1 & -5 \end{pmatrix} * \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} 8 \\ -10 \\ 10 \end{pmatrix}$$

Теперь при подстановке в формулы мы получим итерационную форму верно. К сожалению, это преобразование начальной системы к "благоприятному" виду - чистая аналитика, поэтому записать его в программный код очень сложно, а если даже и попытаться, то в некоторых случаях вероятно возникновение ошибок.

Переходим ко второму этапу: "Проверка условия сходимости" (формулу смотрите выше). Если система не проходит проверку, то приближения не будут сходиться к реальному решению, и ответ получен не будет. В этом случае можно попытаться получить другую "благоприятную" форму. Если условие сходимости выполнено, то стратегия метода простых итераций применима и осуществляется переход к третьему этапу.
В конечном счете, получаем систему линейных алгебраических уравнений в итерационной форме:

$$\begin{cases} x_1=0x^0_1-0.25x^0_2+0.5x^0_3+2 \\ x_2=0.2x^0_1+0x^0_2+0.2x^0_3+2 \\ x_3=0.6x^0_1+0.2x^0_2+0x^0_3-2 \end{cases},$$

где x_1, x_2, x_3 – приближения, получаемые на текущей итерации за счет приближений полученных на предыдущей итерации - x^0_1, x^0_2, x^0_3.
Итерационный процесс в методе простых итераций идет до тех пор, пока вектор приближений не достигнет заданной точности, т.е. пока не выполнится условие выхода:
$Max|x_i-x^0_i|<\varepsilon$, где ε – требуемая точность.
Далее представлен код, написанный в среде **PascalABC.Net**.

```
Program MetodProstIter; {метод простых итераций}
Var
n:integer; {количество переменных или количество уравнений, как кому удобно - }
        {в любом случае они должны быть равны (m=n)}
A:array of array of real; {матрица коэффициентов}
b:array Of real; {вектор-столбец свободных членов}
C:array of Array of real; {матрица Якоби - итерационная форма матрицы A}
d:array of real; {итерационная форма вектора свободных членов}
Err:Boolean; {переменная, по значению которой после выполнения процедуры проверки}
          {сходимости определяется соответствие-несоответствие условию сходимости}
X:array of Real; {вектор неизвестных}

procedure InputA(var n:integer); {ввод матрицы A}
var
i,j:Integer;
begin
  SetLength(A,n); {именно эта встроенная процедура задает правую границу массива}
              {в зависимоти от количества переменных}
  for i:=0 To n-1 Do
  begin
    SetLength(A[i],n); {многомерные массивы в PascalABC можно определять как массивы
массивов}
  end;
  for i:=0 To n-1 Do
  begin    for j:=0 To n-1 Do     read(A[i,j]);    writeln('');  end;
```

```
end;

procedure InputB(var n:integer); {ввод вектора B}
var
i:Integer;
begin  SetLength(b,n); for i:=0 To n-1 Do   readln(b[i]); end;

Procedure IterForm(A:array of Array of real;b:array of real;n:integer);
 {получение итерационной формы системы}

var i,j:Integer;
begin
  SetLength(C,n);   for i:=0 To n-1 Do     Setlength(C[i],n);  end;
  SetLength(d,n);
  for i:=0 To n-1 Do   begin
    for j:=0 To n-1 Do     begin
      if i=j then         C[i,j]:=0       else        C[i,j]:=-A[i,j]/A[i,i];
                                end;
    d[i]:=b[i]/a[i,i];
                                end;
end;

Procedure ProverkaShodimosti(C:array of array of real;d:array of real;n:integer);
 {проверка системы на сходимость}
var i,j:Integer;
summ:real;
begin
  summ:=0;   for i:=0 To n-1 Do     for j:=0 To n-1 Do       summ:=summ+C[i,j]*C[i,j];
  if SQRT(Abs(summ))>1 then   begin
    writeln('Данная система не удовлетворяет условию сходимости');     Err:=True;
  end                              Else Err:=False;
end;

Procedure ProstIterMetode(C:array of array of real;d:array of real;n:integer);
 {собственно, сама стратегия метода простых итераций}
var i,j:Integer;
X0:array of real;
delta:real;
E:array of real;
begin
  SetLength(X0,n);   SetLength(X,n);   Setlength(E,n);
  X0:=Copy(d);
  repeat
  begin     for i:=0 To n-1 Do     begin
      X[i]:=0;
      for j:=0 To n-1 Do       begin
        X[i]:=X[i]+C[i,j]*X0[j];
                                  end;
      X[i]:=X[i]+d[i];       E[i]:=Abs(X[i]-X0[i]);
                                     end;
```

```
   delta:=E[0];
   for i:=1 To n-1 Do      if delta<E[i] then delta:=E[i];
   X0:=Copy(X);
   end;
  Until delta<=0.000001;
  writeln('решение системы равно вектору:');
  For i:=0 To n-1 Do     Writeln(X[i]);
end;

BEGIN
Err:=False;
writeln('Введите количество переменных'); Readln(n);
writeln('Введите матрицу коэффициентов'); InputA(n);
writeln('введите матрицу свободных членов'); InputB(n);
IterForm(A,b,n); ProverkaShodimosti(C,d,n);
if Err=False then ProstIterMetode(C,d,n);
END.
```

Нахождение корня уравнения методом итераций на языке высокого уровня:

```
Program Confirm_2;
function F(x:real):real;
begin F:=3*sin(sqrt(x))+0.35*x-3.8; end;
label m1, m2, m3;
var
eps,f1,f2,x1,x2,a,b:real;
begin
writeln('Нахождение корня уравнения 3*sin(sqrt(x))+0.35*x-3.8=0');
writeln('на интервале [7, 8] методом итераций');
writeln;
a:=7; b:=8; eps:=0.00001; {метод итераций}
x1:=a; f1:=F(x1);
begin
m3: x2:=x1+eps; f2:=F(x2); if f1*f2 > 0 then goto m1 else goto m2;
m1:  x1:=x2; f1:=f2;  goto m3;
m2: writeln('по методу итераций: x = ', (x1 + x2)/2 :0:5);
end;
readln;
end.
```

Для вычисления итераций используют алгоритмы-рекурсии.

Реку́рсия — определение, описание, изображение какого-либо объекта или процесса внутри самого этого объекта или процесса, то есть ситуация, когда объект является частью самого себя. Термин «рекурсия» используется в различных специальных областях знаний — от лингвистики до логики, но наиболее широкое применение находит в математике и информатике.

Рассмотрим пример: рекурсионный способ нахождения факториала в паскале Найти факториал в pascal можно также посредством вызова функции (с помощью рекурсии) .

```
function fact(x:byte):real;
begin  if x=0 then fact:=1  else fact:=fact(x-1)*x;  end;
```

В математике рекурсия имеет отношение к методу определения функций и числовых рядов: рекурсивно заданная функция определяет своё значение через обращение к себе самой с другими аргументами. При этом возможно два варианта:

Конечная рекурсивная функция. Она задаётся таким образом, чтобы для любого конечного аргумента за конечное число рекурсивных обращений привести к одному из отдельно определённых частных случаев, вычисляемых без рекурсии.

Бесконечная рекурсивная функция. Она задаётся в виде обращения к самой себе во всех случаях (по крайней мере, для некоторых из аргументов). Подобным образом могут задаваться бесконечные ряды, бесконечные непрерывные дроби и так далее. Практическое вычисление точного значения здесь, естественно, невозможно, поскольку потребует бесконечного времени. Требуемый результат находится аналитическими методами. Тем не менее, если речь идёт не о получении абсолютно точного значения, а о вычислении заданного приближения искомого значения, то тут в некоторых случаях возможно прямое использование рекурсивного определения: вычисления по нему ведутся до тех пор, пока необходимая точность не будет достигнута.

Другим примером рекурсии в математике является *числовой ряд, заданный рекуррентной формулой*, когда каждый следующий член ряда вычисляется как результат функции от n предыдущих членов. Таким образом, с помощью конечного выражения (представляющего собой совокупность реккурентной формулы и набора значений для первых n членов ряда) может даваться определение бесконечного ряда.

С рекурсией тесно связана математическая индукция: она является естественным способом доказательства свойств функций на натуральных числах, рекурсивно заданных через свои меньшие значения.

В математике отдельно рассматривается класс так называемых «примитивно рекурсивных» функций. Определение примитивно рекурсивной функции также рекурсивно, оно задаёт набор примитивных функций и набор правил; функция является примитивно рекурсивной, если она может быть представлена как комбинация примитивно рекурсивных функций, образованная по этим правилам.

В программировании рекурсия — вызов функции (процедуры) из неё же самой, непосредственно (*простая рекурсия*) или через другие функции (*сложная* или*косвенная рекурсия)* Количество вложенных вызовов функции или процедуры называется глубиной рекурсии. Рекурсивная программа позволяет описать повторяющееся или даже потенциально бесконечное вычисление, причём без явных повторений частей программы и использования циклов. Структурно рекурсивная функция на верхнем уровне всегда представляет собой команду ветвления (выбор одной из двух или более альтернатив в зависимости от условия (условий), которое в данном случае уместно назвать «условием

прекращения рекурсии»), имеющей две или более альтернативные ветви, из которых хотя бы одна является *рекурсивной* и хотя бы одна — *терминальной*. Рекурсивная ветвь выполняется, когда условие прекращения рекурсии ложно, и содержит хотя бы один рекурсивный вызов — прямой или опосредованный вызов функцией самой себя. Терминальная ветвь выполняется, когда условие прекращения рекурсии истинно; она возвращает некоторое значение, не выполняя рекурсивного вызова. Правильно написанная рекурсивная функция должна гарантировать, что через конечное число рекурсивных вызовов будет достигнуто выполнение условия прекращения рекурсии, в результате чего цепочка последовательных рекурсивных вызовов прервётся и выполнится возврат.

Помимо функций, выполняющих один рекурсивный вызов в каждой рекурсивной ветви, бывают случаи «параллельной рекурсии», когда на одной рекурсивной ветви делается два или более рекурсивных вызова. Параллельная рекурсия типична при обработке сложных структур данных, таких как деревья. Простейший пример параллельно-рекурсивной функции — вычисление ряда Фибоначчи, где для получения значения n-го члена необходимо вычислить (n-1)-й и (n-2)-й.

Реализация рекурсивных вызовов функций в практически применяемых языках и средах программирования, как правило, опирается на механизм стека вызовов — адрес возврата и локальные переменные функции записываются в стек, благодаря чему каждый следующий рекурсивный вызов этой функции пользуется своим набором локальных переменных и за счёт этого работает корректно. Оборотной стороной этого довольно простого по структуре механизма является то, что на каждый рекурсивный вызов требуется некоторое количество оперативной памяти компьютера, и при чрезмерно большой глубине рекурсии может наступить переполнение стека вызовов.

Вопрос о желательности использования рекурсивных функций в программировании неоднозначен: с одной стороны, рекурсивная форма может быть структурно проще и нагляднее, в особенности, когда сам реализуемый алгоритм по сути рекурсивен. Кроме того, в некоторых декларативных или чисто функциональных языках (таких как Пролог или Haskell) просто нет синтаксических средств для организации циклов, и рекурсия в них — единственный доступный механизм организации повторяющихся вычислений. С другой стороны, обычно рекомендуется избегать рекурсивных программ, которые приводят (или в некоторых условиях могут приводить) к слишком большой глубине рекурсии. Так, широко распространённый в учебной литературе пример рекурсивного вычисления факториала является, скорее, примером того, как *не надо* применять рекурсию, так как приводит к достаточно большой глубине рекурсии и имеет очевидную реализацию в виде обычного циклического алгоритма.

Имеется специальный тип рекурсии, называемый «хвостовой рекурсией» (структура рекурсивного алгоритма такова, что рекурсивный вызов является

последней выполняемой операцией в функции, а его результат непосредственно возвращается в качестве результата функции). Интерпретаторы и компиляторы функциональных языков программирования, поддерживающие оптимизацию кода (исходного или исполняемого), автоматически преобразуют хвостовую рекурсию к итерации, благодаря чему обеспечивается выполнение алгоритмов с хвостовой рекурсией в ограниченном объёме памяти. Такие рекурсивные вычисления, даже если они формально бесконечны (например, когда с помощью рекурсии организуется работа командного интерпретатора, принимающего команды пользователя), никогда не приводят к исчерпанию памяти. Однако далеко не всегда стандарты языков программирования чётко определяют, каким именно условиям должна удовлетворять рекурсивная функция, чтобы транслятор гарантированно преобразовал её в итерацию. Одно из редких исключений — язык Scheme (диалект языка Lisp), описание которого содержит все необходимые сведения.

Теоретически, любую рекурсивную функцию можно заменить циклом и стеком. Однако такая модификация, как правило, бессмысленна, так как приводит лишь к замене автоматического сохранения контекста в стеке вызовов на ручное выполнение тех же операций с тем же расходом памяти. Исключением может быть ситуация, когда рекурсивный алгоритм приходится моделировать на языке, в котором рекурсия запрещена.

Доказательство корректности программ[**править** | **править вики-текст**]

В отличие от явно-циклических программ, для доказательства корректности рекурсивных нет необходимости искусственно вводить инвариант. Аналитическое доказательство корректности рекурсивной функции сводится к методу математической индукции, то есть к доказательству следующих утверждений:

1. Корректность использования рекурсивного обращения. Доказывается, что результат, вычисляемый в любой рекурсивной ветви функции, будет верным при условии, что соответствующие рекурсивные вызовы, в свою очередь, вернут верный результат.

2. Корректность всех терминальных ветвей. Доказывается, что все терминальные ветви возвращают верные значения. Как правило, это доказывается элементарно, так как терминальные ветви обычно никаких вычислений не содержат.

3. Достижимость терминальной ветви для любого корректного набора параметров после конечного числа рекурсивных вызовов. Доказывается, что изменение параметров вызова функции, которое производится при рекурсивном обращении, через конечное число рекурсивных вызовов приведёт к одному из наборов параметров, для которых задана терминальная ветвь.

Из суммы первого и второго утверждений следует, что в случае достижения терминальной ветви (а это значит — во всех случаях, когда вычисление функции окажется конечным) функция вернёт правильный результат. Третье положение доказывает, что конечным будет любое вычисление. Следовательно,

любой вызов функции с корректными параметрами вернёт правильный результат (с очевидной оговоркой — если глубина рекурсии не окажется настолько большой, что вызовет переполнение памяти).

Примеры рекурсий

```
void func(int n, long x, long y) {
        if (x>=deg(n)+1 && y<=deg(n)) {    if (n>1) { cout << "WHITE";          return; }
                else {   cout << "BLACK";                    return;          }
                                        }
        if (n % 2 == 0 && x<=deg(n) && y>=deg(n)+1) { cout << "BLACK";return;}
        if (n % 2 != 0 && x<=deg(n) && y>=deg(n)+1) {  cout << "WHITE";return;}
        if (n==1 && (x==1 && y==1 || x==2 && y==2)) {cout << "BLACK";return;}
        if (n>1) {if (x>=deg(n)+1 && y>=deg(n)+1) {func(n-1,x-deg(n),y-deg(n));}
                else func(n-1,x,y); } }
void func(long l, long r) {if (r==l) {  return; }else { long mid = (r-l)/2; for (long i=l; i<=l+mid; i++)
{ long temp = ab[i];    ab[i]=ab[r+l-i]; ab[r+l-i]=temp; }
                func(l, l+mid);  func(l+mid+1, r);     }
}
int func(int a, int b) { int l, r, value, k=0;     if (a > b) {                return 0;         }
        if (b-a == 0) { return ans[a];  }
        else {   value=ans[a];  for (int i=a+1; i<=b; i++) {    if (ans[i]>value) {value=ans[i];}
                                        }
                for (int i=a; i<=b; i++) { if (ans[i]==value) {k=i;     break;  }
                }                l=func(a,k-1);          r=func(k+1,b);                  return r+value-l;
        }
}
```

Пример рекурсии ная языке Pyton

```
def F(n):
   if n == 1:
      return 1
   if n == 2:
      return 3
   if n > 2:
      return F(n-1) * n + F(n-2) * (n-1)
print(F(5))
```

Классическим примером бесконечной рекурсии являются два поставленные друг напротив друга зеркала: в них образуются два коридора из уменьшающихся отражений зеркал.

Другим примером бесконечной рекурсии является эффект самовозбуждения (положительной обратной связи) у электронных схем усиления, когда сигнал с выхода попадает на вход, усиливается, снова попадает на вход схемы и снова усиливается. Усилители, для которых такой режим работы является штатным, называются автогенераторы.

Рекурсивная модель — новое направление когнитивной психологии, предназначенное для описания функционирования психики в самых различных ситуациях социального взаимодействия; в частности, рекурсивная модель

призвана моделировать то, как субъект оценивает ситуации, принимает решения, переосмысливает прошлый опыт и т.п.

Для рекурсивной модели характерна гибкость описания ситуаций, т.к. её конструкты, в отличие от конструктов большинства психологических теорий, строго не фиксированы, они создаются каждый раз под конкретную ситуацию согласно определённым правилам.

Рекурсивные процессы всё более и более широко применяются в современной науке. В частности, А. В. Анисимов, развивая идеи рекурсивности при создании систем искусственного интеллекта, открыл новое направление в лингвистике, в основу которого положен алгоритмический анализ и рекурсивный синтез структуры речи.

Несмотря на то, что в лингвистике благодаря рекурсии удалось получить важные результаты, в психологии рекурсия до сих пор систематически не применялась. Таким образом, рекурсивная модель функционирования психики является пионерской.

При разработке рекурсивной модели создан особый, специфический язык, который представляет собой язык современной научной психологии, приспособленный к нуждам рекурсивной модели и дополненный специальным графическим языком.

Рекурсивная модель является инструментом, который открывает для людей фрактально-рекурсивную психологическую реальность, она поможет изменить психику и даже, до некоторой степени, сознание людей. Кроме того, она будет служить основой при разработке психокоррекционных техник, направленных на привлечение дополнительных психических ресурсов к решению психологических проблем, а также на уточнение, преобразование и организацию прошлого опыта.

Рекурсивная модель способна стимулировать активность человека, направленную на выявление скрытых возможностей и опасностей ситуации, и, кроме того, на поиск путей удовлетворения актуализированных потребностей, в том числе, на достижение стоящих перед человеком целей.

Графические рекурсивные алгоритмы

Ситуация, когда алгоритм вызывает себя в качестве вспомогательного, называется рекурсией (это слово происходит от латинского recursio -- возвращение). Рекурсивными бывают подпрограммы-процедуры и подпрограммы-функции. В рекурсивных подпрограммах в теле процедуры или функции имеется вызов этих же подпрограмм. Процесс обращения к себе самому может длиться бесконечно. Для обеспечения остановки устанавливают барьер в виде некоторого условия.

Пример 1. Опишем алгоритм рисования окружности радиуса r с центром в точке (x,y) с четырьмя окружностями вокруг. При этом необходимо знать расстояния (r1) от точки(x,y) до центров окружностей окружения (радиусы орбиты), которые, очевидно равны.

```
procedure krug(x,y,r,stepN,stepMax:integer);
```

```
begin {вычисляем координаты центра i-й окружности};
krug(x+r,y,r div 2); krug(x,y+r,r div 2); krug(x-r,y,r div 2); krug(x,y-r,r div 2);
Arc(x,y,r,0,360); {рисуем окружность с центром в точке (x,y) и радиусом r}
end;
```

Таким образом, в описанной процедуре осуществляется вызов ее же самой (рекурсия) в качестве вспомогательной. Если осуществить вызов этой же процедуры из основной программы с начальными параметрами, то рекурсивные вызовы никогда не закончатся и программа будет работать бесконечно. Для того, чтобы этого не случилось, можно ввести в качестве аргумента процедуры некоторую величину stepN, которая при каждом новом вызове процедуры будет увеличиваться на 1, а в тело процедуры что его операторы должны выполняться только при stepMax>stepN, то есть это условие должно играть роль своеобразной «заглушки», ограничивающей число вызовов.

Ниже приведена программа, полностью решающая поставленную задачу.

В основной программе запрашиваются число уровней stepMax. Центр самой окружности располагается в центре экрана.

```
uses crt,graphABC;
Var stepMax, x, y, r : integer;
procedure krug(x,y,r,stepN,stepMax:integer);
begin
if stepMax>stepN then begin {задаём координаты окружностей а так же их размеры}
krug(x+r,y,r div 2, stepN+1, stepMax); krug(x,y+r,r div 2, stepN+1, stepMax);
krug(x-r,y,r div 2, stepN+1, stepMax); krug(x,y-r,r div 2, stepN+1, stepMax);
end;
Arc(x,y,r,0,360); {получаем окружноть}
end;
begin {начало основной программы}
read(stepMax);
x:=700; {начальные координаты} y:=1000;
r:=90; {начальный радиус} SetWindowSize(700,700); {размер графического окна}
krug(WindowWidth div 2, WindowHeight div 2{ Размещаем рисунок в центре графич окна
}, r, 0, stepMax);
end.
```

Изменяя в процессе демонстрации работы программы stepmax можно получать множество привлекательных рисунков.

Фракталы

Пример 1. Кривые Гильберта

Этот узор состоит из суперпозиции пяти кривых. Три наложенные друг на друга кривые имеют форму H_1, H_2 и H_3. H_{i+1} получается соединением четырех экземпляров H_i вдвое меньшего размера, повернутых соответствующим образом и "стянутых" вместе тремя прямыми линиями. H_1 можно считать состоящей из четырех вхождений пустой H_0, соединенных этими же тремя линиями. Кривая H_i называется кривой Гильберта i-го порядка в честь ее первооткрывателя Д.Гильберта. Каждая кривая H_i состоит из четырех вдвое меньших H_{i-1}, то

процедура для рисования Hi будет включать четыре обращения для рисования H_{i-1}, соответствующим образом повернутых и уменьшенных вдвое. Обозначим эти четыре части через A, B, C и D.

Это кривые Гильберта 1-го порядка.

Соединительные прямые будем обозначать стрелками, указывающими в соответствующем направлении. Будем предполагать, что направление задается целым параметром i как i·45 градусов. Тогда получаем:

Для построения A, B, C, D получается следующая схема рекурсий:

Кривые Гильберта 2-го порядка.

Для рисования линии будем использовать процедуру:

Line(i, s: integer), где i - направление, s - длина отрезка.

```
procedure Line( i, s: integer);
begin x:=i*45/180*pi; LineTo(PenX{текущие координаты} + Round(s*cos(x)), PenY +
Round(s*sin(x)));
end;
```

Используя эту процедуру, с помощью рекурсивных обращений напишем процедуру, соответствующую схеме A:

```
Procedure A(i, s: integer);
begin if i>0 then begin
D(i-1,s); Line(4,s); A(i-1,s); Line(6,s); A(i-1,s); Line(0,s); B(i-1,s);
end;
end;
```

Аналогично составляются процедуры для B, C, D. Процедура A инициируется главной программой по одному разу аждой из кривых Гильберта. Главная программа определяет начальную точку кривой, т.е. исходные координаты (x0,y0) и начальное значение длины отрезка u (желательно, чтобы u=2n).

Эта программа рисует кривые Гильберта 5-го порядка.

```
Uses Crt, Graphabc;
Var i,x0,y0,u : integer;
x : Real;
procedure Line( i, s: integer);
begin
x:=i*45/180*pi;
LineTo(PenX+ Round(s*cos(x)), PenY + Round(s*sin(x)));
END;
Procedure B(i, s: integer);forward; Procedure C(i, s: integer);forward;
Procedure D(i, s: integer);forward; Procedure A(i, s: integer);
begin
if i>0 then
begin D(i-1,s); Line(4,s); A(i-1,s); Line(6,s) ; A(i-1,s); Line(0,s); B(i-1,s);
end;
end;
Procedure B(i, s: integer);
begin
if i>0 then begin C(i-1,s); Line(2,s); B(i-1,s); Line(0,s); B(i-1,s); Line(6,s); A(i-1,s);
end;
end;
```

```
Procedure C(i, s: integer);
begin
if i>0 then begin
B(i-1,s); Line(0,s); C(i-1,s); Line(2,s); C(i-1,s); Line(4,s); D(i-1,s);
end;
end;
Procedure D(i, s: integer);
begin
if i>0 then begin
A(i-1,s); Line(6,s); D(i-1,s); Line(4,s); D(i-1,s); Line(2,s); C(i-1,s);
end;
end;
begin
SetWindowSize(700,540);
write (' ','КРИВЫЕ ГИЛЬБЕРТА');
begin
x0:=400; y0:=350;{координаты начала рисования} u:=128{длина отрезка}; i:=1;
while i<=5 {задаем порядок} do
begin
MoveTo(x0,y0); {задаем координаты пера}
A(i,u); i:=i+1; u:=u div 2;
x0:=x0+(u div 2); y0:=y0+(u div 2);
end;
end;
end.
```

Кривые Серпинского

Аналогично, путем наложения друг на друга нескольких кривых, получается рисунок из кривых Серпинского. Главное отличие кривой Серпинского от кривой Гильберта в том, что первая кривая замкнута. Значит, основная рекурсивная схема должна давать разомкнутую кривую, четыре части которой соединяются линиями, не принадлежащими самому рекурсивному образу. Четыре составляющих образа обозначим через A, B, C, D

Соединительные прямые будем обозначать стрелками, указывающими в соответствующем направлении. Будем предполагать, что направление задается целым параметром i как i·45 градусов.

Основной образ кривых Серпинского задается программой:

```
begin if i>0 then begin
A(i-1,s); Line(7,s); B(i-1,s); Line(0,2*s); D(i-1,s); Line(1,s); A(i-1,s);
end;
end;
```

Аналогично получаются процедуры для B, C, D. Главная программа строится по образу S. Ее задача - установить начальные значения для координат рисунка и задать единичную длину линий.

Эта программа рисует кривые Серпинского 4-го порядка.

```
Uses Crt, Graphabc;
Var x0,y0,u,i : integer;
```

```
X : Real;
procedure Line( i, s: integer);
begin
x:=i*45/180*pi;
LineTo(PenX{текущие координаты} + Round(s*cos(x)), PenY + Round(s*sin(x)));
end;
Procedure B(i, s: integer);forward; Procedure C(i, s: integer);forward;
Procedure D(i, s: integer);forward;  procedure A(i,s: integer);
begin
if i>0 then begin A(i-1,s); Line(7,s); B(i-1,s); Line(0,2*s); D(i-1,s); Line(1,s); A(i-1,s); end;
end;
Procedure B(i, s: integer);
begin
if i>0 then     begin B(i-1,s); Line(5,s); C(i-1,s); Line(6,2*s); A(i-1,s); Line(7,s); B(i-1,s); end;
end;
Procedure C(i, s: integer);
begin
if i>0 then  begin C(i-1,s); Line(3,s); D(i-1,s); Line(4,2*s); B(i-1,s); Line(5,s); C(i-1,s); end;
end;
Procedure D(i, s: integer);
begin
if i>0 then begin D(i-1,s); Line(1,s); A(i-1,s); Line(2,2*s); C(i-1,s); Line(3,s); D(i-1,s); end;
end;
begin
SetWindowSize(900,1000);
write (' ', 'КРИВЫЕ СЕРПИНСКОГО ЧЕТВЁРТОГО ПОРЯДКА');
x0:=120; y0:=700; {координаты начала рисования} u:=13; i:=1;
for i:=1 to 4 do begin MoveTo(x0,y0);
A(i,u); Line(7,u);  B(i,u); Line(5,u); C(i,u); Line(3,u); D(i,u); Line(1,u);
end;
end.
```

Приведем пример простой программы square, строящей на экране компьютера изображение вписанных друг в друга квадратов и использующей для этого рекурсивный вызов функции.

Анализ изображения: на рисунке большой квадрат со вписанными в него квадратами меньшего размера. Причём, каждый вписанный квадрат имеет свои координаты, которые вычисляются по координатам описанного квадрата. Всего изображено 6 уровней квадратов. Для того, чтобы составить программу построения этого изображения, можно:

описать функцию изображения одного квадрата со вписанным квадратом меньшего размера.

для изображения каждого квадрата следующего уровня использовать эту же функцию, только с другими значениями параметров которые будут вычисляться автоматически в той же функции рисования квадратов.

Опишем алгоритм рисования:

Сначала вычерчивается первый четырёхугольник (квадрат). После этого вызывается функция Kvad строящая внутри этого четырехугольника (квадрата) малый квадрат так, что вершины его лежат точно на серединах сторон большого квадрата. В обращение включен целочисленный аргумент n, определяющий глубину рекурсии.

Для рисования квадратов будем использовать линии определенной длины. При этом надо задавать координаты каждой линии (x1,y1), (x2,y2). Для этого будем использовать параметры графического окна:

```
xc:=WindowWidth div 2; {возвращаем ширину графического окна}
yc:=WindowHeight div 2; {возвращаем высоту - // - // - }
ax:=xc-a; ay:=yc-a; {находим координаты для первой линии}
bx:=xc+a; by:=yc-a; {находим координаты для второй линии}
cx:=xc+a; cy:=yc+a; {находим координаты для третьей линии}
dx:=xc-a; dy:=yc+a; {находим координаты для четвертой линии}
```

где a - длина линии.

```
function Kvad(xc,yc,a:integer; ax,ay,bx,by,cx,cy,dx,dy,m:real; n:integer) :integer;
begin
if n=0 then Kvad:=0
else
begin {координаты линий округляем}
line(round(ax),round(ay),round(bx),round(by));
line(round(bx),round(by),round(cx),round(cy));
line(round(cx),round(cy),round(dx),round(dy));
line(round(dx),round(dy),round(ax),round(ay));
dec(n); {уменьшаем n на 1}
```

Рекурсивный вызов функции и вычисление новых значений производится так:

```
Kvad:=Kvad(xc, yc, a, ax+(bx-ax)*m, ay+(by-ay)*m, bx+(cx-bx)*m, by+(cy-by)*m,cx+(dx-cx)*m,
cy+(dy-cy)*m, dx+(ax-dx)*m, dy+(ay-dy)*m, m, n);
end;
end;
```

m - вспомогательная переменная, в данном случае используется для нахождения середины стороны.

Таким образом, в описанной функции мы осуществляем вызов ее же самой в качестве вспомогательной.

Если осуществить вызов этой же функции из основной программы с начальными параметрами, то рекурсивные вызовы никогда не закончатся и программа будет работать бесконечно. Для того, чтобы этого не случилось, можно ввести в качестве аргумента функции некоторую величину n, которая при каждом новом вызове процедуры будет уменьшаться на 1 (применяем dec(n)), а в тело функции что его операторы должны выполняться только при n<>0, то есть это условие должно играть роль своеобразной «заглушки», ограничивающей число вызовов.

Рекурсия может быть использована для получения линейного рисунка, например «дерева», например:

```
Uses Crt, Graphabc;
Var x, fi : Real;
```

```
i, x0, y0, u, divisor : integer;
```
Для рисования линии будем использовать процедуру: Line(i, s: integer), где i - направление, s - длина отрезка.
```
procedure Line( i, s: integer);
begin
x:=(i*fi-90)*pi/180;
LineTo(PenX{текущие координаты}+ Round(s*cos(x)), PenY + Round(s*sin(x))); {рисуем линию под определенным углом}
end;
procedure A(i,s,k, kmax: integer); //k - шаг рекурсии, kmax - порядок рекурсии
var cX, cY: integer; //координаты разветвления
begin
Line(i,s); //рисуем ножку ветви
if k<kmax then begin
cX:= PenX; //запоминаем координаты ветвления
cY:= PenY; A(i,s div divisor,k+1,kmax); //рисуем среднюю ветвь
MoveTo(cX,cY); A(i-1,s div divisor,k+1,kmax); //рисуем левую ветвь
MoveTo(cX,cY); A(i+1,s div divisor,k+1,kmax); //рисуем правую ветвь
end
else begin
Line(i,s); //если рекурсия окончена, дорисовываем ветку
end;
end;
begin
read(i); //порядок рекурсии
SetWindowSize(700,640);
x0:=330; //начальные координаты
y0:=500;
u:=243; //длина самой крупной ножки
divisor:=3; //во сколько раз длина ветки на следующем шаге меньше, чем
на предыдущем.
fi:=70; //угол между ветвями
MoveTo(x0,y0); //переходим к начальным координатам
A(0,u,0,i); //рисуем ветку
end.
```

Разработаем программу, строящую на экране компьютера изображение описанных кругов.

```
uses crt,graphABC,events;
var
stepMax,x,y,r:integer;
```
напишем процедуру krug которая рисует окружность и описывает её шестью окружностями меньщего размера:
```
procedure krug(x,y,r,stepN,stepMax:integer);
var nr:integer;
begin
if stepMax>stepN then
begin circle(x,y,r); {рисуем основной круг}
```
Вычисляем центр каждого круга:

```
krug(x+round(2*r),y, r div 3, stepN+1, stepMax); {круг с права}
krug(x-round(2*r),y, r div 3, stepN+1, stepMax); {круг с лева}
krug(x+round(2*r*cos(PI/3)),y-round(2*r*sin(PI/3)),r div 3, stepN+1, stepMax); {верхний правый круг}
krug(x-round(2*r*cos(PI/3)),y-round(2*r*sin(PI/3)),r div 3, stepN+1, stepMax); {левый верхний круг}
krug(x+round(2*r*cos(PI/3)),y+round(2*r*sin(PI/3)),r div 3, stepN+1, stepMax); {нижний правый круг}
krug(x-round(2*r*cos(PI/3)),y+round(2*r*sin(PI/3)),r div 3, stepN+1, stepMax); {нижний левый круг}
end;
end;
begin
write('[фигуру какого порядка следует нарисовать]','_'); read(stepMax);
clearwindow;
write('[введите желаемый радиус]','_'); read(r);
clearwindow;
writeln (' ','на экране окружности ' ,stepMax, ' го',' порядка');
x:=350; {начальные координаты}
y:=400; SetWindowSize(700,700);
krug(x,y,r,0,stepMax); {вызываем процедуру krug }
end.
```

Дерево рекурсивных вызовов

Проанализируем пример 3:

Для получения окружност со вписанными окружностями (второго порядка) требуется сделать следующие действия:

Шаг 1.

```
procedure krug(x,y,r,stepN,stepMax:integer);
var nr:integer;
begin
if stepMax>stepN then circle(x,y,r);
end;
```

Шаг 2.

Вычислим радиус для следующей окружности: nr:=r div 3;

Вызываем в процедуре krug ещё одну процедуру krug и передаём ей новые параметры:

krug(x,y,nr, stepN+1, stepMax); получаем:

окружность в центре которого ещё одна окружность.

Шаг 3.

Добавляем в основную процедуру krug вторую процедуру krug со следующими параметрами:

krug(x,y+2*nr,nr, stepN+1, stepMax); передвигаемся по игрику

Шаг 4.

Добавляем третью процедуру krug : krug(x,y-2*nr,nr, stepN+1, stepMax);

Шаг 5.

Добавляем четвертую процедуру krug:

Для нахождения центра круга будем использовать тригонометрические функции.

krug(x+2*round(nr*sin(2*PI/3)),y+nr,nr, stepN+1, stepMax);

Шаг 6.

Аналогично добавляем пятую процедуру krug:

krug(x-2*round(nr*sin(2*PI/3)),y+nr,nr, stepN+1, stepMax);

Шаг 7.

Добавляем шестую процедуру krug:

krug(x+2*round(nr*sin(2*PI/3)),y-nr,nr, stepN+1, stepMax);

Шаг 8.

Добавляем седьмую процедуру krug:

krug(x-2*round(nr*sin(2*PI/3)),y-nr,nr, stepN+1, stepMax);

получаем конечный результат:

«Заглушка» срабатывает когда stepN становится больше stepMax

При каждом повторе процедуры krug, stepN увеличивается на единицу.

Решение нелинейного уравнения методом итерации в Excel

Задание: на отрезке [a,b] требуется найти **корень нелинейного уравнения методом итерации** в табличном процессоре Excel с использованием **циклических ссылок.**

$x-x^3+1=0 \quad a=1 \quad b=2$

Решение:

Найдем корень нелинейного уравнения **в табличном процессоре Excel методом итерации** с использованием циклических ссылок. Для включения **режима циклических вычислений в Excel**2003 в меню Сервис/Параметры/вкладка Вычисления следует поставить флажок Итерации и флажок выбора вида ведения вычислений: автоматически. В MS Excel 2010 следует зайти в меню Файл/Параметры/Формулы и поставить флажок в поле "Включить итеративные вычисления".

Используем формулу:

$$X_{n+1} = X_n - \frac{f(X_n)}{M}, \quad n = 0, 1, 2, \ldots,$$

M – максимальное значение производной на промежутке (по модулю). Найдем M, для этого вычислим

f’(1)=-2 f’(2)=-11

Т. к. значение производных <0, возьмем функцию с противоположным знаком: f(x)=-x+x3-1.

M=11

В ячейку A7 введем значение a =1, в ячейку B7 введем формулу расчета текущего значения x: =ЕСЛИ(B7=0;A7;B7-(-B7+СТЕПЕНЬ(B7;3)-1)/11)

В ячейку C7 введем формулу для контроля значения f(x): =B7-СТЕПЕНЬ(B7;3)+1

Получим корень уравнения x=1,375

4			
5	Метод итерации		
6	*Хнач*	*Хтек*	*F(Хтек)*
7	1,000	1,325	0,000
8			

В А7 введем начальное приближение = 2, получим корень x=1,375
Получен тот же результат, значит корень на данном промежутке один.

4			
5	Метод итерации		
6	*Хнач*	*Хтек*	*F(Хтек)*
7	2,000	1,325	0,000
8			

Порядок выполнения работы.
1. Изучить теоретический материал.
2. Методом простых итераций найти корень уравнения (задается преподавателем). Решение реализовать на языке высокого уровня и в электронной таблице.
3. Поснить работу алгоритма вычисления факториала с помощью рекурсии.
4. Построить кривые Серпинского и предложить их применение в математической биологии и медицине.
5. Преложить метод реализции п.4 в MatCad и Excel.
6. Оформить отчет, включающий: результаты выполнения работы (алгоритмы, программные коды, скри-шоты) и ответы на контрольные вопросы (не менее трех).

Контрольные вопросы:
1. Что называется итерацией?
2. Что называется рекурсией?
3. Для чего применяются кривые Серпинского?
4. Какова роль фракталов в природе?
5. Какие физиологические процессы можно отнести к рекурсивным?
6. Что такое метод дихатомии?
7. Каким образом можно применять рекурсию при вычисления ряда (например, Тейлора для гармонической функции)?
8. Какой итерационный процесс будет бесконечным: расходящийся, сходящийся?
9. Как представить процесс дифференциальной диагностики заболевания в виде итарации (рекурсии)?
10. Как реализуется итерация в логическом базисе с помощью логических функций и переменных?
11. Приведите примеры семантических рекурсий (например: «у попа была собака, он ее любил…»)
12. Как осуществляется рекурсивный диагностический процесс?

Информационные сайты: http://web-pascal.narod.ru/stat/recurs.htm, **http://it.kgsu.ru**, **http://ru.wikipedia.org**, **http://tvd-home.ru/recursion**, http://pascal.helpov.net

13. Основы работы в медицинских информационных базах данных

Цель работы: изучение принципов построения информационной базы данных Medline и приобретение навыков работы с ней.

Краткие теоретические сведения

Наибольшее распространение получила Medline (MEDlars onLINE) — крупнейшая библиографическая база статей по медицинским наукам, созданная Национальной медицинской библиотекой США (U.S. National Library of Medicine, NLM). Охватывает около 75 % мировых медицинских изданий. Использует словарь MeSH.

Преимущества базы:

1. Быстрый поиск статей по заданной теме с получением для них абстрактов и библиографических данных;
2. Предоставляется возможность связаться с авторами публикации.

Недостатки:

3. Абстракты для старых статей могут быть неполны.

Существует несколько интерфейсов, с помощью которых осуществляется доступ к базе данных Medline. Среди них свободные, такие как PubMed и HubMed, и коммерческие, например, Ovid Technologies, SwetsWise и некоторые другие.

База поддерживается Национальной медицинской библиотекой США (NLM). Включает описания и рефераты из 4.600 медицинских и биологических журналов, публикуемых в более чем 70 странах мира. Нижняя хронологическая граница MEDLINE - 1951 год. На 2010 год ее наполнение составляло более 19 миллионов записей. Актуализация базы проводится еженедельно. Доступ к MEDLINE открыт на сервере NLM через службу PubMed. Для работы оптимальнее выбрать пункт меню Limits, раскрытие которого приводит к появлению многоуровневого меню, позволяющего сформировать запрос с использованием многочисленных фильтров: датой ввода в базу, типом и видом издания, языком текста и иными специфическими для медицины параметрами. MEDLINE обладает близкой к образцовой подсистему работы с данными. Система позволяет просматривать записи, отмечая релевантные документы и сохранять выделенные документы на локальном диске пользователя (функция Clipboard). Каждая запись в перечне результатов поиска снабжена ссылкой Related Articles, щелчек мышью по которой приводит к появлению перечня статей, содержание которых аналогично данной. Эта функция крайне полезна при проведении эвристического поиска, учитывающего ассоциативные связи и скрытые закономерности. Фиксируется "история" разысканий с возможностью вернуться к любому их этапу. Качество библиографических записей MEDLINE, которая среди прочих содержит большой объем сведений о статьях из российской медицинской периодики, можно считать образцовым: все описания

включают многочисленные классификационные рубрики, большинство снабжено развернутыми рефератами.

Наиболее популярный "шлюз" в MEDLINE — сервер Национальной медицинской библиотеки США http://www.nlm.nih.gov/. Доступ к этой базе данных осуществляется на этом сервере через две службы — PubMed и Internet Grateful Med http://www.nlm.nih.gov/databases/freemedl.html. Вторая служба позволяет получить доступ не только к базе данных MEDLINE, но и к базам AIDSLINE, HISTLINE, DIRLINE, OLDMEDLINE, SDLINE и др. Доступ в MEDLINE предоставляют также многие сайты, посвященные медицине. (например, http://healthgate.com, http://www.kfinder.com, http://php.silverplatter.com и другие).

Кроме Medline существует и множество других баз данных. Приведем несколько примеров: Embase — биомедицинская литература, в основном европейские и японские фармакологические журналы; Biosis — биологическая литература, очень широкий спектр; CAB Health — инфекционные заболевания, в частности СПИД и заболевания, передаваемые половым путем; Science Citation Index — основные журналы по всем научным тематикам; позволяет осуществлять поиск цитат. Доступ к этим каталогам можно получить через службы BIDS http:// www.bids.ac.uk, EDINA http://edina.ed.ac.uk/ или Dialoghttp://www.krinfo.com/dialog/dweb/dwebfly.html.

Кроме того, на сервере UnCover (http://www.carl.org/ или http://uncweb.carl.org/) содержится информация более чем о 6 миллионах научных статей и программное обеспечение для их поиска и отправки факсом (за плату) заказчику.

Поиск медицинских учреждений, специалистов

Hospitals on the Net — http: www.hon.ch/cgi-bin/find? — возможность поиска больниц и клиник, а также других медицинских учреждений в Сети.

Scholary societies of the world — http://www.lib.uwaterloo.ca/society/overview.html — названия обществ расположены в алфавитном порядке, по предмету или типу.

World list of Universities — http://www.mit.edu:8001/people/cdemello/univ.html — списки университетов в алфавитном и географическом порядке.

В России для поиска людей (в том числе по адресу электронной почты) можно попытаться воспользоваться поиском E-ROSS http://www.dubna.ru/eros

Интерфейс PubMed прост и понятен. Есть только одно поле для ввода поискового запроса и клавиша "Go" рядом с ним. Но этим функции системы не ограничены. Что еще получает пользователь? Во-первых, это разнообразные способы уточнить ваш запрос, введя в него определенные ограничения (Limits). Для вашего удобства система ведет историю поисковых запросов, что позволяет изменять поисковую стратегию, объединяя между собой разные запросы. Можно сохранять промежуточные результаты поиска в Clipboard, а затем использовать их вместе на ваше усмотрение. Есть инструмент Details, для

получения подробной информации о том, как PubMed понял ваш запрос, и допустили ли вы какие-либо ошибки.

Помимо работы с запросами можно изменять форматы вывода документов. Этих форматов огромное количество, наиболее употребимые из них: Summary, Brief, Abstrat, AbsatractPlus, Citation, MEDLINE. Подробнее о форматах можно посмотреть в разделе помощи [4]. Заслуживают внимания возможность отправки результатов поиска на email, а так же функция для просмотра самой свежей информации по выбранной теме в вашем RSS-ридере.

Описанные функции доступны любому, кто пользуется сервисом PubMed, но зарегистрированным пользователям предоставляется еще больше возможностей для работы. Это возможность сохранения поисковых запросов (история запросов для незарегистрированных пользователей хранится лишь в течение одной сессии работы), составление собственных коллекций статей и еще ряд дополнительных возможностей для настройки под себя поисковой системы PubMed.

Другие сетевые международные базы данных медицинской информации.

База данных **PubMed Central (PMC) .** Это созданный и поддерживаемый NCBI цифровой архив биомедицинских публикаций состоящий из журналов, предоставляющих бесплатный доступ к полным текстам статей. Другими словами все статьи, которые вы найдете на PMC можно так же свободно получить и на сайтах журналов, где они опубликованы. Роль NLM в данном случае обеспечить не бесплатность статей, а доступ ко многим публикациям с одного сайта. Поиск осуществляется в том же интерфейсе, что и в PubMed. Разница лишь в выдаче результатов. Количество результатов будет значительно меньше, но зато все они полнотекстовые. Кстати переход между этимм базами осуществялется довольно просто - выбором базы PMC в выпадающем меню справа от слова Search, при работе в интерфейсе PubMed. Об инструментах поиска я уже писал выше, поэтому повторяться не буду. При сравнении с HighWire Press этот сервис выгодно отличается средствами поиска, но проигрывает по общему количеству статей.

Однако многие журналы не ограничиваются только бесплатным распространением своих материалов. Относительно небольшая часть коллекции PMC публикуется на условиях Creative Commons License и ей подобных лицензий. Это дает больше свободы при расространении и использовании статей, по сравнению с традиционно лицензированными работами. Очевидным плюсом этой системы является возможность получения полных тестов и интеграция с другими сервисами, предоставляемыми NCBI. Минусы проявляются только тогда, когда оказывается, что нужная статья стоит денег.

База дынных **The Cochrane Library** - основной продукт деятельности Кокрановского сотрудничества. Это электронная база данных, называемая Кокрановской библиотекой. Содержащиеся в библиотеке систематические обзоры содержат строго доказанные научные факты. Кокрановское Сотрудничество - это международная некоммерческая организация. Ее основная

задача – собирать новейшую, достоверную информацию о результатах медицинских вмешательств. Кокрановский систематический обзор, в отличие от обычного обзора литературы, строится по более жестким правилам. Эти правила ограничивают круг исследований, результаты которых могут быть использованы для написания обзора. Более подробно можно почитать на русском языке здесь [8] и здесь [9], на английском здесь. Доступ к библиотеке осуществялется через Wiley InterScience. Методы поиска принципиально не отличаются от таковых на предыдущих сайтах. Так же как и в PubMed возможен поиск по терминам MeSH [11]. Доступ к оглавлению и резюме обзоров свободен. Для доступа к полным текстам нужна подписка. Особенность этой библиотеки - это подкасты, своего рода дайджесты избранных обзоров. Подкасты выходят нечасто и на английском языке. Т.е., Кокрановская библиотека содержит ограниченный круг публикаций - метаинформацию о радомизированных клинических исследованиях. Ее достоинство - полностью достоверный материал обзоров и наличие довольно большого количества мультимедийных материалов, чего не было у описанных выше сайтов. Помимо подкастов здесь можно найти видео о библиотеке и проводимых мероприятиях.

Порядок выполнения

1. Осуществите поиск информации по тематике самостоятельной работы (выдается преподавателем) в упомянутых в теоретических сведениях баз данных медицинской информации.
2. Изучите интерфейс баз данных медицинской информации.
3. Оформите отчет, включающий: характеристики изученных баз данных медицинской информации, интерфейс (приведите скрин-шоты диалога с пользователем), гносеологические возможности, выводы о недостатках и преимуществах определенных баз данных, краткие реферативные обзоры, полученные по результатам выполнения п.1.

14. Компьютерные сети. Электронная почта

Краткие теоретические сведения.

Современность эпоха характеризуется как эпоха глобального информационного общества, содержание которой составляют информационные технологии, реализованные в том числе с помощью компьютерных сетей разнообразных конфигураций и иерархических уровней.

Интернет играет огромное значение в решении задач в здравоохранении и практической медицине: доступ к медицинским библиотекам, дистанционное обучение, телемедицина, поиск современных научных медицинских данных, телеконференции и т.д. Этому способствуют технические и гносеологические преимущества виртуального телеобщения. В связи с этим, врачу-кибернетику необходимы навыки владения преимуществами сетевого общения с осознанием недостатков такового.

При поиске информации в браузере применяются команды логического объединения и исключения. Символы «+» и «-» в запросе позволяют добавлять или исключать какие-либо слова из текста. Слово, помеченное «+», будет обязательно присутствовать в документах, которые найдет поисковая система по запросу. Слово, помеченное «-», будет отсутствовать в выдаче. Команды «+» и «-» должны быть написаны слитно со словом, к которому они относятся. В противном случае поисковая машина начнет рассматривать их как элементы запроса, а не как команды. Также применяются логические операторы при поиске информации:

- логическое И» (обозначается как амперсанд (&)) - позволяет перечислить слова, которые обязательно должны встречаться в пределах одного предложения в искомом документе;
- «логическое ИЛИ» (обозначается символом «|») – позволяет осуществлять поиск по документам, в тексте которых присутствует только одно из перечисленных слов.

К типовым русскоязычным и иностранным медицинским системам поискового типа относятся: http://www.medpoisk.ru/, http://www.ncbi.nlm.nih.gov/pubmed/, www.scirus.com, www.medexplorer.com, http://scholar.google.com, www.scienceresearch.com, http://www.ncbi.nlm.nih.gov/ и http://research.bmn.com/.

При необходимости перевода используются специализированные языковые переводчики. Для выполнения транслитерации (требуется при работе с зарубежными информационными данными) используются специальные программы.

Для исключения нарушения авторских прав применяются так называемые «антиплагиатские» программы. Окончательное решение об авторстве текста может быть принято только в судебном порядке согласно действующему законодательству.

На данном занятии у студентов формируется овладение следующими составляющими профессиональных компетенций**:** способностью и готовностью к работе с медико-технической аппаратурой, используемой в работе с пациентами, владеть компьютерной техникой, получать информацию из различных источников, работать с информацией в глобальных компьютерных сетях; применять возможности современных информационных технологий для решения профессиональных задач,

Порядок выполнения:

1. Осуществите поиск информации по ключевой фразе «Медицинские информационные технологии» первых десяти результатов в различных поисковых машинах. Сделайте аналитические выводы.
2. Найдите в компьютерной сети и изучите презентацию на тему «Создание электронной почты».
3. Найдите в компьютерной сети и изучите презентацию на тему «Автоматизированное рабочее место врача».
4. Осуществите анализ предоставляемой в компьютерной сети медицинской аппаратуры (например, танометров или глюкометров, танометров И глюкометров). Приведите результаты использования в командной строке поисковой машины различных операторовю
5. Найдите в сети сайт по продаже литературы по медицинской тематике (включая медицинскую кибернетику).
6. Изучите структуры электронных магазинов.
7. Создайте временный почтовый ящик и отправьте «сами себе» текст любого послания с учетом этики общения в сети с прикрепленными файлами.
8. Используя разные переводчики, переведите любой текст (логически связанное предложение с техническими и медицинскими терминами) на английский, немецкий, французский, белорусский, украинский, испанский, китайский языки с указанием специализированной настройки (общий перевод, технический, гуманитарный, медицинский). Сделайте обратный перевод. По результатам сделайте выводы. Выполните транслитерацию (латинскую). Проверьте авторство русскоязычного текста путем применения «антиплагиатской» программы.
9. Ответьте на контрольные вопросы.

По результатам выполнения задания составьте отчет.

Контрольные вопросы:

1. Какое ПО необходимо для работы в глобальной сети?
2. Какие преимущества дает врачу использование Интернета?
3. Дайте характеристику всемирной информационной сети WWW.
4. Охарактеризуйте работу систем для поиска информации общего назначения.

5. Как происходит процесс ранжирования информации в поисковой системе?
6. Назовите медицинские ресурсы Интернет.
7. Перечислите основные этапы возникновения и перспективы развития глобальных компьютерных сетей.
8. Перечислите основные возможности электронной почты.
9. В чем заключается функциональное назначение файлообменных сетей.
10. Приведите примеры (название, функции, структуру, организация диалога, возможности) сайтов по доказательной медицине в стране и за рубежом.
11. Что такое «серая литература» в компьютерной сети?
12. Для чего предназначена поисковая система Нигма (ее возможности)?
13. Что такое интеллектуальные поисковые системы? Приведите примеры.
14. Перечислите основные элементы настройки браузера.
15. Что такое «облачные» технологии?

15 Конфедициальность медицинских данных. Защита медицинской информации. Антивирусные средства

Цель работы: изучение основных видов компьютерных преступлений и способы защиты медицинской информации.

Краткие теоретические сведения

Без должного внимания к вопросам обеспечения безопасности медицинских данных последствия перехода общества к новым информационным технологиям могут быть катастрофическими как для врачей, так и для пациентов. Неправомерное искажение или фальсификация, уничтожение или разглашение определенной части информации, равно как и дезорганизация процессов ее обработки и передачи в информационно-управляюших системах наносят серьезный материальный и моральный урон многим субъектам (государству, юридическим и физическим лицам), участвующим в процессах автоматизированного информационного взаимодействия. К сожалению, как и любое другое достижение человеческого гения, компьютер, решая одни технические, экономические и социальные проблемы, одновременно порождает и другие, порою не менее сложные. Если в должной мере не позаботиться о нейтрализации сопутствующих прогрессу негативных факторов, то эффект от внедрения новейших достижений науки и техники может оказаться в целом отрицательным.

Виды вредоносных программ

Рассмотрим основные виды вредоносных программ.

БЛОКИРОВЩИКИ WINDOWS

Блокировщики Windows – это вредоносные программы, которые, согласно классификации, именуются Trojan.Winlock. **Блокировщики Windows** при старте системы Windows выводят поверх всех окон сообщение о том, что доступ в систему заблокирован, и для того, чтобы данное окно исчезло, необходимо отправить платное СМС-сообщение. В качестве причины блокировки программа могла информировать о том, что на компьютере установлена якобы нелицензионная операционная система или другое программное обеспечение (реже используются другие «поводы»). Известны случаи распространения конструкторов данных вредоносных программ за определённую сумму – приобрести их мог любой желающий. В последнее время **блокировщики Windows** стали более агрессивными. СМС-сообщения для разблокировки существенно подорожали. Некоторые модификации могут и не содержать в себе правильного кода для разблокировки, и, соответственно, пользователь после отправки денег злоумышленникам остается ни с чем. Данные программы не удаляются автоматически из системы по прошествии некоторого времени. **Блокировщики Windows** научились препятствовать запуску множества программ, способных упростить исследование блокировщика на заражённой системе или просто завершающих работу системы при попытке запуска такого ПО.

В случае заражения системы очередной модификацией Trojan.Winlock не следует перенаправлять деньги злоумышленникам! В случае подобной атаки немедленно обратитесь в техподдержку используемого антивируса.

КОМПЬЮТЕРНЫЙ ВИРУС «ЧЕРВЬ»

Категория программ **компьютерный вирус червь** использует для распространения сетевые ресурсы. Черви проникают в компьютер, вычисляют сетевые адреса других компьютеров и рассылают по этим адресам свои копии. Помимо сетевых адресов часто используются данные адресной книги почтовых клиентов. Представители этого класса вредоносных программ иногда создают рабочие файлы на дисках системы, но могут вообще не обращаться к ресурсам компьютера (за исключением оперативной памяти).

Вирусы - программы, которые заражают другие программы - добавляют в них свой код, чтобы получить управление при запуске зараженных файлов. Это простое определение дает возможность выявить основное действие, выполняемое вирусом - заражение. Скорость распространения вирусов несколько ниже, чем у червей.

ВИРУС ТРОЯНСКАЯ ПРОГРАММА

Программы, которые выполняют на поражаемых компьютерах несанкционированные пользователем действия, т.е. в зависимости от каких-либо условий уничтожают информацию на дисках, приводят систему к «зависанию», воруют конфиденциальную информацию и т.д. **Вирус Троянская программа** (вопреки распространённому мнению) не является вирусом в традиционном понимании этого термина, т.е. не заражает другие программы или данные; троянские программы не способны самостоятельно проникать на компьютеры и распространяются злоумышленниками под видом «полезного» программного обеспечения. При этом вред, наносимый ими, может во много раз превышать потери от традиционной вирусной атаки. Внутри данной классификации можно выделить некоторые **типы вредоносных программ,** особо активные в последний период:

Лжеантивирусы

Они попадают под категорию Троянских программ - Trojan.Fakealert. Эти программы при запуске внешне похожи на настоящие антивирусные программы, но не являются таковыми. **Лжеантивирусы** имеют цель завлечь пользователя на специально подготовленный вредоносный сайт, на котором он должен приобрести якобы полную версию продукта. Как правило, **лжеантивирусы** распространяются в виде приложений к спам-письмам или через специально подготовленные вредоносные сайты. При этом чаще всего таким образом передаётся загрузчик лжеантивируса, который при запуске загружает с сервера злоумышленников компоненты, составляющие основной функционал. Упор во вредоносном ПО этого типа делается на визуальную часть – программа отображает системные окна Windows, которые сообщают о том, что данный антивирус якобы интегрирован в систему.

Основное окно программы показывает процесс сканирования компьютера и имитирует обнаружение вирусов.
После того, как пользователь заплатит деньги за якобы полную версию такого антивируса, его беды отнюдь не заканчиваются - он остаётся "на крючке", и в систему могут быть загружены какие-либо другие вредоносные объекты.

Рукиты

Эти вредоносные программы скрывают своё присутствие в системе, а также позволяют работать в скрытом от глаз пользователя и большинства антивирусов режиме другим вредоносным программам, которые они загружают с вредоносных интернет-сайтов. Например,**Рукиты** могут находиться в составе какой-либо другой вирусной программы или находиться в составе антивируса. Наиболее заметными вредоносными программами данного класса стало семейство BackDoor.Tdss (название приводится по классификации Dr.Web). За 2009 год компания "Доктор Веб" оперативно выпустила несколько горячих дополнений сканера с графическим интерфейсом, включающего в себя обновлённый антируткит-модуль Dr.Web Shield для противодействия новым руткит-технологиям. **Рукиты** – одна из последних модификаций – оснащены инструментами сокрытия в системе. К примеру, специально создаваемый зашифрованный виртуальный диск и механизм обхода некоторых типов поведенческих анализаторов.

Сетевые вирусы (Черви)

К данной категории относятся программы, распространяющие свои копии по локальным и/или глобальным сетям с целью:
* проникновения на удаленные компьютеры;
* запуска своей копии на удаленном компьютере;
* дальнейшего распространения на другие компьютеры в сети.

Для своего распространения сетевые черви используют разнообразные компьютерные и мобильные сети: электронную почту, системы обмена мгновенными сообщениями, файлообменные (P2P) и IRC-сети, LAN, сети обмена данными между мобильными устройствами (телефонами, карманными компьютерами) и т. д.
Большинство известных червей распространяется в виде файлов: вложение в электронное письмо, ссылка на зараженный файл на каком-либо Web- или FTP-ресурсе в ICQ- и IRC-сообщениях, файл в каталоге обмена P2P (Peer to Peer) и т. д.
Некоторые черви (так называемые "бесфайловые" или "пакетные" черви) распространяются в виде сетевых пакетов, проникают непосредственно в память компьютера и активизируют свой код.
Черви проникают в зараженные машины вполне естественным путем, без каких-либо действий со стороны пользователя. Они ближе всех остальных вирусов подобрались к модели своих биологических прототипов и потому чрезвычайно разрушительны и опасны. Их не берут никакие превентивные меры защиты, антивирусные сканеры и вакцины до сих пор остаются крайне неэффективными

средствами борьбы. Нашествие червей нельзя предвидеть и практически невозможно остановить.

Черви, в отличие от вирусов, для своего распространения активно используют протоколы и возможности локальных и глобальных сетей, поэтому они и называются сетевыми. Основным принципом работы сетевого вируса является возможность самостоятельно передать свой код на удаленный сервер или рабочую станцию. "Полноценные" сетевые вирусы при этом обладают еще и возможностью запустить на выполнение свой код на удаленном компьютере или, по крайней мере, "подтолкнуть" пользователя к запуску зараженного файла. Для внедрения в заражаемую систему червь может использовать различные механизмы: дыры, слабые пароли, уязвимости базовых и прикладных протоколов, открытые системы и человеческий фактор.

Для проникновения на удаленные компьютеры и запуска своей копии черви используют различные методы: социальный инжиниринг (например, текст электронного письма, призывающий открыть вложенный файл), недочеты в конфигурации сети (например, копирование на диск, открытый на полный доступ), ошибки в службах безопасности операционных систем и приложений. Так же для проникновения на компьютер жертвы, черви могу использовать специального сборщика. Это небольшая программа, которая сама не является вирусом. Сборщик забрасывается на поражаемый компьютер, а потом по частям скачивает червя из сети, собирает его и запускает. Так как червь проникает на компьютер по частям, антивирусные программы не могут его обнаружить.

Бытует ошибочное мнение, что сетевым является любой вирус, распространяющийся в компьютерной сети. Но в таком случае практически все вирусы были бы сетевыми, даже наиболее примитивные из них: ведь самый обычный нерезидентный вирус при заражении файлов не разбирается - сетевой (удаленный) это диск или локальный. В результате такой вирус способен заражать файлы в пределах сети, но отнести его к сетевым вирусам никак нельзя.

Сетевые вирусы прошлого распространялись в компьютерной сети и, как правило, так же как и компаньон - вирусы, не изменяли файлы или сектора на дисках. Они проникали в память компьютера из компьютерной сети, вычисляли сетевые адреса других компьютеров и рассылали по этим адресам свои копии. Эти вирусы иногда также создавали рабочие файлы на дисках системы, но могли вообще не обращаться к ресурсам компьютера (за исключением оперативной памяти).

Некоторые черви обладают также свойствами других разновидностей вредоносного программного обеспечения. Например, некоторые черви содержат троянские функции или способны заражать выполняемые файлы на локальном диске, т. е. имеют свойство троянской программы и/или компьютерного вируса.

Троянские программы

Данный тип вредоносных программ не является вирусами. Троян - это программа, предоставляющая удаленный доступ к чужому компьютеру,

позволяющая проводить различные манипуляции на нем, отсылать конфиденциальную информацию (пароли, номера кредитных карт, Интернет - аккаунты, логии компьютера и т.д.). Изначально троянский конь не несет в себе деструктивных функций, а предназначен для управления чужими компьютерами. Трояны, как правило, не заражают другие файлы, а являются автономными отдельными программами, которые после попадания на компьютер, копируют себя в системные папки под неприметными или заслуживающими доверия именами (winrun32dll.exe). Затем они прописывают себя в ветви реестра отвечающей за запуск программ при загрузке операционной системы (HKEY_LOCAL_MACHINE\SOFT WARE\Microsoft\Windows\CurrentVersion\Run), а также все разделы с названием Run, RunOnce, Runservices, RunservicesOnce. Современные троянские кони для своей анонимности и скрытности в системе используют стелс, и даже полиморфные технологии, что значительно затрудняет борьбу с ними.

Хотя троянские программы существуют довольно давно и отличаются большим разнообразием, но наиболее громкие события в этой области связаны с программой, называемой BackOrifice (англ. задний проход). Информация о BackOrifice впервые была опубликована 21 июля 1998 года. Группа хакеров, называющая себя "Культ мертвой коровы" - "Cult of the Dead Cow", создала троянца для Windows 95/98. Установленный на машине жертвы, BackOrifice позволял любому, знающему номер порта и пароль, выполнять на машине жертвы некие привилегированные операции как: выполнять команды операционной системы, просматривать файлы, скачивать и закачивать любые файлы, манипулировать регистром (registry), получать список активных процессов, завершать произвольный процесс, незаметно порождать новый процесс путем запуска сервиса, получать полную копию клавиатурного ввода и многое другое. Графический клиент BackOrifice был способен, кроме того, получать содержимое экрана жертвы. Попросту говоря, используя BackOrifice, можно делать с компьютером все. И даже более того, производить такие операции, о возможности которых обычный пользователь и не подозревает. Собственно, программа состоит из двух частей: сервера и клиента. Серверная часть внедряется на компьютер жертвы. После этого подключиться к компьютеру-жертве можно, используя клиентскую программу.

В будущем возможно появление таких троянских коней, которые буду прописывать себя в системные библиотеки или другие системные файлы.

Трояны делятся на три основных типа: Mail Senders, BackDoor, Log Writers или KeyLoger.

Mail Senders

Это - тип Троянов, работающих на основе отправки информации "хозяину". На данный момент это очень распространенный вид Троянов. С помощью такого типа "коней" люди, настроившие их могут получать по почте аккаунты Интернета, пароли ICQ, почтовые пароли, пароли к ЧАТам. И это в лучшем случае. В худшем же жертва даже не будет знать о том, что некто читает его

почту, входит в Интернет через его аккаунт, пользуется UIN'ом ICQ для распространения таких же Троянов пользователям контакт листа. MailSender никак не зависит от "хозяина", он живет своей жизнью в зараженном компьютере, так как в него все закладывается в момент настройки - Троян все выполняет по плану.

BackDoor

Если переводить дословно означает - задняя дверь или черный ход. Это - тип Троянов, функции которого включают в себя все, на что способен Троян типа Mail Sender, плюс еще десяток-другой функций удаленного администрирования (управления чужым компьютером с другой машины, например, через Интернет). Такой Троян ждет соединения со стороны клиента (неотъемлемая часть всякого трояна, с помощью которой посылаются команды на сервер). Трояны такого типа дают кому угодно полный доступ к зараженному компьютеру. После того как клиент на зараженной машине подключается к Интернету или локальной сети, троянская программа отправляет собранную информацию своему хозяину и открывает доступ к пораженному компьютеру (открывает определенные сетевые порты в системе и сообщает о них хозяину).

Бэкдоры делятся на два основных типа:

Локальный бэкдор - предоставит какие-то привилегии локально (например, на компьютере есть несколько учетных записей, но пользовательские аккаунты не имеют прав администратора. Подобного рода трояны как раз и предоставляют права администратора тому пользователю, которые его внедрил);

Удаленный бэкдор - может предоставить shell к машине удаленно.

Существует так же два вида предоставления shell-доступа: BindShell и Back Connect.

BindShell - самый распространенный, работает по архитектуре "клиент-сервер", то есть бэкдор ожидает соединение.

Back Connect - применяется для обхода брандмауэров, бэкдор сам пытается соединиться с компьютером хакера.

Log Writers или Key loggers

Это последний тип Тронов, из основных, копирующий всю информацию, вводимую с клавиатуры, и записывающий ее в файл, который впоследствии будет либо отправлен на определенный E-Mail адрес, либо просмотрен через FTP (File Transfer Protocol).

Первоначально программные продукты этого типа предназначались исключительно для записи информации о нажатиях клавиш клавиатуры, в том числе и системных, в специализированный журнал регистрации (log-файл), который впоследствии изучался человеком, установившим эту программу. В последнее время программные продукты, имеющие данное название, выполняют много дополнительных функций - это перехват информации из окон, перехват кликов мыши, экрана и активных окон, ведение учета всех полученных и отправленных писем, мониторинг файловой активности, системного реестра и очереди заданий, отправленных на принтер, перехват

звука с микрофона и видео с веб-камеры, подключенных к компьютеру и др. Кейлоггеры могут быть встроены в коммерческие, бесплатные и условно-бесплатные программы.

В свою очередь кейлоггеры можно разделить на пять типов:

1. Программы-шпионы, разрабатываемые под эгидой правительственных организаций (пример - продукт Magic Lantern, проект под названием Cyber Knight, США).
2. Программы-шпионы, которые могут создаваться разработчиками различных операционных систем и включаться ими в состав ядра операционной системы.
3. Программы-шпионы, которые созданы в ограниченном количестве (часто только в одной или нескольких копиях) для решения конкретной задачи, связанной с похищением критической информации с компьютера пользователя (например программы, применяемые хакерами-профессионалами). Такие программы могут представлять собой немного видоизмененные открытые исходные коды программ-шпионов, взятые из Интернета и скомпилированные самим хакером, что позволяет изменить сигнатуру программы-шпиона.
4. Коммерческие, особенно, корпоративные программные продукты, которые очень редко вносятся в сигнатурные базы, а если и вносятся, то только по политическим мотивам (пример - программные продукты таких известных фирм, как WinWhat-Where Corporation, SpectorSoft Cor-poration, ExploreAnywhere Software LLC, Omniquad, Ltd).
5. Программы-шпионы, представляющие собой keylogging-модули, входящие в состав программ-вирусов. До внесения сигнатурных данных в вирусную базу данные модули являются неизвестными. Пример - всемирно известные вирусы, натворившие много бед в последние годы, имеющие в своем составе модуль перехвата нажатий клавиатуры и отправки полученной информации в сеть Интернет.

Также есть еще два типа троянских программ, которые заслуживают отдельного внимания, это: Trojan-Dropper и Trojan-Downloader. Конечные цели у них абсолютно идентичны - установка на компьютер другой вредоносной программы, которая может быть как червем, так и "троянцем". Отличается только принцип их действия.

Trojan-Dropper

"Дропперы" могут содержать в себе уже известную вредоносную программу или наоборот - устанавливать новую ее версию. Также "дропперы" могут устанавливать не одну, а сразу несколько вредоносных программ, принципиально отличающихся по поведению и даже написанных разными людьми.

Фактически "дропперы" являются своеобразными архивами, внутрь которых может быть помещено все что угодно. Очень часто они применяются для установки в систему уже известных "троянцев", поскольку написать "дроппер" гораздо проще, чем переписывать "троянца", пытаясь сделать его недетектируемым для антивируса. Весьма значительную часть "дропперов"

составляют их реализации на скрипт-языках VBS и JS, что объясняется сравнительной простотой программирования на них и универсальностью подобных программ.

Trojan-Downloader

Даунлоадеры, или **загрузчики**, активно используются вирусописателями как по причинам, описанным выше для "дропперов" (скрытая установка уже известных троянцев), так и по причине их меньшего по сравнению с "дропперами" размера, а также благодаря возможности обновлять устанавливаемые троянские программы. Здесь также выделяется группа программ на скрипт-языках, причем, как правило, использующих различные уязвимости в Internet Explorer.

Оба эти класса вредоносных программ используются для установки на компьютеры не только троянских программ, но и различных рекламных (adware) или порнографических (pornware) программ.

RootKit

Термин RootKit исторически пришёл из мира Unix, и под этим термином понимается набор утилит, которые хакер устанавливает на взломанном им компьютере после получения первоначального доступа. Этот набор, как правило, включает в себя разнообразные утилиты для заметания следов вторжения в систему, хакерский инструментарий (снифферы, сканеры) и троянские программы, замещающие основные Unix утилиты. RootKit позволяет хакеру закрепиться во взломанной системе и сокрыть следы своей деятельности. На текущий момент руткиты зачастую с легкостью обнаруживаются и удаляются при помощи антивирусных пакетов.

Снифферы (Нюхачи)

Это программное обеспечение, которое позволяет просматривать содержимое сетевых пакетов, перехватывать трафик в сети и анализировать его.

DoS, DDoS - сетевые атаки

Программы данного типа реализуют атаки на удаленные сервера, посылая на них многочисленные запросы, что приводит к отказу в обслуживании, если ресурсы атакуемого сервера недостаточны для обработки всех поступающих запросов (DoS = Denial of Service).

DoS-программы реализуют атаку с одного компьютера с ведома пользователя. DDoS-программы (Distributed DoS) реализуют распределенные атаки с разных компьютеров, причем без ведома пользователя зараженного компьютера. Для этого DDoS-программа засылается любым способом на компьютер "жертв-посредников" и после запуска в зависимости от текущей даты или по команде от "хозяина" начинает DoS-атаку на указанный сервер в сети.

Некоторые компьютерные черви содержат в себе DoS-процедуры, атакующие сайты, которые по каким-либо причинам "невзлюбил" автор червя. Так, червь Codered 20 августа 2001 организовал успешную атаку на официальный сайт президента США, а червь Mydoom.a 1 февраля 2004 года "выключил" сайт SCO, производителя дистрибутивов UNIX.

7 мая 1997 года был обнародован принцип самой, пожалуй, нашумевшей DOS-атаки под названием WinNuke. Ее жертвами становились Windows-системы. Автор метода поместил его описание и исходный текст программы в несколько news-конференций. Ввиду его крайней простоты практически каждый человек мог вооружиться этим новейшим оружием. Очевидной первой жертвой стал www.microsoft.com. Данный сервер был выведен из строя более двух суток. Microsoft.com прекратил откликаться в пятницу вечером (9 мая) и только к обеду понедельника вновь обрел устойчивость.

Exploit, HackTool - взломщики удаленных компьютеров

Хакерские утилиты данного класса предназначены для проникновения в удаленные компьютеры с целью дальнейшего управления ими (используя методы троянских программ типа "backdoor") или для внедрения во взломанную систему других вредоносных программ.

Хакерские утилиты типа "exploit" при этом используют уязвимости в операционных системах или приложениях, установленных на атакуемом компьютере.

Nuker - фатальные сетевые атаки

Утилиты, отправляющие специально оформленные запросы на атакуемые компьютеры в сети, в результате чего атакуемая система прекращает работу. Используют уязвимости в программном обеспечении и операционных системах, в результате чего сетевой запрос специального вида вызывает критическую ошибку в атакуемом приложении.

Прочие вредоносные программы

К прочим вредоносным программам относятся разнообразные программы, которые не представляют угрозы непосредственно компьютеру, на котором исполняются, а разработаны для создания других вирусов или троянских программ, организации DoS-атак на удаленные сервера, взлома других компьютеров и т. п.

Flooder - "замусоривание" сети

Данные хакерские утилиты используются для "забивания мусором" (бесполезными сообщениями) каналов Интернета - IRC-каналов, компьютерных пейджинговых сетей, электронной почты и т. д.

Constructor - конструкторы вирусов и троянских программ

Конструкторы вирусов и троянских программ - это утилиты, предназначенные для изготовления новых компьютерных вирусов и "троянцев". Известны конструкторы вирусов для DOS, Windows и макро-вирусов. Они позволяют генерировать исходные тексты вирусов, объектные модули, и/или непосредственно зараженные файлы.

Некоторые конструкторы снабжены стандартным оконным интерфейсом, где при помощи системы меню можно выбрать тип вируса, поражаемые объекты, наличие или отсутствие самошифровка, противодействие отладчику, внутренние текстовые строки, выбрать эффекты, сопровождающие работу

вируса и т. п. Прочие конструкторы не имеют интерфейса и считывают информацию о типе вируса из конфигурационного файла.

Bad-Joke, Hoax - злые шутки, введение пользователя в заблуждение

К ним относятся программы, которые не причиняют компьютеру какого-либо прямого вреда, однако выводят сообщения о том, что такой вред уже причинен, либо будет причинен при каких-либо условиях, либо предупреждают пользователя о несуществующей опасности. К "злым шуткам" относятся, например, программы, которые "пугают" пользователя сообщениями о форматировании диска (хотя никакого форматирования на самом деле не происходит), детектируют вирусы в незараженных файлах, выводят странные вирусоподобные сообщения и т. д. - в зависимости от чувства юмора автора такой программы.

FileCryptor, PolyCryptor - скрытие от антивирусных программ

Хакерские утилиты, использующиеся для шифрования других вредоносных программ с целью скрытия их содержимого от антивирусной проверки.

PolyEngine - полиморфные генераторы

Полиморфные генераторы не являются вирусами в прямом смысле этого слова, поскольку в их алгоритм не закладываются функции размножения, т. е. открытия, закрытия и записи в файлы, чтения и записи секторов и т. д. Главной функцией подобного рода программ является шифрование тела вируса и генерация соответствующего расшифровщика.

Обычно полиморфные генераторы распространяются их авторами без ограничений в виде файла-архива. Основным файлом в архиве любого генератора является объектный модуль, содержащий этот генератор. Во всех встречавшихся генераторах этот модуль содержит внешнюю (external) функцию - вызов программы генератора.

VirTool

Утилиты, предназначенные для облегчения написания компьютерных вирусов и для их изучения в хакерских целях.

Социальный инжиниринг

Социальный инжиниринг не относится к вредоносным программам или вирусам. Он призвана запутать пользователей, усыпить их бдительность, замаскировать вредоносную программу под какой либо патч или полезную программу, заманить пользователя на заведомо зараженный сайт. В первую очередь на уловки соц. Инжиниринга попадаются не опытные пользователи всемирной паутины, но в последнее время вирусописатели идут на всевозможные хитрости, что даже опытные пользователи не всегда способны распознать уловку от правды. Примерами данной технологии являются: всевозможные письма с текстом открыть прикрепленный к письму файл или зайти на сайт по указанной ссылке, для скачивания обновлений, патчей, просьбы якобы от службы технической поддержки или администраторов почтового сервера отослать свой пароль к вашему электронному ящику и пр.. Так, например червь Swen, выдавал себя за специальный патч от компании

Microsoft, якобы устраняющий все известные уязвимости. Письмо, содержало легко узнаваемые элементы официального сайта Microsoft, ссылки на другие ресурсы данной компании и грамотно составленный текст. Социальный инжиниринг активно используется не только в вирусах, но и самими хакерами для получения конфиденциальной информации от пользователей.

Условно опасные программы

Иногда антивирус не может без помощи пользователя определить, опасна или нет та или иная программа. Некоторые программы обладают набором функций, которые могут причинить вред пользователю только при выполнении ряда условий. Более того, подобные программы могут легально продаваться и использоваться в повседневной работе, например, системных администраторов. Однако в руках злоумышленника такие программы могут обернуться инструментом, с помощью которого можно причинить вред ничего не подозревающему пользователю.

В настоящее время к условно опасным программам **Лаборатория Касперского** относит программы классов Riskware, Adware и Pornware.

Riskware

К классу программ Riskware относятся легальные программы (некоторые из них свободно продаются и широко используются в легальных целях), которые, тем не менее, в руках злоумышленника способны причинить вред пользователю и его данным.

В списке программ класса Riskware можно обнаружить легальные утилиты удаленного администрирования, программы-клиенты IRC, звонилки-дайлеры, скачиватели-даунлоадеры, мониторы любой активности, утилиты для работы с паролями, а также многочисленные интернет-серверы служб FTP, Web, Proxy и Telnet. Все эти программы не являются вредоносными сами по себе, однако обладают функционалом, которым могут воспользоваться злоумышленники для причинения вреда пользователям. Возьмем, например, программу удаленного администрирования WinVNC. Данная программа позволяет получать доступ к интерфейсу удаленного компьютера и используется для удаленного управления и наблюдения за удаленной машиной. Таким образом, данная программа является легальной, свободно распространяемой и необходимой в работе добропорядочных системных администраторов или других технических специалистов. В качестве другого примера утилита mIRC. Это легальная программа, являющаяся клиентом IRC-сети. Расширенным функционалом утилиты mIRC могут воспользоваться злоумышленники - регулярно появляются троянские программы (в частности, бэкдоры), использующие функции mIRC в своей работе. Так, любой IRC-бэкдор способен без ведома пользователя дописать в файл конфигурации mIRC собственные скрипты и успешно выполнить свои деструктивные функции на пораженной машине. При этом пользователь mIRC не будет даже подозревать о функционировании на его компьютере вредоносной троянской программы. Зачастую вредоносные программы самостоятельно устанавливают на пользовательский компьютер

клиент mIRC для последующего использования его в собственных целях. В качестве места размещения mIRC в этом случае, как правило, выступает папка Windows и ее подпапки. Обнаружение mIRC в этих папках практически однозначно свидетельствует о факте заражения компьютера какими-то вредоносными программами.

Adware

Рекламное программное обеспечение (Adware, Advware, Spyware, Browser Hijackers) предназначено для показа рекламных сообщений - чаще всего, в виде графических баннеров - и перенаправления поисковых запросов на рекламные веб-страницы. За исключением показов рекламы, подобные программы, как правило, никак не проявляют своего присутствия в системе - отсутствует значок в системном трее, нет упоминаний об установленных файлах в меню программ. Зачастую у Adware-программ нет процедуры деинсталляции.

Проникновение

На компьютеры пользователей Adware попадает двумя способами:

- путем встраивания рекламных компонентов в бесплатное и условно-бесплатное программное обеспечение (freeware, shareware);
- путем несанкционированной установки рекламных компонентов при посещении пользователем "зараженных" веб-страниц.

Большинство программ freeware и shareware прекращает показ рекламы после их покупки и/или регистрации. Подобные программы часто используют встроенные Adware-утилиты сторонних производителей. В некоторых случаях эти Adware-утилиты остаются установленными на компьютере пользователя, и после регистрации программ, с которыми они изначально попали в операционную систему. При этом, удаление Adware-компонента, всё еще используемого какой-либо программой для показа рекламы, может привести к сбоям в функционировании этой программы.

Базовое назначение Adware данного типа - неявная форма оплаты программного обеспечения, осуществляемая за счет показа пользователю рекламной информации (рекламодатели платят за показ их рекламы рекламному агентству, рекламное агентство - разработчику Adware). Adware помогает сократить расходы как разработчикам программного обеспечения (доход от Adware стимулирует их к написанию новых и совершенствованию существующих программ), так и самим пользователям.

В случае установки рекламных компонентов при посещении пользователем "зараженных" веб-страниц в большинстве случаев используются хакерские технологии: проникновение в компьютер через дыры в системе безопасности интернет-браузера, а также использование троянских программ, предназначенных для скрытой установки программного обеспечения (Trojan-Downloader или Trojan-Dropper). Adware-программы, действующие подобным образом, часто называют **Browser Hijackers**.

Доставка рекламы

Известны два основных способа доставки рекламной информации:

- скачивание рекламных текстов и изображений с веб- или FTP-серверов, принадлежащих рекламодателю;
- перенаправление поисковых запросов интернет-браузера на рекламный веб-сайт.

Перенаправление запросов в некоторых случаях происходит только при отсутствии запрашиваемой пользователем веб-страницы, т.е. при ошибке в наборе адреса страницы.

Утечка информации

Многие рекламные системы помимо доставки рекламы также собирают конфиденциальную информацию о компьютере и пользователе:

- IP-адрес компьютера;
- версию установленной операционной системы и интернет-браузера;
- список часто посещаемых пользователем интернет-ресурсов;
- поисковые запросы;
- прочие данные, которые можно использовать при проведении последующих рекламных кампаний.

По этой причине Adware-программы часто называют **Spyware** (не следует путать рекламное Spyware с троянскими шпионскими программами Trojan-Spy).

Pornware

К программам класса Pornware, по мнению экспертов **Лаборатории Касперского**, относятся утилиты, так или иначе связанные с показом пользователям информации порнографического характера. На данный момент в классе выделяется три поведения: Porn-Dialer, Porn-Downloader и Porn-Tool. Дайлеры дозваниваются до порнографических телефонных служб, а даунлоадеры скачивают на пользовательский компьютер порнографические материалы. К последнему поведению класса Pornware относятся всевозможные утилиты, так или иначе связанные с поиском и показом порнографических материалов (например, специальные панели инструментов для интернет-браузера и особые видеоплееры). Программы класса Pornware могут быть установлены пользователем на свой компьютер сознательно, с целью поиска и получения порнографической информации. В этом случае они не являются вредоносными. С другой стороны, те же самые программы могут быть установлены на пользовательский компьютер злоумышленниками - через использование уязвимостей операционной системы и интернет-браузера или при помощи вредоносных троянских программ классов Trojan-Downloader или Trojan-Dropper. Делается это обычно с целью рекламы платных порнографических сайтов и сервисов, на которые пользователь сам по себе никогда не обратил бы внимания.

Антивирусные программы.

Наличие антивируса на современном компьютере или ноутбуке – вынужденная необходимость, которую никак нельзя обойти стороной при установке программного обеспечения на компьютер. Цель антивирусной программы, или антивируса, – обнаружение компьютерных вирусов, а также вредоносных

программ, которые, как вариант, блокируют нормальною работу компьютера, а также антивирусы являются способом профилактики, защиты файлов и операционной системы от вредоносных кодов программ-вирусов. Во-первых, антивирусы можно подразделить на две категории: программы, которые непрерывно сканируют потоки данных, например, интернет-трафика, принудительно запускаемые программы с целью сканирования определенных, указанных объектов. Во-вторых, антивирусные программы различаются по виду (способу) защиты от вирусов. Тут можно выделить следующие: Программы-детекторы, или сканеры, находят вирусы в оперативной памяти и на внешних носителях, выводя сообщение при обнаружении вируса. Программы-доктора, (фаги, программы-вакцины) находят зараженные файлы и "лечат" их. Среди этого вида программ существуют полифаги, которые способны удалять разнообразные виды вирусов, самые известные из антивирусов-полифагов Norton AntiVirus, Doctor Web, Kaspersky Antivirus. Программы-ревизоры являются наиболее надежными в плане защиты от вирусов. Ревизоры запоминают исходное состояние программ, каталогов, системных областей диска до момента инфицирования компьютера, затем сравнивают текущее состояние с первоначальным, выводя найденные изменения на дисплей. Программы-мониторы (файерволы, брандмауэры) начинают свою работу при запуске операционной системы, постоянно находятся в памяти компьютера и осуществляют автоматическую проверку файлов по принципу "здесь и сейчас". Программы-фильтры (сторожа) обнаруживают вирус на ранней стадии, пока он не начал размножаться. Программы-сторожа - небольшие резидентные программы, целью которых является обнаружение действий, характерных для вирусов.

Главная задача антивирусных средств - обнаружить вирус. При этом возможны ошибки первого рода (несрабатывание) и второго рода (ложная тревога).

Программы-детекторы определяют наличие известных вирусов в загрузочных секторах, файлах и оперативной памяти, но не удаляют их. Эти программы практически не подвержены ошибкам второго рода. Эти программы используют два алгоритма поиска.

-Первый из них основан на поиске сигнатур или регулярных выражений, имеющихся во всех экземплярах вируса и нигде более.

-Второй проверяет область программы в начале файла, например, на наличие перехода на определенное смещение от конца файла, что нередко делают вирусы.

Программы-фаги (полифаги, доктора, дезинфекторы). Эти программы кроме обнаружения вирусов восстанавливают (т.е. "лечат") диски и файлы, если это возможно.

Полифаги имеют существенный недостаток, заключающийся в том, что им известно хоть и очень большое, но конечное число вирусов. Если в вашу машину попал вирус, неизвестный фагу, то фаг будет бессилен вам помочь и даже не предупредит об опасности. В этих случаях следует использовать другие

антивирусные программы, например ревизоры.
Ревизоры. Они основаны на сравнении контрольной информации о файлах, загрузочных сектора и др., сохраненной ранее, с текущей информацией. Ревизоры устойчивы к ошибкам второго рода.
Хороший ревизор, запускаясь с некоторой периодичностью (как правило, раз в сутки), сохраняет информацию обо всех уязвимых с точки зрения вирусов компонентах вашего компьютера. Вирусы не всесильны. Каким-либо образом они проявляют себя, и дело ревизора отследить подозрительные изменения конфигурации.
Доктора-ревизоры. Эти программы в случае обнаружения изменений могут вернуть объект изменения к исходному состоянию.
Эвристические анализаторы. Они используются для борьбы с полиморфными вирусами, перед которыми фаги бессильны.
Сторожа (или программы-фильтры). Это резидентные программы, хранящиеся в оперативной памяти, цель которых - вообще не пропустить вирус на компьютер, контролируя подозрительные действия, которые могут быть вызваны вирусами: обращения к дискам, попытки производить запись в определенную группу файлов, некоторые сектора дисков и т.д. Однако многие из этих программ не удобны из-за постоянных вопросов типа: "Разрешать запись в такой-то файл?".
Иммунизаторы (программы-вакцины) изменяют программы и диски таким образом, что это не отражается на их нормальной работе, но вирусы считают их уже зараженными и больше не заражают. Однако эти программы малоэффективны и нашли ограниченное применение. Программы-приманки входят в состав некоторых антивирусных продуктов. Эти программы периодически выполняются. После выполнения приманки антивирус сканирует ее, проверяя, не заражена ли она.
Известно, что периодическое сканирование ПК и дискет снижает потенциальный ущерб на 50 %, наличие резидентной антивирусной программы - на 95 %. Эффективной является многоуровневая стратегия защиты от вирусов: детекторы, сторожа, ревизоры (в зависимости от степени риска).
Список бесплатных антивирусных программ, для дополнительной проверки компьютера на наличие вирусов. Каждая из программ имеет свою антивирусную базу данных, поэтому полезно будет проверить свой компьютер на вирусы бесплатно хотя бы частью этих утилит.
Каждая из представленных программ поможет проверить компьютер на вирусы в реальном времени самой свежей антивирусной базой - ежедневное обновление.
1. **Malwarebytes' Anti-Malware** - антивирусная программа для быстрой проверки системы и поиска различного вредоносного ПО.
2. **AVG Anti-Virus Free** - (русский) - популярный антивирус, в случае домашнего применения используется бесплатно. Основные преимущества, быстрой, простой, не требователен к ресурсам.

3. Dr.Web CureIt! - Бесплатная утилита Dr.Web CureIt! Лучшее средство для проверки Вашего компьютера и лечения от вирусов реальном времени.
3. NOD32 On-DemandScanner - антивирусный сканер бесплатная версия антивируса NOD32, может быстро и легко обнаружить и удалить вирусы, интернетовские черви, троянских коней и другие вредные программы.
4. **Avast! AntivirusHome** - бесплатный антивирусный сканер. Использует резидентный и обычный сканеры, сканирует архивов, проверяет почту, интегрируется в систему Windows, русский.
5. Panda Cloud Antivirus Free - бесплатный антивирус для защиты от вирусов шпионов, руткитов и вредоносного ПО. Использует облачные технологии поиска вирусов.
6. Norton Internet Security - бесплатная подписка в течении месяца на один год. NIS-360 имеет антивирус, файрвол и нужные инструменты от всех видов угроз.
7. Malicious Software Removal Tool - официальное бесплатное средство для борьбы с вредоносными программами для в Windows. Проверяет компьютеры на присутствие популярных вредных программ, таких как Blaster, Sasser и Mydoom.
8. **Immunet Protect** - бесплатный облачный антивирус. Преимущества перед коллегами: не конфликтует с другим антивирусным, не большие размеры, высокая скорость, постоянные обновления базы.
9. **Microsoft Security Essentials** - бесплатная антивирусная программа, имеет все необходимые средства защиты компьютера. Простая в использовании.
10. Comodo AntiVirus - тоже простой в эксплуатации бесплатный антивирус, обладает средствами борьбы с вирусами, интернетными червями и троянами.
11. ThreatFire Free Edition - бесплатно и комплексно защитит компьютер, не любит злоумышленников распространяющих программы шпионы типа Spyware. Знает и удаляет все популярные вирусы.
12. Panda Antivirus Pro - очень эффективный файрвол и антивирус одновременно. Бесплатен в течении 3 месяцев со дня активации. Полноценная защищает и удаляет вирусы и другую грязь с компьютера.
13. Avira AntiVirPersonal - надежно находит и удаляет вирусы, трояны и неизвестные макровирусы - бесплатен для домашнего использования.
15. **PC Tools AntiVirus Free Edition** - полноценная бесплатная антивирусная программа. Отличается от платной тем, что не осуществляется оперативная техническая поддержка.
16. Panda Internet Security - пользуйтесь бесплатно 3 месяца. Включает в себя прекрасный антивирус и файрвол, присутствует резервное копирование, фильтр антиспама, фильтр веб-сайтов.
17. My Free Antivirus - бесплатное использование этого антивирусного пакета доставит вам удовольствие. Отлично защищает и лечит компьютер от вредоносных программ и файлов.
18. Kaspersky Security Suite CBE - бесплатная версия защиты от Касперского,

является приложением журнала Computer Bild. Имеет комплексную защиту в которую входят: антивирус, фаервол, антишпион.
19. **Portable ClamWin** - антивирус предназначен специально для работы с переносными носителями информации. Портативная версия является прекрасным защитником, не уступающим в функциональности полноценным программам.
20. ClamWin Free Antivirus - свободный к бесплатному использованию антивирусный сканер для Microsoft Windows. Имеет планировщик, постоянно обновляет вирусные базы, интегрирован в контекстное меню.
21. **BitDefender Free Edition** – новое бесплатное средство защиты компьютера, торопитесь опробовать, пока версия бесплатная.
22. **Multi Virus Cleaner** - еще один бесплатный способ проверки вашего компьютера на присутствие вирусов.
23. **Clam AntiVirus** - бесплатный, но мощный антивирус с открытым исходным кодом, портирован на Windows с UNIX, Linux.
24. **Simple Machine Protect** - бесплатный портативный антивирус. Удаляет Spyware и все известные виды троянов, вирусов, червей, шпионов.
25. **Rising Antivirus Free Edition** - дает возможность полной защиты от всех видов вирусов. Является бесплатным антивирусом.
26. **YandexOnlineKav** - Антивирус Касперского для Яндекс.Онлайн. Бесплатный антивирус от Kaspersky. Спец. версия, имеет ряд ограничений функций и распространяется вместе с Я.Онлайн.
27. **Dr.Web Trojan.Encoder** - бесплатная утилита, которая расшифровывает поврежденные файлы после атаки троянов.
28. **Hazard Shield** - быстрое, безопасное удаление вирусов, троянов, spyware, malware и другой «нечисти».
29. AVL Mini - не смотря на название является мощным антивирусом. Имеет не подкупный фаервол и spyware монитор, распространяется бесплатно.
30. **RunScanner** - бесплатная программа, которая находит все запускаемые объекты и приложения, контролирует автозапуск, ненужные в работе драйвера. Борется со шпионами и рекламой.
31. **Free eScanAV Anti-Virus Toolkit** (MWAV) - бесплатная утилита, не требует установки и может запускаться непосредственно из любого места, на ваш компьютер, USB Drive илиCD-ROM. Поддерживает все версии Windows, включая Win8.

Порядок выполнения работы:
1. Изучите теоретический материал.
2. Классифицировать основные вредоносные программы, способы их проникновения на компьютер и деструктивное действие.
2. Разработать схему с указанием известных Вам вредоносных программ, например:

3. Проанализировать преимущества и недостатки различных антивирусных программ (не менее четырех) и представьте результаты в виде таблицы, в которой обязательно указать: название, степень доступности, платформы, особенности (преимущества и недостатки).
4. Оформить отчет, включающий результаты выполнения работы и краткие ответы не менее чем на пять контрольных вопросов.

Контрольные вопросы

1. Какие виды компьютерных преступлений могут наблюдаться при обработке медицинской информации?
2. Какие Вы знаете вредоносные программы?
3. Что означает несанкционированный доступ?
4. Приведите пример системы паролей для обеспечения конфиденциальности данных.
5. Что такое электронно-цифровая подпись?
6. Как технически обеспечивается вопрос конфиденциальности и защиты данных?
7. Кто имеет полный доступ к данным больного?
8. Что такое компьютерные вирусы? Чем они отличаются от биологических?
9. Методы защиты компьютерной информации.
10. Как используются искусственные иммунные сети в антивирусных программах?

Описание вредоносных программ – серверы: http://www.stopinfection.narod.ru, http://hackers100.narod.ru, http://broxer.narod.ru,http://www.viruslist.com, http://logic-bratsk.ru, http://www.offt.ru, http://www.almanet.info

16. Синтез, анализ и проверка диагностических решающих правил

Цель работы: овладение навыками структурно-параметрической идентификации диагностических правил с применением средств вычислительной техники.

Краткие теоретические сведения.

Под диагностическими правилами понимается процедура вывода заключения о соотнесении состояния анализируемого объекта или процесса к определенному классу или области на основании временно-пространственной регистрации существенных характеристик.

Любой объект (процесс) с точки зрения диагностики подвергается анализу со стороны исследователя, который, как правило, априори знает, какие существенные характеристики ему следует регистрировать для решения диагностической задачи. То есть, в этом случае, исследователь уже владеет набором диагностических правил, которые либо опровергают, либо подтверждают выдвинутую им рабочую гипотезу о состоянии объекта. Так как о каждом состоянии объекта может выдвигаться различное количество гипотез, то, следовательно, диагностические правила каждой из них не должны в случае объединения поглощать друг друга, и, вообще говоря, должны иметь минимальное количество пересечений как по регистрируемым параметрам, так и по диапазонам их изменений.

В общем случае диагностическое правило имеет вид, например, продукции: если значение $P=P_0$, то состояние $S=S_0$.

$P=F(S, t, dS)$, (1)

где S - состояние; t - время; dS - диагноз изменения характеристик состояния.

Если зависимость (1) достаточно хорошо идентифицирована (с заданной степенью точности или неопределенности), то нетрудно построить эксперто-диагностическую систему продукционного типа с указанием исследователю технологии реализации необходимой информации для достаточно достоверной диагностики гипотетического состояния.

Рассмотрим **логический механизм** синтеза правила (1).

1 этап. Организация мониторинга состояния заданной глубины и полноты.

2 этап. Выделение множества ортогональных и информативных признаков с точки зрения вариативности. То есть, с одной стороны, селектируем сильно коррелированные характеристики, с другой стороны, отбираем те из них, вариативность которых (отношение дисперсии к среднему значению) выше определенного порогового уровня (например 10%).

3 этап. Кодируем состояния (лучше в двоичном коде): с учителем - то есть исследователь знает состояния, без учителя - выполняется кластер-анализ и задаются состояния или вводится пороговый принцип. Таким образом, получаем значения «логической» функции $Y=(Y_{i1},Y_{i2},Y_{il})$. Если состояний

не много, то рекомендуется применять унитарное кодирование с минимизацией Хеменгового расстояния соседних состояний.

4 этап. Кодируем значение признакового пространства, следующим образом (во всех случаях рекомендуется унитарный код). По каждому оставленному признаку выделяем определенный набор состояний, как попадание значения признака в определенный диапазон. Диапазон определяется либо:

1) Экспертом, исходя из его знаний и жизненного опыта.

2) Исследователем, по анализу частоты распределений значений и личного опыта. При достаточно небольшом количестве признаков анализ гистограммы рекомендуется проводить визуально, наблюдая все признаки одновременно (в концепции системный подход).

3) Автоматически (с применением ЭВМ) по следующему алгоритму.

Исследователь задает количество состояний по каждому признаку n_i (каждое из них кодируется, желательно в унитарном коде). Определяется медиана Mo и дисперсия G_o. Определяется удельное отклонение как $G_y = G_o/(n_{i-1})$. В качестве первого диапазона (состояния) выбирается величина внутри диапазона $M_o \pm G_o$. Все значения X_i попавшие в данный диапазон кодируются определенным состоянием S_o. Величина n_i декрементируется и повторяется описанный процесс над «оставшимися» данными. Так продолжается до тех пор, пока n_i не станет равно 0 и всем оставшимся значениям будет присвоено состояние S_n. Граничные значения $M_o \pm G_{yo}$ либо включаются в одно из состояний, либо, что более оптимально, кодируются знаком переходной функции.

5 этап. Определяем функциональные зависимости между полученными булевыми функциями (парные и множественные) и парное Хеминговое расстояние. Те признаки, у которых это расстояние равно нулю, селектируются путем оставления одного из них с наибольшей вариативностью.

Явный вид логической зависимости между булевыми переменными X_k, k=1,m определяются следующим образом. На первом шаге проверяются условия независимости: поскольку каждая булева функция может иметь два значения истинности, то m булевых функций может образовывать 2^m комбинаций значений истинности. Согласно определению m-булевых функций независимы, если в совокупности при всех возможных значениях аргументов они могут принимать 2^m комбинаций значений истинности. Т.е для проверки независимости необходимо вычислить их изображающие числа и проверить, образуют ли они полный набор чисел. Если да, то функции независимы, в противном случае - зависимы.

На втором шаге в базисе булевых функций выписывают в последовательные строки изображающие числа и определяют какие числа отсутствуют в наборе столбцов (повторяющиеся значения чисел считают один раз). Столбцы набора представляют собой комбинации значений истинности функций $X_i,..., X_m$, при которых соответствующие элементарные произведения составленные из $X_i,..., X_m$ истинны.

Таким образом, если идентифицируется зависимость:

$F(X_i,...,X_n)=1$ (2),

то, следовательно, имеющиеся в наборе столбцы указывают номера тех колонок базиса в $(X_1,...,X_n)$, которые совпадают с номерами изображающего числа $\#F(X_1,...,X_m)$, на которых функция F истинна.

Например, пусть задан протокол мониторинга трех логических функций:

X_1 11001010
X_2 10101100
X_3 11001100

Выпишем последовательно все столбцы в этом наборе изображающих чисел как строки и укажем справа их десятичные значения:

111=7, 101=5, 010=2, 000=0, 111=7, 110=6, 001=1, 000=0

Видно, что десятичные эквиваленты 3 и 4 отсутствуют, а это означает, что по отношению и в (X_1, X_2, X_3) изображающее число связи $F(X_1, X_2, X_3) = 1$ имеет вид $\#F(X_1, X_2, X_3) = 1$.

Минимизируя полученную функцию, получаем:

$$\#F=\bar{X}1\bar{X}3+X2X3+X1\bar{X}2=1$$

Проверяем:

X1	X2	$\bar{X}3$	$X1\bar{X}3$	X2X3	$X1\bar{X}2$	F
1	1	1	0	1	0	1
1	0	1	0	0	1	1
0	1	0	1	0	0	1
0	0	0	1	0	0	1
1	1	1	0	1	0	1
0	1	1	0	1	0	1
1	0	0	0	0	1	1
0	0	0	1	0	0	1

Таким образом, определяется как логические функции связаны между собой.

6 этап. Идентифицируем логические функции Y=F(X) - парная зависимость и/или Y=F({X}) (3) - множественная зависимость. Заметим, что возможен вариант отсутствия тех или иных функциональных зависимостей.

7 этап. Переходим от полученных булевских функций либо к продукционным диагностическим правилам, либо к схемотехническому решению идентфикационного диагностического устройства. Однако, второй вариант менее устойчив и мобилен в случае достаточно быстрого изменения окружающей среды, приводящего к изменению в функционировании анализируемого объекта (системы, процесса), а, следовательно, и вида идентифицированных функций.

Как и во множественном регрессионном анализе, при синтезе зависимостей (3) для получения более строгого результата (минимизации пересечений понятий в диагностических, классификационных правилах каждого состояния) рекомендуется руководствоваться правилом максимальной

организации (независимости) факторного пространства. Для этого необходимо добиться максимальной независимости X между собой, т.е. в идеале не должно существовать функциональных зависимостей между X_i. Т.е., если на пятом этапе идентифицируется F(x)=1 , то необходимо изменить множество X: либо путем исключения переменных (по критерию вариативности), что чревато в общем случае, потерей информации; либо изменить кодирование вводимых сигналов путем уменьшения количества состояний и/или изменения (экспертным путем) диагностических классов состояний. При достаточно мощной вычислительной технике и сравнительно небольшом размере факторного пространства (до 100 признаков) эти проблемы могут быть решены переборным путем. В противном случае, следует применять методы целенаправленного случайного поиска.

Как и в регрессионном анализе возможно формирование продукционных диагностических правил с учетом фактора запаздывания.

Одним из основных предназначений медицинской экспертной системы является функционирование в системе поддержки принятия решений на различных этапах лечебно-диагностического процесса, в частности – рекомендации в соотнесении состояния пациента к определенной нозологической группе.

Качество работы экспертной системы в этом случае определяется как уверенность предлагаемого решения. Существует множество различных подходов в оценки указанной уверенности в зависимости от: характера анализируемой информации, представительности (репрезентативности) обучающей и экзаменационной выборок на этапе проектирования экспертной системы, полноты и достоверности анализируемой информации, опыта пользователя системы и т.п.

Для оценки уверенности в правильности рекомендуемых диагностических решений медицинская экспертная система на конечном этапе проектирования тестируется на репрезентативной выборке экспериментального материала путем вычисления специальных статистических показателей качества, отражающих в себе значения ошибок первого и второго рода в правильности рекомендованных решений.

Наибольшее распространение указанного тестирования получил следующий подход.

Диагностика определенной нозологической группы проводится либо относительно некоторого класса условно здоровых людей (класс А) и класса, характерного для определенной нозологии (класс Б). (В случае дифференциальной диагностики осуществляется сравнение между различными нозологиями – в этом случае под «классом А» понимается одна из нозологий, под «классом Б» - другая).

По результатам тестирования составляется таблица вида:

Таблица 1 - Распределения результатов диагностики

Обследуемые	«золотой стандарт» (истина)		Всего
	болен	здоров	
Болен (положительный результат теста)	Истинно-положительный результат a	Ложно-положительный результат b	a+b
Здоров (отрицательный результат теста)	Ложноотрицательный результат c	Истинно-отрицательный результат d	c+d
Всего	a+c	b+d	a+b+c+d

В качестве показателей качества, характеризующих статистическую достоверность медицинской экспертной системы, при первичной оценки качества ее работы, выбираются: диагностическая чувствительность (ДЧ), диагностическая специфичность (ДС), прогностическая значимость положительных ($ПЗ^+$) и отрицательных ($ПЗ^-$) результатов испытаний и диагностическая эффективность (ДЭ).Указанные показатели качества рассчитываются в соответствии со следующими выражениями:

$$ДЧ = \frac{a}{a+c}, ДС = \frac{d}{d+b}, ДЭ = \frac{a+d}{a+b+c+d}, ПЗ^+ = \frac{a}{a+b}, ПЗ^- = \frac{d}{c+d},$$

где: a – истинно положительный результат равный количеству пациентов из класса заболеваний Б правильно классифицируемых;

b – ложноположительный результат равный количеству относительно здоровых людей класса А ошибочно отнесенных экспертной системой к классу Б;

c – ложноотрицательный результат равный количеству людей из класса Б отнесенных экспертной системой к классу А;

d – истинно отрицательный результат равный количеству людей из класса А правильно классифицированных экспертной системой.

При проектировании медицинской экспертной системы Заказчиком проекта для рассмотренных показателей качества определяются определенные пороговые значения, превышение которых говорит о приемлемом для эксплуатации качестве и, следовательно, возможности применения ЭС в системе поддержке принятия решения в лечебно-диагностическом процессе.

Порядок выполнения работы.

1. Самостоятельно изучите теоретический материал.
2. Согласно номеру варианта (равен порядковому номеру студента в журнале группы) N сформируйте протокол мониторинга наблюдения за процессом X, следующим образом:

Индекс переменной X	Индекс протокола Z (см. таблицу 1)
1	mod(n,8)+1
2	mod(n,8)+2
3	mod(n,8)+3

4 ¦ mod(n,8)+4

3. Выберите в качестве выходной величины, определяющей состояние процесса переменную мониторинга с индексом 4.

4. Зададитесь числом состоянием по Y- 3, X1-5, X2- 3, X3 -2.

5. Определите диапазон изменений состояний, причем для Y_1,X_1,- автоматически по дисперсии (здесь и далее рекомендуется использовать интегрированные среды типа EXCEL, STATISTICA), X_2 - экспертным путем анализа гистограммы, X_3 - экспериментальным путем заданием одного порога (<,>=).

6. Закодируйте состояние по X, У - т.е. получите характеристические числа (вид) булевых функций $X_б$ и $У_б$.

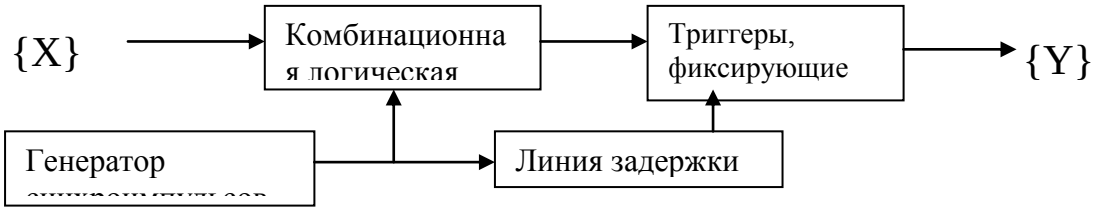
7. Проанализируйте взаимозависимость между булевскими факторами X и сформируйте наиболее ортогональное факторное пространство (векторы которого в наименьшей степени зависимы между собой).

8. Идентифицируйте в минимальном виде (с помощью карт Карно) логические функции $У_б=F(X_б)$.

9. Перейдите от булевского представления к логико-семантическому и сформулируйте диагностические правила продукционного типа.

10. Проверьте качество полученных правил вычислив диагностические специфичность, чувствительность и эффективность.

10. Составьте схемотехническое решение диагностических правил по следующей структуре:



Рассчитайте скважность синхроимпульсов и их характер для обеспечения устойчивой работы схем, считая время срабатывания любого логического элемента 10мс, время срабатывания триггера 50мс.

11. Оформите отчет с указанием последовательности своих действий, необходимых комментариев и выводов, кратких ответов не менее чем на 4 вопроса, аннотацию информационных источников, указанных в библиографии.

Контрольные вопросы:

1. Что определяет решающее правило?
2. Какие типы решающих правил применяют в диагностическом процессе при обработке результатов мониторинга?
3. В чем заключается логический способ синтеза решающего правила?
4. Каким образом осуществляется бинарное кодирование признакового пространства при синтезе логических решающих правил?

5. Как формулируется решающее правило продукционного типа?
6. Как осуществляется семантическое описание решающего правила?
7. Каким образом реализуется схемотехническая реализация решающего правила на определенной электронной базе?
8. Как проверяется качество применения решающего правила?

Результаты регионального мониторинга

год	Всего родилось	Всего заболело	Врожденные пороки (ВП)	асфаксия	Умерло всего	Умерло от ВП	Умерло от асфиксии
1	1657	90	22	6	17	4	5
2	2081	170	24	9	32	3	9
3	2173	201	20	25	32	7	5
4	2676	198	41	17	34	5	5
5	2557	191	51	47	21	3	10
6	2522	586	83	78	23	1	3
7	2893	252	30	19	31	3	7
8	2956	270	45	8	32	6	5
9	2650	197	38	12	25	3	6
10	3036	213	42	36	36	4	12
11	3165	230	37	32	21	2	5
12	3181	218	61	42	27	8	3
13	2930	216	65	58	18	1	6
14	2491	202	41	55	20	2	5
15	2964	185	37	39	25	2	1
16	2425	290	65	87	22	2	7
17	2432	238	50	53	19	3	4
18	2388	196	34	65	19	0	9
19	2290	197	34	58	22	4	9
20	2995	193	45	27	28	5	7

Год	Ревм. пораж. сердца	Инфаркт миокарда	Гипертан. болезнь	стенокардия	Септический эндокартит	летальность
1	107	8	171	30	5	18
2	147	4	151	38	9	20
3	146	22	124	42	4	17
4	122	27	145	56	10	20

5	104	37	134	83	8	23
6	77	37	110	33	11	23
7	82	24	156	38	8	20
8	104	37	100	40	4	13
9	88	21	87	45	35	18
10	75	17	111	41	6	2
11	44	28	71	29	3	17
12	56	4	120	38	8	12
13	44	16	100	31	7	7
14	44	12	104	28	5	24
15	57	15	97	35	15	10
16	32	17	84	33	14	5
17	36	15	116	40	20	16
18	44	20	176	44	2	28
19	51	23	157	37	11	13
20	48	18	162	43	17	17

17. Медицинские информационные системы: базовые сведения

Цель работы: изучить классификацию и принципы представления данных и результатов исследований больных, в медицинских информационных системах, основы медицинской деонтологии.

Краткие теоретические сведения.

Вопросы классификации медицинских информационных систем (далее МИС) рассматриваются в отечественной и зарубежной литературе с конца семидесятых годов двадцатого века. Именно тогда в крупных медицинских центрах большинства развитых стран начали внедряться информационные системы, направленные на автоматизацию широкого круга процессов, связанных с оказанием медицинской помощи пациенту и обработкой данных. С этого периода можно говорить о создании в развитых странах мира первых специализированных программных средств для здравоохранения и начале формирования группы компаний-разработчиков, впоследствии создавших рынок медицинских информационных систем.

Возникновение первоначального спроса на медицинские информационные системы порождает разнообразие предложения и потребность в его системном анализе. Принимая во внимание высокий уровень сложности и оригинальности медицинских информационных систем, наличие значительного количества организаций – разработчиков этого продукта в России, растущий уровень спроса со стороны медицинских организаций, не обладающих специальными IT - подразделениями, возникает потребность в простых и удобных инструментах для их системного анализа. Они призваны предоставить возможность руководителям разобраться в многообразии существующих предложений, сформулировать задачу, предварительно просчитать финансовые, кадровые и технические возможности, т.е. провести подготовительную работу для осознанного выбора оптимального решения.

Для сравнения параметров различных систем и дальнейшего анализа их свойств необходимым условием является построение классификации медицинских информационных систем.

В России основоположником системной классификации МИС выступил С.А. Гаспарян. В период 1978-2001 год он опубликовал три варианта классификации. В последней своей версии рассматриваются пять основных групп медицинских информационных систем, включающих в себя:

- технологические информационные системы (ТИМС);
- банки информации медицинских служб (БИМС);
- статистические информационные медицинские системы ;
- научно-исследовательские информационные медицинские системы;
- обучающие (образовательные) информационные медицинские системы.

Свой вариант классификации предложил в 2001 году Г.А. Хай, структурировавший системы по следующим типам: медико-технологические, справочные, базы данных, приборно-компьтерные системы или измерительно-

вычислительные комплексы, микропроцессорные системы, системы обработки и передачи изображений, сервисные, автоматизированные системы управления.

За рубежом в настоящее время функционирует несколько сервисов[18-21], позволяющих самостоятельно провести предварительное сравнение возможностей систем различных производителей и функционального назначения. В их основе лежат уникальные алгоритмы, во многом схожие между собой и позволяющие формулировать многопараметрические задачи с получением выборки, соответствующей заданным условиям. Указанные сервисы ориентированы на рынки определенной группы стран, с точки зрения предложения, и распределенную структуру системы здравоохранения. В этой связи для нужд российского здравоохранения они могут быть использованы лишь в обзорном плане, т.к. на них представлены нелокализованные для России продукты для организаций здравоохранения, находящихся в иной организационно-финансовой модели. Соответственно подходы, применяемые при построении зарубежных классификаторов медицинских информационных систем не могут использоваться в нашей стране в качестве прототипа и рассматриваются нами только с точки зрения полноты анализа существующих в мире подходов и принципов построения.

Наиболее значимой работой выполненной в нашей стране по теме настоящего исследования является разработка стандарта организации СТО МОСЗ 91500.16.0002-2004 «Информационные системы в здравоохранении. Общие требования»[6] в разработке которого принимали участие Лебедев Г.С., Емелин И.В., Столбов А.П. и другие ведущие специалисты в области медицинской информатики.

В стандарте вводится следующая классификация:

- Медико-технологические ИС (МТИС), предназначенные для информационного обеспечения процессов диагностики, лечения, реабилитации и профилактики пациентов в лечебно-профилактических учреждениях.
- Информационно-справочные системы (БИИС), содержащие банки медицинской информации для информационного обслуживания медицинских учреждений и служб управления здравоохранением.
- Статистические ИС (СМИС) органов управления здравоохранением.
- Научно-исследовательские ИС (НИИС), предназначенные для информационного обеспечения медицинских исследований в клинических научно-исследовательских институтах (НИИ).
- Обучающие ИС (ОМИС), предназначенные для информационного обеспечения процессов обучения в медицинских учебных заведениях.

Примечательно, что в трактовке разработчиков стандарта к медицинским информационным системам (МИС) относятся все информационные системы, применяемые в здравоохранении и связанные с обработкой медицинской

информации, в то время как в современных документах к МИС относят только системы, устанавливаемые в медицинских учреждениях для учетно-статистической обработки данных.

Бурное развитие информационных технологий в России и в мире внесло значительные изменения в представления об общих принципах и функциональности глобальных отраслевых и локальных информационных систем. За исторически короткий промежуток с момента выхода документа, описанного в предыдущем абзаце появились новые технологии и подходы к сбору, обработке, передаче, хранению и анализу крупных объемов слабоструктурированных данных, которые, в значительной мере изменили саму парадигму построения указанных систем. Изменения перетерпели представления не только технологические аспекты, связанные с выходом на рынок устройств нового поколения, но и подходы к работе с первичными данными.

Остановимся на наиболее значительных, с нашей точки зрения, изменениях в принципах построения крупных распределенных систем и взаимодействующих с ними информационных системах в российском здравоохранении.

Во-первых, современное развитие технологий информационного обеспечения позволило перейти к персонифицированному учету сведений о конкретном физическом лице, являющимся источником первичных сигналов, вызывающих реакцию других элементов системы. Современная логика построения информационной среды призвана обеспечить прямое взаимодействие с конкретным физическим лицом, посредством различных средств и инструментов относящихся к системе здравоохранения и обеспечивающей ее работу информационной среде. Считается, что человек, находящийся в любой точке страны, может быть в реальном времени идентифицирован и обеспечен информацией, необходимой для оказания ему медицинской помощи. Конечно, выполнение указанной задачи требует соблюдения целого ряда условий, как самим человеком, так и системой здравоохранения (наличие универсального идентификатора, например, номера страхового полиса или СНИЛС, возможность коммуникаций, доступность учреждений здравоохранения, их техническая оснащенность и т.д.). Более того, целый ряд задач, необходимых для этого находится еще в начальной фазе – постановки или уточнения. Однако, такой подход рассматривается, как реализуемый в обозримом будущем и закладывается в логику проектирования и построения современных систем, входящих в информационную инфраструктуру здравоохранения страны. Напомним, что предыдущие подходы были основаны на информационном обеспечении организаций (их групп, ведомств и т.д.) их взаимодействии друг с другом и были выстроены с точки зрения решения задач различных элементов и организации их взаимодействия. Эти изменения возможно реализовать путем создания технического и

информационного ядра системы, средств коммуникации и нормативного обеспечения параметров функционирования информационной среды.

Важнейшую роль в развитии информационных технологий в здравоохранении играют Федеральные законы «Об основах охраны здоровья граждан в Российской Федерации» [4], и Федеральный закон «Об обязательном медицинском страховании» [5], которые впервые определили следующие основные правовые нормы:

- Порядок организации системы документооборота в сфере охраны здоровья в электронном виде – впервые появилась законодательная возможность ведения медицинской документации в электронном виде, при этом в [5] вводится норма подписания медицинских документов электронной подписью;
- Вводятся соответствующие номенклатуры в сфере охраны здоровья;
- Вводится обязанность ведения медицинскими организациями информационных ресурсов в сети Интернет;
- Вводится право медицинских организаций создавать локальные информационные системы;
- Вводится понятие персонифицированного учета при осуществлении медицинской деятельности и то, что в информационном обмене участвуют медицинские организации государственной и частной системы здравоохранения;
- Вводится состав данных персонифицированного учета о специалисте и о пациенте.

Подходы к построению информационной среды в здравоохранении изложены в Концепции создания Единой Государственной информационной системы в сфере здравоохранения (ЕГИСЗ), утвержденной приказом Министерства здравоохранения и социального развития Российской Федерации от 28 апреля 2011 № 364 [7]. Именно ЕГИСЗ призвана взять на себя системообразующие функции информационной среды в данной области.

Во-вторых, внедрение технологий удаленной обработки данных («облачных технологий») коренным образом меняет представления о принципах их сбора, передачи и обработки.

И, наконец, в-третьих, важнейшим условием функционирования системы является нормативно и технически реализованное решение «Универсальная электронная медицинская карта пациента», которая является основным источником информации о физическом лице для учреждений здравоохранения всех типов и связанных с ними организаций (министерства здравоохранения, страховых компаний, пенсионного и других фондов и т.д.).

Естественно, при формировании информационной среды процесс изменения технологий и подходов к решению отдельных задач будет еще многократно изменяться. Тем не менее, реализация трех вышеперечисленных

принципов позволяет выстроить классификацию медицинских информационных систем, их подсистем и приложений, которая, на наш взгляд, будет сохранять актуальность ближайшие 5-10 лет.

Прежде чем привести саму классификацию, хотелось бы остановиться на эволюции взглядов на построение и систематизацию, происшедшие за последние 5-7 лет.

В исторической ретроспективе в этом вопросе наблюдается общая тенденция смещения акцентов от технических характеристик программных средств к их функциональности и другим потребительским свойствам, таким, как надежность, простота эксплуатации, безопасность обрабатываемых данных, наличию дополнительных интеллектуальных функций помощи врачу (проактивных функций), интероперабельности, наличию специализированных модулей и др.

Это связанно с развитием самих программных продуктов, аппаратных возможностей, общих представлений о роли информационных технологий и их месте в профессиональной среде. Несомненно, ключевыми факторами развития в данном процессе выступают повышение компьютерной грамотности медицинских работников, рост доступности для них программных и аппаратных средств, унификация и стандартизация самих первичных данных, методов их передачи и обработки, позволяющих обеспечить их однократный ввод и многократное использование, возможность их размещения и обработки в удаленных центрах обработки данных (облачная архитектура), доступность данных для врачей и пациентов.

При построении описанных выше систем классификаций авторы решали задачу органичного построения иерархии систем с учетом их технических, функциональных характеристик и связанных с ними групп решаемых задач. При этом технические характеристики определялись уровнем развития информационных систем, а функциональные – в первую очередь были связаны с требованиями и особенностями организационно-технологического построения отрасли и медицинских учреждений различных типов. При беглом анализе становится очевидна главная тенденция их трансформации – в классификациях очевидно наблюдается смещение акцентов с организационно-технических свойств систем к их функциональным свойствам, ориентированным на конечного потребителя. Это отражает общие процессы в эволюции информационных систем в отрасли – за 30-40 лет они прошли путь от крупных централизованных программ, решающих узкие задачи, к распределенной информационной системе, основанной на доступности различных персональных устройств, современных средств обработки, передачи данных, формирования новой культуры информационного общества.

В процессе совершенствования информационных систем, задачи конечного пользователя становятся все более определяющими, что смещает приоритеты построения современных классификаторов к двум основным полюсам - пациенту, за которым закрепляются относящиеся к нему

персональные данные и требованиям отраслевых государственных служб, которые на сегодняшний день являются основным заказчиком и регулятором рынка МИС.

Впоследствии, с развитием рынка, повышением уровня доступности МИС для лечебных учреждений и врачей, и формированием распределенной отраслевой информационной инфраструктуры, мы ожидаем смещения второго полюса функциональной значимости в сторону задач врача в широком смысле этого слова.

Наиболее конкурентными в будущем, по нашему мнению окажутся системы позволяющие обеспечить дополнительные функции для врачей различных специальностей, участвующих в оказании медицинской услуги конкретному пациенту.

Все вышеприведенное относится к акцентам, которые оказывают наибольшее влияние при построении той или иной классификации, однако не исключает использования более широких современных требований к техническому и инфраструктурному уровню программных средств, обусловленных их интеграцией в глобальные полиотраслевые универсальные системы.

В данной работе, посвященной вопросам информационного обеспечения отдельной отрасли в определенной стране мы не имеем возможности более подробно остановиться на описании данных свойств информационных систем, отсылая читателя к источникам, в которых указанные вопросы освещены гораздо более подробно.

Одной из особенностей предложенной классификации является применение подхода гармонизированного построения структуры медицинских информационных систем, их компонентов и приложений по классам функций, позволяющее без нарушения структуры вносить в нее необходимые изменения. В целом построение классификации использует иерархический принцип, основанный на роли тех или иных программных продуктов, с точки зрения их функций в национальной информационной среде в сфере здравоохранения. Раздел же, посвященный МИС максимально адаптирован к сложившемуся функционалу информационного обеспечения процессов в медицинских учреждениях.

Классификация информационных систем в здравоохранении, исходя из их функционального назначения приведена в табл.1. Функции сгруппированы по сегментам и группам.

Таблица 1.Классификация функций информационных систем в здравоохранении.

Код	**Функциональное назначение информационной системы (компонента)**
1	Сегмент централизованных общесистемных компонентов
11	Подсистема интеграции прикладных систем
12	Подсистема ведения каталога пользователей Системы

13	Подсистема ведения реестра нормативно-справочной информации, словарей медицинских терминологий, регистр электронных документов
14	Подсистема управления удостоверяющим центром и инфраструктурой открытых ключей
15	Подсистема управления эксплуатацией
16	Подсистема ведения электронной почты и прочих информационно-технических сервисов общего пользования
2	Сегмент прикладных компонентов
21	Федеральные системы
211	Федеральные транзакционные системы
2111	Системы ведения расписания приемов специалистов, проведения консультаций, в том числе телемедицинских, и загрузки мощностей медицинской организации, а также электронной записи на прием к врачу
2112	Системы, обеспечивающие проведение телемедицинских консультаций и консилиумов, в том числе с использованием мобильных устройств
2113	Системы, обеспечивающие направление на проведение диагностических исследований, проведение медицинского обследования (консультации, экспертизы) и получение медицинской помощи в иные медицинские организации
2114	Системы, обеспечивающие бухгалтерский и управленческий учет финансово-хозяйственной деятельности медицинских организаций, в том числе автоматизирующие функции взаимодействия со страховыми медицинскими организациями в части формирования и оплаты счетов за оказанную медицинскую помощь
2115	Системы, обеспечивающие кадровый учет в медицинских организациях
212	Федеральные управленческие системы
2121	Регистр паспортов медицинских организаций
2122	Регистр медицинской техники и изделий медицинского назначения
2123	Регистр медицинского и фармацевтического персонала
2124	Система мониторинга реализации программ в здравоохранении
2125	Система ведения интегрированной электронной медицинской карты, а также создаваемых на ее основе специализированных регистров по отдельным нозологиям и категориям граждан, в том числе обеспечивающая персонифицированный учет медицинской помощи и лекарственного обеспечения
2126	Аналитические системы, а также системы автоматизированного контроля и поддержки принятия управленческих решений на основе анализа первичных данных
2127	ИС сбора и анализа медицинской статистики
213	Федеральные справочные системы
2131	Федеральная электронная медицинская библиотека, содержащая, в том числе, электронные медицинские публикации, электронные справочники лекарственных средств и заболеваний, стандарты оказания медицинской помощи, протоколы лечения, иную нормативно-справочную информацию
2132	Библиотека экспертных медицинских систем, обеспечивающих автоматизацию процесса поддержки принятия врачебных решений на базе формализованных баз знаний и прецедентной информации
2133	Электронные образовательные курсы, программы дистанционного обучения и повышения квалификации в области медицины и фармацевтики
2134	Системы группового профессионального общения для медицинского и фармацевтического персонала и студентов медицинских и фармацевтических средних специальных и высших учебных заведений
2135	Средства обеспечения доступа к первичным данным, содержащимся в Системе, необходимым для проведения научных исследований в сфере здравоохранения
2136	Средства информирования граждан о деятельности системы здравоохранения
22	Региональные системы

221	Медицинские информационные системы
2211	ИС управления ресурсами медицинского учреждения
221101	ИС бухгалтерского учета
221102	ИС кадрового учета
221103	ИС экономического учета и отчетности
221104	ИС учета медицинского оборудования
221105	ИС управления материально-техническим обеспечением и основными средствами
221106	ИС управления, учета и контроля движения лекарственных средств, изделий медицинского назначения, расходных материалов
221107	ИС управление питанием пациентов в стационаре
221108	ИС учета движения автотранспорта
221109	ИС расчета стоимости медицинских услуг
221110	ИС управление расписаниями, записи пациентов на обслуживание, управления занятостью ресурсов и распределение потоков пациентов в лечебно-профилактическом учреждении (электронная регистратура)
221111	ИС ведения договоров с оплачивающими организациями, программ прикрепления
221112	ИС документооборота
221113	Сайт медицинского учреждения в сети Интернет
2212	ИС экономико-статистического учета
221201	ИС учета прикрепленного контингента, регистрации и учета обслуживаемых граждан
221202	ИС поддержания деятельности приемного отделения стационара, ведения плановой очереди на госпитализацию, управления коечным фондом
221203	ИС регистрации заболеваний в поликлинике
221204	ИС учета выполненных объемов медицинской помощи
221205	ИС учета выбывших пациентов стационара
221206	ИС учета и планирования иммунопрофилактики
221207	ИС организации взаиморасчетов
221208	ИС анализа деятельности и формирование статистической отчетности
221209	ИС учета оказания скорой и неотложной медицинской помощи
221210	ИС учета временной нетрудоспособностью граждан
221211	ИС учета дополнительного лекарственного обеспечения
221212	ИС учета дополнительной диспансеризации граждан
221213	ИС автоматизированной поддержки центров здоровья
2213	Электронная медицинская карта (электронная история болезни, электронный паспорт здоровья)
221301	ИС ведения протоколов осмотров врачей, ведущих амбулаторный прием
221302	ИС ведение дневника лечащего врача
221303	ИС поддержки патологоанатомической деятельности
221304	ИС ведения диспансерной группы пациентов по хроническим заболеваниям
221305	ИС поддержки клинико-экспертной работы
221306	ИС учета назначения и применения лекарственных средств
221307	ИС Деятельность стоматологии
221308	ИС автоматизированного учета протоколов лечебно-диагностических мероприятий
221309	ИС планирования и учета оперативных вмешательств
221310	ИС учета и мониторинга анестезиологических пособий
221311	ИС ведения карты интенсивной терапии
221312	ИС ведения перфузионной карты пациента
221313	Медико-технологические ИС
221314	Мониторные системы
221315	Приборно-компъютерные комплексы
221316	Лабораторная информационная система (ЛИС)
221317	Радиологические информационные системы (РИС)

221318	Системы передачи и обработки изображений (PACS-системы)
221319	Подсистема комплексной диагностики (в том числе функциональная, инструментальная, вычислительная)
221320	ИС поддержки деятельности в области трансфузиологии
221321	Экспертные системы
221322	ИС ведения научно-исследовательских работ
222	Системы удаленного мониторинга состояния здоровья отдельных категорий пациентов
223	Системы архивного хранения и предоставления доступа к медицинским изображениям (региональные PACS-системы)
224	Обеспечивающие системы, функциональность которых не реализована в рамках прикладных компонентов федерального уровня
225	Региональные управляющие информационные системы в сфере здравоохранения
22501	Информационная система управления кадрами органа исполнительной власти в сфере здравоохранения (территориальный орган управления здравоохранением – ТОУЗ)
22502	Информационная система управления финансами и материально-техническим обеспечением ТОУЗ
22503	Информационная система управления организацией закупок на поставку товаров, выполнение работ, оказание услуг ТОУЗ
22504	Информационная система электронного документооборота ТОУЗ
22505	Региональные информационные порталы по вопросам здравоохранения
226	Компоненты для обеспечения информационного, лингвистического и процессного взаимодействия
22601	Технологическое хранилище данных информационных систем учреждений здравоохранения и медицинского оборудования, включая средства их проверки на достоверность, очистки и передачи в федеральный центр обработки данных
22602	Прочие информационно-технические сервисы
22603	Средства взаимодействия с общесистемными компонентами федерального уровня Системы
22604	Подсистема доступа к каталогу пользователей Системы, создаваемого на федеральном уровне Системы
22605	Подсистема доступа к нормативно-справочной информации и словарям медицинских терминологий, создаваемого на федеральном уровне Системы
22606	Сервисы взаимодействия с инфраструктурой открытых ключей электронной цифровой подписи
22607	Центр поддержки пользователей по вопросам функционирования централизованных информационных систем на региональном уровне Системы, в том числе информационных систем, размещаемых централизованно и используемых удаленно
22608	Система обеспечения информационной безопасности
22609	Сервисы взаимодействия с системой межведомственного электронного взаимодействия, инфраструктурой выдачи и обслуживания универсальных электронных карт, единым порталом государственных и муниципальных услуг, региональным порталом государственных и муниципальных услуг и иными системами, создаваемыми в рамках инфраструктуры электронного правительства
22610	Сервисы взаимодействия с внешними информационными системами, включая информационные системы государственных внебюджетных фондов, страховых медицинских организаций и иными информационными системами

Приведенная классификация позволяет довольно точно описать функциональный состав информационной системы и ее роль в единой государственной информационной системе в сфере здравоохранения.

Вместе с тем, распределенный принцип финансирования, примененный для работ по информатизации здравоохранения, подразумевает довольно высокий уровень самостоятельности при выборе архитектуры построения информационной системы уровня ЛПУ, муниципального и регионального образования. При этом построенная система должна жестко обеспечивать интеграцию в системы старшего уровня, что связано с необходимостью соблюдения условий сбора, обработки, хранения и передачи данных, выполнения нормативных регламентов.

В этой связи нам представляется актуальной формирование таблицы минимального необходимого набора параметров информационных систем, которая бы позволила сравнить возможности программных продуктов с близким или смежным функционалом.

Такой набор данных без сомнения был бы полезен руководителям организаций в здравоохранении при создании или модернизации их информационных систем. Приведение параметров информационных систем в унифицированной форме также могло бы стать хорошим драйвером образования рынка информационных продуктов в рассматриваемом отраслевом сегменте, создав предпосылки для цивилизованной конкуренции разработчиков.

В таблице 2 приведены основные параметры информационной системы для здравоохранения и расшифровка указанных параметров.

Таблица 2. Основные параметры информационной системы

№ п/п	**Наименование параметра**	**Описание параметра**
1.	Наименование МИС	Указывается торговая марка медицинской системы, параметры которой вносятся в таблицу (специализированный программный продукт, независимый функциональный модуль или полная комплексная система – по выбору разработчика. Важно строгое соответствие данных, приведенных в таблице программному продукту, указанному в графе «Наименование МИС».)
2.	Тип системы	Указывается тип системы. Необходимо вписать один из следующих вариантов: • комплексная интегрированная медицинская информационная система (многофункциональная); • специализированная система (указать специализацию); • функциональная (указать функционал) подсистема; • независимый функциональный модуль (указать функционал); • специализированное приложение (указать специализацию); • компоненты для обеспечения информационного, лингвистического и процессного взаимодействия; • аппаратно-программное решение
3.	Классы функций системы	Указываются коды функций, реализуемых системой, в соответствии с табл.1. при этом заносятся все числовые коды имеющихся в системе функций и наименование старшего класса функций из отмеченных в данной таблице
4.	Масштаб	Указывается максимальный масштаб реализованного успешного

	реализованного внедрения (федеральный, региональный, муниципальный, ЛПУ)	внедрения системы (не менее одного года) в сфере здравоохранения : федеральный (более 3 регионов), региональный, муниципальный, ЛПУ.
5.	Среднее количество внедрений за лтчетный период, в год	Указывается среднее количество медицинских организаций, в которых внедрена система (в полном объеме или частично) в среднем за год в диапазоне указанного временного интервала
6.	Среднее количество инсталляций за отчетный период, в год	Указывается среднее количество автоматизированных рабочих мест в медицинских организациях, в которых внедрена система (предоставлена возможность пользования ей в полном объеме или частично) в среднем за год в диапазоне указанного временного интервала
7.	Краткое описание архитектуры системы	В данной графе дается краткое описание архитектуры сложных многоуровневых систем с точки зрения обеспечиваемых ими для потребителя групп функций, поддержки модульности / компонентности, типа модели клиент-сервер (если используется), перечень основных функциональных модулей, входящих в состав интегрированной системы (не более 500 символов)
8.	Объем обрабатываемых данных	В данной графе указывается класс системы с точки зрения возможностей объема обрабатываемых персональных данных (количество субъектов персональных данных, персональные данные которых обрабатываются в информационной системе) – Хнпд: 1. В информационной системе одновременно обрабатываются персональные данные более чем 100 000 субъектов персональных данных или персональные данные субъектов персональных данных в пределах субъекта Российской Федерации или Российской Федерации в целом; 2. В информационной системе одновременно обрабатываются персональные данные от 1000 до 100 000 субъектов персональных данных или персональные данные субъектов персональных данных, работающих в отрасли экономики Российской Федерации, в органе государственной власти, проживающих в пределах муниципального образования; 3. В информационной системе одновременно обрабатываются данные менее чем 1000 субъектов персональных данных или персональные данные субъектов персональных данных в пределах конкретной организации
9.	Масштабируемость (увеличение производительности системы в 10 раз, кол-во месяцев.)	Указывается количество времени, необходимое разработчику для адаптации данного программного продукта под десятикратное увеличение количества пользователей и объема обрабатываемой информации
10.	Платформа, лицензионное ПО	Серверы, СУБД, ОС, языки программирования, на которых написано данное ПО, указать какое лицензионное ПО используется
11.	Поддержка сервиса «Электронная медицинская карта»	Поддержка сервиса «Ведение электронных медицинских карт пациентов» на уровне требований не менее установленных Минздравсоцразвития России и национальными стандартами.
12.	Возможность генерации отчетов государственной статистики и	Формирование отчетов в соответствии с требованиями федерального статистического наблюдения и системы ОМС, возможность экспорта их в электронном виде в формате ODF (Open Document Format - открытый формат документов)

	документов в ТФОМС	
13.	Поддержка стандартов лечения и связанных с ними справочников и классификаторов	Поддержка государственных стандартов лечения с помощью встроенных обновляемых справочников и классификаторов, общего механизма работы с планами лечения и справочника шаблонов
14.	Поддержка просмотра, передачи, хранения и обработки изображений	Наличие встроенного PACS решения, обеспечивающего просмотр изображений по протоколу DICOM и подключение DICOM-оборудования, а также механизмов для организации структурированного хранилища изображений, поиска, просмотра и редактирования изображений
15.	Возможность генерации и печати бумажных форм	Возможность генерации и вывода на печать медицинской карты пациента, справок, выписок, отчетов и других установленных форм в соответствии с действующими нормами и требованиями Минздравсоцразвития России, ФОМС, ПФ РФ, ФСС, ФМБА России, региональными и ведомственными нормативами, определенными для реализуемых программой функций
16.	Возможность проведения телемедицинских консультаций	Наличие необходимого количества автоматизированных рабочих мест для доступа к системе, обеспечивающей проведение телемедицинских консультаций и консилиумов, создаваемой на федеральном уровне, с учетом технических требований, разрабатываемых Минздравсоцразвития России
17.	Наличие общехозяйственных, кадровых, финансовых и управленческих модулей	Указать наличие соответствующих модулей и аналитических функций
18.	Возможности пользовательской настройки интерфейса АРМ	Наличие встроенного редактора форм, позволяющего создавать новые формы и поля ввода, менять внешний вид рабочего стола и взаиморасположение основных объектов интерфейса сохраняя обязательные поля для ввода
19.	Поддержка принятия врачебных решений	Наличие встроенного модуля, позволяющего на базе имеющихся справочников и базы знаний, накопленной в системе, выдавать рекомендации и ограничения при назначении лекарственных препаратов, процедур и т.д. В случае наличия более сложных экспертных систем, предлагающих на основе накопленных о пациенте данных, вероятностные прогнозы, просьба указать отдельно
20.	Наличие сервиса создания и поддержки web- сайта	Встроенный сайт ЛПУ с интегрированными функциями МИС – расписание, запись на прием к врачу, перечень специалистов, справочная информация и т.д.
21.	Разработчик (Дистрибьютор)	
22.	Международное наименование МИС (в случае локализованной версии)	
23.	Почтовый адрес и сайт разработчика	
24.	Контакты	
25.	Наличие других программных продуктов для учреждений и органов управления	

	здравоохранением	
26.	Сервисное обслуживание в регионах (количество регионов и количество сертифицированных специалистов поддержки, включая партнеров.)	Указать количество регионов и сервисных инженеров и консультантов, включая сотрудников компаний-партнеров, обеспечивающих поддержку функционирования программного продукта в данном регионе. В случае, если сервисная сеть продукта охватывает более 5 регионов, просьба привести указанные данные в отдельной таблице, сохранив в таблице 2 цифры общих данных
27.	Стоимость приобретения, проектирования, разработки, эксплуатации, аренды, адаптации и внедрения системы в ЛПУ на 10/50/300 рабочих мест, в первый год эксплуатации, руб.	
28.	Совокупная стоимость владения системой в ЛПУ на 10/50/300 рабочих мест во второй и последующие годы эксплуатации (диапазон), руб. в год	Указать минимальное и максимальное из расчета на год, включая стоимость лицензий на прикладное и общесистемное ПО, аренду, обновление и сервисное обслуживание, в среднем, со второго по пятый год эксплуатации

Таким образом, внедрение единой классификации информационных систем позволит эффективно рассчитать стоимость внедрения и сопровождения систем, грамотно сформулировать требования к внедряемым системам и правильно определить необходимые ресурсы.

Пример схемы классификации С.А. Гаспаряна.

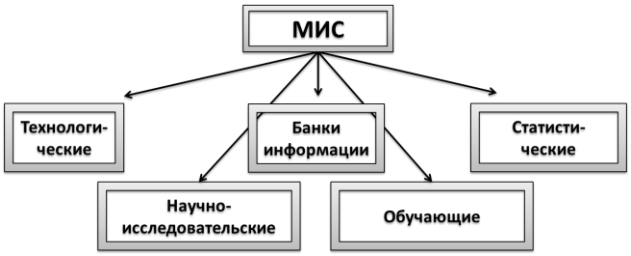


Схема иерархической классификации МИС различных уровней.

Порядок выполнения работы.

1. Изучите теоретический материал.
2. Составьте пояснение в форме презентаций приведенных в кратких теоретических сведениях схем классификации.
3. Подготовьте доклад для обсуждения на практическом занятии аспектов медицинской деентологии.
4. Проанализируйте функциональное назначение и содержимое информационных источников
в интернете (указанных в литературе).
5. Оформите отчет, включающий результаты выполнения работы и ответы не менее чем на три контрольных вопроса.

Контрольные вопросы.

1. Что представляет собой информационная медицинская система?
2. Охарактеризуйте отечественные классификации МИС.
3. Дайте характеристику зарубежной классификации МИС.4
4. Опишите классификацию МИС, основанную на иерархическом принципе построения системы здравоохранения и оказания пациенту медицинской помощи.
5. Что представляет собой задание на разработку системы поддержки принятия решений в медицине? Какие разделы оно включает?
7. На каких принципах должно базироваться создание МИС?
8. Какие требования предъявляются к МИС?
9. Назовите стандарты, нашедшие широкое применение при разработке и взаимодействии МИС.
10. Что собой представляет и для чего используют стандарт *HL11* ?
11. Для чего нужна международная систематизированная номенклатура медицинских терминов *SNOMED CT?*
12. Для чего используется стандарт *DICOM1?*
13. Что собой представляет организационное обеспечение функционирования МИС?
14. Как регламентируется работа медицинских учреждений в условиях функционирования МИС?

Информационные источники

1.. http://www.ctsguides.com
2. http://www.klasresearch.com
3. http://www.softwareadvice.com/medical/
4. http://www.healthtechnologyreview.com/medical-emr-software-reviews.php

18. Экспертные системы дифференциальной диагностики

Цель работы: изучение принципов построения и эксплуатации современных экспертных медицинских систем дифференциальной диагностики.

Краткие теоретические сведения.

Диагностика – это раздел медицинской науки изучающей методы распознавания болезней в процессе исследования, наблюдения больного и мышления врача с целью установления диагноза.

Дифференциальная диагностика – это сравнение клинической картины заболевания у данного конкретного больного со всеми заболеваниями, имеющими сходную клиническую картину, для исключения всех болезней, кроме одной.

Это сравнение должно проводиться не хаотично, не стихийно, а по строгим правилам и в определенном порядке. Иными словами, мыслительный процесс при дифференциальной диагностике должен подчиняться такой же строгой схеме, как, например, проведение лабораторного исследования или анализ электрокардиограммы. Проведение дифференциальной диагностики состоит из следующих пяти этапов:

- группировка симптомов для выделения синдромов;
- выделение ведущего синдрома;
- генерация диагностической гипотезы;
- построение дифференциально-диагностического ряда;
- исключение синдромносходных заболеваний и формулировка окончательного диагноза .

Рассмотрим последовательность действий на этапах проведения дифференциальной диагностики.

Первый этап – группировка симптомов для выделения синдромов, был достаточно описан выше, но хотелось бы подчеркнуть, что при выделении синдрома следует оценить его особенности. Например, недостаточно констатировать наличие у больного изолированного мочевого синдрома, следует оценить его симптоматологию для идентификации его как клубочкового либо канальцевого и уточнить степень выраженности протеинурии, гематурии, лейкоцитурии, оценить ее характер и т.д. Подробная патогенетическая характеристика синдромов абсолютно необходима для плодотворной деятельности на следующих этапах.

На втором этапе ведущий синдром выделяется по тем же критериям, что и ведущий симптом. Главное здесь – плодотворность генерации диагностической гипотезы. Например, у больного с синдромами бивентрикулярной сердечной недостаточности, тахисистолической мерцательной аритмией и кардиомегалией проведение дифференциального диагноза достаточно сложно, т.к. комбинация этих синдромов закономерно прослеживается на поздних этапах почти всех сердечно-сосудистых заболеваний.

На третьем этапе формируется диагностическая гипотеза. Диагностическая гипотеза – это рабочий диагноз, который создается методом аналогии на основе синдромного анализа.

Характерными чертами диагностической гипотезы являются:

- неполнота аргументации;
- уникальность диагностической гипотезы.

Неполнота аргументации означает, что рабочий диагноз формулируется у постели больного на базе данных анамнеза и физикального обследования. Следует иметь в виду, что как анамнестическое, так и физикальное обследование, несмотря на стереотипизм их реализации, не предусматривают некоторых вопросов, приемов, методов, которые имеют существенное значение для диагностики относительно редких заболеваний. Например, проведение провокационных проб для диагностики кровоточивости петехиально-пятнистого типа, оценка ширины пространства между относительной и абсолютной тупостью сердца для верификации гидроперикарда и т.д.

Под уникальностью диагностической гипотезы в данном случае, имеется в виду то, что диагностическая гипотеза генерируется в процессе работы с каждым больным. Она может звучать одинаково, при работе с довольно большим количеством больных, например «гипертоническая болезнь», но аргументация в пользу этого предположения каждый раз создается заново.

Считается, что в процессе генерации диагностической гипотезы возможен ряд ошибок логического порядка:

- ошибка поспешного обобщения;
- ошибка в умозаключении по аналогии;
- ошибка ложной последовательности.

Ошибкой поспешного обобщения является учет диагностической гипотезой не всех, а лишь некоторых клинических данных. Например, ослабление дыхания в нижней доле справа и тупой перкуторный звук в этой области может привести к рабочему диагнозу пневмонии, если не учесть ослабление бронхофонии и смещение средостения в здоровую сторону. Тогда это заболевание не пневмония, а экссудативный плеврит или гидроторакс.

Ошибка в умозаключении по аналогии происходит, когда обобщаются процессы, имеющие разный патогенез. Например, при раке пищевода с дисфагией, возникшем у больного с вибрационной болезнью, проявляющейся в основном синдромом Рейно, возможно заключение о системной склеродермии.

Ошибка ложной последовательности возможна, если умозаключение движется по пути «после этого, значит вследствие этого», тогда как далеко не всегда последовательное возникновение симптомов означает наличие между ними причинно – следственной связи.

Так, весьма частое сочетание хронического пиелонефрита и артериальной гипертензии не всегда означает, что гипертония является нефрогенной. Чаще всего в этой ситуации артериальная гипертония оказывается гипертонической болезнью.

На четвертом этапе строится дифференциально - диагностический ряд, что требует от врача широкой эрудиции с одной стороны и навыков системного мышления с другой. Следует обратить особое внимание, что дифференциально - диагностический ряд - не простое перечисление заболеваний, имеющих один отличительный признак (выделенный в качестве ведущего синдрома), на большие или меньшие группы. Разделение это осуществляется на патогенетической основе. Так, заболевания, обусловливающие возникновение лихорадки, представлены инфекциями, опухолями и системными заболеваниями соединительной ткани. Выделяется также искусственная лихорадка.

Такого рода внутри синдромные классификации чрезвычайно полезны, т.к. позволяют в дальнейшем проводить дифференциальную диагностику, исключая заболевания не по одному, а целыми группами. Работа по составлению дифференциально-диагностического ряда сопровождается его написанием. Это служит гарантией того, что ни одно из заболеваний не будет забыто в процессе дифференциальной диагностики.

На пятом этапе осуществляется процесс исключения синдромно - сходных заболеваний в два этапа. На первом этапе производятся межгрупповые сопоставления и исключение всех групп заболеваний, кроме одной. На второй фазе происходит сопоставление заболеваний внутри оставшейся группы и исключение всех болезней, кроме одной.

Это достаточно сложная работа. На практике она осуществляется в соответствии с основными принципами дифференциальной диагностики.

Определяются пять основных принципов дифференциальной диагностики.

Первый принцип – принцип существенного различия**.**

Принцип существенного различия означает, что наблюдаемый случай болезни не принадлежит к сравниваемому с ним виду заболеваний, т.к. в клинической картине у данного больного отсутствует синдром, обязательно имеющий место при сравниваемой болезни, или группе болезней. Например, артериальная гипертония у данного больного не может быть вазоренальной, т.к. отсутствует синдром стеноза почечной артерии (нет сосудистого шума, почки симметричны как по размерам, так и функционально, проба с капотеном отрицательная).

Принцип существенного различия имеет и другую трактовку: заболевание не относится к сравниваемому с ним виду, т.к. у больного имеется синдром (симптом), никогда не встречающийся при сравниваемом заболевании. М.П. Кончаловский учил, что «необходимо знать не только положительные, но и отрицательные признаки того или другого патологического процесса». Например, артериальная гипертония не может быть обусловлена болезнью Иценко-Кушинга, т.к. у больного имеется синдром стеноза почечной артерии. Принцип существенного

различия – основной принцип дифференциальной диагностики, но и он имеет относительное значение, особенно в латентной стадии заболеваний.

Второй принцип – принцип исключения через противоположность. Этот принцип можно сформулировать так: у больного имеется симптом, прямо противоположный тому, который закономерно выявляется при сравниваемой болезни.

Третий принцип – *Принцип несовпадения признаков* – важнейший и наиболее часто используемый. В данном случае речь идет о существенном сходстве симптома и в то же время – о его отличительных характеристиках при различных заболеваниях. Как раз эти отличия, иногда до степени противоположностей, и позволяют проводить дифференциацию. Самые частые примеры – это дифференциальная диагностика сердечной и бронхиальной астмы, отеков сердечного и почечного происхождения и т.д. Путь дифференциации весьма упрощается, если рассматривать симптом не как отдельно взятый признак болезни, а как составляющую синдрома. Например, удушье при бронхиальной астме – составляющая бронхо - обструктивного синдрома, а при сердечной – синдрома сердечной недостаточности.

Четвертый принцип - *Принцип индуктивного дифференциального диагноза* предусматривает постоянную работу в библиотеке. По прочтении соответствующей предполагаемому заболеванию литературы неизбежно возникает потребность повторить обследование больного с фиксацией внимания на специфических проявлениях болезни. Например, при подозрении на гипертрофическую кардиомиопатию повторить пальпацию сердца для верификации двойного верхушечного толчка, аускультацию сердца для выслушивания дополнительных тонов и т.д.

Принцип индуктивного дифференциального диагноза служит важнейшим инструментом при назначении дополнительных исследований. В сущности, программа дополнительных исследований реализует еще один принцип дифференциального диагноза.

Пятый принцип – *принцип проверки диагноза.* В зависимости от случая, проверка диагноза может быть либо очень короткой (в самых простых случаях она не нужна), либо продолжительной, трудной для врача и больного. Иногда приходится двигаться от простых методов к все более и более сложным, если простые и необременительные способы диагностики на момент их использования не дают определенных результатов. Поэтому, несмотря на разработку таких современных способов диагностики, как ультразвуковые, компьютерные томографические и т.п., следует иметь в виду, что для постановки окончательного диагноза может понадобиться более или менее длительное наблюдение. Идея диагноза путем наблюдения (diagnosis per observationae) отнюдь не пассивна. Задача врача на этом этапе – активно выявлять новые признаки и динамику предсуществовавших признаков заболевания[2].

Одним из вариантов постановки диагноза путем наблюдения является *диагноз по лечебному эффекту* (diagnosis ex juvantibus). Классический пример

такой диагностики – дифференциальная диагностика неспецифического воспалительного процесса в легких и туберкулеза путем назначения неспецифической антибактериальной терапии, а затем противотуберкулезного лечения.

Как самостоятельный способ диагностики описана постановка *диагноза по результатам операции* (diagnosis sub operatione). Это редкий способ диагностики. Описан случай наблюдения больного с ведущим симптомом болей в животе. При компьютерной томографии в брюшной полости было верифицировано опухолевидное образование, по поводу которого предпринималась лапароскопия. При ревизии брюшной полости опухоли не обнаружено. При биопсии и последующем морфологическом исследовании сосудов брыжейки поставлен диагноз нодозного полиартерита.

Постановка диагноза путем наблюдения возможна не всегда, а лишь в наиболее сложных и прогностических относительно «безобидных» случаях.

Решение о проверке рабочего диагноза, дифференциальную диагностику и формулировку клинического диагноза осуществляют в течение нескольких минут, как это делают терапевты поликлиник на амбулаторном приеме и нескольких суток в клиниках.

Постановка точного диагноза иногда сопряжена с неоправданно высоким риском некоторых диагностических тестов. Назначая биопсию печени, почек и т.д., нужно быть абсолютно уверенным в их пользе для дальнейшего лечения. Перед проведением опасного исследования необходимо оценить вероятность, последствия и обратимость возможных осложнений. Самое главное, что нужно сделать перед назначением «широкозахватного» обследования, это спросить себя, как повлияют его результаты на дальнейшее лечение. Во всяком случае, искусство обследовать больного во многом синонимично искусству диагностировать и зиждется на глубоких знаниях обширного спектра патологических процессов, собственном опыте и здравом смысле врача.

Применение методики дифференциальной диагностики по одному ведущему синдрому плодотворна, но иногда, особенно при сложных, многосиндромных клинических картинах моно-синдромный дифференциальный диагноз не приводит к успеху. Диагностическая гипотеза (рабочий диагноз) не оказывается ни подтвержденной, ни отвергнутой. В этой ситуации все приходится начинать сначала, выделяя другой синдром в качестве основного. При безуспешности этой попытки следует перейти к дифференциальной диагностике по третьему синдрому и т.д.

Вообще при полисиндромных процессах иногда целесообразно взглянуть на вещи более широко: все многообразие клиники может быть обусловлено не одним заболеванием, а их сочетанием. В-третьих, сочетание синдромов должно наводить на размышления о возможности злокачественного заболевания.

Назначение экспертных систем дифференциальной диагностики и их особенности.

Современные технические возможности позволяют выйти на качественно новый уровень представления течения заболевания, а именно визуально, на основе соответствующих математических моделей, пространственно смоделировать типовое развитие патологического процесса при конкретном заболевании. Уже сейчас, на современном этапе развития медицины, информационные нагрузки достигают пределов человеческих возможностей. Возникает дилемма: либо жертвовать полнотой анализа информации, либо шире использовать различные методы компьютерной поддержки принятия решений. Медицинские экспертные системы позволяют врачу не только проверить собственные диагностические предположения, но и обратиться к компьютеру за консультацией в трудных диагностических случаях.

Вводимая информация может быть не только текстовая, но и графическая, в виде кардиограмм, рентгенограмм и т.д. Система анализирует все параметры, учитывая историю болезни пациента, хранящуюся в базе данных, и предлагает врачу свой вариант диагноза и возможных действий. При этом она основывается на базе знаний квалифицированных специалистов, которая создавалась при разработке программы. Поэтому такую систему правильнее назвать «Справочник-консультант». Конечно, окончательное решение принимает врач.

Поскольку эта сфера применения очень ответственна, существующие экспертные системы используют достаточно сложные алгоритмы, уменьшающие вероятность неверных действий, поэтому здесь трудно рассчитывать на универсальность и многопрофильность.

Одна из самых известных в мире консультационных экспертных систем – *MYCIN*, предназначенная для медицинской диагностики инфекционных заболеваний крови, сопоставляет симптомы исследуемой болезни с симптомами болезней, накопленных в базе знаний. Врач отвечает на вопросы ЭС о симптомах болезни, а затем, получив достаточно фактов, ЭС помогает врачу поставить диагноз и дает рекомендации по лечению.

В качестве примера российской разработки можно привести комплекс *РОФЭС* (Регистратор Оценки Функционально-Эмоционального Состояния). Система состоит из комплекса датчиков и программного обеспечения, с помощью которых производится диагностика и анализ состояния организма. Она может осуществить подбор препаратов для конкретного человека, провести частотную и цветотерапию (с использованием обратной связи), определить гормональный профиль и риск опасных видов заболеваний. Таким образом, РОФЭС производит не только оценку, но и управление состоянием здоровья.

Экспертная система ДИН

С целью повышения качества диагностики и эффективности лечения критических состояний у детей в Московском НИИ педиатрии и детской хирургии создана экспертная система ДИН, ориентированная на решение задачи распознавания неотложного состояния у детей в терминах синдрома или нескольких синдромов (точнее, характеризующих их состояний, которые

отражают степень выраженности синдрома) при предъявлении признаков заболеваний, под которыми следует понимать анамнестические, клинические и лабораторные проявления .

База знаний ЭС содержит описания 34 синдромов, которые включают 84 состояния. Для системы это список диагностических предположений-гипотез. База экспертных знаний врача-реаниматолога содержит более 1000 диагностических критериев и заключений о динамике развития неотложного состояния.

С целью ускорения работы системы, ориентированной на угрожающие жизни состояния, все множество синдромов подразделено на 14 групп в соответствии с преобладающими нарушениями со стороны той или иной системы жизнеобеспечения, и каждая группа имеет свои дифференцирующие симптомы.

Знания о синдромах представлены таким образом, что охватывают:

-условие выдвижения гипотезы о возможности возникновения синдрома, которое является некоторой логической комбинацией симптомов;

-клиническую картину, то есть симптомы заболевания;

- необходимые и достаточные условия для подтверждения гипотезы;

-информацию о дополнительных синдромах, состоящих в некоторых отношениях с рассматриваемым: 1) причинно-следственные связи, предполагающие информацию о синдромах, которые могут быть причиной данного синдрома или, наоборот, являться его следствием; 2) временные связи, позволяющие как прогнозировать состояние ребенка, так и восстанавливать возможный анамнез болезни; 3) ассоциативные связи, дающие возможность учитывать, на фоне каких состояний может развиться данный синдром, фоном для каких синдромов он может служить и с какими синдромами может быть совместим, то есть какие синдромы могут встречаться одновременно;

- информацию о состояниях, взаимоисключающих друг друга;

- сведения о дифференцируемых синдромах (синдромах-конкурентах).

Все знания о синдромах подразделяются на декларативные - для описания самого синдрома (клиническая картина, дополнительные синдромы) - и процедурные, указывающие на то, как использовать знания в процессе диагностики.

Функциональная схема ДИН соответствуй схеме классической ЭС: интерфейс, блок представления знаний, блок метазнаний, блок механизма логического вывода, блок рабочей области, блок пополнения и модернизации знаний.

Информационно-диагностическая система по наследственным болезням у детей «Диаген».

Система "ДИАГЕН" - предназначена для консультативной помощи врачам при диагностике редко встречающихся и трудно распознаваемых заболеваний наследственной природы на долабораторном этапе обследования.

База знаний системы включает сведения о 1200 моногенных и хромосомных болезнях, а модуль визуального представления данных - более 1000 фотографий больных с наследственными болезнями.

Система включает три основных блока:

- диагностический, выдающий дифференциально-диагностический ряд;
- справочник, который предоставляет полную информацию о признаках и синдромах, хранящихся в системе (ориентирован в первую очередь на врача-педиатра широкого профиля);
- архив, который обеспечивает хранение и повторное использование данных о диагностированных больных.

Система может быть использована при решении вопроса о предварительном (долабораторном) диагнозе при подозрении на моногенную и хромосомную патологию как справочная система, а также в учебном процессе.

Система "ДИАГЕН" реализована для IBM-совместимых персональных компьютеров. Программное обеспечение написано на языке Borland C++ и функционирует под MS DOS.

Экспертная система Мутант

Система *Мутант* обеспечивает проведение первичного скрининга заболеваний с целью получения диагностической информации, необходимой для организации более углубленного обследования пациентов. Предлагаемый подход к дифференциальной диагностике основан на интервьюировании пациента по определенной схеме.

Вначале пациенту предлагается ответить на первичный (основной) опросник, включающий паспортные и некоторые антропометрические данные (масса тела, пульс, артериальное давление), личный и семейный анамнез**,** перечень основных жалоб, обычно приводящих пациента к врачу-терапевту.

На основании компьютерной обработки ответов пациента делаются заключения двух типов:- о наличии риска определенных заболеваний (например, ишемической болезни сердца, сахарного диабета, хронического бронхита, рака легких и др.). В этом случае при отсутствии жалоб указывается, какие дополнительные исследования следует провести;- при наличии тех или иных жалоб пациент адресуется к соответствующему дополнительному (специализированному) опроснику, содержащему углубленную проработку каждого симптома по различным характеристикам, например, точная локализация болей, их характер, иррадиация, продолжительность, сопутствующие явления, условия прекращения и курирования. Для этого по каждому разделу предлагается набор альтернативных ответов.

При обработке полученных ответов дополнительно используются такие элементы основного опросника как, например, сведения о поле, возрасте, анамнезе, факторах риска и других жалобах**.** На этом основании формируется диагностическое заключение в форме синдрома (например, сердечная или легочная недостаточность, синдром малоабсорбации и др.) или определенной нозологической единицы с указанием степени его вероятности в соответствии с

количеством информационных синдромов, полученных от больного (50-75%, 76-95%, >95%). Врач должен учитывать, что диагностическое заключение носит рекомендательный характер. В дальнейшем врач может провести более углубленное обследование пациента, заведя в системе данные объективного осмотра пациента (осмотр головы, живота, ног и т.д.) и результаты дополнительных методов исследования (анализы крови, мочи, кала, мокроты и т.д.). В результате система уточняет раннее выставленные диагнозы.

При наличии нескольких жалоб пациенту предлагается несколько специализированных опросников, по результатам ответов на них формируется диагностическое заключение.

Экспертная система Мутант состоит из следующих подсистем:

- подсистема консультаций (ПК);
- подсистема пополнения знаний (ППЗ).

Моделью представления знаний описываемой **ЭС** является продукционная система.

Фактологическими знаниями (базой фактов) ЭС Мутант являются:

- карта первичного опроса (КПО) и специализированные опросники (СО);
- список диагностируемых заболеваний;
- список анализов, рекомендуемых системой;
- список анализов, которые могут быть сделаны в данном медицинском учреждении;
- список врачей - специалистов, рекомендуемых системой**;**
- список врачей - специалистов, которыми располагает данное медицинское учреждение.

Процедурные знания (базы знаний) для системы содержат правила двух типов:

- правило 1-го типа - Если (условие), то (медицинское заключение)*;*
- правило 2-го типа - Если (условие), то (список опросников). Здесь условием является список выбранных экспертом вопросов из карты первичного опроса и одного из специализированных опросников**.**

Медицинское заключение включает в себя:

- список предполагаемых диагнозов;
- список рекомендуемых системой анализов;
- список рекомендуемых системой врачей-специалистов**.**

Список опросников содержит номера специализированных опросников, которые необходимо предложить пациенту для уточнения медицинского заключения.

База знаний состоит из двух частей, в каждой из которых хранятся правила только одного типа. Каждое правило представлено одним термом. Все термы, соответствующие правилам, относящимся к одному опроснику, и группируются в одну цепь. Каждое правило при занесении в БЗ автоматически получает уникальный номер**,** который не может быть модифицирован, и удаляется из БЗ только при удалении этого правила.

Механизм вывода ЭС Мутант заключается в сопоставлении ответов пациента на вопросы карты первичного опроса и специализированных опросников с условной частью правил из БЗ**.** Поскольку задачей данной системы является выявление всех возможных предварительных заключений, то для ее решения применяется прямой вывод**.** При этом в зависимости от отношения количества ответов пациента, совпавших с условной частью правила 1-го типа, к общему количеству условий в данном правиле, система различает три степени достоверности диагнозов:

- диагноз маловероятен (50-75%);
- диагноз вероятен (76-95%);
- диагноз весьма вероятен (более 95%).

Для работы со знаниями врач-эксперт использует две подсистемы ведения БЗ, каждая из которых предназначена для работы со знаниями 1-го или, соответственно, 2-го типа.

Каждая из подсистем предлагает эксперту следующие виды работы:

- ввод нового правила в БЗ;
- коррекция правила в БЗ;
- удаление правила из БЗ;
- просмотр БЗ.

Применением базы знаний системы Мутант было ее совместное использование с базой данных медицинских карт. Данный подход позволяет не только хранить результаты сеанса работы системы с пациентом, но и дает возможность врачу занести в БД свои собственные наблюдения, результаты анализов и т.п.

В этих системах реализован интерактивный подход в использовании экспертной системы, когда в результате ответов пациента непосредственно формируется заключение. Однако авторы сочли возможным использование экспертной системы Мутант и в скрытом режиме в качестве контролирующей системы.

Экспертная система FIRSTConsult

Ресурс *FIRSTConsult* (прежнее название - PDxMD) является частью MD Consult Clinical Knowledge System. Эта система научно обоснованной электронной клинической информации о первичной медицинской помощи предназначена для улучшения процесса принятия решений специалистами посредством предоставления специализированных диагностических средств и непрерывно обновляемых данных о последних тенденциях в области диагностики, вариантах ведения пациентов и результатах лечения пациентов. Она обеспечивает эффективное принятие решений, сочетая подход на основе передовой практики с гибкостью, необходимой для учета индивидуальных различий между пациентами и предпочтений врача; ускоряет клинические исследования с целью постановки диагноза и оптимизирует результаты, предлагая планирование научно обоснованного лечения и повышая эффективность использования времени и ресурсов; снижает риск ошибок и

улучшает клиническую документацию, предоставляя всеобъемлющую и постоянно обновляемую информацию. Ресурс FIRSTConsult доступен в трех форматах: он-лайн, для карманных компьютеров и в печатном. Формат FIRSTConsult Online обновляется еженедельно и состоит из инструмента «Интерактивная дифференциальная диагностика», научно обоснованных Файлов по медицинским состояниям, Файлов с информацией о пациентах, Справочных центров и Процедурных файлов. Формат FIRSTConsult Handheld включает в себя инструмент «Интерактивная дифференциальная диагностика» и Файлов по медицинским состояниям, обновляемых ежеквартально.

Файлы по медицинским состояниям охватывают более 450 медицинских состояний, которые чаще всего наблюдаются врачами первичной медицинской помощи. Каждое состояние включает в себя самую необходимую краткую информацию из других разделов; справочную информацию, включая коды ICD 9 и эпидемиологию; диагностику, включая дифференциальную диагностику, признаки и симптомы, предлагаемые вопросы для истории болезни, ссылки на литературу, полезную при принятии решений медицинскими учреждениями, и клинические тесты, которые можно провести; терапию с кратким перечнем терапевтических вариантов, подробными сведениями о лекарствах и проблемах пациентов и медицинских учреждений; результаты, включая эффективность терапии, прогноз и осложнения; профилактику с указанием факторов риска, рекомендаций в отношении образа жизни и сохранения здоровья, и скрининг; ресурсы, включая основную литературу и ссылку на информацию для пациента; а также научные данные, увязанные с библиографическими ссылками и клиническими руководствами.

Файлы по дифференциальной диагностике позволяют пользователям ознакомиться с таблицей диагностических алгоритмов, в которой перечисляются потенциальные диагнозы с разбивкой по возрасту и распространенности для более чем 330 признаков и симптомов. Потенциальные критические состояния показаны красным. Выбор диагноза выводит на экран краткую сводку, включая клиническую картину начала заболевания, соотношение процентов среди мужчин и женщин, особенности у разных этносов, характер, географические особенности, клинический картину течения заболевания и сопутствующие заболевания со ссылкой на Файлы по медицинским состояниям. Файлы с информацией о пациентах содержат раздаточные материалы для пациентов, написанные в формате вопросов и ответов. Эти материалы имеют ссылку на Файлы по медицинскому состоянию и могут быть адаптированы под конкретного пациента. Раздаточные материалы доступны различных языках в различных форматах и предназначаются для пациента, у которого было диагностировано данное состояние, либо для общего сведения.

Базовые структуры экспертной медицинской системы дифференциальной диагностики.

1. Входные данные

Входные данные можно условно разбить на два больших блока. Первый блок - это данные, поступающие из пользовательского интерфейса. Второй блок - это содержимое базы знаний, заполненной экспертом. База знаний должна храниться на жестком диске в виде четырех файлов. Структура базы знаний будет описана более подробно в одном из последующих пунктах.

Таким образом, входными данными для разработки алгоритма программы системы медицинской диагностики являются:

- ответы пользователя на вопросы системы;
- база данных с описаниями симптомов;
- база данных с описанием болезней;
- таблица соответствий между болезнями и симптомами;
- таблица «весов» (вероятностей) симптомов для болезней;

2. Выходные данные

Выходными данными экспертной системы является диагноз, построенный на основе наблюдаемых симптомов и базы знаний о болезнях. Этот диагноз появляется на экран дисплея ПВМ.

В процессе работы система предлагает несколько сгенерированных рабочих версий окончательного диагноза, и по требованию пользователя селектирует лишние гипотезы, которые имеют вес, меньший, чем некоторое значение, заранее заданное разработчиком программы или опытным пользователем.

3. Уточнение первоначального списка диагнозов.

Имея начальный список диагнозов, система осуществляет их дифференциацию. Самыми распространенными методами логического вывода являются прямая цепочка рассуждений (прямой вывод) и обратная цепочка рассуждений (обратный вывод). В основном, при решении задач диагностики используется обратный вывод. Можно сказать, что обратный вывод более эффективен, когда пользователь должен выбирать из набора возможных последствий в случае медицинской диагностики. В предлагаемой системе реализуется механизм смешанного вывода, который позволяет и прямой вывод от фактов к заключениям , и обратный – чтобы подтвердить или опровергнуть гипотезу.

В процессе уточнения информации система, задавая пользователю вопросы, осуществляет «отсеивание» лишних гипотез, имеющих малый вес. Для просчета веса гипотез система должна открыть таблицу весов. В таблице весов указывается вес данного симптома для данного диагноза на основе априорных знаний эксперта.

Порог уверенности заранее задается в настройках приложения и использованием интуитивно понятного интерфейса.

4. Принятие окончательного решения.

В процессе предыдущих шагов выявляется несколько версий окончательного результата, которые система должна распределять по порядку возрастания вероятности того или иного диагноза.

Вероятности диагнозов считаются по таблице весов.

Предлагается следующий алгоритм подсчитывания веса:

а) выбирается диагноз из списка диагнозов, сформированного на предыдущих этапах;

б) система просматривает, какие симптомы из списка симптомов имеют отношение к данному диагнозу;

в) суммируются веса, определенные по таблице весов, всех симптомов, имеющих отношение к данному диагнозу;

г) фиксирование конечного веса диагноза;

д) после подсчета весов всех диагнозов выбирается диагноз, имеющий максимальный вес;

е) система выбирает те диагнозы, оценки вероятности которых выше порога.

5. Представление конечного результата.

Система предоставляет врачу диагнозы, которые были выбраны в предыдущем пункте в порядке процентного убывания на экран монитора ПВМ и предоставлять пользователю возможность корректировать базу знаний на основе новых экспертных знаний и своего опыта.

Порядок выполнения работы.

1. Изучить теоретический материал.

2. С помощью поисковых систем найти любую медицинскую экспертную систему дифференциальной диагностики (например, по адресу: //http://ubertech.homedns.org:8083).

3. В режиме опытной эксплуатации системы изучить: интерфейс системы, алгоритм эксплуатации системы, режимы функционирования, формы отображения выходной информации.

4. Ознакомиться через поисковые системы с существующими в настоящие время экспертными системами диагностического характера, построенными на принципах дифференциальной диагностики.

5. Оформить отчет, в который включить: скрин-шоты, отражающие интерфейс экспертной системы с пояснениями по п.3 и результаты сравнительного исследования по п.5 (портал, название, предназначение, реализуемые функции), выводы, аннотацию кратких теоретических сведений размером 350-400 слов**.**

19 . Анализ информативности признаков.

Цель работы: овладение средствами вычислительной техники и методами формирования информативного множества характеризующих состояния пациента признаков.

Краткие теоретические сведения

Анализ различных информационных источников, показывает, что в настоящее время применяются способы формирования множества характеристик биообъекта (состояния организма пациента) основанные на применении следующих методов и алгоритмов:

- Full – полный перебор различных сочетаний характеристик до достижения приемлемого классификационного (диагностического) эффекта,
- Add – последовательное добавление характеристик;
- Del – последовательное исключение характеристик до момента предшествующему исчезновению эффекта; - AddDel – одновременное выполнение процедур алгоритмов Add и Del; - Prob – каждой характеристики биообъекта у признаку определяются веса, а затем применяются процедуры указанных алгоритмов; фрактальный анализ - применяется для тензорных данных;
- Grad - аналогичен алгоритму AddDel, но включение и исключение характеристик в итоговое множество осуществляется не «по одному», а «кортежно».

Во всех случаях используются оценки информативности характеристик, вычисленные, как правило, специализированными для конкретных задач и объектов, во многом субъективными приемами.

Наиболее часто для численной оценки информативности характеристик объекта или процесса по результатам активного или пассивного наблюдения (включая мониторинг и скрининг) применяются статистические методы, учитывающие например: качество кластеризации, на основе методов классификации, на основе энтропии, на основе непараметрических оценок плотности и др.

В простейшем случае для селекции множества информативных признаков используют различные статистические критерии. В данной работе рассмотрим два из них: корреляцию, вариативностью и пересечением доверительных интервалов. Эти критерии могут быть легко получены инструментальными средствами электронных таблиц (например, Excel).

Во всех случаях первоначально задается ограничение на количество признаков в информативном множестве – Nmax.

В первом способе, признаки ранжируются путем анализа матрицы модулей коэффициентов парной корреляционной связи между признаками $R_{i,j}$. Для каждой пары строк i и j соответствующих признаков подсчитывается сумма коэффициентов парной корреляции за вычетом «своей» парной корреляции

$S_{i,j} = \sum_{k=1}^{N}(R_{i,k}) - 1 - R_{i,j}$. Рассчитываем коэффициент «парной информативности» $INF_{i,j} = \frac{R_{i,j}}{S_{i,j}}$. Упорядочиваем пары по мере убывания полученного коэффициента. Осуществляем селекцию признакового пространства, включая в него признаки (доводя размерность множества до Nmax, по мере убывания $INF_{i,j}$).

Во втором случае (способе), для каждого признака определяется вариативность (отношение дисперсии к математическому ожиданию). Следуя гипотезе, что наиболее информативный признак обладает наибольшей изменчивостью, упорядочиваем признаки по мере убывания коэффициентов вариативности и формируем информативное множество.

В первом и втором способе формируем множества информативных признаков для каждого информативного класса, - затем определяется итоговое как пересечение данных множеств с расширение до размерности Nmax путем добавления признаков согласно ранее проведенными ранжированиями.

В третьем способе в каждом альтернативном классе определяются доверительные интервалы для каждого класса, которые упорядочиваются по мере возрастания пересеченности интервалов. Далее формируется необходимое множество. В данном случае применяется гипотеза о том, что наибольшей информативностью обладают наиболее специфические признаки, т.е. имеющие наименьшее пересечение в альтернативных классах.

Порядок выполнения работы

1. Изучите теоретический материал.
2. Задайтесь размерностью множества информативных признаков по формуле $Nmax = N\ mod\ 4 + 3$, где N – Ваш порядковый номер в списке студентов в группе.
3. Сформируйте множество информативных признаков тремя способами, сравните результаты, сделайте выводы. (В качестве исходных данных используйте материалы приложения к лабораторной работе №1).
4. Оформите отчет, включающий полученные результаты и действия по их выполнению, выводы, описание методов определения информативности признаков (не менее 3, не описанных в кратких теоретических сведениях к работе)

Приложение

эр	Hb	ЦП	лейк	эоз	пал	сегм	лимф	мон	заболевание
4,6	142	0,93	7,05	1	14	45	38	2	рак желудка
4,2	115	0,82	7,8	2	4	81	10	2	рак желудка
3,4	107	0,94	8	1	6	64	27	2	рак желудка
3	87	0,87	7,3	4	1	55	36	4	рак желудка
3,4	100	0,88	6,15	0	4	77	17	2	рак желудка

5,0	170	1,02	7,1	2	5	59	30	4	рак желудка
5	150	0,90	3,7	6	1	71	21	1	рак желудка
3,6	105	0,88	17,6	0	7	64	26	2	рак желудка
5,3	108	0,61	4,2	4	1	47	46	2	рак желудка
4,16	132	0,95	5,1	2	4	43	47	4	рак желудка
4,3	145	1,01	4,55	4	5	52	36	3	рак желудка
4,0	128	0,96	7,6	4	2	57	29	7	рак желудка
5,0	160	0,96	6,8	0	4	72	23	1	рак желудка
4,46	146	0,98	6,5	4	1	66	27	2	рак желудка
5,4	176	0,98	7,8	2	2	67	29	0	рак желудка
3,1	108	1,05	3,7	1	2	62	31	4	рак желудка
4,0	124	0,93	4,4	2	7	66	22	3	рак желудка
5,1	164	0,96	7,9	0	5	29	60	6	рак желудка
4,25	138	0,97	6,2	2	7	59	28	4	рак желудка
4,8	160	1,00	6,7	2	2	70	25	1	рак желудка
4,3	150	1,05	12	1	1	59	30	9	рак желудка
4,0	113	0,85	5,85	6	4	48	37	5	рак желудка
4,8	153	0,96	7,1	6	1	61	30	2	рак желудка
3,6	114	0,95	13,1	0	2	73	24	1	рак желудка
3,35	103	0,92	9,1	1	3	58	27	17	рак желудка
4,5	150	1,00	8	4	1	56	36	3	рак желудка
эр	Hb	ЦП	лейк	эоз	пал	сегм	лимф	мон	
4,6	130	0,85	6,7	1	3	44	38	14	Цирроз
3,7	117	0,95	3,6	5	3	50	32	10	цирроз
3,3	110	1,00	4,4	1	3	70	23	3	цирроз
4,4	140	0,95	13,2	3	2	63	30	2	цирроз
4,5	150	1,00	7	1	6	50	34	9	цирроз
4,4	132	0,90	4,4	0	5	60	33	2	цирроз
4,1	136	1,00	5	1	1	70	24	4	цирроз
4	130	0,98	4,9	3	2	71	20	4	цирроз
3,5	115	0,99	7,8	1	6	51	38	4	цирроз
4,2	132	0,94	6	3	4	67	20	6	цирроз
4,3	140	0,98	4,3	2	2	71	17	8	цирроз
4	118	0,89	8,3	0	5	61	26	8	цирроз
4,5	144	0,96	6,4	2	1	60	33	4	цирроз
4,60	150	0,98	5,6	7	4	46	36	7	цирроз
4	130	0,98	7,0	5	6	54	26	9	цирроз
3,9	130	1,00	9,6	1	19	48	15	5	цирроз
4,0	112	0,84	15,8	4	18	61	13	4	цирроз
4,6	150	0,98	5,4	1	2	52	37	8	цирроз
4,6	150	0,98	5,4	2	5	51	34	8	цирроз
4,6	145	0,95	5,2	1	8	68	20	3	цирроз
4,6	150	0,98	6,6	2	3	64	42	12	цирроз
3,2	108	1,01	3	2	5	48	49	6	цирроз

4,5	141	0,94	8,4	2	4	58	24	12	цирроз
4,2	140	1,00	7,1	1	3	60	33	3	цирроз
4	130	0,98	7,0	5	6	54	26	9	цирроз
3,7	112	0,91	3,9	2	1	61	32	4	цирроз
4,0	130	0,98	4	3	1	57	35	4	цирроз
3,7	114	0,92	3,8	3	2	50	41	4	цирроз
4,4	144	0,98	7	2	12	48	26	12	цирроз
эр	Hb	ЦП	лейк	эоз	пал	сегм	лимф	мон	
3,9	109	0,84	4,6	3	3	67	24	3	желчКамБол
4,0	132	0,99	7	3	5	84	10	1	желчКамБол
4,4	144	0,98	9,1	4	7	72	20	1	желчКамБол
3,6	126	1,05	4,4	2	8	54	37	1	желчКамБол
3,5	106	0,91	11,9	1	4	73	18	5	желчКамБол
3,0	100	1,00	6,0	3	1	60	30	6	желчКамБол
3,9	139	1,07	6,4	2	4	59	34	1	желчКамБол
3,8	127	1,00	5,9	3	4	40	50	6	желчКамБол
3,1	103	1,00	10,2	4	1	68	17	10	желчКамБол
3,8	120	0,95	6,8	1	1	58	37	3	желчКамБол
4,1	134	0,98	8,9	2	5	58	36	1	желчКамБол
4,0	120	0,90	7,2	3	1	78	19	2	желчКамБол
3,5	126	1,08	5,3	1	1	51	46	2	желчКамБол
3,5	114	0,98	5,2	3	3	59	33	2	желчКамБол
3,9	126	0,97	9,3	1	11	66	20	2	желчКамБол
4,35	138	0,95	19,0	2	8	86	3	1	желчКамБол
3,8	124	0,98	7,2	2	7	69	27	2	желчКамБол
3,8	128	1,01	10,2	1	8	77	14	1	желчКамБол
3,5	110	0,94	9,4	3	6	68	24	2	желчКамБол
3,5	114	0,98	14,9	2	3	79	15	3	желчКамБол
3,9	126	0,97	9,9	4	5	76	18	1	желчКамБол
3,9	120	0,92	17,9	3	5	67	24	4	желчКамБол
4,5	140	0,93	12,7	1	8	87	3	1	желчКамБол
3,3	110	1,00	3,2	2	1	66	27	4	желчКамБол
4,2	140	1,00	9,6	1	4	67	25	3	желчКамБол
4,1	130	0,95	7,8	3	10	58	26	3	желчКамБол

20. Построение и анализ регрессионных моделей, характеризующих состояние пациента

Цель работы: овладение навыками использования инструментария Excel для построения и анализа структур математических моделей, отражающих регрессии между характеризующих состояние пациента параметров.

Краткие теоретические сведения

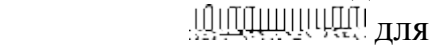
Термину регрессионная модель, используемому в регрессионном анализе, можно сопоставить синонимы: «теория», «гипотеза». Эти термины пришли из статистики, в частности из раздела «проверка статистических гипотез». Регрессионная модель есть прежде всего гипотеза, которая должна быть подвергнута статистической проверке, после чего она принимается или отвергается. Регрессионная модель $f(w,x)$ — это параметрическое семейство функций, задающее отображение $f: W \times X \longrightarrow Y;$

где $w \in W$ - пространство параметров, $x \in X$ - пространство свободных переменных, Y - пространство зависимых переменных.

Так как регрессионный анализ предполагает поиск зависимости математического ожидания случайной величины от свободных переменных $E(y|x) = f(x)$, то в её состав входит аддитивная случайная величина ε: $y = f(w,x) + \varepsilon$.

Предположение о характере распределения случайной величины ν называются гипотезой порождения данных. Эта гипотеза играет центральную роль в выборе критерия оценки качества модели и, как следствие, в способе настройки параметров модели.

Модель является настроенной (обученной) когда зафиксированы её параметры, то есть модель задаёт отображение

 для фиксированного значения $\overline{w}$.

Различают математическую модель и регрессионную модель. Математическая модель предполагает участие аналитика в конструировании функции, которая описывает некоторую известную закономерность. Математическая модель является интерпретируемой - объясняемой в рамках исследуемой закономерности. При построении математической модели сначала создаётся параметрическое семейство функций, затем с помощью измеряемых данных выполняется идентификация модели - нахождение её параметров. Известная функциональная зависимость объясняющей переменной и переменной отклика - основное отличие математического моделирования от регрессионного анализа.

Недостаток математического моделирования состоит в том, что измеряемые данные используются для верификации, но не для построения модели, вследствие чего можно получить неадекватную модель. Также затруднительно получить модель сложного явления, в котором взаимосвязано большое число различных факторов.

Регрессионная модель объединяет широкий класс универсальных функций, которые описывают некоторую закономерность. При этом для построения модели в основном используются измеряемые данные, а не знание свойств исследуемой закономерности. Такая модель часто неинтерпретируема, но более точна. Это объясняется либо большим числом моделей-претендентов, которые используются для построения оптимальной модели, либо большой сложностью модели. Нахождение параметров регрессионной модели называется обучением модели.

Недостатки регрессионного анализа: модели, имеющие слишком малую сложность, могут оказаться неточными, а модели, имеющие избыточную сложность, могут оказаться переобученными.

Примеры регрессионных моделей: линейные функции, алгебраические полиномы, ряды Чебышёва, нейронные сети без обратной связи, например, однослойный персептрон Розенблатта, радиальные базисные функции и прочее.

И регрессионная, и математическая модель, как правило, задают непрерывное (часто функциональное) отображение. Требование непрерывности обусловлено классом решаемых задач: чаще всего это описание физических, химических и других явлений, где требование непрерывности выставляется естественным образом.

Иногда на отображение f накладываться ограничения монотонности, гладкости, измеримости, и некоторые другие. Теоретически, никто не запрещает работать с функциями произвольного вида, и допускать в моделях существование не только точек разрыва, но и задавать конечное, неупорядоченное множество значений свободной переменной, то есть, превращать задачи регрессии в задачи классификации.

LRM - Линейная регрессионная модель

Линейный регрессионный анализ - это самый распространенный инструмент для описания связи между факторами и какой-то зависимой величиной. Применяя модель, надо обязательно проверять выполняются ли предпосылки для использования линейной модели. Если мы перешли через границы применимости используемого метода оценивания, то интерпретация полученных результатов будет некорректной.

Сформируем основные предпосылки:

1. Нулевое математическое ожидание ошибок;
2. Диагональность ковариационной матрицы ошибок;
3. Отсутствие гетероскедастичности (неоднородности в наблюдениях, отсутствие определенного и относительно постоянного закона распределения случайной величины) в модели.

Нарушение любой из этих предпосылок ведет к искажению полученных результатов. Можно не обнаружить существующую зависимость или построить ложную модель. Поэтому, за кажущейся простотой метода скрывается целый комплекс проблем, неочевидных на первый взгляд.

GRM - Общая регрессионная модель

Для построения общей регрессионной модели используется общий метод наименьших квадратов, который имеет более широкую область применения и строит хорошие модели даже при отсутсвии выполнения некоторых предпосылок, указанных выше.

Одной из чаще всего используемых возможностей Excel является *экстраполяция* данных – например, для анализа имеющихся фактических данных, оценки тенденции их изменения и получения на этой основе краткосрочного прогноза на будущее.

В этом случае используется линейная экстраполяция данных на основе наименьшего квадратичного отклонения – отыскивается линейная зависимость данных, такая, которая бы минимизировала сумму квадратов разностей между имеющимися фактическими данными и соответствующими значениями на прямой линейного тренда (интерполяционной или экстраполяционной зависимости). На основе найденной зависимости можно сделать статистически оправданное предположение об ожидаемых будущих значениях изучаемого ряда данных.

Парные регрессионные модели можно построить с помощью встроенных функций инструментария Excel, для параметрической идентификации множественной регрессии необходимо использовать «Пакет анализа». Этот инструментарий, помимо всего прочего, умеет рассчитывать параметры регрессии, по тому же МНК, всего в несколько кликов, собственно, о том как этим инструментом пользоваться дальше и пойдет речь.

Активируем Пакет анализа

По умолчанию эта надстройка отключена и в меню вкладок вы ее не найдете, поэтому пошагово рассмотрим как ее активировать.

Активируем вкладку Файл, в открывшемся меню ищем пункт Параметры и кликаем на него.

В открывшемся окне, слева, ищем пункт Надстройки и активируем его, в этой вкладке внизу будет выпадающий список управления, где по умолчанию будет написано Надстройки Excel, справа от выпадающего списка будет кнопка Перейти, на нее и нужно нажать.

Всплывающее окошко предложит выбрать доступные надстройки, в нем необходимо поставить галочку напротив Пакет анализа и заодно, на всякий случай, Поиск решения (тоже полезная штука), а затем подтвердить выбор кликнув по кнопочке ОК.

Приведем *инструкцию по поиску параметров линейной регрессии с помощью Пакета анализа.*

После активации надстройки Пакета анализа она будет всегда доступна во вкладке главного меню Данные под ссылкой Анализ данных. В активном окошке инструмента Анализа данных из списка возможностей ищем и выбираем Регрессия. Далее откроется окошко для настройки и выбора исходных данных для вычисления параметров регрессионной модели. Здесь нужно указать интервалы исходных данных, а именно описываемого параметра

(Y) и влияющих на него факторов (X), как это на рисунке ниже, остальные параметры, в принципе, необязательны к настройке.

Расчетные значения модели и прогноз

Модель строится не только чтобы показать величину зависимостей изучаемого параметра от влияющих факторов, но и для прогноза. На рисунке ниже эти расчеты сделаны в экселе в отдельном столбце.

Фактические значения (те что имели место в реальности) и расчетные значения по модели на этом же рисунке отображены в виде графиков, чтобы показать разность, а значит погрешность модели.

Регрессия позволяет осуществлять прогноз и во времени - это делают с помощью авторегрессионной модели - модели, в которой влияющими факторами являются сам исследуемый объект и время, то есть моделируется зависимость показателя от того каким он был в прошлом. (В более общем смысле, в качестве аргументов авторегресионной модели могут выступать и различные другие факторы и различные лаги). Включение в регрессионные модели переменных, измеренных на порядковом и номинальном уровнях, является во многих случаях абсолютно необходимой задачей – в том числе диагностического характера. Построив регрессионные модели в различных состояниях (кластерах) здоровья человека, можно по анализу структурно-параметрических различий осуществлять диагностический процесс (особенно, в случае дифференциальной диагностики).

При экстраполяции нельзя далеко уходить от экспериментальной области. За ее приделами характер зависимости может измениться.

Причинно-следственная связь в социально-гигиеническом мониторинге (СГМ) и место регрессионных моделей в схеме доказательства.

Одна из важнейших задач СГМ - установление (доказательство наличия) причинно-следственных связей между здоровьем населения и внешними факторами. Принципы доказательства наличия причинно-следственных связей, которые стали классическими и цитируются во многих литературных источниках, сформулировал английский статистик О.Б. Хилл в 1965 г. Однако по прошествии 40 лет в Российской гигиенической науке находится немало исследователей, своими трудами опровергающих эти принципы.

Приведем, принципы О.Б. Хилла. В данном случае речь идет о том, какого рода доказательства дают основания предполагать, что связь между явлениями обусловлена причинно-следственными отношениями. По мнению авторов, в клинической эпидемиологии невозможно доказать причинно-следственный характер связи, полностью исключив все сомнения.

В частности, для доказательства причинной связи необходимо устранить систематические и случайные ошибки при отборе пациентов и при измерении их параметров, оценить наличие вмешивающихся (косвенных) факторов, что не всегда возможно.

Приведем следующие доказательства наличия возможной причинно-следственной связи.

1. Последовательность событий во времени (причина должна предшествовать следствию). Например, в Лондоне в 1952-1954 г. в отдельные периоды времени сначала отмечалось резкое увеличение загрязнения атмосферного воздуха, а затем рост заболеваемости. Такая временная последовательность событий является определенным доказательством того, что загрязнение ОС может быть причиной роста заболеваемости. Нарушение последовательности событий служит сильным аргументом против возможной причинно-следственной связи.

2. Сила эффекта. Причинно-следственная связь кажется более вероятной, если причина вызывает сильный эффект, например, увеличение в 15 раз распространенности рецидивирующего бронхита у детей, проживающих в зонах экологического неблагополучия, по сравнению с контролем служит веским аргументом в пользу причинно-следственной связи между загрязнением атмосферного воздуха и появлением бронхита.

3. Зависимость эффекта от дозы (при усилении воздействия заболеваемость повышается). Достаточно сильным аргументом в пользу причинно-следственной связи может служить наблюдаемое усиление эффекта при усилении воздействия, например, явно выраженное увеличение смертности от рака легкого при увеличении количества выкуриваемых сигарет.

4. Обратимость (при ослаблении воздействия заболеваемость снижается). Так, заболеваемость органов дыхания у жителей Лондона резко сократилась после принятия «Акта о чистом воздухе». По мнению авторов , обратимость связи – сильное, хотя и не бесспорное доказательство ее причинно-следственного характера.

5. Устойчивость, воспроизводимость эффекта (эффект наблюдается разными исследователями независимо от места, условий и времени). Очевидно, что наличие эффекта в одном месте и его отсутствие в другом (при равных условиях), свидетельствует не в пользу причинно-следственной связи между предполагаемыми причиной и следствием.

6. Биологическое правдоподобие (эффект согласуется с современными научными представлениями). В конкретном случае изучения связи здоровья населения с загрязнением ОС ряд авторов говорит о необходимости этиопатогенетического анализа связи конкретных изменений здоровья населения с конкретными вредными факторами среды обитания. При наличии такого анализа обнаружение действия каких-либо токсикантов ОС на увеличение распространенности конкретной патологии выглядит значительно более правдоподобным, чем без него.

7. Специфичность (одна причина приводит к одному эффекту). В области СГМ, когда изучается воздействие на популяцию комплекса причин одновременно, условие специфичности может оказаться не очень актуальным.

8. Аналогия (причинно-следственная связь уже установлена для сходного воздействия или болезни). Доказательство причинно-следственной связи усиливает наличии аналогичных примеров для точно установленных причин.

Таким образом, важнейшая задача СГМ – установление причинно-следственных связей между здоровьем населения и внешними факторами – решается путем комплексного исследования, включающего ряд этапов. Какова роль статистических моделей, и, более конкретно моделей регрессионного типа в этом комплексном исследовании?

Корреляционная связь (и ей соответствующая регрессионная модель) не является причинно-следственной; ее наличие может лишь указывать на существование возможной причинно-следственной связи. Более того, корреляционная связь (даже статистически значимая) может вообще не иметь никакого отношения к причинно-следственной связи. Во-первых, в статистических исследованиях встречаются ложные (случайные) корреляции, которые ни на чем не основаны (просто противоречат здравому смыслу); их появление обусловлено случайным стечением обстоятельств при формировании выборки из генеральной совокупности и не должно приниматься во внимание именно из-за противоречия здравому смыслу. Во-вторых, возможно существование признаков, действительно связанных между собой, когда ни один из них не является причиной или следствием другого. Пример такой связанной пары признаков – рост и вес человека. Их взаимосвязь очень сильна, однако нельзя сказать, что высокий рост является причиной большого веса, и наоборот. Просто эти два признака связаны с развитием всего организма, а не друг с другом.

Для того чтобы статистические модели выполняли свою функцию одного из «блоков» в системе доказательства причинно-следственных связей, они должны быть корректно построены, а полученные результаты корректно представлены и проанализированы. Такая работа может быть выполнена только совместными усилиями специалистов в области СГМ и специалистов в области моделирования. В идеальном варианте специалисты по мониторингу должны иметь представление об основах моделирования, а специалисты по моделированию – опыт построения и трактовки моделей в области мониторинга. С.А. Айвазян, перечисляя актуальные направления исследований в области прикладной математической статистики, называет, среди прочих, следующее направление: формализация (математическая постановка) реальных задач статистического анализа данных в различных предметных областях; выработка на базе этого опыта типовых математических постановок задач, выходящих за … рамки жестких канонических моделей. Это важный и трудный этап математико-статистического моделирования….

Искусство реалистического моделирования формально не предусмотрено ни в одном из разделов статистической науки, его развитие никак и ничем не стимулируется. Для выполнения этого этапа необходимо сочетание глубокого знания математических основ статистической обработки информации и опыта

выполнения собственных исследований на конкретном (реальном) статистическом материале.

Крайним выражением «пользовательских настроений» среди медиков можно считать высказывание: «От нас не требуется теперь глубоких, детальных знаний математики, поскольку врачей-исследователей, так сказать, «ведет компьютерная программа анализа».

Таким образом:

- Во – первых, большинство заболеваний, которые исследуются в социально-гигиеническом мониторинге – это экологически зависимые болезни, связь которых с факторами ОС существует, но она не столь сильна, чтобы быть очевидной. Вследствие этого, доказательство наличия связи между состоянием здоровья населения и состоянием ОС является непростой задачей; в ряде случаев решить такую задачу вообще не удается (не удается доказать наличие влияния факторов окружающей среды на здоровье населения).
- Во вторых, модели регрессионного типа предназначены для первичной проверки гипотез (выдвинутых по соображениям предметной области) о наличии статистической связи между показателями здоровья и факторами ОС. Существенное значение в доказательстве связи с участием регрессионных моделей имеет первичная информация, выбранная исследователем для статистического анализа. Выбранные показатели (показатели здоровья населения и состояния ОС) должны быть максимально связаны между собой с предметной точки зрения (с точки зрения этиологии), а также быть объективными и непротиворечивыми. Желательно, чтобы показатели были выбраны с учетом требований, которые выдвигает доказательная медицина к установлению причинно-следственных связей.
- В третьих, в системе доказательства наличия причинно-следственных связей модели регрессионного типа являются одним из необходимых, но явно недостаточных звеньев. Поэтому выводы, сделанные только по результатам моделирования, должны быть взвешенными, аргументированными и предельно осторожными (предположительными).

В настоящее время имеются различные статистические пакеты обработки данных, с помощью которых можно эффективно решать на компьютере статистические задачи для случайных величин с нормальным (гауссовским) законом распределения. Однако экспериментальные исследования показывают, что отклонение от нормального закона распределения характерны для погрешностей различных измерительных приборов. В случае негауссовских распределений случайных величин возможности решения статистических задач резко сужаются, поскольку во многих случаях отсутствуют проверенные и эффективные методики решения таких задач аналитическими или численными методами.

Порядок выполнения работы

1. Изучить теоретический материал

2. Изучение и овладение навыками применения возможностей регрессионного моделирования в диагностических целях средствами Excel. По данным лабораторной работы №1 (см. Приложение):
- сформировать множество информативных признаков (количество признаков – 4, идентификаторы признаков определяются по датчику случайных чисел равномерного распределения в диапазоне (1, n*5), где n - порядковый номер студента в группе, от полученного в заданном диапазоне чисел определяется остаток от деления на 4 и прибавляется 1. По полученному числу последовательно определяется идентификатор (имя) признака. Процедура продолжается до тех пор пока не будет сформировано множество из 4 признаков.
- формируются обучающие и экзаменационные выборки в каждом классе (рекомендуется использовать метод «золотого сечения» соотношения размеров выборок);
- определяются парные лучшие регрессии (общим количеством 12) в Excel (см. лабораторную работу, посвященную корреляционному анализу) и составляется граф-связности по значимым моделям – в каждом классе;
- определяется множественная линейная регрессия между признаками (регистрируемыми параметрами) – в каждом классе;
- сравниваются результаты предыдущих вычислительных экспериментов;
- считая полученные регрессионные модели прогностическими осуществляется вычисление ошибок прогнозирования (средних относительных) на экзаменационных выборках в каждом классе (парные и множественные), делаются выводы о точности прогноза в классах и возможностях использования моделей в диагностических целях (средняя относительна ошибка не должна превышать уровень 0,32).
3. Исследуются информационные возможности авторегрессионных моделей. Выдвигается гипотеза о том, что авторегрессионная модель несет информации о функциональной зависимости динамики временного ряда биологического сигнала. Исследования проводятся по следующим этапам:
- задаются функции (не менее 50 измерений): линейная, параболическая, гармоническая, реальная (ФПГ – до и после нагрузки);
- путем использования дифференциального исчисления (первой производной) найдите формулу для авторегрессии первого порядка (зависимости значения функции от предыдущего состояния)
- постройте авторегрессионные модели первого порядка для всех указанных функций (включая сигнала ФПГ);
- сделайте выводы о качестве авторегрессионных функций и их прогностических (диагностических) возможностях.
4. Оформите отчет, включающий: результаты проведенных исследований со сделанными умозаключениями (выводами), краткими ответами на контрольные вопросы (не менее пяти : 5,9, 10 вопросы – обязательны), краткими резюме о содержании информационных источников, упомянутых в библиографии.

Контрольные вопросы:

1. Как выбрать тип и структуру модели, какому именно семейству она должна принадлежать?
2. Какова гипотеза порождения данных, каково распределение случайной переменной?
3. Какой целевой функцией оценить качество аппроксимации?
4. Каким способом отыскать параметры модели, каков должен быть алгоритм оптимизации параметров?
5. Что подразумевается под восстановлением значения по регрессионной модели?
6. Что такое экстраполяция?
7. Как проверяется адекватность регрессионных моделей?
8. Что такое множественная линейная регрессия?
9. Как синтезировать множественную нелинейную регрессию с различными опорными функциями?
10. Каким образом построить регрессионные модель (включая авторегрессии) для нефункциональных зависимостей?

Информационные источники

1. Инструкция создание регрессионных моделей в MS Excel (видео-урок) /URL:https://www.youtube.com/watch?v=3hlCwCzHUv4
2. Линейная регрессия в Excel через Анализ данных /URL: http://archie-goodwin.net/load/specializirovannye_blogi/ms_office /linejnaja_regressija_v_excel_cherez_analiz_dannykh/28-1-0-391
3. Линейная регрессионная модель /URL: http://nickart.spb.ru /analysis/lrm.php
4. Общие регрессионные модели. /URL: http://www.statsoft.ru /home/textbook/modules/stgenregmod.html
5. Роль математики в медицине /URL: http://stud24.ru /mathematic/rol-matematiki-v-medicine/174048-506827-page3.html

21. Разведочный статистический анализ в доказательной медицине

Цель работы: овладение навыками использования «Пакета анализа» в Excel для проведения разведочного статистического анализа медицинских данных в рамках методологии доказательной медицины.

Краткие теоретические сведения

Доказательная медицина (Evidence-based medicine — меицина, основанная на доказательствах) – это подход к медицинской практике, при котором решения о применении профилактических, диагностических и лечебных мероприятий принимаются исходя из имеющихся доказательств их эффективности и безопасности, а такие доказательства подвергаются поиску, сравнению, обобщению и широкому распространению для использования в интересах пациентов.

В основе доказательной медицины лежит проверка эффективности и безопасности методик диагностики, профилактики и лечения в клинических исследованиях. Под практикой доказательной медицины понимают использование данных, полученных из клинических исследований, в повседневной клинической работе врача.

Некоторые правила проведения клинических исследований изложены в стандарте GCP (Good Clinical Practice, «Надлежащая клиническая практика»), правила производства лекарственных средств — в стандарте GMP, правила выполнения лабораторных исследований — в стандарте GLP.

В начале 1990-х годов разработана рейтинговая система оценки клинических исследований, где с возрастанием порядкового номера доказательности качество клинических исследований снижается. Уровни принято обозначать римскими цифрами (I, II, II, III, IV) или буквами латинского алфавита (A, B, C, D). Цифры обозначают уровень доказательность результатов научных исследований. Буквы обозначают уровень доказательности принятых рекомендаций.

Класс (уровень) I (A): большие двойные слепые плацебо контролируемые исследования, а также данные, полученные при мета-анализе нескольких рандомизированных контролируемых исследований.

Класс (уровень) II (B): небольшие рандомизированные контролируемые исследования, в которых статистические расчёты проводятся на ограниченном числе пациентов.

Класс (уровень) III (C): нерандомизированные клинические исследования на ограниченном количестве пациентов.

Класс (уровень) IV (D): выработка группой экспертов консенсуса по определённой проблеме.

Международная система доказательной медицины развивается в геометрической прогрессии: с момента её становления в начале 90-х годов и по

настоящее время число центров, монографий и форумов по проблеме исчисляется десятками, количество публикаций - сотнями. Агентство политики здравоохранения и науки США субсидировало в 1997 году сроком на 5 лет 12 таких центров, созданных при ведущих университетах и научных организациях различных штатов; растёт число центров по отдельным проблемам (здоровье детей, первичная помощь, общая практика, психическое здоровье и др.).

Общим для всего направления является использование принципа доказательности на любом уровне принятия решений — от государственной программы до назначения индивидуальной терапии.

Таким образом, доказательный подход к медицинской практике подразумевает использование лечебных, профилактических и диагностических действий с доказанной эффективностью, что предполагает поиск информации, сравнение, проведение исследований и мета-анализов. Эффективность доказывается не собственным опытом или личным мнением.

В узком понимании подразумевается такой способ медицинской практики отдельного врача, когда он использует в своей работе только то, что имеет качественную доказательную базу эффективности (к этому не относится собственный опыт и личное мнение, они ставятся на последнее место — когда всё доказанное кончилось или случай неординарный, тогда и личный опыт подойдёт). Ясно, что объективность сама по себе субъективна, особенно в медицине, но личный опыт субъективнее всего остального, поэтому к нему так и относятся.

Подход к лечению только на основе опыта и впечатлений называется «импрессионистская медицина».

В основе ДМ лежит клиническая эпидемиология, которая изучает распространение заболеваний и разрабатывает методологию клинических испытаний таким образом, чтобы они приводили к научно обоснованным выводам, минимизируя случайные и систематические ошибки, а также заинтересованность участников. В целом это такой свод принципов для опровержения или подтверждения научных данных.

Противники доказательной медицины любят разглагольствовать в стиле, мол, если не доказано действие, ещё не значит, что его нет. Такой подход отсылает нас к области веры, а не науки: именно в религии не нужно ничего доказывать. Так что не стоит обращать внимания на такие логические ошибки, это просто аргумент адептов астрала.

Последнее время контроль нередко проводится не с плацебо, а с конвенциональным лечением (уже утверждённым и многократно проверенным), в тех случаях, когда нельзя пациентов оставить совсем без лечения. Тогда одной группе дают стандартную терапию, а другой стандартную + исследуемую, и сравнивают различия. Каждое исследование должно иметь подтверждения своих выводов в массе подобных, иначе практические выводы сделать нельзя; для анализа и оценки статей есть толпы экспертов, которые в итоге выдают простым смертным врачам клинические рекомендации (гайдлайны,

«guidelines») на основе лучших доказательств. При изучении работ надо строго разделять показатели процесса (любые изменения параметров) и показатели собственно результата (именно они имеют клиническую значимость), к которым приводят те изменения. Во время чтения публикаций или спора со сторонником какого-либо метода нужно придерживаться этого разделения, поскольку не составляет труда показать действие возможного фактора на процесс, а вот выяснение достоверного результата и его положительной связи с тем фактором требует серьёзной работы.

Исследовательские работы можно отнести к тому или иному виду по весу доказательности, что зависит от его структуры (по уменьшению крутизны):

Систематический обзор посредством мета-анализа: формируется множество пачка схожих клинических испытаний одного метода, высчитываются их общие и различные параметры, анализируется совпадение/несовпадение результатов.

РКИ (рандомизированное клиническое исследование, «randomized clinical trial», «RCT») является основой доказательности доказательности - различияь, что же явилось следствием воздействия, а что — случайностью. Заключается в динамическом наблюдении профилактического – диагностического - лечебного вмешательства, которые применяются к случайно сформированным (рандомизированным) группам из конкретной выборки пациентов. Все возможные факторы одинаково действуют на группы подопытных, только в одной это будет полностью плацебо-эффект, а во второй — непосредственно эффект медицинского вмешательства, из которого можно вычесть первый и получить кристаллизованную достоверность в виде подтверждения/опровержения изначальной гипотезы.

Популяционное (проспективное, когортное, продольное) исследование осуществляется следующим образом: выделяются два отряда населения (когорты), например, которые подвергались фактору риска и не подверженные ему, затем за ними длительно наблюдают, обследуют и сравнивают данные. Используется для определения прогноза и причин заболеваний, их факторов риска и уровня заболеваемости, что весьма трудоёмко из-за необходимости больших выборок (новые случаи заболеваний могут оказаться слишком редкими) и длительности наблюдения этих больших групп.

Аналитическое одномоментное исследование (поперечное): используется для оценки эффективности диагностики, распространённости исходов и течения заболеваний практически в реальном времени — по сути это срез базы данных по каким-либо критериям.

Исследование случай—контроль (ретроспективное): осуществляется статистический анализ архива историй болезни, что позволяет получить относительно точные данные (без внешнего влияния — ведь все наблюдения произошли уже до анализа), на основе которых вполне можно выдвинуть какую-либо гипотезу.

Описание серии случаев: широко используется, но по сути это тот самый «личный многолетний опыт», доказательной ценности имеющий совсем

чуть-чуть, поскольку если человек хочет что-то видеть, то он увидит это и на первый, и на десятый, и на тысячный раз.

Обзор ресурсов доказательной медицины.

Кокрановское сотрудничество. «Cochrane Collaboration» - крупнейшая международная организация для предоставления пруфов или опровержения лекарств/методов с помощью метаанализов, объединяя многие разнородные данные в общие темы. Выводы мета-анализов Кокрейна намного более важны, чем чьи-либо личные сведения о многолетнем опыте применения какого-либо фуфломицина.

NCBI - The National Center for Biotechnology Information - Национальный (американский, естественно) центр биотехнологической информации. Нельзя утверждать, что там есть всё, но базы данных крупнее, чем у них, попросту не существует.

"Data Mining". StatSoft определяет понятие "Data Mining" как процесс аналитического исследования больших массивов информации (обычно экономического характера) с целью выявления определенных закономерностей и систематических взаимосвязей между переменными, которые затем можно применить к новым совокупностям данных. Этот процесс включает три основных этапа: исследование, построение модели или структуры и ее проверку. В идеальном случае, при достаточном количестве данных можно организовать итеративную процедуру для построения устойчивой (робастной) модели. В то же время, в реальной ситуации практически невозможно проверить экономическую модель на стадии анализа и поэтому начальные результаты имеют характер эвристик, которые можно использовать в процессе принятия решения (например, "Имеющиеся данные свиделельствуют о том, что у женщин частота приема снотворных средств увеличивается с возрастом быстрее, чем у мужчин.").

Методы Data Mining приобретают все большую популярность в качестве инструмента для анализа экономической информации, особенно в тех случаях, когда предполагается, что из имеющихся данных можно будет извлечь знания для принятия решений в условиях неопределенности. Хотя в последнее время возрос интерес к разработке новых методов анализа данных, специально предназначенных для сферы бизнеса (например, Деревья классификации), в целом системы Data Mining по-прежнему основываются на классических принципах разведочного анализа данных (РАД) и построения моделей и используют те же подходы и методы.

Имеется, однако, важное отличие процедуры Data Mining от классического разведочного анализа данных (РАД) : системы Data Mining в большей степени ориентированы на практическое приложение полученных результатов, чем на выяснение природы явления. Иными словами, при Data Mining нас не очень интересует конкретный вид зависимостей между переменными задачи. Выяснение природы участвующих здесь функций или

конкретной формы интерактивных многомерных зависимостей между переменными не является главной целью этой процедуры.

Основное внимание уделяется поиску решений, на основе которых можно было бы строить достоверные прогнозы. Таким образом, в области Data Mining принят такой подход к анализу данных и извлечению знаний, который иногда характеризуют словами "черный ящик". При этом используются не только классические приемы разведочного анализа данных, но и такие методы, как нейронные сети , которые позволяют строить достоверные прогнозы, не уточняя конкретный вид тех зависимостей, на которых такой прогноз основан.

Data Mining часто рассматривается как естественное развитие концепции хранилищ данных.

Система STATISTICA предлагает пользователю широкий выбор методов разведочного анализа данных. Программа вычисляет практически все используемые описательные статистики общего характера: медиану, моду, квартили, заданные пользователем процентили, среднее значение и стандартное отклонение, квартильный размах, доверительные интервалы для среднего, асимметрию и эксцесс (и их стандартные ошибки), гармоническое и геометрическое среднее, а также многие специальные описательные статистики. Как и во всех других модулях системы STATISTICA, проведение разведочного анализа данных поддерживают разнообразные графики и диаграммы, в т.ч. различные виды диаграмм размаха и гистограмм, гистограммы двумерных распределений (трехмерные и категоризованные), двух- и трехмерные диаграммы рассеяния с помеченными подмножествами данных, нормальные и полунормальные вероятностные графики и графики с исключенным трендом, К-К и В-В графики и т.д. Имеется набор критериев для подгонки нормального распределения к данным (критерии Колмогорова-Смирнова, Лилиефорса и Шапиро-Уилкса).

Процедуры для подгонки многих других типов распределений можно найти также в описании анализа процессов и графических возможностей системы. Практически все описательные статистики и графики могут быть построены для данных, категоризованных (сгруппированных) по значениям одной или нескольких группирующих переменных. Например, с помощью нескольких щелчков мыши можно сгруппировать имеющиеся данные о людях по полу и возрасту и затем просмотреть категоризованные гистограммы, диаграммы размаха, графики на нормальной вероятностной бумаге, диаграммы рассеяния и т.д. В случае, если было выбрано более двух категоризующих переменных, автоматически будет построен каскад соответствующих графиков.

Имеется возможность производить категоризацию по числовым (непрерывным) переменным, например, можно потребовать, чтобы значения переменной были разбиты на заданное число интервалов; с помощью средства перекодировки в реальном времени можно задать конкретный специальный способ перекодировки переменной (возможности практически сколь угодно сложной перекодировки доступны в любой момент, причем перекодировка может быть

задана через соотношения между любыми переменными файла данных). В дополнение к этому в системе имеется специализированная процедура иерархической группировки, позволяющая осуществлять категоризацию данных по многим (до шести) переменным и строить различные категоризованные графики, описательные статистики и корреляционные матрицы для подгрупп (пользователь может в интерактивном режиме сделать неучитываемыми некоторые из факторов в полной таблице группировок и изучать статистики для маргинальных таблиц).

Многочисленные возможности форматирования и расстановки меток позволяют получать таблицы и отчеты презентационного качества, содержащие длинные метки и описания переменных. При этом важно отметить, что процедура группировки выполняется для чрезвычайно больших объемов данных (например, по одной категоризующей переменной можно построить до 300 групп), а ее результаты содержат все соответствующие статистики дисперсионного анализа (включая полные таблицы ANOVA, критерии проверки гипотез типа критерия Левена однородности дисперсии, семь различных апостериорных (post-hoc) критериев и т.д.).

Как и во всех других модулях системы STATISTICA, для достижения высокой - не имеющей аналогов в сравнении с другими пакетами - точности результатов здесь можно производить вычисления с повышенной точностью (если нужно - с четырехкратной). Благодаря интерактивному характеру системы изучение данных становится очень простым делом. Например, графики и диаграммы для разведочных статистик можно получать непосредственно из данных любых выходных таблиц, просто указав мышью на отдельные ячейки или группы ячеек. Одним щелчком мыши можно получать каскады графиков (в том числе сложных, например, со множественными категориями), которые затем можно просматривать в режиме подобном демонстрации слайдов, просто нажимая кнопкуПродолжить. В дополнение к большому числу готовых статистических графиков пользователь может самостоятельно задавать различные типы визуализации исходных данных, описательных статистик, взаимосвязей между статистиками, группировок и категоризаций с помощью средств прямого доступа (point-and-click), что позволяет существенно уменьшить требуемое количество действий мышью. Средства графического разведочного анализа объединены с собственно статистичекими процедурами, что существенно облегчает визуальный анализ данных (например, в интерактивном режиме можно удалять выбросы, выделять подмножества данных, осуществлять сглаживание и подгонку функций, а богатые средства работы с кистью позволяют легко выявлять и/или выделять нужные данные).

Разведочный статистический анализ данных средствами Excel. Microsoft Excel предлагает средства для анализа статистических данных. Такие встроенные функции, как СРЗНАЧ (AVERAGE), МЕДИАНА (MEDIAN) и МОДА (MODE), могут использоваться для проведения анализа данных. Если встроенных

статистических функций недостаточно, необходимо обратиться к пакету Анализ данных.

Пакет ***Анализ данных***, являющийся надстройкой, содержит коллекцию функций и инструментов, расширяющих встроенные аналитические возможности Excel. В частности, пакет Анализ данных можно использовать для создания гистограмм, ранжирования данных, извлечения случайных или периодических выборок из набора данных, проведения регрессионного анализа, получения основных статистических характеристик выборки, генерации случайных чисел с различным распределением, а также для обработки данных с помощью преобразования Фурье и других преобразований.

Пакет Анализ данных доступен при каждом запуске Excel. Функции пакета Анализ данных можно использовать точно так же, как и любые другие функции Excel, а чтобы получить к ним доступ, выполните описанные ниже действия:

1. Выберите в меню Сервис команду Анализ данных. При первом выборе этой команды Excel загружает файл с диска. Затем на экране появится окно диалогаАнализ данных

2. Чтобы использовать какой-либо из инструментов анализа, выберите его имя в списке и нажмите кнопку ОК.

3. Заполните открывшееся окно диалога. В большинстве случаев это означает задание входного диапазона с данными, которые вы собираетесь анализировать, задание выходного диапазона, куда должны быть помещены результаты, и выбор нужных параметров.

Если команда *Анализ данных* отсутствует в меню *Сервис* или формула, содержащая функцию из пакета анализа, возвращает ошибочное значение #ИМЯ?(#NAME?), выберите в меню Сервис команду Надстройки, затем *Пакет анализа* в списке надстроек, после чего нажмите кнопку ОК. Если Пакет анализаотсутствует в списке надстроек, вы должны установить его, запустив программу Setup.

При анализе данных часто возникает необходимость определения различных статистических характеристик или параметров распределения.

С помощью Microsoft Excel можно анализировать распределение, используя несколько инструментов:

- встроенные статистические функции,
- функции для оценки разброса данных,
- инструмент Описательная статистика (Descriptive Statistics), который предоставляет удобные сводные таблицы основных параметров распределения,
- инструменты Гистограмма (Histogram),
- Ранг и персентиль (Rank and Percentile).

Встроенные статистические функции Microsoft Excel применяются при проведении статистического анализа данных. Кроме них Excel также предлагает

более сложные функции ЛИНЕЙН (LINEST), ЛГРФПРИБЛ (LOGEST), ТЕНДЕНЦИЯ (TREND) и РОСТ (GROWTH), которые работают с числовыми массивами.

Описательная статистика (Descriptive Statistics) позволяет создать таблицу основных статистических характеристик для одного или нескольких множеств входных значений. Выходной диапазон содержит таблицу со статистическими характеристиками для каждой переменной входного диапазона: среднее, стандартная ошибка, медиана, мода, стандартное отклонение и дисперсия выборки, коэффициент эксцесса, коэффициент асимметрии, размах, минимальное значение, максимальное значение, сумма, количество значений, k-е наибольшее и наименьшее значения (для любого заданного k) и доверительный интервал для среднего.

Для использования режима *Описательная статистика* в меню *Сервис* выберите команду *Анализ данных*, затем в списке *Инструменты* анализа окна диалога *Анализ данных* выберите инструмент *Описательная статистика* и нажмите кнопку *ОК*. Появится окно диалога,

Инструмент *Описательная статистика* требует задания входного диапазона, который может содержать одну или несколько переменных, и выходного диапазона. Необходимо указать, как расположены переменные в столбцах или в строках. Установите флажок *Метки* в первой строке, если первая строка во входном диапазоне содержит названия столбцов. Excel использует эти метки для создания заголовков в выходной таблице.

Чтобы получить представленную выше таблицу статистических характеристик, установите флажки в области *Параметры вывода*.

Подобно другим инструментам пакета анализа, *Описательная статистика* создает *таблицу констант*. Если эта таблица не устраивает исследователя, можно получить большинство из перечисленных ниже статистических характеристик с помощью других инструментов пакета анализа или формул с использованием встроенных функций Excel.

Анализ данных можно осуществить и через «***Диспетчер сценариев***». Чтобы провести анализ данных в Excel 2010 необходимо активировать «Диспетчер сценариев». Это идет по схеме: вкладка «Данные», активируем кнопочку «Работа с данными» - «Анализ «Что если»» - «Сценарии». После увидите окошко, в котором необходимо задействовать команду «Добавить» и появится следующее окошко «Создание сценария».

В случае если у вас уже есть готовые сценарии, то их можно изменить. Здесь необходимо указать имя сценария. Называйте его так, чтобы можно было быстро определить, для чего он применяется, скажем, через несколько месяцев. Поле «Изменяемые ячейки» сообщает сценарию, откуда необходимо брать исходные данные, так что указывайте адреса ячеек, опираясь на собственные нужды. Они могут не быть смежными и тогда их адреса указываются через

запятую (не более 32 изменяемых ячеек на сценарий). Примечание самостоятельно заполняется, получая сведения об авторе и время создания сценария. Его можно дополнять по необходимости. После указания всех параметров сохраните изменения (кнопка «Ок»). В результате появится окошко «Значение ячеек», где будут отображены все внесенные изменения. После заполнения данной формы также сохраните изменения. Вы автоматически вернетесь к «Диспетчеру». Теперь осталось только воспользоваться готовым сценарием. Чтобы вывести результаты расчета сценария, необходимо указать его название в окне «Диспетчера» и активировать функцию «Вывести». Изменения вносятся в сценарий с помощью функции «Изменить». Также можно просмотреть отчет, активировав одноименную функцию. В новом окошке укажите его тип (это может быть сводная таблица или структурированный список) и ячейки результата (в них будут находиться формулы, результаты которых и нужно подвергнуть анализу).

Отчет представляет все изменяемые значения для любого сценария («Изменяемые») и значения формул, которые были вычислены при помощи этих значений («Результат»). Поскольку мы сначала присвоили имена для всех изменяемых ячеек и ячеек, в которых будет сохранен результат, то и при создании самого сценария идет вывод не адресов этих секторов, а их имена. Сам же отчет, в результате, выглядит максимально понятно. Это дает возможность проанализировать самое разное количество возможных вариантов. Разные пользователи могут хранить необходимые для анализа сведения в совершенно отдельных файлах-«книгах», но их можно собрать вместе и объединить в один сценарий, что еще больше ускорит работу. В результате можно легко получить необходимый отчет, опирающийся даже на информацию полученную таким путем. Визуальный анализ Такой анализ данных в Excel 2010 хорошо подходит для создания отчетов, что позволяет эффективно систематизировать, скажем, данные относительно совершенно любой деятельности за определенный временной отрезок. Допустим, у вас уже есть необходимые данные, но их необходимо подготовить для анализа.

Заходим на вкладку «Вставка» и приступаем к созданию сводной таблицы. Новый лист предложит макет стандартной такой таблицы. Все параметры, фигурировавшие в начальной таблице, будут перечислены справа. С помощью мыши перетаскиваем их в графу «Название строк». Делаем это с «Датами» и «Менеджерами», но у вас, скорее всего, будут иные наименования. В графу «Значения» помещаем «Объемы продаж», «Выручку» и «Прибыль». После этого таблица самостоятельно отформатируется и станет «лентой». Кстати, размещение элементов в «Названии строк» играет существенную роль. В случае, если «Менеджеры» выше «Дат», то все данные будут разбиваться соответственно имен этих работников. Если выше «Даты» - то соответственно календарных дат.

Теперь необходимо провести оформление созданной таблицы. Форматируем ее, как таблицу (вкладка «Главная» - «Форматировать как таблицу»). Появится

список самых разных шаблонов – необходимо выбрать тот, который более удобен именно для вас. После этого Excel самостоятельно определит границы, но их всегда можно отрегулировать и вручную.

Сохраняем параметры (кнопка «Ок») и смотрим на то, что получилось. Теперь уже можно сортировать параметры. Однако визуально просматривать значения и сравнивать с плановыми может оказаться весьма сложным делом. Особенно, если таких значений очень много. Допустим, что месячная выруская каждого нашего менеджера не должна быть меньше 100 000. Просматривать все самостоятельно не нужно – это займет слишком много времени и сил. Поэтому просто проводим условное форматирование (вкладка «Вставка» - «Условное форматирование» - «Набор значков») по понравившемуся шаблону. Например, «Светофор». Создаем правила форматирования (вводим показатели напротив значков), что позволит автоматически оценить работу сотрудника, как отличную, стабильную и неудовлетворительную. Показатели вводим напротив каждого кружочка в «Значение», в «Тип» устанавливаем «Числа», а не «Процент». Мы установили такой показатель, как 100 000 и 90 000, а третий выставиться автоматически, чтобы подключить оставшиеся значения. Сохраняем. Согласитесь, что теперь можно гораздо быстрее проанализировать данные и определить, кто из менеджеров хорошо справляется со своей задачей, а кого можно уже и увольнять.

Однако это только самые простые из возможностей современного Excel 2010. В нем появились дополнительные элементы, называемые «Цветовыми шкалами» и «Гистограммами». Давайте попробуем использовать именно их. Итак, выделяем значения в ячейках и форматируем их (вкладка «Вставка» - «Условное форматирование» - «Гистограммы»). Выпадающее меню демонстрирует список шаблонов (доступен предварительный осмотр при наведении курсора мыши на наименования).

Выбираем удобную цветовую схему. В результате получаем ячейки, залитые горизонтальными столбцами разной величины. Они в графическом плане отображают присутствующее в ячейке значение. Теперь можно уже даже скользнув взглядом по таблице понять, насколько плохо выполняет свои обязанности наш «Менеджер 5». Примечательно, если значение уйдет ниже минуса, то график сместится в противоположную от ячейки сторону, явно намекая на отрицательный результат. При использовании компонента «Цветовые шкалы» происходит заливка ячейки соответствующим цветом, который полностью соответствует результату. В результате наименьшее значение получит красный цвет, среднее – желтый, а высокое – зеленый.

Естественно, такую схему можно подобрать самостоятельно. Это более наглядный пример, чем применение «Набора значков», однако суть у них одна. Впрочем, есть еще один способ, который основан на применении срезов. Допустим, наши менеджеры проработали в компании уже не один год. Естественно, что дат станет гораздо больше и просмотреть такой документ, даже с применением форматирования, будет крайне сложно. Не говоря уже об

анализе данных. В этом случае добраться до интересующей нас даты можно одним из двух способов.

При построении любой сводной таблицы в правой части располагаются элементы, которые мы можем расположить в любые удобные поля. Если обратится к «Датам» и нажать на кнопочку с изображением стрелочки, то вызовется выпадающее меню. Остается только отфильтровать информацию по дате. Появляется большой список, предлагающий всевозможные варианты форматирования.

Воспользуемся помесячной сортировкой. Для этого необходимо открыть «Все даты за период» и выбрать нужный месяц. Для нас это «Октябрь». Это позволит значительно сократить нашу таблицу, оставив только интересующие нас значения.

Теперь давайте рассмотрим использование среза. Данный инструмент анализа великолепно подходит для цифровых данных. Открываем вкладку «Вставка» и ищем там «Срез». В результате открывается меню «Вставка среза», где необходимо отметить показатель, согласно которому и будет идти выборка интересующих нас значений. В данном случае это будет колонка «Даты», хотя можно выбрать любую из интересующих.

Подтверждаем и смотрим на страничку с рамкой и имеющимися значениями. Осталось только переместить ее в любое удобное место и отрегулировать размер, чтобы было видно все значения. При необходимости можно поменять цвет среза, чтобы сделать его более понятным (шаблоны находятся на верхней панели).

Теперь можно одним кликом выбрать необходимую дату и посмотреть на результаты работы наших менеджеров. Естественно, что благодаря своей гибкости использование среза более удобно, чем применение фильтра по дате. К тому же можно указать несколько значений для выборки.

В Excel 2010 можно применять инфокривые графики. Для этого необходимо выделить ячейку напротив строки с данными и сделать ее активной. Далее вновь используем вкладку «Вставка», раздел «Инфокривые» (он еще может носить название «Сперклайны»). Выделяем нашу строку, как диапазон данных и подтверждаем. В результате в выбранной ячейке появится небольшой график. Сделаем это же для результатов всех наших сотрудников. Теперь растянем каждую ячейку на другие строчки. Для этого можно просто потянуть за край, отмеченный точкой или же сделать на нем двойной клик. Как и при работе с иными полезными функциями, можно самостоятельно подбирать стиль инфокривой (верхняя панель, режим «конструктор»). Такой график хорошо отражает тренд и, если значений достаточно много, то визуально легко определяются взлеты и падения, а также начало возможного систематического падения или роста.

Сама инфокривая может быть одного из 3 типов: график, столбец – он отображает обрабатываемые данные маленькими столбиками. Чем больше данных, тем тоньше будут столбики, но они способны наглядно

продемонстрировать минимальное и максимальное значение; - «Выигрыш / Проигрыш» - ячейка условно делиться на две части.
При положительном результате квадратики помещаются в верхнюю часть, а при отрицательном – в нижнюю. «Ноль» в этом случае вообще не отображается.

Порядок выполнения работы

1. Изучите теоретический материал.
2. Составьте множество исходных данных для двух классов заболеваний по материалам Приложения к лабораторной работе №1 (классы выбираются по вариантам 1-2, 2-3, 1-2 согласно порядковому номеру в журнале.
3. Для каждого класса составьте выполните описательную статистику (программа STATISTICA или «Пакет анализа»). Сделайте выводы по отличительным особенностям для каждого класса.
4. Постройте гистограммы распределения по каждому признаку по разным классам на одном графике. Сделайте вывод о диагностических возможностях (специфичности) признаков.
5. В каждом классе постройте граф связности между признаками согласно парным коэффициентам корреляции (положительную корреляцию отобразите сплошной линией, отрицательную – пунктиром, значимость коэффициента корреляции (модульную величину) отобразите цветом). Сделайте выводы по результатам анализа различий между графами.
6. Оформите отчет, включающий результаты выполнения работы, ответы на контрольные вопросы, аннотацию каждого из приведенных в библиографии информационных источников (100-350 слов).

Контрольные вопросы

1. Что показывает гистограмма?
2. Что такое нормальное распределения случайной величины?
3. Какие диаграммы целесообразно использовать только для альтернативных вопросов?
4. Как называется разница между наибольшим и наименьшим значениями?
5. Какая статистика показывает, в какую сторону относительно среднего сдвинуто большинство значений распределения?
6. Кем была создана программа SPSS? Для чего она предназначена?
7. Альтернативный вопрос предполагает: А) выбор респондентом одного варианта ответа Б) выбор респондентом нескольких вариантов ответа В) предложение респондентом своего варианта ответа?
8. Какое действие позволяет управлять расположением данных внутри ячейки: А) выравнивание Б) шкалирование В) расширение?
9. Для чего нужен контроль данных: А) для исправления ошибок Б) для добавления новых переменных В) чтобы избежать повторений?
10. Какая процедура позволяет строить статистические ряды распределений: А) «частоты» Б) «статистики» В) «анализ» ?

11. Какой из типов переменной чаще всего используется как основание для деления объектов на группы: А) количественный Б) порядковый В) категориальный?
12. Для чего выполняют агрегирование данных?
13. Какой из типов столбчатых диаграмм используется для визуализации таблиц сопряженности: А) простые Б) составные В) кластерные ?
14. Что является первым этапом статистического анализа данных: А) описательный анализ Б) сравнительный анализ В) частотный анализ Г) факторный анализ?
15. Какие типы диаграмм целесообразно использовать для визуализации перекрестных таблиц: А) простые и кластерные Б) кластерные и состыкованные В) простые и состыкованные?
16. Приведите классическое определение разведочного анализа.

Информационные источники

1. Доказательная медицина //URL: http://encyclopatia.ru/
2. Доказательная медицина для всех//URL: http://medspecial.ru/
3. Доказательная медицина. Презентация //URL: http://www.kurskmed.com/uvr_docmed/uploads/9b3c40a.pdf
4. Пакет Анализа в Excel //URL: http://studopedia.su/12_127574_analiz-dannih-v-Excel.html
5. Разведочный анализ в STATISTIKA //URL: http://www.exponenta.ru/SOFT/STATIST/statistica5_5/1/1.asp
6. Разведочный анализ. Лекция //URL: http://www.slideshare.net/DEVTYPE/ss-53792094
7. Разведочный анализ в Excel //URL: http://advanceduser.ru/microsoft-excel/provodim-analiz-dannykh-v-excel-2010.html

22. Прогнозирование развития заболеваемости в регионе

Цель работы: Овладение навыками идентификации прогностических моделей уровня заболеваемости в регионе с учетом экологического фактора.

Краткие теоретические сведения.

Ухудшение экологической обстановки и социальной среды существенно отражается на состоянии здоровья человека. Здоровье человека и биосферы неразделимо связаны и определяются множеством компонент. Взаимодействуя с миром в ходе своей деятельности, человек ощущает на себе ответную реакцию окружающей среды.

К настоящему времени достаточно хорошо изучена зависимость здоровья от различных факторов. Кроме неизбежных природных явлений (таких как изменение солнечной активности), на здоровье человека могут влиять экологические факторы, вызываемые им самим в ходе своей деятельности. Указанное обусловливает необходимость управления деятельностью основных загрязнителей окружающей среды на основе математических методов (моделей), позволяющих достаточно адекватно оценить влияние различных факторов на рассматриваемый класс заболеваний с целью прогноза и (или) управления динамикой последнего.

Региональные экологические проблемы, сформировавшиеся в результате загрязнения окружающей среды из-за деятельности человека, требуют для своего решения использования региональных информационных систем (РИС). Одна из задач таких систем должна состоять в своевременном определении воздействия загрязняющих веществ на здоровье человека на основании анализа накопленной информации о состоянии окружающей среды и медико-биологической информации и прогнозирования уровня региональной заболеваемости. Исследование и прогноз выполняется на основе имитационного моделирования.

К настоящему времени собрано достаточное количество фактов о зависимости здоровья человека от различных факторов окружающей среды. Нарушения генома человека в основном связываются с такими показателями загрязнения биосферы, как повышение радиоактивности и химические загрязнения, что ведет к увеличению числа онкологических заболеваний, психических нарушений, патологии при беременности, деторождении. Психические расстройства могут возникать в зоне экологического бедствия даже при наличии хорошей социальной среды, богатых биологических свойств, психического и социального благополучия.

Для характеристики уровня экологической напряженности региона используют понятие экологической нормы, которое отражает определенные параметры сохранения приспособительных структур и функций экосистемы определенного иерархического уровня. Такое определение нормы может указывать на степень максимально допустимого воздействия человека и общества на окружающую

среду, которое обеспечивает функционирование и сохранение структуры и динамических качеств экосистемы в целом.

Поражение городского населения возможно главным образом через атмосферу, экологических природных систем - через все природные среды.

Таким образом, следует выделить два аспекта проблемы для изучения влияния антропогенной деятельности человека на заболеваемость:

а) воздействие на импактном уровне (на относительно небольшой территории);

б) массовое воздействие на природу, природные экосистемы на фоновом (как правило, невысоком) уровне, но на обширных территориях, практически по всей территории земного шара.

В концептуальном моделировании принято рассматривать три этапа:

- сбор и анализ априорной информации о предметной области и проблемной среде;
- концептуальный анализ предметной области с учетом требований пользователей;
- концептуальный синтез или собственно построение концептуальной модели предметной области.

Общая технология экологического управления в регионе состоит из трех этапов.

Целью первого этапа является получение информации о фактическом загрязнении сред региона. Учитывается как анализ источников антропогенного загрязнения региона, так и анализ естественных процессов, приводящих к фоновым концентрациям загрязняющих веществ в средах региона.

На втором этапе оценивается влияние состояния среды на заболеваемость населения.

На третьем этапе строится прогноз заболеваемости населения в зависимости от состояния среды и изменение самой среды, с последующей выдачей рекомендаций планирующим, природоохранным и хозяйственным органам.

Разработка пути возможного оздоровления и профилактика уровня заболеваемости в регионе в автоматизированной системе основывается на оценке влияния выбросов отдельных предприятий на ту или иную заболеваемость с последующей выдачей рекомендаций планирующим, природоохранным и хозяйственным органам о проведении мероприятий, призванных скорректировать выбросы соответствующих предприятий.Таблица связей, полученная при помощи автоматизированной системой моделирования, призвана ставить в соответствие предприятиям региона мероприятия по снижению выбросов.

В общем виде методика анализа вклада выбросов отдельного предприятия на уровень заболеваемости населения выглядит следующим образом:

1) Определяется список экологических факторов, обусловленных выбросами V_I -V_p в окружающую среду, влияющих на уровень заболеваемости по нозологии N_k.

2) Используя полученную математическую модель влияния факторов окружающей среды на уровень заболеваемости осуществляется прогноз о показателях заболеваемости в конкретном регионе.
3) На основании полученных показателях заболеваемости оценивается вклад каждого выброса в рост уровня заболеваемости населения по определенной нозологии в регионе.
4) Определяется вклад каждого выброса конкретного предприятия административного района в рост уровня заболеваемости путем нахождения доли выброса этого предприятия к общим выбросам по району.
5) Определяется общий вклад предприятия т в рост уровня заболеваемости по данной нозологии.
6) Определяется экономический ущерб U_k, вызванный влиянием деятельности предприятия на рост данной заболеваемости.
7) Шаги 1-6 повторяются для других нозологии, тем самым определяется общий экономический ущерб, вызванный влиянием деятельности предприятия на рост всей заболеваемости населения и определяется размер штрафа R_m для этого предприятия (т), эквивалентный сумме ущерба.
Далее из списка мероприятий для уменьшения показателей выбросов автоматически выбираются конкретные мероприятия, которые необходимо провести на данных предприятиях для уменьшения показателей выбросов, что в свою очередь должно привести к уменьшению заболеваемости системы кровоснабжения у населения.
Методика определения снижения значений выбросов загрязняющих веществ конкретными предприятиями состоит в следующем.
На первом этапе строится прогноз заболеваемости населения региона на следующий год при помощи автоматизированной системы математического моделирования.
Далее, определяется рост заболеваемости в процентном отношении для каждого района и для каждой нозологии.
На третьем этапе осуществляется коррекция значений выбросов предприятий региона с целью уменьшения уровня заболеваемости по следующему алгоритму. Задается порог роста заболеваемости hi, при котором считается целесообразным принятие мер. Также определяется шаг снижения выбросов предприятием h.
После определения заболеваемости и района, в котором рост заболеваемости ожидается выше порогового уровня, определяются выбросы, значимо влияющие на эту заболеваемость по заданному критерию. Затем определяются предприятия региона, вносящие наибольший вклад в сброс этих веществ, и моделируется снижение уровня выбросов на заболеваемость.
Далее прогнозные значения уровня заболеваемости пересчитываются с новыми значениями выбросов загрязняющих веществ и проверяется условие роста заболеваемости в этом районе. Операция повторяется до тех пор, пока рост заболеваемости не станет ниже hi.

К наиболее распространенным методам прогнозирования используемых в настоящее время относятся:

1. Регрессионные модели – строятся на базе сложившихся закономерностей развития событий с использованием специальных методов подбора вида экстраполирующей функции и определения значений её параметров [http://prognoz.org/lib/metody-prognozirovaniya];
2. Адаптивное сглаживание – метод предполагает постоянный пересмотр выбранных значений альфафактора. Коэффициент пересматривают по завершении каждого прогнозного периода и определяют то его значение, при котором прогноз был бы безошибочным [бизнес-учебники.рф/logist/metodyi-pronozirovaniya.html];
3. Факторный анализ – метод многомерного статического анализа, позволяющий на основе экспериментального наблюдения признаков объекта выделить группу переменных, определяющих корреляционную взаимосвязь между признаками [http://prognoz.org/lib/faktornyi-analiz];
4. Многомерная фильтрация;
5. Имитационные модели – метод, позволяющий строить модели, описывающие процессы так, как они проходили бы в действительности. Такую модель можно «проиграть» во времени как для одного испытания, так и заданного их множества. При этом результаты будут определяться случайным характерным процессом [http://ru.wikipedia.org/wiki/Имитационное_моделирование];
6. Метод группового учета аргументов – семейство индуктивных алгоритмов для математического моделирования мультипараметрических данных. Метод основан на рекурсивном селективном отборе моделей, на основе которых строятся более сложные модели. Точность моделирования на каждом следующем шаге рекурсии увеличивается за счет усложнения модели [http://ru.wikipedia.org/wiki/Метод_группового_учета_аргументов];
7. Экспоненциальное сглаживание тренда – метод математического преобразования используемый при прогнозировании временных рядов [http://ru.wikipedia.org/wiki/Экспоненциальное_сглаживание];
8. Спектральные методы – методы, основанные на изучении спектров излучения, поглощения и рассеивания [Крешков А.П. Основы аналитической химии. Теоретические основы. Количественный анализ];
9. Метод скользящей средней – метод дает возможность выравнивать динамический ряд на основе его средних характеристик;
10. Сплайн-функции – функции, область определения которых разбита на конечное число отрезков, на каждом их которых сплайн совпадает с некоторым алгебраическим полиномом [http://ru.wikipedia.org/wiki/Сплайн];
11. Оптимальные фильтры;
12. Метод Бокса-Дженкинса – метод прогнозирования на основе авторегрессионных моделей интегрированного скользящего среднего[http://bizness-gruppa.ru/box-jenkins-metod-boksa-dzhenkinsa];

13. Метод Марковских цепей – последовательность случайных событий с конечным или счётным числом исходов. Характеризующаяся тем свойством, что при фиксированном настоящем независимо от прошлого [http://ru.wikipedia.org/wiki/Цепь_Маркова];
14. Метод разностных уравнений - применяется в основном для анализа динамических характеристик импульсных стабилизаторов напряжения [Бердников Д.В. Применение метода разностных уравнений для синтеза ИСН];
15. Авторегрессионная модель – модель временных рядов, в которой значения временного ряда в данный момент линейно зависят от предыдущих значений этого же ряда [http://ru.wikipedia.org/wiki/Авторегрессионная_модель];
16. Вероятностный метод – метод, позволяющий с достаточной точностью определить, в каких пределах будет изменяться искомая величина, или с какой вероятностью можно ожидать какого-либо события [http://pictoris.ru/1/4/index.html].

Порядок выполнения работы.

1. Определить вариант задания по формуле Nвар=(4*mod(N))+1, где N - порядковый номер в группе. Согласно приложению и Nвар сформировать протокол мониторинга экологической ситуации города {XNвар, Y1, Y2, Y3}, где X - показатель уровня заболеваемости населения, Y - показатель загрязненности города определенным веществом.
2. Построить линейные и-или нелинейные регрессионные модели вида (отобрать лучшие по критерию детерменированности):

2.1. $X=f(Y_1)$; $X=f(Y_2)$; $X=f(Y_3)$; $X=f(Y_1,Y_2)$; $X=f(Y_1,Y_3)$; $X=f(Y_2,Y_3)$; $X=f(Y_1,Y_2,Y_3)$;

2.2 Повторить пункт 2.1, используя в качестве Y_i, i=1..3 значения Yij с нарастающим шагом, т.е.

$$Y'_{ij} = \sum_{k=1}^{j-1} Y_{ik}$$

3. На основе информационных источников изучить, что влияет на уменьшение Y_i и уменьшение влияния Y_i.
4. На основании анализа математических моделей, индентифицированных в пункте 2 и результатов «экспертного» анализа пункта 3, сформулировать предложения управляющего воздействия на факторы Y с целью улучшения показателя отклика X. Оцените доминантность влияния уровня определенного загрязнения на параметр здоровья. Сделайте вывод о структуре наиболее адекватной модели по критериям корреляции и СКО. Сравните частоты ритмической модели (если она адекватна) с внешними космогеологическими частотами (см. Приложение) и сделайте выводы. Ритмическую модель в данной работе предлагается получить путем анализа автокорреляционной функции или визуального анализа динамики заболеваемости во времени или визуального анализа поведения первой производной, вычисленной численными методами.

5. Оформите отчет, включающий в себя результаты работы (возможны скриншоты), выводы, краткие ответы на контрольные вопросы, аннотацию (300-350 слов информационных источников указанных в библиографии или иных).

Контрольные вопросы:

1. К какому классу систем можно отнести город?
2. С какими природными циклами наиболее коррелирует динамика определенных заболеваний?
3. Каким образом связаны между собой уровни заболеваемости населения и уровни антропогенного воздействия на окружающую среду (на примере уровней загрязнителей)?
4. Почему антропогенное воздействие следует учитывать с нарастающим эффектом?
5. Каким образом используются регрессионные и авторегрессионные математические модели для прогнозирования заболеваний?
6. Как осуществляется прогнозирование в Excel с помощью линии тренда?
7. Каким образом можно прогнозировать ритмические тенденции региональной заболеваемости?
8. Каким образом можно использовать логические функции (модели) для прогнозирования заболеваний?
9. Можно ли использовать искусственные нейронные сети для прогнозирования заболеваемости?
10. Каким образом можно использовать прогностические модели для удаления артефактов и восстановления пропущенных значений в мониторинге заболеваемости или состояния пациента в процессе терапевтического воздействия?

Приложения к практической работе

Таблица 1. - Основные природные ритмы

29.5 суток	Синодический месяц Луны	
206 суток	Сейсмическая активность Луны	
27.2 суток	Драконический месяц Луны	
8.85лет=3232 суток	Вращение линии апсид Луны	
78 месяцев	Геомагнитная активность	
26 месяцев	Стратосферная циркуляция + космические лучи	
2.5 - 4 года	Напряженность геомагнитного поля	
3.4 года	Гориз. составл. напряжен. геомагнитного поля	2.2, 3.5, 4.8, 6.1,
12.2 лет	Спектр скорости суточного вращения Земли	
26 месяцев	Изменение Ar на Солнце	
4.6 года	Метеополе	
11±4 (для 20 века 10±1)лет	Солнечная активность	
36 месяцев, 55 лет	М вариации числа Вольфа	

Таблица 2. Уровни показателей здоровья городского населения

Годы	Смертность	рождаемость	Прививки от дифтерии	Прививки от столбняка	Прививки от кори	Прививки от гепатита
1	13,2	14,99	10,2	24,6	8,9	15,8
2	11,9	13,36	11,0	30,5	7,7	34,3
3	12,8	11,4	9,5	29,3	9,6	24,9
4	13,5	12,23	10,3	27,7	10,6	28,5
5	13,4	12,92	11,3	43	11,7	18,9
6	13,0	11,67	11,2	44,7	11,5	18,2
7	13,7	11,55	12,4	47,5	10,7	13,0
8	13,0	13,3	12,6	44,1	11,0	19,1
9	14,2	14,7	11,7	44,4	11,9	24,2
10	14,6	14,1	12,1	46,6	18,2	28,2
11	13,0	14,6	13,2	39,5	16,6	12,3
12	13,3	14,8	13,0	40,0	16,3	19,51
13	13,6	13,8	13,5	38,7	17,4	10,12
14	13,6	11,8	14,5	40,9	26,8	19,91
15	14,6	10,8	13,8	34,2	25,1	12,54
16	14,7	9,7	11,6	28,3	12,1	4,2
17	16,5	9,0	11,0	31,1	11,0	85,7
18	18,0	9,3	7,3	16,6	5,6	72,2
19	16,7	8,6	11,5	38,7	3,7	2,02
20	17,0	8,3	11,1	52,5	11,4	22,6
21	16,5	7,9	10,4	27,6	8,9	19,7

Таблица 3. -Уровни загрязнения города (условные единицы)

годы	Пыль	Оксид углерода	Диоксид азота	фенол	формальдегид	марганец
1	6	1,5	2,667	8	3,5	0,8
2	4	0,667	2,222	8	2,75	0,16
3	6	1,45	1,333	6	2,25	1,7
4	2	1,067	4,444	6	2,75	1,7
5	4	1,833	0,889	6	3,25	2,5
6	4	1,833	0,889	4	3,25	1,1
7	6	1,667	1,333	6	3,75	2,0
8	6	1,667	1,333	10	3,0	0,7
9	8	2	1,156	8	2,75	0,6
10	4	1,667	0,889	7,4	2,0	1,2
11	4	1,667	1,333	8	3,0	3,5
12	2	1,667	1,33	8	3,75	1,4
13	2	1,667	1,778	10	3,5	3,2
14	2	1,667	1,333	6	3,25	1,4
15	2	1,663	2,222	4	3,0	0,2
16	2	1,667	1,333	4	3,0	0,6
17	4	1,667	1,333	2	2,5	0,4
18	5	2,333	1,333	8	2,25	0,5
19	1,8	2,5	1,333	4	3,0	0,6
20	1,8	2,67	1,3	4	3,25	0,1
21	2	2,5	1,778	6	3,75	0,3

Годы	Свинец	БПК5	Нефтепродукты (вода)	Азот аммон (вода)	Азот нитит (вода)	Медь
1	1,2	1,147	0,12	3,282	0,911	0,8
2	0,333	2,247	2,6	2,487	0,933	5,2
3	0,4	1,563	4,6	2,256	0,389	4,4
4	1,267	1,397	6,6	1,59	0,733	2,08
5	1,267	1,58	3,4	2,359	0,633	4
6	1,133	1,063	0,8	2,154	0,411	1,6
7	1,533	0,973	1,2	2,667	0,611	1,6
8	1,133	1,163	0,6	1,667	0,367	1,6
9	1,2	0,523	0,6	0,846	0,2	1,2
10	0,467	0,69	2,6	1,231	0,289	0,8
11	1,2	0,68	0,6	2,026	0,411	0,8
12	0,467	0,807	0,6	4,359	0,511	0,8
13	0,8	1,363	2,2	3,974	0,7	1,2
14	1,2	0,563	0,8	1,897	0,278	1,2
15	1,667	1,103	0,6	1,872	0,5	0,8
16	0,867	1,033	1,8	1,308	0,378	2,0
17	2,667	1,163	0,8	3,821	0,789	1,2
18	2,667	1,52	0,8	3,538	0,511	2,0
19	1,067	0,897	0,12	2,872	0,056	1,6
20	0,08	1,553	3	3,385	0,411	0,8
21	0,2	0,867	0,6	4,359	0,511	0,8

23. Алгоритмизация типовых вычислительных процессов

Цель работы: освоение навыками формализации и алгоритмизацией типовых вычислительных процессов, возникающих при обработке медицинских документов.

Краткие теоретические сведения.

Название "алгоритм" произошло от латинской формы имени величайшего среднеазиатского математика Мухаммеда ибн Муса ал-Хорезми (Alhorithmi), жившего в 783—850 гг. В своей книге "Об индийском счете" он изложил правила записи натуральных чисел с помощью арабских цифр и правила действий над ними "столбиком", знакомые теперь каждому. В XII веке эта книга была переведена на латынь и получила широкое распространение в Европе.

Человек ежедневно встречается с необходимостью следовать тем или иным правилам, выполнять различные инструкции и указания.

Для решения задачи надо знать, что дано, что следует получить и какие действия и в каком порядке следует для этого выполнить. Предписание, определяющее порядок выполнения действий над данными с целью получения искомых результатов, и есть алгоритм.

Алгоритм - заранее заданное понятное и точное предписание возможному исполнителю совершить определенную последовательность действий для получения решения задачи за конечное число шагов.

Исполнитель алгоритма — это некоторая абстрактная или реальная (техническая, биологическая или биотехническая) система, способная выполнить действия, предписываемые алгоритмом. Исполнителя характеризуют: среда, элементарные действия, система команд, отказы.

Среда (или обстановка) — это "место обитания" исполнителя.

Каждый исполнитель может выполнять команды только из некоторого строго заданного списка — системы команд исполнителя. Для каждой команды должны быть заданы условия применимости (в каких состояниях среды может быть выполнена команда) и описаны результаты выполнения команды.

Отказы исполнителя возникают, если команда вызывается при недопустимом для нее состоянии среды. Обычно исполнитель ничего не знает о цели алгоритма. Он выполняет все полученные
команды, не задавая вопросов "почему" и "зачем".

В медицинской информатике универсальным исполнителем алгоритмов является компьютер.

Основные свойства алгоритмов следующие:

1. Понятность для исполнителя — исполнитель алгоритма должен понимать, как его выполнять. Иными словами, имея алгоритм и произвольный вариант исходных данных, исполнитель должен знать, как надо действовать для выполнения этого алгоритма.

2. Дискретность (прерывность, раздельность) — алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов (этапов).
3. Определенность — 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.
4. Результативность (или конечность) состоит в том, что за конечное число шагов алгоритм либо должен приводить к решению задачи, либо после конечного числа шагов останавливаться из-за невозможности получить решение с выдачей соответствующего сообщения, либо неограниченно продолжаться в течение времени, отведенного для исполнения алгоритма, с выдачей промежуточных результатов.
5. Массовость 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.
Наиболее распространены следующие формы представления алгоритмов: словесная (запись на естественном языке); графическая (изображения из графических символов); псевдокоды (полуформализованные описания алгоритмов на условном
алгоритмическом языке, включающие в себя как элементы языка
программирования, так и фразы естественного языка,
Графический способ представления алгоритмов является более компактным и наглядным по сравнению со словесным.
При графическом представлении алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.
Такое графическое представление называется схемой алгоритма или блок-схемой. В блок-схеме каждому типу действий (вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий, окончанию обработки и т.п.) соответствует определенная геометрическая фигура, представленная в виде блочного символа.
Блочные символы соединяются линиями переходов, определяющими очередность выполнения действий. 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, который называется «процедурой» и-или «предопределенный процесс» обозначается как . 

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. Внутри блока записывается параметр цикла, для которого указываются его начальное значение, граничное условие и шаг изменения значения параметра для каждого повторения.
Алгоритмы можно представлять как некоторые структуры, состоящие из отдельных базовых (т.е. основных) элементов. Логическая структура любого алгоритма может быть представлена комбинацией трех базовых структур: следование, ветвление, цикл.
Характерной особенностью базовых структур является наличие в них одного входа и одного выхода.
1. *Базовая структура "следование".* Образуется последовательностью действий, следующих одно за другим:
действие 1
действие 2
. . . . . .
действие n
2. *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.
Примеры структуры ветвление записанной в виде текста:
если x > 0 то y := sin(x);
если a > b то a := 2*a; b := 1 иначе b := 2*b;
выбор
при n = 1: y := sin(x)
при n = 2: y := cos(x)
при n = 3: y := 0;

выбор
при a > 5: i := i+1
при a = 0: j := j+1
иначе i := 10; j:=0.
3. *Базовая структура "цикл".* Обеспечивает многократное выполнение некоторой совокупности действий, которая называется телом цикла. Основные разновидности циклов: с заранее известным количеством повторений, с предусловием, с постусловием.

Цикл типа пока (с предусловием). Предписывает выполнять тело цикла до тех пор, пока выполняется условие, записанное после слова пока.
Цикл типа для (с заранее известным количеством повторений). Предписывает выполнять тело цикла для всех значений некоторой переменной (параметра цикла) в заданном диапазоне.
Вычисление сумм — типичная циклическая задача.
Алгоритм, в состав которого входит итерационный цикл, называется итерационным алгоритмом. Итерационные алгоритмы используются при реализации итерационных численных методов.
В итерационных алгоритмах необходимо обеспечить обязательное достижение условия выхода из цикла (сходимость итерационного процесса). В противном случае произойдет "зацикливание" алгоритма, т.е. не будет выполняться основное свойство алгоритма — результативность.
Возможны случаи, когда внутри тела цикла необходимо повторять некоторую последовательность операторов, т. е. организовать внутренний цикл. Такая структура получила название цикла в цикле или вложенных циклов. Глубина вложения циклов (то есть количество вложенных друг в друга циклов) может быть различной. При использовании такой структуры для экономии машинного времени необходимо выносить из внутреннего цикла во внешний все операторы, которые не зависят от параметра внутреннего цикла.
Решение задач с помощью компьютера включает в себя следующие основные этапы, часть из которых осуществляется без участия компьютера.
1. Постановка задачи: сбор информации о задаче, формулировка условия задачи, определение конечных целей решения задачи, определение формы выдачи результатов, описание данных (их типов, диапазонов величин, структуры и т.п.).
2. Анализ и исследование задачи, модели: анализ существующих аналогов, анализ технических и программных средств, разработка математической модели, разработка структур данных.
3. Разработка алгоритма: выбор метода проектирования алгоритма, выбор формы записи алгоритма (блок-схемы, псевдокод и др.), выбор тестов и метода тестирования, проектирование алгоритма.
4. Программирование: выбор языка программирования, уточнение способов организации данных, запись алгоритма на выбранном языке программирования.
5. Тестирование и отладка: синтаксическая отладка, отладка семантики и логической структуры, тестовые расчеты и анализ результатов тестирования, совершенствование программы.
6. Анализ результатов решения задачи и уточнение в случае необходимости математической модели с повторным выполнением этапов 2 — 5.
7. Сопровождение программы: доработка программы для решения конкретных задач, составление документации к решенной задаче, к математической модели, к алгоритму, к программе, к набору тестов, к использованию.

Перед алгоритмизацией процесса осуществляется его формализация и составление математико-логической модели - системы соотношений, отражающих существенные свойства объекта или явления.
Выделив наиболее важные факторы, можно пренебречь менее существенными. Наиболее эффективно математическую модель можно реализовать на компьютере в виде алгоритмической модели — так называемого "вычислительного эксперимента". В общем случае результаты вычислительного эксперимента могут оказаться и несоответствующими действительности, если в модели не будут учтены какие-то важные стороны действительности.
Создавая математико-логическую модель для решения задачи, нужно:
1. выделить предположения, на которых будет основываться математическая модель;
2. определить, что считать исходными данными и результатами;
3. записать математические (и-или логические соотношения), связывающие результаты с исходными данными.

Порядок выполнения работы
1. Изучите теоретический материал.
2. Формализуйте и составьте блок-схемы для реализации следующих задач.

Задание 1. Составить алгоритм вычисления h по формуле $h=\frac{2}{a}\sqrt{p(p-a)}$, где $p=\frac{a+b+c}{2}$, a,b,c - стороны треугольника. Предварительно должна быть осуществлена проверка, характеризуют ли числа a,b,c прямоугольный треугольник.
Задание 2 Составить алгоритм, который определит наименьшее из двух чисел a и b и выведет на экран квадрат наибольшего числа. При этом, а изменяется от величины (a-sin(b)) до величины (a+cos(b)) с шагом $h=\begin{cases}\frac{a}{b+100} & a>b.b\neq -100\\ \frac{b}{a-100} & b>a.a\neq 100\end{cases}$.
Задание 3 В картотеке находится 20 больных. Составить алгоритм, который вычислит сумму элементов массива M[1..20] целых чисел (роста больных), имеющих четный индекс, если эта сумма больше 2500 и каждое слагаемое больше 120 см.
Задание 4 Составить алгоритм, который в массиве, состоящем из экзаменационных оценок, заменяет двойки на тройки, пересчитывает средний балл и определяет количество выполненных замен и выводит результат на экран.
Задание 5 Составить алгоритм, который определяет сколько раз температура пациента поднималась выше нуля. Информация о температуре задана в виде массива. Определить среднюю температуру и отклонение от нее ранее введенных показателей.

Задание 6 Составить алгоритм, который вычислит и выведет на экран сумму элементов массива M[1..10,1..10] целых чисел, имеющих четный индекс и лежащих под главной диагональю.
Задание 7 Составить алгоритм, который подсчитает количество фамилий больных, заканчивающий на «ий» и состоящий не менее чем из 5 букв.
Задание 8 Составить алгоритм, который определяет число слов в тексте, количество гласных букв в которых превышает заданное пользователем число.
Задание 9 Составить алгоритм, который заменяет все элементы массива M[1..12] действительных чисел на элементы с противоположным знаком, если порядковый номер минимального числа больше порядкового номера максимального.
Задание 10 Составить алгоритм, который определяет число вхождений в строку группы первых трех букв Вашей фамилии, если следующая буква гласная, и первых двух букв, если – согласная.

2. Составьте отчет, включающий: задания, формализация (математические или логические модели), блок-схемы алгоритмов, алгоритмы в текстовой реализации, пояснения к алгоритмам, ответы на контрольные вопросы (не менее 5).

Примечания:
1. Задания п.2 выбираются с учетом варианта, номер которого соответствует порядковому номеру в журнале – см. Таблицу

№ варианта	Задания к п.2
1	1,3,4,6,7,8,9,10
2	1,2,4,5,6,8,9,10
3	1,2,3,5,6,7,9,10
4	1,2,3,4,6,7,8,10
5	1,3,4,6,7,8,9,10
6	2,3,4,5,7,8,9,10
7	1,2,4,5,6,8,9,10
8	1,2,3,4,5,6,7,9
9	1,2,3,4,5,7,8,10
10	1,2,3,4,5,6,8,9
11	1,3,4,5,6,7,9,10
12	1,2,3,5,7,8,9,10
13	1,2,3,5,6,7,9,10
14	3,4,5,6,7,8,9,10
15	2,3,5,6,7,8,9,10

2. Выполнение практического занятия рассчитано на 8-10 академических часов, поэтому контроль ее выполнения рекомендуется осуществлять по мере выполнения различных этапов (решения задач).

Контрольные вопросы.

1. В какой последовательности целесообразно изучать моделирование и алгоритмизацию?
2. Приведите примеры моделей, которые создаются в различных отраслях знаний.
4. Почему для создания моделей используются алгоритмы и формальные языки?
5. Какой исполнитель алгоритма использовался С. Пейпертом для изучения алгоритмизации?
6. Приведите названия различных исполнителей алгоритмов, используемых для обучения.
7. Приведите перечень основных условий, которым должен удовлетворять учебный исполнитель алгоритмов.
8. Что называют архитектурой исполнителя алгоритмов?
9. Что такое ЛогоМиры и для чего они применяются?
10. Почему для описания алгоритмов используют блок-схемы?
11. Чем отличается алгоритмический язык от языка программирования?
12. Приведите пример вложенного цикла.
13. Приведите пример бесконечного цикла.
14. Как стандартизуется изображение элементов блок-схем алгоритмов?
15. Как обрабатываются матричные структуры?
16. Каким образом осуществляется принудительное окончание циклического процесса?
17. Чем отличаются циклы с пост- и с пред-условиями? Как они взаимозаменяются?
18. Как и когда осуществляется процедура безусловного перехода?
19. Что такое структурированный алгоритм?
20. Приведите условное обозначение основных элементов алгоритмов (не менее 7)?
21. Приведите алгоритм начала «Евгения Онегина»?
22. Какова роль алгоритма в дифференциальной диагностике заболеваний?

24. Операционная система Windows. Сервисные программы. Программы архиваторы

Цель: ознакомительное изучение операционной системы Windows, как базовой операционной системы применяемой в АРМ медицинских работников, и основным принципам работыс архиваторами.

Краткие теоретические сведения.

Операционная система - комплекс программ, обеспечивающий управление аппаратными средствами компьютера, организующий работу с файлами и выполнение прикладных программ, осуществляющий ввод и вывод данных.

Операционная система — это первый и основной набор программ, загружающийся в компьютер. Помимо вышеуказанных функций ОС может осуществлять и другие, например предоставление общего пользовательского интерфейса. В настоящее время наиболее известными операционными системами являются ОС семейства Microsoft Windows и UNIX-подобные системы.

Интерфейсные функции:

- Управление аппаратными средствами, устройствами ввода-вывода
- Файловая система
- Поддержка многозадачности (разделение использования памяти, времени выполнения)
- Ограничение доступа, многопользовательский режим работы (если взять к примеру ДОС, то он не может быть многопользовательским)
- Сеть (взять спектрум в пример…)
- Внутренние функции:
- Обработка прерываний
- Виртуальная память
- «Планировщик» задач
- Буферы ввода-вывода
- Обслуживание драйверов устройств

В общей вычислительной системе компьютера расположена «ось» между встроенным программным обеспечением компьютера (тут он, BIOS и все то, что заставляет работать самые простые команды: сложение, вычитание и сдвиг регистра) и программными приложениями пользователя (вот тут уже область пользователя: приложения, файлы и др.).

Чем же занимается операционная система? На самом деле работы у нее много и она всегда занята (потому уж не злитесь на нее сильно, когда она заставляет подождать несколько секунд, ведь дел у нее невпроворот). Заведует она вводом и выводом данных и раздает эти и задачи тем или иным устройствам, загружает программы в оперативную память и выполняет их, да и вообще управляет оперативной памятью, раздавая ее направо и налево разным запущенным вами приложениям в зависимости от того, как будет рациональнее использовать ее.

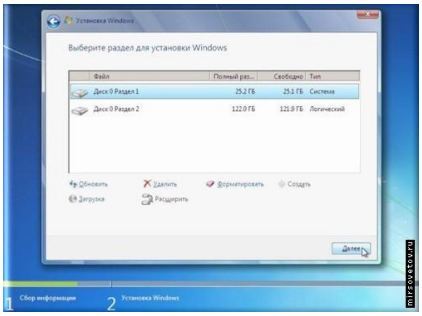
Операционная система так же управляет доступом ко всем источникам данных (съемным и оптическим дискам, flash-носителям и т.д.), защищает данные и саму себя (от взломщиков, вредоносных программ и пользовательских ошибок). Все та же ОС отвечает и за многозадачность вашего ПК, обеспечивает возможность работы на компьютере множества пользователей. Если вы думаете, что без операционной системы мы видели бы только черный экран и белые буквы с цифрами, ошибаетесь – мы не видели бы и этого, потому как для отображения букв уже нужна ОС.

Какие бывают ОС? На этот вопрос обычно отвечают так: операционные системы бывают Windows (сюда же и DOS относят, как правило), MacOS и различные его версии и Unix-подобные. На самом деле различие их намного сложнее и шире, однако же, это уже немного другая история (история курса «Операционных систем», преподаваемого будущим администраторам сетей и систем). Исследования показали, что на октябрь 2011 года Unix-подобные системы предпочли всего 0,84% пользователей, в то время как операционные системы от Apple (MacOS) используют 7,18%. Больше всего же пользователей различных версий Windows – 90,13% (источником статистических данных является сайт statcounter.com). Если сравнить данные результаты с аналогичными показателя за 2010 год, то можно заметить, что пользователи маленькими шагами переходят с Windows на другие системы.

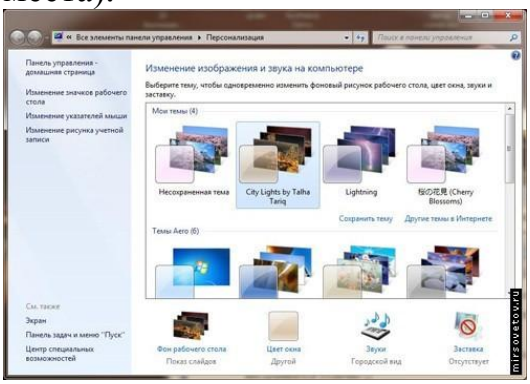
Операционная система Windows

Эволюция Windows. Началось все с DOS – с простого синего экрана и белых букв. Наверное, и до сих пор эта система осталась самой быстрой и надежной, ведь в ней было максимум связи с аппаратной составляющей и минимум визуальных эффектов. Но это было не слишком-то функционально, потому начали появляться различные версии: первая Windows 1.01, затем «усовершенствованная» Windows 2.03. На первых настольных ПК мы, скорее, вспомним Windows 95, потом Windows 98, с которым у меня, почему-то и до сих пор ассоциируется кличка «пенек» и анекдот «про старый пентиум». Сильно нашумел Windows Millenium, от которого многого ожидали, а затем Windows Vista, попавшая в то же положение. Windows 7 на сегодня считается достаточно стабильной и удобной для пользователя системой, и с этим сложно не согласиться.

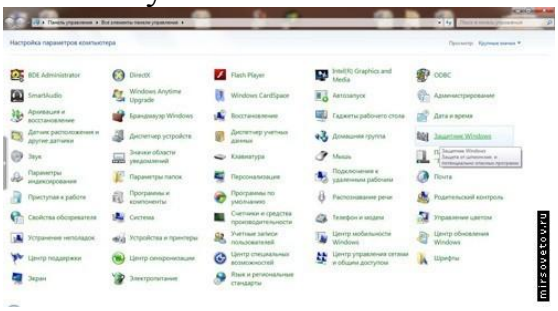
Установка Windows 7 не занимает много времени. При желании полная комплектация ОС располагается на компьютере за 12-15 минут, а при желании поменьше – за 25-30. Во время установки пользователь получает достаточно точные указания о происходящих процессах и его действиях, что важно для тех, кто свой компьютер обслуживает самостоятельно, не имея глубоких познаний в этой области.



Данная версия многое взяла от предыдущей – Windows Vista требовательность к аппаратным ресурсам. Если сравнивать с предыдущими версиями Windows, то сильнее стала производительность жесткого диска, несколько ослабла графика. Но ее отлично компенсирует далеко шагнувшая аппаратная часть. Считая же общий результат, Windows 7 сдала пару позиций в пользу Windows XP. Интерфейс «дружественный» и рабочий стол настраивается отлично (на рисунке ниже можем видеть окно персонализации виртуального рабочего места).



Вся система количество настроек значительно расширила, отчего более опытному пользователю – только плюс, а менее «подкованному» – не минус.



Настраивается теперь и звучание, и отображение, и многое другое. Операционная система обязательно посоветует пользователю, что нужно сделать, дабы увеличить производительность в той или иной сфере ее работы, повысить те или другие ее показатели или решить вдруг возникшую проблему с сетью или отладкой программы.

Стабильность и безопасность. Тщательному контролю подлежат все носители информации, программы от неизвестных разработчиков и файлы с несвойственной данному расширению активности. О каждом своем шаге, как и раньше, ОС докладывает пользователю.

Среди основных достоинств следует отметить:

- многоуровневость безопасности (от самого чувствительного, который будет бить тревогу чуть не при каждом запуске мало-мальски вредного ПО и до самого «спокойного», оповещающего лишь о том, что некая программа намеревается внести изменения в системе);
- наличие множества новых функций, например, функции «Библиотеки», призванной заменить устаревшие «Мои документы»;
- расширенность основного меню и «Панели инструментов».

Среди недостатков оказалась другая новая функция – «HomeGroups», которая предназначена для предоставления целых папок пользователям через сеть.

Linux Mint 11

Большинство пользователей Unix-подобных операционных систем – это просто какой-то «вражеский лагерь» для пользователей Windows. Первые не довольны Windows по причине ее «медлительности, кучи ошибок и вообще вечной сырости», вторые же считают Unix-системы «чем-то крайне непонятным и не пользовательским». Точки зрения обоих «противников» понятны, а вот о том ошибочны ли они – судите сами.

История Linux. Первые Unix были не достаточно адаптированы для рядового пользователя. Отличительной особенностью Unix-систем всегда было то, что они не монолитны, как Windows: их ядро состоит из множества самостоятельных модулей, которые могут работать независимо. Это позволяет «пересобрать» систему прямо в процессе работы с ней, что, конечно, нравится тем, кто хоть немного программирует. За счет своей атомарности такая «ось» лучше (читай стабильнее и быстрее) работает.

История самого Linux началась с попытки создания свободной операционной системы как говорят «с нуля» в рамках проекта GNU. Linux получила ядро от Unix и был снабжен множеством «примочек». Если первые версии этой ОС были консольными и «непонятными» для большинства пользователей, то последние уже вполне схожи с Windows по внешнему виду и набору функционала и даже позволяют работать с привычными для пользователей этой ОС приложениями.

Рассмотрения версию Linux Mint 11.

Установка Linux Mint практически полностью автоматизирована и потребует участия пользователя лишь для нескольких нажатий: указания языка, начального логин-пароля и настроек сети. Обо всем, что происходит во время установки, как и о том, что может или не может сделать пользователь, мастер установки уведомляет в «письменной форме».

Требования к аппаратной части, как и у всех Unix-систем, у Linux Mint минимальны. Этой ОС вполне хватит и 512 Мб оперативной памяти, интегрированного видеоадаптера и запаса памяти на жестком диске в 5 Гб. Также потребуется DVD или USB. 32-битная операционная система отлично будет работать как на 32-разрядном, так и на 64-разрядном процессоре.
Интерфейс дружественный. Окно настройки «внешнего вида» ОС представлено на рисунке.

Доступ ко всему богатству операционной системы осуществляется через преобразившееся меню, которое напоминает Windows 7.

В число уже установленных программ вошли:
- браузеры (в частности, Mozilla Firefox);
- аналог программного пакета MS Office, позволяющего сохранять документы в различных форматах (от txt и до docx);
- программы для прослушивания музыки, просмотра видео и изображений, а также для редактирования последних;
- программы для онлайн общения (Jabber) и т.д.

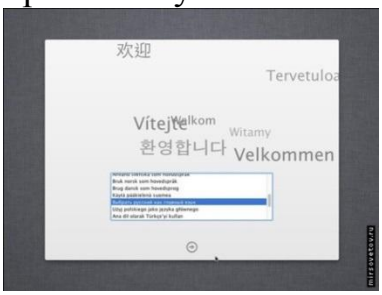
Безопасность Linux Mint – это, прежде всего, безопасность всех «не Windows» систем, ведь большинство вирусов написаны именно для Windows, а вот Linux и его собратья остаются в стороне. Но помимо этого, конечно, существует множество дополнительных средств, призванных охранять пользователя и верного его информационного друга. Работают для этого файерволы, настраиваются параметры сети и т.д.

Стабильность системы обеспечивается ее атомарностью. Дело в том, что ошибка, произошедшая в одном из модулей, на другие ну никак не повлияет. Потому ОС спокойно внесет в журнал событий данные об ошибке и ее причинах, перезагрузит этот модуль и станет работать дальше, а пользователь и вовсе просто ничего не заметит. Постоянных оповещений об ошибках вы так же наблюдать не сможете… Unix – это вообще очень молчаливая и уравновешенная система.

Достоинства следует отметить следующие:
- в сравнении с предыдущими версиями значительно улучшен интерфейс и настраиваемость;
- наличие минимального набора программного обеспечения позволяет установить систему и сразу же приступить к работе с ней;
- драйвера и кодеки, поставляемые с ОС, как правило, подходят для любого начального оборудования пользователя;
- исправлена несовместимость ОС со стандартом Wi-Fi;
Linux прекрасно станет рядом с любой другой операционной системой и даже не подумает о том, чтобы начать с ней конфликтовать, потому отказываться от привычной уже «оси» не стоит.

MacOS Lion 10.7.2

MacOS – это не просто операционная система, это стиль жизни. Дело в том, что MacOS изначально адаптирован под «свою» аппаратную часть и на привычных для нас IBM работает, но очень неважно.
История MacOS. Разработанная как надежная и производительная операционная система MacOS изначально удивляла всех тем, что работала она исключительно на компьютерах производства Apple. Первая MacOS появилась в 1984 году, но что-то более или менее похожее на ту красоту, которую мы привыкли видеть на «маках» изобрели только ближе в 2000 году. При этом первая версия именовалась очень просто: System 1.0, а вот приятные слуху «Lion», «Leopard» и т.д. – это уже достояние наших времен. В процессе своего развития MacOS прошла путь от полной нестабильности (да, были времена, когда ошибка работы 1 приложения всю систему просто выключала) до абсолютной надежности, многозадачности и всего того, о чем мы дальше и поговорим.
Установка MacOS Lion осуществляется легко и непринужденно в режиме диалога пользователя и его ПК. Свой уникальный стиль MacOS Lion также проявляет уже на этом этапе, как видно на рисунке.



Требования к аппаратной части великоваты, но оправданы. Указано жесткое требование касательно производителя «железа» – только Apple. ОС обязательно попросит вас поставить ее на процессор Intel i3 или и того новее, а еще напомнит о том, что стоит позаботиться о том, чтобы в запасе у нее оказалось: не менее 2 Гб оперативной памяти; 16 Гб запаса жесткого иска; достойной всех ее прелестей видеокарты (думаю, не менее 512 Мб, хотя это и не указывается в требованиях).

Производительность MacOS Lion напрямую зависит от того, какую аппаратную начинку вы дадите в ее распоряжение. Но, в любом случае, читателям МирСоветов не стоит беспокоиться о том, что ресурсы будут использоваться нерационально или же поддерживающий параллельность процессор вдруг окажется не у дел. В общем-то, производительность повысилась в сравнении с предыдущими версия, но не сильно, потому пользователь не заметит особой разницы. Рабочий стол организован наиболее красиво из остальных рассмотренных ОС.

Разрешена пользователю настройка всего, что «душеньке его угодно», для чего и реализованы «Системные настройки», не слишком масштабные, но очень функциональные: все настройки разбиты на категории и четко разграничены.

Операционная система поставляется с набором программного обеспечения, потому, установив ее, можно сразу же приступать к работе. Правда, вот еще некоторое время придется потратить на то, чтобы привыкнуть ко всей этой роскоши, скорости и функциональности, а главное – к тому, что «привычных и любимых» (для пользователей Windows) приложений тут нет и не будет, поскольку с этой ОС большинство из них просто несовместимы. Зато тут есть их аналоги, не уступающие им ни по удобству, ни по качеству (например, любимый браузер Mozilla легко заменит не менее функциональный Safari).

Безопасность MacOS Lion основывается на том, что эта ОС была разработана уже в эпоху сети интернет, а потому все возможные проблемы здесь обязательно предусмотрены. Файервол и антивирус тут тоже свой, и встроенный, разумеется. Ну и правило «MacOS – это «не Windows» система» тоже успешно работает.

Стабильность обеспечивается Unix FreeBSD ядром, полноценной многозадачностью системы, аппаратной стабильностью (не забываем о требовании аппаратной платформы от Apple) и многолетней практикой компании-производителя и разработчиков операционной системы. Достоинства MacOS Lion можно перечислять долго, отметим только те, что появились в этой версии и являются абсолютными нововведениями:

- управление визуальными компонентами стало проще и красивее благодаря системе жестов Multi-Touch;
- все окна могут быть полноэкранными, что значительно упрощает управление данными;
- многие программы, поставляемые вместе с ОС, претерпели качественные изменения, после чего стали удобнее, функциональнее и красивее;
- стоимость этой версии удивляет всех, кто знает цены от Apple.

Среди недостатков стоит отметить то, что не все качественные изменения приложений оказались к лучшему. Например, испортился Dashboard («виджет»), многие пользователи жалуются на то, что календарь для них стал неудобен.

В целом, система очень приятная: не утерян уникальный стиль MacOS, внесены отличные изменения, скорость работы впечатляет.

К наиболее популярным операционным системам относятся разработки компании Microsoft.

Выбор операционной системы – это наиболее важный выбор для любого пользователя. Нет такого понятия, как «лучшая операционная система», которое всем нравится. Apple Mac компьютеры уже давно считаются самыми простыми компьютерами в использовании, благодаря простоте операционной системы Mac. Таким образом, для пользователей ПК, выбор операционной системы, по существу происходит между Linux и Windows. Linux часто рассматривается как система, которую используется только вундеркинды. На самом деле, это просто не соответствует действительности. С каждым новым выпуском, как Windows, так и Linux, система получается более надежней, и обладает большими возможностями, и проще в использовании. И если вы полный новичок в использовании компьютера, то, возможно, Linux освоить так же просто, как и Windows. Таким образом, "лучшая операционная система" является личным выбором. Для компьютеров «до 2005» лучшая операционной системой являются: Windows XP (или более новые из семейства) или Linux, такая как Ubuntu . Эти системы дают лучшую производительность на «старом» оборудовании.

Рассмотрим *типовые операции в операционной системе*:

1) Корзина: Когда вы удаляете файл, он перемещается в корзину. С корзины можно будет восстановить файлы. А чтобы удалить файл полностью, вам надо удалить его из корзины.
2) Папки на рабочем столе: Вы можете хранить папки, файлы или ярлыки на рабочем столе.
3) Открытые папки: двойной щелчок открывает папку в проводнике.
4) Фон: Фон рабочего стола. Вы можете использовать свои фотографии, в качестве фона рабочего стола, или же можете выбрать один из встроенных изображений.
5) Кнопка Пуск: Нажмите кнопку Пуск, чтобы открыть меню, которая позволяет получить доступ к приложениям, файлам и настройкам. Также меню Пуск используется для того, чтобы вкл/выкл. компьютер.
6) Значки на панели задач: Некоторые программы будут иметь значки на панели задач для быстрого доступа.
7) Панель задач: Панель задач содержит меню Пуск, быстрый доступ, дату и время. Когда вы открываете программу или файл, он появиться на панели задач, и вы можете легко переключаться между открытыми программами.
8) Дата и время: В Правой стороне панели задач показывается текущая дата и время. Там также есть различные настройки, такие как громкость звука, настройки интернета и т.д.

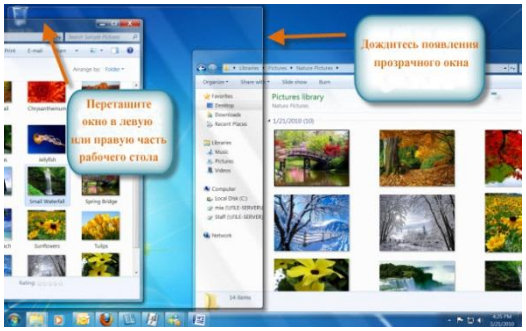
Функция Aero

Windows 7 использует группу функций под названием Windows Aero. Aero представляет собой визуальный рабочий стол, что сочетает в себе полупрозрачные окна, привлекательные цвета и графические эффекты с удобной функциональностью. Aero включает в себя Peek, Snake и Flip.

Snap

Функция Snap позволяет изменить размер открытых окон. Изображения ниже показывают два окна, до и после использования функции Snap.

Перетащите окно в левую или правую часть экрана до появления прозрачного окна, потом опустите окно.
Ваши окна должны встать на свое место.



Peek
Функция Peek позволяет просматривать открытые окна на панели задач в виде миниатюр. Просто наведите указатель мыши на иконки открытых программ на панели задач, появится миниатюры открытых окон.

Наведите указатель мыши на иконку открытого окна в панели задач, появится миниатюра окна. Теперь наведите курсор на миниатюру появится окно полностью.
Нажмите на миниатюру, чтобы открыть окно или нажмите на кнопку «X» чтобы закрыть окно.

Snake
Когда ваш рабочий стол завален открытыми окнами, вы можете использовать функцию Snake, чтобы оставить открытым одно окно, а все остальные свернуть.
Нажмите на верхнюю часть окна, удерживая окно, встряхните. Все остальные окна изчезнут, кроме той которую вы удерживали.
Заново встряхните окно, и все закрытые окна вновь появятся.

Flip

Функции Flip и Flip 3d — 2 способа просмотреть миниатюры всех открытых окон одновременно. При использовании Flip появиться миниатюры всех окон подряд, а Flip 3d отобразит все окна в виде стопки.

Удерживая клавишу Alt, нажмите на клавишу Tab. Не отпуская клавишу Alt, нажимайте на клавишу Tab, чтобы переключаться меду окнами. Остановитесь на том окне, которое требуется открыть и оно появится в полном экране.

Flip 3d (3d версия функции Flip)

Нажмите и удерживайте кнопку Windows (Пуск) на клавиатуре, а затем нажмите на клавишу Tab. Используйте клавишу Tab, чтобы пролистать все открытые окна. Вместо того, чтобы, удерживать нажатой клавишу Windows, вы можете нажать клавишу Ctrl + Windows и нажать Tab, чтобы Flip 3D оставался открытым.

Особенности Панели задач

Microsoft улучшила панель задач. Она стала очень удобной для просмотра и доступа к файлам и окнам.

Просмотр задач

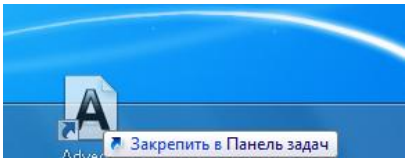
Когда вы открываете несколько окон одного типа, например, один и тот же браузер открыт несколько раз, то значки в панели задач будут выглядеть как на картинке.

Функция Aero Peek покажет вам все окна, при наведении указателя на значок на панели задач.

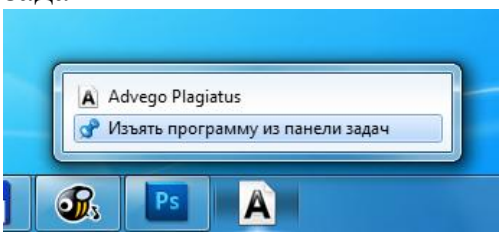
Если хотите, вы можете изменить порядок значков, простым перетаскиванием.

Закрепление программы на панели задач

Вы можете закрепить программы на панели задач. Это делается для того чтобы иметь к ним быстрый доступ. Чтобы закрепить программу, перетащите его на панель задач.



А чтобы убрать программу из быстрого доступа, просто щелкните правой кнопкой мыши на значок и выберите команду «Изъять программу из панели задач»



Дополнительные функции панели задач

1) Скрытые значки: Нажмите для просмотра дополнительных настроек, и значков.

2) Action Center: Просмотр важных уведомлений. На значке появиться красный крестик, если есть уведомления.

3) Значок доступа в Интернет.

4) Настройка громкости звука.

5) Кнопка «Свернуть все окна»

Свернуть все окна

Вы можете легко свернуть все окна одним нажатием.Наведите указатель мыши на кнопку «свернуть все окна», с лева от даты и времени.

Окна станут прозрачными и позволяет вам увидеть рабочий стол.

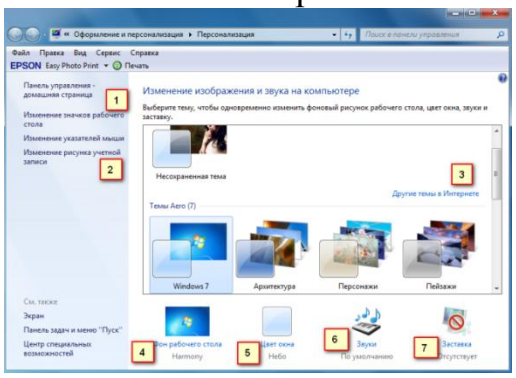
Нажмите на кнопку «Показать рабочий стол», чтобы свернуть все окна. Еще раз нажмите, чтобы развернуть все окна.

Персонализация фона и темы рабочего стола

В Windows 7 есть новые привлекательные темы и фоны на выбор. Они включают в себя яркие фотографии, цифровые изображения и Aero темы, с использованием цветов и стеклянных эффектов в привлекательном виде.

Темы и фон

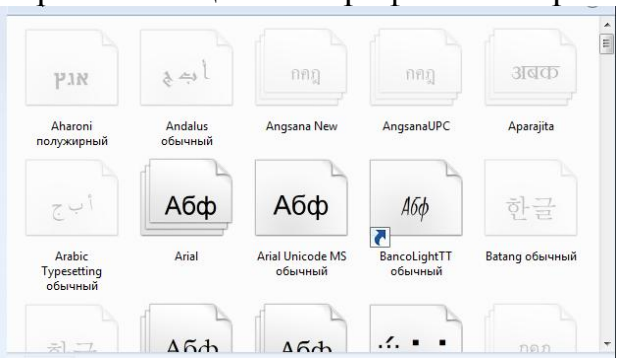
В строке поиска в меню Пуск наберите «персонализация». Или нажмите правой кнопкой мыши на рабочем столе и выберите Персонализация.



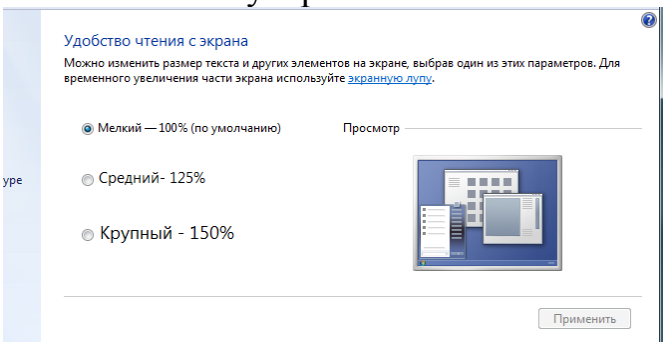
1) Вы можете изменить внешний вид значков вашего рабочего стола.
2) Вы можете изменить картинку вашей учетной записи.
3) Темы для рабочего стола
4) Вы можете поставить в качестве фона свое изображение, или выбрать одно из имеющихся в коллекции Windows. Вы также можете создать слайд-шоу, и ваш фон будет меняться время от времени автоматически.
5) Вы можете выбрать цвет для ваших окон и панели задач.
6) Если хотите, можете изменить звуки для Windows 7.
7) Windows 7 предлагает различные экранные заставки, которые можно выбрать и настроить.

Настройка шрифта и ClearType

Вы можете настроить параметры шрифта и ClearType на рабочем столе на основе ваших предпочтений.Чтобы изменить Шрифт: На панели поиска в меню Пуск наберите «Шрифты». Или Пуск -> Панель управления -> оформление и персонализация -> Шрифты. Выберите понравившийся Шрифт.



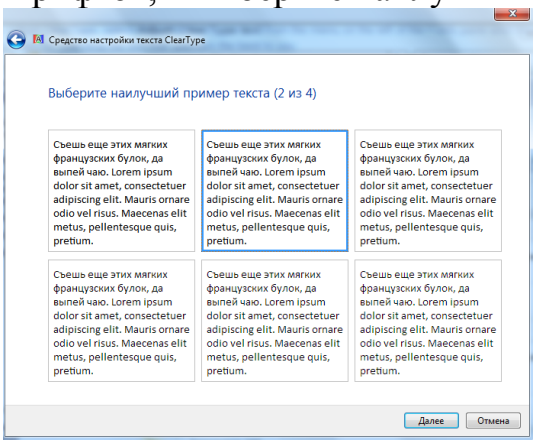
Чтобы изменить размер Шрифта: В левой части панели Шрифтов нажмите на надпись «Изменение размера Шрифта». Выберите нужный размер шрифта и нажмите кнопку Применить.



Обратите внимание, что больший размер шрифта может помешать отображению некоторых элементов на экране.

ClearType.

Вы также можете настроить ClearType для вашего экрана. ClearType помогает улучшить читаемость текста на ЖК-мониторах и экранах. Для настройки ClearType, выберите Настройка текста ClearType из меню, в левой части панели шрифтов, и выберите наилучший пример текста из предложенных.



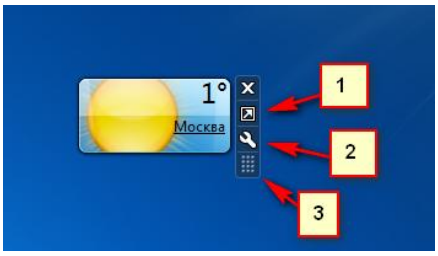
Гаджеты

Гаджеты — это мини-программы, которые устанавливаются на рабочий стол. Гаджеты могут показывать вам информацию о погоде, дате и времени, новостях

и многое другое. Чтобы добавить гаджеты на рабочий стол: Щелкните правой кнопкой мыши на рабочем столе и выберите гаджеты.

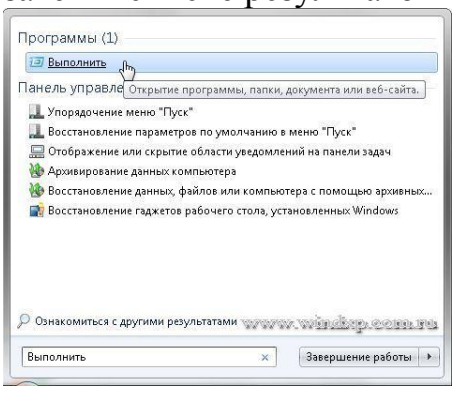
Выберите гаджеты и перетащите их в любое место на рабочем столе.

Настройка Гаджетов

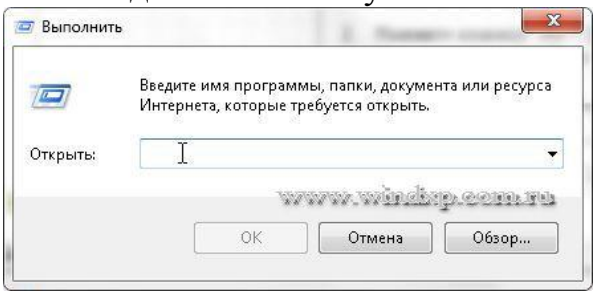


1) Нажмите на эту кнопку, чтобы изменить размер гаджета.
2) Кнопка опций вашего гаджета.
3) Эту кнопку используйте для перетаскивания.

Стандартные программы Windows 7. Запуск программ с помощью команды «Выполнить» . При установке операционной системы, по умолчанию ставится целый пакет стандартных программ Windows, о которых простому пользователю просто ничего не известно. Часть стандартных программ можно найти в менюПуск-Все программы. Дальше открываем разделы: Обслуживание, Стандартные, и далее Служебные, Специальные возможности. Часть программ предназначена для специалистов и доступ к этим программам можно осуществить только через командную строку или через команду «Выполнить».С помощью команды «Выполнить» можно быстро запускать программы, открывать файлы и папки, а также переходить на веб-сайты, если компьютер подключен к Интернету. Для запуска команды «Выполнить», можно применить один из способов. Нажмите кнопку Пуск. В поле поиска введите Выполнить, а затем в списке результатов щелкните Выполнить



Нажмите клавишуWin+R
И последний способ Пуск - Выполнить



Осталось ввести имя программы для ее запуска

К наиболее часто используемым программа относятся следующие.
- **appwiz.cpl** - Программы и компоненты. Программу можно удалить, если она больше не нужна или необходимо освободить место на жестком диске. Для удаления программ или изменения их конфигурации путем добавления и удаления определенных компонентов можно использовать компонент «Программы и компоненты».
- **control** - Панель управления. Используют для изменения параметров Windows. Эти параметры практически полностью определяют внешний вид и работу Windows и позволяют настроить Windows наиболее подходящим для пользователя образом
- **cipher.exe** - утилита для обслуживания EFS (Шифрованная файловая система (EFS) - это компонент Windows, позволяющий сохранять сведения на жестком диске в зашифрованном формате. Шифрование - это самая сильная защита, которую предоставляет Windows для защиты данных). Так же с помощью этой утилиты можно удалить файл, папку или данные с диска без возможности восстановления.
- **cleanmgr** - Очистка диска. Чтобы уменьшить количество неиспользуемых файлов на жестком диске для освобождения места на диске и повышения быстродействия компьютера. Она удаляет временные файлы, очищает корзину и удаляет множество системных файлов и других неиспользуемых элементов.
- **control folders** - Параметры папок. С помощью компонента «Параметры папок», можно изменить методы работы с файлами и папками, а также отображение элементов на компьютере.
- **control desktop** - Персонализация. Можно изменить отдельные части темы (изображения, цвета и звуки), а затем сохранить измененную тему для собственного использования или для совместного использования с другими пользователями.
- **comexp.msc** - Службы компонентов. С помощью оснастки «Службы компонентов» в консоли MMC можно настраивать компоненты COM, приложения COM+ и координатора распределенных транзакций DTC, а также администрировать их. Оснастка «Службы компонентов» предназначена как для системных администраторов, так и для разработчиков приложений. Например,

администраторы могут управлять компонентами, а разработчики могут настраивать требуемое поведение компонента и приложения, например участие в транзакциях и организации пула объектов

- **compmgmt.msc -** Управление компьютером. Консоль управления Microsoft (MMC) группирует средства администрирования, которые используются для администрирования сетей, компьютеров, служб и других системных компонентов.
- **credwiz -** Сохранение имен пользователей и паролей. Если имена пользователей и пароли будут повреждены или уничтожены, то можно будет использовать архивную копию для их восстановления.
- **charmap -** Таблица символов. Служит для вставки специальных символов в документ. Специальные символы - это символы, которых нет на клавиатуре. Эти символы включают сложные знаки математических операций, экспоненциальное представление чисел, символы валют и буквы других языков
- **certmgr.msc -** Сертификаты. В первую очередь сертификаты используются для идентификации пользователей или устройств, проверки подлинности служб или шифрования файлов. Обычно сертификаты используются незаметно для пользователя. Однако иногда появляются уведомления о том, что сертификат недействителен или срок его действия истек. В этих случаях следует выполнить инструкции, представленные в сообщении.
- **taskschd.msc -** Планировщик заданий. Если определенная программа используется регулярно, то при помощи мастера планировщика заданий можно создать задание, благодаря которому эта программа будет запускаться автоматически в соответствии с установленным расписанием. Для выполнения этих действий необходимо войти в систему в качестве администратора. В противном случае изменять можно только параметры текущей учетной записи пользователя.
- **devmgmt.msc -** Диспетчер устройств. С помощью диспетчера устройств можно устанавливать и обновлять драйвера аппаратных устройств, изменять параметры этих устройств и устранять неполадки в их работе. Драйвер устройства представляет собой программное обеспечение, с помощью которого Windows может взаимодействовать с отдельным устройством. Драйвер устройства устанавливается для обеспечения работы Windows с новым оборудованием.
- **diskpart -** Консольная утилита для управление разделами в томах жесткого диска. Параметр /add используется для создания нового раздела, а /delete — для удаления существующего. Переменная device – это название устройства для нового раздела (например, \device\harddisk0). Переменная drive – это буква, назначенная удаляемому разделу (например, D). Переменная partition – это соответствующее правилам именования разделов название удаляемого раздела (например, \device\harddisk0\partition1); она может использоваться вместо переменной drive. Переменная size служит для указания размера нового раздела (в мегабайта) Полный перечень команд утилиты можно получив, ведя HELP

- dxdiag - Пакет мультимедийных технологий DirectX используется многими играми ОС Windows. Если на компьютере не установлена требуемая версия DirectX, указанная на упаковке продукта, то игра может работать неправильно. Если при воспроизведении игры или фильма возникают какие-либо проблемы, средство диагностики DirectX поможет найти их причины. DirectX представляет собой набор технологий, используемых во многих мультимедийных программах ОС Windows

- diskmgmt.msc - Управление дисками. Средство управления дисками в этой версии Windows предназначено для выполнения таких задач управления дисками, как создание и форматирование разделов и томов и назначение букв дисков. Управление жестким диском отличается от управления программами и хранящейся на диске информацией. Некоторые операции управления приводят к изменению жесткого диска, например форматирование или перераспределение разделов. Под управлением информацией на жестком диске подразумевается упорядочение файлов и папок для повышения удобства доступа к информации (или настройка их свойств). Кроме того, для выполнения задач управления дисками вместе с другими программами командной строки можно применять команду DiskPart.

- eventvwr.msc - Просмотр событий. Программа «Просмотр событий» представляет собой оснастку консоли управления Microsoft (MMC) и предназначена для просмотра и управления журналами событий. Это незаменимый инструмент для наблюдения за работоспособностью системы и устранения возникших неполадок (например, ненадлежащий запуск программ или обновлений, загружаемых автоматически).

- eudcedit - Редактор личных знаков. Редактор личных символов позволяет создавать собственные символы и вставлять их в документы с помощью таблицы символов

- ftp - FTP-протокол. File Transfer Protocol (FTP) - это протокол, используемый для передачи файлов через Интернет. Обычно FTP используется, чтобы сделать файлы доступными для загрузки другими пользователями, но может быть использован и при отправке веб-страниц для формирования веб-сайта или для размещения цифровых фотографий на сайте с общим доступом к изображениям.

- firewall.cpl - Брандмауэр Windows. Брандмауэр представляет собой программный или аппаратный комплекс, который проверяет данные, входящие через Интернет или сеть, и, в зависимости от настроек брандмауэра, блокирует их или позволяет им пройти в компьютер. Брандмауэр поможет предотвратить проникновение хакеров или вредоносного программного обеспечения (такого как черви) в ваш компьютер через сеть или Интернет. Брандмауэр также помогает предотвратить отправку вредоносных программ на другие компьютеры.

- iexplore - Internet Explorer. Браузер компании Microsoft

- intl.cpl - Язык и региональные стандарты. Можно изменить формат, используемый в Windows для отображения информации (такой как даты, время,

валюта и единицы измерения), чтобы он соответствовал стандартам выбранного языка. Например, при работе с документами, написанными на двух языках (французском и английском), можно поменять формат на французский и использовать в качестве валюты евро, а даты отображать в формате день/месяц/год.

- **mmc** - Консоль управления (MMC). Место для хранения и отображения средств администрирования, созданных корпорацией Майкрософт и другими поставщиками программного обеспечения. Эти средства называются оснастками и служат для управления оборудованием, программным обеспечением и сетевыми компонентами ОС Windows. Некоторые средства, расположенные в папке «Администрирование» панели управления, например «Управление компьютером», являются оснастками консоли MMC.
- **msconfig** - Конфигурация системы. Программа настройки системы - это дополнительное средство, предназначенное для определения проблем, которые могут помешать запуску ОС Windows в обычном режиме. При запуске Windows можно отключить обычные службы и автоматически загружаемые программы, а затем включать их по одной. Если проблема не возникает, когда служба отключена, но появляется после ее включения, значит эта служба может быть источником проблемы. Программа настройки системы предназначена для поиска и устранения неполадок, но не для управления загрузкой.
- **msinfo32** - Сведения о системе. Компонент «Сведения о системе» (также называемый msinfo32.exe) отображает подробные сведения о конфигурации оборудования, компонентах и программном обеспечении компьютера, включая драйверы
- **msra** - Удаленный помощник. Иногда для устранения неполадки наиболее удобно, чтобы кто-нибудь показал, как это делается. Удаленный помощник Windows - это удобный способ для кого-либо, заслуживающего доверия, например друга или специалиста службы технической поддержки, подключиться к компьютеру пользователя и помочь ему найти решение проблемы, даже если этого специалиста нет рядом
- **msdt** - Средство диагностики технической поддержки Майкрософт. Используется для сбора сведений о неполадках, возникающих при работе компьютера, и последующей отправки этих сведений через Интернет в службу технической поддержки Майкрософт.
- **mmsys.cpl** - Звук. Можно задать воспроизведение звуков компьютером при возникновении определенных событий (событием может быть как действие, выполняемое пользователем, например вход в компьютер, так и действие, выполняемое компьютером, например оповещение о получении нового сообщения электронной почты)
- **odbcad32** - Администратор источников данных ODBC. ODBC - это технология, которая используется программами для получения доступа к различным базам данных (или источникам данных). Например, технологию ODBC можно использовать для импорта данных из базы данных MySQL в

электронную таблицу Microsoft Excel. Для этого необходимо, чтобы на компьютере был установлен требуемый драйвер ODBC и задан источник данных.
OptionalFeatures- Компоненты Windows. Некоторые программы и компоненты в составе ОС Windows, например службы IIS, перед использованием необходимо включить. Некоторые другие функции включены по умолчанию, но их можно выключить, если они не используются
- osk - Экранная клавиатура. Вместо обычной клавиатуры для печати и ввода данных можно использовать экранную клавиатуру. Экранная клавиатура отображается на экране со всеми стандартными клавишами. Можно выбирать клавиши с помощью мыши или другого указывающего устройства, либо использовать единственную клавишу или группу клавиш для переключения между клавишами на экране
- odbcad32 - Администратор источников данных ODBC. ODBC - это технология, которая используется программами для получения доступа к различным базам данных (или источникам данных). Например, технологию ODBC можно использовать для импорта данных из базы данных MySQL в электронную таблицу Microsoft Excel. Для этого необходимо, чтобы на компьютере был установлен требуемый драйвер ODBC и задан источник данных.
- perfmon - Системный монитор Windows. Можно использовать для анализа влияния работы программ на производительность компьютера как в реальном времени, так и посредством сбора данных журнала для последующей обработки. Системный монитор Windows использует счетчики производительности, данные трассировки событий и сведения о конфигурации, которые можно объединять в группы сборщиков данных
- psr - Средство записи действий по воспроизведению неполадок. Средство записи действий по воспроизведению неполадок можно использовать для записи действий, выполняемых на компьютере, включая текстовое описание мест выполняемых щелчков мышью и изображений экрана для каждого щелчка (называемых снимками экрана). Записанные действия можно сохранить в файл, который может использовать специалист службы поддержки или другое лицо, помогающее устранять проблему на компьютере
- powercfg.cpl - Электропитание. Схема управления питанием - это набор аппаратных и системных параметров, управляющих потреблением и экономией питания компьютером. Схемы управления питанием позволяют сэкономить энергию, максимально увеличить быстродействие системы или обеспечить оптимальное соотношение между ними
- rstrui - Восстановление системы. Позволяет отменить изменения, внесенные в систему компьютера, не затрагивая личные файлы, например электронную почту, документы или фотографии. Восстановление системы - это оптимальный выбор при установке программы или драйвера, которые вызвали неожиданное

изменение конфигурации компьютера или ОС Windows, а удаление программы или драйвера не решило проблему

- **regedit** - Редактор реестра. Инструмент, предназначенный для опытных пользователей. Этот инструмент предназначен для просмотра и изменения параметров в системном реестре, в котором содержатся сведения о работе компьютера

- **recdisc** - Создание диска восстановления системы. Параметры восстановления системы помогут восстановить Windows в случае серьезной ошибки. Для использования параметров восстановления системы необходим установочный диск Windows или доступ к параметрам восстановления, предоставленным изготовителем компьютера. Если ни то, ни другое получить не удается, для доступа к параметрам восстановления системы можно создать диск восстановления системы

- **gpedit.msc** - Редактор локальной групповой политики. Редактор локальной групповой политики - это оснастка консоли управления (MMC), которая обеспечивает единый интерфейс управления всеми параметрами объектов локальной групповой политики.

- **sdclt** - Архивация и восстановление. Программа архивации Windows позволяет создать образ системы, который представляет собой точный образ диска. Образ системы также содержит Windows и системные параметры, программы и файлы. Восстановление системы позволяет восстановить состояние системных файлов компьютера на предшествующий момент времени

- **secpol.msc** - Локальная политика безопасности. Используется для просмотра и изменения параметров безопасности групповой политики

- **sfc** - Проверка целостности всех защищенных системных файлов и замена неправильных версий правильными. Запускается в командной строке с правами администратора.

- **sigverif** - Проверка подписи файла. Цифровая подпись представляет собой добавляемую в файлы электронную метку безопасности. Она позволяет проверить издателя файла и помогает определить, был ли изменен файл после добавления к нему цифровой подписи.

- **taskmgr** - Диспетчер задач Windows. Диспетчер задач отображает приложения, процессы и службы, которые в текущий момент запущены на компьютере. С его помощью можно контролировать производительность компьютера или завершать работу приложений, которые не отвечают

- **TabletPC.cpl** - Перо и сенсорные устройства. При работе с планшетным ПК или сенсорным экраном можно выполнять планшетным пером или пальцем движения, называемые жестами, для быстрой навигации и выполнения действий

- **verifier** - Диспетчер проверки драйверов. Драйвер - это программа, обеспечивающая взаимодействие компьютера с оборудованием и устройствами. Без драйверов невозможна нормальная работа подключенного к ПК оборудования, например видеоадаптера или принтера

- **lusrmgr.msc** - Локальные пользователи и группы. Оснастка "Локальные пользователи и группы" служит для создания пользователей и групп, хранимых локально на компьютере, и управления ими
- **wscui.cpl** - Центр поддержки. В Центре поддержки перечислены важные сообщения о параметрах безопасности и обслуживания компьютера, которые требуют внимания пользователя. Красным цветом помечены Важные сообщения, свидетельствующие о значительных проблемах, которые необходимо устранить как можно быстрее
- **fxscover** - Редактор титульных страниц факсов. Компонент «Факсы и сканирование Windows», включенный в эту версию Windows, содержит четыре готовые титульные страницы. В этой программе можно также создавать собственные титульные страницы

Если какая то стандартная программы не работает или работает со сбоями, то можно ее попробовать переустановить. Откройте Компоненты Windows, введя команду *Optional Features* в Пуск - Выполнить. Выберите компонент, который необходимо переустановить, снимите с него галку (При отключении некоторых компонентов появляется предупреждение) жмем ДА и после перезагрузки компьютера, снова отмечаем галкой тот компонент, который отключали. Чтобы изменения вступи в силу, придется еще раз перезагрузиться.

Дополнительные сведения на веб-сайте Windows. Посетите веб-сайт Windows, который содержит дополнительные сведения, материалы для загрузки и идеи по максимально эффективному использованию компьютера под управлением Wind.

АРХИВАТОРЫ

Архиватор — это программа, осуществляющая упаковку одного и более файлов в архив или серию архивов для удобства переноса или хранения, а также распаковку архивов. Большинство современных архиваторов также реализуют сжатие упакованных в архив данных.

Простейшие архиваторы просто последовательно объединяют (упаковывают) содержимое файлов в архив. Архив должен также содержать информацию об именах и длине оригинальных файлов для их восстановления, поэтому большинство архиваторов также сохраняют метаданные файлов, предоставляемые операционной системой, такие, как время создания и права доступа. Такую функциональность реализует tar — стандартный архиватор систем типа UNIX. При необходимости уменьшения размера к tar-архиву применяют сжатие без потерь программами gzip, bzip2 и т. д. Большинство других современных архиваторов содержат сжатие, как встроенную функцию по умолчанию.

Характеристики архиваторов: по степени сжатия и по скорости сжатия. Эти характеристики — обратно зависимые величины. То есть, чем больше скорость сжатия, тем меньше степень сжатия, и наоборот. Программа, создавая архив, обрабатывает как текстовые файлы, так и бинарные файлы. Первые всегда сжимаются в несколько раз (в зависимости от используемого алгоритма).

Сжатие бинарных файлов зависит от их формата. Одни бинарные файлы могут быть сжаты в десятки раз, сжатие же других может и вовсе не уменьшить занимаемый ими объём.

Нахождение для любого входного файла алгоритма с наименьшим возможным размером на выходе, является алгоритмически неразрешимой задачей.

Сжатие данных обычно происходит значительно медленнее, чем обратная операция.

Различными разработчиками были созданы специальные программы для архивации файлов. Как правило, программы для архивации файлов позволяют помещать копии файлов на диске в сжатом виде в архивный файл, извлекать файлы из архива, просматривать оглавление архива и т.д. Разные программы отличаются форматом архивных файлов, скоростью работы, степенью сжатия файлов при помещении в архив, удобством использования.

В настоящее время применяется несколько десятков программ - архиваторов, которые отличаются перечнем функций и параметрами работы, однако лучшие из них имеют примерно одинаковые характеристики. Из числа наиболее популярных программ можно выделить: *ARJ, PKPAK, LHA, ICE, HYPER, ZIP, PAK, ZOO, EXPAND,* разработанные за рубежом, а также *AIN* и *RAR*, разработанные в России. Обычно упаковка и распаковка файлов выполняются одной и той же программой, но в некоторых случаях это осуществляется разными программами, например, программа *PKZIP* производит упаковку файлов, а *PKUNZIP* - распаковку файлов.

Программы-архиваторы позволяют создавать и такие архивы, для извлечения из которых содержащихся в них файлов не требуются какие - либо программы, так как сами архивные файлы могут содержать программу распаковки. Такие архивные файлы называются самораспаковывающимися.

Самораспаковывающийся архивный файл - это загрузочный, исполняемый модуль, который способен к самостоятельной разархивации находящихся в нем файлов без использования программы - архиватора. Самораспаковывающийся архив получил название SFX - архив (SelF - eXtracting). Архивы такого типа в MS DOS обычно создаются в форме .EXE - файла.

Многие программы - архиваторы производят распаковку файлов, выгружая их на диск, но имеются и такие, которые предназначены для создания упакованного исполняемого модуля (программы). В результате такой упаковки создается программный файл с теми же именем и расширением, который при загрузке в оперативную память самораспаковывается и сразу запускается. Вместе с тем возможно и обратное преобразование программного файла в распакованный формат. К числу таких архиваторов относятся программы *PKLITE, LZEXE, UNP*.

Программа *EXPAND*, входящая в состав утилит операционной системы MS DOS и оболочки Windows, применяется для распаковки файлов программных продуктов, поставляемых фирмой Microsoft.

Программы - архиваторы *RAR* и *AIN,* кроме обычного режима сжатия, имеют режим **solid**, в котором создаются архивы с повышенной степенью сжатия и особой структурой организации. В таких архивах все файлы сжимаются как один поток данных, т.е. областью поиска повторяющихся последовательностей символов является вся совокупность файлов, загруженных в архив, и поэтому распаковка каждого файла, если он не первый, связана с обработкой других. Архивы такого типа предпочтительнее использовать для архивирования большого числа однотипных файлов. Управление программой - архиватором осуществляется одним из двух способов:

1) с помощью командной строки MS DOS, в которой формируется команда запуска, содержащая имя программы - архиватора, команду управления и ключи ее настройки, а также имена архивного и исходного файлов; подобное управление характерно для архиваторов ARJ, AIN, ZIP, PAK, LHA и др.;
2) с помощью встроенной оболочки и диалоговых панелей, появляющихся после запуска программы и позволяющих вести управление с использованием меню и функциональных клавиш, что создает для пользователя более комфортные условия работы.

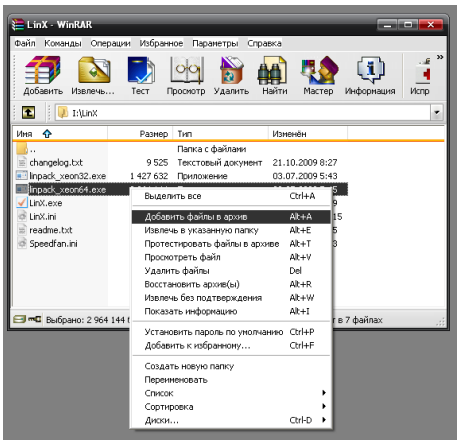
Такое управление имеет программа - архиватор RAR.

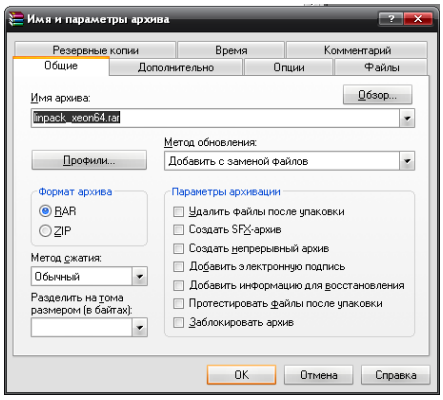
Создание архива из одного документа с помощью WinRAR.

Программа WinRAR работает под управлением ОС Windows и предоставляет пользователю графический интерфейс. Допускает создание архива как из одного, так и из нескольких файлов. При сжатии используется построение дерева Хаффмана. Последние версии программы реализуют алгоритмы многопоточного сжатия и задействуют все вычислительные ядра центрального процессора.

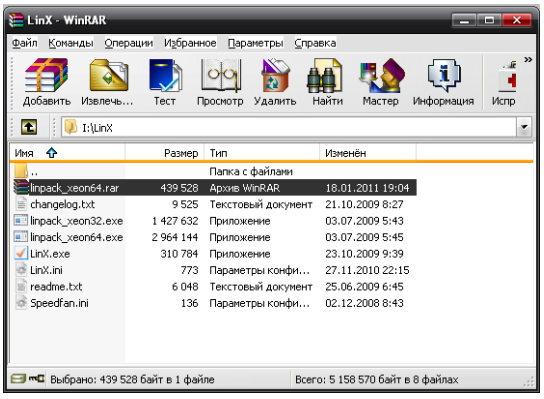
Для создания архива из одного документа средствами WinRAR достаточно выполнить следующие действия:

1. Запустить программу WinRAR.
2. В адресной строке программы WinRAR ввести полный путь к каталогу, содержащему архивируемый файл.
3. Щелчком правой клавиши мыши на имени архивируемого файла вызвать контекстное меню (см. рисунок).
4. При необходимости внести изменения в поля диалогового окна «Имя и параметры архива».
5. Нажать кнопку «Ok» диалогового окна «Имя и параметры архива».





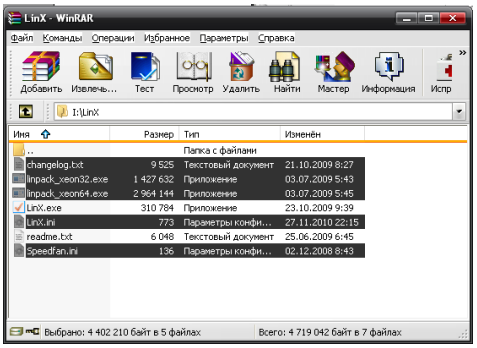
В результате выполнения указанных действий будет создан архив, содержащий единственный файл. Если в поля диалогового окна «Имя и параметры архива» не вносились изменения, то имя полученного архива будет совпадать с именем архивированного файла, архив имеет расширение rar.



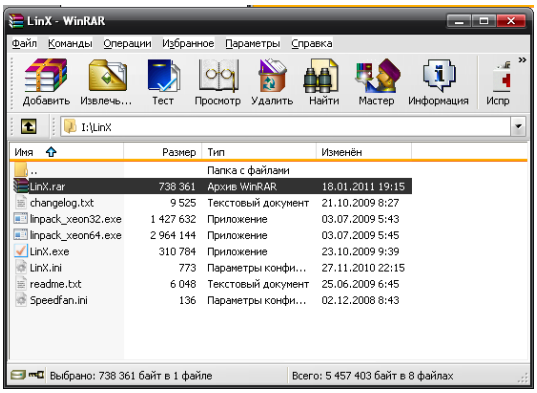
3. Создание архива из нескольких документов

Для создания архива из нескольких документов выполняются пп. 1 и 2 . После этого в окне программы WinRAR выделяются подлежащие

архивированию файлы. Непрерывный диапазон файлов можно выделить, удерживая клавишу Shift, отдельные файлы – удерживая клавишу Ctrl:



Последующие действия аналогичны пп. 3…5 раздела 1.1. В результате будет создан архив, содержащий несколько файлов. Если в поля диалогового окна «Имя и параметры архива» не вносились изменения, то имя полученного архива будет совпадать с именем каталога, в котором находятся архивируемые файлы.



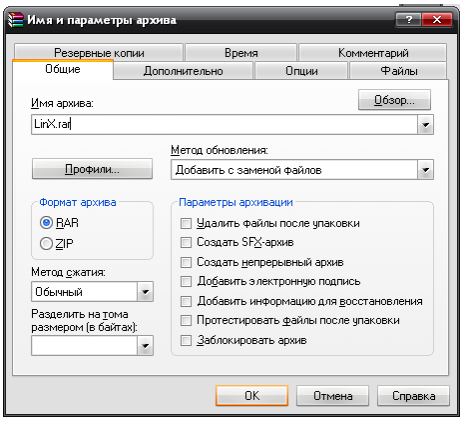
3. Добавление файла в архив

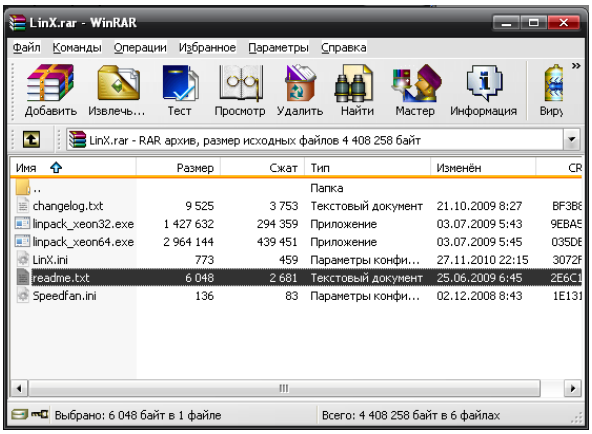
При установке программы WinRAR она интегрируется с оболочкой Windows (программой «Проводник»). Поэтому для добавления одного файла в архив достаточно в окне программы Проводник вызвать на добавляемом файле контекстное меню и выбрать в нём пункт «Добавить в архив…»

При этом появится диалоговое окно «Имя и параметры архива» программы WinRAR. В поле «Имя архива» вкладки «Общие» необходимо указать путь (полный или относительно текущего каталога) к архиву, в который добавляется файл и нажать на кнопку «Ok».

В результате в архив будет добавлен один файл.

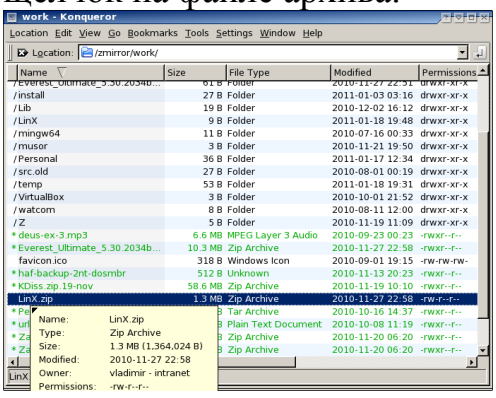
Для добавления нескольких файлов в архив достаточно выделить их в окне программы «Проводник» (способ выделения указан выше).



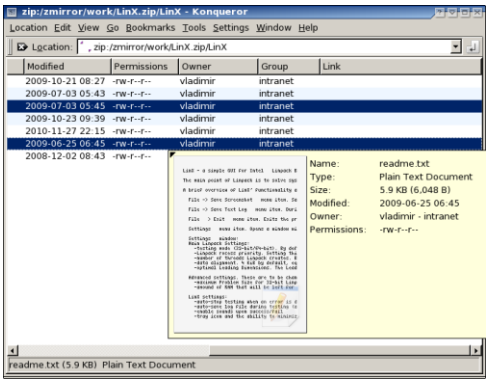


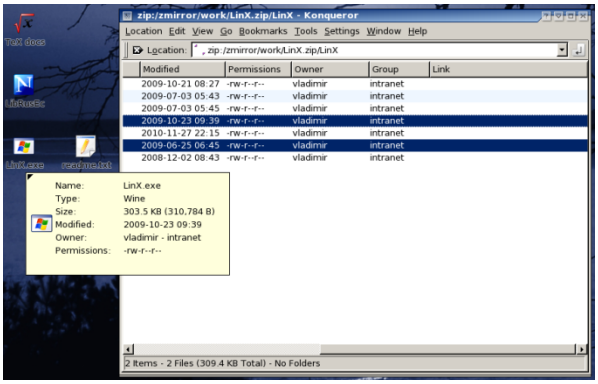
4.Извлечение документов из архива

Если программа-архиватор (WinRAR, WinZip или любой другой для любой рабочей среды любой ОС) интегрирована с рабочей средой, то для извлечения одного или нескольких файлов достаточно в окне проводника сделать двойной щелчок на файле архива.

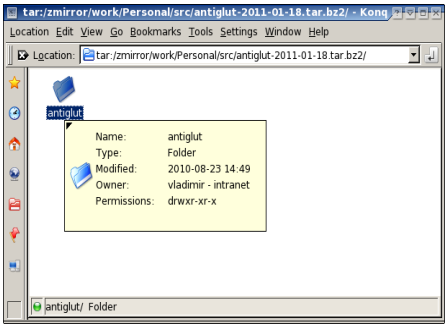


Затем достаточно выделить один или несколько извлекаемых файлов (рис. 10) и перетащить выделенную группу на рабочий стол.

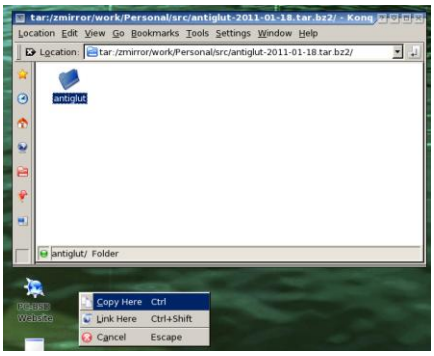


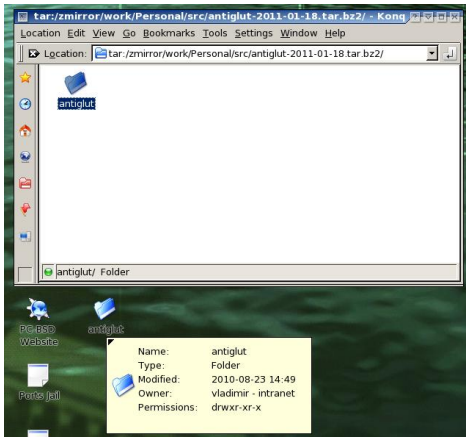


Извлечение всех документов из архива.
Для извлечения всех документов из архива достаточно следовать инструкциям, выделив все содержащиеся в архиве файлы или каталог верхнего уровня в архивном файле.



Затем, следует перетащить выбранный объект (объекты) на рабочий стол.





Порядок выполнения

1. Изучите теоретический материал.
2. Продемонстрируйте группе презентацию на тему «Работа с архиваторами». Ответьте на вопросы учащихся.
3. Продемонстрируйте основные функции работы с операционной системой. Покажите преимущества и недостатки той или иной операционной системы.
4. Составьте отчет, включающий – хронологическое возможностей описание основных операционных систем, применяемых в персональных ЭВМ и краткие ответы не менее чем на 8 контрольных вопросов.

Примечание: рекомендуется пройти соответствующий блок тестов итогового контроля к курсу по тематике практической работы.

Контрольные вопросы.

1. Расскажите принципы работы с буфером обмена.
2. Перечислите способы, которыми можно поместить объект в буфер обмена (четыре способа).
3. Как произвести копирование/перемещение объекта перетаскиванием в пределах одного диска? С диска на диск?
4. Как в окне папки выделить: несколько смежных объектов; несколько несмежных объектов; все объекты?

5. Как удалить объект, минуя Корзину?
6. Что такое ярлык? Перечислите способы создания ярлыка.
7. Как сменить значок ярлыка? Какие файлы содержат значки ярлыков? Как быстро найти объект, на который ссылается ярлык?
8. Назначение программы Проводник? Перечислите способы запуска программы Проводник.
9. Какой файл запускает программу Проводник? Что значит знак "+"/ "–" возле значков некоторых папок в левой панели Проводника?
10. Как можно использовать малую цифровую клавиатуру для управления левой панелью?
11. Каково назначение папки Панель управления?
12. В каких режимах представляется Панель управления? Как переключаться между режимами?
13. Как переместить Панель задач? Как изменить размер Панели задач?
14. Как создать свою панель инструментов в Панели задач? На основе чего создается новая панель инструментов?
15. Как установить/убрать режим отображения на Рабочем столе значков Мой компьютер, Мои документы, Сетевое окружение и Internet Explorer? Как изменить вид их пиктограмм?
16. Как изменить схему цветового оформления?
17. Какие настройки мыши можно изменить и как?
18. Какие настройки Даты и времени можно изменить и как?
19. Какие настройки региональных параметров можно изменить и как?
20. Какие настройки языка и службы текстового ввода можно изменить и как?
21. Какие стили Главного меню существуют и как переключаться между ними?
22. Как добавить объект в верхнюю часть команды Все программы, в команду Все программы, группу программ в команду Все программы?
23. Дайте понятие файла, типа файла, имени файла, полного имени файла.
24. Как можно определить тип файла?
25. Какие ограничения накладываются на имена файлов?
26. Перечислите виды меню Windows. Назовите способы вызова каждого типа меню. Перечислите обозначения, принятые в меню.
27. Что такое «горячие» клавиши? Их назначение?
28. Объясните необходимость архивации файлов.
29. Перечислите способы создания архивов.
30. Перечислите известные Вам программы-архиваторы.
31. Перечислите три способа создания архивного файла.
32. Перечислите три способа добавления файла в архив.
33. Какие форматы поддерживает архиватор?
34. Перечислите три способа извлечения файла из архива.
35. Как создать многотомный архив?
36. Как создать самораспаковывающийся архив?
37. Как записать большой архив на несколько дисков?

Информационные источники

1. Архиваторы. Презентация. http://prezentacii.com/informatike/6285-arhivator.html
2. Архиваторы. Презентация. http://900igr.net/prezentatsii/informatika/Arkhivator/Arkhivator.html
3. Выбор операционной системы. http://itcom.in.ua/stati/ operatsionnye-sistemy/77-vybor-operatsionnoj-sistemy.html
tech/software/overview-operating-systems.html
4. Операционная система Windows для «чайников» http://composs.ru/operacionnaya-sistema-windows-7/
5. Сервисные программы Windows :http://www.windxp.com.ru/win7/articles33.htm
6. Category: Lossless compression algorithms [электронный ресурс]. /URL: http://en.wikipedia.org/wiki/ Category: Lossless_compression_algorithms
7. RAR [электронный ресурс]. /URL http://ru.wikipedia.org/wiki/RAR
8. gzip [электронный ресурс]. – URL http://ru.wikipedia.org/wiki/Gzip
9. bzip2 [электронный ресурс]. – URL http://ru.wikipedia.org/wiki/Bzip2
10. WinRAR archiver, a powerful tool to process RAR and ZIP files [электронный ресурс]. – URL http://www.rarlab.com
11. File Compression Software – Compress files, decompress files – WinZip [электронный ресурс]. – URL http://www.winzip.com/prodpagewz.htm
12. 7-Zip [электронный ресурс]. – URL http://www.7-zip.org

25. Создание Web страниц средствами MS Office

Краткие теоретические сведения.

1.Создание WEB-страниц средствами Word.

В качестве редакторов, упрощающих создание Web-сайтов, можно использовать приложения Microsoft Office – Word, Excel, PowerPoint и другие. При этом пользователь может не знать язык HTML и иметь привычную среду для оформления документа – WYSIWYG (что вижу, то и получаю). Огромное количество людей, использующих Word в своей повседневной работе, становятся потенциальными разработчиками HTML-документов.

Создать Web-станицу в Word можно двумя способами: с помощью мастера или шаблона, либо преобразовав существующий документ Word в формат HTML. При этом Word сам генерирует тэги HTML, хотя и не оптимальным образом.

Первый способ создания HTML-документов достаточно прост – надо начать создание документа «с нуля» и только следовать советам Мастера и использовать те средства, которые имеются в меню программы.

Второй способ - преобразование существующего документа Word в тэги HTML при сохранении файла-Word в формате HTML. Преобразование естественно приводит к тому, что какие-то элементы оформления документа будут утрачены или изменены.

Одной из отличительных особенностей HTML-документов является то, что сам документ содержит только текст, а все остальные объекты встраиваются в документ в момент его отображения Браузером с помощью специальных тэгов и хранятся отдельно. При сохранении HTML-файла в месте размещения документа Word создает на диске папку, в которую помещает сопутствующие ему графические элементы оформления. Например, при сохранении файла с рисунками - frieds.htm, Word создает папку frieds.files, в которой и разместит все рисунки.

Поэтому при создании сайта – группы взаимосвязанных Web-страниц, рекомендуется помещать сайт в отдельную папку, и при перемещении или публикации сайта строго сохранять всю внутреннюю структуру папок.

При подготовке публикации в Интернет материалов, созданных в Word, полезно знать особенности преобразования в формат HTML. Некоторые из них приводятся ниже (таблица 1).

Таблица 1

Элемент документа Word	Преобразование Word → HTML
Размеры шрифтов	В Word изображаются шрифты от 9 до 36 пунктов. Размеры шрифтов HTML изменяются от 1 до 7 и служат Браузеру указанием на размер шрифта
Текстовые эффекты: приподнятый, с тенью, уплотненный и т.д.	Текстовые эффекты не сохраняются, но сам текст остается

Элемент документа Word	Преобразование Word → HTML
Начертания: полужирный, курсив, подчеркивание	Остаются, но некоторые виды подчеркивания преобразуются в сплошную линию
Анимация текста	Анимация не сохраняется, но текст остается. Для придания тексту анимационного эффекта можно использовать бегущую строку (панель «Web-компоненты»)
Графика	Изображения преобразуются в формат GIF или JPEG, если они не были сохранены до этого в таких форматах. Линии преобразуются в горизонтальные линии.
Графические объекты: автофигуры, фигурный текст, надписи и тени	Объекты преобразуются в файлы формата GIF. В среде редактирования Web-страницы можно вставить графические средства «Вставка» - «Рисунок»
Таблицы	Таблицы преобразуются, однако параметры, не поддерживаемые HTML, не сохраняются. Например, цветные границы и границы переменой ширины
Нумерация страниц и колонтитулы	Так как документ HTML считается одной Web-страницей, то понятие колонтитулов отсутствует и нумерация не сохраняется
Поля страниц и многоколонный текст	Чтобы сохранить разметку страницы следует использовать таблицы
Стили	Определенные пользователем стили преобразуются в прямое форматирование, если оно поддерживается HTML

При необходимости вставить на HTML-страницу собственные тэги, в Word предусмотрено непосредственное редактирование HTML-кода. Это можно сделать, вызвав окно редактора через меню «Вид» - «Источник HTML». До того, как перейти в этот режим следует сделать сохранение файла. После завершения редактирования перед закрытием окна, файл также нужно сохранить.

Другая возможность – открыть документ в Браузере и вызвать меню «Вид» - «В виде HTML». По умолчанию редактирование выполняется в Блокноте.

Хотя Word отображает документ практически в том же виде, как он в дальнейшем будет выглядеть в Браузере, предварительный просмотр Web-документа можно выполнить, не покидая Word.

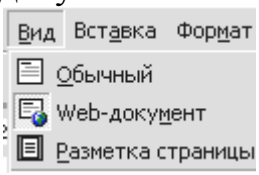


Рисунок 1

На рисунке 1. приведены способы отображения документа Word, устанавливаемые в меню «Вид». При выборе «Web-документ» файл отображается так, как будет выглядеть в Браузере, установленном на компьютере. Через меню «Вид» можно вернуться к обычному режиму работы с документом.

ОСНОВНЫЕ ТЕРМИНЫ

ASCII (American Standard Code for information Interchange – Американский стандартный код для обмена информацией) – семиразрядный код для представления текстовой информации.

GIF (Graphics Interchange Format – формат обмена графикой) – используется для немногоцветных четких изображений, поддерживает прозрачные области и анимацию.

HTML (Hyper Text Markup Language) – язык разметки гипертекстов.

JPEG (Joint Photography Experts Group – объединенная группа экспертов по машиной обработке фотоизображений) – распространенный формат растровой графики, который идеально подходит для сканируемых фотографий. *Web-page (Web-страница)* – электронный документ, предназначенный для размещения (размещенный) в Интернет. *Web-site (Web-сайт)* – совокупность Web-страниц, связанных между собой темой, гиперссылками и близким расположением в Интернет.

Web server (Web-сервер) – сервер, предназначенный для размещения Web-сайтов в Интернет, и предоставляющий услуги по запросам клиентов.

WYSIWYG (What You See Is What You Get – что вижу, то и имею) – тип редакторов, в которых документ отображается так, как он будет напечатан или представлен в Интернет.

Рассмотрим этапы создания WEB-страниц.

1. На рабочем диске подготовить папку для размещения Ваших документов.
2. Создать документ Word следующего содержания.

Главный заголовок, например Объект WordArt:

Далее 3 – 4 абзаца о своем происхождении (краткая автобиография).

Затем разместить текст:

«Далее Вы можете узнать подробности:»

И создать оглавление, состоящее, например, из 2-х пунктов.

Мои увлечения *Мои друзья*

3. Оформить документ и сохранить в своей папке на жестком диске под именем main.doc.
4. Создать документ Word, посвященный Вашим увлечениям. Сохранить документ под именем hobby.doc в своей папке. Документ должен быть оформлен, иметь нижний колонтитул и кроме текста содержать рисунки.
5. Создать книгу Excel с таблицей по приведенному образцу (рис. 2), сохранить под именем friends.xls.

Для вычисления средних значений должны быть использованы *формулы*.

Мои друзья			
Имя	Возраст	Рост	Вес
Вася	18	189	90
Среднее значение	27,5	176	71,75

Рисунок 2. Пример таблицы для файла friends.xls

1. Постройте графики, иллюстрирующие сведения о Ваших друзей. Расположить графики, под таблицей, подогнать размеры таблицы и графиков.
2. Подготовьте лист к печати:

5.2.1. Настроите параметры вкладки «Страница…» в режиме Предварительного просмотра, создать колонтитулы.

6. Установите связи между документами с помощью гиперссылок – для чего необходимо:
1. Открыть главный документ main.doc и последовательно выделяя заголовки разделов, закрепить за ними гиперссылки («Меню - Вставить») на соответствующие документы.
2. Сохранить документ и проверить работоспособность гиперссылки. Возврат в Главный документ выполнять с помощью кнопки  на панели инструментов
7. В главном документе установить закладку на заголовок Мои увлечения. Дать ей название «Хобби». Сохранить документ.
8. Создать в конце каждого вспомогательного документа гиперссылки, обеспечивающие возврат в основной документ.
1. Подготовить рисунок для обеспечения возврата из вспомогательных документов в главный. Например, рисунок  можно получить с помощью создания графической копии активного окна в буфере (Alt+PrintScreen) и дальнейшего редактирования рисунка в редакторе Paint.
2. Вставить в конец каждого из документов рисунок и закрепить за ним гиперссылку на документ main.doc. В файле hobby.doc гиперссылка должна обеспечивать переход на закладку «Хобби».
9. Сохранить документы и проверить работу гиперссылок.

2. Создание группы связанных Web-страниц, методом преобразования подготовленных документов.

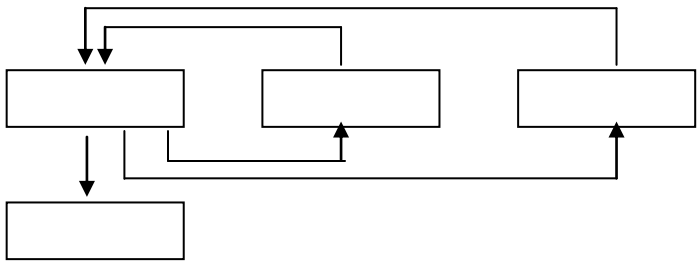
1. Подготовьте папку для Web-документов с именем My_Web.
2. Последовательно раскрывая подготовленные ранее документы, сохраните их в папке My_Web, указав
Тип файла**:** Web-страница (*.htm; *.html)
3. Закрыть все документы, проанализировать изменения, произошедшие в структуре папок.
10. Просмотреть Web-документы, начиная с main.htm. Проанализировать, какие элементы документов изменились или вовсе исчезли. Сделать попытку сделать переход по гиперссылке. Убедиться в том, что связи между Web-страницами нуждаются в редактировании.
11. Отредактировать Web-документы, изменить гиперссылки, выполнить дополнительное оформление.
Внимание: Для перехода из Браузере в режим редактирования нужно воспользоваться меню «Файл» - «Править в Microsoft Word for Windows» или кнопкой  на панели инструментов.
12. Сохранить и закрыть все документы, скопировать папку My_Web на диск A:. Предъявить работу Web-страниц преподавателю.

3. Создание новых Web-документов с помощью приложений MS Office

1. Познакомиться со структурой и составом многостраничного гипертекстового документа, объединяющего четыре страницы (см. приложение).
2. Создать папку с именем *Presentation*, а в ней папку для Ваших рисунков - *Gallery*.
3. Подготовить рисунки для включения их в соответствующие страницы. Сохранить их в папке *Gallery* в виде отдельных файлов формата bmp, gif или любого другого формата, используемого в Интернет. Обратить внимание на размер рисунков и объем файлов. Объем файлов не должен превышать 3 - 10 КБ.
4. Создать отдельный файл для каждой страницы с помощью текстового процессоа Word (стр.1,2,4). Для Страницы 3 использовать готовый файл friends.htm. Сохранять файлы в формате htm или html в папке *Presentation*.
4.1. Запустить текстовый процессор Word и с его помощью создать главную страницу, сохранить файл в формате html под именем index.htm.
4.1.1. При создании структурированного документа рекомендуется использовать таблицу. После размещения объектов снять обрамление таблицы.
4.1.2. Выполнить оформление документа. Для главного заголовков использовать объект WordArt, для прочих – стиль Заголовок
4.1.3. Для оформления фона использовать один из текстурных способов заливки (меню «Формат» – «Фон» – «Способы заливки») или тематическое оформление (меню «Формат» – «Тема»).
4.1.4. Просмотреть в броузере изменения в структуре папок, произошедшие при сохранении Web-странички. Открыть созданную страницу, при необходимости отредактировать ее.
4.2. Создать вторую страницу сайта с помощью редактора Word.
4.2.1. Для создания документа воспользоваться пунктом меню «Файл»-«Создать», в открывшемся диалоговом окне «Создание документа» выбрать вкладку «WEB-станицы» и пиктограмму «Новая WEB-страница». Сохранить чистую WEB-страницу в папке *Presentation*, дав странице имя на английском языке childhood.htm
4.2.2. В качестве заголовка "Мое детство" использовать *Бегущую строку*, отобразив предварительно панель Web-компонентов. Познакомиться с параметрами Бегущей строки, настроить ее так, чтобы она появлялась не более 2-х раз.
4.2.3. Разметку для размещения объектов сделать с помощью таблицы. Рисунок вставить из папки *Gallery*.
4.2.4. Оформить фон страницы, используя двухцветную градиентную заливку. Выбрать цвета близкие к цвету текстуры или темы главной страницы.
4.2.5. Сохранить документ и просмотреть его в Internet Explorer. При необходимости отредактировать.
4.3. Создать третью страницу на основе созданного в Части 1 файла friends.htm.
4.3.1. Скопировать файл и сопутствующую ему папку в папку *Presentation*.

4.3.2. Открыть файл в Браузере, убедиться, что документ отображается правильно. При необходимости отредактировать.
4.4. Создать четвертую страницу сайта с помощью редактора Word.
4.4.1. Подготовить рисунок – вид здания ГУТ со стороны Мойки (найти старый сайт ГУТ). Сохранить рисунок в папке *Gallery*.
4.4.2. Создать файл в папке *Presentation,* с именем **university.htm**.
4.4.3. Вставить в файл рисунок – вид здания ГУТ, предварительно сохраненный в папке *Gallery*. Скопировать или ввести электронный адрес университета. Написать несколько фраз о Вашем факультете и вставить гиперссылку на сайт факультета.
5. Установить связи между документами сайта.
5.1. Открыть в *Word* документ *index.htm***,** и последовательно выделяя пункты "Содержания", вставить гиперссылки на соответствующие документы.
5.2. Сохранить файл и обновить его просмотр в браузере. Проверить правильность выполнения переходов по гиперссылкам.
6. Вставить в конец каждого из документов рисунок (). Создать гиперссылки, обеспечивающие возврат в главный документ, закрепив из за рисунком.
Внимание! Рисунок для переходов на главную страницу также должен находиться в папке *Gallery.*
7. Сохранить изменения в файлах и обновить просмотр сайта в Internet Explorer.
8. Просмотреть содержание каждого из вновь созданных файлов в формате HTML.
8.1. Найти тэги, обеспечивающие вставку гиперссылок, рисунков.
8.2. Убедиться, что ссылки на рисунки имеют относительную адресацию. В противном случае отредактировать их.
9. Скопировать папку *Presentation* со всем ее содержимым на диска A: или на диск C: в папку Temp. Протестировать работу сайта, запустив его просмотр из нового места размещения.
10. Предъявите преподавателю Ваш сайт, начав его просмотр с главной страницы.

Примечание: типовая структура и содержание сайта.



Страница 1 (главная страница)

<table>
<tr><td colspan="3">Приветствие</td></tr>
<tr><td rowspan="2">Содержание
Мое детство
(ссылка на стр. 2)
Мои увлечения
(ссылка на стр. 3)
Мои университеты
(ссылка на стр. 4)</td><td rowspan="2">Представление – краткое резюме</td><td>Рисунок или фотография</td></tr>
<tr><td>Пишите мне (адрес e-mail)</td></tr>
</table>

Страница 2 (Мое детство)

<table>
<tr><td>Мое детство
Фотография или рисунок
Текст
(несколько строк на заданную тему)</td><td>Текст
(комментарии к фотографии)</td></tr>
</table>

Страница 3 (Мои друзья) – использовать ранее созданный Web-документ friends.htm, созданный с помощью Excel.

Страница 4 (Мои университеты)

Полное название Университета, адрес сайта ГУТ, картинка с фотографией главного здания ГУТ

О Вашем факультете, ссылка на сайт Вашего факультета.

Порядок выполнения

1. Изучите теоретический материал.
2. Создайте WEB- страницу рассмотренными в теоретических сведениях способами (содержание страниц сайта может быть выбрано на усмотрение автора).
3. Оформите отчет, включающий ссылку на созданный сайт, текст созданного файла, скриншоты основных режимов работы, ответы на контрольные вопросы.

Контрольные вопросы

1. Какими способами можно создать Web-страницу в Word?
2. Как Word изменяет структуру папок на диске при сохранении новой Web-страницы?
3. Какие новые приемы оформления документа появляются при работе с Web-страницами? Какие становятся недоступны?
4. Как вставить гиперссылку на другой документ? Как вставить рисунок?
5. Как просмотреть, отредактировать тэги HTML-документа, созданного в Word?

26. Автоматизированный комплекс изучения характеристик оператора ЭВМ

Цель работы: овладение навыками подготовки, проведения и системной обработки результатов эксперимента по исследованию психических характеристик состояния человека – оператора ЭВМ.

Краткие теоретические сведения.

Одно из основных звеньев в биотехнических системах составляет поведение организма для достижения целей своего существования (жизнеобеспечения) в условиях изменяющейся окружающей среды при помощи и под обратным воздействием окруженных технических средств. Если рассматривать биомедецинский аспект проблемы оптимизации (удовлетворяемости) подобного поведения, то в качестве организма обычно рассматривается человек, а характеристики его поведения обычно носят психосоматический характер.

При непосредственном контакте человека с окружающей средой в процессе удовлетворения потребностей существования своей популяции на уровне обычных органов чувств возникновения дискомфортных ситуаций средне вероятно. Но если в этой цепочки появляются «усилители» органов чувств, а именно таковыми являются информационно-измерительные приборы с и без интеллектуальным интерфейсом, то напряжение, приводящее при длительном воздействии к дискомфорту, резко нарастает (в основном из-за перенасыщения информационного канала в ситуации следствии несогласования скоростей и объема передаваемой и усвояемой информации). Взаимодействие человеческого организма с информационно-измерительными приборами и информационно-управляющими органами получило название операторской детальности.

Непременное условие эффективного протекания любого вида человеческой деятельности - это внимание. Особую, практическую значимость исследования свойств внимания приобретает при анализе и оптимизации трудовой деятельности. Изменение и содержание характера труда постоянно предъявляет возрастающее требования к сенсорному вниманию, которое характеризуется такими параметрами как: селективность, переключаемость, объем, распределяемость, устойчивость и концентрированность.

Селективность внимания характеризует направленность деятельности человека как звено эргатической системы по переработке информации на ограниченную часть наличного входа, что позволяет выделить в практических ситуациях релевантную и значимую информацию и активно подавлять незначимый и ирелевантный фоны. Изменение селективности сенсорного внимания может служить индикатором изменений функционального состояния и работоспособности человека.

Переключаемость внимания является свойством человеческой деятельности, которое характеризует способность субъекта гибко изменять направленность различных психических процессов в соответствии с текущим изменением деятельности (функционирования биологической части в БТС).

Объем внимания характеризуется способностью человека одновременно охватывать определенное число субоднородных объектов. Количество одновременно воспринимаемых объектов принимается в качестве характеристики объема внимания. Последний в большей степени зависит от умения человека осмысленно связывать и структуировать информацию.

С объемом внимания тесно связана такая характеристика, как *распределяемость зрительного внимания*, которая заключается в способности человека одновременно концентрировать внимание на нескольких разнородных объектах.

Устойчивость внимания тесно связана с показателями стабильности деятельности человеческого организма на протяжении заданного времени.

Концентрированность внимания характеризует интенсивность сосредоточенности человека. Она предполагает активное и направленное сужение внимания на одном или нескольких ситуационных элементов и умение отстранится от помех (незначимой для биологической части БТС в текущий момент времени информации).

Основными функциональными единицами, характеризующими состояние сенсорных систем, являются показатели их абсолютной и разнозностной чувствительностей, связанные обратной зависимостью с показателями абсолютных и разностных порогов. Измерение данных порогов требует строгого контроля многих переменных, - как предъявляемого физического стимула, так и условий проведения исследований. Внешним выражением психических процессов, обеспечивающих действенную практическую связь человека с окружающим миром, является психомоторика.

Формально, психомоторные процессы характеризуются показателями скорости, точности, темпа, выносливости, силы, координированности. Причем, интеллектуализация современного труда уменьшает удельный вес в производственных силовых движениях,- при этом растут требования к скорости и точности. Вышеизложенное вызывает преобразование на современном производстве локальной мышечной нагрузки при ограничении общей подвижности.

Немаловажной характеристикой в цепях обратной связи в управляющих контурах БТС является память, как биологической, так и технической частях системы. И если вторая носит в основном пассивный характер отслеживания и изменения уставок, то первая - активных.

Биопамять, (в частности, человеческого организма) - это процесс психического отражения прошлого опыта посредством его запечатления, сохранения и воспроизведения.

Согласно концепции когнитивной психологии, память включает в себя три блока: сенсорная память, кратковременная оперативная и долговременная памяти. Считается, что сенсорная память связана с краткосрочными процессами последствий в сенсорных системах (например, зрительной) и позволяет в

течении миллисекунд сохранить в модально-специфическом виде следы раздражителей во всем их чувствительном многообразии.

В Таблице 1 приведена обобщенная структурная схема диагностики основных психических свойств человеческого организма, работающего в техническом контакте с окружающей средой. С помощью различных методик, реализуемых соответствующей батареей тестов, осуществляется регистрация определенных показателей, характеризующих состояние внимания, памяти, сенсорных процессов, психомоторики. Блоком выбора (настройки) информативных признаков осуществляется отбор тех из них, которые наиболее чувствительны к характеру решаемой задачи-исследования.

Таблица 1

Внимание	Память	Психомоторика
Селективность Переключаемость Объем Распределяемость Устойчивость Концентрированность	Сенсорная Фильтрация Перекодиров. Кратковрем. Повторение	Координация Скорость Точность Сила Выносливость Темп
Сенсорные системы	Мышление	Личность
Зрительная Слуховая Кинестетическая	Понятийное Наглядно-действенное	Тестовые опросники Конформность Групповая активность|
Исследователь	Система диагностики	Выбор информативных признаков

При выполнении лабораторной работы осуществляется тестирование психических характеристик операторской деятельности по ряду методик, большинство из которых организованы так, что вне зависимости от измеряемых признаков выходными данными являются некоторые коэффициенты функционально-идентичным образом зависимые от: среднего времени латентного периода реакции, среднего разброса этого времени и качества ответов. Темп и сложость диагностического процесса устанавливается в интерактивном режиме исследователем.

Для исследования селективности зрительного внимания предлагается следующая методика. Испытуемому на экране монитора предъявляются в случайном порядке и в заданном темпе цифры натурального ряда и измеряется среднее время его реакции на их появление. Затем цифры предъявляются на фоне визуального шума. В итоге рассчитывается показатель:

$$СВ = to*(N-Co)/(tn*(N-Cn)) , \quad (1)$$

где to, tn - среднее время опознания цифр без помех и с помехой,

Co, Cn - число ошибок испытуемого без помех и с помехой,
N - число предъявлений стимула.
При исследовании переключаемости испытуемому предлагается реагировать на один из двух предупредительных сигналов («0» и «1»; «чет» - «нечет») и на один пусковой («*»). При появлении первого сигнала испытуемый ожидает появление пускового и как можно быстрее нажимает на определенную функциональную клавишу. При появлении второго предупредительного сигнала испытуемый нажимает на клавишу после исчезновения пускового сигнала. Тем самым проверяется умение человека как звена эргатической системы гибко реагировать на изменение существующей установки. В конце эксперимента рассчитывается показатель:
ПВ = N*t / (N - C) , (2)
где C - число ошибок, t - среднее время реакции.
Для исследования устойчивости внимания предлагается методика, в соответствии с которой испытуемому в случайном порядке предъявляются цифры натурального ряда, которые он должен классифицировать по правилу: при появлении четной цифры нажимает на одну клавишу, нечетной - другую. Нажатие приводит к мгновенной смене цифры. Средняя скорость реакции при этом характеризует темп психических процессов. В итоге рассчитывается показатель:
УВ = N*(t - M(t)) / ((N - C)* M(t)) (3)

При реализации методики «поиска сигнала в шуме» тестируется состояние блока сенсорной памяти, операций фильтрации и перекодирования информации. В определенном экспериментатором темпе на экране монитора появляется цифра натурального ряда, выполняющая роль инструкции. Затем предъявляется последовательность неповторяющихся цифр. Испытуемый должен указать (нажатием определенной функциональной клавиши) присутствовала ли в последовательности цифра - инструкция.
При реализации методики «опознание» испытуемому вначале предлагаетсчя шесть натуральных цифр-стимулов, после чего испытуемый должен указать отсутствующую цифру.
При реализации методики «воспроизведение» тестируется состояние кратковременной памяти и операции повторения. На экране дисплея предъявляется случайная неповторяющаяся последовательность цифр натурального ряда. От испытуемого требуется после его исчезновения как можно быстрее и точнее воспроизвести заданную последовательность.
При реализации методики определение отсутствующей цифры испытуемому предъявляется случайная неповторяющаяся последовательность натурального ряда, затем она исчезает и через определенное время предъявляется вновь, но без одной цифры.
Испытуемый должен как можно быстрее воспроизвести эту цифру.

Во всех перечисленных методиках рассчитывается показатель, аналогично формуле (2).
Измерение способности к оценке величины геометрических размеров (элемент мыслительного процесса) осуществляется следующим образом. В левом верхнем углу экрана монитора изображается метрический эталон. Испытуемому предъявляются квадраты разных размеров. Требуется как можно быстрее указать размер стороны квадрата в единицах метрики эталона. Фиксируется разность между истинными размерами фигур и размерами, определенными испытуемым, а так же среднее время реакции.
Исследование эффективности манипулирования образами разно ориентированных объектов (процесс ассоциативного мышления, особенно важный для управляющих быстрыми транспортными средствами в экстремальных условиях) в соответствии с методикой Леоновой А.Б. осуществляется следующим образом. На экране монитора предъявляются две сложные идентичные геометрические фигуры (левая - эталон). Задача испытуемого состоит в определении того, в каком из четырех положений находится правая фигура по отношению к левой: фигуры одинаково ориентированы на плоскости, зеркально отражены, перевернуты, зеркально-перевернуты. Фиксируется время латентного периода и правильность ответа (формула (2)).

Порядок выполнения работы.
1. Изучить теоретический материал.
2. Изучить принципы работы с автоматизированной системой оценки психических характеристик, путем инициализации исполнительного файла и изучения интерфейса системы. На этом этапе студент овладевает навыками самостоятельного освоения новой информации. Программное обеспечение снабжено дружественным интерфейсом и стандартизованным использованием функциональных клавиш ПЭВМ. На данном этапе студент должен изучить: - как запускать систему, - как настроить батарею тестов, - как провести тестирование, - как отобразить и зафиксировать результат.
На выполнение данного этапа отводится 30-40 минут.
3. Провести исследование следующих психических характеристик:
- селективности внимания,
- устойчивости внимания,
- переключаемости внимания,
- поиска сигнала в шуме,
- определения отсутствующей цифры,
- воспроизводимости числового ряда,
- определения геометрического размера фигуры,
- манипулирования объектами,
- опознания отсутствующего элемента /цифры в ряду/ следующим образом.

3.1. Не изменяя стандартную цветовую палитру, настроить батарею тестов на выполнение только одного исследования с 15 предъявляемыми стимулами.
3.2. Провести исследование со следующими временами предъявления стимулов: 1, 2, 3, 4, 5, 10 секунд и зафиксировать в тетради результат каждого эксперимента.
3.3. Повторить п.3.2. в следующих вариантах цветовой палитры:
- фон черный - стимул белый,
- фон белый - стимул черный,
- фон желтый - стимул белый.
4. По данным п.3 оформить следующую таблицу:
Фамилия, И.О. _____ дата рождения _____ пол ___ Методика _____

N опыта		Значение показателя		обозначение
Тпред				X1
Цвет фона				X2
Цвет стимула				X3
Значение показателя психической характеристики				X4

5. Построить графические зависимости X4-X1, X4-X2, X4-X3, X4-F(X2,X3).
6. Осуществить линейную и нелинейную идентификации X4=F(X1) с помощью инструментария Excel.
Примечание: п.п.5 и 6 проводятся вне аудиторных занятий, в ходе самостоятельной работы студентов.
7. Сделать выводы о: влиянии времени предъявления стимулов, соотношения цветов фона и стимула на значение соответствующих зарегистрированных показателей, сформулировать индивидуальную норму.
8. Оформить отчет, в который входит: таблица полученных результатов, графические материалы, идентифицированные математические зависимости, сделанные выводы, ответы на контрольные вопросы (не менее 3)

Контрольные вопросы:
1. Что такое эргатическая система?
2. Каким образом оценивается внимание оператора ЭВМ?
3. Каким образом оценивается реакция оператора ЭВМ?
4. Как влияют цвета фона и стимула (как аналога информационного сигнала) на деятельность оператора ЭВМ?
5. Какие вопросы рассматривает когнитивная психология?
6. Как исследуется устойчивость внимания?
7. Как исследуется селективность внимания?
8. Как исследуется переключаемость внимания?
9. Как исследуется характеристика «поиска сигнала в шуме»?
10. Как исследуется характеристика «определения отсутствующей цифры»?

11. Как исследуется характеристика «воспроизводимость числового ряда»?
12. Как исследуется характеристика «определения геометрического размера фигуры»?
13. Как исследуется характеристика «манипулирования объектами»?
14. Как исследуется характеристика «опознания отсутствующего элемента»?
15. В чем заключается понятие «функциональное состояние» человека, как звена эргатической системы?
16. Каким образом можно корректировать (управлять) функциональным состоянием человека в процессе медицинских процедур?
17. Какую информацию несут характеристики о психическом состоянии оператора ЭВМ?
18. В каких медицинских исследованиях можно исследовать информационные характеристики, представленные в работе, для оценки состояния здоровья человека.