**Министерство сельского хозяйства**
**Российской Федерации**

**Технологический институт-филиал ФГБОУ ВО Ульяновский ГАУ**

О.А. Дмитриев

# ИНФОРМАТИКА

курс лекций

**Димитровград - 2019**

**Дмитриев, О.А.** Информатика: курс лекций / О.А. Дмитриев - Димитровград: Технологический институт – филиал УлГАУ, 2019.- 205 с.

Рецензенты: Голубев Владимир Александрович, кандидат технических наук, доцент кафедры «Эксплуатация мобильных машин и технологического оборудования» ФГБОУ ВО Ульяновский ГАУ

Ротанов Евгений Геннадьевич, кандидат технических наук, доцент кафедры «Естественнонаучные и технические дисциплины», ПКИУПТ (филиал) ФГБОУ ВО «МГУТУ ИМ. К.Г.РАЗУМОВСКОГО (ПКУ)»

Информатика: курс лекций предназначен для подготовки бакалавров очной и заочной форм обучения по направлению подготовки 23.03.03 «Эксплуатация транспортно-технологических машин и комплексов».

Утверждено
на заседании кафедры «Эксплуатация транспортно-технологических машин и комплексов»
Технологического института – филиала
ФГБОУ ВО Ульяновский ГАУ,
протокол № 1 от 4 сентября 2019г.

Рекомендовано
к изданию методическим советом Технологического института – филиала
ФГБОУ ВО Ульяновский ГАУ
Протокол № 2 от 10 октября 2019г.

# ЧАСТЬ ПЕРВАЯ

## *ГЛАВА 1*

## ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ

### § 1. ИНФОРМАТИКА КАК НАУКА И КАК ВИД ПРАКТИЧЕСКОЙ ДЕЯТЕЛЬНОСТИ

### 1.1. ИСТОРИЯ РАЗВИТИЯ ИНФОРМАТИКИ

Информатика - молодая научная дисциплина, изучающая вопросы, связанные с поиском, сбором, хранением, преобразованием и использованием информации в самых различных сферах человеческой деятельности. Генетически информатика связана с вычислительной техникой, компьютерными системами и сетями, так как именно компьютеры позволяют порождать, хранить и автоматически перерабатывать информацию в таких количествах, что научный подход к информационным процессам становится одновременно необходимым и возможным.

До настоящего времени толкование термина «информатика» (в том смысле как он используется в современной научной и методической литературе) еще не является установившимся и общепринятым. Обратимся к истории вопроса, восходящей ко времени появления электронных вычислительных машин.

После второй мировой войны возникла и начала бурно развиваться кибернетика как наука об общих закономерностях в управлении и связи в различных системах: искусственных, биологических, социальных. Рождение кибернетики принято связывать с опубликованием в 1948 г. американским математиком Норбертом Винером, ставшей знаменитой, книги «Кибернетика или управление и связь в животном и машине». В этой работе были показаны пути создания общей теории управления и заложены основы методов рассмотрения проблем управления и связи для различных систем с единой точки зрения. Развиваясь одновременно с развитием электронно-вычислительных машин, кибернетика со временем превращалась в более общую науку о преобразовании информации. Под информацией в кибернетике понимается любая совокупность сигналов, воздействий или сведений, которые некоторой системой воспринимаются от окружающей среды (входная информация *X),* выдаются в окружающую среду (выходная информация У), а также хранятся в себе (внутренняя, внутрисистемная информация Z), рис. 1.1.

Развитие кибернетики в нашей стране встретило идеологические препятствия. Как писал академик А.И.Берг, «... в 1955-57 гг. и даже позже в нашей литературе были допущены грубые ошибки в оценке значения и возможностей кибернетики. Это нанесло серьезный ущерб развитию науки в нашей стране, привело к задержке в разработке многих теоретических положений и даже самих электронных машин». Достаточно сказать, что еще в философском словаре 1959 года издания кибернетика характеризовалась как «буржуазная лженаука». Причиной этому послужили, с одной стороны, недооценка новой бурно развивающейся науки отдельными учеными «классического» направления, с другой - неумеренное пустословие тех, кто вместо активной разработки конкретных проблем кибернетики в различных областях спекулировал на полуфантастических прогнозах о безграничных возможностях кибернетики, дискредитируя тем самым эту науку.

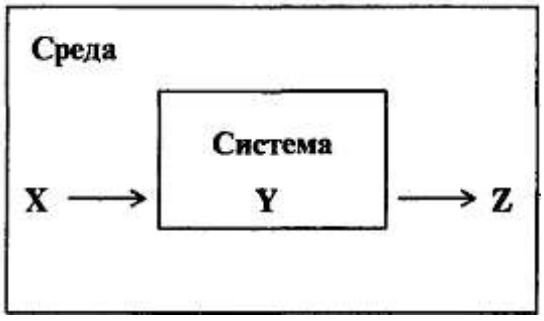


Рис. 1.1. Общая схема обмена информацией между системой и внешней средой

Дело к тому же осложнялось тем, что развитие отечественной кибернетики на лротяжении многих лет сопровождалось серьезными трудностями в реализации крупных государственных проектов, например, создания автоматизированных систем управления (АСУ). Однако за это время удалось накопить значительный опыт создания информационных систем и систем управления технико-экономическими объектами. Требовалось выделить из кибернетики здоровее научное и техническое ядро и консолидировать силы для развития нового движения к давно уже стоящим глобальным целям.

Подойдем сейчас к этому вопросу с терминологической точки зрения. Вскоре вслед за появлением термина «кибернетика» в мировой науке стало использоваться англоязычное «Computer Science», а чуть позже, на рубеже шестидесятых и семидесятых годов, французы ввели получивший сейчас широкое распространение термин «Informatique». В русском языке раннее употребление термина «информатика» связано с узко-конкретной областью изучения структуры и общих свойств научной информации, передаваемой посредством научной литературы. Эта информационно-аналитическая деятельность, совершенно необходимая и сегодня в библиотечном деле, книгоиздании и т.д., уже давно не отражает современного понимания информатики. Как отмечал академик А.П.Ершов, в современных условиях термин информатика «вводится в русский язык в новом и куда более широком значении - как название фундаментальной естественной науки, изучающей процессы передачи и обработки информации. При таком толковании информатика оказывается более непосредственно связанной с философскими и общенаучными категориями, проясняется и ее место в кругу "традиционных" академических научных дисциплин».

Попытку определить, что же такое современная информатика, сделал в 1978 г. Международный конгресс по информатике: «Понятие информатики охватывает области, связанные с разработкой, созданием, использованием и материально-техническим обслуживанием систем обработки информации, включая машины, оборудование, математическое обеспечение, организационные аспекты, а также комплекс промышленного, коммерческого, административного и социального воздействия».

## 1.2. ИНФОРМАТИКА КАК ЕДИНСТВО НАУКИ И ТЕХНОЛОГИИ

Информатика - отнюдь не только «чистая наука». У нее, безусловно, имеется научное ядро, но важная особенность информатики - широчайшие приложения, охватывающие почти все виды человеческой деятельности: производство, управление, науку, образование, проектные разработки, торговлю, финансовую сферу, медицину, криминалистику, охрану окружающей среды и др. И, может быть, главное из них - совершенствование социального управления на основе новых информационных технологий.

Как наука, информатика изучает общие закономерности, свойственные информационным процессам (в самом широком смысле этого понятия). Когда разрабатываются новые носители информации, каналы связи, приемы кодирования, визуального отображения информации и многое другое, конкретная природа этой информации почти не имеет значения. Для разработчика системы управления базами данных (СУБД) важны общие принципы организации и эффективность поиска данных, а не то, какие конкретно данные будут затем заложены в базу многочисленными пользователями. Эти общие закономерности есть предмет информатики как науки.

**Объектом** приложений информатики являются самые различные науки и области практической деятельности, для которых она стала непрерывным источником самых современных технологий, называемых часто «новые информационные технологии» (НИТ). Многообразные информационные технологии, функциони. рующие в разных видах человеческой деятельности (управлении производственным процессом, проектировании, финансовых операциях, образовании и т.п.), имея общие черты, в то же время существенно различаются между собой.

Перечислим наиболее впечатляющие реализации информационных технологий, используя, ставшие традиционными, сокращения.

АСУ - автоматизированные системы управления - комплекс технических и программных средств, которые во взаимодействии с человеком организуют управление объектами в производстве или общественной сфере. Например, в образовании используются системы АСУ-ВУЗ.

АСУТП - автоматизированные системы управления технологическими процессами. Напри-

мер, такая система управляет работой станка с числовым программным управлением (ЧПУ), процессом запуска космического аппарата и т.д.

АСНИ - автоматизированная система научных исследований - программно-аппаратный комплекс, в котором научные приборы сопряжены с компьютером, вводят в него данные измерений автоматически, а компьютер производит обработку этих данных и представление их в наиболее удобной для исследователя форме.

АОС - автоматизированная обучающая система. Есть системы, помогающие учащимся осваивать новый материал, производящие контроль знаний, помогающие преподавателям готовить учебные материалы и т.д.

САПР-система автоматизированного проектирования - программно-аппаратный комплекс, который во взаимодействии с человеком (конструктором, инженером-проектировщиком, архитектором и т.д.) позволяет максимально эффективно проектировать механизмы, здания, узлы сложных агрегатов и др.

Упомянем также диагностические системы в медицине, системы организации продажи билетов, системы ведения бухгалтерско-финансовой деятельности, системы обеспечения редакционно-издательской деятельности - спектр применения информационных технологий чрезвычайно широк.

С развитием информатики возникает вопрос о ее взаимосвязи и разграничении с кибернетикой. При этом требуется уточнение предмета кибернетики, более строгое его толкование. Информатика и кибернетика имеют много общего, основанного на концепции управления, но имеют и объективные различия. Один из подходов разграничения информатики и кибернетики - отнесение к области информатики исследований информационных технологий не в любых кибернетических системах (биологических, технических и т.д.), а только в социальных системах. В то время как за кибернетикой сохраняются исследования общих законов движения информации в произвольных системах, информатика, опираясь на этот теоретический фундамент, изучает конкретные способы и приемы переработки, передачи, использования информации. Впрочем, многим современным ученым такое разделение представляется искусственным, и они просто считают кибернетику одной из составных частей информатики.

## 1.3. СТРУКТУРА СОВРЕМЕННОЙ ИНФОРМАТИКИ

Оставляя в стороне прикладные информационные технологии, опишем составные части «ядра» современной информатики. Каждая из этих частей может рассматриваться как относительно самостоятельная научная дисциплина; взаимоотношения между ними примерно такие же, как между алгеброй, геометрией и математическим анализом в классической математике - все они хоть и самостоятельные дисциплины, но, несомненно, части одной науки.

**Теоретическая информатика** - часть информатики, включающая ряд математических разделов. Она опирается на математическую логику и включает такие разделы как теория алгоритмов и автоматов, теория информации и теория кодирования, теория формальных языков и грамматик, исследование операций и другие. Этот раздел информатики использует математические методы для общего изучения процессов обработки информации.

**Вычислительная техника** - раздел, в котором разрабатываются общие принципы построения вычислительных систем. Речь идет не о технических деталях и электронных схемах (это лежит за пределами информатики как таковой), а о принципиальных решениях на уровне, так называемой, **архитектуры** вычислительных (компьютерных) систем, определяющей состав, назначение, функциональные возможности и принципы взаимодействия устройств. Примеры принципиальных, ставших классическими решений в этой области - неймановская архитектура компьютеров первых поколений, шинная архитектура ЭВМ старших поколений, архитектура параллельной (многопроцессорной) обработки информации.

Программирование - деятельность, связанная с разработкой систем программного обеспечения. Здесь отметим лишь основные разделы современного программирования: создание системного программного обеспечения и создание прикладного программного обеспечения. Среди системного - разработка новых языков программирования и компиляторов к ним, разработка интерфейсных систем (пример - общеизвестная операционная оболочка и система Windows). Среди прикладного программного обеспечения общего назначения самые популярные - система обработки текстов, электронные таблицы (табличные процессоры), системы управления базами дан-

ных. В каждой области предметных приложений информатики существует множество специализированных прикладных программ более узкого назначения.

Информационные **системы** - раздел информатики, связанный с решением вопросов по анализу потоков информации в различных сложных системах, их оптимизации, структурировании, принципах хранения и поиска информации. Информационно-справочные системы, информационно-поисковые системы, гигантские современные глобальные системы хранения и поиска информации (включая широко известный Internet) в последнее десятилетие XX века привлекают внимание все большего круга пользователей. Без теоретического обоснования принципиальных решений в океане информации можно просто захлебнуться. Известным примером решения проблемы на глобальном уровне может служить гипертекстовая поисковая система WWW, а на значительно более низком уровне - справочная система, к услугам которой мы прибегаем, набрав телефонный номер *09'*.

**Искусственный интеллект** - область информатики, в которой решаются сложнейшие проблемы, находящиеся на пересечении с психологией, физиологией, лингвистикой и другими науками. Как научить компьютер мыслить подобно человеку? - Поскольку мы далеко не все знаем о том, как мыслит человек, исследования по искусственному интеллекту, несмотря на полувековую историю, все еще не привели к решению ряда принципиальных проблем. Основные направления разработок, относящихся к этой области - моделирование рассуждений, компьютерная лингвистика, машинный перевод, создание экспертных систем, распознавание образов и другие. От успехов работ в области искусственного интеллекта зависит, в частности, решение такой важнейшей прикладной проблемы как создание интеллектуальных интерфейсных систем взаимодействия человека с компьютером, благодаря которым это взаимодействие будет походить на межчеловеческое и станет более эффективным.

## 1.4. МЕСТО ИНФОРМАТИКИ В СИСТЕМЕ НАУК

Рассмотрим место науки информатики в традиционно сложившейся системе наук (технических, естественных, гуманитарных и т.д.). В частности, это позволило бы найти место общеобразовательного курса информатики в ряду других учебных предметов.

Напомним, что по определению А.П.Ершова информатика- «фундаментальная естественная наука». Академик Б.Н.Наумов определял информатику «как естественную науку, изучающую общие свойства информации, процессы, методы и средства ее обработки (сбор, хранение, преобразование, перемещение, выдача)».

Уточним, что такое **фундаментальная** наука и что такое **естественная** наука. К фундаментальным принято относить те науки, основные понятия которых носят общенаучный характер, используются во многих других науках и видах деятельности. Нет, например, сомнений в фундаментальности столь разных наук как математика и философия. В этом же ряду и информатика, так как понятия «информация», «процессы обработки информации» несомненно имеют общенаучную значимость.

Естественные науки - физика, химия, биология и другие - имеют дело с объективными сущностями мира, существующими независимо от нашего сознания. Отнесение к ним информатики отражает единство законов обработки информации в системах самой разной природы - искусственных, биологических, общественных.

Рис. 1.2. К вопросу о месте информатики в системе наук

Однако, многие ученые подчеркивают, что информатика имеет характерные черты и других групп наук - **технических и гуманитарных** (или общественных).

Черты технической науки придают информатике ее аспекты, связанные с созданием и функционированием машинных систем обработки информации. Так, академик А.А.Дородницын определяет состав информатики как «три неразрывно и существенно связанные части: технические средства, программные и алгоритмические». Первоначальное наименовании школьного предмета «Основы информатики и вычислительной техники» в настоящее время изменено на «Информатика» (включающее в себя разделы, связанные с изучением технических. программных и алгоритмических средств). Науке информатике присущи и некоторые черты гуманитарной (общественной) науки, что обусловлено ее вкладом в развитие и совершенствование социальной сферы. Таким образом, информатика является комплексной, междисциплинарной отраслью научного знания, как это изображено на рис. 1.2.

## 1.5. СОЦИАЛЬНЫЕ АСПЕКТЫ ИНФОРМАТИКИ

Термин «социальные аспекты» применительно к большей части наук, тем более фундаментальных, звучит странно. Вряд ли фраза «Социальные аспекты математики» имеет смысл. Однако, информатика - не только наука. Вспомним цитированное выше определение: «... комплекс промышленного, коммерческого, административного и социального воздействия».

И впрямь, мало какие факторы так влияют на социальную сферу обществ (разумеется, находящихся в состоянии относительно спокойного развития, без войн и катаклизмов) как информатизация. Информатизация общества - процесс проникновения информационных технологий во все сферы жизни и деятельности общества. Многие социологи и политологи полагают, что мир стоит на пороге информационного общества. В. А. Извозчиков предлагает следующее определение: «Будем понимать под термином «информационное» («компьютеризиро-ванное») общество то, во все сферы жизни и деятельности членов которого включены компьютер, телематика, другие средства информатики в качестве орудий интеллектуального труда, открывающих широкий доступ к сокровищам библиотек, позволяющих с огромной скоростью проводить вычисления и перерабатывать любую информацию, моделировать реальные и прогнозируемые события, процессы, явления, управлять производством, автоматизировать обучение и т.д.». Под «телематикой» понимаются службы обработки информации на расстоянии (кроме традиционных телефона и телеграфа).

Последние полвека информатизация является одной из причин перетока людей из сферы прямого материального производства в, так называемую, информационную сферу. Промышленные рабочие и крестьяне, составлявшие в середине XX века более 2/3 населения,,сегодня в развитых странах составляют менее 1/3. Все больше тех, кого называют «белые воротнички» - людей, не создающих материальные ценности непосредственно, а занятых обработкой информации (в самом широком смысле): это и учителя, и банковские служащие, и программисты, и многие другие категории работников. Появились и новые пограничные специальности. Можно ли назвать рабочим программиста, разрабатывающего программы для станков с числовым программным управлением? - По ряду параметров можно, однако его труд не физический, а интеллектуальный.

В табл.1 приведены статистические данные, описывающие изменения в профессиональной структуре труда в США (стране, где информатизация идет особенно быстро) за период с 1970 по 1980 г.

**Таблица 1.1 Изменения в структуре труда США за 10 лет**

| Категория работающих | 1970г., % | 1980г.,% | Относительный прирост численности, % |
|---|---|---|---|
| Работники сервиса | 19,9 | 21,5 | +0,1 |
| Рабочие (промышленные, сельскохозяйственные, фермеры) | 38,7 | 34,2 | -11,6 |
| Занятые обработкой информации (всего) | 41,5 | 44,4 | +6,7 |

| | | | |
|---|---|---|---|
| в том числе: | | | |
| менеджеры | 8,5 | 8,7 | +2,4 |
| конторские служащие | 18,0 | 18,9 | +5,0 |
| специалисты с высшим образованием | 15,0 | 16,8 | +12,0 |

**Таблица 1.2. Профессиональная структура занятости в экономике США (по данным на 1980 г.)**

| Отрасль | Работают с информацией, % | Работают с материальными объектами, % |
|---|---|---|
| Обрабатывающая промышленность | 40 | 60 |
| Транспорт и связь | 44 | 56 |
| Оптовая торговля | 68 | 32 |
| Розничная торговля | 58 | 42 |
| Сфера услуг | 63 | 37 |
| Финансовая деятельность | 92 | 8 |
| Государственные учреждения | 70 | 30 |

Динамика, отраженная в этой таблице, подтверждает сказанное выше. Разумеется, не вся она обусловлена информатизацией, есть и иные факторы, но информатизация вносит решающий вклад. Даже в традиционных сферах деятельности - промышленности и торговле - работа с информацией становится на уровень работы с материальными объектами, в чем убеждают данные, приведенные в табл. 1.2.

За годы, прошедшие с момента публикации этих данных, ситуация изменилась в сторону дальнейшего увеличения доли населения, занятого в профессиональном труде обработкой информации. К середине 90-х годов численность «информационных работников» (к которым причисляют всех, в чьей профессиональной деятельности доминирует умственный труд) достигла в США 60%. Добавим, что за те же годы производительность труда в США за счет научно-технического прогресса (ведь информатизация - его главная движущая сила) в целом выросла на 37%.

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

Казалось бы, компьютеризация и информационные технологии несут в мир одну лишь благодать, но социальная сфера столь сложна, что последствия любого, даже гораздо менее глобального процесса, редко бывают однозначными. Рассмотрим, например, такие социальные последствия информатизации как рост производительности труда, интенсификацию труда, изменение условий труда. Все это, с одной стороны, улучшает условия жизни многих людей, повышает степень материального и интеллектуального комфорта, стимулирует рост числа высокообразованных людей, а с другой - является источником повышенной социальной напряженности. Например, появление на производстве промышленных роботов ведет к полному изменению технологии, которая перестает быть ориентированной на человека. Тем самым меняется номенклатура профессий. Значительная часть людей вынуждена менять либо специальность, либо место работы - рост миграции населения характерен для большинства развитых стран. Государство и частные фирмы поддерживают систему повышения квалификации и переподготовки, но не все люди справляются с сопутствующим стрессом. Прогрессом информатики порожден и другой достаточно опасный для демократического общества процесс - все большее количество данных о каждом гражданине сосредоточивается в разных (государственных и негосударственных) банках данных. Это и данные о профессиональной карьере (базы данных отделов кадров), здоровье (базы данных учреждений здравоохранения), имущественных возможностях (базы данных страховых компаний), перемещении по миру и т.д. (не говоря уже о тех, которые копят специальные службы). В каждом конкрет-

ном случае создание банка может быть оправдано, но в результате возникает система невиданной раньше ни в одном тоталитарном обществе прозрачности личности, чреватой возможным вмешательством государства или злоумышленников в частную жизнь. Одним словом, жизнь в «информационном обществе» легче, по-видимому, не становится, а вот то, что она значительно меняется - несомненно.

Трудно, живя в самом разгаре описанных выше процессов, взвесить, чего в них больше - положительного или отрицательного, да и четких критериев для этого не существует. Тяжелая физическая работа в не слишком комфортабельных условиях, но с уверенностью, что она будет постоянным источником существования для тебя и твоей семьи, с одной стороны, или интеллектуальный труд в комфортабельном офисе, но без уверенности в завтрашнем дне. Что лучше? Конечно, вряд ли стоит уподобляться английским рабочим, ломавшим в конце XVIII века станки, лишавшие их работы, но правительство и общество обязаны помнить об отрицательных социальных последствиях информатизации и научно-технического прогресса в целом и искать компенсационные механизмы.

## 1.6. ПРАВОВЫЕ АСПЕКТЫ ИНФОРМАТИКИ

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

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

• можно ли, не копируя купленную программу, предоставить возможность пользоваться ею другому лицу;

• кому принадлежит авторское право на программу, созданную студентом в ходе выполнения дипломной работы;

• можно ли скопировать купленную программу для себя самого, чтобы иметь резервную копию;

• можно ли декомпилировать программу, чтобы разобраться в ее деталях или исправить ошибки;

• в чем состоит разница между авторским и имущественным правом.

Вопросов, подобных этим, возникает множество. Есть, конечно, такие, ответы на которые очевидны: нельзя создавать вирусы, нельзя хулиганить в сетях, нельзя в некоммерческих телеконференциях запускать коммерческую информацию, нельзя вскрывать и искажать защищенную информацию в чужих базах данных и т.д., т.е. совершать поступки, которые могут быть объектом уголовного преследования. Но на многие вопросы ответы отнюдь не очевидны, а иногда казуистически запутаны, причем не только в нашей стране. Остановимся на правовом регулировании в области информатики в России более подробно.

Необходимо отметить, что регулирование в сфере, связанной с защитой информации, программированием и т.д., является для российского законодательства принципиально новым, еще слабо разработанным направлением. К 1992 году был принят Закон Российской Федерации «О ПРАВОВОЙ ОХРАНЕ ПРОГРАММ ДЛЯ ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН И БАЗ ДАННЫХ», содержащий обширный план приведения российского законодательства в сфере информатики в соответствие с мировой практикой. Действие этого Закона распространяется на отношения, связанные с созданием и использованием программ для ЭВМ и баз данных. Также предусматривалось внести изменения и дополнения в Гражданский кодекс РФ, в Уголовный кодекс РФ, другие законодательные акты, связанные с вопросами правовой охраны программ для электронных вычислительных машин и баз данных, привести решения Правительства РФ в соответствие с Законом, обеспечить пересмотр и отмену государственными ведомствами и другими организациями РФ их нормативных актов, противоречащих указанному Закону, обеспечить принятие нормативных актов в соответствии с указанным Законом и т.д.

Главное содержание данного Закона - юридическое определение понятий, связанных с авторством и распространением компьютерных программ и баз данных, таких как Авторство, Адаптация, База данных, Воспроизведение, Декомпилирование. Использование, Модификация и т.д., а также установление прав, возникающих при создании программ и баз данных - авторских, имущественных, на передачу, защиту, регистрацию, неприкосновенность и т.д.

**Авторское право** распространяется на любые программы для ЭВМ и базы данных (как вы-

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

Предоставляемая настоящим Законом правовая охрана распространяется на все виды программ для ЭВМ (в том числе на операционные системы и программные комплексы), которые могут быть выражены на любом языке и в любой форме, и на базы данных, представляющие собой результат творческого труда по подбору и организации данных. Предоставляемая правовая охрана не распространяется на идеи и принципы, лежащие в основе программы для ЭВМ и базы данных или какого-либо их элемента, в том числе идеи и принципы организации интерфейса и алгоритма, а также языки программирования.

Авторское право на программы для ЭВМ и базы данных возникает в силу их создания. Для признания *у.* осуществления авторского права на программы для ЭВМ и базы данных не требуется опубликования, регистрации или соблюдения иных формальностей. Авторское право на базу данных признается при условии соблюдения авторского права на каждое из произведений, включенных в базу данных.

Автором программы для ЭВМ и базы данных признается физическое лицо, в результате творческой деятельности которого они созданы.

Если программа для ЭВМ и база данных созданы совместной творческой деятельностью двух и более физических лиц, то, независимо от того, состоит ли программа для ЭВМ или база данных из частей, каждая из которых имеет самостоятельное значение, или является неделимой, каждое из этих лиц признается автором такой программы для ЭВМ и базы данных.

Автору программы для ЭВМ или базы данных или иному правообладателю принадлежит исключительное право осуществлять и (или) разрешать осуществление следующих действий:

• выпуск в свет программы для ЭВМ и базы данных;

• воспроизведение программы для ЭВМ и базы данных (полное или частичное) в любой форме, любыми способами;

• распространение программы для ЭВМ и баз данных;

• модификацию программы для ЭВМ и базы данных, в том числе перевод программы для ЭВМ и базы данных с одного языка на другой;

• иное использование программы для ЭВМ и базы данных.

Однако, **имущественные права** на программы для ЭВМ и базы данных, созданные в порядке выполнения служебных обязанностей или по заданию работодателя, принадлежат работодателю, если в договоре между ним и автором не предусмотрено иное. Таким образом, имущественное право на программу, созданную в ходе дипломного проектирования, принадлежит не автору, а вузу - по крайней мере, пока между ними не будет заключено специальное соглашение.

Имущественные права на программу для ЭВМ и базу данных могут быть переданы полностью или частично другим физическим или юридическим лицам по договору. Договор заключается в письменной форме и должен устанавливать следующие существенные условия: объем и способы использования программы для ЭВМ или базы данных, порядок выплаты и размер вознаграждения, срок действия договора.

Лицо, правомерно владеющее экземпляром программы для ЭВМ или базы данных, вправе без получения дополнительного разрешения правообладателя осуществлять любые действия, связанные с функционированием программы для ЭВМ или базы данных в соответствии с их назначением, в том числе запись и хранение в памяти ЭВМ, а также исправление явных ошибок. Запись и хранение в памяти ЭВМ допускаются в отношении одной ЭВМ или одного пользователя в сети, если иное не предусмотрено договором с правообладателем. Также допускается без согласия правообладателя и без выплаты ему дополнительного вознаграждения осуществлять следующие действия:

• адаптацию программы для ЭВМ или базы данных;

• изготавливать или поручать изготовление копии программы для ЭВМ или базы данных при условии, что эт а копия предназначена только для архивных целей и при необходимости (в случае, когда оригинал программы для ЭВМ или базы данных утерян, уничтожен или стал непригодным для использования) для замены правомерно приобретенного экземпляра.

Лицо, правомерно владеющее экземпляром программы для ЭВМ, вправе без согласия пра-

вообладателя и без выплаты дополнительного вознаграждения выполнять **декомпплирование** программы для ЭВМ с тем, чтобы изучить кодирование и структуру этой программы при следующих условиях:

•информация, необходимая для взаимодействия независимо разработанной данным лицом программы для ЭВМ с другими программами, недоступна из других источников;

• информация, полученная в результате этого декомпилирования, может использоваться лишь для организации взаимодействия независимо разработанной данным лицом программы для ЭВМ с другими программами, а не для составления новой программы для ЭВМ, по своему виду существенно схожей с декомпилируемой программой.

Свободная **перепродажа** экземпляра программы для ЭВМ и базы данных допускается без согласия правообладателя и без выплаты ему дополнительного вознаграждения после первой продажи или другой передачи права собственности на этот экземпляр.

Выпуск под своим именем чужой программы для ЭВМ или базы данных, а также незаконное воспроизведение или распространение таких произведений влечет за собой уголовную ответственность.

В настоящее время уголовное законодательство РФ не в полной мере учитывает все возможные компьютерные преступления. Вообще же, в законодательной практике многих стран отмечены различные виды компьютерных преступлений и разработаны методы борьбы с ними.

Компьютерные преступления условно можно разделить на две большие категории:

1) преступления, связанные с вмешательством в работу компьютеров;

2) преступления, использующие компьютеры как необходимые технические средства.

Можно выделить следующие виды компьютерной преступности 1-го вида:

• несанкционированный доступ в компьютерные сети и системы, банки данных с целью шпионажа или диверсии (военного, промышленного, экономического), с целью, так называемого, компьютерного хищения или из хулиганских побуждений; ,

• ввод в программное обеспечение, так называемых, «логических бомб», срабатывающих при определенных условиях (логические бомбы, угрожающие уничтожением данных, могут использоваться для шантажа владельцев информационных систем или выполнять новые, не планировавшиеся владельцем программы, функции при сохранении работоспособности системы; известны случаи, когда программисты вводили в программы финансового учета команды, переводящие на счета этих программистов денежные суммы или скрывающие денежные суммы от учета, что позволяло незаконно получать их);

• разработку и распространение компьютерных вирусов;

• преступную небрежность в разработке, изготовлении и эксплуатации программно-вычислительных комплексов, приведшую к тяжким последствиям;

• подделку компьютерной информации (продукции) и сдачу заказчикам неработоспособных программ, подделку результатов выборов, референдумов;

•хищение компьютерной информации (нарушение авторского права и права владения программными средствами и базами данных).

Среди компьютерных преступлений 2-го вида, т.е. использующих компьютер как средство преступления, следует отметить преступления, спланированные на основе компьютерных моделей, например, в сфере бухгалтерского учета.

Для современного состояния правового регулирования сферы, связанной с информатикой, в России в настоящее время наиболее актуальными являются вопросы, связанные с нарушением авторских прав. Большая часть программного обеспечения, использующегося отдельными программистами и пользователями и целыми организациями, приобретена в результате незаконного копирования, т.е. хищения. Назрела потребность узаконить способы борьбы с этой порочной практикой, поскольку она мешает, прежде всего, развитию самой информатики.

## 1.7. ЭТИЧЕСКИЕ АСПЕКТЫ ИНФОРМАТИКИ

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

Как и в любой другой большой и разветвленной сфере человеческой деятельности, в ин-

форматике к настоящему времени сложились определенные морально-этические нормы поведения и деятельности.

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

Это - положительные аспекты, но есть и отрицательные. Обратим внимание на язык информатиков Сленг российских информатиков построен на искаженных под русское произношение англоязычных терминах и аббревиатурах, введенных иностранными фирмами - разработчиками компьютеров и программного обеспечения в технической документации. Одновременно формируется и набор сленговых слов, заимствованных из русского языка на основе аналогий и ассоциаций по сходству и смежности (например: архивированный - «утоптанный», компьютер - «железо» или «тачка» и т.д.). С тем, что многие специальные термины пришли к нам из США, приходится мириться. Никто сегодня уже не перейдет от термина «принтер» к аналогичному «автоматическое цифровое печатающее устройство» (которым пользовались не так уж давно). Приживаемости подобных слов в отечественной литературе способствует, в частности, их относительная краткость. Однако трудно понять, зачем в телеконференции учителя иногда именуют себя «тичерами» - от этого они лучше не становятся. Итак, одно из этических правил - не искажай родной язык.

Особую остроту этические проблемы приобретают при работе в глобальных телекоммуникационных сетях. Вскрыть защиту чужой базы данных – уголовное преступление. А можно ли позволять себе нецензурные выражения или прозрачные их эвфемизмы? Коммерческую рекламу в некоммерческой телеконференции? Независимо от того, предусмотрено за это законом возмездие или нет, порядочный человек этого делать не станет.

Этика - система норм нравственного поведения человека. Порядочный человек не прочтет содержимое дискеты, забытой соседом на рабочем месте, не потому, что это грозит ему наказанием, а потому, что это безнравственный поступок; не скопирует программу в отсутствие ее хозяина не потому, что на него могут подать в суд, а потому, что этот поступок осудят его коллеги. Всякий раз, собираясь совершить сомнительный поступок в сфере профессиональной деятельности, человек должен задуматься, соответствует ли он этическим нормам, сложившимся в профессиональном сообществе.

***Контрольные вопросы***

1. Что общего и в чем различие информатики и кибернетики?
2. Какие определения информатики вы знаете?
3. Какова общая структура современной информатики?
4. Какие существуют наиболее известные информационные технологии?
5. Какое место занимает информатика в системе наук?
6. Что принято понимать под «информационным обществом»?
7. Каковы основные социальные последствия информатизации общества?
8. Какими нормативными актами регулируются отношения в сфере информатики?
9. В чем состоит авторское право на программные средства и базы данных?
10. В чем состоит имущественное право на программные средства и базы данных?
11. Как осуществляется защита авторских и имущественных прав?
12. Охарактеризуйте виды компьютерных преступлений.
13. Какие этические проблемы существуют, по вашему мнению, в современной информатике?

## § 2. ИНФОРМАЦИЯ, ЕЕ ВИДЫ И СВОЙСТВА

### 2.1. РАЗЛИЧНЫЕ УРОВНИ ПРЕДСТАВЛЕНИЙ ОБ ИНФОРМАЦИИ

Ранее мы неоднократно употребляли термин «информация», никак его при этом не раскры-

вая.

Понятие **информация** является одним из фундаментальных в современной науке вообще и базовым для изучаемой нами информатики. Информацию наряду с веществом и энергией рассматривают в качестве важнейшей сущности мира, в котором мы живем. Однако, если задаться целью формально определить понятие «информация», то сделать это будет чрезвычайно сложно. Аналогичными «неопределяемыми» понятиями, например, в математике является «точка» или «прямая». Так, можно сделать некоторые утверждения, связанные с этими математическими понятиями, но сами они не могут быть определены с помощью более элементарных понятий.

В простейшем бытовом понимании с термином «информация» обычно ассоциируются некоторые сведения, данные, знания и т.п. Информация передается в виде **сообщений,** определяющих форму и представление передаваемой информации. Примерами сообщений являются музыкальное произведение; телепередача; команды регулировщика на перекрестке; текст, распечатанный на принтере; данные, полученные в результате работы составленной вами программы и т.д. При этом предполагается, что имеются «источник информации» и «получатель информации».

Сообщение от источника к получателю передается посредством какой-нибудь среды, являющейся в таком случае «каналом связи» (рис. 1.3). Так, при передаче речевого сообщения в качестве такого канала связи можно рассматривать воздух, в котором распространяются звуковые волны, а в случае передачи письменного сообщения (например, текста, распечатанного на принтере) каналом сообщения можно считать лист бумаги, на котором напечатан текст.


Рис. 1.3. Схема передачи информации

Человеку свойственно субъективное восприятие информации через некоторый набор ее свойств: важность, достоверность, своевременность, доступность и т.д. В этом смысле одно и то же сообщение, передаваемое от источника к получателю, может передавать информацию в разной степени. Так, например, вы хотите сообщить о неисправности компьютера. Для инженера из группы технического обслуживания сообщение «компьютер сломался» явно содержит больше информации, чем для вахтера. Но, в свою очередь, для инженера сообщение «не включается дисплей» содержит информации больше, чем первое, поскольку в большей степени снимает неопределенность, связанную с причиной неисправности компьютера. Как видно, одно и то же сообщение для различных пользователей несет различную информацию.

Использование терминов «больше информации» или «меньше информации» подразумевает некую возможность ее **измерения** (или хотя бы количественного соотнесения). При субъективном восприятии измерение информации возможно лишь в виде установления некоторой порядковой шкалы для оценки «больше» -«меньше», да и то субъективной, поскольку на свете немало людей, для которых, например, оба сообщения, использованных выше в качестве примера, вообще не несут никакой информации. Такое становится невозможным при введении объективных характеристик, из которых для информации важнейшей является количество. Однако, при объективном измерении количества информации следует заведомо отрешиться от восприятия ее с точки зрения субъективных свойств, примеры которых перечислены выше. Более того, не исключено, что не всякая информация будет иметь объективно измеряемое количество - все зависит от того, как будут введены единицы измерения. Не исключено и то, что при разных способах введения единиц измерения информация, содержащаяся в двух допускающих измерение сообщениях, будет по разному соотноситься.

## 2.2. НЕПРЕРЫВНАЯ И ДИСКРЕТНАЯ ИНФОРМАЦИЯ

Чтобы сообщение было передано от источника к получателю, необходима некоторая материальная субстанция - **носитель** информации. Сообщение, передаваемое с помощью носителя, назовем сигналом. В общем случае **сигнал** - это изменяющийся во времени физический процесс. Та-

кой процесс может содержать различные характеристики (например, при передаче электрических сигналов могут изменяться напряжение и сила тока). Та из характеристик, которая используется для представления сообщений, называется **параметром сигнала.**

В случае когда параметр сигнала принимает последовательное во времени конечное число значений (при этом все они могут быть пронумерованы), сигнал называется **дискретным,** а сообщение, передаваемое с помощью таких сигналов -дискретным сообщением. Информация, передаваемая источником, в этом случае также называется дискретной. Если же источник вырабатывает непрерывное сообщение (соответственно параметр сигнала - непрерывная функция от времени), соответствующая информация называется непрерывной. Пример дискретного сообщения - процесс чтения книги, информация в которой представлена текстом, т.е. дискретной последовательностью отдельных значков (букв). Примером непрерывного сообщения служит человеческая речь, передаваемая модулированной звуковой волной; параметром сигнала в этом случае является давление, создаваемое этой волной в точке нахождения приемника - человеческого уха.

Непрерывное сообщение может быть представлено непрерывной функцией, заданной на некотором отрезке [а, Ь] (см. рис. 1.4). Непрерывное сообщение можно преобразовать в дискретное (такая процедура называется **дискретизацией**). Для этого из бесконечного множества значений этой функции (параметра сигнала) выбирается их определенное число, которое приближенно может характеризовать остальные значения. Один из способов такого выбора состоит в следующем. Область определения функции разбивается точками $x_1$, $x_2$,... $x_n$, на отрезки равной длины и на каждом из этих отрезков значение функции принимается постоянным и равным, например, среднему значению на этом отрезке; полученная на этом этапе функция называется в математике ступенчатой. Следующий шаг - проецирование значений «ступенек» на ось значений функции (ось ординат). Полученная таким образом последовательность значений функции $y_1$, $y_2$, ... $y_n$. является дискретным представлением непреравной функции, точность которого можно неограниченно улучшать путем уменьшения длин отрезков разбиения области значений аргумента.

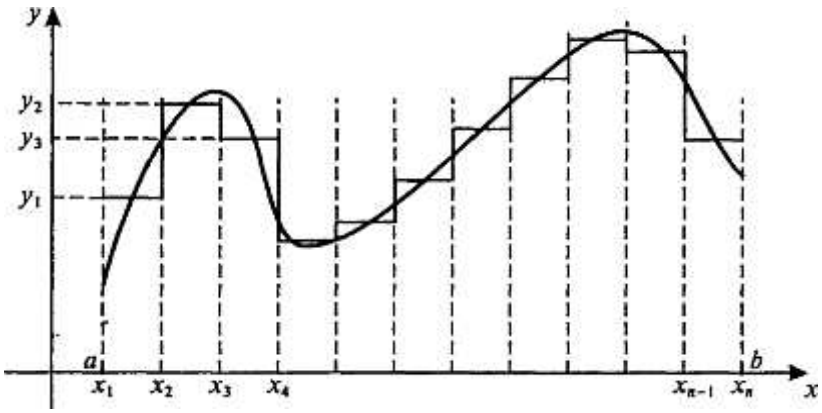



Рис. 1.4. Процедура дискретизации непрерывного сообщения

Ось значений функции можно разбить на отрезки с заданным шагом и отобразить каждый из выделенных отрезков из области определения функции в соответствующий отрезок из множества значений (рис. 1.4). В итоге получим конечное множество чисел, определяемых, например, по середине или одной из границ таких отрезков.

Таким образом, любое сообщение может быть представлено как дискретное, иначе говоря последовательностью знаков некоторого алфавита.

Возможность дискретизации непрерывного сигнала с любой желаемой точностью (для возрастания точности достаточно уменьшить шаг) принципиально важна с точки зрения информатики. Компьютер - цифровая машина, т. е- внутреннее представление информации в нем дискретно. Дискретизация входной информации (если она непрерывна) позволяет сделать ее пригодной для компьютерной обработки.

Существуют и другие вычислительные машины - аналоговые ЭВМ. Они используются обычно для решения задач специального характера и широкой публике практически не известны. Эти ЭВМ в принципе не нуждаются в дискретизации входной информации, так как ее внутреннее представление у них непрерывно. В этом случае все наоборот - если внешняя информация дискретна, то ее «перед употреблением» необходимо преобразовать в непрерывную.

## 2.3. ЕДИНИЦЫ КОЛИЧЕСТВА ИНФОРМАЦИИ:

## ВЕРОЯТНОСТНЫЙ И ОБЪЕМНЫЙ ПОДХОДЫ

Определить понятие «количество информации» довольно сложно. В решении этой проблемы существуют два основных подхода. Исторически они возникли почти одновременно. В конце 40-х годов XX века один из основоположников кибернетики американский математик Клод Шеннон развил вероятностный подход к измерению количества информации, а работы по созданию ЭВМ привели к «объемному» подходу.

### Вероятностный подход

Рассмотрим в качестве примера опыт, связанный с бросанием правильной игральной .кости, имеющей N граней (наиболее распространенным является случай шестигранной кости: N = 6). Результаты данного опыта могут быть следующие: выпадение грани с одним из следующих знаков: 1,2,... N.

Введем в рассмотрение численную величину, измеряющую неопределенность -**энтропию** (обозначим ее H). Величины N и H связаны между собой некоторой функциональной зависимостью:

$$H = f(N), \tag{1.1}$$

а сама функция *f* является возрастающей, неотрицательной и определенной (в рассматриваемом нами примере) для N = 1, 2,... 6.

Рассмотрим процедуру бросания кости более подробно:

1) готовимся бросить кость; исход опыта неизвестен, т.е. имеется некоторая неопределенность; обозначим ее H1;

2) кость брошена; информация об исходе данного опыта получена; обозначим количество этой информации через I;

3) обозначим неопределенность данного опыта после его осуществления через H2. За количество информации, которое получено в ходе осуществления опыта, примем разность неопределенностей «до» и «после» опыта:

$$I = H1 - H2 \tag{1.2}$$

Очевидно, что в случае, когда получен конкретный результат, имевшаяся неопределенность снята *(H2* = 0), и, таким образом, количество полученной информации совпадает с первоначальной энтропией. Иначе говоря, неопределенность, заключенная в опыте, совпадает с информацией об исходе этого опыта. Заметим, что значение H2 могло быть и не равным нулю, например, в случае, когда в ходе опыта следующей выпала грань со значением, большим «3».

Следующим важным моментом является определение вида функции *f* в формуле (1.1). Если варьировать число граней *N* и число бросаний кости (обозначим эту величину через *M),* общее число исходов (векторов длины M, состоящих из знаков 1,2,.... *N)* будет равно *N* в степени *M:*

$$X=N^M. \tag{1.3}$$

Так, в случае двух бросаний кости с шестью гранями имеем: $X = 6^2 = 36$. Фактически каждый исход *X* есть некоторая пара *(X1, X2),* где *X1* и *X2* - соответственно исходы первого и второго бросаний (общее число таких пар - *X).*

Ситуацию с бросанием *M* раз кости можно рассматривать как некую сложную систему, состоящую из независимых друг от друга подсистем - «однократных бросаний кости». Энтропия такой системы в *M* раз больше, чем энтропия одной системы (так называемый «принцип аддитивности энтропии»):

$$f(6^M) = M \cdot f(6)$$

Данную формулу можно распространить и на случай любого *N:*

$$F(N^M) = M \cdot f(N) \qquad (1.4)$$

Прологарифмируем левую и правую части формулы (1.3): ln $X$ = $M$ · ln $N$, $M$ = ln $X$/1n $M$. Подставляем полученное для $M$ значение в формулу (1.4):

$$f(x) = \frac{\ln X}{\ln M} \cdot f(N).$$

Обозначив через $K$ положительную константу , получим: $f(X) = K \cdot ln\ X$, или, с учетом (1.1), $H=K \cdot \ln N$. Обычно принимают $K$ = 1 / ln 2. Таким образом

$$H = \log_2 N. \qquad (1.5)$$

Это - **формула Хартли.**

Важным при введение какой-либо величины является вопрос о том, что принимать за единицу ее измерения. Очевидно, $H$ будет равно единице при $N = 2$. Иначе говоря, в качестве единицы принимается количество информации, связанное с проведением опыта, состоящего в получении одного из двух равновероятных исходов (примером такого опыта может служить бросание монеты при котором возможны два исхода: «орел», «решка»). Такая единица количества информации называется «бит».

Все $N$ исходов рассмотренного выше опыта являются равновероятными и поэтому можно считать, что на «долю» каждого исхода приходится одна $N$-я часть общей неопределенности опыта: ($\log_2 N$)1$N$. При этом вероятность $i$-го исхода $P_i$ равняется, очевидно, 1/$N$.

Таким образом,

$$H = \sum_{i=1}^{N} P_i \cdot \log_2\left(\frac{1}{P_i}\right).$$

Та же формула (1.6) принимается за меру энтропии в случае, когда вероятности различных исходов опыта **неравновероятны** (т.е. $P_i$ могут быть различны). Формула (1.6) называется **формулой Шеннона.**

В качестве примера определим количество информации, связанное с появлением каждого символа в сообщениях, записанных на русском языке. Будем считать, что русский алфавит состоит из 33 букв и знака «пробел» для разделения слов. По формуле (1.5)

$$H = \log_2 34 \approx 5 \text{ бит}.$$

Однако, в словах русского языка (равно как и в словах других языков) различные буквы встречаются неодинаково часто. Ниже приведена табл. 1.3 вероятностей частоты употребления различных знаков русского алфавита, полученная на основе анализа очень больших по объему текстов.

Воспользуемся для подсчета $H$ формулой (1.6); $H \approx 4{,}72$ бит. Полученное значение $H$, как и можно было предположить, меньше вычисленного ранее. Величина $H$, вычисляемая по формуле (1.5), является максимальным количеством информации, которое могло бы приходиться на один знак.

**Таблица 1.3. Частотность букв русского языка**

| i | Символ | P($i$) | $i$ | Символ | P($i$) | $i$ | Символ | P($i$) |
|---|---|---|---|---|---|---|---|---|
| 1 | Пробел | 0,175 | 13 | | 0,028 | 24 | Г | 0.012 |
| 2 | 0 | 0,090 | 14 | М | 0,026 | 25 | Ч | 0,012 |
| 3 | Е | 0,072 | 15 | Д | 0,025 | 26 | И | 0,010 |
| 4 | Ё | 0,072 | 16 | П | 0,023 | 27 | Х | 0,009 |
| 5 | А | 0,062 | 17 | У | 0,021 | 28 | Ж | 0,007 |
| 6 | И | 0,062 | 18 | Я | 0,018 | 29 | Ю | 0,006 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7 | Т | 0,053 | 19 | Ы | 0,016 | 30 | Ш | 0.006 |
| 8 | Н | 0,053 | 20 | З | 0.016 | 31 | Ц | 0,004 |
| 9 | С | 0,045 | 21 | Ь | 0,014 | 32 | Щ | 0,003 |
| 10 | Р | 0,040 | 22 | Ъ | 0,014 | 33 | Э | 0,003 |
| 11 | В | 0,038 | 23 | Б | 0,014 | 34 | Ф | 0,002 |
| 12 | Л | 0,035 | | | | | | |

Аналогичные подсчеты $H$ можно провести и для других языков, например, использующих латинский алфавит - английского, немецкого, французского и др. (26 различных букв и «пробел»). По формуле (1.5) получим

$$H = \log_2 27 \approx 4{,}76 \text{ бит.}$$

Как и в случае русского языка, частота появления тех или иных знаков не одинакова.

Если расположить все буквы данных языков в порядке убывания вероятностей, то получим следующие последовательности:

АНГЛИЙСКИЙ ЯЗЫК: «пробел», E, T, A, O, N, R, …
НЕМЕЦКИЙ ЯЗЫК: «пробел», E, N, I, S, T, R, …
ФРАНЦУЗСКИЙ ЯЗЫК: «пробел», E, S, A, N, I, T, …

Рассмотрим алфавит, состоящий из двух знаков 0 и 1. Если считать, что со знаками 0 и 1 в двоичном алфавите связаны одинаковые вероятности их появления *(P(0) = P(1)* = 0,5), то количество информации на один знак при двоичном кодировании будет равно

$$H = 1\log_2 2 = 1 \text{ бит.}$$

Таким образом, количество информации (в битах), заключенное в двоичном слове, равно числу двоичных знаков в нем.

### Объемный подход

В двоичной системе счисления знаки 0 и 1 будем называть **битами** (от английского выражения Binary digiTs - двоичные цифры). Отметим, что создатели компьютеров отдают предпочтение именно двоичной системе счисления потому, что в техническом устройстве наиболее просто реализовать два противоположных физических состояния: некоторый физический элемент, имеющий два различных состояния: намагниченность в двух противоположных направлениях; прибор, пропускающий или нет электрический ток; конденсатор, заряженный или незаряженный и т.п. В компьютере бит является наименьшей возможной единицей информации. Объем информации, записанной двоичными знаками в памяти компьютера или на внешнем носителе информации подсчитывается просто по количеству требуемых для такой записи двоичных символов. При этом, в частности, невозможно нецелое число битов (в отличие от вероятностного подхода).

Для удобства использования введены и более крупные, чем бит, единицы количества информации. Так, двоичное слово из восьми знаков содержит один, **байт информации,** 1024 байта образуют **килобайт** (кбайт), 1024 килобайта - **мегабайт** (Мбайт), а 1024 мегабайта - **гигабайт** (Гбайт).

Между вероятностным и объемным количеством информации соотношение неоднозначное. Далеко не всякий текст, записанный двоичными символами, допускает измерение объема информации в кибернетическом смысле, но заведомо допускает его в объемном. Далее, если некоторое сообщение допускает измеримость количества информации в обоих смыслах, то они не обязательно совпадают, при этом кибернетическое количество информации не может быть больше объемного.

В дальнейшем тексте данного учебника практически всегда количество информации понимается в объемном смысле.

## 2.4. ИНФОРМАЦИЯ: БОЛЕЕ ШИРОКИЙ ВЗГЛЯД

Как ни важно измерение информации, нельзя сводить к нему все связанные с этим понятия-

ем проблемы. При анализе информации социального (в широким смысле) происхождения на первый план могут выступить такие ее свойства. как истинность, своевременность, ценность, полнота и т.д. Их невозможно оценить в терминах «уменьшение неопределенности» (вероятностный подход) или числа символов (объемный подход). Обращение к качественной стороне информации породило иные подходы к ее оценке. При **аксиологическом** подходе стремятся исходить из ценности, практической значимости информации, т.е. качественных характеристик, значимых в социальной системе. При **семантическом** подходе информация рассматривается с точки зрения как формы, так и содержания. При этом информацию связывают с **тезаурусом,** т.е. полнотой систематизированного набора данных о предмете информации. Отметим, что эти подходы не исключают количественного анализа, но он становится существенно сложнее и должен базироваться на современных методах математической статистики.

Понятие информации нельзя считать лишь техническим,.междисциплинарным и даже наддисциплинарным термином. Информация - это фундаментальная философская категория. Дискуссии ученых о философских аспектах информации надежно показали несводимость информации ни к одной из этих категорий. Концепции и толкования, возникающие на пути догматических подходов, оказываются слишком частными, односторонними, не охватывающими всего объема этого понятия.

Попытки рассмотреть категорию информации с позиций основного вопроса философии привели к возникновению двух противостоящих концепций - так называемых, **функциональной и атрибутивной.** «Атрибутисты» квалифицируют информацию как свойство всех материальных объектов, т.е. как атрибут материи. «функционалисты» связывают информацию лишь с функционированием сложных, самоорганизующихся систем. Оба подхода, скорее всего, неполны. Дело в том, что природа сознания, духа по сути своей является информационной, т.е. создание суть менее общее понятие по отношению к категории «информация». Нельзя признать корректными попытки сведения более общего понятия к менее общему. Таким образом, информация и информационные процессы, если иметь в виду решение основного вопроса философии, опосредуют материальное и духовное, т.е. вместо классической постановки этого вопроса получается два новых: о соотношении материи и информации и о соотношении информации и сознания (духа);

Можно попытаться дать философское определение информации с помощью указания на связь определяемого понятия с категориями **отражения и активности.** Информация есть содержание образа, формируемого в процессе отражения. Активность входит в это определение в виде представления о формировании некоего образа в процессе отражения некоторого субъект-объектного отношения. При этом не требуется указания на связь информации с материей, поскольку как субъект, так и объект процесса отражения могут принадлежать как к материальной, так и к духовной сфере социальной жизни. Однако существенно подчеркнуть, что материалистическое решение основного вопроса философии требует признания необходимости существования материальной среды - носителя информации в процессе такого отражения. Итак, информацию следует трактовать как имманентный (неотъемлемо присущий) атрибут материи, необходимый момент ее самодвижения и саморазвития. Эта категория приобретает особое значение применительно к высшим формам движения материи - биологической и социальной.

Данное выше определение схватывает важнейшие характеристики информации. Оно не противоречит тем знаниям, которые накоплены по этой проблематике, а наоборот, является выражением наиболее значимых.

Современная практика психологии, социологии, информатики диктует необходимость перехода к информационной трактовке сознания. Такая, трактовка оказывается чрезвычайно плодотворной и позволяет, например, рассмотреть с общих позиций индивидуальное и общественное сознание. Генетически индивидуальное и общественное сознание неразрывны и в то же время общественное сознание не есть простая сумма индивидуальных, поскольку оно включает информационные потоки и процессы между индивидуальными сознаниями.

В социальном плане человеческая деятельность предстает как взаимодействие реальных человеческих коммуникаций с предметами материального мира. Поступившая извне к человеку информация является отпечатком, снимком сущностных сил природы или другого человека. Таким образом, с единых методологических позиций может быть рассмотрена деятельность индивидуального и общественного сознания, экономическая, политическая, образовательная деятельность различных субъектов социальной системы.

Данное выше определение информации как философской категории не только затрагивает физические аспекты существования информации, но и фиксирует ее социальную значимость.

Одной из важнейших черт функционирования современного общества выступает его информационная оснащенность. В ходе своего развития человеческое общество прошло через пять информационных революций. Первая из них была связана с введением языка, вторая - письменности, третья - книгопечатания, четвертая -телесвязи, и, наконец, пятая - компьютеров (а также магнитных и оптических носителей хранения информации). Каждый раз новые информационные технологии поднимали информированность общества на несколько порядков, радикально меняя объем и глубину знания, а вместе с этим и уровень культуры в целом.

Одна из целей философского анализа понятия информации - указать место информационных технологий в развитии форм движения материи, в прогрессе человечества и, в том числе, в развитии разума как высшей отражательной способности материи. На протяжении десятков тысяч лет сфера разума развивалась исключительно через общественную форму сознания. С появлением компьютеров начались разработки систем искусственного интеллекта, идущих по пути моделирования общих интеллектуальных функций индивидуального сознания.

## 2.5. ИНФОРМАЦИЯ И ФИЗИЧЕСКИЙ МИР

Известно большое количество работ, посвященных физической трактовке информации. Эти работы в значительной мере построены на основе аналогии формулы Больцмана, описывающей энтропию статистической системы материальных частиц, и формулы Хартли.

Заметим, что при всех выводах формулы Больцмана явно или неявно предполагается, что макроскопическое состояние системы, к которому относится функция энтропии, реализуется на микроскопическом уровне как сочетание механических состояний очень большого числа частиц, образующих систему (молекул). Задачи же кодирования и передачи информации, для решения которых Хартли и Шенноном была развита вероятностная мера информации, имели в виду очень узкое техническое понимание информации, почти не имеющее отношения к полному объему этого понятия. Таким образом, большинство рассуждений, использующих термодинамические свойства энтропии применительно к информации нашей реальности, носят спекулятивный характер. В частности, являются необоснованными использование понятия «энтропия» для систем с конечным и небольшим числом состояний, а также попытки расширительного методологического толкования .результатов теории вне довольно примитивных механических моделей, для которых они были получены. Энтропия и негэнтропия - интегральные характеристики протекания стохастических процессов - лишь параллельны информации и превращаются в нее в частном случае.

Информацию следует считать особым видом ресурса, при этом имеется ввиду толкование «ресурса» как запаса неких знаний материальных предметов или энергетических, структурных или каких-либо других характеристик предмета. В отличие от ресурсов, связанных с материальными предметами, информационные ресурсы являются неистощимыми и предполагают существенно иные методы воспроизведения и обновления, чем материальные ресурсы.

Рассмотрим некоторый набор свойств информации:

- запоминаемость;
- передаваемость;
- преобразуемость;
- воспроизводимость;
- стираемость.

Свойство **запоминаемости** - одно из самых важных. Запоминаемую информацию будем называть макроскопической (имея ввиду пространственные масштабы запоминающей ячейки и время запоминания). Именно с макроскопической информацией мы имеем дело в реальной практике.

**Передаваемость** информации с помощью каналов связи (в том числе с помехами) хорошо исследована в рамках теории информации К. Шеннона. В данном случае имеется ввиду несколько иной аспект - способность информации к копированию, т.е. к тому, что она может быть «запомнена» другой макроскопической системой и при этом останется тождественной самой себе. Очевидно, что количество информации не должно возрастать при копировании.

**Воспроизводимость** информации тесно связана с ее передаваемостью и не является ее независимым базовым свойством. Если передаваемость означает, что не следует считать существен-

ными пространственные отношения между частями системы, между которыми передается информация, то воспроизводимость характеризует неиссякаемость и неистощимость информации, т.е. что при копировании информация остается тождественной самой себе.

Фундаментальное свойство информации - **преобразуемость.** Оно означает, что информация может менять способ и форму своего существования. Копируемость есть разновидность преобразования информации, при котором ее количество не меняется. В общем случае количество информации в процессах преобразования меняется, но возрастать не может. Свойство **стираемости** информации также не является независимым. Оно связано с таким преобразованием информации (передачей), при котором ее количество уменьшается и становится равным нулю.

Данных свойств информации недостаточно для формирования ее меры, так как они относятся к физическому уровню информационных процессов.

Подводя итог сказанному в п. 2.4 - 2.5, отметим, что предпринимаются (но отнюдь не завершены) усилия ученых, представляющих самые разные области знания, построить единую теорию, которая призвана формализовать понятие информации и информа-ционного процесса, описать превращения информации в процессах самой разной природы. Движение информации есть сущность процессов управления, которые суть проявление имманентной активности материи, ее способности к самодвижению. С момента возникновения кибернетики управление рассматривается применительно ко всем формам движения материи, а не только к высшим (биологической и социальной). Многие проявления движения в неживых - искусственных (технических) и естественных (природных) - системах также обладают общими признаками управления, хотя их исследуют в химии, физике, механике в энергетической, а не в информационной системе представлений. Информационные аспекты в таких системах составляют предмет новой междисциплинарной науки - синергетики.

Высшей формой информации, проявляющейся в управлении в социальных системах, являются знания. Это наддисциплинарное понятие, широко используемое в педагогике и исследованиях по искусственному интеллекту, также претендует на роль важнейшей философской категории. В философском плане познание следует рассматривать как один из функциональных аспектов управления. Такой подход открывает путь к системному пониманию генезиса процессов познания, его основ и перспектив.

***Контрольные вопросы***

1. Какая форма представления информации - непрерывная или дискретная -приемлема для компьютеров и почему?
2. В чем состоит процедура дискретизации непрерывной информации?
3. Как определяется понятие энтропии?
4. Каким образом определяется единица количества информации при кибернетическом подходе?
5. Каковы особенности определения количества информации, связанной с появлением различных знаков в сообщениях?

## § 3. СИСТЕМЫ СЧИСЛЕНИЯ

### 3.1. ПОЗИЦИОННЫЕ СИСТЕМЫ СЧИСЛЕНИЯ

**Система счисления** - принятый способ записи чисел и сопоставления этим записям реальных значений. Все системы счисления можно разделить на два класса: **позиционные и непозиционные.** Для записи чисел в различных системах счисления используется некоторое количество отличных друг от друга знаков. Число таких знаков в позиционной системе счисления называется **основанием системы счисления.** Ниже приведена табл. 1.4, содержащая наименования некоторых позиционных систем счисления и перечень знаков (цифр), из которых образуются в них числа.

**Таблица 1.4. Некоторые системы счисления**

| Основание | Система счисления | Знаки |
|---|---|---|
| 2 | Двоичная | 0,1 |
| 3 | Троичная | 0,1.2 |
| 4 | Четвертичная | 0,1,2,3 |
| 5 | Пятиричная | 0,1,2,3,4 |
| 8 | Восьмиричная | 0,1,2,3,4,5,6,7 |
| 10 | Десятичная | 0,1,2,3,4,5,6,7,8,9 |
| 12 | Двенадцатиричная | 0,1,2,3,4,5,6,7,8,9,A,B |
| 16 | Шестнадцатиричная | 0,1,2,3,4,5,6,7,8,9,A.B,D,E,F |

В позиционной системе счисления число может быть представлено в виде суммы произведений коэффициентов на степени основания системы счисления:

$$A_nA_{n-1}A_{n-2} \dots A_1,A_0,A_{-1},A_{-2} =$$
$$A_nB^n + A_{n-1}B^{n-1} + ... + A_1B^1 + A_0B^0 + A_{-1}B^{-1} + A_{-2}B^{-2} + ...$$

(знак «точка» отделяет целую часть числа от дробной; знак «звездочка» здесь и ниже используется для обозначения операции умножения). Таким образом, значение каждого знака в числе зависит от позиции, которую занимает знак в записи числа. Именно поэтому такие системы счисления называют позиционными. Примеры (десятичный индекс внизу указывает основание системы счисления):

$$23,43_{(10)} = 2*10^1 + 3*10^\circ + 4*10^{-1} + 3*10^{-2}$$

(в данном примере знак «3» в одном случае означает число единиц, а в другом - число сотых долей единицы);

$$692_{(10)} = 6*\ 10^2 + 9*10^1 + 2.$$

(«Шестьсот девяносто два» с формальной точки зрения представляется в виде «шесть умножить на десять в степени два, плюс девять умножить на десять в степени один, плюс два»).

$$1101_{(2)} = 1*2^3 + 1*2^2 + 0*2^1 + 1*2^\circ;$$
$$112_{(3)} = 1*3^2 + 1*3^1 + 2*3^\circ;$$
$$341,5_{(8)} = 3*8^2 + 4*8^1 + 1*8^\circ + 5*8^{-1};$$
$$A1F4_{(16)} = A*16^2 + 1*16^1 + F*16^\circ + 4*16^{-1}.$$

При работе с компьютерами приходится параллельно использовать несколько позиционных систем счисления (чаще всего двоичную, десятичную и шестнадцатиричную), поэтому большое практическое значение имеют процедуры перевода чисел из одной системы счисления в другую. Заметим**,** что во всех приведенных выше примерах результат является десятичным числом, и, таким образом, способ перевода чисел из любой позиционной системы счисления в десятичную уже продемонстрирован.

Чтобы перевести целую часть числа из десятичной системы в систему с основанием *B,* необходимо разделить ее на *B.* Остаток даст младший разряд числа. Полученное при этом частное необходимо вновь разделить на *B* - остаток даст следующий разряд числа и т.д. Для перевода дробной части ее необходимо умножить на *B.* Целая часть полученного произведения будет первым (после запятой, отделяющей целую часть от дробной) знаком. Дробную же часть произведения необходимо вновь умножить на *B.* Целая часть полученного числа будет следующим знаком и т.д.

Отметим, что кроме рассмотренных выше позиционных систем счисления существуют такие, в которых значение знака не зависит от того места, которое он занимает в числе. Такие систе-

мы счисления называются непозиционными. Наиболее известным примером непозиционной системы является римская. В этой системе используется 7 знаков (I, V, X, L, C, D, M), которые соответствуют следующим величинам:

1(1) V(5) X(10) L(50) C (100) D(500) M(1000)

*Примеры:* III (три), LIX (пятьдесят девять), DLV (пятьсот пятьдесят пять).

Недостатком непозиционных систем, из-за которых они представляют лишь исторический интерес, является отсутствие формальных правил записи чисел и, соответственно, арифметических действий над ними (хотя по традиции римскими числами часто пользуются при нумерации глав в книгах, веков в истории и др.).

## 3.2. ДВОИЧНАЯ СИСТЕМА СЧИСЛЕНИЯ

Особая значимость двоичной системы счисления в информатике определяется тем, что внутреннее представление любой информации в компьютере является двоичным, т.е. описываемым наборами только из двух знаков (0 и 1).

Конкретизируем описанный выше способ в случае перевода чисел из десятичной системы в двоичную. Целая и дробная части переводятся порознь. Для перевода целой части (или просто целого) числа необходимо разделить ее на основание системы счисления и продолжать делить частные от деления до тех пор, пока частное не станет равным 0. Значения получившихся остатков, взятые в обратной последовательности, образуют искомое двоичное число. Например:

| | Остаток |
|---|---|
| 25 : 2 = 12 | (1), |
| 12 : 2 = 6 | (0), |
| 6 : 2 = 3 | (0), |
| 3 : 2 = 1 | (1), |
| 1 : 2 = 0 | (1). |

Таким образом

$$25_{(10)}=11001_{(2)}.$$

Для перевода дробной части (или числа, у которого «0» целых) надо умножить ее на 2. Целая часть произведения будет первой цифрой числа в двоичной системе. Затем, отбрасывая у результата целую часть, вновь умножаем на 2 и т.д. Заметим, что конечная десятичная дробь при этом вполне может стать бесконечной {периодической) двоичной. Например:

0,73 • 2 = 1,46 (целая часть 1),
0,46 • 2 = 0,92 (целая часть 0 ),
0,92 • 2 = 1,84 (целая часть 1),
0,84 • 2 = 1,68 (целая часть 1) и т.д.

В итоге

$$0,73_{(10)} =0,1011...{}_{(2)}.$$

Над числами, записанными в любой системе счисления, можно; производить различные арифметические операции. Так, для сложения и умножения двоичных чисел необходимо использовать табл. 1.5.

**Таблица 1.5. Таблицы сложения и умножения в двоичной системе**

| + | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 10 |

| * | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Заметим, что при двоичном сложении 1 + 1 возникает перенос единицы в старший разряд - точь-в-точь как в десятичной арифметике:

```
  1001
+   11
------
= 1100
```

```
  1001
*   11
------
  1001
+ 1001
------
= 10101
```

## 3.3. ВОСЬМЕРИЧНАЯ И ШЕСТНАДЦАТИРИЧНАЯ СИСТЕМЫ СЧИСЛЕНИЯ

С точки зрения изучения принципов представления и обработки информации в компьютере, обсуждаемые в этом пункте системы представляют большой интерес.

Хотя компьютер «знает» только двоичную систему счисления, часто с целью уменьшения количества записываемых на бумаге или вводимых с клавиатуры компьютера знаков бывает удобнее пользоваться восьмеричными или шестнадцатиричными числами, тем более что, как будет показано далее, процедура взаимного перевода чисел из каждой из этих систем в двоичную очень проста - гораздо проще переводов между любой из этих трех систем и десятичной.

Перевод чисел из десятичной системы счисления в восьмеричную производится (по аналогии с двоичной системой счисления) с помощью делений и умножений на 8. Например, переведем число $58{,}32_{(10)}$:

58 : 8 = 7 (2 в остатке),
7 : 8 = 0 (7 в остатке).
0,32 • 8 = 2,56,
0,56 • 8 = 4,48,
0,48-8=3,84,...

Таким образом,

$58{,}32_{(10)} = 72{,}243\ldots_{(8)}$

(из конечной дроби в одной системе может получиться бесконечная дробь в другой).

Перевод чисел из десятичной системы счисления в шестнадцатеричную производится аналогично.

С практической точки зрения представляет интерес процедура взаимного преобразования двоичных, восьмеричных и шестнадцатиричных чисел. Для этого воспользуемся табл. 1.6 чисел от 0 до 15 (в десятичной системе счисления), представленных в других системах счисления.

Для перевода целого двоичного числа в восьмеричное необходимо разбить его справа налево на группы по 3 цифры (самая левая группа может содержать менее трех двоичных цифр), а затем каждой группе поставить в соответствие ее восьмеричный эквивалент. Например:

11011001= 11011001, т.е. $11011001_{(2)} = 331_{(8)}$.

Заметим, что группу из трех двоичных цифр часто называют «двоичной триадой».

Перевод целого двоичного числа в шестнадцатиричное производится путем разбиения данного числа на группы по 4 цифры - «двоичные тетрады»:

$1100011011001 = 1\ 1000\ 1101\ 1001$, т.е. $1100011011001_{(2)} = 18D9_{(16)}$.

Для перевода дробных частей двоичных чисел в восьмеричную или шестнадцатиричную системы аналогичное разбиение на триады или тетрады производится от точки вправо (с дополнением недостающих последних цифр нулями):

$0{,}1100011101_{(2)} = 0{,}110\ 001\ 110\ 100 = 0{,}6164_{(8)}$,
$0{,}1100011101_{(2)} = 0{,}1100\ 0111\ 0100 = 0{,}C74_{(16)}$.

Перевод восьмеричных (шестнадцатиричных) чисел в двоичные производится обратным путем - сопоставлением каждому знаку числа соответствующей тройки (четверки) двоичных цифр.

Таблица 1.6 Соответствие чисел в различных системах счисления

| Десятичная | Шестнадцатиричная | Восьмеричная | Двоичная |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 10 |
| 3 | 3 | 3 | 11 |
| 4 | 4 | 4 | 100 |
| 5 | 5 | 5 | 101 |
| 6 | 6 | 6 | 110 |
| 7 | 7 | 7 | 111 |
| 8 | 8 | 10 | 1000 |
| 9 | 9 | 11 | 1001 |
| 10 | A | 12 | 1010 |
| 11 | B | 13 | L011 |
| 12 | C | 14 | 1100 |
| 13 | D | 15 | 1101 |
| 14 | E | 16 | 1110 |
| 15 | F | 17 | 1111 |

Преобразования чисел из двоичной в восьмеричную и шестнадцатиричную системы и наоборот столь просты (по сравнению с операциями между этими тремя системами и привычной нам десятичной) потому, что числа 8 и 16 являются целыми степенями числа 2. Этой простотой и объясняется популярность восьмеричной и шестнадцатиричной систем в вычислительной технике и программировании.

Арифметические действия с числами в восьмеричной и шестнадцатиричной системах счисления выполняются по аналогии с двоичной и десятичной системами. Для этого необходимо воспользоваться соответствующими таблицами. Для примера табл. 1.7 иллюстрирует сложение и умножение восьмеричных чисел.

Рассмотрим еще один возможный способ перевода чисел из одной позиционной системы счисления в другую - *метод вычитания степеней.* В этом случае из числа последовательно вычитается максимально допустимая степень требуемого основания, умноженная на максимально возможный коэффициент, меньший основания; этот коэффициент и является значащей цифрой числа в новой системе. Например, число $114_{(10)}$:

$114 - 2^6 = 114 - 64 = 50$,
$50 - 2^5 = 50 - 32 = 18$,
$18 - 2^4 = 2$,
$2 - 2^1 = 0$.

Таким образом, $114_{(10)} = 1110010_{(2)}$.

$114 - 1 \cdot 8^2 = 114 - 64 = 50,$
$50 - 6 \cdot 8^1 = 50 - 48 = 2,$
$2 - 2 \cdot 8^\circ = 2 - 2 = 0.$

Итак, $114_{(10)} = 162_{(8)}$.

**Таблица 1.7 Таблицы сложения и умножения в восьмеричной системе**

Сложение

| + | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 |
| 2 | 2 | 3 | 4 | 5 | 6 | 7 | 10 | 11 |
| 3 | 3 | 4 | 5 | 6 | 7 | 10 | 11 | 12 |
| 4 | 4 | 5 | 6 | 7 | 10 | 11 | 12 | 13 |
| 5 | 5 | 6 | 7 | 10 | 11 | 12 | 13 | 14 |
| 6 | 6 | 7 | 10 | 11 | 12 | 13 | 14 | 15 |
| 7 | 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |

Умножение

| * | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 | 0 | 2 | 4 | 6 | 10 | 12 | 14 | 16 |
| 3 | 0 | 3 | 6 | 11 | 14 | 17 | 22 | 25 |
| 4 | 0 | 4 | 10 | 14 | 20 | 24 | 30 | 34 |
| 5 | 0 | 5 | 12 | 17 | 24 | 31 | 36 | 43 |
| 6 | 0 | 6 | 14 | 22 | 30 | 36 | 44 | 52 |
| 7 | 0 | 7 | 16 | 25 | 34 | 43 | 52 | 61 |

***Контрольные вопросы***

1. В чем отличие позиционной системы счисления от непозиционной?
2. Каковы способы перевода чисел из одной системы счисления в другую?
3. В чем заключается преимущество использования восьмеричной и шестнадцатиричной систем счисления в вычислительной технике?
4. Как выглядят таблицы сложения и умножения в шестнадцатиричной системе?

# § 4. КОДИРОВАНИЕ ИНФОРМАЦИИ.

## 4.1. АБСТРАКТНЫЙ АЛФАВИТ

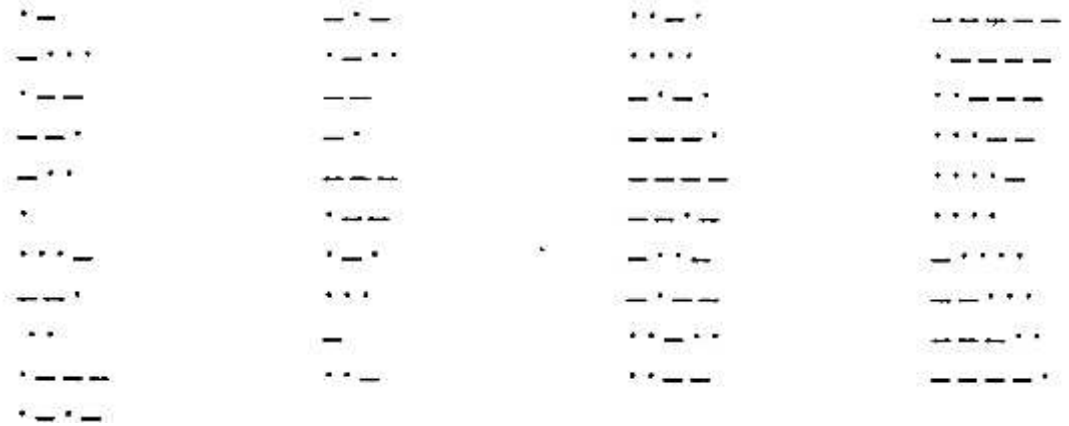
Информация передается в виде сообщений. Дискретная информация записывается с помощью некоторого конечного набора знаков, которые будем называть буквами, не вкладывая в это слово привычного ограниченного значения (типа «русские буквы» или «латинские буквы»). Буква в данном расширенном понимании - любой из знаков, которые некоторым соглашением установлены для общения. Например, при привычной передаче сообщений на русском языке такими знаками будут русские буквы - прописные и строчные, знаки препинания, пробел; если в тексте есть числа - то и цифры. Вообще, буквой будем называть элемент некоторого конечного множества (набора) отличных друг от друга знаков. Множество знаков, в котором определен их порядок, назовем алфавитом (общеизвестен порядок знаков в русском алфавите: А, Б,..., Я).

Рассмотрим некоторые примеры алфавитов.

1, Алфавит прописных русских букв:

А Б В Г Д Е Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ъ Ы Ь Э Ю Я

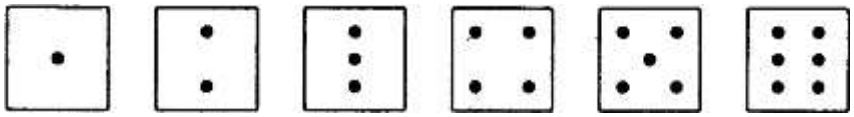
2. Алфавит Морзе:



3. Алфавит клавиатурных символов ПЭВМ IBM (русифицированная клавиатура):

` 1 2 3 4 5 6 7 8 9 0 - =
~ ! @ # $ % ^ & * ( ) _ +
q w e r t y u i o p [ ]
Q W E R T Y U I O P { }
a s d f g h j k l ; : "
A S D F G H J K L : « »
z x c v b n m , . /
Z X C V B N M < > ?
й ц у к е н г ш щ з х ъ
Й Ц У К Е Н Г Ш Щ З Х Ъ
ф ы в а п р о л д ж э
Ф Ы В А П Р О Л Д Ж Э
я ч с м и т ь б ю
Я Ч С М И Т Ь Б Ю

4. Алфавит знаков правильной шестигранной игральной кости:



5. Алфавит арабских цифр:

0 1 2 3 4 5 6 7 8 9

6. Алфавит шестнадцатиричных цифр:

0 1 2 3 4 5 6 7 8 9 A B C D E F

Этот пример, в частности, показывает, что знаки одного алфавита могут образовываться из знаков других алфавитов.

7. Алфавит двоичных цифр:

0 1

Алфавит 7 является одним из примеров, так называемых, «двоичных» алфавитов, т.е. алфавитов, состоящих из двух знаков. Другими примерами являются двоичные алфавиты 8 и 9:

8. Двоичный алфавит «точка, «тире»:. _

9. Двоичный алфавит «плюс», «минус»: + -

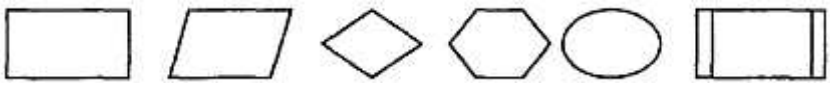
10. Алфавит прописных латинских букв:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

11. Алфавит римской системы счисления:

I V X L C D M

12. Алфавит языка блок-схем изображения алгоритмов:



13. Алфавит языка программирования Паскаль (см. в главе 3).

## 4.2. КОДИРОВАНИЕ И ДЕКОДИРОВАНИЕ

В канале связи сообщение, составленное из символов (букв) одного алфавита, может пре-

образовываться в сообщение из символов (букв) другого алфавита. Правило, описывающее однозначное соответствие букв алфавитов при таком преобразовании, называют кодом. Саму процедуру преобразования сообщения называют перекодировкой. Подобное преобразование сообщения может осуществляться в момент поступления сообщения от источника в канал связи (кодирование) и в момент приема сообщения получателем (декодирование). Устройства, обеспечивающие кодирование и декодирование, будем называть соответственно кодировщиком и декодировщиком. На рис. 1.5 приведена схема, иллюстрирующая процесс передачи сообщения в случае перекодировки, а также воздействия помех (см. следующий пункт).


Рис. 1.5. Процесс передачи сообщения от источника к приемнику

Рассмотрим некоторые примеры кодов.

1. Азбука Морзе в русском варианте (алфавиту, составленному из алфавита русских заглавных букв и алфавита арабских цифр ставится в соответствие алфавит Морзе):

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| А | ·– | К | –·– | Ф | ··–· | 0 | ––––– |
| Б | –··· | Л | ·–·· | Х | ···· | 1 | ·–––– |
| В | ·–– | М | –– | Ц | –·–· | 2 | ··––– |
| Г | ––· | Н | –· | Ч | ––––· | 3 | ···–– |
| Д | –·· | О | ––– | Ш | –––– | 4 | ····– |
| Е | · | П | ·––· | Щ | ––·– | 5 | ····· |
| Ж | ···– | Р | ·–· | Ь, Ъ | –··– | 6 | –···· |
| З | ––·· | С | ··· | Ы | –·–– | 7 | ––··· |
| И | ·· | Т | – | Э | ··–·· | 8 | –––·· |
| Й | ·––– | У | ··– | Ю | ··–– | 9 | ––––· |
| | | Я | ·–·– | | | | |

2. Код Трисиме (знакам латинского алфавита ставятся в соответствие комбинации из трех знаков: 1,2,3):

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| A 111 | D 121 | G 131 | J211 | M221 | P231 | S311 | V321 | Y331 |
| B 112 | E 122 | H 132 | K212 | N222 | Q232 | T312 | W322 | Z332 |
| C 113 | F 123 | I 133 | L213 | O223 | R233 | U313 | X323 | .333 |

Код Трисиме является примером, так называемого, равномерного кода (такого, в котором все кодовые комбинации содержат одинаковое число знаков - в данном случае три). Пример неравномерного кода - азбука Морзе.

3. Кодирование чисел знаками различных систем счисления см. §3.

## 4.3. ПОНЯТИЕ О ТЕОРЕМАХ ШЕННОНА

Ранее отмечалось, что при передаче сообщений по каналам связи могут возникать помехи, способные привести к искажению принимаемых знаков. Так, например, если вы попытаетесь в ветреную погоду передать речевое сообщению человеку, находящемуся от вас на значительном расстоянии, то оно может быть сильно искажено такой помехой, как ветер. Вообще, передача сообщений при наличии помех является серьезной теоретической и практической задачей. Ее значимость возрастает в связи с повсеместным внедрением компьютерных телекоммуникаций, в которых помехи неизбежны. При работе с кодированной информацией, искажаемой помехами, можно выделить следующие основные проблемы: установления самого факта того, что произошло искажение информации; выяснения того, в каком конкретно месте передаваемого текста это произошло; исправления ошибки, хотя бы с некоторой степенью достоверности.

Помехи в передачи информации - вполне обычное дело во всех сферах профессиональной деятельности и в быту. Один из примеров был приведен выше, другие примеры - разговор по телефону, в трубке которого «трещит», вождение автомобиля в тумане и т.д. Чаще всего человек вполне справляется с каждой из указанных выше задач, хотя и не всегда отдает себе отчет, как он это делает (т.е. неалгоритмически, а исходя из каких-то ассоциативных связей). Известно, что ес-

тественный язык обладает большой **избыточностью** (в европейских языках - до 7%), чем объясняется большая помехоустойчивость сообщений, составленных из знаков алфавитов таких языков. Примером, иллюстрирующим устойчивость русского языка к помехам, может служить предложение «в словох всо глосноо зомононо боквой о». Здесь 26% символов «поражены», однако это не приводит к потере смысла. Таким образом, в данном случае избыточность является полезным свойством.

Избыточность могла бы быть использована и при передаче кодированных сообщений в технических системах. Например, каждый фрагмент текста («предложение») передается трижды, и верным считается та пара фрагментов, которая полностью совпала. Однако, большая избыточность приводит к большим временным затратам при передаче информации и требует большого объема памяти при ее хранении. Впервые теоретическое исследование эффективного кодирования предпринял К.Шеннон.

**Первая теорема** Шеннона декларирует возможность создания системы эффективного кодирования дискретных сообщений, у которой среднее число двоичных символов на один символ сообщения асимптотически стремится к энтропии источника сообщений (в отсутствии помех).

Задача эффективного кодирования описывается триадой:

$$X = \{X\,4i\}$$ - кодирующее устройство - $B$.

Здесь $X$, $B$ - соответственно входной и выходной алфавит. Под множеством $x_i$ можно понимать любые знаки (буквы, слова, предложения). $B$ - множество, число элементов которого в случае кодирования знаков числами определяется основанием системы счисления (например, $m = 2$). Кодирующее устройство сопоставляет каждому сообщению $x_i$ из $X$ кодовую комбинацию, составленную из $n_i$ символов множества $B$. Ограничением данной задачи является отсутствие помех. Требуется оценить минимальную среднюю длину кодовой комбинации.

Для решения данной задачи должна быть известна вероятность $P_i$ появления сообщения $x_i$, которому соответствует определенное количество символов $n_i$ алфавита $B$. Тогда математическое ожидание количества символов из $B$ определится следующим образом:

$$n_{cp} = n_i P_i \text{ (средняя величина).}$$

Этому среднему числу символов алфавита $B$ соответствует максимальная энтропия $Hmax = n_{\text{ср}} \log m$. Для обеспечения передачи информации, содержащейся в сообщениях $X$ кодовыми комбинациями из $B$, должно выполняться условие $H4\max \geq H(x)$, или $n_{cp} \log m \geq - P_i \log P_i$. В этом случае закодированное сообщение имеет избыточность $n_{cp} \geq H(x) / \log m$, $n_{\min} = H(x) / \log m$.

Коэффициент избыточности

$$K_{\text{u}} = (H_{\max} - H(x)) / H_{\max} = (n_{\text{ср}} - n_{\min}) / n_{\text{ср}}$$

Выпишем эти значения в виде табл. 1.8. Имеем:

$$N_{\min} = H(x) / \log 2 = 2{,}85, \quad K_{\text{u}} = (2{,}92 - 2{,}85) / 2{,}92 = 0{,}024,$$

т.е. код практически не имеет избыточности. Видно, что среднее число двоичных символов стремится к энтропии источника сообщений.

**Таблица 1.8 Пример к первой теореме Шеннона**

| $N$ | $Px_i$ | $x_i$ | Код | $n_i$ | $n_i \cdot P_i$ | $Px_i \cdot \log Px_i$ |
|---|---|---|---|---|---|---|
| 1 | 0,19 | $X_1$ | 10 | 2 | 0,38 | -4,5522 |
| 2 | 0,16 | $X_2$ | 001 | 3 | 0,48 | -4,2301 |
| 3 | 0.16 | $X_3$ | 011 | 3 | 0,48 | -4,2301 |
| 4 | 0,15 | $X_4$ | 101 | 3 | 0,45 | -4,1054 |
| 5 | 0,12 | $X_5$ | 111 | 3 | 0,36 | -3,6706 |
| 6 | 0,11 | $X_6$ | 111 | 3 | 0,33 | - 3,5028 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 7 | 0,09 | $X_7$ | 1011 | 4 | 0,36 | -3,1265 |
| 8 | 0,02 | $X_8$ | 1001 | 4 | 0,08 | -3,1288 |
| | Σ=1 | | | | Σ=2,92 | Σ=2,85 |

**Вторая теорема Шеннона** гласит, что при наличии помех в канале всегда можно найти такую систему кодирования, при которой сообщения будут переданы с заданной достоверностью. При наличии ограничения пропускная способность канала должна превышать производительность источника сообщений. Таким образом, вторая теорема Шеннона устанавливает принципы помехоустойчивого кодирования. Для дискретного канала с помехами теорема утверждает, что, если скорость создания сообщений меньше или равна пропускной способности канала, то существует код, обеспечивающий передачу со сколь угодно мглой частотой ошибок.

Доказательство теоремы основывается на следующих рассуждениях. Первоначально последовательность $X = \{xi\}$ кодируется символами из $B$ так, что достигается максимальная пропускная способность (канал не имеет помех). Затем в последовательность из $B$ длины $n$ вводится $r$ символов и по каналу передается новая последовательность из $n + r$ символов. Число возможных последовательностей длины и + $m$ больше числа возможных последовательностей длины $n$. Множество всех последовательностей длины $n + r$ может быть разбито на $n$ подмножеств, каждому из которых сопоставлена одна из последовательностей длины $n$. При наличии помехи на последовательность из $n + r$ выводит ее из соответствующего подмножества с вероятностью сколь угодно малой.

Это позволяет определять на приемной стороне канала, какому подмножеству принадлежит искаженная помехами принятая последовательность длины $n + r$, и тем самым восстановить исходную последовательность длины $n$.

Эта теорема не дает конкретного метода построения кода, но указывает на пределы достижимого в создании помехоустойчивых кодов, стимулирует поиск новых путей решения этой проблемы.

## 4.4. МЕЖДУНАРОДНЫЕ СИСТЕМЫ БАЙТОВОГО КОДИРОВАНИЯ

Информатика и ее приложения интернациональны. Это связано как с объективными потребностями человечества в единых правилах и законах хранения, передачи и обработки информации, так и с тем, что в этой сфере деятельности (особенно в ее прикладной части) заметен приоритет одной страны, которая благодаря этому получает возможность «диктовать моду».

Компьютер считают универсальным преобразователем информации. Тексты на естественных языках и числа, математические и специальные символы - одним словом все, что в быту или в профессиональной деятельности может быть необходимо человеку, должно иметь возможность быть введенным в компьютер.

В силу безусловного приоритета двоичной системы счисления при внутреннем представлении информации в компьютере кодирование «внешних» символов основывается на сопоставлении каждому из них определенной группы двоичных знаков. При этом из технических соображений и из соображений удобства кодирования-декодирования следует пользоваться равномерными кодами, т.е. двоичными группами равной длины.

Попробуем подсчитать наиболее короткую длину такой комбинации с точки зрения человека, заинтересованного в использовании лишь одного естественного алфавита - скажем, английского: 26 букв следует умножить на 2 (прописные и строчные) - итого 52; 10 цифр, будем считать, 10 знаков препинания; 10 разделительных знаков (три вида скобок, пробел и др.), знаки привычных математических действий, несколько специальных символов (типа #, $, & и др.) — итого ~ 100. Точный подсчет здесь не нужен, поскольку нам предстоит решить простейшую задачу: имея, скажем, равномерный код из групп по $N$ двоичных знаков, сколько можно образовать разных кодовых комбинаций. Ответ очевиден $K = 2^N$. Итак, при $N = 6$ $K = 64$ - явно мало, при $N = 7$ $K = 128$ - вполне достаточно.

Однако, для кодирования нескольких (хотя бы двух) естественных алфавитов (плюс все отмеченные выше знаки) и этого недостаточно. Минимально достаточное значение $N$ в этом случае 8; имея 256 комбинаций двоичных символов, вполне можно решить указанную задачу. Поскольку 8 двоичных символов составляют 1 байт, то говорят о системах «байтового» кодирования.

Наиболее распространены две такие системы: EBCDIC (Extended Binary Coded Decimal In-

terchange Code) и ASCII (American Standard Information Interchange).

Первая - исторически тяготеет к «большим» машинам, вторая чаще используется на мини- и микро-ЭВМ (включая персональные компьютеры). Ознакомимся подробнее именно с ASCII, созданной в 1963 г.

В своей первоначальной версии это - система семибитного кодирования. Она ограничивалась одним естественным алфавитом (английским), цифрами и набором различных символов, включая «символы пишущей машинки» (привычные знаки препинания, знаки математических действий и др.) и «управляющие символы». Примеры последних легко найти на клавиатуре компьютера: для микро-ЭВМ, например, DEL - знак удаления символа.

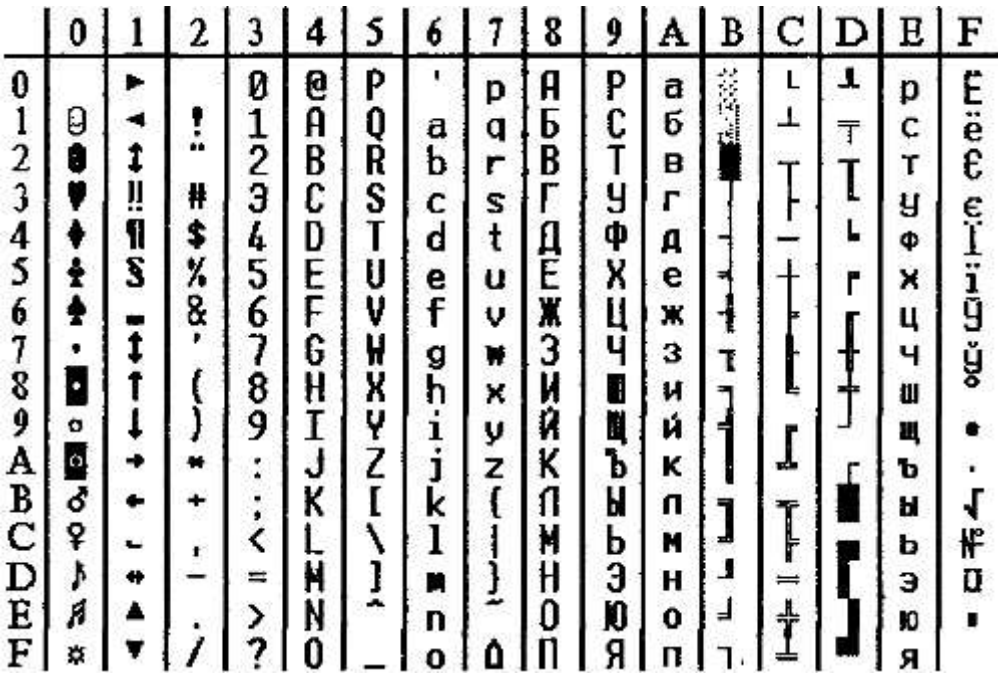
В следующей версии фирма IBM перешла на расширенную 8-битную кодировку. В ней первые 128 символов совпадают с исходными и имеют коды со старшим битом равным нулю, а остальные коды отданы под буквы некоторых европейских языков, в основе которых лежит латиница, греческие буквы, математические символы (скажем, знак квадратного корня) и символы псевдографики. С помощью последних можно создавать таблицы, несложные схемы и др.

Для представления букв русского языка (кириллицы) в рамках ASCII было предложено несколько версий. Первоначально был разработан ГОСТ под названием КОИ-7, оказавшийся по ряду причин крайне неудачным; ныне он практически не используется.

В табл. 1.9 приведена часто используемая в нашей стране модифицированная альтернативная кодировка. В левую часть входят исходные коды ASCII; в правую часть (расширение ASCII) вставлены буквы кириллицы взамен букв, немецкого, французского алфавитов (не совпадающих по написанию с английскими), греческих букв, некоторых спецсимволов.

Знакам алфавита ПЭВМ ставятся в соответствие шестнадцатиричные числа по правилу: первая - номер столбца, вторая - номер строки. Например: английская 'A' - код 41, русская 'и' - код A8.

Таблица 1.9 Таблица кодов ASCII (расширенная)



Одним из достоинств этой системы кодировки русских букв является их естественное упорядочение, т.е. номера букв следуют друг за другом в том же порядке, в каком сами буквы стоят в русском алфавите. Это очень существенно при решении ряда задач обработки текстов, когда требуется выполнить или использовать лексикографическое упорядочение слов.

Из сказанного выше следует, что даже 8-битная кодировка недостаточна для кодирования всех символов, которые хотелось бы иметь в расширенном алфавите. Все препятствия могут быть сняты при переходе на 16-битную кодировку Unicode, допускающую 65536 кодовых комбинаций.

***Контрольные вопросы***

1. Как определяется алфавит?

2. Что такое код?

3. Как объяснить большую помехоустойчивость передаваемых сообщений, составленных на русском языке?

4. Что определяют первая и вторая теоремы Шеннона?

# § 5. ЭЛЕМЕНТЫ ТЕОРИИ ГРАФОВ

## 5.1. ОСНОВНЫЕ ПОНЯТИЯ

Такая структура, как **граф** (в качестве синонима используется также термин «сеть») , имеет самые различные применения в информатике и в смежных прикладных областях, поэтому познакомимся с основными понятиями теории графов.

Граф *G = (V, E)* задается парой конечных множеств *V* и *E.* Элементы первого множества *V1, v2,..., vM* называются **вершинами** графа (при графическом представлении им соответствуют точки). Элементы второго множества *e1, e2, ...,eN* называют **ребрами.** Каждое ребро определяется парой вершин (при графическом представлении ребро соединяет две вершины графа). Если ребра графа определяются упорядоченными парами вершин, то такой граф называют **ориентированным** (на чертеже при изображении ориентированного графа на каждом ребре ставят стрелку, указывающую его направление). Ориентированный граф с пятью вершинами и семью ребрами изображен на рис. 1.6.

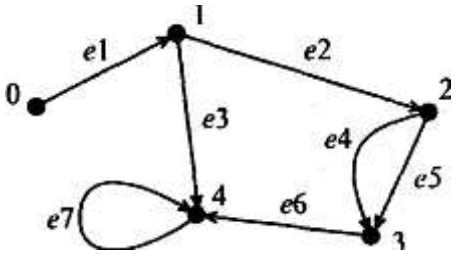



*Рис. 1.6.* Пример ориентированного графа

Если две вершины соединены двумя или более ребрами, то эти ребра называют параллельными (например, ребра e4 и *e5).* Если начало и конец ребра совпадают, то такое ребро называется **петлей** (например, ребро *e7).* Граф без петель и параллельных ребер называется простым.

Если ребро *ek* определяется вершинами *vi* и *vj* (будем обозначать этот факт следующим образом: *ek = (vi, vj),* то говорят, что ребро *ek* **инцидентно** вершинам *vi* и *vj.* Две вершины *vi* и *vj* называются смежными, если в графе существует ребро *(vi, vj).*

Последовательность вершин *vi1, vi2,..., vik,* таких, что каждая пара *(vi,(j-1), vij)* при $1 < j \le k$ определяет ребро, называется **маршрутом** в графе *G.* Вершины *vil* и *vik* называют **концевыми** вершинами маршрута, все остальные входящие в него вершины - **внутренними.**

Маршрут, в котором все определяемые им ребра различны, называют цепью. Цепь считают замкнутой, если ее концевые вершины совпадают. Замкнутая цепь, в которой все вершины (за исключением концевых) различны, называется циклом. Незамкнутая цепь, в которой все вершины различны, носит название путь. Если в ориентированном графе существует путь из *vi* в *vj,* то говорят, что вершина *vj* достижима из вершины *vi.*

Две вершины *vi* и *vj* называют связанными в графе *G,* если в нем существует путь, для которого эти вершины являются концевыми. Граф *G* называется связным, если каждые две вершины в нем являются связанными. На рис. 1.7 изображен простой неориентированный связный граф.

Последовательность вершин *v1, v5,* i4, *v3 ,* например, определяет путь, а последовательность вершин *v1, v5,* i4, *v3, vl, v1* - цикл. **Деревом** будем называть неориентированный связный граф без циклов. **Лес** - это любой граф без циклов. На рис. 1.8 показаны возможные деревья с пятью вершинами.

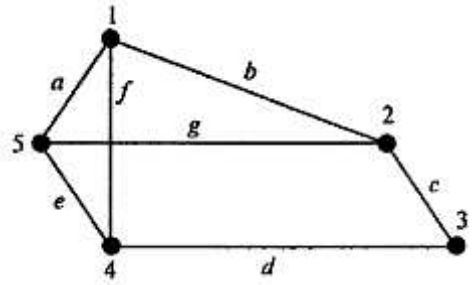



*Рис. 1.7.* Пример неориентированного связного графа

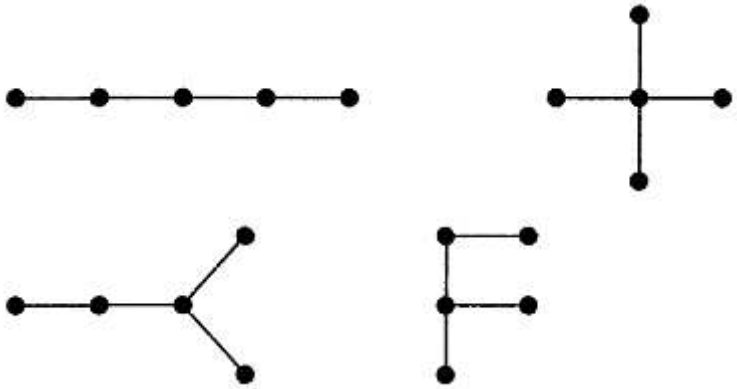


*Рис. 1.8.* Примеры деревьев

Анализ приведенных здесь понятий и определений показывает, что в качестве моделей графы удобно использовать в тех случаях, когда рассматривается система каких-либо объектов, между которыми существуют определенные связи, отношения, когда изучается структура системы, возможности ее функционирования. В информатике графы используются в разделах: операционные системы, алгоритмизация, структуры данных, информационное моделирование и др.

## 5.2. ПРЕДСТАВЛЕНИЕ ГРАФОВ

Важным вопросом, особенно для приложений теории графов, является определение возможных способов представления графов. Самый простой способ - полное перечисление множеств *V* и *E.* Однако, очевидно, что в этом случае выявление у графа различных характеристик и свойств будет крайне затруднительным. Граф можно представить в виде некоторого графического изображения и визуально определить некоторые свойства и характеристики заданного графа. Однако, при наличии в графе'большого числа ребер и вершин этот способ также мало пригоден. Рассматривая различные возможные способы представления графов, мы должны иметь в виду потребность ввода соответствующей информации в компьютер. В этой связи ввод информации в числовом виде предпочтителен, хотя современные технические средства допускают ввод и графической информации (таблиц, текста, графиков, рисунков и т.д.), после чего может производиться обработка такой информации.

**Матрица смежности.** Если вершины графа G помечены метками *v1, v2,..., vn,* то элементы матрицы смежности *A(G)* размера *V, xV* определяются следующим образом: *A(i.j)* = 1, если *vi* смежна с *vj; A(ij)* = 0 в противном случае (рис. 1.9, *а).*

**Матрица инцидентности.** Если вершины графа *G* помечены метками *v1, v2,..., vm,* а ребра - метками *e1, e2,..., en,* то элементы **матрицы инцидентности** *I(G)* размера *M* x *N* определяются правилом: *B(ij)* = 1, если *vi* инцидентна *ej; B(iJ)* = 0 в противном случае (см. рис. 1.9, *б).*

*A(G):*

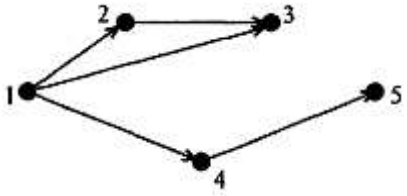
| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 1 |
| 2 | 1 | 0 | 1 | 0 | 1 |
| 3 | 0 | 1 | 0 | 1 | 0 |
| 4 | 1 | 0 | 1 | 0 | 1 |
| 5 | 1 | 1 | 0 | 1 | 0 |

*I(G):*

| | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 5 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |

*Рис. 1.9, а.* Матрица смежности *Рис. 1.9, б.* Матрица инцидентности

Для ориентированного графа *G,* имеющего *N* вершин можно рассмотреть матрицу **достижимости** *C(G)* размера *N* x *N,* элементы которой определяются так: C(*I*, *J)* = 1, если вершина *vj* достижима из *vi; C(I, J)* = 0 в противном случае. Ниже приведен пример ориентированного графа и его матрицы достижимости, рис. 1.10.




C(G):

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 0 | 0 | 0 |

*Рис. 1.10.* Матрица достижимости ориентированного графа

***Контрольные вопросы***

1. Каким образом определяется граф?
2. Что является путем в графе?
3. Как определяется такой вид графа, как дерево?
4. Какими способами можно задать граф?

## § 6. АЛГОРИТМ И ЕГО СВОЙСТВА

### 6.1. РАЗЛИЧНЫЕ ПОДХОДЫ К ПОНЯТИЮ «АЛГОРИТМ»

Понятие алгоритма - одно из фундаментальных понятий информатики. Алгоритмизация наряду с моделированием выступает в качестве общего метода информатики. К реализации определенных алгоритмов сводятся процессы управления в различных системах, что делает понятие алгоритма близким и кибернетике.

Алгоритмы являются объектом систематического исследования пограничной между математикой и информатикой научной дисциплины, примыкающей к математической логике - **теории алгоритмов.**

Особенность положения состоит в том, что при решении практических задач, предполагающих разработку алгоритмов для реализации на ЭВМ, и тем более при использовании на практике информационных технологий, можно, как правило, не опираться на высокую формализацию данного понятия. Поэтому представляется целесообразным познакомиться с алгоритмами и алгоритмизацией на основе содержательного толкования сущности понятия алгоритма и рассмотрения основных его свойств. При таком подходе алгоритмизация более выступает как набор определенных практических приемов, особых специфических навыков рационального мышления в рамках заданных языковых средств. Можно провести аналогию между этим обстоятельством и рассмотренным выше подходом к измерению информации: тонкие математические построения при «кибернетическом» подходе не очень нужны при использовании гораздо более простого «объемного» подхода при практической работе с компьютером.

Само слово «алгоритм» происходит от algorithmi - латинской формы написания имени великого математика IX века аль-Хорезми, который сформулировал правила выполнения арифметических действий. Первоначально под алгоритмами и понимали только правила выполнения четырех арифметических действий над многозначными числами.

### 6.2. ПОНЯТИЕ ИСПОЛНИТЕЛЯ АЛГОРИТМА

Понятие исполнителя невозможно определить с помощью какой-либо формализации. Исполнителем может быть человек, группа людей, робот, станок, компьютер, язык программирования и т.д. Важнейшим свойством, характеризующим любого из этих исполнителей, является то, что исполнитель умеет выполнять некоторые команды. Так, исполнитель-человек умеет выполнять такие команды как «встать», «сесть», «включить компьютер» и т.д., а исполнитель-язык программирования Бейсик - команды PRINT, END, LIST и другие аналогичные. Вся совокупность ко-

манд, которые данный исполнитель умеет выполнять, называется системой команд **исполнителя** (СКИ).

В качестве примера (рис. 1.11) рассмотрим исполнителя-робота, работа которого состоит в собственном перемещении по рабочему полю (квадрату произвольного размера, разделенному на клетки) и перемещении объектов, в начальный момент времени находящихся на «складе» (правая верхняя клетка).

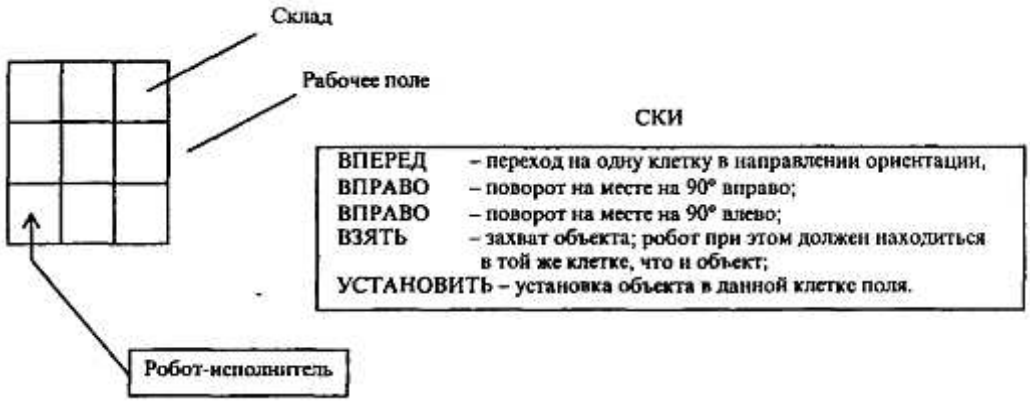



*Рис. 1.11.* Исполнитель-робот

Одно из принципиальных обстоятельств состоит в том, что исполнитель не вникает в смысл того, что он делает, но получает необходимый результат. В таком случае говорят, что исполнитель действует **формально,** т. е. отвлекается от содержания поставленной задачи и только строго выполняет некоторые правила, инструкции.

Это - важная особенность алгоритмов. Наличие алгоритма формализует процесс решения задачи, исключает рассуждение исполнителя. Использование алгоритма дает возможность решать задачу формально, механически исполняя команды алгоритма в указанной последовательности. Целесообразность предусматриваемых алгоритмом действий обеспечивается точным анализом со стороны того, кто составляет этот алгоритм.

Введение в рассмотрение понятия «исполнитель» позволяет определить алгоритм как понятное и точное предписание исполнителю совершить последовательность действий, направленных на достижение поставленной цели. В случае исполнителя-робота мы имеем пример алгоритма «в обстановке», характеризующегося отсутствием каких-либо величин. Наиболее же распространенными и привычными являются алгоритмы работы с величинами - числовыми, символьными, логическими и т.д.

## 6.3. ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ АЛГОРИТМОВ

Алгоритм, составленный для некоторого исполнителя, можно представить различными способами: с помощью графического или словесного описания, в виде таблицы, последовательностью формул, записанным на алгоритмическом языке (языке программирования). Остановимся на графическом описании алгоритма, называемом блок-схемой. Этот способ имеет ряд преимуществ благодаря наглядности, обеспечивающей, в частности, высокую «читаемость» алгоритма и явное отображение управления в нем.

Прежде всего определим понятие блок-схемы. Блок-схема - это ориентированный граф, указывающий порядок исполнения команд алгоритма; вершины такого графа могут быть одного из трех типов (рис. 1.12).

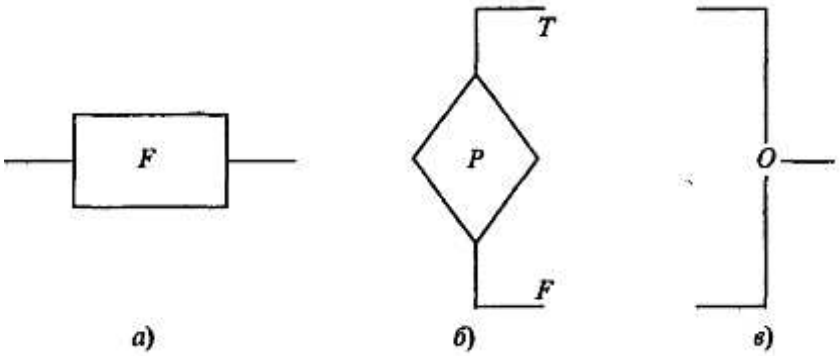



*Рис. 1.12.* Три типа вершин графа

На рис. 1.12 изображены «функциональная» (*а*) вершина (имеющая один вход и один выход); «предикатная» *(б)* вершина, имеющая один вход и два выхода (в этом случае функция *P* передает управление по одной из ветвей в зависимости от значения *P (T,* т.е. true, означает «истина», *F,* т.е. false - «ложь»); «объединяющая» (в) вершина (вершина «слияния»), обеспечивающая передачу управления от одного из двух входов к выходу. Иногда вместо Т пишут «да» (либо знак +), вместо *F*- «нет» (либо знак -).

Из данных элементарных блок-схем можно построить четыре блок-схемы (рис. 1.13), имеющих особое значение для практики алгоритмизации.

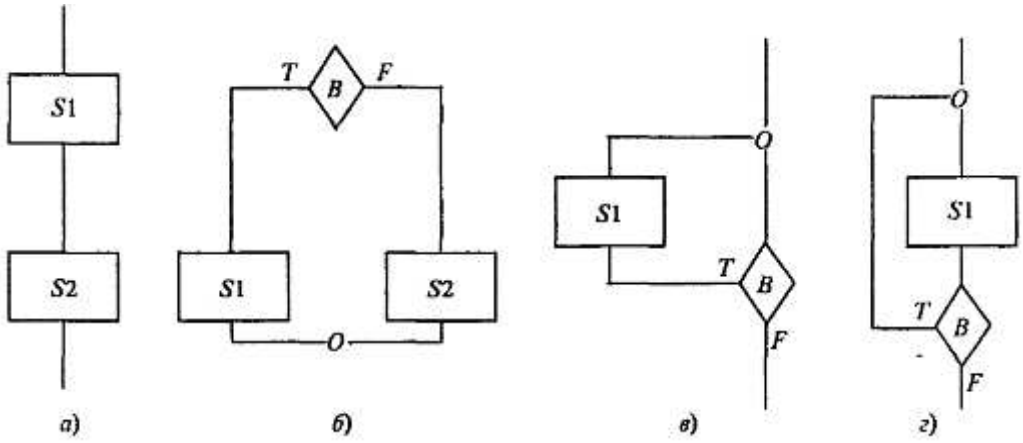



*Рис. 1.13.* Основные алгоритмические структуры

На рис. 1.13 изображены следующие блок-схемы: *а* - **композиция,** или следование; *б* - **альтернатива,** или **развилка,** *в* и *г* **-** блок-схемы, каждую из которых называют **итерацией,** или **циклом** (с предусловием (*в*), с постусловием (*г*)). *S1* и *S2* представляют собой в общем случае некоторые серии команд для соответствующего исполнителя, *B* - это условие, в зависимости от истинности *(T)* или ложности *(F)* которого управление передаётся по одной из двух ветвей. Можно доказать, что для составления любого алгоритма достаточно представленных выше четырех блок-схем, если пользоваться их последовательностями и/или суперпозициями.

Блок-схема «альтернатива» может иметь и сокращенную форму, в которой отсутствует ветвь *S2* (рис. 1.14, *а).* Развитием блок-схемы типа альтернатива является блок-схема «выбор» (рис. 1.14, б).

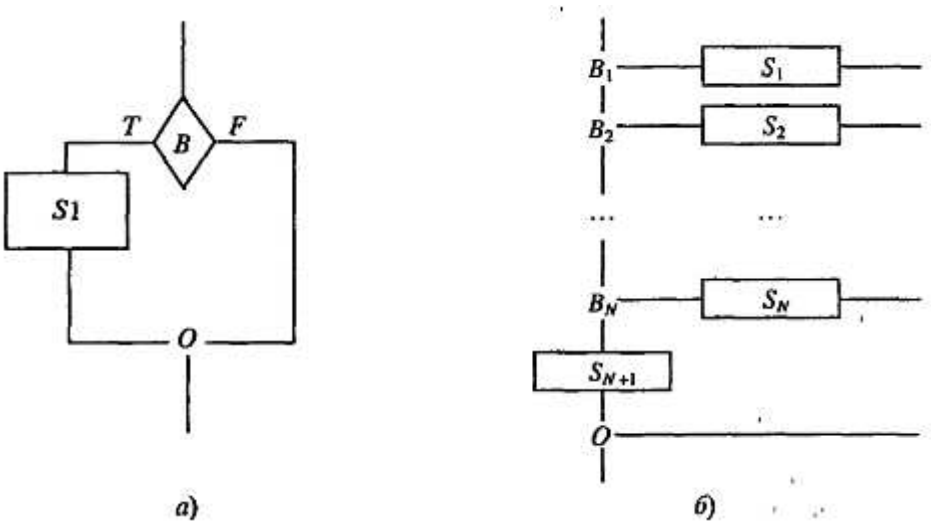



*Рис. 1.14.* Развитие структуры типа «альтернатива»;
о) - неполная развилка; *б)* - структура «выбор»

На практике при составлении блок-схем оказывается удобным использовать и другие графические знаки (некоторые из них приведены на рис. 1.15).

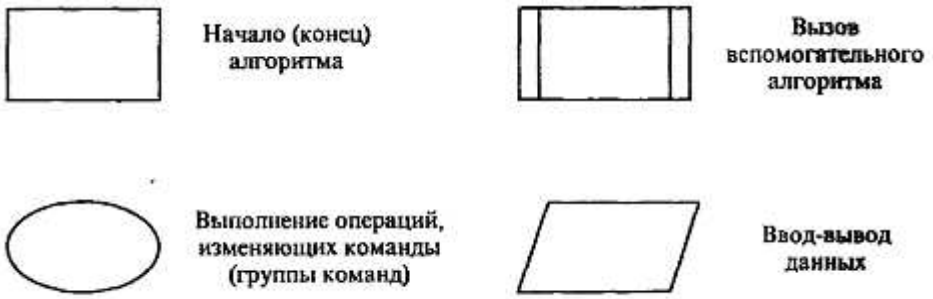



*Рис. 1.15.* Некоторые дополнительные конструкции для изображения блок-схем алгоритмов

## 6.4. СВОЙСТВА АЛГОРИТМОВ

Алгоритм должен быть составлен таким образом, чтобы исполнитель, в расчете на которого он создан, мог однозначно и точно следовать командам алгоритма и эффективно получать определенный результат. Это накладывает на записи алгоритмов ряд обязательных требований, суть которых вытекает, вообще говоря, из приведенного выше неформального толкования понятия алгоритма. Сформулируем эти требования в виде перечня свойств, которым должны удовлетворять алгоритмы, адресуемые заданному исполнителю.

1. Одно из первых требований, которое предъявляется к алгоритму, состоит в том, что описываемый процесс должен быть разбит на последовательность отдельных шагов. Возникающая в результате такого разбиения запись представляет собой упорядоченную совокупность четко разделенных друг от друга предписаний (директив, команд, операторов), образующих прерывную (или, как говорят, дискретную) структуру алгоритма. Только выполнив требования одного предписания, можно приступить к выполнению следующего. Дискретная структура алгоритмической записи может. Например, подчеркиваться сквозной нумерацией отдельных команд алгоритма, хотя это требование не является обязательным. Рассмотренное свойство алгоритмов называют дискретностью.

2. Используемые на практике алгоритмы составляются с ориентацией на определенного исполнителя. Чтобы составить для него алгоритм, нужно знать, какие команды этот исполнитель может понять и исполнить, а какие - не может. Мы знаем, что у каждого исполнителя имеется своя система команд. Очевидно, составляя запись алгоритма для определенного исполнителя, можно использовать лишь те команды, которые имеются в его СКИ. Это свойство алгоритмов будем называть понятностью.

3. Будучи понятным, алгоритм не должен содержать предписаний, смысл которых может восприниматься неоднозначно, т.е. одна и та же команда, будучи понятна разным исполнителям, после исполнения каждым из них должна давать одинаковый результат.

Запись алгоритма должна быть настолько четкой, полной и продуманной в деталях, чтобы у исполнителя не могло возникнуть потребности в принятии решений, не предусмотренных составителем алгоритма. Говоря иначе, алгоритм не должен оставлять места для произвола исполнителя. Кроме того, в алгоритмах недопустимы также ситуации, когда после выполнения очередной команды алгоритма исполнителю неясно, какая из команд алгоритма должна выполняться на следующем шаге.

Отмеченное свойства алгоритмов называют **определенностью** или **детерминированностью.**

4. Обязательное требование к алгоритмам - **результативность.** Смысл этого требования состоит в том, что при точном исполнении всех предписаний алгоритма процесс должен прекратиться за конечное число шагов и при этом должен получиться определенный результат. Вывод о том, что решения не существует - тоже результат.

5. Наиболее распространены алгоритмы, обеспечивающие решение не одной конкретной задачи, а некоторого класса задач данного типа. Это свойство алгоритма называют **массовостью.** В простейшем случае массовость обеспечивает возможность использования различных исходных данных.

## 6.5. ПОНЯТИЕ АЛГОРИТМИЧЕСКОГО ЯЗЫКА

Достаточно распространенным способом представления алгоритма является его запись на алгоритмическом **языке,** представляющем в общем случае систему обозначений и правил для единообразной и точной записи алгоритмов и исполнения их. Отметим, что между понятиями «алгоритмический язык» и «языки программирования» есть различие; прежде всего, под исполнителем в алгоритмическом языке может подразумеваться не только компьютер, но и устройство для работы «в обстановке». Программа, записанная на алгоритмическом языке, не обязательно предназначена компьютеру. Практическая же реализация алгоритмического языка -отдельный вопрос в каждом конкретном случае.

Как и каждый язык, алгоритмический язык имеет свой словарь. Основу этого словаря составляют слова, употребляемые для записи команд, входящих в систему команд исполнителя того или иного алгоритма. Такие команды называют простыми командами. В алгоритмическом языке используют слова, смысл и способ употребления которых задан раз и навсегда. Эти слова называют служебными. Использование служебных слов делает запись алгоритма более наглядной, а форму представления различных алгоритмов - единообразной.

Алгоритм, записанный на алгоритмическом языке, должен иметь название. Название желательно выбирать так, чтобы было ясно, решение какой задачи описывает данный алгоритм. Для выделения названия алгоритма перед ним записывают служебное слово АЛГ (АЛГоритм). За названием алгоритма (обычно с новой строки) записывают его команды. Для указания начала и конца алгоритма его команды заключают в пару служебных слов НАЧ (НАЧало) и КОН (КОНец). Команды записывают последовательно.

Последовательность записи алгоритма:

```
АЛГ название алгоритма
НАЧ
        серия команд алгоритма
КОН
```

Например, алгоритм, определяющий движение исполнителя-робота, может иметь вид:

```
АЛГ в_склад
НАЧ
                вперед
                поворот на 90° направо
                вперед
КОН
```

При построении новых алгоритмов могут использоваться алгоритмы, составленные ранее. Алгоритмы, целиком используемые в составе других алгоритмов, называют вспомогательными алгоритмами. Вспомогательным может оказаться любой алгоритм из числа ранее составленных. Не исключается также, что вспомогательным в определенной ситуации может оказаться алгоритм, сам содержащий ссылку на вспомогательные алгоритмы.

Очень часто при составлении алгоритмов возникает необходимость использования в качестве вспомогательного одного и того же алгоритма, который к тому же может быть весьма сложным и громоздким. Было бы нерационально, начиная работу, каждый раз заново составлять и запоминать такой алгоритм для его последующего использования. Поэтому в практике широко используют, так называемые, встроенные (или стандартные) вспомогательные алгоритмы, т.е. такие алгоритмы, которые постоянно имеются в распоряжении исполнителя. Обращение к таким алгоритмам осуществляется так же, как и к «обычным» вспомогательным алгоритмам. У исполнителя-робота встроенным вспомогательным алгоритмом может быть перемещение в склад из любой точки рабочего поля; у исполнителя-язык программирования Бейсик это, например, встроенный алгоритм «SIN».

Алгоритм может содержать обращение к самому себе как вспомогательному и в этом случае его называют **рекурсивным.** Если команда обращения алгоритма к самому себе находится в самом алгоритме, то такую рекурсию называют прямой. Возможны случаи, когда рекурсивный вызов данного алгоритма происходит из вспомогательного алгоритма, к которому в данном алгоритме имеется обращение. Такая рекурсия называется **косвенной.** Пример прямой рекурсии:

```
АЛГ   движение
НАЧ
      вперед
      вперед
      вправо
      движение
КОН
```

Алгоритмы, при исполнении которых порядок следования команд определяется в зависимости от результатов проверки некоторых условий, называют **разветвляющимися.** Для их описания в алгоритмическом языке используют специальную составную команду - команда **ветвления.** Она соответствует блок-схеме «альтернатива» и также может иметь полную или сокращенную форму. Применительно к исполнителю-роботу условием может быть проверка нахождения робота у края рабочего поля (край/не_край); проверка наличия объекта в текущей клетке (есть/нет) и некоторые другие:

```
ЕСЛИ условие          ЕСЛИ условие     ЕСЛИ край
      ТО серия 1      ТО серия         ТО вправо
      ИНАЧЕ серия2    ВСЕ              ИНАЧЕ вперед
ВСЕ                                    ВСЕ
```

Ниже приводится запись на алгоритмическом языке команды выбора (см. рис. 1.14, б), являющейся развитием команды ветвления:

```
ВЫБОР
      ПРИ условие 1:        серия 1
      ПРИ условие 2:        серия 2
      …
      ПРИ условие N:        серия N
      ИНАЧЕ           серия N+1
ВСЕ
```

Алгоритмы, при исполнении которых отдельные команды или серии команд выполняются неоднократно, называют циклическими. Для организации циклических алгоритмов в алгоритми-

ческом языке используют специальную составную команду цикла. Она соответствует блок-схемам типа «итерация» и может принимать следующий вид:

```
ПОКА условие                НЦ
    НЦ                          серия
        Серия                   ДО условие
    КЦ                      КЦ
```

В случае составления алгоритмов работы с величинами можно рассмотреть и другие возможные алгоритмические конструкции, например, цикл с параметром или выбор. Подробно эти конструкции будут рассматриваться при знакомстве с реальными языками программирования.

В заключение данного параграфа приведем алгоритм, составленный для исполнителя-робота, по которому робот переносит все объекты со склада в левый нижний угол рабочего поля (поле может иметь произвольные размеры):

```
АЛГ перенос             АЛГ в_угол3             АЛГ до_края
НАЧ                     НАЧ                     НАЧ
    в_угол3                 до_края                 ПОКА не_край
    ЕСЛИ есть               вправо                  НЦ
        ТО                      до_края                 вперед
            взять               вправо              КЦ
            в_угол3             КОН                 КОН
            установить
            перенос
        ИНАЧЕ
            в_угол3
    ВСЕ
КОН
```

***Контрольные вопросы***

1. Каковы возможные подходы к определению понятия алгоритм?
2. Кто (что) может быть исполнителем алгоритма?
3. В чем особенности графического способа представления алгоритмов?
4. Каковы основные алгоритмические структуры?
5. Чем определяются свойства алгоритмов «дискретность», «определенность», «понятность», «результативность», «массовость»?
6. Что такое алгоритмический язык?

## §7. ФОРМАЛИЗАЦИЯ ПОНЯТИЯ «АЛГОРИТМ»

### 7.1. ПОСТАНОВКА ПРОБЛЕМЫ

Понятие алгоритма, введенное в предыдущем параграфе, можно назвать понятием алгоритма в интуитивном смысле. Оно имеет нечеткий, неформальный характер, ссылается на некоторые точно не определенные, но интуитивно понятные вещи. Например, при определении и обсуждении свойств алгоритма мы исходили из возможностей некоторого исполнителя алгоритма. Его наличие предполагалось, но ничего определенного о нем не было известно. Говоря языком математики, ни аксиоматического, ни исчерпывающего конструктивного определения исполнителя мы так и не дали.

Установленные в предыдущем параграфе свойства алгоритмов следует называть эмпирическими. Они выявлены на основе обобщения свойств алгоритмов различной природы и имеют прикладной характер. Этих свойств достаточно для практического программирования, для создания обширного круга программ для компьютеров, станков с ЧПУ, промышленных роботов и т.д. Однако, как фундаментальное научное понятие алгоритм требует более обстоятельного изучения. Оно невозможно без уточнения понятия «алгоритм», более строгого его описания или, как еще говорят, без его **формализации.**

Известно несколько подходов к формализации понятия «алгоритм»:

• теория конечных и бесконечных автоматов;

• теория вычислимых (рекурсивных) функций;

• λ-исчисление Черча.

Все эти возникшие исторически независимо друг от друга подходы оказались впоследствии эквивалентными. Главная цель формализации понятия алгоритма такова: подойти к решению проблемы алгоритмической разрешимости различных математических задач, т.е. ответить на вопрос, может ли быть построен алгоритм, приводящий к решению задачи. Мы рассмотрим постановку этой проблемы и некоторые результаты теории алгоритмической разрешимости задач, но вначале обсудим формализацию понятия алгоритма в теории автоматов на примере машин Поста, Тьюринга, а также нормальных алгоритмов Маркова, а затем - основы теории рекурсивных функций. Идеи λ-исчислений Черча реализованы в языке программирования LISP (глава 3).

Вместе с тем, формально определенный любым из известных способов алгоритм не может в практическом программировании заменить то, что мы называли алгоритмами в предыдущем параграфе. Основная причина состоит в том, что формальное определение резко сужает круг рассматриваемых задач, делая многие практически важные задачи недоступными для рассмотрения.

## 7.2. МАШИНА ПОСТА

Абстрактные (т.е. существующие не реально, а лишь в воображении) машины Поста и Тьюринга, предназначенные для доказательств различных утверждений о свойствах программ для них, были предложены независимо друг от друга (и практически одновременно) в 1936 г. американским математиком Эмилем Постом и английским математиком Алланом Тьюрингом. Эти машины представляют собой универсальных исполнителей, являющихся полностью детерминированными, позволяющих «вводить» начальные данные, и после выполнения программ «читать» результат. Машина Поста менее популярна, хотя она значительно проще машины Тьюринга. С ее помощью можно вести обучение первым навыкам составления программ для ЭВМ.

Абстрактная машина Поста представляет собой бесконечную ленту, разделенную на одинаковые клетки, каждая из которых может быть либо пустой, либо заполненной меткой «V», и головки, которая может перемещаться вдоль ленты на одну клетку вправо или влево, наносить в клетку ленты метку, если этой метки там ранее не было, стирать метку, если она была, или проверять наличие в клетке метки. Информация о заполненных метками клетках ленты характеризует состояние ленты, которое может меняться в процессе работы машины. В каждый момент времени головка («-») находится над одной из клеток ленты и, как говорят, обозревает ее. Информация о местоположения головки вместе с состоянием ленты характеризует состояние машины Поста, рис. 1.16.

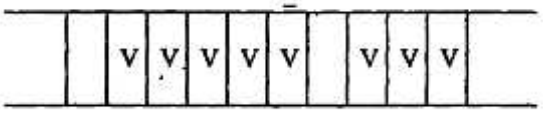



*Рис. 1.16.* Абстрактная машина Поста

Команда машины Поста имеет следующую структуру:

*n Km,*

где *n* - порядковый номер команды, *K*-действие, выполняемое головкой, *m* - номер следующей команды, подлежащей выполнению.

Существует всего шесть команд машины Поста, рис. 1.17:

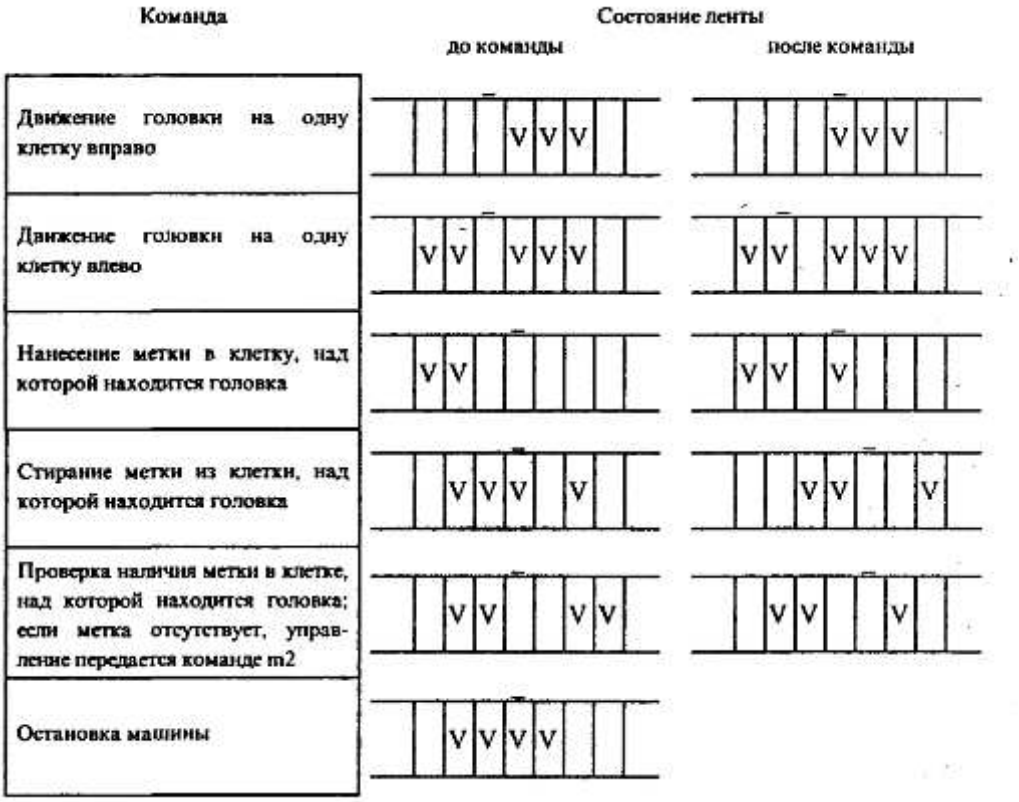



*Рис. 1.17.* Команды машины Поста

Ситуации, в которых головка должна наносить метку там, где она уже имеется, или, наоборот, стирать метку там, где ее нет, являются аварийными (недопустимыми).

Программой для машины Поста будем называть непустой список команд, такой что 1) на $n$-м месте команда с номером $n$; 2) номер $m$ каждой команды совпадает с номером какой-либо команды списка.

С точки зрения свойств алгоритмов, изучаемых с помощью машины Поста, наибольший интерес представляют причины останова машины при выполнении программы:

1) останов по команде «стоп»; такой останов называется результативным и указывает на корректность алгоритма (программы);

2) останов при выполнении недопустимой команды; в этом случае останов называется безрезультативным;

3) машина не останавливается никогда; в этом и в предыдущем случае мы имеем дело с некорректным алгоритмом (программой).

Будем понимать под начальным состояние головки против пустой клетки левее самой левой метки на ленте.

Рассмотрим реализацию некоторых типичных элементов программ машины Поста.

1. Пусть задано исходное состояние головки и требуется на пустой ленте написать две метки: одну в секцию под головкой, вторую справа от нее. Это можно сделать по следующей программе (справа от команды показан результат ее выполнения):

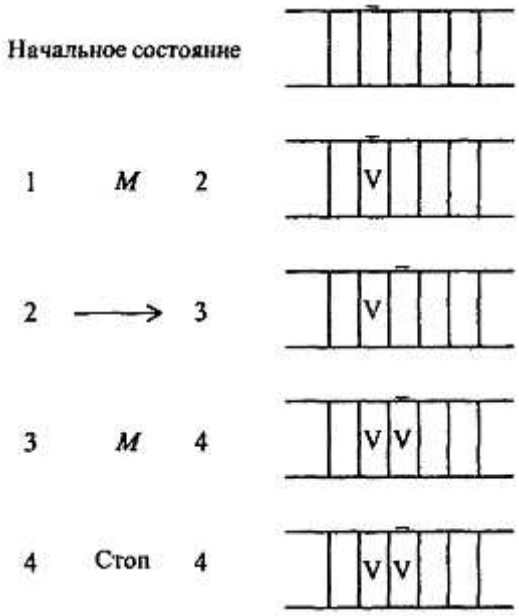
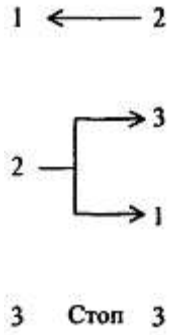



*Рис. 1.18.* Пример элемента программы машины Поста

2. Покажем, как можно воспользоваться командой условного перехода для организации циклического процесса. Пусть на ленте имеется запись из нескольких меток подряд и головка находится над самой крайней меткой справа. Требуется перевести головку влево до первой пустой позиции.

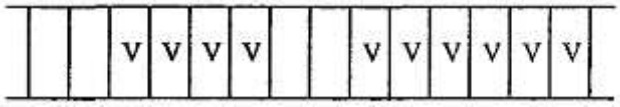
Программа будет иметь следующий вид:




Команда условного перехода является одним из основных средств организации циклических процессов, например, для нахождения первой метки справа (или слева) от головки, расположенной над пустой клеткой; нахождение слева (или справа) от головки пустой клетки, если она расположена над меткой и т.д.

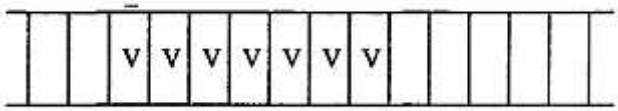
3. Остановимся на представлении чисел на ленте машины Поста и выполнении операций над ними.

Число $k$ представляется на ленте машины Поста идущими подряд $k + 1$ метками (одна метка означает число «О»). Между двумя числами делается интервал как минимум из одной пустой секции на ленте. Например, запись чисел 3 и 5 на ленте машины Поста будет выглядеть так:



Обратим внимание, что используемая в машине Поста система записи чисел является непозиционной.

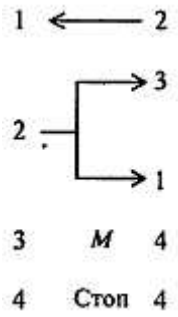
Составим программу для прибавления к произвольному числу единицы. Предположим, что на ленте записано только одно число и головка находится над одной из клеток, в которой находится метка, принадлежащая этому числу:



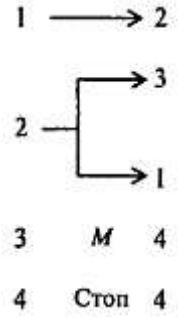
Для решения задачи можно переместить головку влево (или вправо) до первой пустой клет-

ки, а затем нанести метку.

Программа, добавляющая к числу метку слева, имеет вид:

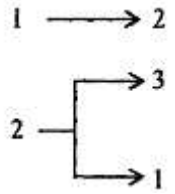



Программа, добавляющая к числу метку справа, имеет вид:

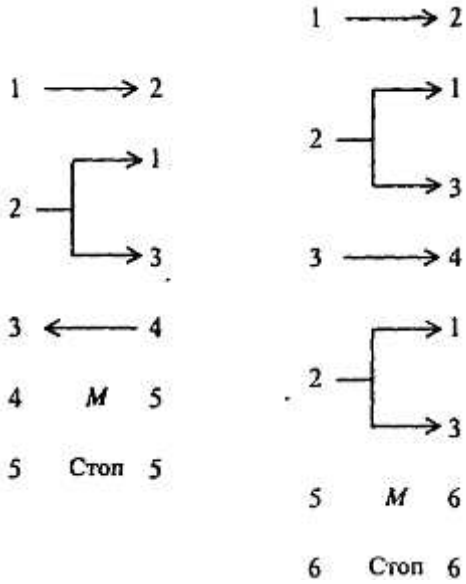



(отличие только в направлении движения головки в первой команде. Проверьте работоспособность этих программ на каких-либо частных примерах).

Предположим, что головка расположена на расстоянии нескольких клеток слева от числа, к которому нужно прибавить единицу. В этом случае программа усложняется. Появится «блок поиска числа» - две команды, приводящие головку в состояние, рассмотренное в предыдущем примере:

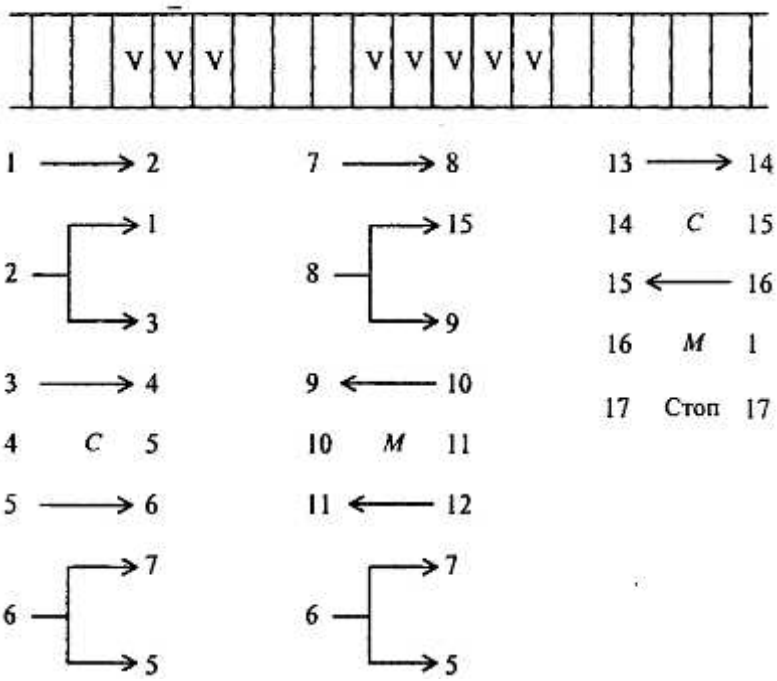



Ниже - полные тексты программ, добавляющие единицу слева и справа, соответственно:

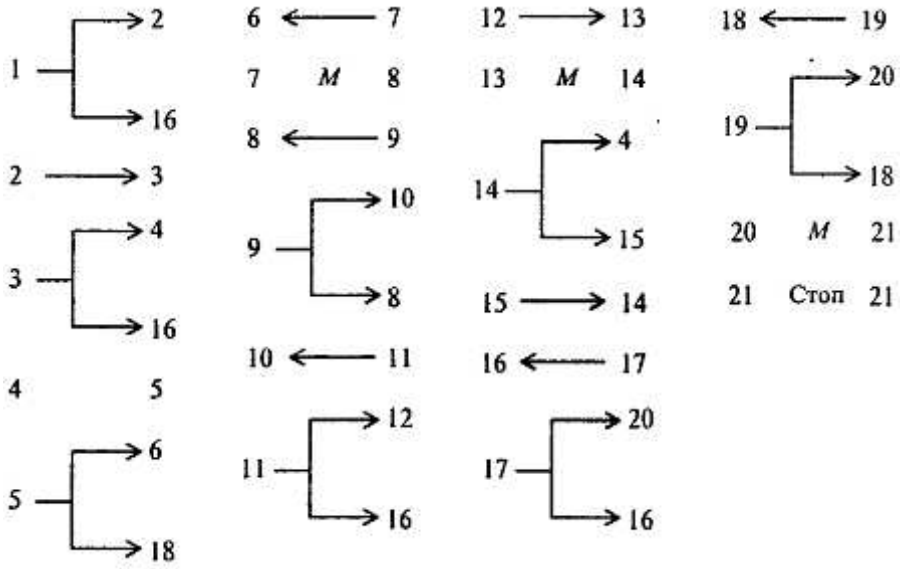



В первом случае не нужно перемещать головку к крайней левой метке числа

4. Приведем программу для сложения целых неотрицательных чисел а и и на машине Поста, когда головка находится над числом *a,* а число *b* находится правее числа *a* на некоторое число клеток. Эта программа реализует следующий алгоритм: первое число постепенно придвигается ко второму до их слияния, а потом стирается одна метка (иначе результат оказался бы на единицу больше правильного).




В случае более сложных начальных условий, когда неизвестно, справа или слева от головки (и на какое число клеток) находится число, можно применить такой принцип поиска числа: двигая головку вправо и влево и отмечая метками степень удаления головки от исходного положения, найти число, а потом уже применить известную программу сложения. При этом проверяется, находится ли головка над одной из меток числа и если да, то задача решена. Иначе проверяется, пуста ли секция справа от головки и следующая за ней; если обе пусты, то делается возврат головки на один шаг и ставится метка, а затем такая же операция выполняется слева и по отмеченной дорожке головка возвращается вправо и т.д. до тех пор, пока головка не натолкнется на число, после чего можно применить ранее рассмотренные выше программы:




Машину Поста можно рассматривать как упрощенную модель ЭВМ. В самом деле, как ЭВМ, так и машина Поста имеют:

• неделимые носители информации (клетки - биты), которые могут быть заполненными или незаполненными;

• ограниченный набор элементарных действий - команд, каждая из которых выполняется за один такт (шаг).

Обе машины работают на основе программы. Однако, в машине Поста информация располагается линейно и читается подряд, а в ЭВМ можно читать информацию по адресу; набор команд ЭВМ значительно шире и выразительнее, чем команды машины Поста и т.д.

## 73. МАШИНА ТЬЮРИНГА

Машина Тьюринга подобна машине Поста, но функционирует несколько иначе.

Машина Тьюринга (МТ) состоит из счетной ленты (разделенной на ячейки и ограниченной слева, но не справа), читающей и пишущей головки, лентопротяжного механизма и операционного исполнительного устройства, которое может находиться в одном из дискретных состояний $qo$, $q_1$, ..., $q_s$, принадлежащих некоторой конечной совокупности (алфавиту внутренних состояний). При этом $q_0$ называется начальным состоянием.

Читающая и пишущая головка может читать буквы рабочего алфавита $A = [a_0, a_1, ..., a_t]$, стирать их и печатать. Каждая ячейка ленты в каждый момент времени занята буквой из множества $A$. Чаще всего встречается буква $a_0$ - «пробел». Головка находится в каждый момент времени над некоторой ячейкой ленты -текущей рабочей ячейкой. Лентопротяжный механизм может перемещать ленту так, что головка оказывается над соседней ячейкой ленты. При этом возможна ситуация выхода за левый край ленты (ЛК), которая является аварийной (недопустимой), или машинного останова (МО), когда машина выполняет предписание об остановке.

Порядок работы МТ (с рабочим алфавитом $a_0, a_1,..., a_t$ и состояниями $q_0, q_1,..., q_s$) описывается таблицей машины Тьюринга. Эта таблица является матрицей с четырьмя столбцами и $(s + 1)$ $(t + 1)$ строками. Каждая строка имеет вид

$$q_i a_j v_{ij} q_{ij}, \quad 0 \le i \le s, \quad 0 \le j \le t, \quad q_{ij} \in \{q_0, q_1, ..., q_s\}.$$

Здесь через $v_{ij}$ обозначен элемент объединения алфавита $\{a_0, a_1,..., a_t\}$ и множества предписаний для лентопротяжного механизма: $l$ - переместить ленту влево, $r$ -переместить ленту вправо, $s$ - остановить машину; $v_{ij}$ - действие МТ, состоящее либо в занесении в ячейку ленты символа алфавита $a_0, a_1, ..., a_t$, либо в движении головки, либо в останове машины; $q_{ij}$ является последующим состоянием.

МТ работает согласно следующим правилам: если МТ находится в состоянии $q_i$, головка прочитывает символ 0 в рабочей ячейке. Пусть строка $q_i\ a_j\ v_{ij}\ q_{ij}$, начинающаяся с символов $q_i$, $a_j$, встречается только один раз в таблице. Если $v_{ij}$ - буква рабочего алфавита, то головка стирает содержимое рабочей ячейки и заносит туда эту букву. Если $v_{ij}$ - команда $r$ или $l$ для лентопротяжного механизма, то лента сдвигается на одну ячейку вправо или влево (если не происходит выход за левый край ленты) соответственно. Если $v_{ij} = s$, то происходит машинный останов.

Машина Тьюринга начинает работу из некоторой начальной конфигурации, включающей в себя начальное состояние (обычно $q_0$) и положение считывающе-записывающей головки над определенной ячейкой ленты, содержащей один из символов рабочего алфавита А.

Отметим, что наличие разнообразных символов в рабочем алфавите МТ позволяет представлять на ленте произвольную текстовую и числовую информацию, а переходы управляющего центра МТ в различные состояния моделируют запоминание машиной Тьюринга промежуточных результатов работы. Таблица, определяющая порядок работы МТ, не является в прямом смысле слова программой (ее предписания выполняются не последовательно, одно за другим, а описывают преобразования символов некоторого текста, находящегося на ленте). Таблицу МТ часто называют схемой машины Тьюринга или попросту отождествляют с самой машиной Тьюринга, коль скоро ее устройство и принцип функционирования известны.

Рассмотрим примеры нескольких схем машины Тьюринга.

1. Алгоритм прибавления единицы к числу $n$ в десятичной системе счисления. Дана десятичная запись числа $n$ (т.е. представление натурального числа $n$ в десятичной системе счисления); требуется получить десятичную запись числа $n + 1$.

Очевидно, что внешний алфавит МТ должен состоять из десяти цифр 0,1,2,3,4,5,6,7,8,9 и символа пробела _. Эти цифры записывают по одной в ячейке (подряд, без пропусков).

Оказывается достаточным иметь два внутренних состояния машины: $q_1$ и $q_2$.

Предположим, что в начальный момент головка находится над одной из цифр числа, а машина находится в состоянии $q_1$. Тогда задача может быть решена в два этапа: движения головки к цифре единиц числа (во внутреннем состоянии $q_1$) и замене этой цифры на единицу большую (с учетом переноса 1 в следующий разряд, если это 9, во внутреннем состоянии $q_2$. Соответствующая схема МТ может иметь вид

| $a_i$ | $q_i$ | |
|---|---|---|
| | $q_1$ | $q_2$ |
| 0 | 0П$q_1$ | 1С$q_2$ |
| 1 | 1П$q_1$ | 2С$q_2$ |
| 2 | 2П$q_1$ | 3С$q_2$ |
| 3 | 3П$q_1$ | 4С$q_2$ |
| 4 | 4П$q_1$ | 5С$q_2$ |
| 5 | 5П$q_1$ | 6С$q_2$ |
| 6 | 6П$q_1$ | 7С$q_2$ |
| 7 | 7П$q_1$ | 8С$q_2$ |
| 8 | 8П$q_1$ | 9С$q_2$ |
| 9 | 9П$q_1$ | 0С$q_2$ |
| - | -Л$q_1$ | 1С$q_2$ |

2. Алгоритм записи числа в десятичной системе счисления.

Дана конечная последовательность меток, записанных в клетки ленты подряд, без пропусков. Требуется записать в десятичной системе число этих меток пересчитать метки).

Суть алгоритма может состоять в том, что к числу 0, записанному на ленте в начале работы машины, машина добавляет 1, стирая метку за меткой, так что вместо нуля возникает число $0 + k$.

Легко могут быть построены алгоритмы сложения чисел, их перемножения, нападения наибольшего общего делителя и т.д. Однако, главная цель введения машин Поста и Тьюринга не программирование для них, а изучение свойств алгоритмов и проблемы алгоритмической разрешимости задач.

В зависимости от числа используемых лент, их назначения и числа состояний устройства управления можно рассматривать различные модификации машин Тьюринга.

Предположим, мы расширили определение МТ, добавив определенное состояние $q$. устройства управления машины. Будем говорить, что если устройство управления переходит в состояние $q_0$ для заданного входного слова $x$, то машина допускает $x$; если устройство переходит в состояние $q_x$, то машина запрещает $x$. Такую машину будем называть машиной Тьюринга с двумя выходами. Могут быть рассмотрены многочисленные варианты машины Тьюринга, имеющие некоторое конечное число лент. В каждой клетке этих лент может находиться один из символов внешнего алфавита $A = \{a_0, a_1, ..., a_n\}$. Устройство управления машиной в каждый момент времени находится в одном из конечного множества состояний $Q = \{q_0, q_1, ..., q_m\}$. Для $K$-ленточной машины конфигурация ее в $i$-й момент времени описывается системой $k$-слов вида:

$$a_{il1} \dots a_{ill} q_i a_{ill+1} \dots s_{i1t};$$
$$a_{ik1} \dots a_{ikl} q_i a_{ikl+1} \dots a_{ikv};$$

первый индекс соответствует моменту времени, второй - номеру ленты, третий -номеру клетки, считая слева направо. Говорят, что машина выполняет команду

$$q_i a_{a1} \dots a_{ak} \rightarrow q_j a_{b1} k_1 \dots a_{bk} k_k,$$
$$\text{К} = \{\text{Л}, \text{С}, \text{П}\}.$$

Если, находясь в состоянии $q_i$ и обозревая ячейки с символами $a_{a1}$ — $a_{ak}$, машина переходит в состояние $q_j$, заменяя содержимое ячеек соответственно символами $a_{b1}$ — $a_{bк}$, то после этого ленты соответственно сдвигаются в направлениях $k_1 \dots k_k$.

До сих пор принималось, что различные алгоритмы осуществляются на различных машинах Тьюринга, отличающихся набором команд, внутренним и внешним алфавитами. Однако, можно построить универсальную машину Тьюринга, способную выполнять любой алгоритм любой машины Тьюринга. Это достигается путем кодирования конфигурации и программы любой данной машины Тьюринга в символах внешнего алфавита универсальной машины. Само кодирование должно выполняться следующим образом:

1) различные символы должны заменяться различными кодовыми группами, но один и тот же символ должен заменяться всюду, где бы он не встретился, одной и той же кодовой группой;

2) строки кодовых записей должны однозначно разбиваться на отдельные кодовые группы;

3) должна иметься возможность распознать кодовые группы, соответствующие командам *Л, П, С,* различать кодовые группы, соответствующие символам внешнего алфавита и внутренним состояниям.

Для сравнения структур различных машин и оценки их сложности необходимо иметь соответствующую меру сложности машин. К.Шеннон предложил рассматривать в качестве такой меры произведение числа символов внешнего алфавита и числа внутренних состояний. Большой интерес вызывает задача построения универсальной машин Тьюринга наименьшей сложности.

Может быть рассмотрено еще одно обобщение машин Тьюринга: их композиции. Операции композиции, выполняемые над алгоритмами, позволяют образовывать новые, более сложные алгоритмы из ранее известных простых алгоритмов. Поскольку машина Тьюринга - алгоритм, то операции композиции применимы и к машинам Тьюринга. Рассмотрим основные из них: произведение, возведение в степень, итерацию.

Пусть заданы машины Тьюринга $T_1$ и $T_2$, имеющие общий внешний алфавит $A = \{a_0, a_1, ..., a_m\}$ и внутренние состояния $Q_1 = \{q_0, q_1, ... q_n\}$ и $Q_2 = \{q_0, q_1, ..., q_t\}$ соответственно. Композицией или произведением машины $T_1$ и машины $T_2$ будем называть машину $T$ с тем же внешним алфавитом $A = \{a_0, a_1, ..., a_m\}$ и набором внутренних состояний $Q = \{q_0, q_1,..., q_2,, q_{n+1},..., q_{n+1}\}$ и программой, эквивалентной последовательному выполнению программ машин $T_1$ и $T_2$:

$$T = T_1 * T_2..$$

Таким же образом определяется операция возведения в степень: *n*-й степенью машины *T* называется произведение *T...T* с *n* сомножителями.

Операция итерации применима к одной машине и определяется следующим образом. Пусть машина $T_1$ имеет несколько заключительных состояний. Выберем ее *r-e* заключительное состояние и отождествим его в схеме машины с ее начальным состоянием. Полученная машина *T* является результатом итерации машины $T_1 : T = T_1$.

Прежде чем остановиться на проблеме алгоритмической разрешимости задач обратимся к другим способам формализации понятия алгоритма.

## 7.4. НОРМАЛЬНЫЕ АЛГОРИТМЫ МАРКОВА

Для формализации понятия алгоритма российский математик А.А.Марков предложил использовать ассоциативные исчисления.

Рассмотрим некоторые понятия ассоциативного исчисления. Пусть имеется алфавит (конечный набор различных символов). Составляющие его символы будем называть буквами. Любая конечная последовательность букв алфавита (линейный их ряд) называется словом в этом алфавите.

Рассмотрим два слова *N* и *M* в некотором алфавите *A.* Если *N* является частью *M,* то говорят, что *N* входит в *M.*

Зададим в некотором алфавите конечную систему подстановок *N - M, S - T,...,* где *N, M, S, T,...* - слова в этом алфавите. Любую подстановку *N-M* можно применить к некоторому слову *K* следующим способом: если в *K* имеется одно или несколько вхождений слова *N,* то любое из них может быть заменено словом *M,* и, наоборот, если имеется вхождение *M,* то его можно заменить словом *N.*

Например, в алфавите *A = {a, b, c}* имеются слова *N = ab, M = bcb, K = abcbcbab,* Заменив в слове *K* слово *N* на *M,* получим *bcbcbcbab* или *abcbcbbcb,* и, наоборот, заменив *M* на *N,* получим *aabcbab* или *abcabab.*

Подстановка *ab - bcb* недопустима к слову *bacb,* так как ни *ab,* ни *bcb* не входит в это слово. К полученным с помощью допустимых подстановок словам можно снова применить допустимые подстановки и т.д. Совокупность всех слов в данном алфавите вместе с системой допустимых подстановок называют **ассоциативным исчислением.** Чтобы задать ассоциативное исчисление, достаточно задать алфавит и систему подстановок.

Слова $P_1$ и $P_2$ в некотором ассоциативном исчислении называются смежными, если одно из них может быть преобразовано в другое однократным применением допустимой подстановки.

Последовательность слов *P,* $P_1$, $P_2$, ..., *M* называется дедуктивной цепочкой, ведущей от

слова *P к* слову *M,* если каждое из двух рядом стоящих слов этой цепочки - смежное.

Слова *P* и *M* называют эквивалентными, если существует цепочка от *P* к *M* и обратно.

Пример

| Алфавит | | Подстановки |
|---|---|---|
| {*a, b, c, d, e*} | *ac - ca,* | *abac - abace* |
| | *ad - da;* | *eca - ae* |
| | *bc - cb;* | *eda - be* |
| | *bd - db;* | *edb - be* |

Слова *abcde* и *acbde* - смежные (подстановка *bc - cb*). Слова *abcde - cadbe* эквивалентны.

Может быть рассмотрен специальный вид ассоциативного исчисления, в котором подстановки являются ориентированными: N → M (стрелка означает, что подстановку разрешается производить лишь слева направо). Для каждого ассоциативного исчисления существует задача: для любых двух слов определить, являются ли они эквивалентными или нет.

Любой процесс вывода формул, математические выкладки и преобразования также являются дедуктивными цепочками в некотором ассоциативном исчислении. Построение ассоциативных исчислений является универсальным методом детерминированной переработки информации и позволяет формализовать понятие алгоритма.

Введем понятие алгоритма на основе ассоциативного исчисления: алгоритмом в алфавите А называется понятное точное предписание, определяющее процесс над словами из А и допускающее любое слово в качестве исходного. Алгоритм в алфавите А задается в виде системы допустимых подстановок, дополненной точным предписанием о том, в каком порядке нужно применять допустимые подстановки и когда наступает остановка.

Пример

| Алфавит: | Система подстановок *B:* |
|---|---|
| A = {*a, b, c*} | *cb - cc* |
| | *cca - ab* |
| | *ab – bca* |

Предписание о применении подстановок: в произвольном слове *P* надо сделать возможные подстановки, заменив левую часть подстановок на правую; повторить процесс с вновь полученным словом.

Так, применяя систему подстановок *B* из рассмотренного примера к словам *babaac* и *bcacabc* получаем:

*babaac* → *bbcaaac* → остановка

*bcacabc* → *bcacbcac* → *bcacccac* → *bcacabc* → бесконечные процесс (остановки нет), так как мы получили исходное слово.

Предложенный А.А.Марковым способ уточнения понятия алгоритма основан на понятии нормального алгоритма, который определяется следующим образом. Пусть задан алфавит *A* и система подстановок *B.* Для произвольного слова *P* подстановки из *B* подбираются в том же порядке, в каком они следуют в *B.* .Если подходящей подстановки нет, то процесс останавливается. В противном случае берется первая из подходящих подстановок и производится замена ее правой частью первого вхождения ее левой части в *P.* Затем все действия повторяются для получившегося слова $P_1$. Если применяется последняя подстановка из системы *B,* процесс останавливается.

Такой набор предписаний вместе с алфавитом *A* и набором подстановок *B* определяют нормальный алгоритм. Процесс останавливается только в двух случаях: 1) когда подходящая подстановка не найдена; 2) когда применена последняя подстановка из их набора. Различные нормальные алгоритмы отличаются друг от друга алфавитами и системами подстановок.

Приведем пример нормального алгоритма, описывающего сложение -натуральных чисел (представленных наборами единиц).

Пример

| Алфавит: | Система подстановок B: |
|---|---|

A = (+, 1)

$1 + \rightarrow + 1$
$+ 1 \rightarrow 1$
$1 \rightarrow 1$

Слово P: 11+11+111

Последовательная переработка слова *P* с помощью нормального алгоритма Маркова проходит через следующие этапы:

$P = 11 + 11 + 111$
$P_1 = 1 + 111 + 111$
$P_2 = + 1111 + 111$
$P_3 = + 111 + 1111$
$P_4 = + 11 + 11111$
$P_5 = + 1 + 111111$
$P_6 = ++ 1111111$
$P_7 = + 1111111$
$P_8 = 1111111$
$P_9 = 1111111$

Нормальный алгоритм Маркова можно рассматривать как универсальную форму задания любого алгоритма. Универсальность нормальных алгоритмов декларируется принципом нормализации: для любого алгоритма в произвольном конечном алфавите *A* можно построить эквивалентный ему нормальный алгоритм над алфавитом *A,*

Разъясним последнее утверждение. В некоторых случаях не удается построить нормальный алгоритм, эквивалентный данному в алфавите *A,* если использовать в подстановках алгоритма только буквы этого алфавита. Однако, можно построить требуемый нормальный алгоритм, производя расширение алфавита *A* (добавляя к нему некоторое число новых букв). В этом случае говорят, что построенный алгоритм является алгоритмом над алфавитом *A,* хотя он будет применяться лишь к словам в исходном алфавите *A*.

Если алгоритм *N* задан в некотором расширении алфавита *A,* то говорят, что *N* есть нормальный алгоритм над алфавитом *A*.

Условимся называть тот или иной алгоритм нормализуемым, если можно построить эквивалентный ему нормальный алгоритм, и ненормализуемым в противном случае. Принцип нормализации теперь может быть высказан в видоизмененной форме: все алгоритмы нормализуемы.

| Данный принцип не может быть строго доказан, поскольку понятие произвольного алгоритма не является строго определенным и основывается на том, что все Известные в настоящее время алгоритмы являются нормализуемыми, а способы ромпозиции алгоритмов, позволяющие строить новые алгоритмы из уже известных, не выводят за пределы класса нормализуемых алгоритмов. Ниже перечислены способы композиции нормальных алгоритмов.

I. *Суперпозиция алгоритмов.* При суперпозиции двух алгоритмов *A* и *B* выходное слово первого алгоритма рассматривается как входное слово второго алгоритма *B*. Результат суперпозиции *C* может быть представлен в виде *C(p) = B(A(p)),*

II. *Объединение алгоритмов.* Объединением алгоритмов *A* и *B* в одном и том же алфавите называется алгоритм С в том же алфавите, преобразующий любое слово *p,* содержащееся в пересечении областей определения алгоритмов *A* и *B,* в записанные рядом слова *A(p)* и *B(p).*

III. *Разветвление алгоритмов.* Разветвление алгоритмов представляет собой композицию *D* трех алгоритмов *A, B* и *C,* причем область определения алгоритма *D* является пересечением областей определения всех трех алгоритмов *A, B* и С, а для любого слова *p* из этого пересечения *D(p) = A(p),* если *C(p) = e, D(p) = B(p),* если *C(p) = e,* где *e* - пустая строка.

IV. *Итерация алгоритмов.* Итерация (повторение) представляет собой такую композицию С двух алгоритмов *A* и *B,* что для любого входного слова *p* соответствующее слово *C(p)* получается в результате последовательного многократного применения алгоритма *A* до тех пор, пока не получится слово, преобразуемое алгоритмом *B*.

Нормальные алгоритмы Маркова являются не только средством теоретических построений, но и основой специализированного языка программирования, применяемого как язык символьных преобразований при разработке систем искусственного интеллекта. Это один из немногих языков, разработанных в России и получивших известность во всем мире.

Существует строгое доказательство того, что по возможностям преобразования нормальные алгоритмы Маркова эквивалентны машинам Тьюринга.

## 7.5. РЕКУРСИВНЫЕ ФУНКЦИИ

Еще одним подходом к проблеме формализации понятия алгоритма являются, так называемые, рекурсивные функции. Исторически этот подход возник первым, поэтому в математических исследованиях, посвященных алгоритмам, он имеет наибольшее распространение.

Рекурсией называется способ задания функции, при котором значение функции при определенном значении аргументов выражается через уже заданные значения функции при других значениях аргументов. Применение рекурсивных функций в теории алгоритмов основано на идее нумерации слов в произвольном алфавите последовательными натуральными числами. Таким образом любой алгоритм можно свести к вычислению значений некоторой целочисленной функции при целочисленных значениях аргументов.

Введем несколько основных понятий. Пусть *X, Y* - два множества. Частичной функцией (или отображением) из *X* в *Y* будем называть пару <*D(f), f*>, состоящую из подмножества $D(f) \subset X$ (называемого областью определения *f)* и отображения $f$: $D(f) \to Y$. Если *D(f)* пусто, то *f* нигде не определена. Будем считать, что существует единственная нигде не определенная частичная функция.

Через *N* будем обозначать множество натуральных чисел. Через $(N)^n$ (при $n \geq 1$) будем обозначать *n*-кратное декартово произведение *N* на себя, т.е. множество упорядоченных *n*-ок ($x_1$ ..., $x_n$), $x_i \subset N$. Основным объектом дальнейших построений будут частичные функции из $(N)^m$ в $(N)^n$ для различных *m* и *n.*

Частичная функция *f* из $(N)^m$ в $(N)^n$ называется вычислимой, если можно указать такой алгоритм («программу»), который для входного набора $x \subset (N)^m$ дает на выходе *f(x),* если $x \subset D(f)$ и нуль, если $x \not\subset D(f)$. В этом определении неформальное понятие алгоритма (программы) оказывается связанным (отождествленным) с понятием вычислимости функции. Вместо алгоритмов далее будут изучаться свойства вычислимых функций. Вместо вычислимых функций оказывается необходимым использовать более широкий класс функций (и более слабое определение) - полувычислимые функции. Частичная функция из *(N)"* в *(N)"* полувычислима, если можно указать такой алгоритм (программу), который для входного набора *x* с *(N)"* дает на выходе *x* е *D(f),* или алгоритм работает неопределенно долго, если *x* е *D(f).* Очевидно, что вычислимые функции полувычислимы, а всюду определенные полувычислимые функции вычислимы.

Частичная функция *f* называется невычислимой, если она не является ни вычислимой, ни полувычислимой.

Из вновь введенных понятий основным является полувычислимость, так как вычислимость сводится к нему. Существуют как невычислимые функции, так и функции, являющиеся полувычислимыми, но не вычислимые. Пример такой функции:

$$g(t) = \begin{cases} 1, \text{если уравнение } P(t,x_1,\ldots,x_n) \text{ разрешимо,} \\ \text{не определена в противном случае.} \end{cases}$$

Можно показать, что существует такой многочлен с целыми коэффициентами $P(t, x_1,\ldots,x_n)$, что *g(t)* - невычислима. Однако, легко видеть, что *g(t)* - полувычислима.

Фундаментальным открытием теории вычислимости явился, так называемый, тезис Черча, который в слабейшей форме имеет следующий вид: можно явно указать а) семейство простейших полувычислимых функций; б) семейство элементарных операций, которые позволяют строить по одним полувычислимым функциям другие полувычислимые функции с тем свойством, что любая полувычислимая функция получается за конечное число шагов, каждый из которых состоит в применении одной из элементарных операций к ранее построенным или к простейшим функциям.

Простейшие функции:

suc: $N \to N$; suc$(x) = x+1$ - определение следующего за *x* числа;

$l^{(n)}$: $(N)^n \to N$; $l^{(n)}(x_1,\ldots, x_n) = 1$, $n \geq 0$ - определение «размерности» области определения функции;

$pr_i^n : (N)^n \to N$; $pr_i^n(x_1,\ldots, x_n) = x_i$, $x \geq 1$ - «проекция» области определения на одну из переменных.

Элементарные операции над частичными функциями:

а) *композиция* (или подстановка) ставит в соответствие паре функций *f* из $(N)^m$ в $(N)^n$ и *g* из $(N)^n$ в $(N)^p$ функцию $h = g_o f$ из $(N)^m$ в $(N)^p$ , которая определяется как

$$D(g \circ f) = f^{-1}(D(g)) = \{x(N)^m \, x D(f), f(x) D(g)\} \; (g \circ f)(x) = g(f(x));$$

*б) соединение* ставит в соответствие частичным функциям $f_i$ из $(N)^{ni}$, $i = 1, ..., k$ функцию ($f$i, ..., $f_k$) из $(N)^m$ в $(N)^{n1}$ x... x $(N)^{nk}$, которая определяется как

$$D((f_1, ..., f_k)) = D(f_1) \bigcap \ldots \bigcap D(f_k),$$
$$(f_1, ..., f_k)(x_1, \ldots, x_m) = \langle f_1(x_1, \ldots, x_m), \ldots, f_k(x_1, ..., x_m) \rangle;$$

в) *рекурсия* ставит в соответствие паре функций $f$ из $(N)^n$ в $N$ и $g$ из $(N)^{n+2}$ в $N$ функцию $h$ из $(N)^{n+2}$ *в N,* которая определяется рекурсией по последнему аргументу

$h(x_1, \ldots, x_n, 1) = f(x_1, \ldots, x_n)$ (начальное условие),
$h(x_1, \ldots, x_n, k+1) = g(x_1, \ldots, x_n, k, h(x_1, \ldots, x_n, k))$ при $k \geq 1$ (рекурсивный шаг).

Область определения $D(h)$ описывается также рекурсивно:

$$\langle x_1, \ldots, x_n, 1 \rangle D(h) \leftrightarrow \langle x_1, \ldots, x_n \rangle D(h);$$
$$\langle x_1, \ldots, x_n, k+1 \rangle D(h) \leftrightarrow \langle x_1, \ldots, x_n, k \rangle D(h);$$
$$\langle x_1, \ldots, x_n, k, h(x_1, \ldots, x_n, k) \rangle D(g) \text{ при } k \geq 1;$$

г) *операция m,* которая ставит в соответствие частичной функции $f$ из $(N)^{n+1}$ в $N$ частичную функцию $h$ из $(N)^n$ в $N$, которая определяется как

$$D(h) = \{\langle x_1, \ldots, x_n, k \rangle \exists x_{n+1} \geq 1,$$
$$f(x_1, \ldots, x_n, x_{n+1}) = 1 \text{ и } \langle x_1, \ldots, x_n, k \rangle D(f) \text{ для всех } k \leq x_{n+1}\},$$
$$h(x_1, \ldots, x_n) = \min\{x_{n+1} f(x_1, \ldots, x_n, x_{n+1}) = 1\}.$$

Операция $m$ позволяет вводить в вычисления перебор объектов для отыскания нужного в бесконечном семействе.

Теперь, когда введены простейшие функции и элементарные операции, можно дать следующие основные определения:

а) последовательность частичных функций $f$i, . . . ,$f_N$ называют частично рекурсивным (соответственно примитивно рекурсивным) описанием функции $f_N = f$, если $f$i - одна из простейших функций; $f_i$ для всех $i \geq 2$ либо является простейшей функцией, либо получается применением одной из элементарных операций к некоторым из функций $f$i,..., $f_{i-1}$ (соответственно одной из элементарных операций, кроме *m);*

б) функция $f$ называется частично рекурсивной (соответственно примитивно рекурсивной), если она допускает частично рекурсивное (соответственно примитивно рекурсивное) описание.

Теперь можно привести тезис Черча в обычной форме:

а) функция $f$ полувычислима, если и только если она частично рекурсивна;

б) функция $f$ вычислима, если и только если рекурсивны $f$ и характеристическая функция $X_{D(f)}$.

Характеристическая функция подмножества $X$ в $Y (X \subset Y)$ есть такая функция, что

$$\chi(x) = \begin{cases} 1, \text{ если } x \in X, \\ 0, \text{ если } x \in Y - X. \end{cases}$$

Тезис Черча может использоваться как определение алгоритмической неразрешимости.

Пусть имеется счетная последовательность «задач» $P_1$, $P_2$, ..., которые имеют ответ «да» или «нет». Такая последовательность носит название «массовой проблемы». Свяжем с ней функцию $f$ из $N$ в $N$:

$$D(f) = \{i \mid P_i \text{ имеет ответ «да»}\};$$

$$f(i) = 1, \text{ если } i \in D(f).$$

Массовая проблема $P$ называется алгоритмически разрешимой, если функции $f$ и $X_{D(f)}$ частично рекурсивны. В противном случае $P$ называется алгоритмически неразрешимой.

***Контрольные вопросы и задания***

1. Для чего необходимо формализовать понятие алгоритма?
2. Что означает фраза: «Машины Поста и Тьюринга являются абстрактными машинами»?
3. Для чего предназначены машины Поста и Тьюринга?
4. Как «устроена» машина Поста?
5. Перечислите и запишите команды машины Поста.
6. С помощью бумаги, карандаша и стиральной резинки «исполните» вместо машины Поста программы сложения чисел из текста.
7. Составьте (и проверьте) программу для машины Поста, создающую на ленте копию заданной последовательности меток справа от нее.
8. Пользуясь предыдущей программой, составьте программу умножения чисел для машины Поста.
9. Как «устроена» машина Тьюринга?
10. Каков принцип исполнения программы машиной Тьюринга?
11. Сравните машины Поста и Тьюринга. Укажите различия.
12. Выполните вместо машины Тьюринга примеры программ из текста.
13. Каким образом могут быть обобщена машина Тьюринга?
14. Что такое ассоциативное исчисление?
15. Постройте дедуктивную цепочку от слова «мука» к слову «торт», заменяя каждый раз по одной букве так, чтобы каждый раз получалось слово.
16. Дайте определение нормального алгоритма Маркова.
17. В чем состоит принцип нормализации алгоритмов?
18. Охарактеризуйте способы композиции нормальных алгоритмов.
19. Как алгоритм может быть связан с рекурсивной функцией?
20. Дайте определения частичной, полувычислимой и вычислимой функции.
21. В чем состоит тезис Черча в слабейшей и в обычной формах?
22. Перечислите простейшие функции.
23. Перечислите элементарные операции.
24. Чем отличается рекурсивная функция от примитивно-рекурсивной?
25. Дайте определение частично-рекурсивной функции.
26. Что называется массовой проблемой? Что означает алгоритмическая разрешимость массовой проблемы?

# § 8. ПРИНЦИПЫ РАЗРАБОТКИ АЛГОРИТМОВ И ПРОГРАММ ДЛЯ РЕШЕНИЯ ПРИКЛАДНЫХ ЗАДАЧ

## 8.1. ОПЕРАЦИОНАЛЬНЫЙ ПОДХОД

В настоящее время создание алгоритмов - написание программ для электронных вычислительных машин - стало видом человеческой деятельности. Важнейший конструктивный компонент программирования, не зависящий от особенностей синтаксиса языков программирования и специфики функционирования конкретных вычислительных машин, - разработка алгоритма.

Подходы к созданию алгоритмов и требования к ним существенно изменялись в ходе эволюции компьютеров. Первоначально, в эпоху ЭВМ 1 -го и 2-го поколений, когда они были еще мало распространены, машинное время было дорого, а возможности ЭВМ очень скромны (с точки зрения сегодняшних достижений), основным требованием к алгоритму была его узко понимаемая эффективность:

1) минимальные требования в отношении оперативной памяти компьютера -.программа должна была использовать наименьшее возможное число ячеек оперативной памяти компьютера;

2) минимальное время исполнения (минимальное число операций). При этом программы составлялись из команд, непосредственно или почти непосредственно исполнявшихся компьютером (точнее говоря, процессором):

• операции присваивания;

• простейших арифметических операций;

• операций сравнения чисел;

• операторов безусловного и условных переходов (изменяющих порядок вычисления команд в программе);

• операторов вызова подпрограмм (вспомогательных алгоритмов).

Такой подход в программировании (создании алгоритмов), ориентированный на непосредственно выполняемые компьютером операции, можно назвать операциональным.

Рассмотрим подробнее операции, которые выполняются компьютером и которые являются шагами программы при операциональном подходе.

Операция присваивания состоит в том, что некоторое значение фигурирующей в программе величины помещается в ячейку памяти компьютера. Эта ячейка может либо принадлежать оперативной памяти, либо находиться в арифметико-логическом устройстве, выполняющем основные операции (это устройство - часть процессора). После операции присваивания указанное значение сохраняется в ячейке памяти, куда оно было помещено, пока не будет заменено другим в результате другого присваивания. Ячейка памяти, где размещается значение, в программе обозначается именем (идентификатором) соответствующей переменной. Примеры идентификаторов: *a, x, $y_1$, $y_2$*. Важно помнить, что переменные и, соответственно, их значения могут быть разных типов - числовые (целые или действительные), литерные или логические. Значения различных типов представляются (кодируются) в компьютере по-разному, поэтому они должны соответствовать типам переменных, которым они присваиваются. При разработке алгоритма следует всегда помнить и 'тщательно различать типы переменных.

Набор простейших арифметических операций «сложения» (+), «вычитания» (-), «умножения» (*) и «деления» (/) (причем во многих случаях следует тщательно отличать деление, выполняемое над целыми числами - в этом случае операция деления распадается на деление нацело и вычисление остатка от деления) позволяет записывать арифметические выражения с использованием числовых констант и идентификаторов переменных. Для определения порядка операций в выражениях чаще всего используют стандартное математическое соглашение о старшинстве операции, согласно которому старшими и выполняемыми в 1-ю очередь являются умножение и деление, а младшими - сложение и вычитание. Для изменения «естественного» порядка выполняемых операций служат скобки. Сравните, например, порядок операций в выражениях:

$$(a + 2) * x \text{ и } a + 2 * x.$$

Что же касается порядка выполнения операций одного старшинства, то они, как правило, выполняются в порядке записи в выражении.

Операция сравнения числовых значений фактически сводится к определению знака разности этих значений. Этот знак отображается с помощью специальной ячейки памяти (флага знака результата) вычислительного устройства компьютера и может использоваться при выполнении условных переходов между командами (шагами) алгоритма.

Чтобы понять, что такое условные и безусловные переходы при выполнении алгоритма, надо исходить из того, что шаги или команды алгоритма обладают метками или адресами, и, помимо естественного порядка выполнения команд соответственно их записи, возможен и другой порядок, при котором последовательность выполнения команд определяется переходами на команды с определенными метками или адресами. **Безусловным** называется переход, для которого изменение порядка выполнения команд раз и навсегда определено и не зависит ни от каких условий. Условным называется переход, для которого порядок выполнения команд определяется по некоторому условию, чаще всего условию сравнения величин числовых типов.

Операция вызова подпрограммы (вспомогательного алгоритма) - это такой переход в последовательности команд алгоритма, при котором на определенном этапе выполнения алгоритма

происходит вначале переход на другую программу (подпрограмму по отношению к той, откуда совершен переход), а затем, после ее завершения, возврат в точку вызова подпрограммы и продолжение выполнения команд, начиная со следующей после команды вызова подпрограммы, в-их естественном порядке. Очевидно, что операция вызова подпрограммы представляет собой переход, при котором запоминается адрес команды, следующей за командой вызова подпрограммы, что позволяет вернуться к исходному алгоритму (головной программе) после выполнения вспомогательного алгоритма (подпрограммы).

Отметим, что универсальность существующих компьютеров достигается за счет определенного набора команд, типа только что описанного, и автоматического механизма их выполнения, а проблема решения задачи с помощью компьютера состоит лишь в преобразовании решаемой задачи в последовательность этих команд. В качестве примера рассмотрим операциональное представление алгоритма вычисления квадратного корня из положительного числа *a* с помощью рекуррентной формулы:

$$x_{n+1} = (x_n + a/x_n)/2,$$

$$n = 0,1,2,...$$

Можно показать, что $\lim_{n\to\infty} x_n = \sqrt{a}$ .

Будем обозначать через $x_0$ нулевое приближение (за него в данном случае можно принять любое положительное число), через $\varepsilon$ заданную точность вычислений и через $c_0$ какое-нибудь число, удовлетворяющее условию $0 < c_0 < \sqrt{a}$ , необходимое для оценки достигнутой точности с помощью неравенства

$$x_n < \sqrt{a} < \frac{x_n^2 - a}{2c_0}.$$

Алгоритм вычисления $\sqrt{a}$ .

1) ввести числа *a,* $\varepsilon$ $x_0$, $c_0$;
2) присвоить переменной *x* значение *y,*
3) присвоить переменной *y* значение *a/x;*
4) присвоить переменной *y* значение *x* + *y,*
5) присвоить переменной *x* значение *y/2;*
6) присвоить переменной *y* значение $x^2$;
7) присвоить переменной *y* значение *y-a;*
8) присвоить переменной *y* значение $y/c_0$;
9) присвоить переменной $\delta$ значение *y/2;*
10) сравнить $\delta$ и $\varepsilon$; если $\delta > \varepsilon$ то перейти к команде 3, иначе перейти к следующей команде;
11) вывести числа *x, a* и $\varepsilon$;
12) стоп.

В этом алгоритме все команды, кроме 10, предполагают переход к следующим за ними по записи командам, и лишь команда 10, являющаяся командой условного перехода, меняет порядок исполнения команд - после нее в нарушение порядка может выполняться команда 3, т.е. она определяет циклическую конструкцию в алгоритме.

Поясним эту программу. Команда 2 помещает значение начального приближения $x_0$ в ячейку памяти, в которой хранятся значения переменной *x* (на каждом этапе вычислений в этой ячейке хранится значение *x,* равное значению одного из членов рекуррентной последовательности $x_n$).

Команды 3-5 вычисляют по числу *x* число *(x + a/x)/2.* Результат помещается в ячейку памяти, в которой хранится значение переменной *x,* при этом старое значение «стирается» новым. Команды 6-9 вычисляют величину

$$\delta = (x^2 - a)/2c_0,$$

с помощью которой оценивается сверху разность $x - \sqrt{a}$ . Важное значение имеет команда

10. По ней не производятся вычисления, а сравниваются между собой вычисленное значение 5 и заданная точность $\varepsilon$ Если $\delta > \varepsilon$ то управляющее устройство вернет вычислительный процесс к команде 3 и заставит повторить процесс.

В противном случае, когда требуемая точность достигнута, печатается полученный результат и работа прекращается.

Данный алгоритм весьма экономичен: в качестве рабочих он использует всего две ячейки памяти (для переменных *x* и *y),* его команды так продуманы, что никакие операции не выполняются с избыточным повторением.

В данном примере не были использованы какие-либо специальные обозначения команд, чтобы сделать их независимыми от языка конкретных ЭВМ (такие языки называют Ассемблерами), чтобы стал ясен общий характер операционального подхода к разработке алгоритмов. Однако, ориентированность этого подхода на возможности и особенности ЭВМ с появлением большого числа компьютеров 3-го и особенно 4-го поколений не позволяла перейти к массовому промышленному программированию и стала сдерживать развитие вычислительной техники. Отметим основные недостатки алгоритмов, к которым приводил операциональный подход:

• злоупотребление командой условного и безусловного переходов зачастую приводило к очень запутанной структуре программы, напоминавшей по образному сравнению «блюдо спагетти»;

• вместе с разнообразными уловками, направленными на повышение эффективности программы (т.е. минимальных требований к оперативной памяти и минимального времени выполнения), это приводило к непонятности программ, их ненадежности, трудностям в отладке и модификации, делая программирование трудоемким, сложным и чрезвычайно дорогостоящим.

Необходимость ориентироваться на ограниченный набор команд компьютера, на его скромные возможности приводила к огромной трудоемкости, к сложности программ, к проблемам, связанным с ошибками в них. В результате узким местом в развитии вычислительной техники оказалось именно программирование.

## 8.2. СТРУКТУРНЫЙ ПОДХОД

С появлением массовых ЭВМ 3-го поколения устаревшая технология программирования оказалась основным фактором, сдерживающим развитие и распространение компьютерных (информационных) технологий, что подтолкнуло ведущие в этой сфере деятельности фирмы, в первую очередь IBM, к разработке новых методологий программирования. Появившийся в начале 1970-х годов новый подход к разработке алгоритмов получил название структурного.

С появлением структурного программирования описанные выше трудности были во многом преодолены. В основе технологических принципов структурного программирования лежит утверждение о том, что логическая структура программы может быть выражена комбинацией трех базовых структур: следования, ветвления и цикла (это содержание теоремы Бема-Якопини).

**Следование** - самая важная из структур. Она означает, что действия могут быть выполнены друг за другом, рис. 1.19:

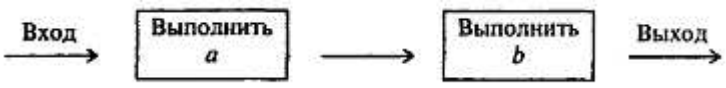



*Рис. 1.19.* Структура «следование»

Эти прямоугольники могут представлять как одну единственную команду, так и множество операторов, необходимых для выполнения сложной обработки данных.

**Ветвление** - это структура, обеспечивающая выбор между двумя альтернативами. Выполняется проверка, а затем выбирается один из путей (рис. 1.20).

Эта структура называется также «ЕСЛИ - ТО - ИНАЧЕ», или «развилка». Каждый из путей (ТО или ИНАЧЕ) ведет к общей точке слияния, так что выполнение программы продолжается независимо от того, какой путь был выбран.

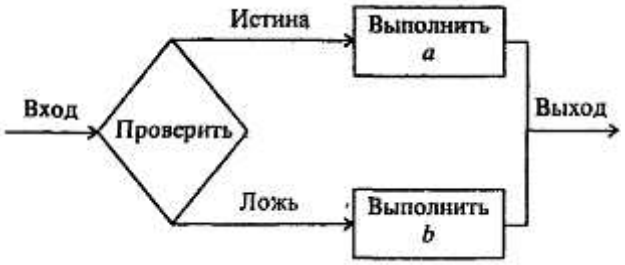



*Рис. 1.20.* Структура «ветвление»

Может оказаться, что для одного из результатов проверки ничего предпринимать не надо. В этом случае можно применять только один обрабатывающий блок, рис.1.21:

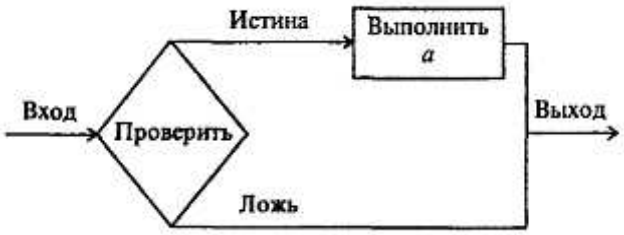



*Рис. 1.21.* Структура «неполное ветвление»

Цикл (или повторение) предусматривает повторное выполнение некоторого Набора команд программы. Если бы циклы не существовали, вряд ли занятие программированием было бы оправданным: циклы позволяют записать длинные последовательности операций обработки данных с помощью небольшого числа повторяющихся команд. Разновидности цикла изображены на рис. 1.22 и рис. 1.23.

Цикл начинается с проверки логического выражения. Если оно истинно, то выполняется «*a*», затем все повторяется снова, пока логическое выражение сохраняет значение «истина». Как только оно становится ложным, выполнение операций «а» прекращается и управление передается по программе дальше.

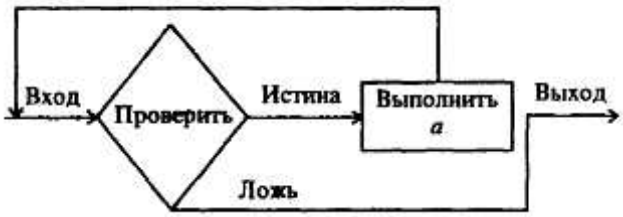



*Рис. 1.22.* Структура цикла «пока»

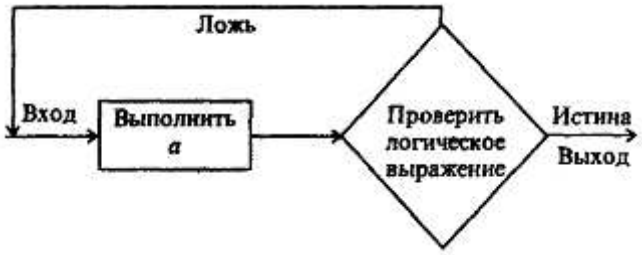



*Рис. 1.23.* Структура цикла «до»

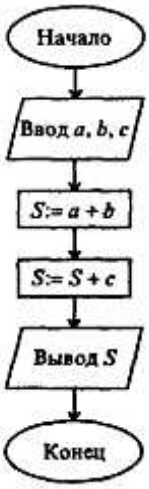



*Рис. 1.24.* Нахождение суммы трех чисел

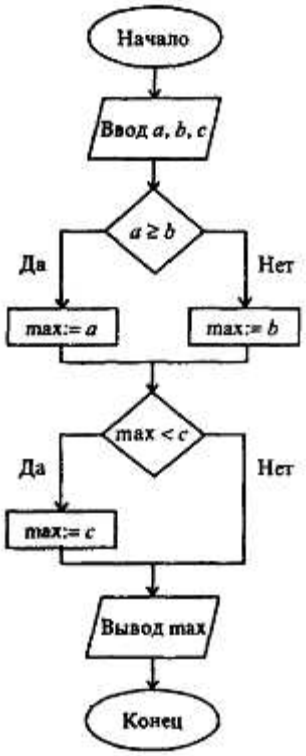



*Рис. 1.25.* Нахождение наибольшего из трех чисел

Эти структуры можно комбинировать одну с другой - как путем организации их следований, так и путем создания суперпозиций (вложений одной структуры в другую) - сколь угодно разнообразно для выражения логики алгоритма решения любой задачи. Используя описанные структуры, можно полностью исключить использование каких-либо еще операторов условного и безусловного перехода, что является важным признаком структурного программирования. Направление выполнения команд часто изображают сверху вниз. На рис. 1.24 - 1.26 приведены простейшие примеры структурной реализации алгоритмов работы с величинами.

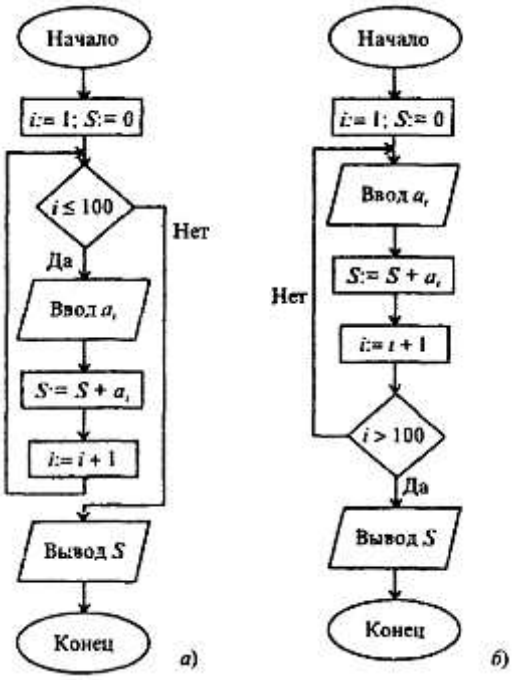



*Рис. 1.26.* Нахождение суммы 100 чисел

Умение образовывать из базовых структур их суперпозиции в соответствии с условиями конкретной задачи - одно из важнейших в программировании. Допустим, надо ввести в память компьютера 100 чисел и по дороге отсуммировать те из них, которые положительны. Ясно, что ввод - операция циклическая, а внутри этого цикла находится развилка, в которой проверяется знак числа и производится суммирование. Схематически соответствующая суперпозиция изображена на рис. 1.27.

Так как выражение, управляющее циклом, проверяется в самом начале, то в случае, если условие сразу окажется ложным, операторы циклической части «*a*» могут вообще не выполняться. Операторы циклической части «*a*» должны изменять переменную (или переменные), влияющие на значение логического выражения, иначе программа «зациклится» - будет выполняться бесконечно. Рассмотренная циклическая конструкция называется также цикл «пока», или «цикл с предусловием».

Существует и иная конструкция цикла, которая предусматривает проверку условия, по которому, наоборот, выполнение команд циклической части прекращается, после команд циклической части (см. рис. 1.23).

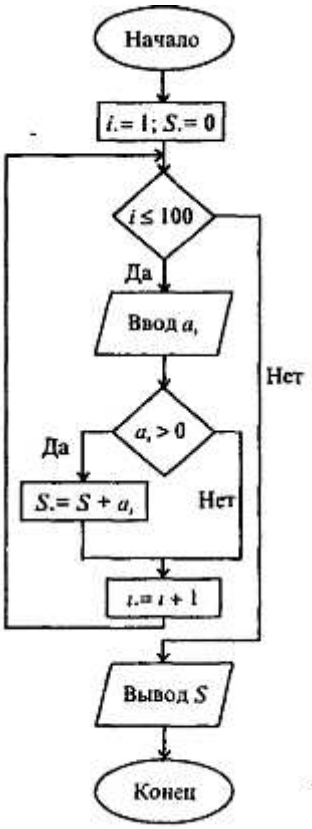



*Рис 1.27* Алгоритм типа развилка, вложенная в цикл,
для нахождения суммы положительных чисел из 100 возможных

Схематические изображения нескольких суперпозиций базовых алгоритмических структур представлены ниже на рис. 1.28-1.31.

Еще одним важным компонентом структурного подхода к разработке алгоритмов является модульность. Модуль - это последовательность логически связанных операций, оформленных как отдельная часть программы. Использование модулей имеет следующие преимущества:

1) возможность создания программы несколькими программистами;

2) простота проектирования и последующих модификаций программы;

3) упрощение отладки программы - поиска и устранения в ней ошибок;

4) возможность использования готовых библиотек наиболее употребительных модулей.

Но, пожалуй, самым важным достижением структурного подхода к разработке алгоритмов является нисходящее проектирование программ, основанное на идее уровней абстракции, которые становятся уровнями модулей в разрабатываемой программе. На этапе проектирования строится схема иерархии, изображающая эти уровни. Схема иерархии позволяет программисту сначала сконцентрировать внимание на определении того, что надо сделать в программе, а лишь затем решать, как это надо делать. При нисходящем проектировании исходная, подлежащая решению задача разбивается на ряд подзадач, подчиненных по своему содержанию главной задаче. Такое разбиение называется детализацией или декомпозицией.

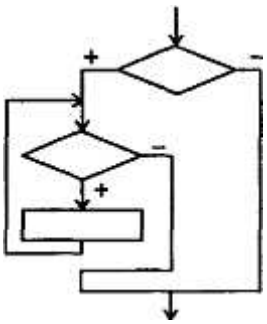



*Рис. 1.28.* Алгоритм типа «цикл,
вложенный в неполную развилку»

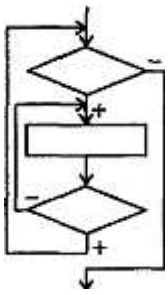



*Рис. 1.29.* Алгоритм типа «цикл в цикле»

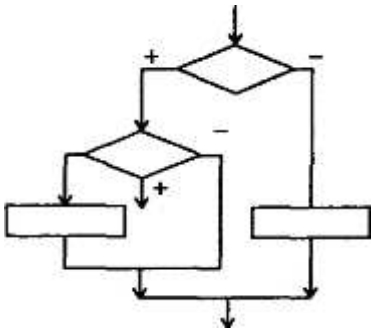


*Рис. 1.30.* Алгоритм типа «развилка в развилке»

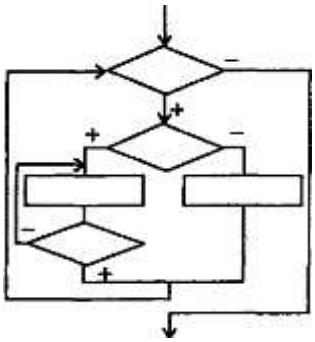


*Рис. 1.31.* Иллюстрация трехкратного вложения одной базовой структуры в другую

На следующем этапе эти задачи, в свою очередь, разбиваются на более мелкие подчиненные подзадачи и так далее, до уровня относительно небольших подзадач, вторые требуют для решения небольших модулей в 3 - 5 строк. Такой метод роектирования программ позволяет преодолевать проблему сложности разработки программы (и ее последующей отладки и сопровождения).

## 8.3. НОВЕЙШИЕ МЕТОДОЛОГИИ РАЗРАБОТКИ ПРОГРАММ ДЛЯ ЭВМ

Структурный подход сыграл огромную роль в программировании и вычислительной технике. С его использованием создан большой запас программного обеспечения, решено множество практически значимых задач. Однако, развитие программирования на этом не остановилось. Сегодня дополняющим структурное программирование, создающим основу для разработки современных аудиовизуаль-программных комплексов стало объектное (иногда говорят объектно-ориентированное) программирование, а противостоящим ему при решении определенных классов задач является декларативное программирование, выраженное двумя разными подходами - функциональным и логическим.

Здесь мы ограничимся кратчайшей характеристикой этих направлений, более детально описанных в гл. 3.

Само структурное программирование, наиболее отчетливо выраженное в языке Паскаль (PASCAL), возникло в ходе развития процедурно-ориентированного подхода, заложенного в исторически первом из языков программирования - Фортране (FORTRAN). Во всех языках этого направления разработчик алгоритма (он же, как правило, и программист) описывает, какими действиями следует реализовать процесс. В основе языков этой группы лежат понятия команд (операторов) и данных. Отдельные группы операторов могут объединяться во вспомогательные алгоритмы (процедуры, подпрограммы).

Объект - основное понятие объектного программирования - в первом приближении похож на процедуру. Однако, процедура (подпрограмма) «оживает» лишь внутри той программы, к которой она относится, а объект может вести себя вполне независимо. Он может относиться к иной предметной области нежели основная программа, быть исполненным в ином стиле. Объекты достаточно причудливо связываются друг с другом, могут перенимать свойства друг у друга («наследование»). В объектно-ориентированном подходе исходная задача представляется как совокупность взаимодействующих объектов. Наиболее популярные реализации объектного программирования созданы на основе языков Паскаль, Бейсик (BASIC).

Декларативный подход в разработке компьютерных программ появился в начале 70-х годов. Он не получил столь широкого применения как процедурный, поскольку был направлен на относительно узкий круг задач искусственного интеллекта. При его применении программист описывает свойства исходных данных, их взаимосвязи, свойства, которыми должен обладать результат, а не алгоритм получения результата. Разумеется, для получения результата этот алгоритм все равно нужен, но он должен порождаться автоматически той системой, которая поддерживает декларативно-ориентированный язык программирования. При логическом варианте такого подхода (прежде всего это относится к языку Пролог, PROLOG) задача описывается совокупностью фактов и правил в некотором формальном логическом языке, при функциональном варианте - в виде функциональных соотношений между фактами (язык Лисп, LISP).

Процедурно-ориентированное программирование развивается и в другом направлении - так называемого, параллельного программирования. В привычных алгоритмах и программах действия

совершаются последовательно одно за другим. Однако, логика решения множества задач вполне допускает одновременное выполнение нескольких операций, что ведет к многократному увеличению эффективности. Реализация параллельных алгоритмов на ЭВМ стала возможной с появлением многопроцессорных компьютеров, в которых специалисты видят будущее вычислительной техники.

***Контрольные вопросы и задания***

1. Какие требования предъявлялись к алгоритмам для компьютеров первых поколений?
2. Какой подход к созданию алгоритмов называется операциональным?
3. Охарактеризуйте операции, которые использовались при разработке программ при операциональном подходе.
4. Проделайте программу вычисления $\sqrt{x}$ из текста с помощью ручки, бумаги и калькулятора.
5. В чем состоят недостатки операционального подхода к программированию?
6. Охарактеризуйте базовые структуры алгоритмов.
7. В чем состоит модульность при структурной разработке алгоритмов?
8. Что такое нисходящее проектирование программ?

# § 9. СТРУКТУРЫ ДАННЫХ

## 9.1. ДАННЫЕ И ИХ ОБРАБОТКА

Суть всех алгоритмов (и компьютерных программ) состоит в том, что они описывают преобразование некоторых начальных данных в конечные. Какие-то данные алгоритм (программа) может использовать как промежуточные. Естественно, что представление и организация данных имеет при разработке алгоритма первостепенное значение. Вопрос о том, как должны быть организованы данные, необходимо решать до того, как начинается разработка алгоритма (программы). Ведь прежде, чем выполнить какие-либо операции, надо иметь объекты, к которым они будут применяться, и четко представлять себе структуру объектов, которые будут получены.

Развитие вычислительной техники и программирования сопровождалось эволюцией представлений о роли данных и их организации. Одним из свойств компьютеров является способность хранить огромные объемы информации и обеспечивать легкий доступ к ней. Информация, подлежащая обработке, в некотором смысле представляет абстракцию фрагмента реального мира. Мы говорим о данных как об абстрактном представлении реальности, поскольку некоторые свойства и характеристики реальных объектов при этом игнорируются (как несущественные ддя данной задачи). Например, каждый сотрудник в списке сотрудников некоторого учреждения представлен множеством данных. Это множество может включать идентифицирующие данные (например, фамилию), данные, относящиеся к тому, что сотрудник делает или к тому, что делают для него. Однако маловероятно, что будут включены такие сведения, как цвет глаз или волос, рост и вес.

Решая конкретную задачу, необходимо выбрать множество данных, представляющих реальную ситуацию. Затем надлежит выбрать способ представления этой информации. Представление данных определяется исходя из средств и возможностей, допускаемых компьютером и его программным обеспечением. Однако очень важную роль играют и свойства самих данных, операции, которые должны выполняться над ними. С развитием вычислительной техники и программирования средства и возможности представления данных получили большое развитие и теперь позволяют использовать как простейшие неструктурированные данные, так и данные более сложных типов, полученные с помощью комбинации простейших данных. Такие данные называют **структурированными,** поскольку они обладают некоторой организацией. Современные средства программирования позволяют оперировать с множествами, массивами, записями, файлами (очередями).

В более сложных случаях программист может задать динамические структуры данных, память для хранения которых выделяется прямо в процессе выполнения программы. К таким данным относят линейные списки (одно- и двунаправленные), стеки, деревья, графы.

В последние годы получило развитие, так называемое, объектно-ориентированное про-

граммирование, в котором в известной мере устранено противостояние . данных и программ. Объект - это некое образование, состоящее не только из данных, но и из процедур их обработки.

Остановимся подробнее на свойствах различных представлений данных или, как | еще говорят, типах данных.

## 9.2. ПРОСТЫЕ (НЕСТРУКТУРИРОВАННЫЕ) ТИПЫ ДАННЫХ

В математике принято классифицировать величины в соответствии с их характеристиками. Различают целые, вещественные, комплексные и логические величины, величины, представляющие собой отдельные значения, множества значений или множества множеств. Аналогично этому в информатике любая константа, переменная, выражение или функция относится к некоторому типу. Фактически тип характеризует множество значений, которые может принимать константа, переменная, выражение или функция. Широко используется правило, по которому тип явно указывается в описании константы, переменной или функции. К данным каждого типа применимы определенные операции и их поведение подчиняется некоторым аксиомам.

Так, над целыми числами могут выполняться операции сложения (+), вычитания (-) и умножения (*). Существуют две различные операции, связанные с делением и не выводящие за границы множества целых чисел: 1) определение целой части от деления одного числа на другое; 2) определение остатка от деления одного числа на другое.

Целые числа, используемые компьютером, имеют те же свойства (подчиняются тем же аксиомам), что и целые числа в арифметике. Все вычисления с ними выполняются абсолютно точно (не приближенно). Имеется только одно отличие в свойствах компьютерных целых чисел от тех, которыми оперирует абстрактная математика, а именно ограниченность диапазона: для каждой компьютерной системы имеется самое большое допустимое в ней целое число $M_{+\infty}$ (и самое малое, отрицательное $M_{-\infty}$). Обычно выполняется соотношение

$$M_{+\infty} + 1 = M_{-\infty} \; (M_{-\infty} - 1 = M_{+\infty}),$$

т. е. прибавив единицу к самому большому допустимому положительному числу, мы получим модуль самого малого допустимого отрицательного. Это свойство компьютерных чисел связано с особенностями их кодирования в ячейках памяти компьютера.

Над действительными (или вещественными) числами могут быть выполнены операции сложения (+), вычитания (-), умножения (*) и деления (/), так же, как и над математическими действительными числами. Однако, все операции над действительными числами выполняются с точностью, не превосходящей некоторого фиксированного значения, вследствие того, что представления чисел в памяти компьютера имеют ограниченную длину (зависящую от конкретного компьютера и используемой системы программирования). Так, например, в машинной арифметике может быть 1/3 = 0,33333333, тогда как математически точное десятичное представление дроби 1/3 - бесконечно длинное число.

Главным свойством литерных (символьных) данных является их упорядоченность, т.е. свойство быть сравнимыми. Обычным признаком значения символьной или текстовой величины являются кавычки, и справедливо 'a' < *'b'*, *'b'* < *'c'*, *'c'* < *'d'* и т.д. Каждый символ имеет определенный числовой код (например, код символа латинской буквы "A" в большинстве кодировок 63) и упорядочение происходит в соответствии с этими числовыми кодами. Как правило, имеются функции, позволяющие получить по символу его код и символ по коду.

Для строчных величин единственной выполнимой операцией является конкатенация («сложение») строк. Например, *'abcd'* + *'efg'* = *'abcdefg'*. В конкретных системах обычно определены функции, определяющие длину строк, вхождение в них тех или иных подстрок, вырезающие из строк некоторые фрагменты.

К логическим данным, способным принимать значения «истина» («true») или «ложь» («false»), применимы основные операции логики высказываний: конъюнкция (логическое «и» - «and»), дизъюнкция (логическое «или» - «or»), отрицание (логическое «не» - «not»). Иногда можно использовать операции импликации («если»), эквиваленции («если и только если») и т.п. Эти логические операции определяются таблицей истинности, табл. 1.9.

**Таблица 1.9 Таблица истинности для логических операций**

| A | B | «и» (A and B) | «или» (A and B) | «не» (not A) |
|---|---|---|---|---|
| И | И | И | И | Л |
| И | Л | Л | И | Л |
| Л | И | Л | И | И |
| Л | Л | И | Л | И |

Часто при решении задач оказываются полезными определенные разработчиком типы данных, представляющие собой перечисление некоторых констант или ограниченный с обеих сторон диапазон определенных ранее данных. Например, создается тип данных из двух символов *'a'* и '*b*' или из целых чисел в диапазоне от 1 до 100 и т. д.

## 9.3. СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ

Описанные выше типы данных называют простыми. Основной признак, по которому можно определить величину простого типа, таков: одно имя - одно значение.

Значительно большие возможности заключают в себе структурированные данные, определяемые разработчиком программы (в пределах возможностей используемого им языка программирования), К структурированию данных разработчика программы толкает как логика прикладной задачи, так и чисто утилитарное соображение: при наличии в задаче большого количества входных и выходных данных отдельное именование каждого из них может оказаться практически невозможным.

Разумеется, действия разработчика алгоритма и программы ограничены возможностями того языка программирования, на который он ориентируется. В разных языках возможности структуризации переменных на уровне сложных структур не совпадают, но многие структуры давно стали традиционными и реализованы в большинстве практически используемых языков программирования.

Структурированные типы данных классифицируют по следующим основным признакам: однородная - неоднородная, упорядоченная - неупорядоченная, прямой доступ - последовательный доступ, статическая - динамическая. Эти признаки противостоят друг другу лишь внутри пары, а вне этого могут сочетаться.

Если все элементы, образующие структуру, однотипны (например - целые числа или символы), то структура является однородной; если же в ней «перепутаны» элементы разной природы (например, числа чередуются с символами), то неоднородной.

Структуру называют **упорядоченной,** если, между ее элементами определен порядок следования. Примером упорядоченной математической структуры служит числовая последовательность, в которой у каждого элемента (кроме первого) есть предыдущий и последующий. Наличие индекса в записи элементов структуры уже указывает на ее упорядоченность (хотя индекс для этого не является обязательным признаком).

По способу доступа упорядоченные структуры бывают прямого и последовательного доступа. При прямом доступе каждый элемент структуры доступен пользователю в любой момент независимо от других элементов. Глядя на линейную таблицу чисел мы можем списать или заменить сразу, допустим, десятый элемент. Однако, если эта таблица не на бумаге, а, скажем, каким-то образом записана на магнитофонную ленту, то сразу десятое число нам недоступно - надо сначала извлечь девять предшествующих. В последнем случае мы имеем дело с последовательным доступом.

Если у структуры размер (длина, количество элементов) не может быть изменен «на ходу», а фиксирован заранее, то такую структуру называют статической. Программные средства информатики иногда позволяют не фиксировать размер структуры, а устанавливать его по ходу решения задачи и менять при необходимости, что бывает очень удобно. Такую структуру называют динамической. Например, при описании закономерностей движения очереди в магазине мы не знаем заранее, сколько человек в ней будет в тот или иной момент, и соответствующую структуру данных (например, список фамилий участников очереди) лучше представлять динамической.

**Массивы**

Самым традиционным и широко известным из структурированных типов данных является **массив** (иначе называемый регулярным типом) - однородная упорядоченная статическая структура прямого доступа.

Массивом называют однородный набор величин одного и того же типа, называемых компонентами массива, объединенных одним общим именем (**идентификатором**) и идентифицируемых (адресуемых) вычисляемым **индексом.** Это определение подчеркивает, что все однотипные компоненты массива имеют одно и то же имя, но различаются по индексам, которые могут иметь характер целых чисел из некоторого диапазона, литер, перечисленных констант. Индексы позволяют адресовать компоненты массива, т.е. получить доступ в произвольный момент времени к любой из них как к одиночной переменной (рис. 1.32). Обычный прием работы с массивом - выборочное изменение отдельных его компоневт.

Вычисляемые индексы позволяют использовать единое обозначение элементов массива для описания массовых однотипных операций в циклических конструкциях программ. Важной особенностью массива является его статичность. Массив должен быть описан в программе (т.е. определены тип и число компонент) и его характеристики не могут быть изменены в ходе выполнения программы.

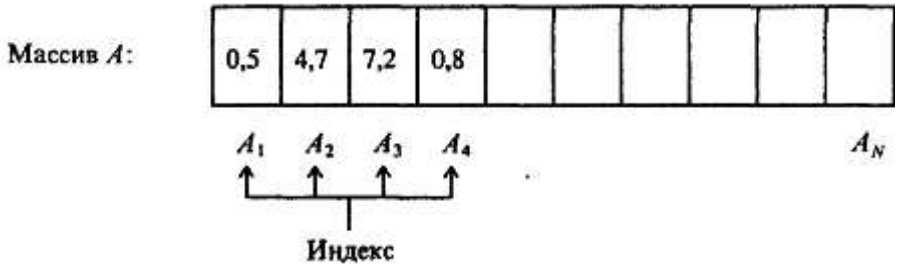



*Рис. 1.32.* Одномерный массив - набор элементов (компонентов)

Компонентами массива могут быть не только простейшие данные, но и структурные, в том числе массивы. В этом случае мы получаем массив массивов - многомерный **массив.** Для индексации элементарных компонент в этом случае может потребоваться два, три и более индексов.

В некоторых системах программирования существуют специальные виды массивов. Например, массив литер (символов) определяется как строка.

Данные, хранящиеся в массивах, находятся в оперативной памяти компьютера. Это, с одной стороны, ускоряет доступ к ним в ходе решения задачи, а с другой -налагает ограничения на объем возможной информации, организованной в виде массивов. Не следует поэтому, без крайней необходимости, создавать новые массивы для перемещения данных из уже существующих массивов.

Рассмотрим в качестве примера задачу сортировки набора некоторых данных, для которых имеют смысл отношения «больше» или «меньше». Представьте себе, что надо карточки в картотеке разместить в порядке возрастания записанных на них чисел. Используем для сортировки набора чисел (т.е. записи их в порядке возрастания) одномерный (линейный) массив. Дадим ему имя *A,* тогда $a_1$, $a_2$, $a_3$,..., $a_n$ - компоненты массива.

Существует огромное число методов сортировки массивов. Рассмотрим один из самых простых (но не самых быстрых) - метод выбора.

В начале процесса имеем заполненный числами массив (неотсортированный). Процесс сортировки строится по индукции. Допустим, мы уже отсортировали часть массива и имеем упорядоченную последовательность

$$a_1 < a_2 < \ldots < a_{i-1}$$

и оставшуюся неотсортированной последовательность

$$a_i, a_{i+1}, \ldots a_N.$$

При каждом шаге, начиная с $i = 1$, из неотсортированной части последовательности извле-

кается наименьший элемент $x = a_i$, и меняется местами с $i$-м элементом. Затем этот процесс повторяется для $i = 2$, $i = 3$ и т.д., до тех пор пока не останется один, самый большой элемент.

Этот алгоритм потребует многократного нахождения наименьшего элемента массива. Этот «вспомогательный» алгоритм поиска наименьшего среди $a_i, \ldots, a_N$ может быть следующим:

1) фиксируется в качестве значения вспомогательной переменной $m$ первый слева элемент массива: $m = a_i$ (в конце процесса $m$ будет иметь значение наименьшего элемента);

2) выполняется сравнение $m$ *с* элементом массива $a_j$, (начиная с номера $j = i + 1$) и, если $a_j < m$, то $m$ заменяется на $a_j$;

3) далее выполняется сравнение $m$ с очередным элементом массива, т.е. $j$ увеличивается на единицу и шаги 2, 3 выполняются снова, до тех пор пока у не достигнет максимального значения индекса элемента массива.

После выполнения этих предписаний переменная $m$ будет соответствовать наименьшему элементу массива.

Двумерный массив визуально представляется плоской таблицей, табл. 1.10. При наличии одного имени (идентификатора) для всех компонентов каждый из них фиксируется значениями двух индексов, указывающих номер строки и номер столбца, на пересечении которых находится эта компонента.

Рассмотрим пример обработки данных, хранящихся в двумерном массиве. Допустим, что на некоторой территории (например, страны) «квадратно-гнездовым» способом расставлены температурные датчики, и их показания обраны в одном центре (что вполне близко к реальной деятельности метеослужбы). Тогда в таблицу - двумерный массив - попадут значения температуры $t_{ij}$ в сответствующих точках. Требуется, просматривая таблицу построчно, найти те точки (т.е. индексы узлов), между которыми температура принимает некоторое заданное значение $T$.

**Таблица 1.10 Графический образ двумерного массива**

| $i$ \ $j$ | 1 | 2 | 3 | 4 | … |
|---|---|---|---|---|---|
| 1 | $a_{11}$ | $a_{12}$ | $a_{13}$ | $a_{14}$ | … |
| 2 | $a_{21}$ | $a_{22}$ | $a_{23}$ | $a_{24}$ | … |
| 3 | $a_{31}$ | $a_{32}$ | $a_{33}$ | $a_{34}$ | … |
| 4 | $a_{41}$ | $a_{42}$ | $a_{43}$ | $a_{44}$ | … |
| … | … | … | … | ... | … |

Пусть в таблице $n$ строк и $m$ столбцов. Вспомогательным алгоритмом в данной задаче может быть алгоритм поиска нужных узлов в одной строке. Пусть эта строка имеет номер $k$. Алгоритмы записаны без комментариев для самостоятельного разбора.

*Вспомогательный алгоритм (k):*

1) положить $j = 1$;
2) если $t_{k,j} < T < t_{k,j+1}$, то см. п. 2;
3) увеличить $j$ на 1,
4) если $j < m$, то вернуться к п. 2;
5) задача решена, ответ: $(k,j), (k,j + 1)$;
6)конец.

*Основной алгоритм:*

1) положить $k = 1$;
2) выполнить вспомогательный алгоритм ($K$);
3) увеличить $k$ на 1;
4) если $k > n$, то вернуться к п.2;
5)конец.

**Записи, множества, файлы**

Обобщением массива является комбинированный тип данных - **запись**, являющаяся неоднородной упорядоченной статической структурой прямого доступа. Запись есть набор именованных компонент - **полей** (часто разного типа), объединенных одним общим именем и идентифицируемых (адресуемых) с помощью как имени записи, так и имен полей, рис. 1.33.

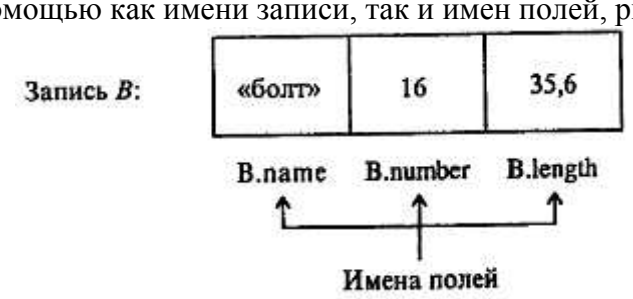



*Рис. 1.33.* Иллюстрация «записи».

Запись *B* состоит из трех полей, имеющих последовательно типы «текст», «целое число», «вещественное число»: 1-е поле - название детали, 2-е - условный номер по каталогу, 3-е - длина. При работе с одной единственной записью (что бывает нечасто), имя поля можно использовать как обычную переменную, т.е. можно изменять значение поля с помощью операции присваивания или любых других операций, доступных над величинами данного типа. Если же данная запись - лишь часть набора данных, то имя поля состоит из двух частей и называется **составным именем поля** (на рис. 1.33 составные имена B. name, B. number, B. length).

Для облегчения работы с полями в различных языках программирования существуют средства, облегчающие их адресацию.

И записи, и массивы обладают одним общим свойством - произвольным доступом к компонентам. Записи более универсальны в том смысле, что для них не требуется идентичности типов их компонент. Массивы обеспечивают большую гибкость -индексы их компонент можно вычислять в отличие от имен полей записей.

Существенно иные возможности дает структура данных, моделирующая свойства мате**ма**тического объекта - множества.

Над множеством могут быть выполнены следующие операции:

1) объединение множеств (операция сложения '+');

2) пересечение множеств (операция умножения '*');

3) теоретико-множественная разность (вычитание множеств '-');

4) проверка принадлежности элемента множеству.

Различия между множеством и массивом очень существенны: размер множества заранее не оговаривается (хотя и ограничен компьютерной реализацией, например, 255), не существует иного способа доступа к элементам множества, кроме как проверкой принадлежности множеству.

Более сложной, чем рассмотренные выше из предусмотренных в современных системах программирования структур данных, является очередь (**файл**).

Понятие «файл» при всей своей привычности употребляется в информатике в нескольких не совсем совпадающих смыслах. Здесь мы остановимся лишь на представлении о файле как однородной упорядоченной динамической структуре последовательного доступа - очереди.

**Очередь** есть линейно упорядоченный набор следующих друг за другом компонент, доступ к которым происходит по следующим правилам:

1) новые компоненты могут добавляться лишь в хвост очереди;

2) значения компонент могут читаться (извлекаться) лишь в порядке следования компонент от головы к хвосту очереди.

Размер очереди заранее не оговаривается и теоретически может считаться бесконечным. Для запоминания (хранения) компонент очереди часто используют внешние запоминающие устройства большой емкости - магнитные диски и ленты. Отсюда другое название очереди - файл (по английски это слово имеет ряд значений, в том числе «картотека», «шеренга», «очередь»).

Исторически слово «файл» стало впервые применяться в информатике для обозначения последовательного набора каких-либо данных или команд (программа), хранящихся на внешнем запоминающем устройстве. Несколько позже были осознаны абстрактные, не зависящие от магнитных дисков и лент, свойства очереди как структуры данных, полезные при решении многих задач обработки - информации. Такой принцип извлечения и добавления компонент к очереди часто;

называется «первым вошел - первым вышел» (английская аббревиатура - «FIFO»), рис. 1.34.

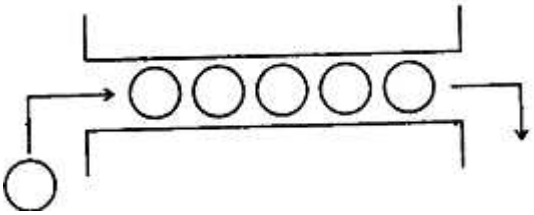


*Рис. 1.34.* Иллюстрация «очереди»

В языках программирования существуют и такие разновидности файлов, которые не подчиняются условию последовательности доступа к его компонентам (так называемые, файлы прямого доступа). Они уже не являются очередями.

**Суперпозиция структур данных**

Из рассмотренных структур данных можно создавать различные суперпозиции (вопрос о допустимости той или иной суперпозиции в конкретном языке программирования следует искать в его описании).

Рассмотрим в качестве примера такую часто используемую суперпозицию как **файл записей** - обычную, например, при создании баз данных. Итак, имеется файл по имени F, содержащий некоторое количество таких записей, как на рис. 1.30. Составим алгоритм подсчета количества болтов, у которых длина (length) заключена в пределах от 3 до 40:

1) положить $k = 0$ (в конце работы $k$ - число искомых болтов);
2) прочесть первую запись из файла;
3) если B.name = 'болт' и 30 < B.lenght < 40, то увеличить $k$ на 1;
4) если файл уже опустел, то идти к п. 7, иначе - к п. 5;
5) прочесть следующую запись из файла;
6) идти к п.3;
7) конец работы; $k$ - число искомых болтов.

**Стек**

Существует (и часто используется) и другая структура данных, в которой тот элемент, который первый в нее помещался, выходит последним и, наоборот, тот, который последним входит, выходит первым (английская аббревиатура «LIFO»). Такая структура получила название **стек** (или магазин - по сходству с магазином стрелкового оружия), рис. 1.35.

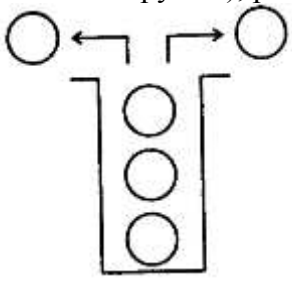


*Рис. 1 35.* Иллюстрация «стека»

Стеки и принцип LIFO находят очень широкое применение в информатике. Рассмотрим в качестве примера использование стека при вычислении значения арифметического выражения.

Вычисление значения выражения требует соблюдения старшинства операций. Операции по старшинству при вычислении значений математических выражений располагаются в следующем порядке: вычисление значений функций (включая возведения в степень), умножения и деления, сложения и вычитания. Изменить такой «естественный» порядок операций можно с помощью скобок.

Например, вычисление известного из школьного курса математики выражения $b^2$- 4*a*c включает предварительное установление порядка выполнения операций:

14 2 3
$b^2 - 4*a*c$

Для этого выражение просматривают несколько раз. Выполнение каждой операции дает некоторое число, которое приходится записывать отдельно от выражения, запоминая тот фрагмент выражения, для которого число является значением.

Сейчас рассмотрим экономный алгоритм вычисления значения выражения, использующий два магазина для перестановки элементов выражения (с учетом старшинства операций) и для хранения промежуточных результатов. Магазины обозначим $M_1$ и $M_2$, в $M_1$ будут попадать знаки операций, в $M_2$ - числа, участвующие в записи выражения, значения переменных и все промежуточные числовые значения.

Ограничимся выражениями, состоящими только из чисел и переменных без индекса, связанных знаками операций, *, /, +, -. Знак «минус» будет знаком лишь двухместной операции вычитания, выражения типа « - $a$ + I» исключаются из рассмотрения. От этих ограничений можно было бы и отказаться, но это удлинило бы изложение. Пока предположим также, что в выражении нет скобок.

Опишем алгоритм вычисления. Исходное выражение читается слева направо; если прочитано число, то оно заносится в $M_2$, если переменная - в $M_2$ заносится ее значение; если же прочитан знак операции, то необходимо различать несколько случаев.

1) $M_1$ пуст; прочитанный знак помещается на вершину $M_1$.

2) прочитанный знак помещается на вершину $M_1$, если он обозначает операцию, которая старше и поэтому должна выполняться до операции, знак которой был расположен на вершине $M_1$.

3) если операции равноправны или если та, знак которой только что прочитан в выражении, должна выполняться позднее, необходимо применить операцию, знак которой расположен на вершине $M_1$, к двум верхним числам из $M_2$ (число на вершине - второй операнд, число под ним - первый); знак операции на вершине $M_1$ удаляется из $M_1$, вместо двух верхних чисел в $M_2$ помещается результат выполнения над ними операции.

В некоторый момент в исходном выражении не остается символов. Если пуст и $M_1$, то вычисление окончено, результат находится в $M_2$; в противном случае знаки операции извлекаются по очереди из $M_1$ и соответствующие операции применяются к числам из $M_2$.

Рассмотрим вычисление выражения $b^2$ - 4*$a$*$c$; значения переменных $a$, $b$, $c$ обозначим A, $B$, $C$. Знак возведения в степень обозначим, как часто делается, стрелкой вверх.

| | | | |
|---|---|---|---|
| $M_2$: | $B$ | $M_1$: | ↑ |
| $M_2$: | 2<br>$B$ | $M_1$: | ↑ |
| $M_2$: | $B$↑2 | $M_1$: | |
| $M_2$: | 4<br>$B$↑2 | $M_1$: | – |
| $M_2$: | $A$<br>4<br>$B$↑2 | $M_1$: | *<br>– |
| $M_2$: | 4*$A$<br>$B$↑2 | $M_1$: | – |
| $M_2$: | $C$<br>4*$A$<br>$B$↑2 | $M_1$: | *<br>– |
| $M_2$: | 4*$A$*$C$<br>$B$↑2 | $M_1$: | – |
| $M_2$: | $B$↑2 – 4*$A$*$C$ | $M_1$: | |

Про знак операции говорят, что он имеет более высокий приоритет в сравнении с другим знаком, если обозначаемая им операция старше. В других случаях говорят о равных приоритетах или более низком приоритете. Рассмотренные знаки операций распадаются на группы равных по

приоритету:

↑

*, /

+, –

Группы упорядочены по убыванию приоритета.

Теперь дадим правило работы со скобками. Левая скобка заносится в $M_1$ сразу после прочтения. Прочтение правой скобки влечет выполнение всех операций, знаки которых находятся в *Mi* выше левой скобки; после выполнения этих операций обе скобки уничтожаются. Вот что будет происходить при выполнении *(a + b) * c*:

| | | | |
|---|---|---|---|
| $M_2$: | A | $M_1$: | ( |
| $M_2$: | B | $M_1$: | + |
| | A | | ( |
| $M_2$: | A + B | $M_1$: | ( |
| $M_2$: | A + B | $M_1$: | |
| $M_2$: | C | $M_1$: | * |
| | A+B | | |
| $M_2$: | (A+B)* C | $M_1$: | |

**Иерархическая организация данных**

Во всех рассмотренных выше структурах отдельные элементы (компоненты, поля, составляющие) структуры были формально равноправны. Существует, однако, широкий круг задач, в которых одни данные естественным образом «подвязаны» к другим. В этом случае возникает соподчиненная (иерархическая) структура данных. Ограничимся конкретным примером. Представим себе генеалогическое дерево, корень которого - имя человека, на следующем уровне - имена его родителей, еще на следующем - имена родителей родителей и т.д. Такая структура называется **двоичным деревом,** рис. 1.36.

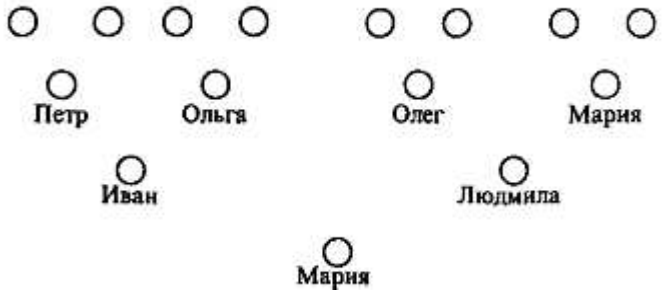



Рис. 1.36. Структура типа «двоичное дерево»;
пара ближайших по горизонтали кружков -мужское и женское имя

Как структурировать эти данные (имена)? Для помещения их в текстовый массив и запись трудно придумать логически оправданный порядок следования. Самое разумное - создать динамическую структуру типа той, что изображена на рис. 1.36. современные языки программирования позволяют это делать и обрабатывать такие структуры с высокой эффективностью.

***Контрольные вопросы и задания***

1. Какое значение имеет выбор представления и организации данных при разработке программы?
2. Какие данные можно отнести к простейшим неструктурированным?
3. Какие данные называют структурированными?
4. Охарактеризуйте свойства данных целого, действительного типа.

5. Какими свойствами обладают литерные и строковые величины?

6. Что называют логическими данными?

7. Что такое массив?

8. Решение каких задач требует использования массивов?

9. Разберите алгоритм сортировки массива, приведенный в тексте. Предложите свой алгоритм, отличный от данного.

10. Как определяется тип «множество»?

11. Чем отличается комбинированный тип данных (запись) от массива?

12. Что такое очередь (файл)? Какое применение имеют файлы?

13. Что такое стек?

14. Разберите по аналогии с примером, имеющимся в тексте о стеке, вычисление других арифметических выражений с использованием стеков.

15. Придумайте примеры естественной иерархической организации данных.

# § 10. ПОНЯТИЕ ОБ ИНФОРМАЦИОННОМ МОДЕЛИРОВАНИИ

## 10.1. МОДЕЛИРОВАНИЕ КАК МЕТОД РЕШЕНИЯ ПРИКЛАДНЫХ ЗАДАЧ

С точки зрения информатики, решение любой производственной или научной задачи описывается следующей технологической цепочкой: «реальный объект -модель - алгоритм - программа - результаты - реальный объект». В этой цепочке очень важную роль играет звено «модель», как необходимый, обязательный этап решения этой задачи. Под моделью при этом понимается некоторый мысленный образ реального объекта (системы), отражающий существенные свойства объекта и заменяющий его в процессе решения задачи.

**Модель** - очень широкое понятие, включающее в себя множество способов представления изучаемой реальности. Различают модели материальные (натурные) и идеальные (абстрактные). Материальные модели основываются на чем-то объективном, существующем независимо от человеческого сознания (каких-либо телах или процессах). Материальные модели делят на физические (например авто- и авиамодели) и аналоговые, основанные на процессах, аналогичных в каком-то отношении изучаемому (например, процессы в электрических цепях оказываются аналогичными многим механическим, химическим, биологическим и даже социальным процессам и могут быть использованы для их моделирования). Границу между физическими и аналоговыми моделями провести можно весьма приблизительно и такая классификация моделей носит условный характер.

Еще более сложную картину представляют идеальные модели, неразрывным образом связанные с человеческим мышлением, воображением, восприятием. Среди идеальных моделей можно выделить интуитивные модели, к которым относятся, например, произведения искусства - живопись, скульптура, литература, театр и т.д., но единого подхода к классификации остальных видов идеальных моделей нет. Иногда эти модели все разом относят к информационным. В основе такого подхода лежит расширительное толкование понятия «информация»: «информацией является почти все на свете, а может быть, даже вообще все». Такой подход является не вполне оправданным, так как он переносит информационную природу познания на суть используемых в процессе моделей - при этом любая модель является информационной. Более продуктивным представляется такой подход к классификации идеальных моделей, при котором различают следующие.

**1. Вербальные** (текстовые) модели. Эти модели используют последовательности предложений на формализованных диалектах естественного языка для описания той или иной области действительности (примерами такого рода моделей являются милицейский протокол, правила дорожного движения, настоящий учебник).

**2. Математические** модели - очень широкий класс знаковых моделей (основанных на формальных языках над конечными алфавитами), широко использующих те или иные математические методы. Например, можно рассмотреть математическую модель звезды. Эта модель будет представлять собой сложную систему уравнений, описывающих физические процессы, происходящие в недрах звезды. Математической моделью другого рода являются, например, математические соотношения, позволяющие рассчитать оптимальный (наилучший с экономической точки зрения) план работы какого-либо предприятия.

**3. Информационные** модели - класс знаковых моделей, описывающих информационные

процессы (возникновение, передачу, преобразование и использование информации) в системах самой разнообразной природы.

Граница между вербальными, математическими и информационными моделями может быть проведена весьма условно; возможно, информационные модели следовало бы считать подклассом математических моделей. Однако, в рамках информатики как самостоятельной науки, отдельной от математики, физики, лингвистики и других наук, выделение класса информационных моделей является целесообразным. Информатика имеет самое непосредственное отношение и к математическим моделям, поскольку они являются основой применения компьютера при решении задач различной природы: математическая модель исследуемого процесса или явления на определенной стадии исследования преобразуется в компьютерную (вычислительную) модель, которая затем превращается в алгоритм и компьютерную программу, рис. 1.37.


*Рис. 1.37.* Обобщенная схема компьютерного математического моделирования

## 10.2. ОСНОВНЫЕ ПОНЯТИЯ ИНФОРМАЦИОННОГО МОДЕЛИРОВАНИЯ

Остановимся на информационных моделях, отражающих процессы возникновения, передачи, преобразования и использования информации в системах различной природы. Начнем с определения простейших понятий информационного моделирования.

**Экземпляром** будем называть представление предмета реального мира с помощью некоторого набора его характеристик, существенных для решения данной информационной задачи (служащей контекстом построения информационной модели). Множество экземпляров, имеющих одни и те же характеристики и подчиняющиеся одним и тем же правилам, называется объектом.

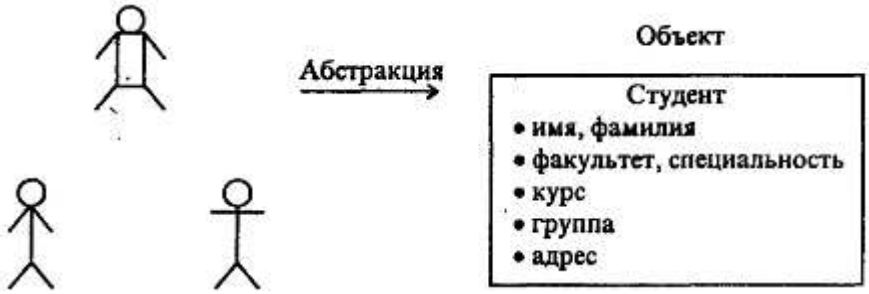



*Рис. 1.38.* Пример абстрагирования при построении информационной модели

Таким образом, объект есть абстракция предметов реального мира, объединяемых общими характеристиками и поведением, рис. 1.38.

Информационная модель какой-либо реальной системы состоит из объектов. Каждый объект в модели должен быть обеспечен уникальным и значимым **именем** (а также идентификатором, служащим ключом для указания этого объекта, связи его с другими объектами модели). Таким образом обозначение, наименование объекта -это элементарная процедура, лежащая в основе информационного моделирования.

Объект представляет собой один типичный (но неопределенный) экземпляр чего-то в реальном мире и является простейшей информационной моделью. Объекты представляют некие «сущности» предметов реального мира, связанные с решаемой задачей.

Большинство объектов, с которыми приходится встречаться, относятся к одной из следующих категорий:

- реальные объекты;
- роли;
- события;
- взаимодействия;
- спецификации.

**Реальный объект -** это абстракция физически существующих предметов. Например, на автомобильном заводе это кузов автомобиля, двигатель, коробка передач; при перевозке грузов это контейнер, средство перевозки.

**Роль** - абстракция цели или назначения человека, части оборудования или учреждения (ор-

ганизации). Например, в университете как в учебном заведении это студент, преподаватель, декан; в университете как в учреждении это приемная комиссия, отдел кадров, бухгалтерия, деканат.

**Событие** - абстракция чего-то случившегося. Например, поступление заявления от абитуриента в приемную комиссию Университета, сдача (или несдача) экзамена.

**Взаимодействия** - объекты, получаемые из отношений между другими объектами. Например, сделка, контракт (договор) между двумя сторонами, свидетельство об образовании, выдаваемое учебным заведением его выпускнику.

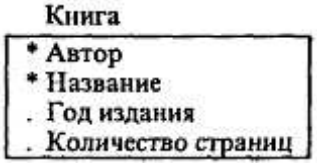
**Объекты-спецификации** используются для представления правил, стандартов или критериев качества. Например, перечень знаний, умений и навыков выпускника математического факультета, рецепт проявления фотопленки.

Для каждого объекта должно существовать его **описание** - короткое информационное утверждение, позволяющее установить, является некоторый предмет экземпляром объекта или нет. Например, описание объекта «Абитуриент университета» может быть следующим: человек в возрасте до 35 лет, имеющий среднее образование, подавший в приемную комиссию документы и заявление о приеме.

Предметы реального мира имеют **характеристики** (такие, например, как имя, название, регистрационный номер, дата изготовления, вес и т.д.). Каждая отдельная характеристика, общая для всех возможных экземпляров объекта, называется атрибутом. Для каждого экземпляра атрибут принимает определенное значение. Так, объект **Книга** имеет атрибуты **Автор**, **Название, Год издания. Число страниц.**

У каждого объекта должен быть **идентификатор** - множество из одного или более атрибутов, значения которых определяют каждый экземпляр объекта. Для книги атрибуты Автор и Название совместно образуют идентификатор. В тоже время Год издания и Число страниц идентификаторами быть не могут - ни врозь, ни совместно, так как не определяют объект. Объект может иметь и несколько идентификаторов, каждый из которых составлен из одного или нескольких атрибутов. Один из них может быть выбран как привилегированный для соответствующей ситуации.

Объект может быть представлен вместе со своими атрибутами несколькими различными способами. Графически объект может быть изображен в виде рамки, содержащей имя объекта и имена атрибутов. Атрибуты, которые составляют привилегированный идентификатор объекта, могут быть выделены (например, символом * слева от имени атрибута):




В эквивалентном текстовом представлении это может иметь следующий вид:

Книга (Автор. Название. Год издания. Число страниц).

Привилегированный идентификатор подчеркивается.

Еще одним способом представления объекта информационной модели является таблица. В этой интерпретации каждый экземпляр объекта является строкой в таблице, а значения атрибутов, соответствующих каждому экземпляру, - клетками строки, табл. 1.11.

**Таблица 1.11 Таблица как представление информационной модели**

| Автор | Книга | | |
|---|---|---|---|
| | Название | Год издания | Число страниц |
| Грин А. | Бегущая по волнам | 1988 | 279 |
| Стивенсон Р. П. | Остров сокровищ | 1992 | 269 |
| Скотт В. | Ричард Львиное Сердце | 1993 | 349 |
| Гончаров И. А. | Обрыв | 1986 | 598 |

Можно классифицировать атрибуты по принадлежности к одному из трех различных типов:

- описательные;
- указывающие;

• вспомогательные.

**Описательные атрибуты** представляют факты, внутренне присущие каждому экземпляру объекта. Если значение описательного атрибута изменится, то это говорит о том, что некоторая характеристика экземпляра изменилась, но сам экземпляр остался прежним.

**Указательные атрибуты** могут использоваться как идентификаторы (или часть идентификаторов) экземпляра. Если значение указывающих атрибутов изменяется, то это говорит лишь о том, что новое имя дается тому же самому экземпляру.

**Вспомогательные атрибуты** используются для связи экземпляра одного объекта с экземпляром другого объекта.

Рассмотрим пример:

Автомобиль
* гос. номер
. марка
. цвет
. владелец

Атрибут «цвет» является описательным, атрибуты «гос. номер» и «марка» - указательными, атрибут «владелец» - вспомогательным, служащим для связи экземпляра объекта Автомобиль с экземпляром объекта Автолюбитель. Если значение вспомогательного атрибута изменится, это говорит о том, что теперь другие экземпляры объектов связаны между собой.

## 10.3. СВЯЗИ МЕЖДУ ОБЪЕКТАМИ

В реальном мире между предметами существуют различные отношения. Если предметы моделируются как объекты, то отношения, которые систематически возникают между различными видами объектов, отражаются в информационных моделях как связи. Каждая **связь** задается в модели определенным именем. Связь в графической форме представляется как линия между связанными объектами и обозначается идентификатором связи.

Существует три вида связи: один-к-одному (рис. 1.39), один-ко-многнм (рис. 1.40) и многие-ко-многим (рис. 1.41).

Связь один-к-одному существует, когда один экземпляр одного объекта связан с единственным экземпляром другого. Связь один-к-одному обозначается стрелками ←и→.


*Рис. 1.39.* Пример связи «одии-к-одному»

Связь один-ко-многим существует, когда один экземпляр первого объекта связан с одним (или более) экземпляром второго объекта, но каждый экземпляр второго объекта связан только с одним экземпляром первого. Множественность связи изображается двойной стрелкой →→.


*Рис. 1.40.* Пример связи «один-ко-многим»

Связь многие-ко-многим существует, когда один экземпляр первого объекта связан с одним или большим количеством экземпляров второго и каждый экземпляр второго связан с одним или многими экземплярами первого. Этот тип связи изображается двусторонней стрелкой ↔

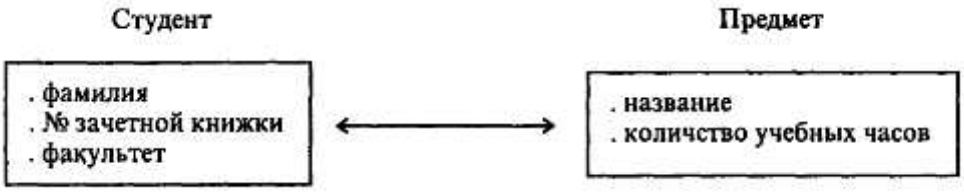



*Рис. 1.41.* Пример связи «многие-ко-многим»

Помимо множественности, связи могут подразделяться на безусловные и условные. В безусловной связи для участия в ней требуется каждый экземпляр объекта. В условной связи принимают участие не все экземпляры объекта. Связь может быть условной как с одной, так и с обеих сторон.

Все связи в информационной модели требуют описания, которое, как минимум, включает:

- идентификатор связи;
- формулировку сущности связи;
- вид связи (ее множественность и условность);
- способ описания связи с помощью вспомогательных атрибутов объектов.

Дальнейшее развитие представлений информационного моделирования связано с развитием понятия связи, структур, ими образуемых, и задач, которые могут быть решены на этих структурах. Нам уже известна простая последовательная структура экземпляров - очередь, см. рис. 1.34. Возможными обобщениями информационных моделей являются циклическая структура, таблица (см. табл. 1.10), стек (см. рис. 1.35).

Очень важную роль играет древовидная информационная модель, являющаяся одной из самых распространенных типов классификационных структур. Эта модель строится на основе связи, отражающей отношение части к целому: *«А* есть часть М» или «М управляет *А»*. Очевидно, древовидная связь является безусловной связью типа один-ко-многим и графически изображена на рис. 1.42, *в.* На этом же рисунке для сравнения приведены схемы информационных моделей типа «очередь» (а) и «цикл» *(б).*

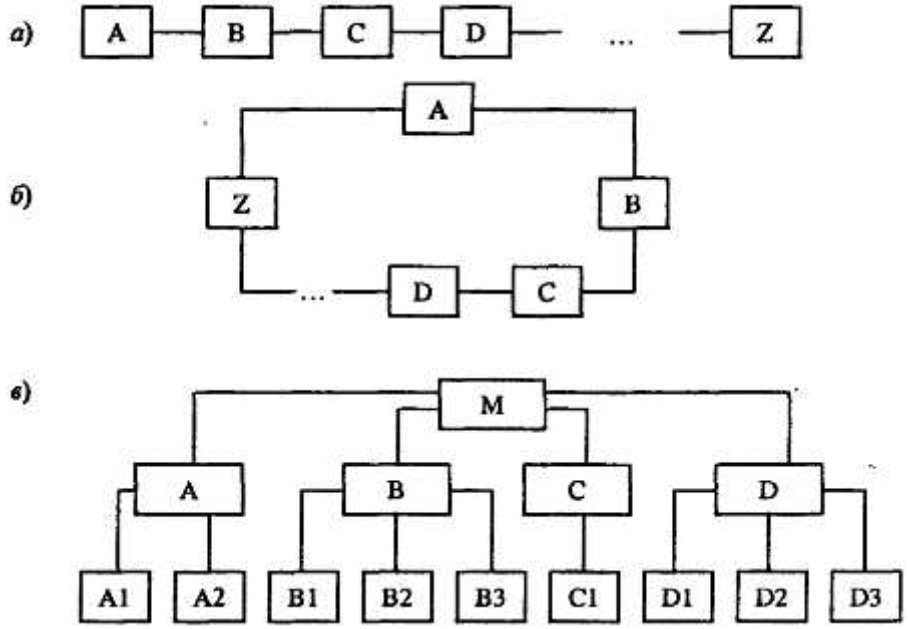



*Рис. 1.42.* Информационные модели типа «очередь» (а), «цикл» (б), «дерево» *(в)*

Таким образом, типы данных в программировании, обсуждавшиеся в предыдущем параграфе, тесно связаны с определенными информационными моделями данных.

Еще более общей информационной моделью является, так называемая, графовая структура, рис. 1.43. Графовые структуры являются основой решения огромного количества задач информационного моделирования.

Многие прикладные задачи информационного моделирования были поставлены и изучены достаточно давно, в 50-60-х годах, в связи с активно развивавшимися тогда исследованиями и разработками по научным основам управления в системах различной природы и в связи с попытками смоделировать с помощью компьютеров психическую деятельность человека при решении твор-

ческих интеллектуальных задач. Научное знание и модели, которые были получены в ходе решения этих задач, объединены в науке под названием «Кибернетика», в рамках которой существует раздел «Исследования по искусственному интеллекту».

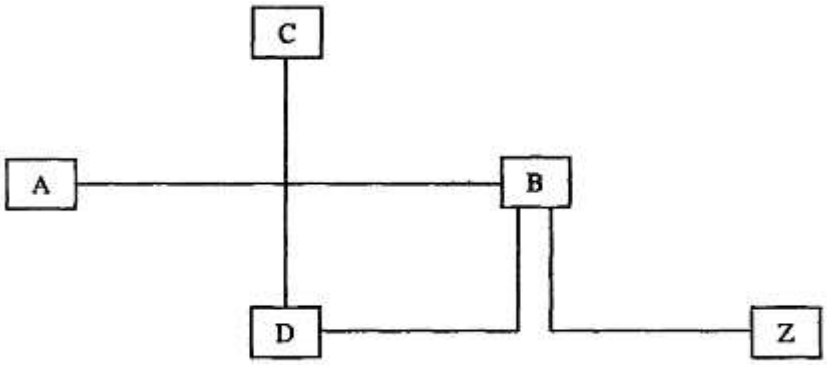



*Рис. 1.43.* Информационная модель типа «граф»

***Контрольные вопросы и задания***

1. Что означает понятие «модель» в научном познании?
2. Какие типы моделей существуют?
3. Что такое «информационная модель»?
4. Что такое «объект» с точки зрения информационного моделирования? Какие типы объектов можно различать?
5. Что такое «атрибуты»? Какими они бывают?
6. Что такое «связь»? Какие типы связи различают?
7. Разработайте примеры древовидных структур данных из окружающей реальности.

## § 11. НЕКОТОРЫЕ КИБЕРНЕТИЧЕСКИЕ АСПЕКТЫ ИНФОРМАТИКИ

### 11.1. ПРЕДМЕТ КИБЕРНЕТИКИ

Слово «кибернетика» происходит от греческого слова, означающего в переводе «кормчий». Его современное значение связано с научной областью, начало которой положила книга американского ученого Норберта Винера «Кибернетика или управление и связь в животном и машине», вышедшая в 1948 г. Вскоре предметом новой науки стали не только биологические и технические системы, но и системы любой природы, способные воспринимать, хранить и перерабатывать информацию и использовать ее для управления и регулирования. В изданной в 1947 г. «Энциклопедии кибернетики» говорится, что это «... наука об общих законах получения, хранения, передачи и преобразования информации в сложных управляющих системах. При этом под управляющими системами здесь понимаются не только технические, а и любые биологические, административные и социальные системы». Таким образом, кибернетика и информатика являются, скорее всего, единой наукой. Сегодня кибернетику все чаще считают частью информатики, ее «высшим» разделом, в какой-то степени аналогичным по положению «высшей математике» по отношению ко всей математике вообще (примерно в таком же положении по отношению к информатике находится и наука «Искусственный интеллект»). Информатика в целом шире кибернетики, так как в информатике имеются аспекты, связанные с архитектурой и программированием ЭВМ, которые непосредственно к кибернетике отнести нельзя.

Кибернетические разделы информатики богаты подходами и моделями в исследовании разнообразных систем и используют в качестве аппарата многие разделы фундаментальной и прикладной математики.

Классическим и до известной степени самостоятельным разделом кибернетики считают **исследование операций.** Под этим термином понимают применение математических методов для обоснования решений в различных областях целенаправленной человеческой деятельности.

Поясним, что понимается под «решением». Пусть предпринимается некоторое мероприятие (в производственной, экономической или социальной сфере), направленное на достижение определенной цели - такое мероприятие называется «операцией». У лица (или группы лиц), ответственного за проведение этого мероприятия, имеется возможность выбора, как его организовать.

Например, можно выбрать виды продукции, которые будут выпускаться, оборудование, которое при этом будет применяться, так или иначе распределить имеющиеся средства и т.д. «Операция» есть управляемое мероприятие.

**Решение** есть выбор из ряда возможностей, имеющихся у ответственного лица. Решения могут быть удачными и неудачными, разумными и неразумными. Оптимальными называют решения, по тем или другим принципам более предпочтитель-ные, чем другие. Цель исследования операций - математическое (количественное) обоснование оптимальных решений.

Исследование операций включает в себя следующие разделы:

1) **математическое программирование** (обоснование планов, программ хозяйственной деятельности); оно включает в себя относительно самостоятельные разделы: **линейное программирование, нелинейное программирование, динамическое программирование** (во всех этих названиях термин «программирование» возник исторически и не имеет отношения к программированию ЭВМ);

**2) теорию массового обслуживания,** опирающуюся на теорию случайных процессов;

3) **теорию игр,** позволяющую обосновывать решения, принимаемые в условиях неполноты информации.

Отметим, что эти разделы не связаны непосредственно с ЭВМ и техническими системами. Иным, быстро развивавшимся в 70-х-80-х годах, разделом кибернетики были системы автоматического (автоматизированного) регулирования. Этот раздел имест замкнутый, автономный характер, исторически сложившийся самостоятельно. Он тесно связан с разработкой технических систем автоматизированного регулирования и управления технологическими и производственными процессами.

Еще одним классическим разделом кибернетики является распознавание образов, возникшее из задачи моделирования в технических системах восприятия человеком знаков, предметов и речи, а также формирования у человека понятий (обучение в простейшем, техническом смысле). Этот раздел в значительной мере возник из технических потребностей **робототехники.** Например, требуется, чтобы робот-сборщик распознавал нужные детали. При автоматической сортировке (или отбраковке) деталей необходима способность распознавания.

Вершиной кибернетики (и всей информатики в целом) является раздел, посвященный проблемам **искусственного интеллекта.** Большинство современных систем управления обладают свойством принятия решений - свойством интеллектуальности, т.е. в них смоделирована интеллектуальная деятельность человека при принятии решений.

## 11.2. УПРАВЛЯЕМЫЕ СИСТЕМЫ

Несмотря на такое многообразие задач, решаемых в разных разделах кибернетики, разнообразие моделей, подходов и методов, кибернетика остается единой наукой благодаря использованию общей методологии, основанной на **теории систем и системном анализе.**

**Система** - это предельно широкое, начальное, не определяемое строго понятие. Предполагается, что система обладает структурой, т.е. состоит из относительно обособленных частей (элементов), находящихся, тем не менее, в существенной взаимосвязи и взаимодействии. Существенность взаимодействия состоит в том, что благодаря ему элементы системы приобретают все вместе некую новую функцию, новое свойство, которыми не обладает ни один из элементов в отдельности. В этом состоит отличие системы от сети, также состоящей из отдельных элементов, но не связанных между собой существенными отношениями. Сравните, например, предприятие, цеха которого образуют систему, поскольку лишь все вместе приобретают свойство выпускать конечную продукцию (и ни один из них в отдельности с этой задачей не справится), и сеть магазинов, которые могут работать независимо друг от друга.

Кибернетика как наука об управлении изучает не все системы вообще, а только управляемые системы. Зато область интересов и приложений кибернетики распространяется на самые разнообразные биологические, экономические, социальные системы.

Одной из характерных особенностей управляемой системы является возможность переходить в различные состояния под влиянием различных управляющих воздействий. Всегда существует некое множество состояний системы, из которых производится выбор предпочтительного состояния.

Отвлекаясь от конкретных особенностей отдельных кибернетических систем и выделяя

общие для некоторого множества систем закономерности, описывающие изменение их состояния при различных управляющих воздействиях, мы приходим к понятию абстрактной **кибернетической системы.** Ее составляющими являются не конкретные предметы, а абстрактные элементы, характеризующиеся определенными свойствами, общими для широкого класса объектов.

Поскольку под кибернетическими системами понимаются управляемые системы, в них должен присутствовать механизм, осуществляющий функции управления. Чаще всего этот механизм реализуется в виде органов, специально предназначенных для управления, рис. 1.44.

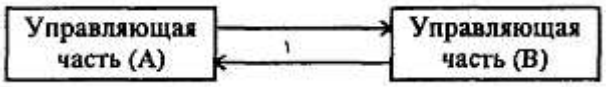



*Рис. 1.44.* Схематическое изображение кибернетической системы в виде совокупности управляющей (А) и управляемой (В) частей

Стрелками на рисунке обозначены воздействия, которыми обмениваются части системы. Стрелка, идущая от управляющей части системы к управляемой, обозначает сигналы управления. Управляющая часть системы, вырабатывающая сигналы управления, называется управляющим устройством. Управляющее устройство может вырабатывать сигналы управления, обычно на основе информации о состоянии управляемой системы (изображены на рисунке стрелкой от управляемой части системы к управляющей ее части), о требуемом ее состоянии, о возмущающих воздействиях. Совокупность правил, по которым информация, поступающая в управляющее устройство, перерабатывается в сигналы управления, называется алгоритмом управления.

На основе введенных понятий можно определить понятие «управление». Управление - это воздействие на объект, выбранное из множества возможных воздействий на основе имеющейся для этого информации, улучшающее функционирование или развитие данного объекта.

В системах управления решаются четыре основных типа задач управления: 1) регулирование (стабилизация), 2) выполнение программы, 3) слежение и 4) оптимизация.

Задачами регулирования являются задачи поддержания параметров системы -управляемых величин - вблизи некоторых неизменных заданных значений *{x}* несмотря на действие возмущений *M,* влияющих на значения *{x}*. Здесь имеется в виду активная защита от возмущений, принципиально отличающаяся от пассивного способа защиты. **Пассивная защита** заключается в придании объекту таких свойств, чтобы зависимость интересующих нас параметров от внешних возмущений была мала. Примером пассивной защиты является теплоизоляция для поддержания заданной температуры системы, антикоррозионные покрытия деталей машин. **Активная защита** предполагает выработку в управляющих системах управляющих воздействий, противодействующих возмущениям. Так, задача поддержания необходимой температуры системы может быть решена с помощью управляемого подогрева или охлаждения.

Задача выполнения программы возникает в случаях, когда заданные значения управляемых величин *{x}* изменяются во времени известным образом, например, в производстве при выполнении работ согласно заранее намеченному графику. В биологических системах примерами выполнения программы являются развитие организмов из яйцеклеток, сезонные перелеты птиц, метаморфозы насекомых.

Задача слежения - поддержание как можно более точного соответствия некоторого управляемого параметра $X_0(t)$ текущему состоянию системы, меняющемуся непредвидимым образом. Необходимость в слежении возникает, например, при управлении производством товаров в условиях изменения спроса.

Задачи оптимизации - установления наилучшего в определенном смысле режима работы или состояния управляемого объекта - встречаются весьма часто. Примерами являются: управление технологическими процессами с целью минимизации потерь сырья и т.д.

Системы, в которых для формирования управляющих воздействий не используется информация о значениях, которые управляемые величины принимают в процессе управления, называются **разомкнутыми системами управления.** Структура такой системы показана на рис. 1.45.

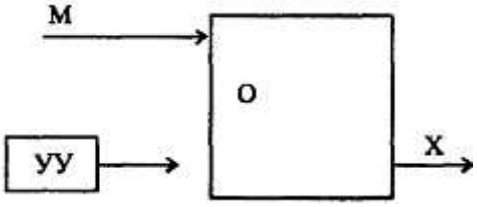


*Рис. 1.45.* Алгоритм управления, реализуемый управляющим устройством УУ, которое обеспечивает слежение за возмущением М и компенсацию этого возмущения, без использования управляемой величины Х

Напротив, в замкнутых системах управления для формирования управляющих воздействий используется информация о значении управляемых величин. Структура такой системы показана на рис. 1.46.

**Обратная связь** является одним из важнейших понятий кибернетики, помогающим понять многие явления, которые происходят в управляемых системах различной природы. Обратную связь можно обнаружить при изучении процессов, протекающих в живых организмах, экономических структурах, системах автоматического регулирования. Обратная связь, увеличивающая влияние входного воздействия на управляемые параметры системы, называется положительной, уменьшающая влияние входного воздействия - отрицательной.

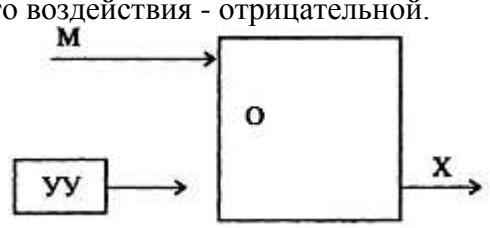



*Рис. 1.46.* Связь между выходными параметрами А" и входными У одного и того же элемента управляемой системы называется обратной связью

**Положительная обратная связь** используется во многих технических устройствах для усиления, увеличения значений входных воздействий. Отрицательная обратная связь используется для восстановления равновесия, нарушенного внешним воздействием на систему.

## 11.3. ФУНКЦИИ ЧЕЛОВЕКА И МАШИНЫ В СИСТЕМАХ УПРАВЛЕНИЯ

Хорошо изученной сферой применения кибернетических методов является технологическая и производственная сфера, управление промышленным предприятием. Задачи, возникающие в управлении предприятием среднего и большого масштаба, уже весьма сложны, но допускают решение с использованием электронно-вычислительных машин. Системы управления хозяйством предприятий или территорий (регионов, городов), использующие ЭВМ для переработки и хранения информации, получили название автоматизированных систем управления (АСУ). По своему характеру такие системы являются человеко-машинными, т.е. наряду с использованием мощных компьютеров предполагающими наличие в них человека с его естественным интеллектом. В человеко-машинных системах предполагается следующее разделение функций человека и машины: **машина** хранит и перерабатывает большие массивы информации, осуществляет информационное обеспечение принятия решений человеком; **человек** принимает управленческие решения.

Чаще в человеко-машинных системах компьютеры выполняют рутинную, нетворческую, трудоемкую переработку информации, освобождая человеку время для творческой деятельности. Однако целью развития компьютерной (информационной) технологии управления является полная автоматизация деятельности, включающая частичное или полное освобождение человека от необходимости принятия решений. Это связано не только со стремлением разгрузить человека, но и с тем, что развитие техники и технологий привело к ситуациям, когда человек в силу присущих ему физиологических и психических ограничений просто не успевает принимать решения в реальном масштабе времени протекания процесса, что грозит катастрофическими последствиями. Примеры - необходимость включения аварийной защиты ядерного ректора, реакция на события, проистекающие при запусках космических аппаратов и т.д.

Система, заменяющая человека, должна будет обладать интеллектом, в какой-то мере подобным человеческому - **искусственным интеллектом.** Исследовательское направление в области систем искусственного интеллекта также относится к кибернетике, однако вследствие его важности для перспектив всей информатики в целом мы рассмотрим его в отдельном параграфе.

***Контрольные вопросы и задания***

1. Каков предмет науки «Кибернетика»?
2. Охарактеризуйте задачи, решаемые в научном разделе «исследование операций».

3. Какое место в кибернетике занимает теория автоматического управления и регулирования?

4. Что означает понятие «система»?

5. Что такое «система управления»?

6. Охарактеризуйте задачи, возникающие в системах управления.

7. Что такое «обратная связь»? Приведите примеры обратной связи в окружающих вас управляемых системах.

8. Что такое АСУ?

9. Каково место человека и ЭВМ в человеко-машинных системах управления?

# § 12. ПОНЯТИЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

## 12.1. НАПРАВЛЕНИЯ ИССЛЕДОВАНИЙ И РАЗРАБОТОК В ОБЛАСТИ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Научное направление, связанное с машинным моделированием человеческих интеллектуальных функций - искусственный интеллект - возникло в середине 60-х годов XX столетия. Его возникновение непосредственно связано с общим направлением научной и инженерной мысли, которое привело к созданию компьютера -направлением на автоматизацию человеческой интеллектуальной деятельности, на то, чтобы сложные интеллектуальные задачи, считавшиеся прерогативой человека, решались техническими средствами.

Говоря о сложных интеллектуальных задачах следует понимать, что всего 300 -400 лет назад перемножение больших чисел вполне относилось к таковым; однако, усвоив в детстве правило умножения столбиком, современный человек пользуется им не задумываясь, и вряд ли эта задача сегодня является «сложной интеллектуальной». По-видимому, в круг таковых следует включить те задачи, для решения которых нет «автоматических» правил, т.е. нет алгоритма (пусть даже и очень сложного), следование которому всегда приводит к успеху. Есди для решения задачи, которая нам сегодня представляется относящейся к указанному кругу, в будущем придумают четкий алгоритм, она перестанет быть «сложной интеллектуальной».

Несмотря на свою краткость, история исследований и разработок систем искусственного интеллекта может быть разделена на четыре периода:

• 60-е - начало 70-х годов XX века - исследования по «общему интеллекту», попытки смоделировать общие интеллектуальные процессы, свойственные человеку: свободный диалог, решение разнообразных задач, доказательство теорем, различные игры (типа шашек, шахмат и т.д.), сочинение стихов и музыки и т.д.;

• 70-е годы - исследования и разработка подходов к формальному представлению знаний и умозаключений, попытки свести интеллектуальную деятельность к формальным преобразованиям символов, строк и т.д.;

• с конца 70-х годов - разработка специализированных на определенных предметных областях интеллектуальных систем, имеющих прикладное практическое значение (экспертных систем);

• 90-е годы - фронтальные работы по созданию ЭВМ 5-го поколения, построенных на иных принципах, чем обычные универсальные ЭВМ, и программного обеспечения для них.

В настоящее время «искусственный интеллект» - мощная ветвь информатики, имеющая как фундаментальные, чисто научные основы, так и весьма развитые технические, прикладные аспекты, связанные с созданием и эксплуатацией работоспособных образцов интеллектуальных систем. Значение этих работ для развития информатики таково, что именно от их успеха зависит появление ЭВМ нового 5-го поколения. Именно этот качественный скачок возможностей компьютеров - обретение ими в полной мере интеллектуальных возможностей - положен целью развития вычислительной техники в ближайшей перспективе и является признаком компьютерной техники нового поколения.

Любая задача, для которой не известен алгоритм решения, может быть отнесена к сфере искусственного интеллекта. Примерами могут быть игра в шахматы, медицинская диагностика, составление резюме текста или перевода его на иностранный язык - для решения этих задач не существует четких алгоритмов. Еще две характерные особенности задач искусственного интеллекта: преобладающее использование информации в символьной (а не в числовой) форме и наличие выбора между многими вариантами в условиях неопределенности.

Перечислим отдельные направления, где применяются методы искусственного интеллекта.

1. Восприятие и распознавание образов (задача, упоминавшаяся ранее, как одно из направлений кибернетики). Теперь под этим понимаются не просто технические системы, воспринимающие визуальную и звуковую информацию, кодирующие и размещающие ее в памяти, проблемы понимания и логического рассуждения в процессе обработки визуальной и речевой информации.

2. Математика и автоматическое доказательство теорем.

3. Игры. Как и формальные системы в математике, игры, характеризующиеся конечным числом ситуаций и четко определенными правилами, с самого начала исследований по искусственному интеллекту привлекли к себе внимание как предпочтительные объекты исследования, полигон для применения новых методов. Интеллектуальными системами был быстро достигнут и превзойден уровень человека средних способностей, однако уровень лучших специалистов не достигнут до сих пор. Возникшие трудности оказались характерными и для многих других ситуаций, так как в своих «локальных» действиях человек использует весь объем знаний, который он накопил за всю свою жизнь.

4. Решение задач. В данном случае понятие «решение» используется в широком смысле, относится к постановке, анализу и представлению конкретных ситуаций, а рассматриваемые задачи - те, которые встречаются в повседневной жизни, для решения которых требуется изобретательность и способность к обобщению.

5. Понимание естественного языка. Здесь ставится задача анализа и генерации текстов, их внутреннего представления, выявление знаний, необходимых для понимания текстов. Трудности связаны, в частности, с тем, что значительная часть информации в обычном диалоге не выражается определенно и ясно. Предложениям естественного языка присуща:

- неполнота;
- неточность;
- нечеткость;
- грамматическая некорректность;
- избыточность;
- зависимость от контекста;
- неоднозначность.

Однако такие свойства языка, являющегося результатом многовекового исторического развития, служат условием функционирования языка как универсального средства общения. Вместе с тем, понимание предложений естественного языка техническими системами с трудом поддается моделированию из-за- этих особенностей языка (да и вопрос о том, что такое «понимание», нуждается в размышлениях). В технических системах должен использоваться формальный язык, смысл предложений которого однозначно определяется их формой. Перевод с естественного языка на формальный является нетривиальной задачей.

6. Выявление и представление знаний экспертов в экспертных системах. Экспертные системы - интеллектуальные системы, вобравшие в себя знания специалистов в конкретных видах деятельности - имеют большое практическое значение, с успехом применяются во многих областях, таких как автоматизированное проектирование, медицинская диагностика, химический анализ и синтез и т.д.

Во всех этих направлениях главные трудности связаны с тем, что недостаточно изучены и поняты принципы человеческой интеллектуальной деятельности, процесс принятия решений и решение задач. Если в 60-х годах широко обсуждался вопрос «может ли компьютер мыслить», то теперь вопрос ставится иначе: «достаточно ли хорошо человек понимает, как он мыслит, чтобы передать эту функцию компьютеру»? В силу этого, работы в области искусственного интеллекта тесно соприкасаются с исследованиями по соответствующим разделам психологии, физиологии, лингвистики.

## 12.2. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ В СИСТЕМАХ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Основной особенностью интеллектуальных систем является то, что они основаны на знаниях, а вернее, на некотором их представлении. **Знания** здесь понимаются как хранимая (с помощью ЭВМ) информация, формализованная в соответствии с некоторыми правилами, которую ЭВМ

может использовать при логическом выводе по определенным алгоритмам. Наиболее фундаментальной и важной проблемой является описание смыслового содержания проблем самого широкого диапазона, т.е. должна использоваться такая форма описания знаний, которая гарантировала бы правильную обработку их содержимого по некоторым формальным правилам. Эта проблема называется проблемой представления знаний.

В настоящее время наиболее известны три подхода **к представлению знаний** в обсуждаемых системах:

• продукционная и логическая модели;
• семантические сети;
• фреймы.

**Продукционные правила** - наиболее простой способ, представления знаний. Он основан на представлении знаний в форме правил, структурированных в соответствии с образцом «ЕСЛИ - ТО». Часть правила «ЕСЛИ» называется посылкой, а «ТО» - выводом или действием. Правило в общем виде записывается так:

ЕСЛИ $A_1$, $A_2$, ...,$A_n$, ТО $B$.

Такая запись означает, что «если все условия от $A_1$ до $A_n$ являются истинными, то $B$ также истинно» или «когда все условия от $A_1$ до $A_n$ выполняются, то следует выполнить действие $B$».

Рассмотрим правило

ЕСЛИ (1) *y* является отцом *x*
(2) z является братом *y*
ТО z является дядей *x*

В данном случае число условий $n = 2$.

В случае $n = 0$ продукция описывает знание, состоящее только из вывода, т.е. факт. Примером такого знания является факт «атомная масса железа 55,847 а.е.м».

Переменные *x, y* и г показывают, что правило содержит некое универсальное, общее знание, абстрагированное от конкретных значений переменных. Одна и та же переменная, использованная в выводе и различных посылках, может получать различные конкретные значения.

Знания, представленные в интеллектуальной системе, образуют **базу знаний.** В интеллектуальную систему входит также **механизм выводов,** который позволяет на основе знаний, имеющихся в базе знаний, получать новые знания.

Проиллюстрируем сказанное. Положим, что в базе знаний вместе с описанным выше правилом содержатся и такие знания:

ЕСЛИ (1) z является отцом *x*
(2) z является отцом *y*
(3) *x и y* не являются одним и тем же человеком
ТО *x* и *y* являются братьями
Иван является отцом Сергея
Иван является отцом Павла
Сергей является отцом Николая

Из представленных знаний можно формально вывести заключение, что Павел является дядей Николая. При этом считается, что одинаковые переменные, входящие в разные правила, независимы; объекты, имена которых эти переменные могут получать, никак не связаны между собой. Формализованная процедура, использующая сопоставление (при котором устанавливается, совпадают ли между собой две формы представления, включая подстановку возможных значений переменных), поиск в базе знаний, возврат к исходному состоянию при неудачной попытке решения, представляет собой механизм выводов.

Простота и наглядность представления знаний с помощью продукций обусловила его применение во многих системах, которые называются продукционными.

**Семантическая сеть** - иной подход к представлению знаний, который основан на изображении понятий (сущностей) с помощью точек (узлов) и отношений между ними с помощью дуг на плоскости. Семантические сети способны отображать структуру знаний во всей сложности их

взаимосвязей, увязать в единое целое объекты и их свойства. В качестве примера может быть приведена часть семантической сети, относящейся к понятию «фрукты» (рис. 1.47).

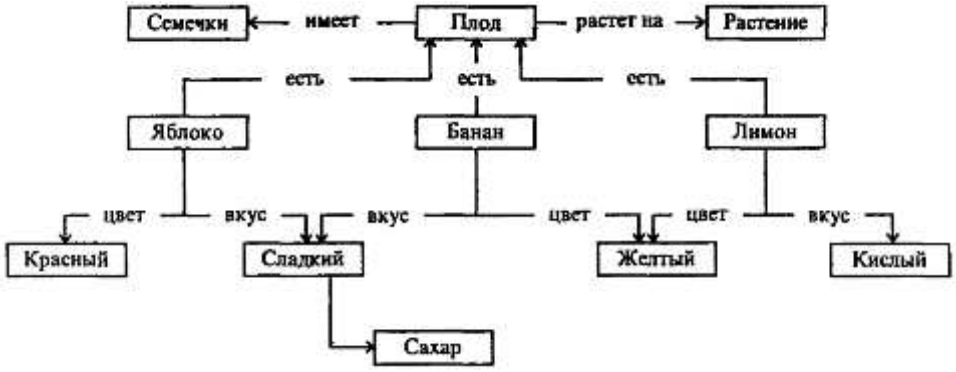



*Рис. 1.47.* Пример семантической сети

**Фреймовая система** имеет все свойства, присущие языку представления знаний, и одновременно являет собой новый способ обработки информации. Слово «фрейм» в переводе с английского языка означает «рамка». Фрейм является единицей представления знаний об объекте, которую можно описать некоторой совокупностью понятий и сущностей. Фрейм имеет определенную внутреннюю структуру, состоящую из множества элементов, называемых **слотами.** Каждый слот, в своюо-чередь, представляется определенной структурой данных, процедурой, или может быть связан с другим фреймом.

**Фрейм: человек**

| | | |
|---|---|---|
| Класс | : | Животное |
| Структурный элемент | : | Голова, шея, руки, ноги,... |
| Рост | : | 30-220 см |
| Масса | : | 1 - 200 кг |
| Хвост | : | Нет |
| Фрейм аналогии | : | Обезьяна |

Существуют и другие, менее распространенные подходы к представлению знаний в интеллектуальных системах, в том числе гибридные, на основе уже описанных подходов.

Перечислим главные особенности машинного представления данных.

1. Внутренняя интерпретируемость. Обеспечивается наличием у каждой информационной единицы своего уникального имени, по которому система находит ее для ответа на запросы, в которых это имя упомянуто.

2. Структурированность. Информационные единицы должны обладать гибкой структурой, для них должен выполняться «принцип матрешки», т.е. вложенности одних информационных единиц в другие, должна существовать возможность установления соотношений типа «часть - целое», «род - вид», «элемент - класс» между отдельными информационными единицами.

3. Связность. Должна быть предусмотрена возможность установления связей различного типа между информационными единицами, которые бы характеризовали отношения между информационными единицами. Эти отношения могут быть как декларативными (описательными), так и процедурными (функциональными).

4. Семантическая метрика. Позволяет устанавливать ситуационную близость информационных единиц, т.е. величину ассоциативной связи между ними. Такая близость позволяет выделять в знаниях некоторые типовые ситуации, строить аналогии.

5. Активность. Выполнение действий в интеллектуальной системе должно инициироваться не какими-либо внешними причинами, а текущим состоянием представленных в системе знаний. Появление новых фактов или описание событий, установление связей должны стать источником активности системы.

## 12.3. МОДЕЛИРОВАНИЕ РАССУЖДЕНИЙ

**Рассуждение** - один из важнейших видов мыслительной деятельности человека, в результате которого он формулирует на основе некоторых предложений, высказываний, суждений новые

предложения, высказывания, суждения. Действительный механизм рассуждений человека остается пока недостаточно исследованным. Человеческим рассуждениям присущи: неформальность, нечеткость, нелогичность, широкое использование образов, эмоций и чувств, что делает чрезвычайно трудными их исследование и моделирование. К настоящему времени лучше всего изучены логические рассуждения и разработано много механизмов дедуктивных выводов, реализованных в различных интеллектуальных системах, основанных на представлении знаний с помощью логики предикатов 1-го порядка.

Предикат - это конструкция вида $P(t_1, t_2, ..., t_n)$, выражающая какую-то связь между некоторыми объектами или свойствами объектов. Обозначение этой связи или свойства, *P,* называют «предикатным символом»; $t_1, t_2, .... t_n$ обозначают объекты, связанные свойством (предикатом) *P* и называют термами.

Термы могут быть только трех следующих типов:

1) константа (обозначает индивидуальный объект или понятие);

2) переменная (обозначает в разное время различные объекты);

3) составной терм – функция $f(t_1, t_2, ... t_n)$, имеющая в качестве своих аргументов *m* термов $t_1, t_2, ... t_m$.

*Примеры.*

1. Предложение «Волга впадает в Каспийское море» можно записать в виде предиката

впадает (Волга, Каспийское море).

«Впадает» - предикатный символ; «Волга» и «Каспийское море» - термы-константы. Мы могли обозначить отношение «впадает» и объекты «Волга» и «Каспийское море» символами.

Вместо термов-констант можно рассматривать переменные:

впадает (X, Каспийское море)

или

впадает *(X,Y).*

Это тоже предикаты.

2. Отношение $x + 1 < y$ можно записать в виде предиката *A(x,y).* Предикатный символ *A* здесь обозначает то, что останется от $x + 1 < y$, если выбросить из этой записи переменные *x* и *y.*

Итак, предикат - это логическая функция, принимающая значения «истина» или «ложь» в зависимости от значений своих аргументов. Количество аргументов *y* предиката называют его **арностью.**

Так, для наших примеров предикат «впадает» имеет арность 2 и при *X* = «Волга», а *Y* = «Каспийское море» истинен, а при *X* = «Дон», *Y* = «Бискайский залив» ложен. Предикат *A* в примере 2 также имеет арность 2, истинен при *X* = 1, *Y* = 3 и ложен при *X*=3, *Y*=1.

Предикаты могут быть объединены в формулы с помощью логических связок (союзов): ^ («и», конъюнкция), v («или», дизъюнкция), ~ («не», отрицание), → («следует», импликация), ↔ («тогда, и только тогда, когда», эквиваленция).

Ниже приведены таблицы истинности этих союзов, позволяющие определить, истинно или ложно значение формулы-связки при различных значениях, входящих в нее предикатов *A* и *B.*

Математически строго формулы логики предикатов определяются рекурсивно:

1) предикат есть формула;

2) если *A* и *B* - формулы, то *A, A^B,A*v*B, A→B*, *A↔B* - тоже формулы;

3) других формул не бывает.

Многие формулы логики предикатов требуют использования **кванторов,** определяющих область значений переменных - аргументов предикатов. Используются кванторы общности (перевернутое A от английского «All» - все) и квантор существования - (перевернутое E от английского «Exists» - «существует»). Запись $\forall x$ читается «для любого *x»,* «для каждого *x»;* $\exists x$ - «существует *x»,* «хотя бы для одного *x».* Кванторы связывают переменные предикатов, на которые они дейст-

вуют, и превращают предикаты в высказывания.

**Таблица 1.12 Истинность связок предикатов (И - истина, Л - ложь)**

| A | B | A^B | AvB | ~A | A→B | A↔B |
|---|---|---|---|---|---|---|
| И | И | И | И | Л | И | И |
| И | Л | Л | И | Л | Л | Л |
| Л | И | Л | И | И | И | Л |
| Л | Л | Л | Л | И | И | И |

Пример.

Введем обозначения: *A(x)* - студент *x* учится отлично; *B(x)* - студент *x* получает повышенную стипендию. Теперь формула *A* (Иванов) → *B* (Иванов) означает: студент Иванов учится отлично, следовательно, студент Иванов получает повышенную стипендию, а формула с квантором общности (Vx) *(A(x) —> B(x))* означает: каждый студент, который учится отлично, получает повышенную стипендию.

Из всевозможных формул нам потребуется только один их вид, называемый **фразами Хорна.** Фразы Хорна содержат в общем случае импликацию и конъюнкцию предикатов *A, B1, B2,. . . , Bn* следующим образом: *B1, B2,. . . Bn → A,* или в более удобных обозначениях:

*A:-B1, B2,...Bn*

(читается: *A* если *B1* и *B2* и ... и *Bn).*

Очевидно, фраза Хорна является формой записи некоего правила, и в дальнейшем будет называться правилом. Предикат *A* называется заголовком или головой правила, а предикаты *B1, B2,... ,Bn*- его подцелями.

Очевидно, что отдельный предикат является частным случаем фразы Хорна: *A.*

Другой частный случай фразы Хорна - правило без головы

*:-B1, B2,...,Bn,*

или

*:-B.*

Такая фраза Хорна называется вопросом. Мы будем записывать «: - *B»* в виде «? - *B»,* а «:- *B1, B2,.... Bn»* в виде «? – *B1, B2,.... Bn».*

Поясним логический смысл такой формулы. Напомним, что импликация *A: - B (B —> A)* может быть выра-жена через отрицание и дизъюнкцию : *~B*v*A* (проверьте это с помощью таблицы истинности). Значит, если отбросить *A,* останется только ~B - отрицание *B.* Формула ← *B1*, ..., *Bn* означает отрицание конъюнкции ~{*B1^B2^...^Bn),* что по закону де Моргана ~(*X*^*Y*) = (~*X*)v(~*Y*) равно (~*B1*)v(~*B2*) v ... v(~*Bn*) - дизъюнкции отрицаний.

Множество фраз Хорна применительно к некоторой проблемной области образует теорию (в логическом смысле).

Пример.

Рассмотрим предметную область: сдачу экзамена по некоторой дисциплине. Введем обозначения:

*A* - студент успешно сдает экзамен.

B - студент посещал занятия.

C - студент освоил учебный материал.

*D* - студент занимался самостоятельно.

*E* - студент подготовил шпаргалку.

Введем знания о предметной области:

*Студент успешно сдаст экзамен, если студент освоил учебный материал.*

*Студент освоил учебный материал, если студент посещал занятия* и *студент занимался самостоятельно. Студент посещал занятия. Студент занимался самостоятельно.*

Форма логической записи:

A:-C;
C:-B,D;
B;
D.

В приведенном примере можно выполнить логический вывод. Так, из истинности фактов *B* и *D* и правила *C:-B, D* следует истинность C, и из правила *A:-C* - истинность предиката *A,* т.е., студент успешно сдаст экзамен. Кроме того, правила *A :-C* и *C:-B, D* можно было бы переписать в виде *A :-B, D.*

В этих случаях используют правила вывода, называемые **методом резолюций.**

Рассмотрим простейшую форму резолюции. Допустим, имеются «родительские» предложения

отрицание: *~A* импликация: *A :-B.*

В результате одного шага резолютивного вывода получаем новое предложение: *B,* которое называется **резольвентой.** В этом случае резолюция соответствует стандартному пропозициональному правилу вывода:

| | |
|---|---|
| допуская, что не | *A* |
| и | *A,* если *B* |
| выводим | не *B.* |

Еще более простой случай:

| | |
|---|---|
| отрицание: | *~A* |
| *ф*акт: | *A.* |

Резольвента - противоречие.

В общем случае имеются родительские предложения

*~(A1^ ... ^An)*
*Ak:-B1,...,Bm* *1 <k <n.*

В качестве резольвенты за один шаг вывода получается *~(A1^...^Ak-1 ^B1^...^Bm^Ak+1^...^An).*

Таким образом, резолюция является подстановкой предикатов - подцелей *B1*,... *Bn,* вместо соответствующего предиката *Ak* из отрицания. Отрицание инициирует логический вывод и поэтому называется запросом (или вопросом) и обозначается *A1*, *A2, ... , An.* Смысл метода резолюций состоит в том, что строится отрицание конъюнкции и проверяется, истинно его значение или ложно. Если значение результирующей конъюнкции ложь, значит, получилось противоречие и, поскольку на старте было отрицание предикатов, выполнено доказательство «от обратного». Если получено значение «истина», то доказательство не выполнено.

*Пример.*

Пусть предикат дает *(X,Y,Z)* означает, что «*X* дает *Y* некоторому объекту Z», а предикат *получают (X, Y)* означает, что «*Y* получает Z». Пусть знания об этих отношениях выражаются предложениями

1) *получает(вы,* сила):- дает (логика, сила, вы);

2) *дает* (логика, сила, вы).

Задача, которую нужно решить, состоит в том, чтобы ответить на вопрос: получаете ли вы

силу ?

Представим этот вопрос в виде отрицания -*получает(вы,* сила). Резолюция предложения 1 и отрицания приводит к ~*дает* (логика, сила, вы), что вместе с фактом 2 приводит к противоречию. Следовательно, ответом исходной задачи является «да».

Пока что мы рассмотрели резолюцию для высказываний или предикатов без переменных. Если же вывод производится для множества предикатов с переменными в качестве аргументов, эти переменные в ходе вывода получают значения соответствующих констант или, как еще говорят, конкретизируются константами.

Рассмотрим это на примере.

*Пример.*

Рассмотрим следующие родительские предложения:

1) ~*получает* (вы. У);

2) *получает (X,* сила) :- *дает* (Z, сила, *X).*

Они содержат три переменные *X,Y* и Z, которые неявно находятся под действием квантора общности. Так, предложение 1 утверждает, что «для всех *Y* вы не получаете *Y*», а 2 - «для всех Z любой *X* получает силу, если Z дает силу *X».* Правило резолюции требует совпадения предиката из отрицания 1 и головы правила 2. Это означает, что переменные получают значения (конкретизируются) соответственно их месту в предложениях 1 и 2 следующим образом: *X = вы, Y* = сила. Предикат *получает (вы,* сила) называется общим примером для предикатов *получает(вы Y*) и *получает (X,* сила).

Изложенные положения логики предикатов находят реализацию и дальнейшее развитие в языке программирования Пролог.

## 12.4. ИНТЕЛЛЕКТУАЛЬНЫЙ ИНТЕРФЕЙС ИНФОРМАЦИОННОЙ СИСТЕМЫ

Анализ развития средств вычислительной техники позволяет утверждать, что она постоянно эволюционирует в двух направлениях. Первое из них связано с улучшением параметров существующих компьютеров, повышением их быстродействия, увеличением объемов их оперативной и дисковой памяти, а также с совершенствованием и модификацией программных средств, ориентированными на повышение эффективности выполнения ими своих функций. Это можно назвать развитием по горизонтали.

Второе направление определяет изменения в технологии обработки информации, приводящие к улучшению использования компьютерных систем. Развитие в этом направлении связано с появлением новых типов компьютеров и качественно новых программных средств, дополняющих уже существующие. Такое развитие можно назвать развитием по вертикали.

Развитие программных средств идет по пути увеличения их дружественности, т.е. такого упрощения управления ими, что от пользователя не требуется специальной подготовки, и система создает максимально комфортные условия для его работы. Основной ориентир в совершенствовании вычислительных систем - превращение их в удобного партнера конечного пользователя при решении задач в ходе его профессиональной деятельности.

Для обеспечения наибольшей дружественности интерфейса программного средства с пользователем первый должен стать интеллектуальным. **Интеллектуальный интерфейс,** обеспечивающий непосредственное взаимодействие конечного пользователя и компьютера при решении задачи в составе человеко-машинной системы, должен выполнять три группы функций:

• обеспечение для пользователя возможности постановки задачи для ЭВМ путем сообщения только ее условия (без задания программы решения);

• обеспечение для пользователя возможности формирования сред решения задачи с использованием только терминов и понятий из области профессиональной деятельности пользователя, естественных форм представления информации;

• обеспечение гибкого диалога с использованием разнообразных средств, в том числе не регламентируемых заранее, с коррекцией возможных ошибок пользователя.

Структура системы (рис. 1.48), удовлетворяющей требованиям новой технологии решения задач, состоит из трех компонент:

• **исполнительной системы**, представляющей собой совокупность средств, обеспечивающих выполнение программ;

• **базы знаний**, содержащей систему знаний о проблемной среде;

• **интеллектуального интерфейса**, обеспечивающего возможность адаптации вычислительной системы к пользователю.

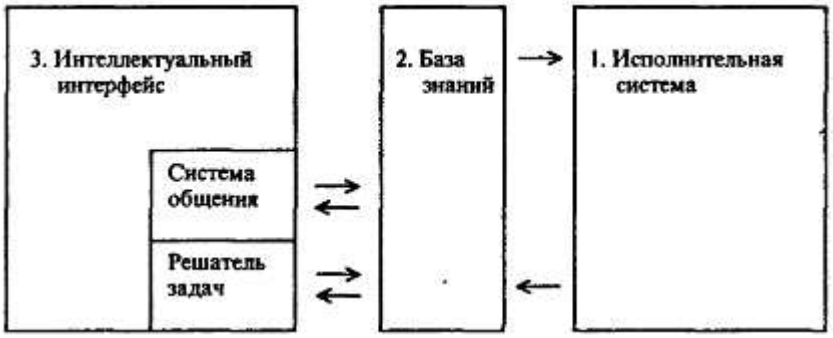



*Рис. 1.48.* Структура современной системы решения прикладных задач

Такая система существенно отличается от создававшихся на более ранних этапах развития информатики и вычислительной техники. Путь реализации новейших информационных технологий предполагает использование вычислительных систем, построенных на основе представления знаний предметной области задачи и интеллектуального интерфейса.

## 12.5. СТРУКТУРА СОВРЕМЕННОЙ СИСТЕМЫ РЕШЕНИЯ ПРИКЛАДНЫХ ЗАДАЧ

Разработки систем искусственного интеллекта шли сначала по пути моделирования общих интеллектуальных функций индивидуального сознания. Однако, развитие вычислительной техники и программного обеспечения в 90-х годах опровергает прогнозы предыдущих десятилетий о скором переходе к ЭВМ 5-го поколения. Интеллектуальные функции основной массы программных систем общения на естественном языке пока не находят широкого внедрения в промышленных масштабах.

Характерную инфляцию претерпело такое понятие, как «новая информационная технология». Первоначально это понятие означало интеллектуальный интерфейс к базе данных, позволяющий прикладным пользователям общаться с ней непосредственно на естественном языке. Ныне под «новыми информационными технологиями» понимают просто технологии, существенно использующие вычислительную технику в обработке информации, в том числе основанные на применении текстовых и табличных процессоров, а также информационных систем.

Столкнувшись с непреодолимыми проблемами, разработчики систем, обладающих «общим» искусственным интеллектом, пошли по пути все большей и большей специализации, вначале по направлению к экспертным системам, затем - к отдельным очень специфичным интеллектуальным функциям, встроенным в инструментальные программные средства, не считавшиеся до настоящего времени сферой разработок по искусственному интеллекту. Например, такие системы сейчас часто обладают возможностями аналитических математических вычислений, перевода технических и деловых текстов, распознавания текста при вводе сканером, синтаксического анализа фраз и предложений, самонастраиваемостью и т.д.

Парадигма исследований и разработок в области искусственного интеллекта постепенно пересматривается. По-видимому, возможности скорого развития программных систем, моделирующих интеллектуальные функции индивидуального сознания, в значительной мере исчерпаны. Необходимо обратить внимание на новые возможности, которые открывают в отношении общественного сознания информационные системы и сети. Развитие вычислительных систем и сетей ведет» по-видимому, к созданию нового типа общественного сознания, в которое информационные средства будут органично встроены как технологическая среда обработки и передачи информации. После этого человечество получит именно гибридный человеко-машинный интеллект не столько в масштабе индивидуального сознания, сколько в сфере социальной практики.

***Контрольные вопросы и задания***

1. Какова теория возникновения и развития исследований по искусственному интеллекту?
2. Каковы отличительные черты задач из сферы искусственного интеллекта?

3. Охарактеризуйте направления исследований по искусственному интеллекту.

4. Что такое «знания» с точки зрения систем искусственного интеллекта?

5. В чем состоит метод представлений знаний с помощью продукций?

6. На чем основано представление знаний с помощью семантической сети?

7. Как фреймовые системы могут использоваться для представления знаний?

8. В чем отличия представления знаний в интеллектуальных системах от представления просто данных?

9. Что значит понятие «предикат»?

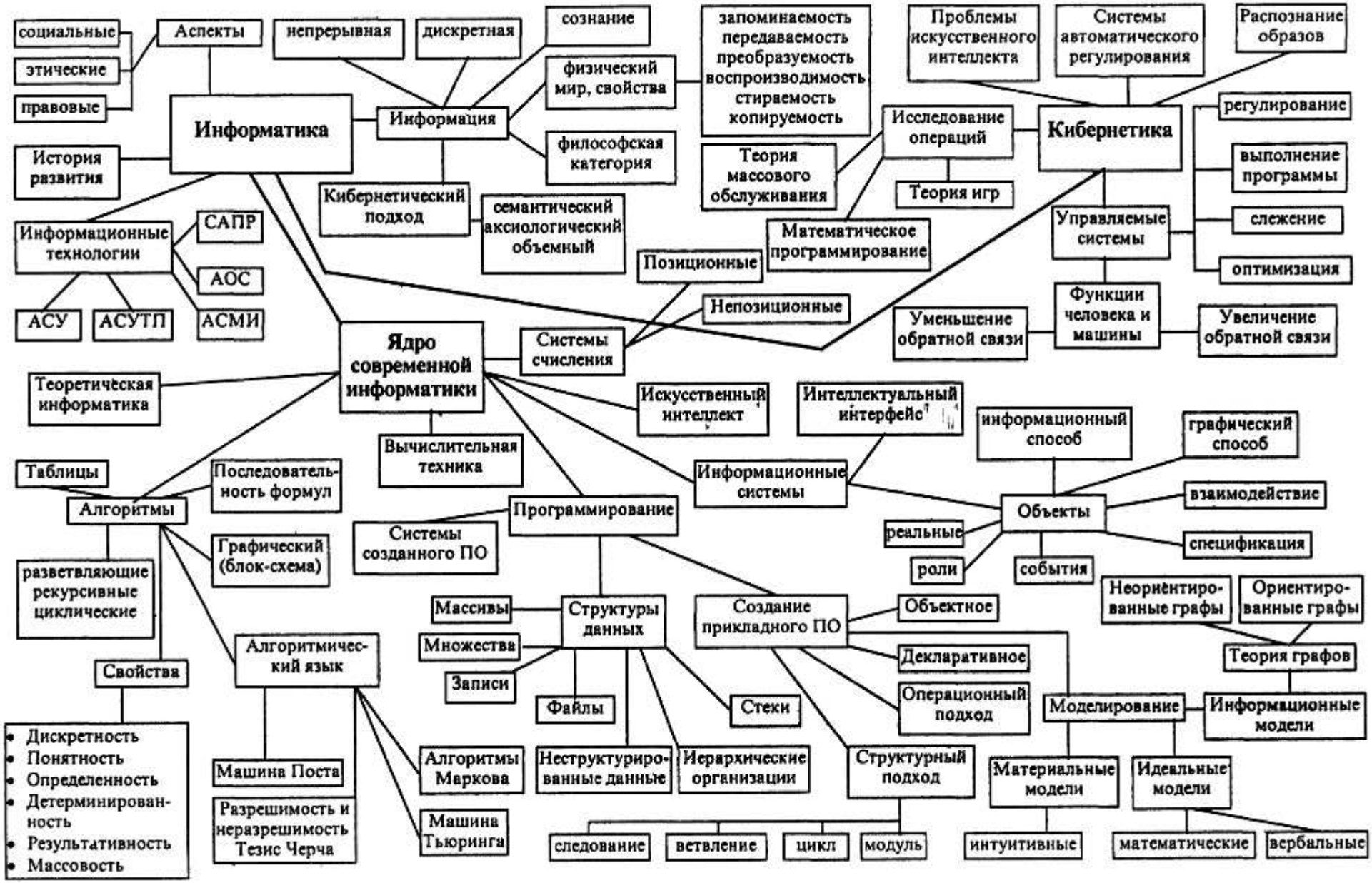
10. Что такое «фраза Хорна»?

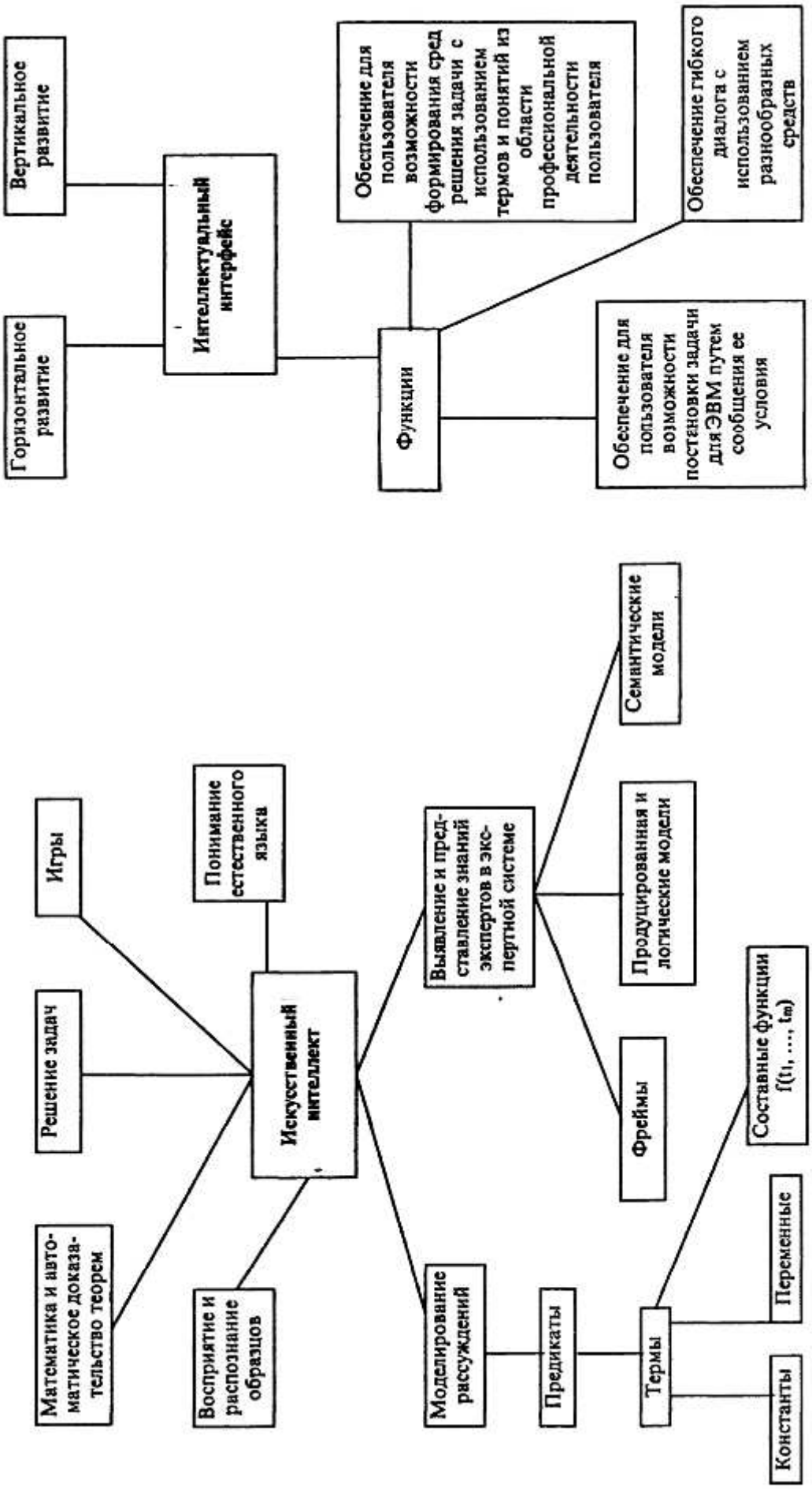
11. Как происходит логический вывод с помощью метода резолюций?

12. В каком направлении развиваются интерфейсные части информационных систем?

13. В чем состоит принцип дружественности программных средств?

14. Какова структура перспективных информационных систем будущего?


Информатика
Аспекты
социальные
этические
правовые
История развития
Информационные технологии
САПР
АОС
АСМИ
АСУ
АСУТП
Информация
непрерывная
дискретная
сознание
физический мир, свойства
философская категория
запоминаемость
передаваемость
преобразуемость
воспроизводимость
стираемость
копируемость
Кибернетический подход
семантический
аксиологический
объемный
Ядро современной информатики
Системы счисления
Позиционные
Непозиционные
Теоретическая информатика
Вычислительная техника
Искусственный интеллект
Интеллектуальный интерфейс
Информационные системы
Проблемы искусственного интеллекта
Системы автоматического регулирования
Распознание образов
Кибернетика
Исследование операций
Теория массового обслуживания
Теория игр
Математическое программирование
регулирование
выполнение программы
слежение
оптимизация
Управляемые системы
Функции человека и машины
Уменьшение обратной связи
Увеличение обратной связи
Объекты
информационный способ
графический способ
взаимодействие
спецификация
реальные
роли
события
Программирование
Системы созданного ПО
Создание прикладного ПО
Объектное
Декларативное
Операционный подход
Структурный подход
Структуры данных
Массивы
Множества
Записи
Файлы
Стеки
Неструктурированные данные
Иерархические организации
Алгоритмы
Таблицы
Последовательность формул
Графический (блок-схема)
разветвляющие рекурсивные циклические
Свойства
Дискретность
Понятность
Определенность
Детерминированность
Результативность
Массовость
Алгоритмический язык
Машина Поста
Разрешимость и неразрешимость Тезис Черча
Алгоритмы Маркова
Машина Тьюринга
следование
ветвление
цикл
модуль
Моделирование
Материальные модели
Идеальные модели
интуитивные
математические
вербальные
Неориентированные графы
Ориентированные графы
Теория графов
Информационные модели




**Дополнительная литература к главе 1**

1. *Абрамов Ю. Ф.* Картина мира и информация (философские очерки). - Иркутск: ИГУ,

1988.

2. *Абдеев Р. Ф.* Философия информационной цивилизации. ~ М.: ВЛАДОС, 1994.

*3.Аиламазян А. К., Стась Е.В.* Информатика и теория развития. - М.: Наука, 1992.

4. *Ахундов М.Д.* Эволюция и смена научных картин мира. В кн.: Философия, естествознание, социальное развитие. - М.: Наука, 1989.

5. *Брой М.* Информатика. Основополагающее введение. В 3-х частях. - М.: Диалог-МИФИ, 1996.

6. Будущее искусственного интеллекта. - М.: Наука, 1991.

7. *Вейценбаум Дж.* Возможности вычислительных машин и человеческий разум. -М.: Радио и связь, 1982.

8. *Венда В. Ф.* Системы гибридного интеллекта: эволюция, психология, информатика. - М.: Машиностроение, 1990.

9. *Вирт Н.* Алгоритмы и структуры данных. - М.: Мир, 1989.

10. *Вирт Н.* Алгоритмы + структура данных = программы. - М.: Мир, 1985.

11. *Глинский Б. А.* Философские и социальные аспекты информатики. - М.: Наука, 1990.

12. *Глушков В. М.* Основы безбумажной информатики. /Изд. 2-е. - М.: Наука, 1987.

13. *Глушков В. М.* Кибернетика. Вопросы теории и практики. - М.: Наука, 1986.

14. *Голицын Г. А., Петров В. М.* Информация, поведение, творчество. - М.: Наука, 1991.

15. *Гриценко В. И., Паньшин Б.Н.* Информационная технология: вопросы развития и применения. - Киев: Наукова Думка, 1988.

16. *Громов Г. Р.* Национальные информационные ресурсы. Проблемы промышленной эксплуатации. - М.: Наука, 1985.

17. *Громов Г. Р.* Очерки информационной технологии. - М.: Инфоарт, 1992.

18. *Дмитриев В. И.* Прикладная теория информации. - М.: Высшая школа, 1989.

19. Интеллектуальные процессы и их моделирование. - М.: Наука, 1987.

20. Информатизация общества и бизнес. - М.: ИНИОН, 1992.

21. Информатика и культура. - М. Наука, 1990.

22. Информационая революция: наука, экономика, технология. Серия «Информация. Наука. Общество». - М.: ИНИОН, 1993.

23. Искусственный интеллект: - В 3-х кн. Кн.3. Программные и аппаратные средства: Справочник / Под ред В.Н.Захарова, В.Ф.Хорошевского. - М.: Радио и связь, 1990.

24. *Каныгин Ю. М., Калитич Г. И.* Основы теоретической информатики. - Киев: Наукова Думка, 1990.

25. Когнитивная психология и искусственный интеллект. Серия «Информация. Наука. Общество». - М.: ИНИОН, 1993.

26. *Лорьер Ж.-Л.* Системы искусственного интеллекта. - М.: Финансы и статистика, 1982.

27. *Марков М.* Технология и эффективность социального управления. - М.: Прогресс, 1982.

28. *Матросов В.Л.* Теория алгоритмов. - М.: Прометей, 1989.

29. *Минский М.* Вычисления и автоматы. - М.: Мир, 1971.

30. *Новик И. Б., Абдуллаев Ф.Ш.* Введение в информационный мир. - М.: Наука, 1991.

31. Перспективы информатизации общества. Ч.1,2. Серия «Информация. Наука. Общество». - М.: ИНИОН, 1990.

32. *Першиков В. И., Савинков В. М.* Толковый словарь по информатике. - М.: Финансы и статистика,1991.

33. *Попов Э.П.* Экспертные системы: Решение неформализованных задач в диалоге с ЭВМ. - М.: Наука, 1987.

34. *Поспелов Г. С.* Искусственный интеллект - основа новой информационной технологии. - М.: Наука, 1988.

35. *Рахитов А. И.* Философия компьютерной революции. - М.: ИПЛ, 1991.

36. *Свириденко С. С.* Современные информационные технологии. - М.: Радио и связь, 1989.

37. Словарь по кибернетике / Под ред. В.С.Михалевича. - Киев: Гл. ред. УСЭ, 1989.

38. *Советов Б. Я.* Информационная технология. - М.: Высшая школа, 1992.

39. Становление информатики. Сб. статей из серии «Кибернетика». - М.:, Наука, 1986.

40. *Суханов А. П.* Информация и прогресс. - Новосибирск: Наука, 1988.

41. Толковый словарь по искусственному интеллекту / Авторы-составители А-Н.Аверкин,

М.Г.Гаазе-Рапопорт, Д.А.Поспелов. - М.: Радио и связь, 1992.

42. *УинстонП.* Искусственный интеллект. - М.: Мир, 1980.

43. *Урсул А.Д.* Информатизация общества. Введение в социальную информатику. - М.:АОН,1990.

44. *Успенский В. А.* Машина Поста. - М.: Наука, 1988

45. *Успенский В. А., Семенов А. Л.* Террия алгоритмов: основные открытия и приложения. - М.: Наука, 1987.

46. *Хартли Р.* Передача информации. В сб. «Теория информации и ее приложения». - М.: ИЛ, 1959.

47. *Шеннон К.-Э.* Работы по теории информации и кибернетике. - М.: ИЛ, 1963.

# *ГЛАВА 2*

# ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ

> Многие вещи нам непонятны не потому, что наши понятия слабы, но потому, что сии вещи не входят в круг наших понятий
>
> *Козьма Прутков*

## ВВЕДЕНИЕ

Давно ушли в прошлое времена компьютеров первых поколений, когда аппаратные средства (часто называемые в нашей литературе жаргонным словом «железо» или американским термином «hardware») были главным предметом вожделения специалистов. В те времена, в 50-60-е годы, вопрос о программных средствах стоял так: если есть - хорошо, нет - сами напишем, только дайте ЭВМ. Сегодня, прежде всего в силу экономических обстоятельств, т.е. поменявшегося соотношения стоимости в диаде «аппаратные средства / программные средства» (последние часто зовутся «software»), и высокой стоимости разработки удовлетворяющих современным требованиям, но отсутствующих по той или иной причине, программных средств, при приобретении компьютера чаще обращают внимание на наличие для него доступного (по факту и по цене) программного обеспечения. На гигантских международных выставках типа CEBIT новинки программного обеспечения уже давно доминируют над новыми аппаратурными разработками. Полный комплект программного обеспечения, необходимого для организации, скажем, автоматизированного рабочего места (АРМ) инженера-проектировщика, научного работника (физика, химика, биолога и т.д.) по стоимости превосходит (порой в несколько раз) стоимость компьютера адекватного класса.

Всевозможные программные средства, которых, видимо, насчитывается уже сотни тысяч для компьютеров различных типов, можно разделить на несколько классов в зависимости от назначения:

- операционные системы;
- системы программирования;
- инструментальные программные средства, интегрированные пакеты;
- прикладные программы.

Эти классы программных средств будут подробно рассмотрены в настоящей главе. В ней упоминаются десятки программных средств, о каждом из которых написано немало отдельных книг. О некоторых из этих программ ниже дано достаточно информации, чтобы начать работу с ними, но ни об одной нет исчерпывающих сведений. Для более полного знакомства с этими программными средствами нужно обратиться к встроенной помощи, справочникам, многочисленным книгам. Цель этой главы - сформировать понимание принципов работы с основными видами программного обеспечения персональных компьютеров, обработки информации с их помощью.

## § 1. ОПЕРАЦИОННЫЕ СИСТЕМЫ

### 1.1. НАЗНАЧЕНИЕ И ОСНОВНЫЕ ФУНКЦИИ ОПЕРАЦИОННЫХ СИСТЕМ

Особое место среди программных средств всех типов занимают операционные системы, являясь ядром программного обеспечения.

Операционная система - это комплекс программ, обеспечивающих

• управление ресурсами, т.е. согласованную работу всех аппаратных средств компьютера;

•управление процессами, т.е. выполнение программ, их взаимодействие с устройствами компьютера, с данными;

• пользовательский интерфейс, т.е. диалог пользователя с компьютером, выполнение определенных простых команд - операций по обработке информации.

Такое определение операционной системы уже апеллирует к ее функциям, поэтому рассмотрим эти функции подробнее.

Операционные системы — наиболее машиннозависимый вид программного обеспечения, ориентированный на конкретные модели компьютеров, поскольку они напрямую управляют их устройствами или, как еще говорят, обеспечивают интерфейс между пользователем и аппаратной частью компьютера.

В той мере, в какой это необходимо для понимания функций операционных систем, аппаратную часть компьютера можно представлять себе состоящей из следующих элементов:

• центрального процессора, имеющего определенную архитектуру (структуру регистров, набор и форму представления команд, формат обрабатываемых данных и т.д.) и характеризующегося производительностью, т.е. количеством простейших операций, выполняемых в единицу времени, а также другими качествами;

• оперативной памяти, характеризующейся емкостью (объемом) и скоростью обмена данными (прежде всего с центральным процессором);

• периферийных устройств, среди которых имеются

• устройства ввода (клавиатура, мышь, сканер и др.);

• устройства вывода (дисплей, принтер, графопостроитель и др.);

• внешние запоминающие устройства (дисководы для магнитных и оптических дисков, устройства для работы с лентами и др.);

• мультимедийные устройства.

Все эти аппаратные устройства обобщенно называют ресурсами компьютера.

В сравнении с оперативной памятью внешние запоминающие устройства обладают практически неограниченной емкостью. Так, емкость встроенного накопителя персональных компьютеров - винчестера - обычно в 50-100 раз больше объема оперативной памяти. Для других устройств - накопителей на гибких магнитных дисках и оптических дисках - используются сменные носители информации, однако время доступа к информации на внешних запоминающих устройствах значительно больше, чем к информации в оперативной памяти (в тысячи раз). Медленнее, чем центральный процессор, работают и устройства ввода - вывода.

За время существования компьютеров операционные системы претерпели значительную эволюцию. Так, первые операционные системы были однопользовательскими и однозадачными. Эффективность использования ресурсов компьютера в этом случае оказывалось невысокой из-за простоев всех, кроме одного работающего периферийного устройств компьютера. Например, при вводе данных простаивал центральный процессор, устройства вывода и внешние запоминающие устройства.

По мере роста возможностей, производительности и изменениях в соотношении стоимости устройств компьютера положение стало нетерпимым, что привело к появлению **многозадачных** операционных систем, остававшихся однопользовательскими.

Такие операционные системы обеспечивают постановку заданий в очередь на выполнение, параллельное выполнение заданий, разделение ресурсов компьютера между выполняющимися заданиями. Так, например, одно задание может выполнять ввод данных, другое - выполняться центральным процессором, третье - выводить данные, четвертое - стоять в очереди. Важнейшее техническое решение, обусловившее такие возможности, - появление у внешних устройств собственных процессоров (контроллеров).

При многозадачном режиме

• в оперативной памяти находится несколько заданий пользователей;

• время работы процессора разделяется между программами, находящимися в оперативной памяти и готовыми к обслуживанию процессором;

• параллельно с работой процессора происходит обмен информацией с различными внешними устройствами.

Наиболее совершенны и сложны **многопользовательские многозадачные** операционные системы, которые предусматривают одновременное выполнение многих заданий многих пользователей, обеспечивают **разделение ресурсов** компьютера в соответствии с приоритетами пользователей и **защиту данных** каждого пользователя от несанкционированного доступа. В этом случае операционная система работает в режиме **разделения времени,** т.е. обслуживает многих пользователей, работающих каждый со своего терминала.

Суть режима разделения времени состоит в следующем. Каждой программе, находящейся в оперативной памяти и готовой к исполнению, выделяется для исполнения фиксированный, задаваемый в соответствии с приоритетом пользователя интервал времени (интервал мультиплексирования). Если программа не выполнена до конца за этот интервал, ее исполнение принудительно прерывается, и программа переводится в конец очереди. Из начала очереди извлекается следующая программа, которая исполняется в течение соответствующего интервала мультиплексирования, затем поступает в конец очереди и т.д. в соответствии с циклическим алгоритмом. Если интервал мультиплексирования достаточно мал (~200 мс), 'а средняя длина очереди готовых к исполнению программ невелика (~10), то очередной квант времени выделяется программе каждые 2 с. В этих условиях ни один из пользователей практически не ощущает задержек, так как они сравнимы со временем реакции человека.

Одной из разновидностей режима разделения времени является фоновый режим, когда программа с более низким **приоритетом** работает на фоне программы с более высоким приоритетом. Работа в фоновом режиме реального времени аналогична работе секретаря руководителя. Секретарь занимается текущими делами до тех пор, пока начальник не дал срочное поручение.

Помимо рассмотренных режимов организации вычислительного процесса, все большее распространение получаст схема, при которой ЭВМ управляет некоторым внешним процессом, обрабатывая данные и информацию, непосредственно поступающую от объекта управления. Поскольку определяющим фактором являются реально поступающие от объекта управления данные, такой режим называют **режимом реального времени,** а его организация возлагается на специализированную операционную систему.

Остановимся на некоторых понятиях, важных для понимания принципов функционирования всех операционных систем (ОС).

Понятие **процесса** играет ключевую роль и вводится применительно к каждой программе отдельного пользователя. Управление процессами (как целым, так и каждым в отдельности) - важнейшая функция ОС. При исполнении программ на центральном процессоре следует различать следующие характерные состояния (рис. 2.1):

• ***порождение-*** подготовку условий для исполнения процессором;

• ***активное состояние*** (или «Счет») - непосредственное исполнение процессором;

• ***ожидание*** - по причине занятости какого-либо требуемого ресурса;

• ***готовность*** - программа не исполняется, но все необходимые для исполнения программы ресурсы, кроме центрального процессора, предоставлены;

• ***окончание*** - нормальное или аварийное завершение исполнения программы, после которого процессор и другие ресурсы ей не предоставляются.

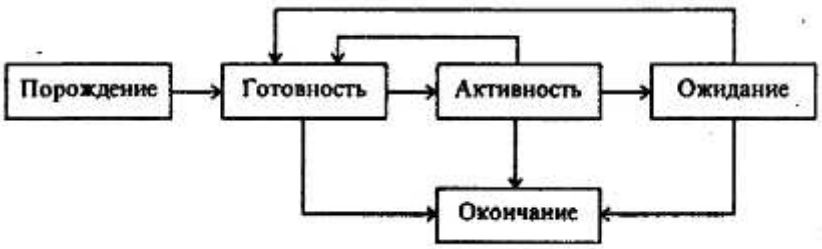



*Рис. 2.1.* Граф состояний переходов процесса из одной фазы в другую

Понятие «ресурс» применительно к вычислительной технике следует понимать как функциональный элемент вычислительной системы, который может быть выделен процессу на определенный промежуток времени. Наряду с физическими ресурсами -реальными устройствами ЭВМ - средствами современных операционных систем могут создаваться и использоваться виртуальные (воображаемые) ресурсы, являющиеся моделями физических. По значимости виртуальные ресур-

сы - одна из важнейших концепций построения современных ОС. Виртуальный ресурс представляет собой модель некоего физического ресурса, создаваемую с помощью другого физического ресурса. Например, характерным представителем виртуального ресурса является оперативная память. Компьютеры, как правило, располагают ограниченной по объему оперативной памятью (физической). Функционально ее объем может быть увеличен путем частичной записи содержимого оперативной памяти на магнитный диск. Если этот процесс организован так, что пользователь воспринимает всю расширенную память как оперативную, то такая «оперативная» .память называется виртуальной.

Наиболее законченным проявлением концепции виртуальности является понятие виртуальной машины, являющееся исходным при программировании на языках высокого уровня, например Паскале. Виртуальная машина есть идеализированная модель реальной машины, изолирующая пользователя от аппаратных особенностей конкретной ЭВМ, воспроизводящая архитектуру реальной машины, но обладающую улучшенными характеристиками:

• бесконечной по объему памятью с произвольно выбираемыми способами доступа к ее данным;

• одним (или несколькими) процессами, описываемыми на удобном для пользователя языке программирования;

• произвольным числом внешних устройств произвольной емкости и доступа. Концепция **прерываний** выполнения программ является базовой при построении любой операционной системы. Из всего многообразия причин прерываний необходимо выделить два вида: первого и второго рода. Системные причины прерываний первого рода возникают в том случае, когда у процесса, находящегося в активном состоянии, возникает потребность либо получить некоторый ресурс или отказаться от него, либо выполнить над ресурсом какие-либо действия. К этой группе относят и, так называемые, внутренние прерывания, связанные с работой процессора (например, арифметическое переполнение или исчезновение порядка в операциях с плавающей запятой). Системные причины прерывания второго рода обусловлены необходимостью проведения синхронизации между параллельными процессами.

При обработке каждого прерывания должна выполняться следующая последовательность действий:

• восприятие запроса на прерывание;

• запоминание состояния прерванного процесса, определяемое значением счетчика команд и других регистров процессора;

• передача управления прерывающей программе, для чего в счетчик команд заносится адрес, соответствующий данному типу прерывания;

• обработка прерывания;

• восстановление прерванного процесса.

В большинстве ЭВМ первые три этапа реализуются аппаратными средствами, а остальные-блоком программ обработки прерываний операционной системы.

В настоящее время используется много типов различных операционных систем для ЭВМ различных видов, однако в их структуре существуют общие принципы. В составе многих операционных систем можно выделить некоторую часть, которая является основой всей системы и называется ядром. В состав ядра входят наиболее часто используемые модули, такие как модуль управления системой прерываний, средства по распределению таких основных ресурсов, как оперативная память и процессор. Программы, входящие в состав ядра, при загрузке ОС помещаются в оперативную память, где они постоянно находятся и используются при функционировании ЭВМ. Такие программы называют резидентными. К резидентным относят также и программы-драйверы, управляющие работой периферийных устройств. Важной частью ОС является **командный процессор** - программа, отвечающая за интерпретацию и исполнение простейших команд, подаваемых пользователем, и его взаимодействие с ядром ОС. Кроме того, к операционной системе следует относить богатый набор **утилит** - обычно небольших программ, обслуживающих различные устройства компьютера (например, утилита форматирования магнитных дисков, утилита восстановления необдуманно удаленных файлов и т.д.).

## 1.2. ПОНЯТИЕ ФАЙЛОВОЙ СИСТЕМЫ

При наличии большого числа программ и данных необходим строгий их учет и системати-

зация. Операционным системам приходится работать с различными потоками данных, разными аппаратными и периферийными устройствами компьютера. Организовать упорядоченное управление всеми этими объектами позволяет файловая система.

На операционные системы персональных компьютеров наложила глубокий отпечаток концепция файловой системы, лежащей в основе операционной системы UNIX. В ОС UNIX подсистема ввода-вывода унифицирует способ доступа как к файлам, так и к периферийным устройствам. Под файлом при этом понимают набор данных на диске, терминале или каком-либо другом устройстве. Таким образом, файловая система - это система управления данными.

Файловые системы операционных систем создают для пользователей некоторое виртуальное представление внешних запоминающих устройств ЭВМ, позволяя работать с ними не на низком уровне команд управления физическими устройствами (например, обращаться к диску с учетом особенностей его адресации), а на высоком уровне наборов и структур данных. Файловая система скрывает от программистов картину реального расположения информации во внешней памяти, обеспечивает независимость программ от особенностей конкретной конфигурации ЭВМ, или, как еще говорят, логический уровень работы с файлами. Файловая система также обеспечивает стандартные реакции на ошибки, возникающие при обмене данными. Пользователь, работая в контексте определенного языка программирования, обычно использует файлы как поименованные совокупности данных, хранимые во внешней памяти и имеющие определенную структуру. При работе с файлами пользователю предоставляются средства для создания новых файлов, операции по считыванию и записи информации и т.д., не затрагивающие конкретные вопросы программирования работы канала по пересылке данных, по управлению внешними устройствами.

Наиболее распространенным видом файлов, внутренняя структура которых обеспечивается файловыми системами различных ОС, являются файлы с последовательной структурой. Такого рода файлы можно рассматривать как набор составных элементов, называемых логическими записями (или блоками), длина которых может быть как фиксированной, так и переменной, и доступ к которым - последовательный, т.е. для обработки (считывания или записи) *i*-й записи должна быть обработана предыдущая (*i-1*)-я запись.

В ряде файловых систем предусматривается использование более сложных логических структур файлов, чем последовательная. Например, записи в файле могут образовывать древовидные структуры, может использоваться индексно-последовательная организация файлов (с упорядочением записей по значению некоторых полей) или, так называемая, библиотечная структура файлов, использующая уровень учетной информации (каталога), облегчающей поиск и доступ к отдельным компонентам файлов. На физическом уровне блоки файла (обычно размером 256 или 512 байт) могут размещаться в памяти непрерывной областью или храниться несмежно. Первый способ хранения файлов, реализованный, например, в ОС РАФОС, приводит к затруднениям при изменении размеров файлов (т.е. к необходимости перезаписи файлов, если их длина увеличивается, или хранения «дыр», если длина уменьшается).

Наиболее развитый механизм несмежного распределения блоков файлов реализован в операционной системе UNIX, в которой размеры файлов могут динамически изменяться в пределах 1 Гбайта. Каждый файл в системе имеет дескриптор, в составе которого хранится список, содержащий 13 номеров блоков на диске и используемый для адресации к тем блокам, которые входят в состав файла. Первые десять элементов списка непосредственно указывают на десять блоков, в которых размещаются данные файла. В одиннадцатом элементе списка указан номер блока, хранящий список из 128 номеров блоков данных, которые принадлежат файлу (это первый уровень косвенной адресации). Двенадцатый элемент ссылается на блок, который содержит список из 128 номеров блоков первого уровня косвенной адресации (это второй уровень косвенной адресации). С помощью тринадцатого элемента указывается ссылка на блок, содержащий список из 128 номеров блоков второго уровня косвенной адресации.

Роль учетного механизма, позволяющего обслуживать десятки и сотни файлов, в файловой системе очень важна. Общим приемом является сведение учетной информации о расположении файлов на магнитном диске в одно место - его каталог (директорий). Каталог представляет собой список элементов, каждый из которых описывает характеристики конкретного файла, используемые для организации доступа к этому файлу - имя файла, его тип, местоположение на диске и длину файла. В простых операционных системах (например ОС РАФОС) местоположение единственного каталога на магнитном диске (дискете) и его размер фиксированы. В более сложных сис-

темах каталог может находиться в любом месте диска, но на него должна иметься ссылка в, так называемой, метке тома, находящейся в фиксированном месте и формируемой при инициализации диска. Более того, каталогов может быть большое число и они могут быть логически связаны в какие-либо информационные структуры. Так, наиболее развитая многоуровневая файловая система UNIX поддерживает иерархическую (древовидную) систему каталогов (рис.2.2). Каждый пользователь может работать в составе этой структуры со своей системой каталогов (со своим поддеревом). Полное имя файла в данной структуре задает путь переходов между каталогами в логической структуре каталогов.


*Рис. 2.2.* Иерархическая система каталогов

Файл обладает уникальным идентификатором (именем), обеспечивающим доступ к файлу. Идентификатор включает в себя собственно имя - буквенно-цифровое обозначение файла, которое может содержать специальные символы (подчеркивание, дефис, ! и т.д.), и расширение имени файла (обычно отделяемое от имени файла точкой). Если имена создаваемых файлов пользователь может задавать произвольно, то в использовании расширений следует придерживаться традиции, согласно которой расширение указывает на тип файла, характер его содержимого. Например, в операционной системе MS-DOS файлы с расширениями

.com
.exe - исполняемые
.bat
.txt - текстовые
.doc
.pas - тексты программ на известных языках программирования: Паскаль, Бейсик,
.bas Си, Фортран, соответственно
.c
.for
.dbf - файл базы данных.

Известны десятки стандартных расширений, используемых при работе с различными программными системами.

В различных ОС существуют определенные ограничения на длину имени и расширения имени файла. Так, в MS-DOS длина имени файла не должна превышать восьми символов, а расширение - трех. В ОС UNIX ограничения значительно менее жесткие.

Имена директорией, начиная от корневого, образующие **путь к файлу**, отделяемые при записи друг от друга косой чертой (\ в DOS, / в UNIX), также как и обозначение диска, относятся к идентификатору файла. Например, в MS-DOS

d:\lang\pascal\work\example.pas

есть файл с именем example и расширением pas, указывающем на то, что это текст программы на Паскале, полный путь к которому:

d:- диск d;

\lang\pascal\work - это структура вложенных директорией, в самом внутреннем из которых находится необходимый файл example.pas.

Каждый каталог рассматривается как файл, имеет собственное имя. Продвижение по дереву при поиске некоторого каталога или файла возможно как вниз по дереву от текущего узла, так и вверх в направлении к корню. В каждом каталоге хранится список имен файлов, а также ссылки на дескрипторы файлов. В дескрипторах сосредоточена подробная информация о файле (список номеров блоков, занимаемых файлом, метод доступа к файлу, дата создания файла, идентификатор владельца, тип файла). В процессе работы могут создаваться новые каталоги и вписываться в требуемое место иерархии.

Файловая система ОС обеспечивает основные операции над файлами: их открытие (что сопровождается копированием учетной информации о файле, обеспечивающей быстрый доступ к нему, в некоторую структуру данных, расположенную в оперативной памяти, подготовкой буферов и каналов для передачи информации), копирование, перемещение, объединение, удаление, закрытие. Вторую группу представляют операции чтения и записи составных элементов файла. Особая группа операций обеспечивает печать содержимого каталогов или файлов, управление правами доступа к файлам, поиска файлов и т.д.

Развитые многопользовательские файловые системы обеспечивают также защиту и разделение данных, хранящихся в файлах, при работе с ними разных пользователей. Так, например, после входа в систему UNIX (который производится по паролю) пользователь получает доступ к ряду системных, групповых и личных каталогов и файлов. Каждый файл и каталог имеет владельца. Обычно это пользователь, создавший их. Владелец может затем назначить тип защиты файла от трех категорий пользователей:

• владельца (самого себя);

• представителей той же группы пользователей, что и владелец (понятие группы полезно при совместной работе над какими-либо проектами);

• всех остальных пользователей системы.

Каждый файл (каталог) имеет три вида разрешения на доступ:

• чтение (r - read) - можно просматривать содержимое файла (каталога);

•запись (w - write) - можно менять содержимое файла (создавать или удалять файлы в каталоге);

• выполнение (x - execute) - можно использовать файл как команду UNIX.

Комбинация видов доступа к файлу записывается последовательностью 9 символов:


Отсутствие права доступа обозначается минусом. Например: rwxr-x--x - файл может быть просмотрен, изменен и запущен на выполнение владельцем, просмотрен и выполнен членами группы, к которой относится владелец, и выполнен всеми остальными пользователями системы.

## 1.3. ОПЕРАЦИОННЫЕ СИСТЕМЫ ДЛЯ КОМПЬЮТЕРОВ ТИПА IBM PC

### Общие сведения о MS DOS

Операционные системы для персональных ЭВМ за время существования этого класса компьютеров с 1975 г. претерпели значительное развитие, сопровождавшееся увеличением разрядности персональных компьютеров (ПК) от 8 до 32, расширением возможностей, улучшением интерфейса с пользователем (табл.2.1).

**Таблица 2.1 Некоторые типы ОС для персональных компьютеров**

| ПК | | |
|---|---|---|
| 8-разрядные | 16-разрядные | 32-разрядные |

| | | |
|---|---|---|
| P/M-80, MSX DOS,<br>МикроDOS,<br>Микрос-80 | MS-DOS, РАФОС,<br>ОС DBK, ИНМОС | UNIX, XENIX,<br>Windows 95,<br>OS/2 |

8-разрядные ОС сохраняют значение в качестве операционных систем простейших учебных и бытовых (игровых) компьютеров. Из-за ограниченного адресного пространства оперативной памяти (65 кбайт) серьезные профессиональные применения таких компьютеров невозможны.

16-разрядные IBM-совместимые компьютеры составляют значительную часть парка профессиональных персональных компьютеров в нашей стране. Самая распространенная ОС для этих компьютеров - однопользовательская однозадачная MS DOS (компании MicroSoft - сокращенно MS; DOS - английская аббревиатура названия «дисковая операционная система»). Первая версия этой ОС была создана одновременно с персональным компьютером IBM PC в 1981 г. и из внешних устройств поддерживала лишь накопители на гибких дисках с дискетами на 160 кбайт. Версия 2.0 связана с появлением модификации PC XT, поддерживала также накопители на жестких дисках до 10 Мбайт, древовидную файловую структуру. Популярная на протяжении ряда лет версия 3.3 (1987 г.) - для поддержки PC AT. Эта модификация ОС адресует 640 кбайт оперативной памяти, что в момент ее появления было прогрессивным моментом, а затем стало сдерживающим прогресс программного обеспечения фактором. Современные версии MS DOS преодолели ограничения на размер оперативного запоминающего устройства (ОЗУ), имеют множество новых команд, содержат встроенные драйверы устройств, графическую оболочку, справочную систему и т.д.

Основные структурные компоненты MS DOS таковы:

- базовая система ввода-вывода (BIOS);
- системный загрузчик (SB);
- драйверы устройств (т.е. программы, поддерживающие их работу);
- базовый модуль;
- командный процессор (называемый также интерпретатором команд);
- утилиты DOS (вспомогательные программы).

Охарактеризуем коротко основные компоненты. BIOS хранится в ПЗУ. Эта программа написана непосредственно в машинных кодах; при включении компьютера она автоматически считывается в ОЗУ, запускается на исполнение и производит беглую проверку работоспособности основных устройств компьютера. Затем BIOS производит поиск на дисках программы запуска операционной системы (программы **начальной загрузки**). BIOS имеет также функции поддержки стандартных периферийных устройств, прежде всего дисплея и клавиатуры.

Программа начальной загрузки, найденная BIOS-ом на диске, обращается последовательно к дисководам A, B и т.д. пока не найдет программу SB - **системный загрузчик**. Эта программа проверяет наличие на диске ядра операционной системы, состоящего из файлов с названиями ibmio.sys - файла расширения BIOS и command.com - командного процессора, загружает их в ОЗУ и запускает на исполнение первую из этих программ. Она дополнительно тестирует оборудование, осуществляет конфигурирование DOS (стандартное при отсутствии файла config.sys - файла конфигурации или нестандартное в соответствии с содержанием файла config.sys), подключает необходимые драйверы и т.д. Далее эта программа устанавливает некоторые указания о способах обработки прерываний (векторы прерываний) и передает управление базовому модулю DOS, который продолжает устанавливать правила обработки прерываний и после этого загружает в ОЗУ командный процессор и передает ему управление.

Пользователь, работающий с DOS без программ - оболочек или дополнительных интерфейсных систем, непосредственно общается с командным процессором. Режим работы - диалоговый, т.е. пользователь отдает команду, ОС выполняет и ждет следующей команды. Способ отдавать команды является достаточно архаичным - текст команды нужно просто набрать на клавиатуре, для чего большую часть команд надо помнить, а для редко встречающихся - пользоваться справочником (либо в виде книги, либо встроенным в DOS).

**Командный процессор,** будучи запущенным, вначале отыскивает и исполняет **программу автозапуска** (файл autoexec.bat), если она есть. Эта программа создается пользователем из команд DOS для того, чтобы произвести некоторые рутинные действия по созданию удобной для начала работы обстановки. Например, если при запуске компьютера вы получаете на экране панели Norton Commander, то лишь потому, что «автозапуск» этой программы предусмотрен тем, кто состав-

лял файл autoexec.bat. Следующее действие командного процессора - выдача на экран приглашения пользователю вводить команду, выглядящее, например, так: C> (если DOS загружалась с диска C).

В ходе работы прикладных програм в ОЗУ постоянно находится лишь малая часть DOS (называемая **резидентной).** Все остальные модули DOS подгружаются лишь по мере потребности в них и удаляются из ОЗУ после отработки.

Файловая система MS DOS поддерживает дисководы, обозначаемые латинской буквой и двоеточием, например:

a:, b:, c:,

иерархическую систему каталогов, заимствованную у системы UNIX, файлы с именами до восьми символов и расширением до трех.

**Общие команды MS DOS**

Общие команды распознаются и выполняются командным процессором command.com. Команды вводятся с клавиатуры, их ввод завершается нажатием клавиши <ВВОД> (<ENTER>).

Общие команды DOS делятся на группы:

- команды работы с дисками;
- команды работы с файлами;
- команды работы с каталогами;
- команды управления системой.

Типовая структура команды выглядит следующим образом:

<имя команды> [<список параметров>] [<список ключей>]

Параметры (аргументы) указывают на те объекты, над которыми совершаются операции, ключи уточняют действие команды. Признак ключа (переключателя) -наличие косой линии ' /'. Квадратные скобки указывают на возможность отсутствия фрагмента.

**DIR**

Команда работы с каталогами; выводит на экран список директорией и файлов, находящихся внутри текущего директория. Если использовать команду DIR без параметров и переключателей, она выводит имена файлов (директорией), их расширения, размеры (в байтах), дату и время создания, их число, общий размер и размер свободного дискового пространства.

Полный синтаксис таков:

DIR [диск:] [путь] [имя_файла] |/ P] //W] [/A[[:]атрибуты]] [/0[[:]порядок_сортировки]] [...]

Параметры

[диск:][путь] указывают дисковод и каталог, оглавление которого нужно просмотреть;

[имя_фаила] указывают файл или группу файлов, список которых необходимо получить.

В имени файла могут быть использованы символы-заместители:

? заменяет один произвольный символ в имени файла;

* заменяет произвольное число произвольных символов.

Например:

DIR *.txt просмотр списка всех файлов с расширением txt;

DIR a?.* просмотр списка файлов с именами из двух знаков, первый из которых буква a, и произвольными расширениями.

Ключи:

/P выводит информацию, пока экран не заполнится, для получения следующих экранов надо нажимать любую клавишу;

/W выводит информацию в сокращенном виде, только имена файлов и директориев (в 5 столбцов);

/A[[:] атрибуты] выводит информацию тех директориев и файлов, атрибуты которых указаны.

Вот некоторые атрибуты:

H . скрытые файлы;

- H все файлы, кроме скрытых;

S системные файлы;

- S все файлы, кроме системных;

D директории;

-D только файлы;

R файлы только для чтения.

Параметр

/O[[:] порядок_сортировки]

управляет порядком сортировки файлов в выдаваемом на экран списке. Без этого параметра имена файлов и директорией выдаются в алфавитном порядке. Задавая его соответствующим образом, можно организовать вывод файлов и директориев в порядке, обратном алфавитному, в алфавитном или обратном порядке по именам расширений, в порядке возрастания или убывания даты и времени последнего изменения содержимого файла или директория, в порядке возрастания или убывание их размеров.

Еще несколько команд той же группы (только имена):

MKDIR (MO) создание нового директория;

CHDIR (CD) переход в другой директорий.

**DEL (ERASE)**

Команда работы с файлами; удаляет файлы.

Синтаксис:

DEL [диск:] [путь] <имя_файла> [/P]

Параметр

[диск:] [путь] <имя_файла>

указывает местонахождение и имя удаляемого файла или группы файлов, если в имени используются символы-заместители.

Ключ /P вызывает запрос подтверждения для каждого удаляемого файла.

**COPY**

Команда работы с файлами; копирует один или более файлов в указанное место, а также может использоваться для слияния файлов. Синтаксис:

COPY [/Y|/-Y] [/AI/B] <файл-источник> [/A|/B] [+ файл-источник [/A|/B] [+ ...]] [файл-результат [/A|/B]] *[/V]*

Параметры состоят из обозначения дисковода, директория и имени файла.

<файл-источник> указывает местоположение и имя файла, содержимое которого необходимо копировать.

<файл-результат> указывает местоположение и имя файла, в который нужно поместить скопированную информацию.

Ключи:

/Y указывает, что команда не должна запрашивать подтверждения при замене существующих файлов;

/V проверка того, что новые файлы записаны правильно.

Еще команда той же группы:

RENAME (REN) - переименование файла или группы файлов;

Примерами команд управления системой служат (приводятся только имена):

COMMAND - запуск командного процессора;

EXIT - выход из командного процессора.

**Дополнительные команды-утилиты**

Помимо команд, распознаваемых и выполняемых командным процессором, в операционной системе имеется большое число утилит - команд, реализованных в виде отдельных программ. В качестве примера рассмотрим утилиту форматирования магнитных дисков.

**FORMAT** - форматирует диск для использования в MS DOS.

Утилита FORMAT создает пустой директорий и таблицы FAT на диске, а также проверяет наличие испорченных областей на диске. Может уничтожить все данные на диске.

Синтаксис:

FORMAT диск: [/V[:метка]] [/Q] [/U] [/F:размер][/B|/S] [/C]
FORMAT диск: [/V[:метка]] [/Q] [/U] [/Тдорожек\N:секторов] [/B|/S] [/C]
FORMAT диск: [/V[:метка]] [/Q] [/U] [/1][/4] [/B|/S] [/C]
FORMAT диск: [/Q] [/U] [/1] [/4] [/8] [/B|/S] [/C]

Параметр

диск: обозначает форматируемый диск (это единственный обязательный параметр утилиты).

Ключи

/V:метка указывает метку диска, используется редко;

/Q указывает, что производится «быстрое» форматирование, т.е. проверку испорченных областей проводить не надо;

/U указывает, что «восстанавливать» информацию до форматирования не потребуется;

/F:размер указывает емкость дискеты;

/S копирование на дискету файлов операционной системы IO.SYS, MSDOS.SYS и COMMAND.COM, что делает ее загрузочной;

/T:дорожек указывает число дорожек на дискете;

/N:секторов задает число секторов на дискете.

**DISKCOPY**

Команда работы с дисками (гибкими); копирует содержимое флоппи-диска в одном дисководе на диск в другом. Ее синтаксис таков

DISKCOPY [dl:][d2:][/l]

Здесь первые два объекта в квадратных скобках - параметры, третий - ключ.

*Примеры.*

DISKCOPY A: B: скопировать дискету в дисководе А на дискету в дисководе В;

DISKCOPY A: скопировать дискету в дисководе А на дискету в текущем дисководе;

DISKCOPY A: B: /I скопировать только первую сторону дискеты.

Еще несколько команд той же группы (только имена; параметры и ключи можно найти в справочниках):

DISKCOMP - сравнение содержимого двух дискет (с целью определить, совпадает ли оно);

CHKDSK - проверка целостности файловой структуры на диске, коррекция ее ошибок;

RECOVER - восстановление (насколько возможно) информации на дефектном диске.

Большое количество утилит MS DOS описано в руководстве по этой системе. Важное значение имеют также драйверы, особенно расширенной оперативной памяти, входящие в состав ОС и позволяющие использовать более 640 кбайт памяти.

Особую роль в системе играют файлы CONFIG.SYS и AUTOEXEC.BAT, читаемые при загрузке системы и задающие ее конфигурацию, загружаемые в память драйверы и резидентные программы, а также дополнительные команды, выполняемые при загрузке системы.

CONFIG.SYS

Выполняется до загрузки командного процессора и содержит вызовы SYS-драйверов. Загружаемые драйверы устанавливаются командой DEVICE, после которой указывается полное имя файла, содержащего драйвер. Например, для подключения драйвера мыши MOUSE.SYS можно задать команду:

DEVICE=C:\DOS\MOUSE.SYS .

Начиная с версии MS DOS 4.0 предусматривается загрузка COM и EXE-драйверов с помощью команды INSTALL. Например,

INSTALL=C:\DOS\MOUSE.COM.

Для эффективной работы с различными типами микропроцессоров компьютера (80286, 80386, 80486, Pentium) и размеров оперативной памяти используют специальные драйверы:

DEVISE=C:\DOS\HIMEM.SYS
DEVISE=C:\DOS\EMM386.EXE NOEMS
DEVISE=C:\DOS\EMM486.EXE.

Кроме загрузки внешних драйверов, CONFIG.SYS загружает свои (внутренние) команды.

Если на компьютере отсутствует кэш жесткого диска (т.е. буферная область ОЗУ, где сохраняется содержание блоков диска), то для ускорения работы с диском вродят команду BUFFERS. Буфер - это часть оперативной памяти размером 532 байт.

Пример:

BUFFERS=20.

С помощью команды FILES можно указать число файлов, которые могут быть одновременно использованы системой и программами.

Команда DOS дает возможность загружать модули операционной системы и некоторые драйверы в область памяти выше 640 кбайт, тем самым увеличивая размер свободной базовой памяти, что важно для ряда прикладных программ.

Ниже приведены примеры типичных файлов конфигураций:

1. для PC 286

```
REM Типичный CONFIG.SYS
DOS=HIGH
REM По возможности загружать модули операционной системы и
REM некоторые драйверы в HMA
```

```
REM (первые 64 кбайта области памяти выше 1 Мбайта)
FILES=20
REM До 20 файлов может быть одновременно открыто
BUFFERS=5
REM Для работы с файлами на жестком диске использовать 5 буферов
DEVICE=C:\DOS\HIMEM.SYS
REM Загрузка менеджера области памяти выше 1 Мбайта
DEVICE=C:\WINDOWS\MOUSE.SYS
REM Загрузка драйвера манипулятора типа «мышь»
DEVICE=C:\STACKER\STACHIGH.SYS
REM Загрузка драйвера поддержки работы с дисками,
REM использующими компрессию данных
```

2. для PC 386

```
REM Типичный CONFIG.SYS
DOS=HIGH, UMB
REM По возможности загружать модули операционной системы и
REM некоторые драйверы в HMA
REM (первые 64 Кбайта области памяти выше 1 Мбайта)
REM и UMB (блоки в области памяти между 640 Кб и 1 Мб)
FILES=20
REM До 20 файлов может быть одновременно открыто
BUFFERS=5
REM Для работы с файлами на жестком диске использовать 5 буферов
DEVICE=C:\DOS\HIMEM.SYS
REM Загрузка менеджера области памяти выше 1 Мбайта
DEVICE=C:\DOS\EMM386.EXE NOEMS
REM Загрузка менеджера расширенной памяти
REM с включенной поддержкой работы с UMB
DEVICEHIGH=C:\WINDOWS\MOUSE.SYS
REM Загрузка драйвера манипулятора типа «мышь»
REM в UMB
DEVICE=C:\STACKER\STACHIGH.SYS
REM Загрузка драйвера поддержки работы с дисками,
REM использующими компрессию данных
```

Не менее важную роль при начальной загрузке компьютера играет файл AUTOEXEC.BAT. Фактически в AUTOEXEC.BAT записаны команды, которые иначе пришлось бы вводить вручную в начале каждого сеанса работы. Например:

```
C:\KEYRUS
REM запуск программы, руссифицирующей ввод с клавиатуры и вывод на экран
C:\DOS\MOUSE
REM запуск драйвера манипулятора типа «мышь»
PROMPT $P$G
REM установка вида приглашения командной строки DOS
PATH C:\;C:\DOS;C:\NC;C:\TOOLS
REM установка путей поиска файлов программ, вызываемых на выполнение
SET TEMP=C:\DOS
REM установка значения переменной окружения, указывающей путь
REM к временным файлам
NC
REM запуск файл-менеджера NORTON COMMANDER
```

## 1.4. ОБОЛОЧКИ ОПЕРАЦИОННЫХ СИСТЕМ

Интерфейс операционной системы DOS не обладает необходимой дружественностью. Команды нужно знать наизусть, посимвольно набирать на клавиатуре и при этом не допускать ошибок. Все это предъявляет высокие требования к квалификации пользователя.

Для облегчения взаимодействия пользователя с компьютером .существуют, так называемые, оболочки операционных систем - программы, делающие наглядным и простым выполнение

базовых операций над файлами, каталогами и др. с использованием меню, защитой от необдуманных и ошибочных действий и разветвленной контекстной помощью. .

Простая оболочка обычно входит в комплект утилит операционной системы MS DOS. Однако, значительно большее распространение получила оболочка под названием «нортон командер» (Norton Commander). Остановимся на ее интерфейсе подробнее, рис. 2.3.

При работе Norton Commander в стандартной настройке (конфигурации) на экране дисплея имеются следующие области:

• правая и левая панель - большие синие прямоугольники, на которых отображаются каталоги (директории) дисков; одна из панелей является активной; обозначение текущего диска и директория вверху активной панели выделены цветом; внутри панели находится указатель;

• командная строка с приглашением, в которой можно непосредственно набирать команды DOS или формировать их с помощью имен файлов на панелях;

• строка подсказки с обозначением команд, закрепленных за функциональными клавишами F1... F10.

Указатель перемещается внутри панели при нажатии клавиш управления курсором; переключить активную панель можно с помощью клавиши <Tab>. Чтобы войти в директорий, нужно указать его имя указателем и нажать клавишу <Ввод> (<Enter>), чтобы выйти из текущего директория - установить указатель на две точки вверху панели выше имен файлов и директориев и нажать клавишу <Ввод> (<Enter>). При нажатии клавиши <Ввод>, когда указатель установлен на имени исполняемого файла (с расширением .bat или .exe или .com), происходит запуск этого файла на исполнение. Если необходимо вызвать имя файла в командную строку для формирования параметров команды, нужно, выделив имя файла указателем, нажать одновременно <Ctrl> и <Enter>. Можно по желанию определить действия Norton Commander над файлами с произвольными расширениями при нажатии клавиши <Ввод>. Так, можно задать, что при указании файла с расширением .txt запускается текстовый редактор и в него загружается указанный файл.

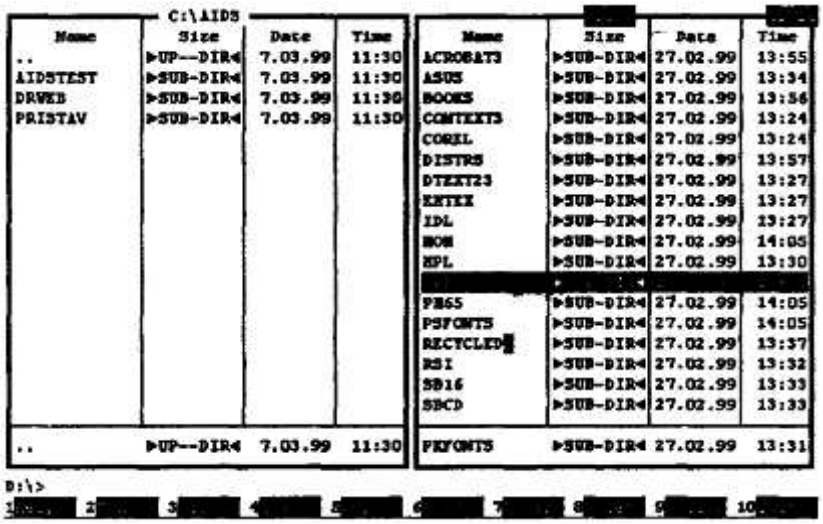



*Рис. 2.3.* Типичный вид панелей Norton Commander

Основные операции выполняются при нажатии функциональных клавиш. Так, при нажатии клавиши

F1 происходит вызов контекстной подсказки (гипертекста с развитой системой ссылок);

F2 вызывается пользовательское меню, в котором по желанию собраны часто выполняемые команды;

F3 вызывается для просмотра (View) в режиме текста или кодов файл, указанный на активной панели;

F4 вызывается простейший редактор для редактирования указанного на активной панели текстового файла;

F5 копируется указанный файл или директорий (или группа отмеченных файлов и директорией) с активной панели на диск, отражаемый пассивной панелью;

F6 происходит переименование (если вводится новое имя) или перемещение файлов или директорий с активной панели на пассивную;

F7 создается директорий на активной панели;

F8 удаляются указанные или отмеченные файлы и директории;

F9 вызывается горизонтальное меню в верхней строке экрана;

F10 происходит выход из Norton Commander.

Выделение файлов происходит при нажатии клавиши <Ins> или при задании фильтра с помощью серой клавиши «+».

Горизонтальное меню, вызываемое клавишей F9, позволяет изменить настройки Norton Commander (например, режим отображения информации на панелях), выполнить перечисленные и некоторые дополнительные, важные для пользователя, команды и действия.

Примечательно, что большинство операций можно выполнить с помощью Norton Commander несколькими способами:

• с помощью меню, выбирая команду с помощью клавиш управления курсором;

• с помощью меню, выбирая команду с помощью манипулятора «мышь»;

• с помощью меню, выбирая команду с помощью выделенной в команде буквы;

• с помощью «(горячих клавиш» - сочетания клавиш при их одновременном нажатии.

По мере роста квалификации пользователи переходят к использованию именно «горячих клавиш». Например, чтобы перейти к другому дисководу на левой панели, нажимают комбинацию <Alt>+<Fl>, на. правой - <Alt>+<F2>; чтобы отключить панели - <Ctrl>+<0> и т.д.

Norton Commander имеет-резидентную часть, которая восстанавливает состояние Norton Commander после окончания работы прикладных программ.

Наряду с Norton Gommander;' нашли применение аналогичные программные средства типа Volkov Commander и DOS Navigator. Имея много общего с Norton Commander, они оказываются более удобными в ряде специфических моментов (таких как работа с архивами, подключение внешних редакторов, соединение файлов и т.д.).

С переходом на персональные компьютеры с процессором 80386 и с увеличенной памятью (не менее 4 Мбайт), на смену Norton Commander и характерному для DOS стилю работы в текстовом режиме пришла оболочка Windows и новый стиль работы с графическим интерфейсом. Идея графического интерфейса Windows заимствована компанией «Microsoft», долгое время специализировавшейся на операционных системах для персональных компьютеров IBM, у операционной системы для компьютеров Apple. Иногда Windows определяют не как оболочку, а как нечто большее, используя термин типа «операционная среда». При этом исходят из того, что если классическая оболочка (такая как Norton Commander) видоизменяет лишь пользовательский интерфейс, то программа типа Windows дополнительно к этому берет на себя управление программами и заданиями, т.е. реализует основные функции операционной системы.

Принципиально важные особенности Windows по сравнению с MS DOS - это многозадачность (допускается одновременное выполнение нескольких процессов) и возможность обмена данными между работающими программами. Важно и то, что Windows использует расширенную оперативную память (много больше 640 кбайт) и подразумевает единый интерфейс всех прикладных программ. Недаром утвердилось понятие «программировать под Windows», т.е. ориентироваться на стандартный графический интерфейс.

Название Windows - «окна» - говорит само за себя. Эта оболочка операционной системы построена на основе графических окон, соответствующих программным средствам и группам программных средств, которыми пользователь может управлять, изменять их размеры, перемещать по экрану, открывать и закрывать по своему желанию.

Оболочка Windows ориентирована на работу с помощью манипулятора «мышь». Все операции в этой среде в высокой степени унифицированы, все программные средства имеют очень схожие интерфейсы и принципы управления, что значительно ускоряет освоение новых программных средств,

Первой версией обсуждаемой операционной среды, получившей широкое распространение, стала Windows 3.1 (Windows for work groups 3.11), рис. 2.4.

При запуске Windows 3.1 пользователь видит на экране окно Program Manager -Диспетчера Программ, в котором имеются пиктограммы (условные схематические обозначения) программ; среди них всегда имеются группы аксессуаров (accessories) и приложений (main), а также другие группы, такие как Microsoft Office. Группу можно активизировать, указав ее пиктограмму курсором и дважды нажав левую кнопку мыши; при этом открывается окно, содержащее пиктограммы

*для* каждого программного средства, относящегося к группе. Чтобы запустить программное средство, надо указать его курсором и дважды нажать левую клавишу мыши.


*Рис.2.4.* Типичный вид среды Windows 3.1

Работающую программу можно снять, указав знак «-» в левом верхнем углу окна и нажав левую кнопку мыши, или отложить (минимизировать), указав знак «-» в правом верхнем углу. Может быть минимизирована и группа, если нажать левую клавишу мыши, предварительно указав на «-» в правом верхнем углу окна группы.

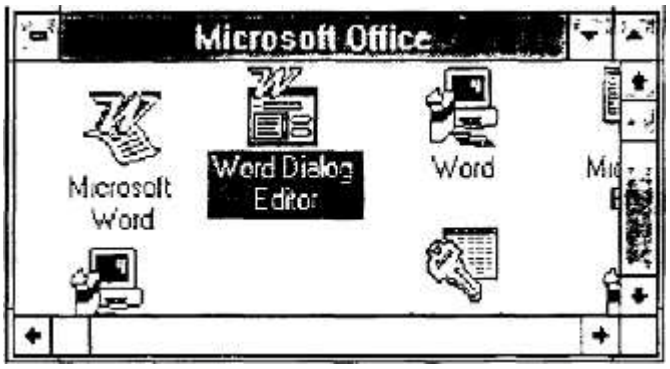



*Рис. 2.5.* Пример группового окна

Существует группа программ, составляющих стандартные приложения Windows. Это относительно небольшие по возможностям (по сравнению со специализированными) прикладные программы: текстовый редактор Write, графический редактор PaintBrush, *картотека, календарь, калькулятор, часы* и т.д. Конечно, возможности редактора Write при оформлении книги не идут в сравнение с текстовым процессором Word-7, но она существенно проще в освоении. В *Картотеке* можно хранить «карточки» с текстами и рисунками и вести поиск информации (например, по ключевым словам), т.е. организовать простую базу данных. *Часы, Календарь* и *Калькулятор* удобно всегда иметь под рукой.

Совместная работа нескольких программ требует возможности обмена данными между ними. При разработке программного обеспечения для MS DOS об этом особо не заботились: даже если разные программы могут обрабатывать один и тот же файл (например, редакторы Лексикон и MultiEdit могут поочередно искать орфографические ошибки в смешанном русско-английском тексте), то надо вначале выйти из одной программы и затем войти в другую, имеющую, как правило, иначе организованный интерфейс, и т.д. В Windows такой проблемы нет: можно, например, ввести в текст, создаваемый с помощью Write или Word, рисунок, созданный с помощью PaintBrush, не выходя ни из одной из этих программ в едином интерфейсе Windows.

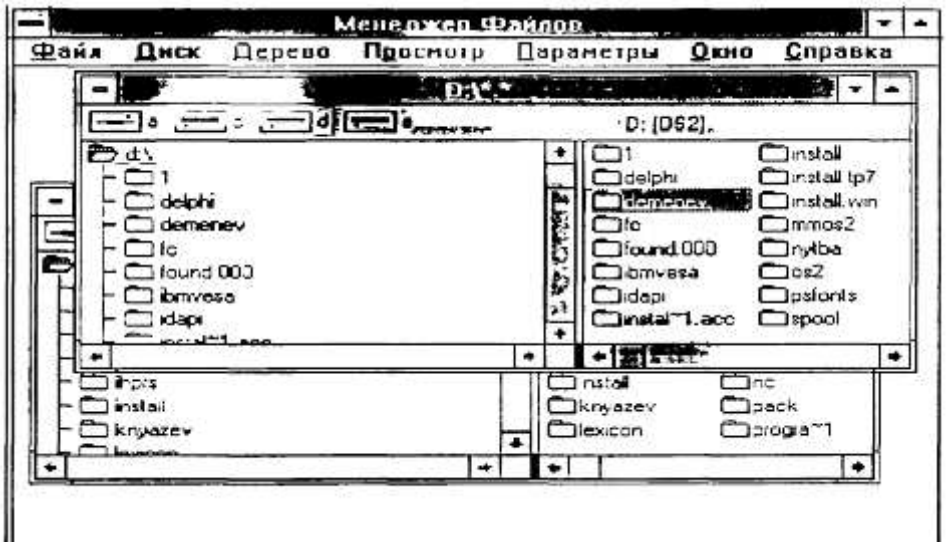



*Рис. 2.6.* Пример окна диспетчера файлов

Для работы с файлами в Windows существует специальная программа - File Manager (Диспетчер Файлов - Менеджер Файлов), рис. 2.6. Она позволяет выполнять все виды работ с файлами: просмотр файловой структуры, копирование, удаление, переименование, пересылку между каталогами и т.д. Пиктограмма Диспетчера Файлов имеет вид шкафа с ящиками, каталога - папки, текущего каталога - раскрытой папки. Благодаря наглядности и простоте использования оболочка Windows получила огромное распространение и стала стандартом для IBM совместимых персональных компьютеров с процессорами 386 и более мощных.

Важно понимать, однако, что многозадачность Windows не есть реальная параллельность в выполнении программ. На машине с одним процессором, которой является IBM PC, истинная параллельность невозможна. Среда Windows создает видимую параллельность, разделяя время между приложениями малыми порциями и постоянно переходя от одного к другому.

С 1995 года на смену операционной оболочке Windows 3.1 для IBM-совместимых персональных компьютеров пришла операционная система Windows'95. В настоящее время эта операционная система бьет все рекорды популярности. По прогнозам ожидается около 100 млн. продаж этой системы к 2000 году. Трудно оценить количество несанкционированных копий этой системы, широко используемой как в России, так и во всем мире. Windows'95 предназначена для установки на персональных компьютерах desktop и компьютерах типа notebook, имеющих процессор типа

Intel 80386DX, 80486, Pentium. Объем оперативной памяти должен быть не менее 4 Мбайт. Новая версия операционной системы лучше использует дополнительную оперативную память, чем предыдущая. Однако многие приложения для нормальной работы требуют от 8 до 16 Мбайт.

Новая версия обеспечивает более высокую производительность и большие возможности за счет применения 32-, а не 16-разрядной передачи данных. Windows'95 является высокопроизводительной, универсальной, надежной, многозадачной и многопотоковой интегрированной 32-разрядной операционной системой нового поколения с расширенными сетевыми возможностями, работающей в защищенном режиме и обеспечивающей графический интерфейс с пользователем. Windows'95 представляет собой интегрированную среду, обеспечивающую эффективный обмен информацией между отдельными программами и предоставляющую пользователю широкие возможности по обработке текстовой, графической, звуковой и видеоинформации. Понятие интегрированности подразумевает также совместное использование ресурсов компьютера всеми программами.

Операционная система позволяет прослушивать компакт-диски, редактировать музыкальные произведения, работать с видеофрагментами. Новая операционная система обладает также улучшенными телекоммуникационными возможностями, которые позволяют использовать ее в локальных и глобальных сетях, работать с электронной почтой. Windows'95 обеспечивает работу пользователя в сети, с электронной почтой, факсом и со средствами мультимедиа, поддерживает большинство приложений DOS и предыдущих версий Windows. Если в предыдущих версиях активное приложение периодически передавало системные ресурсы другим приложениям, работающим в фоновом режиме, то новая операционная система сама управляет ресурсами, используя принцип вытесняющей многозадачности. Приложение, нуждающееся в ресурсах, может приоста-

новить работу до получения ресурса или перейти к другим операциям. Многопоточное выполнение одной задачи позволяет при задержке в выполнении одного потока работать со следующим. Распределение времени между потоками производится с учетом их приоритетов. Приложения работают в защищенном адресном пространстве. После завершения работы приложения память автоматически очищается.

Применение 32-разрядного фонового спулинга печати ускоряет печать больших документов, позволяет минимизировать перерывы в работе. Сопоставление возможностей Windows'95 и Windows for Workgroups 3.11 показало, что при использовании новой операционной системы скорость загрузки и печати 100-страничного WinWord-документа возросла примерно на 30%.

После загрузки Windows'95 на экране появляется изображение, напоминающее рабочий стол. Так же, как на рабочем столе, на его модели (на экране) размещены значки папок с документами и значки быстрого доступа. Одна из основных задач, поставленная разработчиками новой операционной системы, заключалась в обеспечении простоты, удобства, интуитивной очевидности пользовательского интерфейса. Интерфейс Windows'95 спроектирован так, чтобы создать комфортные условия для пользователя и обеспечить объектно-ориентированную и документированную работу. По сравнению с предыдущими версиями улучшилось оформление экрана: появились эффектные трехмерные интерфейсные элементы, изменилось оформление диалогового окна, окон приложений и документов. Улучшенный пользовательский интерфейс, однотипность выполнения всех базовых операций призваны ускорить процесс освоения операционной системы.

Для обозначения различных объектов, с которыми имеет дело пользователь, в Windows используются графические символы. В Windows'95 отдельные файлы объединяют в папки, которые являются аналогом каталогов, использовавшихся в предыдущих версиях. Так же, как в каталоге может находиться несколько каталогов более низкого уровня, папка может состоять из нескольких папок более низкого уровня.

Значительное внимание уделено документо-ориентированной работе с тем, чтобы пользователь в первую очередь уделял внимание документам, а не прикладным программам (документом называется любой файл, обрабатываемый с помощью прикладной программы). Windows'95 позволяет открыть любой документ, не запуская предварительно приложение, в котором оно создано. Объекты (предметы), с которыми мы контактируем в реальной жизни, обладают определенными свойствами. У каждого предмета свой внешний вид, вес, габариты и т.п. Аналогично, объекты Windows имеют свои характеристики. Можно подобрать внешний вид значка, отображающего файл. Файлы имеют размеры, для них задаются атрибуты и т. п.

Windows'95 предоставляет удобные средства быстрого вызова программ, документов и папок с помощью значков быстрого вызова, позволяющих двойным щелчком открыть папку или документ, не запуская предварительно приложение, в котором создавался объект. Чтобы ускорить открытие часто используемых документов и запуск приложений, можно создать к ним сколько угодно значков быстрого вызова и разместить их на рабочем столе в одной или нескольких папках. Например, можно обеспечить быстрый вызов принтера, установив его значок на рабочей поверхности стола. В этом случае, чтобы распечатать файл будет достаточно перетащить мышью его значок на значок принтера. Двойной щелчок значка быстрого вызова *Блокнот* на экране дисплея запустит текстовый процессор.

Чтобы установить значок быстрого вызова к папке (файлу), ее необходимо выделить в окне *Мой компьютер* и выбрать команду *Создание ярлыка* из меню *Файл.* Первоначально значок располагается в конце списка окна. Значок можно переместить или скопировать на рабочий стол или в часто используемую папку с помощью мыши методом Drag and Drop (Перетащить и Отпустить). Другой вариант установки значка быстрого вызова в нужной папке или на рабочем столе - перетащить значок программы или документа в нужную папку, нажав правую кнопку мыши, и воспользоваться командой *Создать ярлык* из динамического меню, которое появится, когда отпустят правую кнопку. Можно не копировать файл (папку) в другую папку, а вставить в нее значок быстрого вызова к этому файлу. Сначала следует выделить файл, затем активизировать команду *Копировать* из меню *Правка.* После перехода в окно, где предполагается разместить значок быстрого вызова, активизируют команду *Вставить ярлык* из меню *Правка.* Изображение значка быстрого .вызова можно изменить с помощью диалогового окна, появляющегося после выделения значка и активизации команды *Свойства* из меню *Файл.* Все значки быстрого вызова связаны с файлами и папками, которые они представляют. При удалении файлов и папок автоматически удаляются и

значки. При удалении значка быстрого вызова файл, с которым он связан, не удаляется. Если переименовать папку или файл, надпись к значку не меняется, однако, связь между ними сохраняется.

Длина имени файла в предыдущих версиях Windows, как и в DOS, была ограничена восемью символами, а расширение - тремя символами после точки. Это вызывало неудобства при вспоминании содержания файла. В Windows'95 эти ограничения сняты. Windows'95 позволяет давать файлам имена, содержащие до 255 символов и включать пробелы, знак плюс, знак равенства, квадратные скобки, точку с запятой и другие знаки препинания. Пробелы, находящиеся в начале и в конце имени, не учитываются. Имя файла можно писать на русском языке. Любые символы, стоящие после последней точки, рассматриваются как расширение, расширение имени зависит от приложения, в котором создавался файл. Имя для папки задается так же, как для файла. Однако для папки не задается расширение.

Чтобы установить атрибуты файла и определить его принадлежность к приложению или документу, используются первые три символа после последней точки в имени файла. Так, для файла CONFIG.SYSTEM устанавливается расширение SYS. Совместимость имен файлов новой операционной системы с предыдущими версиями и DOS обеспечивается поддержкой ранее используемой таблицы размещения файлов (FAT), в которой хранится информация о свободных секторах и о дисковом пространстве, отведенном для файлов. Существенные ограничения таблицы FAT связаны с тем, что она не предназначена для работы с большими дисками объемом > 100 Мбайт. При работе с такими дисками FAT не помещается целиком в памяти и загружается частями, что увеличивает количество перемещений головок при считывании файла. Кроме того, использование FAT приводит к сильной фрагментации больших дисков, что увеличивает потери времени, связанные с их обработкой. Аналогичная таблица для Windows'95 совместима с FAT и поэтому при установке Windows'95 не требуется переформатирование жесткого диска. При использовании длинного имени, созданного Windows'95, в DOS FAT создаются «псевдонимы», которые обеспечивают ввод первых восьми букв из нового имени и добавляют порядковый номер после знака тильда "~". Например, при использовании слова literature (литература) в именах двух файлов в DOS они запишутся как litera~1 для первого документа и litera~2 для второго документа. Инсталлируемая файловая система Windows'95 поддерживает также сетевые файловые системы.

Существенно усовершенствованы в Windows'95 окна. Групповые окна заменены на окна папок, в которых появились очень полезные для работы панели инструментов. Их можно отобразить в окне папки или удалить командой *Панель инструментов* (рис. 2.7.) из меню *Вид.* Панель содержит раскрывающееся окно списка, в котором представлено имя текущей папки, и кнопки, дублирующие часто используемые команды. После раскрытия окна списка в нем видна древовидная диаграмма папок, имеющихся на компьютере. Команду можно быстро активизировать, щелкнув кнопку на панели инструментов. При этом нет необходимости сначала выбирать меню. При подводе указателя мыши к кнопке рядом с ним появляется флажок-подсказка с названием выполняемой команды.

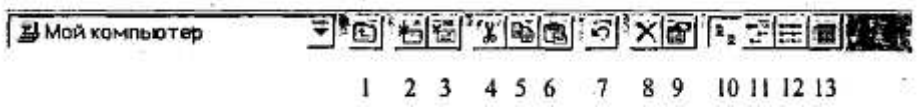



*Рис. 2.7.* Панель инструментов окна папки

Ниже кратко описывается назначение кнопок:

1 - вверх на один уровень (Up one level) - переход в родительскую папку;

2- подключить сетевой диск (доступно при работе в сети);

3 - отключить сетевой диск (доступно при работе в сети);

4 - вырезать (Cut) - перемещение выделенного объекта в Буфер Обмена;

5 - копировать (Copy) - копирование выделенного объекта в Буфер Обмена;

6 - вставить (Paste) - вставка выделенного объекта из Буфера Обмена;

7 - отменить (Undo) - отмена последней команды, восстанавливает вид окна до выполнения предыдущей команды;

8 - удалить (Delete) - уничтожение выделенного файла или папки;

9 - свойства (Properties) - вызов одноименного диалогового окна;

10 - большие значки (Large Icons) - отображение рядом с именами папок и файлов больших

значков;

11 - маленькие значки (Small Icons) - отображение рядом малых значков;

12 - список (List) - отображение сначала папок, затем файлов столбцами так, чтобы надписи к ним следовали в алфавитном порядке;

13 - таблица (Details) - отображение рядом с именами папок и/или файлов их кратких характеристик.

В зависимости от указанного типа файла, Windows'95 позволяет открыть то или иное приложение, использовать определенный набор команд. Для изменения типа файла или значка к нему используют кнопку *Правка.*

При работе с документом следует максимально увеличить область экрана, где может располагаться интересующая нас информация. Однако большую часть экрана часто занимают различные панели. Отображение на экране панели задач регулируется с помощью флажков вкладки *Параметры панели задач* команды *Панель задач* меню *Настройки.* Установка флажка *Автоматически убирать с экрана* позволяет не показывать панель задач и освободить максимум рабочего пространства для работающего приложения.

Windows^95 дает возможность печатать документы на нескольких десятках языков. При начальной установке системы необходимо записать утилиты, позволяющие работать на нужном вам языке. Русифицированная версия содержит кодовую страницу, поддерживающую знакогенератор и раскладку клавиатуры на русском языке. Эта страница обеспечивает корректную работу с именами файлов, содержащими русские буквы не только в Windows, но и в DOS. Язык, на котором будут печататься символы, переключатели, используемые для перехода с одного языка на другой, выбираются в диалоговом окне программы *Клавиатура с* вкладкой *Язык* приложения *Панель управления.* Приложение можно вызвать, щелкнув кнопку *Пуск.* Затем в меню *Настройки* выбирают команду *Панель управления.* Чтобы добавить другой язык для ввода символов, нажимают кнопку *Добавить,* в раскрывающемся окне списка Язык выбирают требуемый и щелкают кнопку ОК.

Для выделения заголовков, смыслового разграничения отдельных фрагментов, при написании формул, индексов используются различные стили и размеры шрифта. Получить справочные данные о шрифте и увидеть его гарнитуру (стиль) позволяет окно, появляющееся после выбора команды *Открыть* из меню *Файл* папки *Шрифты* программы *Мой компьютер.* В окне отражаются название, размер файла, версия, фирма-разработчик, демонстрируются образцы нескольких размеров шрифта. Чтобы удалить используемый шрифт, его надо выделить в окне папки *Шрифты* и выбрать команду *Удалить* из меню *Файл.*

Во многих приложениях гарнитура и размер шрифта задаются в диалоговом окне команды *Шрифт.* Перечень всех установленных шрифтов приводится в окне списка *Шрифт.* Размер в пунктах выбирается в окне *Размер.* Образец написания выбранного шрифта представляется в демонстрационном поле *Образец* в правой части окна. В поле *Цвет* задается цвет символов. Начертание (стиль) шрифта задается в окне *Начертание:* обычный, полужирный, курсив, подчеркнутый.

Меню *Пуск* позволяет выполнить большой набор работ, связанных с запуском приложений, получением справок, поиском и открытием документов, настройкой системы. Видимо поэтому разработчики операционной системы рядом с кнопкой *Пуск* поместили подсказку: «Начните работу с нажатия этой кнопки». Для активизации меню *Пуск* следует щелкнуть кнопку *Пуск* или нажать на клавиши Ctrl+Esc. При остановке указателя мыши на пункте меню со значком треугольника с правой стороны раскрываются окна, содержащие подменю и команды. В свою очередь отдельные пункты появившегося подменю также могут быть отмечены значком треугольника и иметь свои подменю. Каждое подменю содержит группу программ. Для выбора программы необходимо остановить на ней указатель и щелкнуть мышью.

Пункт *Документы* открывает список с названиями последних документов, с которыми работал пользователь. Список может содержать до 15 наименований документов, независимо от приложения, в котором они создавались. Для открытия документа следует щелкнуть на его названии. Следует отметить, что некоторые приложения не добавляют имена файлов в список меню *Документы.* Тогда документ можно открыть, запустив то приложение, в котором он создавался. Как правило, открыть документ позволяет команда *Открыть* из меню *Файл* соответствующего приложения Windows. Кроме того, документ можно открыть двойным щелчком его значка в окне *Мой Компьютер.*

Ряд приложений Windows проверяет, имеются ли в памяти компьютера несохраненные данные. При выходе из приложения без команды *Сохранить* появится предупреждающее сообщение с вопросом о необходимости сохранения последних изменений. При попытке закрыть приложение без указаний как поступить с открытым документом появится запрос: «Сохранить изменения, внесенные в документ?». Три кнопки *-Да, Нет, Отменить* - позволяют сохранить внесенные во время текущего сеанса работы изменения, не вносить изменений или отменить выход из системы.

Наиболее быстрый способ добавить команду/пункт в меню *Пуск* - перетащить мышью значок программы на кнопку этого меню. Новый пункт меню расположится в верхней строке меню. Например, можно создать значок быстрого вызова для программы Norton Commander, разместив сначала на рабочем столе значок программы Norton Commander, а затем перетащив его мышью на кнопку *Пуск.*

Нередко приходится искать нужный файл/папку, так как забыто его имя или место расположения. Если известна папка, где расположен файл, то можно легко найти его по расширению. Сложнее, если не известно название и расширение. Чтобы быстро найти файл или папку на компьютере пользователя или на других компьютерах сети можно использовать команду *Файлы или Папки* из меню *Поиск.* Поиск можно выполнять по следующим критериям:

- по имени файла или папки и по цепочке символов, входящих в имя файла или папки;
- по расширению имени файла;
- по дате последней модификации;
- по размеру;
- по отрывку текста из документа или по заголовку какого-либо раздела.

Использование различных вкладок облегчает поиск файлов по определенным критериям.

Для быстрого вызова часто повторяемых команд можно воспользоваться динамическим меню, которое вызывается правой кнопкой мыши. Динамическое меню (рис. 2.8) содержит часто употребляемые команды. Набор команд зависит от выбранного объекта: значка диска, папки или файла, выделенного текста, панели задач или свободного места экрана.

Если щелкнуть правой кнопкой значок диска в окне программы *Мой компьютер,* то появятся команды: *Открыть, Проводник, Найти, Разделение, Форматировать, Вставить, Создать ярлык. Свойства.* Так же как и для диска, команды динамического меню для файла зависят от типа файла и дублируют меню *Файл.* Правую кнопку мыши удобно использовать для перемещения или копирования файла из одного окна в другое папки *Мой компьютер* или *Проводник.* После транспонирования значка папки/файла с нажатой правой кнопкой мыши появляется динамическое меню, позволяющее указать цель транспортировки: переместить или скопировать объект, создать значок быстрого вызова.

Папка *Мусорная корзина* предназначена для удаления ненужных файлов. Основное различие между выполнением команды *Удалить* в Windows'95 и в других программах состоит в том, что в новой версии операционной системы выбор команды приводит не к удалению файла, а к его перемещению в папку *Мусорная корзина.* Попавший в нее файл сохраняется до тех пор пока Корзина не будет «очищена». Чтобы удалить любой файл, папку или значок быстрого вызова, можно использовать команду *Удалить* или переместить значок удаляемого объекта мышью на значок *Мусорная корзина.* Значок перетаскиваемого объекта исчезнет. Чтобы просмотреть все файлы, находящиеся в Мусорной корзине, необходимо дважды щелкнуть ее значок. Появится окно папки со строкой меню, имеющим стандартный набор команд. Для восстановления файла, папки или значка быстрого вызова необходимо щелкнуть имя восстанавливаемого файла. Если надо восстановить несколько файлов, то имена файлов выделяют при нажатой клавише Ctrl. Затем используют команду *Восстановить* из меню *Файл.*

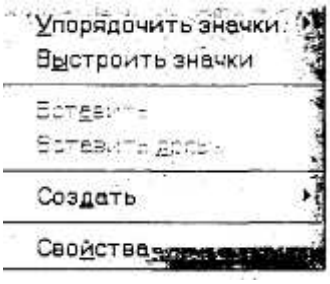


*Рис. 2.8.* Динамическое меню

В настоящее время для IBM PC-совместимых компьютеров выпускаются тысячи наименований винчестеров, адаптеров, контроллеров и других изделий. В ряде случаев их установка на компьютере вызывает значительные трудности с точки зрения совместимости, требует больших затрат времени пользователя для выбора положения перемычек. Компьютеры с Plug and Play - адаптерами не нуждаются в ручной установке DIP-переключателей. В соответствии с технологией Plug and Play после включения компьютера автоматически определяются названия вновь подключенных устройств или плат и их характеристики, выполняется их конфигурирование и автоматически загружаются соответствующие драйверы. Это может происходить и во время сеанса работы в случае изменения аппаратных средств компьютера. Аналогично фиксируется удаление какого-либо устройства или платы, выгружаются драйверы этого устройства с тем, чтобы они не занимали оперативную память и освободили системные ресурсы.

При инсталляции Windows'95 приложение Setup (Установка) определяет адаптеры и драйверы, которые не поддерживают новую технологию, и автоматически делает соответствующие записи в системных файлах. Даже если ваш компьютер не полностью поддерживает стандарт Plug and Play, Windows'95 поможет настроить аппаратное обеспечение с помощью диалогового окна *Свойства с* вкладкой *Ресурсы* рассматриваемого устройства. Для вызова окна *Устройства* сначала следует активизировать значок *Система* Панели Управления и двойным щелчком мыши выбрать устройство. В нижнем поле *Список конфликтующих устройств* указываются устройства, с которыми может конфликтовать рассматриваемое устройство. Рекомендуется установить флажок *Использовать автоматическую настройку,* чтобы операционная система сама производила настройку системных ресурсов.

Технология Plug and Play содержит три основных компонента: операционную систему, поддерживающую Plug and Play, Plug and Play BIOS и Plug and Play - устройства с соответствующими драйверами. Поэтому полное решение проблемы Plug and Play требует поддержки как на программном, так и на аппаратном уровнях.

В ходе работы с Windows'95 иногда приходится производить перезагрузку системы. Различают «холодную» и «горячую» перезагрузку компьютера. «Холодная» перезагрузка выполняется после выключения питания и его повторного включения. Необходимые для работы программы и параметры считываются в оперативную память заново. «Горячая» перезагрузка производится без выключения питания, одновременным нажатием клавиш Alt+Ctri+Del. В этом случае вся информация, хранящаяся в оперативной памяти и не записанная на жесткий диск, стирается. «Горячую» перезагрузку, как правило, используют в том случае, когда программа «зависла» и не реагирует на нажатие на клавиши и на кнопки мыши. В ряде случаев при нажатии на клавиши Alt+Ctri+Del появляется диалоговое окно *Снять Задачу.* Если окно позволит закрыть «зависшую» программу, то необходимость в перезагрузке компьютера отпадет.

Перед тем как выключать питание компьютера, необходимо закрыть все открытые документы и приложения. Выключение питания без закрытия документа может привести к потере данных, повреждению открытых файлов и трудностям с их открытием при последующих сеансах работы. После выключения компьютера без правильного выхода из системы возможны нарушения в логической структуре диска. Их исправить можно с помощью программы ScanDisk, расположенной в группе *Служебные программы.*

Для корректного выхода из Windows надо щелкнуть кнопку *Пуск* и команду *Завершить работу* в появившемся меню. Появится диалоговое окно Завершение Работы. Окно содержит кнопки-переключатели: *Выключить компьютер. Перезагрузить компьютер. Перезагрузить компьютер в режиме эмуляции MS DOS, Войти в систему под другим именем.* Все кнопки закрывают все программы. В нижней части окна Shut Down Windows расположены три кнопки - *Да, Нет, Справка.* Через небольшой промежуток времени после щелчка мышью кнопки *Да* компьютер будет подготовлен к выключению: будут очищены внутренние буферы и кэши дисков, обеспечено сохранение данных. Не следует выключать электропитание до тех пор, пока не появится сообщение: «Теперь питание компьютера можно выключить».

Мы ограничимся лишь приведенным выше кратким описанием принципов работы Windows. Реальное ее освоение (еще в большей мере, чем DOS и Norton Commander) - дело практических занятий за компьютером с помощью как многочисленных специальных руководств, так и встроенного справочника.

***Контрольные вопросы и задания***

1. Охарактеризуйте место операционных систем среди других видов программного обеспечения.
2. Каковы функции операционной системы?
3. Охарактеризуйте основные ступеньки эволюции операционных систем.
4. Каково содержание понятий
   - процесс?
   - ресурс?
   - виртуализация?
   - прерывание?
5. Охарактеризуйте функции основных компонент операционных систем.
6. В чем состоит назначение файловой системы ОС?
7. Что такое файл? Какие структуры файлов поддерживаются различными ОС?
8. Что такое каталог (директорий)? Для чего каталоги служат?
9. Какие операции над файлами обеспечиваются операционными системами?
10. Охарактеризуйте команды операционной системы MS DOS.
11. Опишите интерфейс оболочки ОС Norton Commander.
12. Охарактеризуйте стиль работы с помощью оболочки Windows.
13. По каким показателям Windows'95 превосходит Windows 3.11?
14. Опишите типичное окно Windows'95.
15. Какие функции выполняют меню Пуск? Поиск?
16. Как вызвать динамическое меню и каковы его возможности?

# § 2. ПОНЯТИЕ О СИСТЕМЕ ПРОГРАММИРОВАНИЯ

## 2.1. ОСНОВНЫЕ ФУНКЦИИ И КОМПОНЕНТЫ

Системы программирования - это комплекс инструментальных программных средств, предназначенный для работы с программами на одном из языков программирования. Системы программирования предоставляют сервисные возможности программистам для разработки их собственных компьютерных программ.

В настоящее время разработка любого системного и прикладного программного обеспечения осуществляется с помощью систем программирования, в состав которых входят

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

Системы программирования, как правило, включают в себя

- текстовый редактор (**Edit**), осуществляющий функции записи и редактирования исходного текста программы;
- загрузчик программ (**Load**), позволяющий выбрать из директория нужный текстовый файл программы;
- запускатель программ (**Run**), осуществляющий процесс выполнения программы;
- компилятор **(Compile),** предназначенный для компиляции или интерпретации исходного текста программы в машинный код с диагностикой синтаксических и семантических (логических) ошибок;
- отладчик **(Debug),** выполняющий сервисные функции по отладке и тестированию программы;
- диспетчер файлов (**File**), предоставляющий возможность выполнять операции с файлами:сохранение, поиск, уничтожение и т.п.

Ядро системы программирования составляет язык. Существующие языки программирования можно разделить на две группы: процедурные и непроцедурные, рис. 2.9.

Процедурные (или алгоритмические) программы представляют из себя систему предписаний для решения конкретной задачи. Роль компьютера сводится к механическому выполнению

этих предписаний.

Процедурные языки разделяют на языки низкого и высокого уровня.

Языки низкого уровня (машинно-ориентированные) позволяют создавать программы из машинных кодов, обычно в шестнадцатиричной форме. С ними трудно работать, но созданные с их помощью высококвалифицированным программистом программы занимают меньше места в памяти и работают быстрее. С помощью этих языков удобнее разрабатывать системные программы, драйверы (программы для управления устройствами компьютера), некоторые другие виды программ.


*Рис. 2.9.* Общая классификация языков программирования

Программы на языках высокого уровня близки к естественному (английскому)
языку и представляют набор заданных команд.

Перечислим наиболее известные системы программирования.

1. Фортран (FORmula TRANslating system - система трансляции формул); старейший и по сей день активно используемый в решении задач математической ориентации язык.

2. Бейсик (Beginner's All-purpose Symbolic Instruction Code - универсальный символический код инструкций для начинающих); несмотря на многие недостатки и изобилие плохо совместимых версий - самый популярный по числу пользователей.

3. Алгол (ALGOrithmic Language - алгоритмический язык); сыграл большую роль в теории, но для практического программирования сейчас почти не используется.

4. ПЛ/1 (PL/I Programming Language - язык программирования первый). Многоцелевой язык; сейчас почти не используется.

5. Си (C - «си»); широко используется при создании системного программного обеспечения.

6. Паскаль (Pascal - назван в честь ученого Блеза Паскаля); чрезвычайно популярен как при изучении программирования, так и среди профессионалов. На его базе созданы несколько более мощных языков (Модула, Ада, Дельфи).

7. Кобол (COmmon Business Oriented Language - язык, ориентированный на общий бизнес); в значительной мере вышел из употребления.

8. Дельфи (Delphi) - язык объектно-ориентированного «визуального» программирования; в данный момент чрезвычайно популярен.

9. Джава (Java) - платформенно-независимый язык объектно-ориентированного программирования, чрезвычайно эффективен для создания интерактивных веб-страниц.

Среди непроцедурных языков наиболее известны

1. Лисп (Lisp);

2. Пролог (PROgramming in LOGic);

3. Оккам (назван в честь философа У. Оккама).

Широкое распространение среди разработчиков программ, а также при обучении программированию, получили системы программирования «Турбо» (Turbo) фирмы Borland, ядром которых являются трансляторы с языков программирования Бейсик, Паскаль, Си, Пролог и др. Интерфейс Турбо-оболочки для любых систем программирования внешне совершенно одинаков и предоставляет пользователю стандартный набор функций и команд, описанных выше и отображаемых в главном меню системы.

Рассмотрим технологию разработки программ с использованием популярной системы программирования Турбо-Паскаль 7 (оставив знакомство с самим языком до следующей главы).

В подобных интегрированных системах программирования сделана попытка предоставить разработчику программ максимум сервисных возможностей. Помимо основных функций система Турбо-Паскаль 7 позволяет настроить компилятор на работу в трех режимах: обычном режиме MS

DOS (Real), защищенном режиме (Protected) и в режиме операционной среды Windows (Windows).

После загрузки системы (файл TURBO. EXE), на экране монитора появляется интерфейсное окно, рис. 2.10.

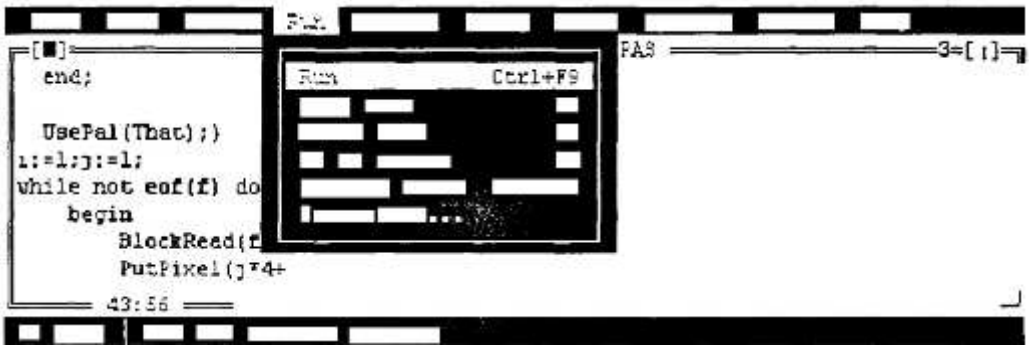



*Рис. 2.10.* Вид экрана интегрированной среды Турбо-Паскаля версии 7 (монтаж)

Главное меню системы (верхняя строка экрана) содержит команды, которые позволяют осуществлять следующие виды работ:

| | |
|---|---|
| File | - работа с файлами (сохранение, загрузка, связь с операционной системой); |
| Edit | - работа с текстовым редактором (после загрузки системы по умолчанию текстовый редактор находится в активном состоянии); |
| Search | - поиск и замена фрагментов текста; |
| Run | -запуск программы на выполнение; |
| Compile | — компиляция программы и установка параметров компиляции; |
| Debug | - установка параметров отладки программы; |
| Tools | - инструментальные программные средства (ненавязчивый сервис); |
| Options | -установка опций интегрированной среды; |
| Window | - работа с окнами; |
| Help | -система помощи и подсказок. |

Для начала работы с системой программирования необходимо иметь проект текста программы, который можно набирать на рабочем поле окна системы. Встроенный текстовый редактор прост и максимально приспособлен для набора текстов программ на языке Паскаль. В нем предусмотрена специальная подсветка управляющих структур, команд. Удобна система контекстной помощи (Shift+Fl), которая вызовет подсказку по набираемому текущему тексту программы в любой момент и в любом месте. Впрочем, текст программы можно приготовить в любом текстовом редакторе, хранящем тексты в ASCII-кодах (например, в Лексиконе); необходимо лишь снабдить имя файла расширением .pas.

Если текст (тексты) программы был ранее сохранен на жестком диске или дискете, то он может быть загружен в поле редактирования с помощью пункта меню **File**.

После окончания формирования текста необходимо откомпилировать программу (пункт меню Compile). Если в программе есть ошибки, то компилятор их укажет. После исправления ошибок можно снова повторить компиляцию.

После удачной компиляции запуск программы осуществляется командой меню **Run.**

Но на этом этапе чаще всего работа не заканчивается. Сложные алгоритмы требуют тестирования и отладки. Многие программы составляются из отдельных модулей, требуют связи с другими программами и системами и т.д. Для решения всех этих проблем предназначены другие команды системы (**Debug**, **Options** и пр.).

Разумеется, программисту, работающему на Паскале, нет нужды самому программировать такие непростые, но часто встречающиеся операции, как вычисление значений математических функций, построение изображений простых геометрических объектов (отрезков прямых, окружностей и т.д.), очистка экрана и множество других. Высокоэффективные, тщательно отлаженные программы таких действий сведены в стандартные модули и надо лишь уметь к ним обратиться. В состав пакета библиотек стандартных модулей входят: **Crt -** работы с экраном, **Graph** -работы с графикой и другие, такие как **Overlay**, **String**, **System**, **Turbo3**, **WinAPI, WinCrt, WinDos, WinPrn, WinTypes, WinProcs.**

## 2.2. ТРАНСЛЯЦИЯ ПРОГРАММ И СОПУТСТВУЮЩИЕ ПРОЦЕССЫ

С появления первых компьютеров программисты серьезно задумывались над проблемой кодирования компьютерных программ. Уже с конца 40-х годов стали появляться первые примитивные языки программирования высокого уровня. В них программист записывал решаемую задачу в виде математических формул, а затем, используя специальную таблицу, переводил символ за символом, преобразовывал эти формулы в двухлитерные коды. В дальнейшем специальная программа (впоследствии названная интерпретатором) превращала эти коды в двоичный машинный код. Первый компилятор был разработан Г. Хоппер в начале 50-х годов; он осуществлял функцию объединения команд и в ходе трансляции производил организацию подпрограмм, выделение памяти компьютера, преобразование команд высокого уровня (в то время псевдокодов) в машинные команды. В дальнейшем компиляторы и интерпретаторы для языков Ассемблера стали развиваться и прочно вошли в практику компьютерного дела.

Идеи трансляции (перекодирования) одних символов в другие легли в основу создания различных языков программирования с соответствующими трансляторами - компиляторами и/или интерпретаторами. Отличие компиляторов от интерпретаторов заключается в процедуре трансляции текста в машинный код. Компилятор преобразует весь текст программы в последовательный набор машинных команд, который в дальнейшем отправляется на выполнение (пример компилятора с языка Паскаль). Интерпретатор же осуществляет трансляцию по принципу синхронного перевода. Каждая отдельная строка программного текста транслируется, а затем, после ее интерпретации, команды этой строки выполняются (пример языка Бейсик). Современные трансляторы с языков программирования высокого уровня, систем управления базами данных интегрируют в себе возможности и достоинства компиляторов и интерпретаторов, а в системы программирования добавляют различные сервисные утилиты по трансляции и отладке создаваемых программ.

Важнейшим элементом в развитии систем программирования выступили подпрограммы. Появление аппарата подпрограмм существенно облегчило процесс разработки системных и прикладных программ. Подпрограммы позволили формировать библиотеки из наиболее часто употребляемых в программах алгоритмов -процедур и функций. В системах программирования обязательно присутствуют стандартные (встроенные в систему) библиотеки подпрограмм. Например, в их число входят подпрограммы вычисления математических функций sin(x), cos(x), abs(x) и др.

В настоящее время распространены пользовательские и прикладные библиотеки подпрограмм. Их число увеличивается. Меняется структура библиотечных подпрограмм. В современных языках получили распространение модули (**Unit**), представляющие специализированные пакеты взаимосвязанных подпрограмм определенного предназначения, например по работе с клавиатурой, с графикой и пр. Развитие объектно-ориентированного программирования позволило создавать библиотеки объектов и подпрограмм с объектными типами данных (**Object**). Примером могут служить оболочки типа **TurboVision**.

Современная программа представляет набор команд, операторов и выражений, в которых имеются ссылки (прямые или косвенные) на различные подпрограммы из существующих в системе программирования библиотек, модулей, объектов. В этой связи исходный текст программы, как правило, занимает по объему места в памяти в несколько раз меньше, чем его оттранслированный вариант в машинных кодах. Как это происходит?

Рассмотрим один из вариантов трансляции программы с языка программирования Паскаль. Исходный текст программы решения квадратного уравнения представлен ниже**:**

```
program KvadUravn;
        var A, B, C, D, XI, X2: REAL;
begin
        writeln;
        writeln( 'введи A,B,C') ; read(A,B,C);
                D:=B*B-4*A*C;
        if D<0 then write('корней нет')
                else begin
                        X1:=(-B+sqrt(D))/(2*A);
                        X2:=(-B-sqrt(D))/(2*A);
                        write('X1=', X1,' X2=', X2);
                end
```

```
end.
```

Предположив, что этот текст (по отношению к процессу трансляции выступающий как **исходный** модуль) сформирован одним из текстовых редакторов, попытаемся отправить его на выполнение. Прежде всего его необходимо перевести в машинный двоичный код (называемый **абсолютным** или **загрузочным модулем**). Для этого на первых этапах осуществляется трансляция (в данном случае, как это реализовано в системах программирования Паскаля, компиляция) исходного текста в машинный код (**объектный модуль**). Однако, объектный модуль не может быть использован для выполнения программы, поскольку в нем нет программ по выполнению процедур ввода (read) и вывода (write, writeln), а также вычисления функции извлечения квадратного корня (sqrt). В исходном тексте программы ссылки на указанные библиотечные подпрограммы отмечены знаком {*}.

Следующий шаг трансляции - компоновка - заключается в подключении к исходному объектному модулю объектных модулей соответствующих подпрограмм в места ссылок на них (исходные тексты этих подпрограмм в системе вовсе отсутствуют). Другими словами, на место процедуры Write помещается подпрограмма, осуществляющая процедуру вывода данных на экран дисплея. Таким образом после компоновки (или, иначе, редактирования связей link editor) возникает абсолютный модуль, намного превышающий по объему размер исходного текста программы, Он и является исполняемым компьютером после его запуска. Расширениями его файлового имени, как правило, являются .com или .exe.

В силу того, что объектные модули не предназначены для непосредственного исполнения, в них обычно нет привязки составляющих их машинных команд к конкретному месту в ОЗУ. Адреса машинных слов бывают условными, что помогает компоновщику размещать объектные модули в свободных местах ОЗУ (заменяя условные адреса команд на конкретные).

Многие системы программирования дополнительно содержат промежуточные этапы трансляции. В этих системах на первом шаге предусмотрена трансляция исходного текста в макроассемблерный код, а затем в объектный модуль. Это связано с историей развития языков программирования, а также с тем, что многие подпрограммы удобнее писать на языке Ассемблера, и подключать их легче на этапе линко-вания ассемблерного модуля с ассемблерными библиотеками подпрограмм.

В современных системах программирования, например, Турбо-Паскаль, Турбо-Си весь этот сложный процесс трансляции с компоновкой подпрограмм скрыт от пользователя и осуществляется специальными компиляторами.

Коротко об отладчиках. Эти программы входят в современные системы программирования и предоставляют средства для просмотра и изменения значений переменных в ходе отладки программы, поиска ошибок и т.д. Использование отладчиков значительно облегчает процесс доводки больших программ.

Заметим, что описанный процесс трансляции характерен для компиляции. Последовательно реализованный интерпретатор объектного модуля фактически не создает. В этом его и недостаток, и достоинство (экономия машинной памяти). Впрочем, у современных ЭВМ, в том числе и персональных, проблема малого ОЗУ отходит на второй план, и интерпретация встречается все реже, так как эффективность этого процесса в целом значительно ниже.

Остается непонятным, как детально происходит трансляция. Пользователь может не уметь сам вручную оттранслировать программу (даже столь короткую, как вышеприведенная), но элементарное понимание этого сложного процесса необходимо.

На первом этапе транслятор производит синтаксический анализ исходной программы - проверяет, не нарушены ли формальные правила, содержащиеся в данном языке программирования. Например, в Паскале текст может встретиться либо внутри текстовой константы (т.е. в апострофах), либо внутри комментария. Если такой текст встретился в другом месте, то это явная ошибка. В системе программирования встроены описания всех синтаксически разрешенных конструкций, и транслятор их применяет к исходной программе. Для задания синтаксиса применяются формы Бэкуса-Наура и синтаксические диаграммы, о которых будет рассказано в следующей главе.

Первой фазой синтаксического анализа является лексический анализ. Он заключается в просмотре литер исходной программы и построении из них лексически допустимых единиц - идентификаторов, ключевых слов языка, чисел и т.д. Во второй фазе эти единицы уже рассматри-

ваются как неделимые и проверяется допустимость их сочетания.

Даже если в синтаксическом смысле исходная программа верна, это не означает, что она имеет смысл в рамках данного языка программирования. На следующем этапе семантического анализа транслятор ищет ошибки такого рода: числа употребления слов BEGIN и END не совпадают; переменные не описаны (в языке, требующем обязательного явного описания переменных), т.е. текст программы *непонятен* (семантика - смысловая сторона языка).

Лишь после того, как в программе все синтаксически правильно и семантически понятно, транслятор переводит операторы программы в машинный код. Это отнюдь не означает, что в программе все благополучно - не исключены ошибки этапа исполнения (деление на ноль, выход за границу массива, переполнение разрядов и т.д.).

Различные фазы компиляции могут быть как последовательными, так и частично перекрывающимися во времени. В зависимости от способа реализации компилятор читает и обрабатывает исходный текст один или несколько раз, называясь соответственно однопроходным, двухпроходным и т.д.

***Контрольные вопросы***

1. Назовите минимальный состав системы программирования, необходимый для разработки программы.
2. Какие имеются сравнительные преимущества и недостатки у компиляторов и интерпретаторов?
3. Назовите основные этапы трансляции программы.

# §3. ПРИКЛАДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ОБЩЕГО НАЗНАЧЕНИЯ

## 3.1. КЛАССИФИКАЦИЯ

Прикладные программы предназначены для того, чтобы обеспечить применение вычислительной техники в различных сферах деятельности человека. Помимо создания новых программных продуктов разработчики прикладных программ большие усилия тратят на совершенствование и модернизацию популярных систем, создание их новых версий. Новые версии, как правило, поддерживают старые, сохраняя преемственность, и включают в себя базовый минимум (стандарт) возможностей.

Один из возможных вариантов классификации программных средств (ПС), составляющих прикладное программное обеспечение (ППО), отражен на рис.2.11. Как и почти всякая классификация, приведенная на рисунке не является единственно возможной. В ней представлены даже не все виды прикладных программ. Тем не менее, использование классификации полезно для создания общего представления о ППО.

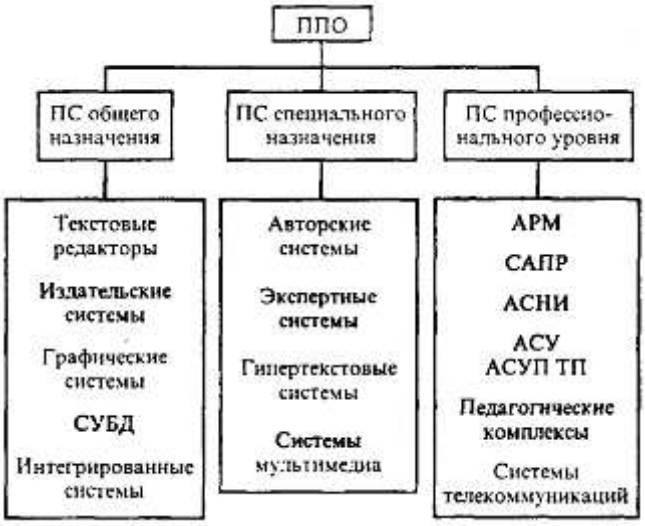



*Рис.2. П.* Классификация прикладного программного обеспечения

## 3.2. ИНСТРУМЕНТАЛЬНЫЕ ПРОГРАММНЫЕ СРЕДСТВА ОБЩЕГО НАЗНАЧЕНИЯ

Несмотря на широкие возможности использования компьютеров для обработки самой разной информации, самыми популярными являются программы, предназначенные для работы с текстами - текстовые редакторы и издательские системы. Текстовыми редакторами называют программы для ввода, обработки, хранения и печатания текстовой информации в удобном для пользователя виде. Эксперты оценивают использование компьютера в качестве печатающей машинки в 80%.

Большую популярность приобрели программы обработки графической информации. Компьютерная графика в настоящее время является одной из самых динамично развивающихся областей программного обеспечения. Она включает в себя ввод, обработку и вывод графической информации - чертежей, рисунков, картин, текстов и т.д. - средствами компьютерной техники. Различные типы графических систем позволяют быстро строить изображения, вводить иллюстрации с помощью сканера или видеокамеры, создавать анимационные ролики.

Графические редакторы позволяют пользоваться различным инструментарием художника, стандартными библиотеками изображений, наборами стандартных шрифтов, редактированием изображений, копированием и перемещением фрагментов по страницам экрана и др. Для выполнения расчетов и дальнейшей обработки числовой информации существуют специальные программы - электронные таблицы. В процессе деятельности любого специалиста часто требуется представить результаты работы в виде таблиц, где одна часть полей занята исходными данными, а другая -. результатами вычислений и графического анализа. Характерными для них является большой объем перерабатываемой информации, необходимость многократных расчетов при изменении исходных данных. Автоматизацией подобной рутинной работы и занимаются электронные таблицы.

Одним из наиболее перспективных направлений развития вычислительной техники является создание специальных аппаратных средств для хранения гигантских массивов информационных данных, и последующей нечисловой обработки их -поиска и сортировки. Для компьютерной обработки подобных **баз данных** используют **системы управления базами данных.** СУБД - это набор средств программного обеспечения, необходимых для создания, обработки и вывода записей баз данных. Различают несколько типов СУБД: *иерархические, сетевые, реляционные.* При работе с СУБД выделяют несколько последовательных этапов:

- проектирование базы данных;
- создание структуры базы данных;
- заполнение базы данных;
- просмотр и редактирование базы данных**;**
- сортировку базы данных;
- поиск необходимой записи;
- выборку информации;
- создание отчетов.

Как правило, большинство популярных систем управления базами данных поддерживают эти этапы и предоставляют удобный инструментарий для их реализации.

Желание объединить функции различных прикладных программ в единую систему привело к созданию интегрированных систем. Универсальные *интегрированные системы* разрабатывались по принципу единой системы, содержащей в качестве элементов текстовые и графические редакторы, электронные таблицы и систему управления базами данных. Примеры: Framework, Works, Мастер. Современная концепция интеграции программных средств - кооперация отдельных прикладных программных систем по типу широко известного пакета MicroSoft Office. Сами системы, входящие в пакет, являются независимыми, более того, они сами представляют локально интегрированный пакет, поскольку помимо основной своей задачи поддерживают функции других систем. Например, текстовый редактор Word обладает возможностью манипулировать с электронными таблицами и базами данных, а в электронной таблице Excel встроен мощный текстовый редактор. Для сопряжения информационных данных из различных программных систем в них предусматривают импорт-экспортную систему обмена с перекодировкой форматов представления данных.

## 3.3. ИНСТРУМЕНТАЛЬНЫЕ ПРОГРАММНЫЕ СРЕДСТВА

Разработчики создают специальные программные системы целевого назначения для специалистов в некоторой предметной области. Такие программы называют авторскими инструментальными системами. **Авторская система** представляет интегрированную среду с заданной интерфейсной оболочкой, которую пользователь может наполнить информационным содержанием своей предметной области.

**Экспертная система** - это программа, которая ведет себя подобно эксперту в некоторой узкой прикладной области. Экспертные системы призваны решать задачи с неопределенностью и неполными исходными данными, требующие для своего решения экспертных знаний.

Кроме того, эти системы должны уметь объяснять свое поведение и свое решение.

Принципиальным отличием экспертных систем от других программ является их адаптивность, т.е. изменчивость в процессе самообучения.

Принято выделять в экспертных системах три основных модуля:

• модуль базы знаний;

• модуль логического вывода;

• интерфейс с пользователем.

Экспертные системы, являющиеся основой искусственного интеллекта, получили широкое распространение в науке (классификация животных и растений по видам, химический анализ), в медицине (постановка диагноза, анализ электрокардиограмм, определение методов лечения), в технике (поиск неисправностей в технических устройствах, слежение за полетом космических кораблей и спутников), в политологии и социологии, криминалистике, лингвистике и т.д.

В последнее время широкую популярность получили программы обработки гипертекстовой информации. Гипертекст – это форма организации текстового материала не в линейной последовательности, а в форме указании возможных переходов (ссылок), связей между отдельными его фрагментами. В обычном тексте используется обычный линейный принцип размещения информации и доступ к нему осуществляется последовательно. В **гипертекстовых системах** информация напоминает текст энциклопедии, и доступ к любому выделенному фрагменту текста осуществляется произвольно по ссылке. Организация информации в гипертекстовой форме используется при создании справочных пособий, словарей, контекстной помощи (Help) в прикладных программах.

Расширение концепции гипертекста на графическую и звуковую информацию приводит к понятию гипермедиа. Идеи **гипермедиа** получили распространение в сетевых технологиях, в частности в Интернет-технологиях. Технология WWW (World Wide Web) позволила структурировать громадные мировые информационные ресурсы посредством гипертекстовых ссылок. Появились программные средства, позволяющие создавать подобные Web-странички. Стали развиваться механизмы поиска нужной информации в лабиринте информационных потоков. Популярными поисковыми средствами в Интернет являются Yahoo, AltaVista, Magellan, Rambler и др.

**Мультимедиа** (multimedia) - это взаимодействие визуальных и аудиоэффектов под управлением интерактивного программного обеспечения. Появление и широкое распространение компакт-дисков (CD-ROM) сделало эффективным использование мультимедиа в рекламной и информационной службе, сетевых телекоммуникационных технологиях,обучении.

Мультимедийные игровые и обучающие системы начинают вытеснять традиционные «бумажные библиотеки». Сегодня в библиотеках CD-ROM можно «гулять» по музеям, Московскому Кремлю и т.д. с помощью «электронного путеводителя».

## 3.4. ПРОГРАММНЫЕ СРЕДСТВА ПРОФЕССИОНАЛЬНОГО УРОВНЯ

Каждая прикладная программа этой группы ориентируются на достаточно узкую предметную область, но проникает в нее максимально глубоко. Так функционируют **АСНИ** - автоматизированные системы научных исследований, каждая из которых «привязана» к определенной области науки, **САПР** - системы автоматизированного проектирования, каждая из которых также работает в узкой области, **АСУ** - автоматизированные системы управления (которых в 60 - 70 годах были разработаны тысячи).

Наконец, еще раз подчеркнем не только условность предложенной выше классификации, но и наличие пересечений. Так, каждую конкретную экспертную систему вполне можно отнести к ППО профессионального уровня; принцип гипертекста реализован в ряде авторских систем и т.д.

## 3.5. ОРГАНИЗАЦИЯ «МЕНЮ» В ПРОГРАММНЫХ СИСТЕМАХ

Прикладные программы нацелены на широкий круг пользователей (непрограммистов) и предполагают диалоговый режим работы человека с компьютером. Широкой популярностью пользуются программы, обладающие дружественным интерфейсом, т.е. таким, который не требует от пользователя больших усилий в работе со всеми необходимыми периферийными устройствами, специальных настроек компьютера и обладает удобной системой управления и диалога.

Интерактивный режим в прикладных программах осуществляется по двум принципам: «смотри и выбирай» и «подтверждай то, что я делаю». Программы не утрачивают работоспособности при ошибках пользователя, позволяют легко и безболезненно исправлять ошибочные действия путем их отмены, а также обращаться в любой момент к контекстной помощи. Все принципы дружественного интерфейса реализуются специальной системой интерактивного (диалогового) общения компьютерной программы и пользователя, называемой «пользовательским меню» или просто «меню».

«Меню» представляет набор команд, указаний и данных, который в любой момент доступен пользователю для выбора дальнейшего действия. Указатель (курсор) имеет возможность циклически сканировать меню и управляется клавишами со стрелками и/или манипулятором «мышь». Выбор команды осуществляется установкой курсора на его пункт и нажатием клавиши ввода <Enter>, или указанием стрелки «мыши» и двойным щелчком ее клавиши. Более быстрый выбор команды может быть осуществлен нажатием специальных клавиш или их комбинацией. Как правило, в них участвуют клавиши с буквой, с которой начинается название команды. По своей организации меню представляет иерархическую структуру с системой вложенных подменю («выплывающие», «ниспадающие», «оконные» и пр.) с возможностью возврата из любого пункта в главное (основное) меню.

Меню бывает текстовым и/или графическим с комментариями по каждому своему пункту. Прикладные программы дополнительно имеют функциональное клавишное меню для быстрого выполнения каких-либо команд («горячие клавиши» -«hot key»). Например, функциональная клавиша F1 чаще используется для экстренного вызова справочной информации (Help - помощь), клавиша F2 - для сохранения данных во внешней памяти.

Фирмы-разработчики программных средств организуют программные меню по своим стандартам и единообразно. Так, например, фирма «Борланд» практикует свой фирменный интерфейс, который легко распознается всеми программистами и пользователями, работающими в системах программирования Турбо: Турбо-Паскаль, Турбо-Бейсик, Турбо-Си и т.п. (см. выше рис.2.10). В верхней части экрана дисплея в Турбо-программах помещается горизонтально главное меню, каждый пункт которого может иметь ниспадающее подменю. В нижней части помещаются команды для функциональных клавиш («горячие» клавиши), рядом - строка статуса, которая дает комментарии к выбираемым командам меню и некоторые параметры состояния прикладной программы.

Таким образом, меню - это некоторый перечень команд (функций), которые имеются в распоряжении пользователя на различных этапах работы с программной системой.

Исторически первым видом «меню» можно считать перечень команд в виде пронумерованного списка возможных функций

Например:

1 - редактирование текста;
2-трансляция программы;
3 - выход.

Оставалось лишь выбрать режим путем нажатия клавиши с его номером и клавиши <Ввод>.

Другой вариант такого меню - ненумерованный список команд, в котором выбор нужной команды осуществляется нажатием клавиши с первой буквой ее имени.

Например:

Редактирование текста;
Трансляция программ;
Выход.

Еще один схожий способ, бывший ранее популярным - наличие справа или слева от списка стрелки, движение которой по вертикали управляется клавишами ↓ и ↑. Установив стрелку против нужной строки и нажав клавишу <Ввод>, осуществляем выбор нужной команды.

Например:

Редактирование текста;
Трансляция программ; <=
Выход.

Подобный вид меню определялся уровнем развития аппаратной и программной частей компьютера и вынужденной необходимостью ориентации на алфавитно-цифровые (символьные) дисплеи.

Дальнейшее развитие числовых меню привело к символьным меню, в которых выбор заданной функции осуществлялся с помощью ввода соответствующего символа или с помощью функциональных клавиш клавиатуры Fl, F2, F3 ...

Принципиальным шагом в организации эффективных пользовательских меню стало использование графических средств. Появилась возможность создать указатель (в разных программных средах - светящийся курсор, стрелка, выделенный прямоугольник и т.п.), перемещающийся с помощью клавиш со стрелками, а выбор пункта меню осуществлять нажатием специальной клавиши, как правило, ENTER или <Пробел>.

Один из признаков дружественности меню - когда подведение указателя к некоторому элементу меню отображает комментарий функционального назначения. Например, в пункте *Текст* может появиться комментарий:

*Операции над текстом в целом: считывание, запись, печать*

Если выбрать этот пункт меню нажатием клавиши ввода ENTER, то появится подменю команд работы с текстом.

Подобный принцип иерархии в построении меню, который включает главное (основное меню) и дерево подменю, позаимствован из организации структуры директорий (каталогов) файловой системы компьютера. Теперь появилась возможность строить не только дружественный интерфейс, но и дизайн. Возникли меню с системой «ниспадающих», «всплывающих», «многооконных» и т.д. подменю.

Современные типы меню строятся с использованием графического и символьного режимов. Символьный принцип в меню используют для выбора быстрых команд. Соответствующим командам назначаются клавиши, их комбинации или функциональные клавиши Fl ... F 12.

Существует определенная традиция действий «горячих» клавиш. В частности, в большинстве программ клавиша F2 сохраняет результаты работы, комбинация ALT+X осуществляет выход из программы и т.д.

Значительные удобства пользователю предоставляет специальный манипулятор «мышь», который позволил серьезно облегчить и предоставить комфортные условия работы. Перемещая с помощью мыши указатель, можно одним нажатием кнопки мыши вызвать требуемую функцию.

Современные программные системы построены на интерактивных меню, использующих принцип «кнопки», которые требуют от пользователя минимальных знаний и действий. В них закладывается удобный и оптимальный для работы человека диалоговый режим. Меню содержат интерактивные формы:

• с шаблоном ответа

*Продолжить?* Y/N

• со справочником ответа

*Какой цвет назначить: черный белый красный голубой*

• с назначением параметров, рис. 2.12;
• многостраничные формы, рис. 2.13.

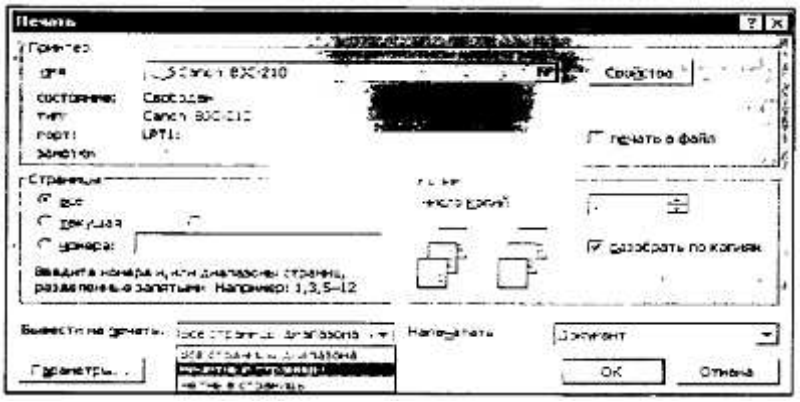



*Рис. 2.12.* Пример меню с назначением параметров (меню формы печати в Word)


*Рис.2.13.* Пример многостраничного меню (свойства обозревателя Internet Explorer 4.0)

Интерфейс современных меню строится таким образом, чтобы запрос-ответ пользователя был однозначен, прост и удобен. В силу большой значимости систем организации меню, в большинстве инструментальных сред для разработки программ предусматриваются специальные процедуры и объекты создания меню. Особенное внимание им уделяется в системах управления базами данных (СУБД), авторских системах, в объектно-ориентированных языках программирования.

При работе е прикладными программами пользователю приходится сталкиваться в первую очередь с меню. От того, как он освоил работу с его пунктами, зависит эффективность использования информационной технологии.

### *Контрольные вопросы ч задания*

1. Что такое текстовые редакторы? Для чего они нужны?
2. Что такое графические редакторы? Для чего они нужны?
3. Что такое электронные таблицы? Для чего они нужны?
4. Что такое СУБД? Для чего они нужны?
5. Что такое интегрированные системы? Для чего они нужны?
6. Что такое экспертные системы? Для чего они нужны?
7. Что такое авторские системы? Для чего они нужны?
8. Что такое гипертекст, гипермедиа?
9. Что такое мультимедиа?
10. Какие бывают типы меню?
11. Для чего в меню используют «горячие» клавиши?
12. Разработайте программу «меню» на одном из языков программирования.

## § 4. СИСТЕМЫ ОБРАБОТКИ ТЕКСТОВ

## 4.1. ЭЛЕМЕНТЫ ИЗДАТЕЛЬСКОГО ДЕЛА

Для того, чтобы уверенно работать с текстовыми редакторами и настольными издательскими системами, необходимо освоить и уяснить некоторые сведения из издательского дела.

Особую значимость при подготовке и формировании текста для издания имеют шрифты. **Шрифты -** основное изобразительное средство издательских систем, с их помощью можно.добиться большой художественной выразительности текста.

Шрифты различают по *гарнитуре* (рисунку), *начертанию, размеру* и *назначению.* Гарнитурой называется совокупность шрифтов одного рисунка во всех начертаниях и кеглях. Кегль - размер шрифта, определяемый размером литеры по верикали, исчисляемый в *пунктах* (1 пункт равен 0,367 мм) Полный комплект гарнитуры содержит шрифты всех начертании и кеглей, а в каждом кегле - русский и латинский (и, если нужно, другие) алфавиты прописных и строчных букв, а также относящиеся к ним знаки.

Различия между буквами разных шрифтов объясняется их различным построением. Среди элементов из которых строятся буквы выделяют:

- основные штрихи (задают структуру буквы);
- дополнительные штрихи (играют вспомогательную и соединительную роль);
- засечки;
- верхние и нижние выносные элементы;
- овалы и полуовалы (с наплывами или без них);
- концевые элементы.

Буквы располагаются по *базовой линии.* Расстояние между строками называют *интерлиньяжем.* Отношение толщины основных и дополнительных элементов определяет контрастность шрифта. Различают неконтрастные, малоконтрастные, контрастные и очень контрастные шрифты. Форму букв шрифта определяют *цветность* и *ритм* (соотношение черного и белого, просветы, наплывы и пр.). Отношение высоты буквы к ее ширине называют шириной шрифта. Бывают сверхузкие, узкие, нормальные, широкие и сверхширокие шрифты. *Насыщенность* шрифта определяется *светлотой.* Бывают сверхсветлые, светлые, нормальные, полужирные, жирные и сверхжирные шрифты. Шрифты могут быть прямыми и наклонными. Наклонный вариант шрифтов часто называют *курсивом.* Шрифты одного типа, но отличающиеся по ширине, насыщенности и наклону являются различными *начертаниями* одного и того шрифта. Начертания одного шрифта составляют *гарнитуру.*

Шрифт на компьютере - это файл или группа файлов, обеспечивающих вывод текста на печать со стилевыми особенностями шрифта. Существуют программы, позволяющие создавать собственные варианты шрифтов. Как правило, такие программы входят в состав текстовых редакторов и издательских систем. Однако, существует довольно большой спектр стандартных (в компьютерном смысле) шрифтов, разработанных полиграфистами. Приведем некоторые из них:

| | |
|---|---|
| Akademy | Аа Бб Вв Гг Дд 1 2 3 4 5 6 7 8 9 0 |
| Arial Cyr | Аа Бб Вв Гг Дд 1 2 3 4 5 6 7 8 9 0 |
| Baltika | Аа Бб Вв Гг Дд 1 2 3 4 5 6 7 8 9 0 |
| Courier New Cyr | Аа Бб Вв Гг Дд 1 2 3 4 5 6 7 8 9 0 |
| DESDEMONA | ÀÀ ÁÁ ÂÂ ÃÃ ÄÄ 1234567890 |

и др. Основной текст данной главы набран шрифтом Times New Roman Cir. Наиболее часто используются следующие гарнитуры:

1. *Литературная* - для набора всех видов изданий, кроме букварей, энциклопедий, а также малоформатных, нормативных, периодических и литературно-художественных изданий;

2. *Обыкновенная новая* - для набора всех видов книжно-журнальных изданий**,** кроме учебников для школы, карманных и нормативных изданий;

3. *Банниковская* и *Академическая* - для набора литературно-художественных, научных, учебных и научно-популярных изданий по гуманитарным областям знаний;

4. *Школьная* - для набора учебников начальной и средней школы, изданий для детей, художественной и научно-популярной литературы, нормативных изданий и массовых иллюстрированных журналов;

5. *Балтика* и *Таймс* - для набора художественной и научно-популярной литературы, вузовских учебников.

Каждый текст, подготовленный к изданию в качестве брошюры или книги, должен пройти техническое редактирование, которое предполагает подготовку *оригинал-макета* готового издания. Издательским текстовым оригинал-макетом является набранный, сверстанный на компьютере и отпечатанный на лазерном (или струйном) принтере текстовой оригинал, представляющий собой точный прообраз будущего издания (по числу страниц, абзацев, рисунку шрифта), предназначенный для изготовления печатной формы. Исходным материалом для подготовки оригинал-макета является отредактированный, вычитанный автором и литературным редактором размеченный текстовый оригинал. Техническое редактирование- процесс достаточно сложный, особенно если в тексте есть таблицы, схемы, формулы и иллюстрации.

## 4.2. ТЕКСТОВЫЕ РЕДАКТОРЫ

Программы-текстовые редакторы предназначены для подготовки текстовых материалов на компьютере.

Поскольку текстовые материалы бывают различной сложности по набору и верстке, существуют и развиваются различные программы обработки текстов. Их можно классифицировать по уровням требований к обработке текстов. Перечислим наиболее популярные из них.

А. Программы для набора и обработки простых текстов:

Lexicon
Refis
Norton Editor
MultiEdit
Word 4.0 и 5.0 и др.

Б. Программы для набора сложных текстов:

ChiWriter
TechWord
Word 5.5, 6.0 и 7.0
Word for Windows 1.1 и 2.2
WordPerfect
TeX, LaTex
и др.

Каждый текстовый редактор имеет свои технические требования к составу и конфигурации компьютерной техники.

Для набора простого текста достаточно иметь

• персональный компьютер IBM PC/XT, AT (или совместимые с ними) с емкостью оперативной памяти 640 кбайт;

• дисплей монохромный CGA, EGA или Hercules 14.

Для набора сложных текстов, содержащих формулы, таблицы, схемы необходимо иметь

• персональный компьютер IBM PC/AT 286 (или совместимые с ним), с емкостью оперативной памяти 1 Мбайт;

• дисплей монохромный CGA, EGA или Hercules 14;

• накопитель на жестком магнитном диске (30 - 40 Мбайт);

• матричный 9- или 24-игольчатый принтер, струйный или лазерный принтер;

• мышь.

Для подготовки черно-белых иллюстраций, логотипов, обложек, титулов, разработки шрифтов необходимо иметь

• персональный компьютер IBM PC/AT 386/486 (или совместимые с ним) с емкостью оперативной памяти 4 Мбайт;

• дисплей цветной VGA, SVGA 14/15;

• накопитель на жестком магнитном диске (более 80 Мбайт);

• лазерный принтер;
• сканер;
• мышь.

Разумеется, подходят не только компьютеры IBM-совместимые, но и другие (например, Macintosh); разные модели IBM указаны лишь поскольку они наиболее распространены для ориентации по уровню требований.

Разработчики программ текстовых редакторов стараются предусмотреть в них предоставление пользователю всех необходимых операций и сервисных возможностей для эффективной обработки текстов Выделим главные из них:

- набор текста в интерактивном режиме;
- редактирование текста;
- работа с фрагментами текста (копирование, перемещение, удаление и т.п.);
- форматирование текста (установка абзаца, перенос, выравнивание границ строки и т.п);
- работа с несколькими текстами одновременно посредством многооконного принципа;
- файловая организация работы с текстами и взаимодействие с операционной системой;
- импорт/экспорт текстов из одного формата в другой, в другие прикладные системы;
- работа с разными шрифтами;
- работа со спецсимволами (математические знаки, индексы и т.п.);
- работа с иллюстративным материалом (таблицы, схемы, формулы и пр.);
- проверка правописания;
- поиск и замена фрагментов текста.

Рассмотрим для примера меню популярного в России текстового редактора Лексикон, рис. 2.14.


*Рис. 2.14.* Главное меню и строка статуса текстового редактора Лексикон

Здесь верхняя строка одновременно является строкой комментариев к пунктам меню, и пользователю легко понять их назначение. Функциональные клавиши имеют достаточно стандартные назначения. Например, F1 - помощь, F9 - переключение с русского на латинский режимы, F10 - выход в меню.

Меню позволяет осуществить настройку редактора на подготовку того или иного документа, связаться с операционной системой. Правила набора, редактирования текста просты и очевидны.

## 4.3. ИЗДАТЕЛЬСКИЕ СИСТЕМЫ

### *Общая характеристика*

Настольные компьютерные издательские системы приобрели широкую популярность в различных сферах производства, бизнеса, науки, культуры и образования. Издательское дело становится актуальным практически для любой организации. Выпуск информационных бюллетеней, рекламных проспектов, собственных малотиражных газет и даже книг теперь становится необходимым атрибутом информационного обеспечения современных учреждений. Пожалуй, из всех новых информационных технологий, компьютерное издательство является наиболее массовой и практически легко внедряемой.

Настольные издательские системы (desktop publishing) представляют собой комплекс аппаратных и программных средств, предназначенных для компьютерного набора, верстки и издания текстовых и иллюстративных материалов.

Минимальные аппаратно-технические требования для развертывания издательских систем таковы:

• персональный компьютер: IBM PC/AT 486 или другой того же класса, с емкостью оперативной памяти 8 Мбайт;

• дисплей: цветной VGA, SVGA 15/17;

• накопитель на жестком магнитном диске (более 160 Мбайт);

• цветной лазерный принтер;

• цветной струйный принтер;

• цветной сканер,

• мышь.

Существуют различные программные системы, среди них наиболее распространены следующие. Word for Windows, Express Publisher, Illustrator for Windows, Ventura Publisher, PageMarker, TeX. Перечисленные программные системы предназначены для компьютерной верстки. Среди программ подготовки иллюстраций можно выделить следующие: CorelDraw, CorelSystem, Designer, DrawPerfect, GalleryEffect, PC Paintbrush, PhotoStyler, Adobe Photoshop и др

Для издательских систем существуют различные сервисные программы обработки текстовых материалов. Среди них можно выделить 7 основных групп:

преобразования растровой графики в векторную;

обработки сканированных изображений;

обработки шрифтов;

проверки правописания;

чтения текстов с помощью сканера;

русификации программ;

программы-переводчики.

В начале 1998 г. компания «Corel» анонсировала 8-ю версию графического пакета CorelDraw, которая работает под управлением Windows 95 или Windows NT, включает дополнительные функции, повышающие производительность труда дизайнеров графики и поддерживает работу в Internet. Работа с издательскими системами предусматривавает

• работу с меню и с диалоговой панелью,

• работу с текстом и иллюстрациями (набор и формирование);

• редактирование текста и иллюстраций,

• оформление текста (форматирование; выбор шрифтов, гарнитуры, стиля, размеров и т.п.);

• настройку экрана.

Издательские системы имеют стандартные правила работы с меню, командами, сервисными утилитами.

Ниже приведены стандартные наименования клавиш и их комбинаций для выполнения обычных манипуляций при работе с большинством текстовых редакторов и настольных издательских систем.

Программы для DOS И WINDOWS:

| | |
|---|---|
| Alt или F10 | Активизировать командное меню |
| Alt + буква | Вызов пункта меню (параметра) |
| Alt + F4 | Выход из программы |
| Alt + Shift + Tab | Переход в окно с предыдущей программой |
| Alt + Tab | Переход в окно со следующей программой |
| BkSp | Удалить символ перед курсором |
| Ctrl + → | Перемещение курсора на слово вправо |
| Ctrl + ← | Перемещение курсора на слово влево |
| Ctrl + Del | Удалить документ |
| Ctrl + End | Перемещение курсора в конец поля ввода |
| Ctrl + Enter | Начать новую страницу |
| Ctrl + Home | Перемещение курсора в начало поля ввода |
| Ctrl + Ins | Копировать в буфер (клипбоард) |
| Ctrl + N | Создать новый документ |
| Ctrl + O | Открыть файл (редактировать уже имеющийся) |
| Ctrl+P | Печать |

| | |
|---|---|
| Ctrl + PgDn | Прокрутка на экран вниз |
| Ctrl + PgUp | Прокрутка на экран вверх |
| Ctrl + Q | Выход из программы |
| Ctrl + S | Сохранить файл |
| Ctrl + X | Выход из программы |
| Del | Удалять символ или выделенный фрагмент |
| Enter | Выполнить действие или команду |
| Esc | Выйти без сохранения выбранных параметров |
| Esc | Отменить выполнение |
| F1 | Помощь |
| F4 | Повторить последнюю выполненную команду |
| Ins | Включить/выключить режим вставки/замены |
| PgDn | Перемещение курсора на экран вниз |
| PgUp | Перемещение курсора на экран вверх |
| Shift + <— | Распространить выделение на символ влево |
| Shift + —> | Распространить выделение на символ вправо |
| Shift + Del | Удалить в буфер (клипбоард) |
| Shift + Ins | Вставить из буфера (клипбоарда) |
| Shift + Tab | Перемещение курсора в предыдущее поле |
| Tab | Перемещение курсора в следующую позицию табуляции |

***Настольная издательская система WORD***

В последнее время все большую популярность среди широкого круга пользователей завоевывает текстовый процессор Word для Windows. Прежде всего из-за высококачественных издательских свойств этого продукта фирмы «Microsoft». Несмотря на то, что большинство людей, работающих с WinWord, используют далеко не все его возможности, они могут создавать красиво оформленные, хорошего качества документы, разрабатывать фирменные бланки, визитки, создавать элементы фирменного стиля, рекламные документы. Возможность вставлять таблицы, рисунки, графики, формулы в тексты, создаваемые в Word, также способствует завоеванию абсолютного первенства этого программного продукта.

Word - это многофункциональная программа обработки текстов, настольная издательская система. Ее предназначение:

• набор, редактирование, верстка текста и таблиц;

• управление всеми пунктами меню, опциями и командами с помощью мыши;

• просмотр на дисплее готового к печати документа без затраты бумаги на дополнительную распечатку;

• вставка рисунков и слайдов;

• заготовка бланков, писем и других документов;

• обмен информацией с другими программами;

• проверка орфографии и поиск синонимов.

Для запуска текстового процессора дважды «щелкните» левой клавишей мыши на значке запускаемой программы. Кратко охарактеризуем возникшее интерфейсное окно.

**Пиктографическое меню -** это строка пиктограмм, состоящая из полей кнопок с изображением той или иной операции на них. В большинстве случаев кнопки дублируют наиболее часто используемые операции, доступные и в обычных меню.

**Панель форматирования** - это строка пиктограмм, состоящая из элементов, необходимых для оформления текста:

• *полей списков* (они справа снабжены стрелкой, направленной вниз; в результате нажатия мышью на стрелку на экране открывается окно списка, в котором перечисляются доступные для выбора элементы списка);

• *полей пиктограмм* (если фрагмент текста маркирован, то нажатие некоторой кнопки на линейке форматирования применяет связанную с данной кнопкой функцию).

**Координатная линейка** располагается над окном документа. С помощью координатной линейки можно изменять абзацные отступы, длину строки набора и ширину колонок.

**Строка состояния** находится на нижней кромке окна Word. В процессе ввода данных в этой строке высвечивается информация о позиции курсора ввода и др.

**Линейки прокрутки** - расположены вдоль правого и нижнего краев рабочего окна. Перетаскивая мышкой бегунок вы можете быстро передвигаться по текста.

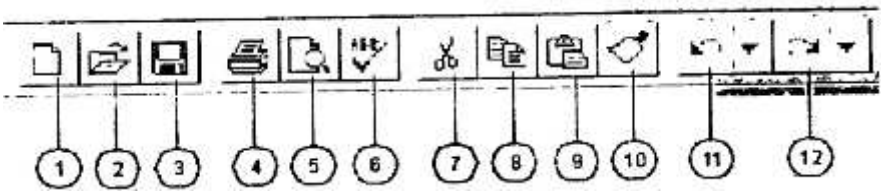



*Рис. 2. 15*,а. Пиктографическое меню программы Word

На рисунке 2.15, а:
1 - создать новый документ;
2 - открыть существующий документ;
3 - сохранение активного документа;
4 - печать активного документа;
5 - предварительный просмотр документа перед печатью;
6 - проверка орфографии активного документа;
7 - удаление выделенного фрагмента в буфер;
8 - копирование выделенного фрагмента в буфер;
9 - вставка содержимого буфера в позицию курсора;
10 - копирование формата;
11 - отмена последнего действия;
12-повторное выполнение последнего отмененного действия.

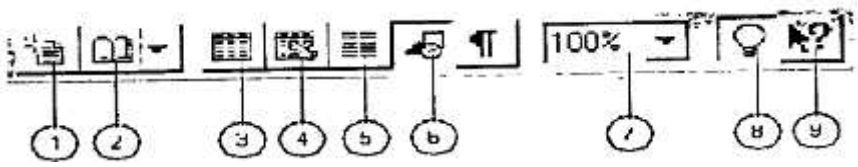



*Рис. 2.15,б.* Пиктографическое меню программы Word (продолжение)

На рисунке 2.15, б:
1 - автоматическое форматирование текста;
2 - вставка адреса из адресной книги;
3 - вставка таблицы;
4 - вставка листа MS Excel;
5 - изменение формата колонок;
6 - отображение панели рисования;
7 - управление масштабом;
8 - отображение панели инструментов мастера подсказок;
9 - помощь.

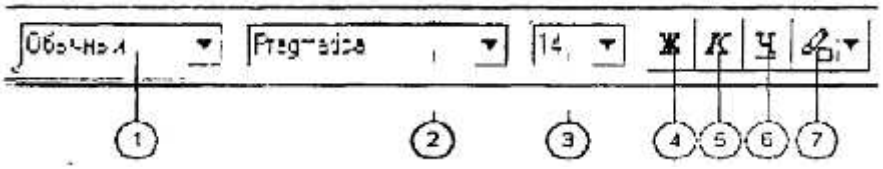



*Рис. 2.15.в.* Панель форматирования программы Word

На рисунке 2.15, в:
1 - применение стиля;
2 - изменение шрифта;

3 - изменение кегля;
4 - установка и отмена полужирного начертания;
5 - установка и отмена курсивного начертания;
6 - установка и отмена подчеркнутого начертания;
7 - выделение цветом выделенного фрагмента.

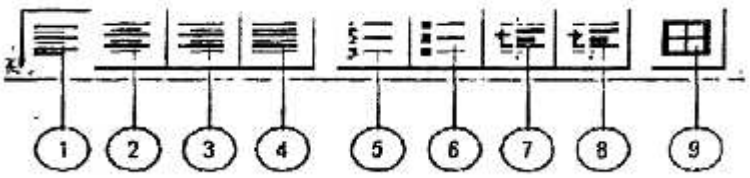



*Рис. 2.15,г.* Панель форматирования программы Word (продолжение)

На рисунке 2.15, г:
1 - выравнивание абзаца по левому краю;
2 - выравнивание абзаца по центру;
3 - выравнивание абзаца по правому краю;
4 - выравнивание абзаца по ширине;
5 - создание нумерованного списка;
6 - создание маркированного списка;
7 - уменьшение отступа или повышение уровня выделенного текста;
8 - увеличение отступа или понижение уровня выделенного текста;
9 - отображение панели инструментов «Обрамление».

Приведенные выше панели используются не только в программе Word; они типичны и для других программ семейства MS Office.

Для выхода из текстового процессора можно дважды «щелкнуть» по соответствующему Word пиктографическому значку системного меню или нажать комбинацию клавиш Alt + F4.

Загрузка документа осуществляется двумя способами:

1) нажатием кнопки с изображением раскрытой папки на панели пиктографического меню:

2) использованием команды меню *Файл* + *Открыть.* Далее выбирается нужный файл и нажимается кнопка ОК.

Редактирование текста заключается в удалении, добавления, копировании и переносе фрагментов текста, проверке орфографии с помощью уже известных клавиш клавиатуры или пиктографического меню. Опишем основные приемы редактирования текста в текстовом процессоре WinWord.

**Удаление буквы** (символа) - это простейшая функция редактирования. Если вы случайно нажали клавишу с ненужной буквой при вводе текста, то чтобы удалить ошибочно введенный символ,

• расположенный слева от курсора, надо нажать на клавишу Backspace, при этом курсор передвигается на одну позицию влево;

• расположенный справа от курсора, надо нажать на клавишу Delete, при этом курсор остается на месте

Для удаления участка текста надо его выделить и нажать клавишу Delete.

**Как выделить участок текста?**

*Полоса выделения* - это невидимая полоса вдоль левой границы окна документа, используемая для выделения текста с помощью мыши. Чтобы выделить текст надо:

• установить указатель мыши на полосе выделения рядом с текстом;

• нажать левую клавишу мыши и, удерживая ее, передвигать мышь в нужном направлении.

**Для отмены удаления**

• на панели инструментов пиктографического меню нажать на кнопку с изображением изогнутой влево стрелки;

• или использовать команду меню *Правка* + *Отменить ввод.*

**Добавление текста:**

установить текстовый курсор на конец абзаца и нажать клавишу Enter, чтобы начать новый абзац.

Существует два вида **копирования и перемещения участков текста.**

1.Ручная техника.

На небольших расстояниях:

• выделите текст, который хотите переместить или скопировать;

• установите указатель мыши на выделенном тексте, нажмите левую кнопку мыши и удерживайте ее - в этот момент вы как бы «схватили» выделенный участок текста. При этом изменится форма гекстового курсора и указателя мыши. Текстовый курсор примет форму штриховой вертикальной линии, а в нижнем конце указателя мыши появится небольшой прямоугольник;

• нажмите клавишу Ctrl, если хотите скопировать выделенный участок текста;

• перемещая указатель мыши, установите штриховой текстовый курсор в ту позицию, куда хотите переместить или скопировать выделенный участок текста, и отпустите кнопку мыши.

2. При переносе текста на большие расстояния воспользуйтесь линейками прокрутки:

• выделите необходимый участок текста;

• прокрутите текст в окне так, чтобы часть, в которую вы хотите перенести или скопировать текст, стала видимой;

• нажмите клавиши Ctrl и Shift и удерживайте их;

• установите указатель мыши в позицию текста, куда хотите вставить свой текст, и нажмите правую кнопку мыши.

Копирование и перемещение участков текста можно также осуществить с помощью Буфера Обмена - участка памяти, в которой временно помещается вырезанный или скопированный текст или графика:

• выделите текст, который хотите переместить или скопировать;

• выберите одну из команд меню *Правка - Вырезать* или *Правка - Копировать*;

• установите текстовый курсор в нужное место;

• выберите команду меню *Правка + Вставить.*

**Сохранение текста**:

а) на панели пиктографического меню нажать на кнопку с изображением дискеты;

б) или использовать команду меню *Файл + Сохранить.* Откроется диалоговое окно, в котором надо выбрать *Сохранить.*

В поле *Файл* диалогового окна *Сохранить* вы должны ввести имя документа, причем буквы в имени должны быть английскими и расширение у файла .doc . По окончании ввода нажмите на кнопку ОК. Откроется диалоговое окно *Сводка.* В него можно ввести дополнительную информацию о документе, которая будет храниться в том же файле, что и документ. Нажать кнопку ОК.

На экране монитора текст может представляться в различном масштабе и в различном виде, за это отвечает кнопка *Команда* меню *Вид*

*Вид + Разметка страницы*

Разбивает весь текст на страницы. Используется режим Макета.

*Вид + Обычный*

Увеличивает страницу на 100%. Изображение текста нормальное.

*Вид + Структура Документа*

Используется режим Эскиза. WinWord способен отображать на экране непечатаемые символы, такие, как пробел, конец абзаца, табуляцию.

Большинство функций форматирования вызываются с помощью панели инструментов форматирования. Панель инструментов форматирования содержит различные поля, с помощью которых можно вызывать функции редактирования.

Для того чтобы отформатировать символ или группу символов с помощью панели инструментов форматирования, надо

• выделить символ или группу символов;

• открыть список шрифтов и выбрать нужный, задать размер шрифта и выбрать стиль его написания;

• щелкнуть мышью в любом месте документа, чтобы убрать выделение.

Для **вставки рисунков в текст** в текстовом процессоре Word предусмотрена специальная операция в строке меню *Вставка.* Можно вставить в текст стандартные картинки текстового процессора Word:

• установите текстовый курсор в один из пустых абзацев;

• выберите команду меню *Вставка + Рисунок,*

• в появившемся диалоговом окне выберите файл с расширением .wmf (эти файлы находятся в каталоге Clipart);

• если хотите просмотреть содержимое файла прямо в диалоговом окне, установите опцию *Просмотр;*

• закончите диалог с помощью кнопки ОК;

• сделайте преобразование рисунка.

**Создание собственных рисунков**, выполненных в графическом редакторе Paintbrush:

• нарисуйте в графическом редакторе Paintbrush картинку, которую хотите вставить в текст;

• с помощью инструмента *Ножницы* отметьте картинку и скопируйте ее в *Буфер Обмена;*

• с помощью комбинации клавиш Alt + Tab перейдите в текстовый процессор Word (если он у вас не загружен, то перейдите в Диспетчер Программ и запустите его);

• выберите команду меню *Правка + Вставить* или соответствующую кнопку в пиктографическом меню;

• сделайте преобразования рисунка.

**Стиль оформления** - это набор описания различных форматов абзацев. Каждый формат абзаца имеет свое имя и состоит из последовательности инструкций форматирования. Стиль хранится совместно с документом.

Создание стиля оформления возможно следующим образом:

• выберите команду меню *Формат + Стиль;*

• в диалоговом окне создания стиля в поле *Имя Стиля* введите имя стиля;

• нажмите кнопку *Определить;*

• для определения форматов в группе *Оформление* следует выбрать желаемую кнопку объекта форматирования *(Символ. ., Абзац..., Табуляция..., Обрамление..., Кадр .. или Язык ..);*

• завершение установок в этих диалоговых окнах производится кнопкой ОК, после этого вы опять возвращаетесь в окно создания стиля;

• после создания стиля надо щелкнуть указателем мыши по кнопке *Заменить;*

• в завершение создания стиля надо нажать кнопку *Применить.*

Если у вас уже есть готовый стиль, то для того чтобы применить его к вашему отмеченному тексту, на панели инструментов *форматирования* нужно выбрать первое слева комбинированное окно и выбрать нужный стиль

С помощью текстового процессора Word можно создавать таблицы путем преобразования текста или создания пустой таблицы с последующим заполнением ячеек.

**Создание таблицы путем преобразования текста**

•введите текст, разделяя будущие ячейки таблицы с помощью табуляции (если текст уже введен, разделите его на ячейки с помощью табуляции);

• выделите текст, который необходимо преобразовать в таблицу;

• включите отображение сетки таблицы с помощью команды меню *Таблица + Линии Сетки,*

• выберите команду меню *Таблица + Преобразовать текст;*

• нажмите кнопку ОК.

**Создание пустой таблицы с последующим заполнением ячеек:**

1-й способ :

• установите текстовый курсор в том месте, где хотите расположить таблицу;

• выберите команду меню *Таблица + Вставить Таблицу;*

• в появившемся диалоговом окне окажите необходимое число столбцов и строк;

2-й способ:

• установите текстовый курсор в том месте, где хотите расположить таблицу;

• в строке пиктографического меню нажмите на кнопку с изображением таблицы, в появившейся сетке выделите необходимое число строк и столбцов.

Разделительные линии между колонками и строками в таблице при печати на бумаге отсутствуют. На экране вы видите только пунктирные линии, служащие для ориентировки, и на печать не выводимые. Для того чтобы эти линии были видны при выводе на печать надо

• выделить таблицу;

• выбрать команду меню *Формат + Обрамление',*

• в появившемся диалоговом окне указать тип *Сетка* и задать толщину линии;

• нажать ОК.

В текстовом процессоре Word у пользователя имеется возможность **одновременно открывать вплоть до девяти окон** документов. Независимо от других документов вы можете производить редактирование активного документа.

Окно, находящееся на первом плане, является активным. Для того чтобы перейти в другое окно, надо выбрать команду меню *Окно + Название окна,* к которому хотите перейти.

Для того чтобы закрыть окно, надо выбрать команду меню *Файл + Закрыть.*

Для перемещения текста из одного окна в другое надо

• загрузить файл с документом, в который хотите переместить текст;

• загрузить файл с документом, из которого хотите переместить текст;

• пометить весь текст, который хотите переместить, и скопировать его в буфер обмена *(Правка + Копировать);*

• перейти в то окно, где загружен документ, в который необходимо переместить текст;

• поставить курсор в то место, в котором должен быть этот текст, и выбрать команду меню *Правка + Вставить.*

Мы описали лишь малую часть возможностей текстового процессора Word, но освоение их поможет производить практически полезную работу с этой сложной программой и понять ее стиль, после чего освоение остальных возможностей не составит труда.

### *Настольная издательская система PageMaker*

PageMaker (PM) - одна из самых мощных и популярных у профессионалов издательских систем. С ее помощью можно подготовить оригинал-макет толстой книги, включающей иллюстрации, формулы, таблицы, другие сложные элементы. Последние версии программы полностью совместимы по интерфейсу с Windows и допускают импорт и конвертацию файлов из любых Windows-приложений; возможен и импорт текстовых файлов. Разработчик современных версий программы PM - фирма «Adobe».

При подготовке к изданию книг PageMaker, предоставляет значительно больше возможностей, чем текстовый процессор Word. Это связано с наличием средств управления проектами, включающими создание шаблонов и стилей, оглавлений и предметных указателей и т.д. К примеру, главы 1 - 6 данной книги подготовлены к изданию в PM.

Для вызова диалоговой панели выбора (установки) параметров работы или выполнения команд необходимо установить курсор мыши на пункт меню или команду и щелкнуть кнопкой или нажать клавиши Alt - F10 и клавишу, соответствующую выделенной букве. Выполнение команд происходит после подтверждения правильности установки всех параметров (выбора значений) активизацией экранной кнопки ОК или нажатием клавиши <Enter>.

Перед началом работы установите принтер и единицу измерения. После этого начните верстку, установив поля и другие параметры, общие для всего издания. Не забудьте выбрать основной шрифт и провести вспомогательные линии.

Изменение масштаба выводимого на экран изображения осуществляется в пункте Page главного меню программы.

Выбрать режим работы с текстом или иллюстрациями можно активизацией соответствующей пиктограммы с помощью мыши или клавиатуры.

Перед началом работы с текстом (иллюстрациями) его необходимо активизировать. Для этого выберите в панели *Инструменты* пиктограмму и щелкните кнопкой мыши по выделенному фрагменту. Можно активизировать одновременно несколько объектов, последовательно выбирая их с помощью мыши при нажатой клавише Shift или отмечая на полосе прямоугольную область, в которой они расположены. Для одновременной активизации всех объектов на полосе нажмите клавиши Ctrl + A или дайте команду *Edit * Select All.*

Для перемещения по тексту издания необходимо переместить указатель мыши в нужное место и щелкнуть левой кнопкой.

Для выделения фрагмента текста выберите соответствующую пиктограмму в панели *Инструменты.* Выделение всего текста издания осуществляется командой *Edit * Select All* или нажатием клавиш Ctrl + A. Выделить фрагмент можно перемещением указателя мыши по тексту с нажа-

той кнопкой или с помощью клавиатуры.

| | |
|---|---|
| Помощь: | *Help* или F1 |
| Выход из программы: | *File* * *Exit* или Alt + F4 |

### *Настольная издательская система TeX*

Описанные выше издательские системы плохо приспособлены к набору и подготовке к печати текстов математического характера, с большим числом сложных формул и графиков, специальных математических символов. Для работы с такими материалами, подготовки соответствующих статей и книг американский математик и теоретик программирования Д. Кнут создал издательскую систему TeX.

В отличие от описанных выше систем, TeX не является системой типа «WYSIWYG» - «что набираю, то и вижу на экране». Если набор текста в TeX элементарен, то набор формул и таблиц является, по-существу, написанием программы на специальном макроязыке, что вознаграждается высоким качеством соответствующих текстов**.**

### *Контрольные вопросы*

1. Какие возможности предоставляют текстовые редакторы?
2. Какие минимальные требования предъявляются к техническим характеристикам компьютера и его периферии для развертывания текстовых редакторов различных уровней?
3. Какие функциональные разделы допускают издательские системы?
4. Подготовьте реферат «История вычислительной техники» с помощью редактора Лексикон; затем, используя правила художественного и технического редактирования, разработайте несколько его вариантов в виде оригинал-макетов с помощью издательской системы Word.

## § 5. СИСТЕМЫ КОМПЬЮТЕРНОЙ ГРАФИКИ

### 5.1. ПРИНЦИПЫ ФОРМИРОВАНИЯ ИЗОБРАЖЕНИЙ НА ЭКРАНЕ

Существует два способа реализации построения изображений на экране дисплея - **векторный** (функциональный) и **растровый**. В первом случае электронный луч поочередно рисует на экране различные знаки - элементы изображения. На современных персональных компьютерах чаще используется растровый способ изображения графической информации, в котором изображение представлено прямоугольной матрицей точек (**пикселов**), имеющих свой цвет из заданного набора цветов (**палитры**). Графический режим осуществляет видеоадаптер, управляющий работой электронной трубки и видеопамятью, в которой запоминается текущее изображение. Адаптер обеспечивает регулярное отображение видеопамяти на экране монитора.

**Растровое изображение** - это совокупность разноцветных точек. Координаты точек определяются декартовой (прямоугольной) системой с началом координат, как правило, в левом верхнем углу экрана. Абсцисса $x$ точки увеличивается слева направо, ордината $y$ - сверху вниз. Таким образом, любая графическая операция сводится к работе с отдельными точками экрана монитора - пикселами. Существуют специальные графические библиотеки программ, которые предназначены для изображения более сложных объектов, являющихся объединением группы пикселов: наиболее употребимые линии, геометрические фигуры, шрифты и т.п.

Большинство языков программирования имеют свои стандартные графические библиотеки. Так, у Бейсика графические команды являются встроенными; системы программирования Турбо-Паскаль содержат графическую библиотеку (модуль Graph, tpu), имеющую в своем составе процедуры и функции обработки простейших графических образов. Тем не менее, многие программисты и разработчики программ предпочитают создавать свои библиотеки графических подпрограмм в соответствии со спецификой своей работы.

В последние годы возрос интерес со стороны пользователей к специальным инструментальным программам машинной графики: графическим редакторам, издательским системам и т.п. В них предоставляется удобный интерфейс для пользователей, автоматизируется большое количество разнообразных действий с графической информацией - от построения простейших рисунков

до создания мультипликационных (анимационных) роликов.

Вывод изображения на экран дисплея и разнообразные действия с ним требуют геометрической грамотности, интуиции. Моделирование образной информации на экране дисплея развивает наблюдательность, пространственное воображение, геометрическую интуицию, конструкторские и изобретательские навыки.

## 5.2. ИЗОБРАЗИТЕЛЬНАЯ ГРАФИКА

При работе с художественной графикой, включая компьютерную графику, необходимо знать следующие понятия.

**Композиция** - строение, соотношение и взаимное расположение частей, сосредоточие идейно-творческого начала, позволяющего автору произведения искусства целенаправленно организовывать главное и второстепенное и добиваться максимальной выразительности содержания и формы в их образном единстве.

Законы композиции:

1) *закон цельности:* во-первых, наличие конструктивной идеи, объединяющей в единое целое все компоненты произведения; во-вторых, связь и взаимная согласованность всех элементов композиции; в-третьих, неповторимость элементов композиции;

2) *закон контрастов:* контраст света и тени определяет форму предметов; контрасты величин, построения сюжета определяют композицию. Правила композиции:

1) *передача ритма:* ритм в удачной композиции одновременно расчленяет компоненты произведения и объединяет их;

2) *композиционный центр:* выделяется объемом, освещенностью и другими средствами, действующими в соответствии с основными законами композиции.

**Мера** - характеризует общие принципы строения, целостность предмета, лежит в основе ритма, гармонии, ансамбля в архитектуре.

**Симметрия** - одинаковость в расположении частей чего-нибудь по противоположным сторонам от точки, прямой или плоскости. Симметрия в композиции создается уравновешенностью ее частей по массам, тону, цвету и форме. Симметрия (геометрическая) - свойство геометрических фигур при котором каждая пара соответственных точек лежит на одном перпендикуляре к данной плоскости по разные стороны и на одинаковом расстоянии от нее.

**Пропорция -** определенное соотношение частей между собой, соразмерность.

**Ритм** - равномерное чередование каких-нибудь элементов.

**Гармония** - соразмерность частей, слияние различных компонентов объекта в единое органичное целое. В истории эстетики гармония - существенная характеристика прекрасного.

**Перспектива** - искусство изображать на плоскости трехмерное пространство в соответствии с тем кажущимся изменением величины, очертаний, четкости предметов, которое обусловлено степенью отдаленности их от точки наблюдения.

При моделировании графических объектов на экране дисплея используют разные методы и способы представления изображений. Можно выделить два принципа моделирования - случайный и детерминированный. На их основе строятся технологии в изобразительной графике, в которой условно можно выделить три направления: художественное, иллюстративное и демонстрационное.

Объектами художественной графики выступают различные узоры, шрифты и другие изображения. При работе с изображениями широко используют простые мотивы и разнообразный геометрический материал. В частности, простые геометрические фигуры в различных сочетаниях и способах размещения (вложения, вращения, симметрии) используются в «живых картинках» для получения муарового эффекта. Удачное сочетание случайного и упорядоченного в любой пропорции с технологиями расположения графических объектов позволяет создавать художественный дизайн.

Самые простые узоры - бордюры - представляют бесконечный ряд равных плоских фигур, расположенных друг за другом таким образом, что элементарная конечная фигура переносится вдоль одного измерения бесконечно. Помимо элементарного мотива для получения бордюра необходимо выбрать группу симметрии бордюра и задать конкретные образующие этой группы из следующего набора движений плоскости:

- параллельный перенос;
- центральная симметрия;

• осевая симметрия;

• скользящая симметрия.

Для бордюр существует четыре абстрактные группы симметрии, которые определяют семь типов симметрии бордюр:

1) один параллельный перенос;

2) одна скользящая симметрия;

3) две осевые симметрии;

4) две центральные симметрии;

5) одна осевая и одна центральная симметрия;

6) один параллельный перенос и одна осевая симметрия;

7) три осевые симметрии.

Существует несколько способов построения бордюр по заданному элементарному мотиву и системе образующих ее группы симметрии. Рассмотрим наиболее простые из **них.** Первый способ основан на простом переборе элементов группы - на элементарный мотив действуют поочередно всеми элементами группы симметрии бордюра. Множество полученных образов и будет представлять собой требуемый бордюр. Этот способ неудобен тем, что на каждом этапе необходимо представить очередной элемент группы в виде композиции подходящих степеней образующих.

Второй способ более предпочтителен. На элементарный мотив действуют одним из образующих. Затем на новую фигуру, составленную из исходного мотива и его образа, действуют любым образующим, который не отображает эту фигуру на себя. Получается фрагмент бордюра, состоящий из предыдущей фигуры и ее образа.

Продолжая этот процесс можно построить любую конечную часть бордюра. Преимущество этого способа заключается в том, что на каждом этапе мы будем иметь дело с конкретной симметрией, с нахождением образа конкретной фигуры.

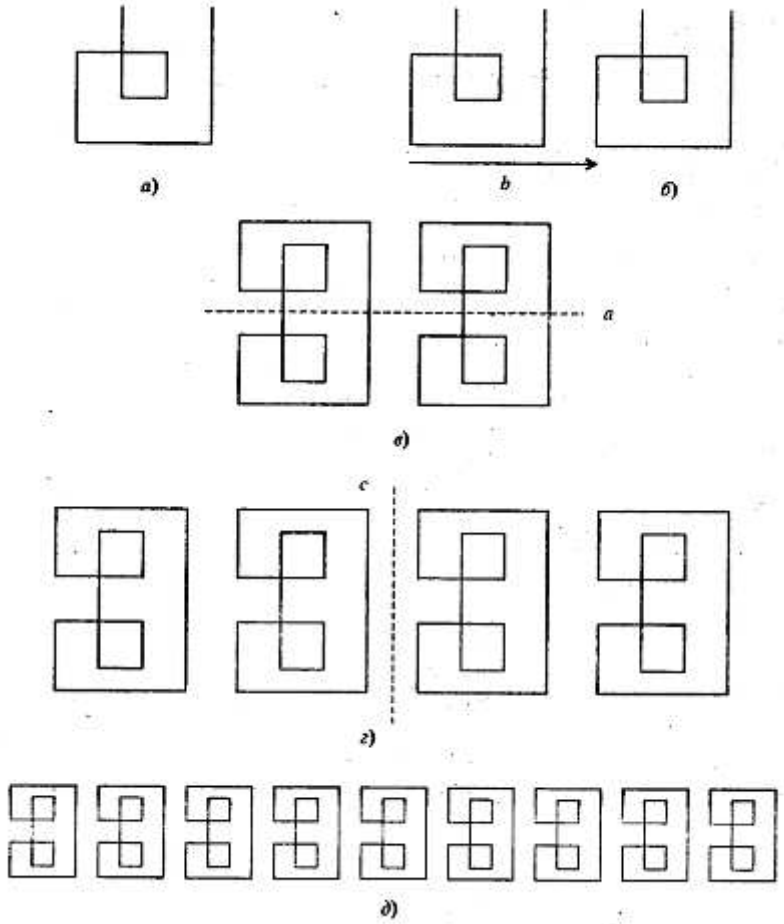



*Рис. 2.16.* Схема построения бордюра

Рассмотрим, например, процесс построения бордюр с помощью образующих шестого типа симметрии. Создадим элементарный мотив узора (рис. 2.16, *а)* и определим образующие: параллельный перенос зададим вектором **b,** (рис. 2.16, *б)* осевую симметрию прямой *а* (рис. 2.16, *в)*

Процесс получения бордюра с помощью заданных образующих вторым из описанных выше способом проиллюстрирован поэтапно. На элементарный мотив действуем параллельным перено-

сом на вектор b, см. схему *б.* Получим фигуру, состоящую из элементарного мотива и его образа. На новую фигуру действуем осевой симметрией с осью *а.* Получим фигуру, изображенную на схеме *в,* на которую действуем параллельным переносом 2b. В результате получим новую фигуру, см. схему *г.* Продолжая этот процесс, получаем фрагмент бордюра нужной длины.

Бордюры используются для окантовки обоев, ковров; для настенной росписи, украшающей здания, подземные переходы; в виде металлических решеток для ограждения парков, мостов и садов и пр.

Более зрелищны, привлекательны и интересны по построению орнаменты. Любой орнамент получается переносом узора с помощью двух параллельных переносов, заданных неколлинеарными векторами. Для любого орнамента можно найти сетку, узлы которой составляют вполне определенную систему равных точек орнамента. Различают пять типов плоских решеток: квадратная, прямоугольная, гексагональная, ромбическая и косая. Тип плоской решетки определяет характер переносной симметрии данного орнамента.

В простейшем случае орнамент характеризуется только переносной симметрией. Для построения такого орнамента надо выбрать соответствующую плоскую решетку, заполнить элементарную ячейку решетки определенным рисунком и затем многократно повторить этот рисунок за счет переносов ячейки без изменения ее ориентации. Для построения более сложных по композиции орнаментов рисунок элементарной ячейки заполняется из основного (элементарного) мотива с помощью образующих этого типа симметрии плоских орнаментов. Существует 17 типов симметрии плоских орнаментов, которые определяются следующим образом:

1) два параллельных переноса:
2) три центральных симметрии;
3) две осевые симметрии и параллельный перенос;
4) две скользящие симметрии с параллельными осями;
5) осевая и скользящая симметрии с параллельными осями;
6) симметрия относительно четырех сторон прямоугольника;
7) одна осевая и две центральные симметрии;
8) две скользящие симметрии с перпендикулярными осями;
9) две осевые симметрии с перпендикулярными осями и одна центральная симметрия;
10). центральная симметрия и вращение на 90°;
11) симметрия относительно трех сторон прямоугольного равнобедренного треугольника;
12) осевая симметрия и вращение на 90°;
13) два вращения на 120°;
14) осевая симметрия и вращение на 120°;
15) симметрия относительно равностороннего треугольника;
16) центральная симметрия и вращение на 120°;
17) симметрия относительно трех сторон прямоугольного треугольника с углом 30°.

Несколько слов о частных случаях орнаментов. Бесконечная плоская фигура $\Phi$ называется плоским орнаментом, если выполняются следующие условия: 1) среди перемещений, отображающих $\Phi$ на себя, существуют неколлинеарные параллельные переносы; 2) среди всех векторов (параллельных переносов), отображающих $\Phi$ на себя, существует вектор наименьшей длины. Фигуру называют линейным орнаментом, если плоская фигура отображается сама на себя при параллельных переносах только одного направления (и противоположного ему), причем среди этих переносов существует перенос наименьшей длины.

Большой интерес представляют паркеты. Паркетом называется разбиение плоскости на многоугольники, при котором каждые два многоугольника либо не пересекаются, либо имеют ровно одну общую вершину, либо имеют общую сторону, причем объединение сторон всех многоугольников является плоским орнаментом. Паркет называется правильным, если все многоугольники разбиения правильные (возможно с различным числом сторон) и любую величину паркета можно перевести в любую другую величину некоторым перемещением, отображающим весь паркет на себя.

Интересны орнаменты, заполненные одинаковыми фигурками без промежутков. Фигурку такого орнамента можно создать с помощью геометрических преобразовании на основе гексагональной сетки.

Процесс моделирования орнаментов на экране компьютера аналогичен моделированию

бордюр: после получения повторяющейся фигурки из элементарного мотива с помощью образующих данного орнамента вывод на экран происходит циклическим изменением координат в двух направлениях. При моделировании орнаментов на основе различных плоских решеток без «заполнения» их элементарной ячейки получаются паркеты. Для осуществления компьютерного моделирования графического объекта выбирают подходящее программное инструментальное средство - графический редактор (систему), например, PaintBrash, CorelDraw и т.п., в котором допустимы все возможности, необходимые для описанной работы. Иногда бывает полезным осуществить построение графического образа программированием (на одном из языков программирования) с использованием графических библиотек.

Узор на плоскости получается из элементарного мотива с помощью образующих элементов его группы симметрии. Для бордюр существует 7 различных наборов образующих его групп симметрии, для орнаментов - 17. Чтобы построить узор на компьютере, необходимо

1) создать элементарный мотив узора (в графическом редакторе, на языке программирования, сканированием с листа бумаги);

2) выбрать и задать образующие;

3) построить образы каждой точки элементарного мотива по его образующим, т.е. получить из элементарного мотива изображение конечной повторяющейся фигуры;

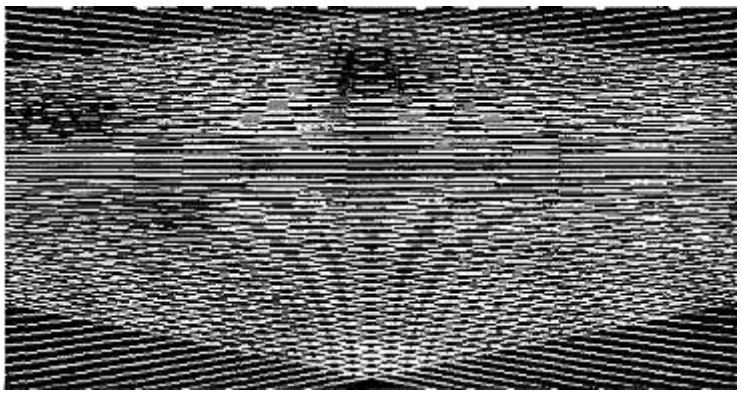
4) средствами имеющегося компьютера и программного обеспечения размножить получившийся «шаблон» (фигурку), циклически меняя координаты в одном или в двух направлениях.

В художественной графике часто обращаются к узорам, получаемым в результате «интерференции» при наложении каких-либо семейств кривых. При пересечении, например, двух семейств прямых под заданным углом получаются «муаровые» эффекты, рис. 2.17.

Моделирование муаровых узоров представляет простой, но занимательный процесс, в котором основная роль принадлежит зрительному восприятию исследователя, который, меняя параметры семейств кривых и их способы пересечения, добивается наилучшей в смысле гармонии красоты узоров.

Используя принцип муаровых узоров, можно моделировать на экране дисплея фейерверки. Простейший одиночный фейерверк можно изобразить по следующему принципу. Из фиксированной точки (*x*, *y)* строят отрезки прямых в точку, задаваемую случайным образом и случайным цветом. Возникающие одиночные фейерверки, пересекаясь друг с другом, создают красивые картинки.

Узоры можно моделировать с помощью простых мотивов, в частности, штрихованного угла. Штрихованный угол задается координатами трех точек и изображается семейством отрезков, соединяющих точки разбиения сторон угла с одинаковыми номерами. Подобные мотивы удобно оформлять в виде подпрограммы, а затем использовать при моделировании различных изображений.



. *Рис. 2.17.* Муаровый рор

Сочетание случайного и упорядоченного в любой пропорции при изображении графических объектов дает эффект. Например, на основе простого мотива - угла, вершина которого находится на горизонтальной прямой, - можно изобразить «лес». При упорядоченном размещении «деревьев» имеем «посадку».

Подобные технологии широко используют в иллюстративной графике. В настоящее время

**иллюстративная графика, в** первую очередь, связана с изображением графического материала в издательских системах.

Иллюстративный материал - схемы, эскизы, географические карты, чертежи и др. -можно создавать различными графическими редакторами, системами. Здесь важна легкость и быстрота формирования и преобразования графических изображений для тех или иных приложений. В последнее время большой интерес вызывает иллюстративный материал, представленный в демонстрационной, динамической форме.

**Демонстрационная графика** связана с динамическими объектами. В технологии изображения динамических объектов используют три основных способа: рисование -стирание, смену кадров (страниц), динамические образы.

Достаточно просто организовать перемещение фрагмента рисунка на экране. Для этого надо

- создать этот фрагмент в нужном месте экрана;
- стереть фрагмент, рисуя его цветом фона или используя процедуру очистки экрана;
- снова нарисовать фрагмент в другом месте экрана, и так далее.

Еще один способ организации движения объектов на экране, широко применяющийся в компьютерных играх, связан с использованием нескольких экранных страниц. В любой момент времени одну из страниц можно сделать видимой и посмотреть ее содержимое на экране. Визуальная страница обычно пассивна, т.е. на ней нельзя выполнять графические процедуры. Другую страницу можно объявить активной. Активная страница невидима для пользователя, но на ней можно подготовить другой рисунок. Меняя страницы ролями, можно создать мультипликационный сюжет

В обучающих программах и в компьютерных играх часто используют динамические образы (в том числе спрайты). При этом используют динамическую память для хранения прямоугольных фрагментов рисунков и последующего вывода их в нужное место экрана.

## 5.3. ГРАФИЧЕСКИЕ РЕДАКТОРЫ

Создать объекты иллюстративной графики, включая динамические ролики, можно средствами программирования, а также графическими редакторами и системами. Рассмотрим два из них.

### Редактор Paint

Графический редактор Paint, входящий в комплект стандартных программ MS Windows 95, позволяет, используя манипулятор «мышь», выполнять черно-белые и цветные рисунки, обрамлять их текстом, выводить на печать. В Paint можно работать с фрагментами графических изображений: копировать, перемещать, поворачивать, изменять размеры, записывать на диск и считывать с диска. С помощью Paint можно обрабатывать графические изображения, а также считывать и записывать в файл полностью или частично изображение с дисплея, если монитор работает в графическом режиме.

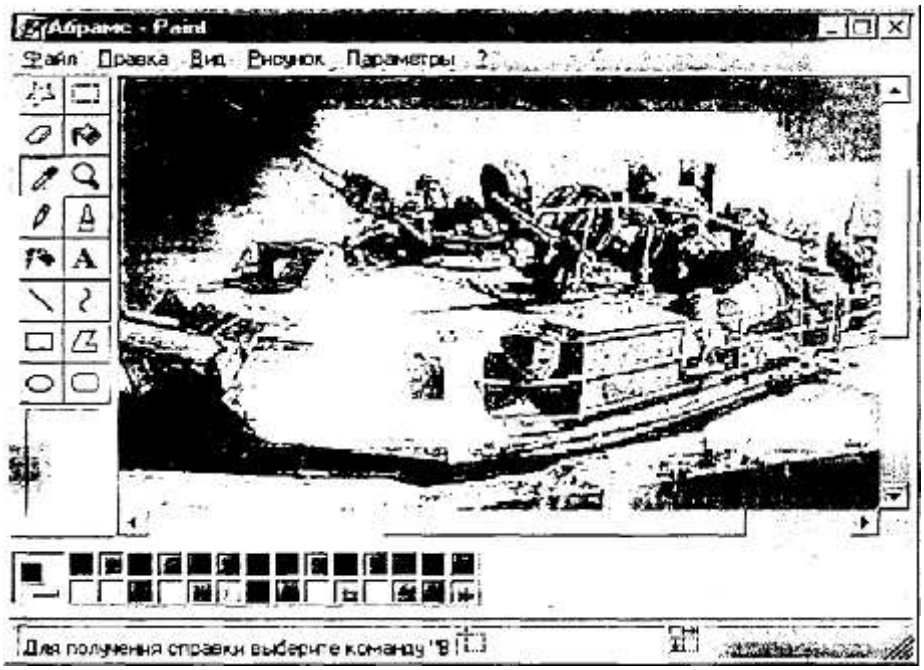



*Рис. 2.18.* Рабочее окно программы Paint

После загрузки пакета появляется рабочий экран редактора (рис. 2.18). Большую часть экрана занимает рабочее поле, окрашенное в фоновый цвет. Над рабочим полем - меню, позволяющее выполнять команды редактирования. Слева от рабочего поля расположена панель инструментов, на которой высвечен инструмент, в данный момент являющийся рабочим. Под рабочим полем находится палитра. С левого края палитры показаны два вложенных квадрата, внутренний из которых окрашен в рабочий цвет, а внешний - в фоновый. В левом нижнем углу экрана выводится калибровочная шкала, которая позволяет устанавливать ширину рабочего инструмента (кисти, резинки и т.д.). Установленная в данный момент ширина инструмента отмечена стрелкой. Вдоль нижнего правого и правого края рабочего поля находятся линейки прокрутки для перемещения рабочего поля по картинке, если размеры картинки больше размеров рабочего поля.

Общие правила работы с редактором таковы. Для выбора (установки) параметров работы и выполнения команд в Paintbrush необходимо поместить указатель мыши на пункт меню и щелкнуть левой кнопкой. Выход из программы: *File/Quit.*

Программа *Freeze* предназначена для сохранения выводимого на экран изображения в графическом PCX-файле с одновременным сохранением оформления экрана, возможностью последующего редактирования данного изображения и вставки его в качестве иллюстрации в текстовые редакторы и настольные издательские системы. Установка программы происходит автоматически при запуске Paintbrush.

**Редактор CorelDraw**

Начиная работать с графическим редактором CorelDraw, мы прежде всего видим его рабочее окно, рис. 2.19.

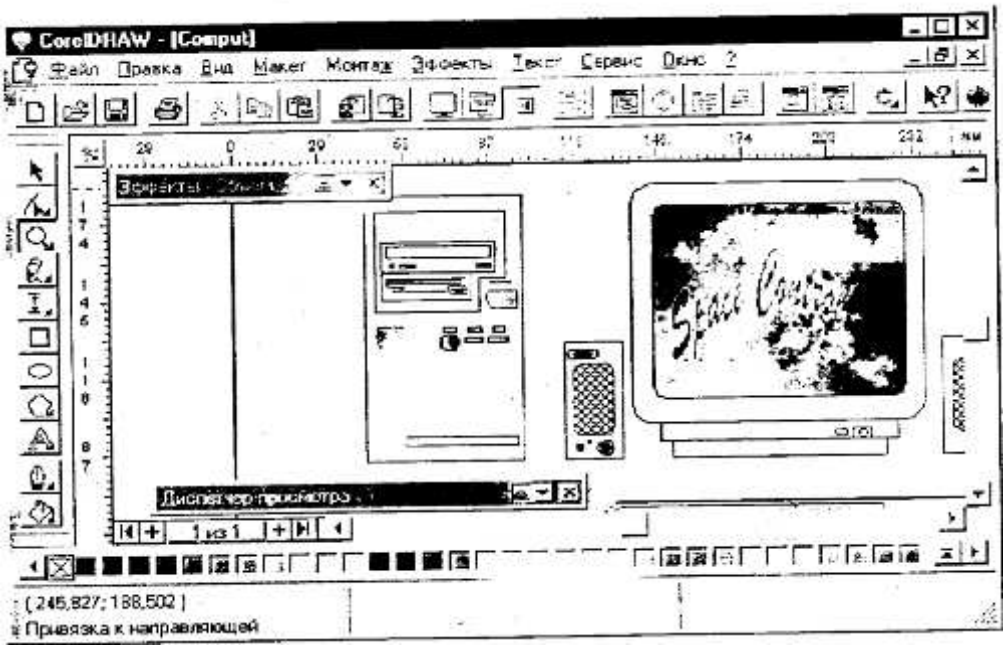



*Рис. 2.19.* Рабочее окно программы CorelDraw

Чтобы активизировать **меню выбора** (установки) **параметров** работы или выполнения команд, установите указатель мыши на пункт меню и щелкните левой кнопкой или нажмите клавишу Alt и клавишу, соответствующую выделенной букве. В некоторых случаях для облегчения выполнения наиболее часто употребляемых команд ввод их с клавиатуры осуществляется нажатием комбинации клавиш или определенной клавиши. Выполнение команд происходит после подтверждения правильности установки всех параметров или выбора значений активизацией экранной кнопки ОК в диалоговой панели или нажатием клавиши Enter.

Выход из меню - переместить указатель мыши за пределы меню и щелкнуть кнопкой или нажать клавишу Esc.

Выход из программы: *File/Quit* или Alt + F4.

Для вызова **диалоговой панели** выбора (установки) параметров работы или выполнения команд необходимо установить указатель мыши на пункт меню или команду и щелкнуть кнопкой или нажать клавишу Alt и клавишу, соответствующую выделенной букве. Выполнение команд происходит после подтверждения правильности установки всех параметров (выбора значений) активизацией экранной кнопки ОК или нажатием клавиши Enter.

Для **редактирования рисунка** следует активизировать пиктограмму с помощью мыши или нажатием клавиши *Пробел,* переместить указатель на любую точку контура рисунка и щелкнуть кнопкой. Выбранный рисунок будет окружен восемью квадратами черного цвета. Можно выделить одновременно несколько объектов, последовательно выбирая их с помощью мыши при нажатой клавише Shift или отмечая на экране прямоугольную область, в которой они расположены. Для одновременного выбора всех рисунков на экране необходимо активизировать пиктограмму, переместить указатель мыши в один из углов выбираемого прямоугольного контура, нажать кнопку и, не отпуская ее, переместить указатель в противоположный угол и отпустить кнопку. Контур будет изображен штриховой линией. Для отмены выбора - переместить указатель мыши за контур и щелкнуть кнопкой.

Для изменения масштаба выводимого на экран рисунка необходимо активизировать соответствующую пиктограмму.

**Работа с текстом** начинается с активизации пиктограммы текста. Для перемещения по тексту в диалоговой панели *Text* необходимо переместить указатель мыши и щелкнуть левой кнопкой. Однако в программе для перемещения предусмотрены также специальные клавиши.

Для выделения фрагмента текста необходимо активизировать пиктограмму с помощью мыши или клавиши *Пробел.* Переместите указатель мыши в один из углов выбираемого прямоугольного контура, нажмите кнопку К, не отпуская ее, переместите указатель в противоположный угол, отпустите кнопку. Весь контур будет окружен штриховой линией. Для выделения отдельного символа 'необходимо активизировать пиктограмму, переместить указатель мыши в один из углов выбираемого прямоугольного контура, нажать кнопку и, не отпуская ее, переместить указа-

тель в противоположный угол. Выбранный объект не будет окружен рамкой из квадратов, однако на нем будут выделены все узловые точки. Выбор нескольких символов осуществляется аналогично при нажатой клавише Shift.

Прежде чем начать оформление ранее введенного текста, его необходимо выделить одним из выше описанных способов.

## 5.4. ДЕЛОВАЯ ГРАФИКА

Одним из первых приложений компьютерной графики стало отображение данных экономических расчетов.

Графические представления расчетных и статистических данных удобно представлять в виде схем, диаграмм, гистограмм и графиков. Различают следующие их виды:

**гистограмма** - группа столбцов, пропорциональных по высоте определенным числовым значениям;

**круговая диаграмма** - секторы круга, углы которых пропорциональны элементам данных;

**линейный график** - отображение исходных величин в виде точек, соединенных отрезками прямых линий;

**временная диаграмма** - последовательность операций или процессов определенной длительности (измерение динамических процессов);

**структурная схема** - представление сложных объектов в виде дерева или графа;

**круговая гистограмма -** представление относительных величин объектов, которым на изображении сопоставляются размеры и расположение кругов в прямоугольной системе координат.

Из числа средств прикладного программного обеспечения общего назначения графическое представление данных лучше всего развито в электронных таблицах и в СУБД.

Одним из первых практических применений машинной графики было автоматическое построение графиков функции в различных системах координат. Обычно графики функций строят в декартовых координатах (в прямоугольной системе, рис. 2.20).

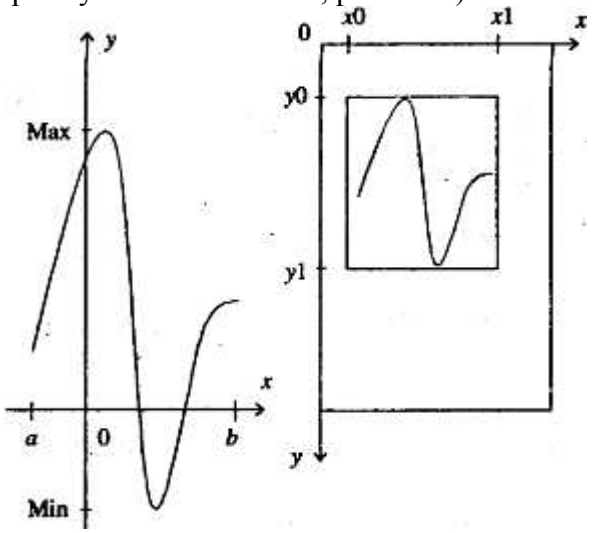



*Рис. 2.20.* Построение на экране графиков функций (в декартовой системе координат)

В общем виде алгоритм построения графика заданной функции $y = f(x)$ на отрезке $[a, b]$ заключается в следующем.

1. Определяем область значений функции, для чего найдем максимальное по модулю значение функции на заданном отрезке $[a, b]$, $m = \max(\text{abs}(f(x)))$ для всех $x$ из $[a, b]$.

Примем для удобства, что минимальное значение функции на отрезке совпадает с максимальным, но с обратным знаком. Таким образом, область значений функции лежит в интервале $[-m, m]$.

Поиск максимума можно осуществить разными способами, например, табулируя функцию $f(x)$ на отрезке с разбиением на $n$ частей и определяя максимальное значение в массиве чисел $Y_i = f(x_i)$, где $x_i = a + i\text{-}(b - a) / n$, для $i = 0,... n$.

2. Задаем координаты окна $x0$, $y0$, $x1$, $y1$ на графическом дисплее, в котором будем строить

график функции.

3. Формулы преобразования координат ***x,*** *y* точек прямоугольника *[a, b]·[-m, m]* обычной декартовой системы в соответствующие координаты *u, v* окна *[x0,x1] · [y0,y1]* графического экрана можно задать в следующем виде :

$$u = x0 + (x - a)(x\ 1 - x0)/(b - a),$$
$$v = (y0 + y1) / 2 – f(x) (y1 – y0) / (2m).$$

Тогда автоматическое построение графика функции на экране дисплея осуществляется путем установки точек $(u_i, v_i)$, соответствующих точкам $(x_i\ f(x_i))$, выбранным в декартовой системе. Часто бывает полезно соединять полученные точки отрезками или специальными линиями, что программы могут делать (или не делать) по пожеланию пользователя.

4. Далее можно оформить график, нарисовав оси координат, нанести масштабные сетки, вывести соответствующие обозначения и комментарии. Оси координат на графическом экране в заданном окне легко построить, вычислив экранные координаты начала выбранной декартовой системы *{xv,yv):*

$$xv = x0 - a(x1 - x0)/(b - a),$$
$$yv=(y0+y1) / 2.$$

## 5.5. ИНЖЕНЕРНАЯ ГРАФИКА

Компьютеризацию чертежных и конструкторских работ проводят давно и в настоящее время используют различные системы автоматизации проектных работ (САПР). Аббревиатуру САПР впервые использовал основоположник этого научного направления Айвен Сазерленд в своих лекциях, прочитанных в Массачусетском технологическом институте в начале 60-х годов. Фактически инженеры применяли компьютеры для решения сложных задач проектирования еще в эпоху ранних послевоенных моделей универсальных компьютеров, а первые образцы специализированного оборудования САПР были созданы уже в середине 50-х годов. Однако широкое распространение САПР обусловлено появлением микропроцессорной техники, предоставившей возможности создавать, модифицировать и обрабатывать сложные графические изображения на экране монитора.

В настоящее время САПР обозначает аппаратно-программный комплекс, поддерживающий процесс проектирования с использованием специальных средств машинной графики, поддерживаемых пакетами программного обеспечения, для решения задач, связанных с проектной деятельностью. В совокупности развитая САПР представляет собой специализированную информационную систему. Сфера применения САПР охватывает такие разные области приложения, как архитектура, гражданское строительство, картография, медицина, геофизика, разработка моделей одежды, издательское дело, реклама.

Полная система САПР состоит из компонентов аппаратного и программного обеспечения. Общими компонентами аппаратного обеспечения системы САПР являются ЦП (центральный процессор), несколько рабочих станций, разделяемая между рабочими станциями периферия.

Состав типичной системы САПР:

дисплей (графический и алфавитно-цифровой);

процессор;

клавиатура;

устройство управления курсором (мышь, дигитайзер);

электронный командный планшет;

принтер.

Одним из наиболее давних и популярных средств автоматизированного проектирования является система АВТОКАД (AutoCad). АВТОКАД не является проблемно-ориентированной системой, т.е. не содержит специализированных баз данных, экспертных систем и многого из того, что входит в состав специализированной интеллектуальной САПР. АВТОКАД - достаточно простая универсальная система. Ее возможности таковы:

- развитая система экранных меню;

- высокая точность графической информации;
- разбивка информации (расслоение);
- прочерчиванне на дисплее координатной сетки;
- средство захвата графических объектов;
- мощное редактирование;
- отображение параметров графических характеристик;
- полуавтоматическая и автоматическая простановка размеров;
- штриховка;
- работа с блоками.

После запуска АВТОКАДа на текстовом экране появляется главное меню:
О - выход;
1 - создание нового чертежа;
2-редактирование существующего чертежа;
3 - вывод на плоттер (графопостроитель);
4 - вывод на принтер;
5-конфигурация;
б - файловые утилиты;
7-шрифты;
8 -стыковка со старыми версиями.

Режимы экранного меню:

| | |
|---|---|
| AUTOCAD | - выход в головное меню; |
| * * * * | - режим объектного захвата; |
| BLOCKS | -работа с блоками; |
| DISPLAY | - работа с изображением без его изменения; |
| SETTINGS | -настройка; |
| DIM | - обезразмеривание; |
| EDIT | -редактирование; |
| DRAW | -рисование; |
| LAYER | -работа со слоями; |
| INQUIRY | -справки о примитивах; |
| UTILTTS | - выход в ДОС, запись чертежей в разных форматах; |
| PLOT | -получение твердой копии и т.д. |

В режиме DRAW (рисуй) имеется возможность строить графические примитивы и проводить с их помощью синтез изображений. Например, здесь существует восемь способов рисования дуг:

- по трем точкам на дуге /3 points/;
- по начальной точке, центру и длине хорды /S, C, L/;
- по начальной точке, центру и заключенному углу /S, C, A/;
- по начальной точке, конечной точке и радиусу /S, E, R/;
- по начальной точке, конечной точке и заключенному углу /S, E, A/;
- по начальной точке, конечной точке и исходному направлению /S, E, D/;
- по продолжению предыдущей линии или дуги /CONTIN:/.

Можно выделить следующие правила изображения дуги:

1) обычно дуга строится против часовой стрелки от точки к точке;

2) если есть в выбранной опции возможность задать угол, то отрицательный угол позволяет рисовать дугу по часовой стрелке;

3) по умолчанию дуга рисуется по начальной, промежуточной и конечной точкам.

При выборе в основном меню АВТОКАД режима *Edit* имеется возможность стирать созданные объекты, возвращать случайно стертые, перемещать и копировать, вычерчивать сопряжения между двумя существующими объектами, снимать фаски, разбивать объект на части, зеркаль-

но отображать, поворачивать, увеличивать и уменьшать, отсекать и т.п.

Практически все команды редактирования запрашивают выбор одного или нескольких объектов для обработки. Совокупность таких объектов называется набором выбора. Когда АВТОКАДу требуется не один объект, а набор выбора, появляется подсказка *Select objects:* (выбрать объекты) и на экране вы видите маленький прямоугольник - мишень для выбора объектов. Способы выбора объектов указываются в опциях соответствующей команды.

## *5.6.* НАУЧНАЯ ГРАФИКА

Компьютерная графика представляет значительный интерес для научных исследований. В частности, она выступает как средство формирования научной документации с использованием специальной нотации - математических знаков, индексов, шрифтов и т.п. В последнее время ученые чаще стали обращаться к имитационному моделированию на компьютере.

В компьютерной графике большое значение имеют методы и способы **геометрического моделирования.** Модели, геометрические преобразования составляют в настоящее время основу теории компьютерной графики и геометрического моделирования. **Аналитические модели** - это набор чисел, логических параметров, играющих роль коэффициентов в уравнениях, которые задают графический объект заданной формы. Например, аналитической моделью окружности на плоскости в параметрической форме являются уравнения

$$x=x0+R\text{-}cosA,$$
$$y=y0+R\text{-}sinA,$$

где *x0*, *y0* - координаты центра, *R* - радиус, *A* - угол. Параметрическое задание образов широко применяется в машинной графике и геометрии. Изображение окружности можно осуществить установкой последовательных точек (близко расположенных), изменяя генерирующий параметр *A* от 0 до 360°.

**Координатные модели** - это массивы координат точек, принадлежащих объектам. Например, поверхность задается массивом точек $Z = f(x, y)$ на координатной сетке $[x_i, y_j]$. Если точки в модели расположены в том же порядке, что и на линии образа, то модели называют упорядоченными. Помимо координат, в модели могут быть указаны дополнительные характеристики проекции касательных или нормальных векторов.

Приближенные модели содержат аппроксимации кривых методами вычислительной геометрии. Например, изображение гладких кривых можно осуществить ломаными линиями: линейными, параболическими или сплайнами. Используя вышеперечисленные геометрические модели, можно создавать различные демонстрационные картины. Например, модель Солнечной системы для наглядности удобно представить в динамической форме. Организуем движение точки (Земли) по окружности, в центре которой размещается круг (имитация Солнца). Установку точки на орбите осуществим по параметрическим формулам окружности:

$$X0 = 320 + r1 \cdot sin(A1);$$
$$Y0 = 240 + rl \cdot cos(Al),$$

где *r1* - радиус орбиты Земли, *A1* - параметрический угол, меняющийся от 0 до 360°. Чтобы организовать движение, достаточно в цикле устанавливать точку с координатами (*x0, y0*) для всех углов *A1*, принимающих значения от 0 до 360° с шагом h. Аналогичная процедура справедлива и для второй точки (Луны), которая изображается по подобным формулам, в которых центр орбиты (Земля) является подвижным:

$$x = x0 + r \cdot sin(A);$$
$$y = y0 + r \cdot cos(A),$$

где *r* - радиус орбиты Луны, *A* - угол вращения.

***Контрольные вопросы и упражнения***

1. Составьте каким-либо средством машинной графики бордюры каждого типа симметрии из следующих элементарных мотивов, рис. 2.21.

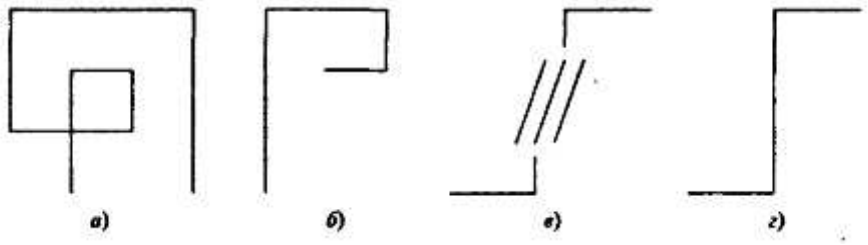



Рис. 2.21.

2. Постройте орнаменты различного типа симметрии из выбранного произвольного элементарного мотива каким-либо средством машинной графики.

3. Задана высота (м) над уровнем моря вершин: Мак-Кинли - 6200, Логан - 6100, Элберт - 440, Робсон - 4000, Митгелл - 2000. Составьте по этим данным столбчатую диаграмму.

4. По данным упражнения 3 составьте круговую диаграмму.

5. Постройте график функции $y = x \sin (1/x)$.

6. Создайте мультипликацию: вращение электрона в модели атома.

7. Изобразите шестиугольную призму.

8. Подготовьте иллюстрацию ко Дню учителя.

9. Организуйте в школе (вузе) компьютерный вернисаж.

# § 6. БАЗЫ ДАННЫХ И СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

## 6.1. ПОНЯТИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Базы данных - важнейшая составная часть **информационных систем.** Здесь мы ограничимся лишь кратчайшими общими сведениями об информационных системах, сосредоточив внимание на базах данных как таковых.

Информационные системы предназначены для хранения и обработки больших объемов информации. Изначально такие системы существовали в письменном виде. Для этого использовались различные картотеки, папки, журналы, библиотечные каталоги и т.д. Любая информационная система должна выполнять три основные функции: ввод данных, запросы по данным, составление отчетов.

**Ввод данных.** Система должна предоставлять возможность накапливания и упорядочивания данных. Необходимо обеспечить просмотр этих данных, внесение в них изменений и дополнений с тем, чтобы поддерживать актуальность информации.

**Запросы по данным.** В системе должна существовать возможность находить и просматривать отдельные части накопленной информации.

**Составление отчетов**. Время от времени возникает необходимость обобщать и анализировать большую группу данных (или даже все данные) информационной системы, представляя ее в виде документа.

Обслуживание информационных систем, реализованных в письменном (бумажном) виде, сопряжено со многими трудностями: чем больше информационная система, тем больше бумаги (карточек) и места требуется для их хранения (в этом можно убедиться на примере библиотеки); много времени тратится на поиск нужной информации. Сложности возникают при обновлении, анализе и обработке данных.

Предположим мы хотим собрать информацию про альбомы музыкальных групп. Пусть имеется информация о некоторых альбомах: 1965, Led Zeppelin 4, Lp, Help!, Atlantic, 1971. Lp(England), EMI. 1970, Flash Gordon, Parlophone, 1980, Led Zeppelin 3, Soundtrack, Lp, Atlantic. Этот список мало о чем говорит. Извлечь какую-либо информацию из этого набора данных практически невозможно.

Представим данные в виде табл. 2.2.

**Таблица 2.2 Информация об альбомах музыкальных групп**

| Название альбома | Год выпуска | Тип | Фирма альбома |
|---|---|---|---|
| Help! | 1965 | Lp (England) | Parlophone |
| Led Zeppelin 4 | 1971 | Lp | Atlantic |
| Led Zeppelin 3 | 1970 | Lp | Atlantic |
| Flash Gordon | 1980 | Soundtrack | EMI |

Теперь воспринимать и использовать информацию стало гораздо удобнее. Представленная таблица является **информационной моделью.** Объектами, отраженными в этой модели, являются музыкальные альбомы (групп), причем все данные взаимосвязаны.

## 6.2. ВИДЫ СТРУКТУР ДАННЫХ

В информатике совокупность взаимосвязанных данных называется **информационной структурой,** или **структурой данных**. В нашем примере объектами модели являются музыкальные альбомы. Свойства же этих объектов находятся в столбцах таблицы («Название альбома», «Год выпуска», «Тип альбома», «Фирма»), их называют **атрибутами** объектов. Таким образом, каждая строка таблицы - есть совокупность атрибутов объекта. Такую строку называют **записью,** а столбец - **полем записи**.

Помимо сведений, указанных в атрибутах, табличная организация данных позволяет получить дополнительную информацию. К примеру, нетрудно узнать (в предположении, что наша табл. 2.2 заполнена данными):

- какая группа выпустила больше альбомов за определенный период;
- число альбомов данной группы;
- сколько имеется альбомов типа Soundtrack (музыка к фильму);
- какая фирма выпустила наибольшее число альбомов данной группы.

Табличная организация данных называется также **реляционной.** Кроме табличной структуры данных существуют другие виды структурной организации данных.

Для **иерархических структур** (рис.2.22) характерна подчиненность объектов нижнего уровня объектам верхнего уровня. Важно отметить, что в дереве, между верхними и нижними объектами, задано отношение «один ко многим» (т.е. одной группе соответствует много альбомов, одному альбому соответствует много песен).

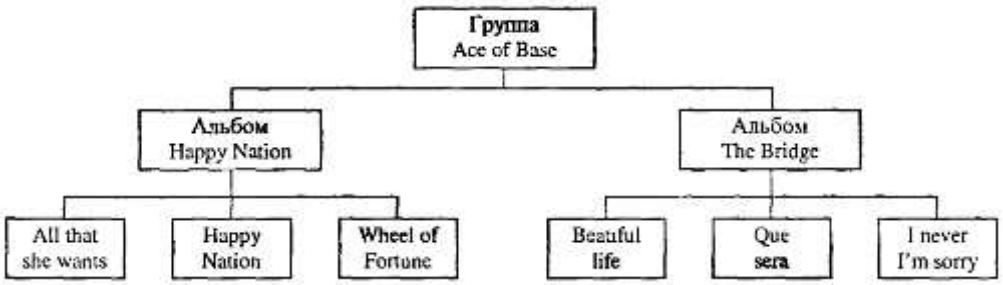



*Рис. 2.22.* Пример иерархической организации данных

Несмотря на то, что в атрибутах, описывающих песню, нет названия альбома, глядя на дерево по линиям связи можно сказать, какая песня принадлежит альбому. Благодаря линиям связи можно определить принадлежность альбома группе. Из данной иерархической структуры можно узнать:

- в каком альбоме больше песен;
- число альбомов выпущенных группой;
- есть ли в альбомах одинаковые песни и т.д.

**Сетевую структуру** данных можно представить в виде схемы, рис. 2.23.


*Рис. 2.23.* Пример сетевой организации данных

Глядя на рис. 2.23, можно определить, какими инструментами владеет музыкант, является ли он вокалистом. В этом случае есть два уровня взаимосвязанных объектов, но отношение между ними «многие ко многим».

Пусть в этой сетевой структуре данные о музыкантах и «инструментах» состоят из следующих атрибутов: музыкант - ФИО, рост, цвет волос, время рождения; инструмент- название инструмента, какой фирмой изготовлен инструмент.

Тогда схема позволяет ответить на следующие вопросы:

• гитары какой фирмы предпочитает большинство музыкантов;

• какой музыкант владеет наибольшим количеством инструментов и др.

Построение структуры данных происходит в следующем порядке:

• определяются объекты описания;

• определяются структуры этих объектов;

• выбирается тип структуры, отображающий отношения между объектами (табличная, иерархическая, сети);

• строится конкретная информационная структура.

## 6.3. ВИДЫ БАЗ ДАННЫХ

Дадим основное определение. База данных - это реализованная с помощью компьютера информационная структура (модель), отражающая состояние объектов и их отношения.

Следует учесть, что это определение не является единственно возможным. Информатика в отношении определений чаще всего не похожа на математику с ее полной однозначностью. Если подойти к понятию «база данных» с чисто пользовательской точки зрения, то возникает другое определение: база данных - совокупность хранимых операционных данных некоторого предприятия. Все дело в том, какой аспект доминирует в рассмотрении; в данной главе первое из определений более уместно.

Поскольку основу любой базы данных составляет информационная структура, базы данных делят на три рассмотренные выше типа: табличные (реляционные), сетевые, иерархические.

Опыт использования баз данных позволяет выделить общий набор их рабочих характеристик:

• **полнота** - чем полнее база данных, тем вероятнее, что она содержит нужную информацию (однако, не должно быть избыточной информации);

• **правильная организация** - чем лучше структурирована база данных, тем легче в ней найти необходимые сведения;

• **актуальность -** любая база данных может быть точной и полной, если она постоянно обновляется, т.е. необходимо, чтобы база данных в каждый момент времени полностью соответствовала состоянию отображаемого ею объекта;

• **удобство для использования** - база данных должна быть проста и удобна в использовании и иметь развитые методы доступа к любой части информации.

Соответственно возможностям организации реляционных, иерархических и сетевых информационых структур, существуют и аналогичные виды баз данных. В них данные представлены в формах, адекватных соответствующим структурам. Однако иерархические и сетевые базы данных являются гораздо менее распространенными, чем реляционные и не могут быть реализованы с помощью наиболее популярных СУБД, входящих в состав программного обеспечения ЭВМ, поэтому на них далее останавливаться не будем.

**Реляционные базы данных**

Наиболее распространенными в практике являются реляционные базы данных. Название «реляционная» (в переводе с английского relation - отношение) связано с тем, что каждая запись в таблице содержит информацию, относящуюся только к одному конкретному объекту.

Всякое отношение должно иметь свое имя. Пусть есть отношение с названием «Альбомы группы». В этом случае структура базы данных, состоящая из одной таблицы, запишется так: *Альбомы группы (название альбома, год выпуска, тип альбома, фирма)*. Однако чаще база данных строится на основе нескольких таблиц, связанных между собой через общие атрибуты. Пусть, например, в базе данных «Рок-энциклопедия» содержатся две таблицы - 2.3, *а* и 2.3, *б.*

**Таблица 2.3, *а* Музыкальные альбомы групп**

| Код альбома | Код группы | Название альбома | Год выпуска | Тип альбома | Фирма |
|---|---|---|---|---|---|
| 25 | 1 | Help! | 1965 | Lp (English) | Pariophone |
| 36 | 2 | Led Zeppelin 4 | l97l | Lp | Atlantic |
| 35 | 2 | Led Zeppelin 4 - | 1970 | Lp | Atlantic |
| 34 | 3 | Flash Gordon | 1980 | Soundtrack | EMI |

**Таблица 2.3, *б Рок* группы**

| Код группы | Название группы | Страна | Дата создания | Дата распада |
|---|---|---|---|---|
| 1 | The Bealles | Англия | 1963 | I970 |
| 2 | Led Zeppelin 4 | Англия | 1989 | - |
| 3 | Flash Gordon | Англия | 199I | - |

Эти две таблицы связаны между собой общим полем «Код группы». Поле «Код альбома» в таблице 2.3, *а* создается для того, чтобы отличать альбомы друг от друга. Это очень важно, так как в таблице могут находиться альбомы с одинаковыми названиями.

Необходимость использования больше одной таблицы станет заметной, если объединить эти таблицы в одну (табл. 2.4).

**Таблица 2.4 Объединение таблиц 23**

| Название группы | Страна | Дата создания | Дата распада | Название альбома | Год выпуска | Тип альбома | Фирма |
|---|---|---|---|---|---|---|---|
| The Beatles | Англия | 1963 | I970 | With the Beatles | 1963 | Lp | Pariophone |
| The Beatles | Англия | 1963 | I970 | Please, please me | 1963 | Lp | Pariophone |
| The Beatles | Англия | 1963 | I970 | Rubber soul | 1963 | Lp | Pariophone |

Из таблицы 2.4 видно, что при внесении в нее данных об альбомах определенной группы каждый раз приходится дублировать информацию первых четырех полей таблицы. Многократное сохранение в БД одних и тех же данных (название группы, страна, дата создания, дата распада) приведет к неэффективному использованию памяти, к тому же существенно возрастет вероятность ошибок при вводе данных. Разбив же данные по таблицам, можно в значительной степени избежать этих трудностей.

Через связь, определенную между этими таблицами, можно узнать

- сколько альбомов выпустила группа;
- выпускались ли альбомы у фирмы EMI;
- в каком году было выпущено максимальное количество альбомов и т.п.

Реляционные базы данных удобны еще и тем, что для получения ответов на различные за-

просы существует разработанный математический аппарат, который называется исчислением отношений или реляционной алгеброй. Ответы на запросы получаются путем «разрезания» и «склеивания» таблиц по строкам и столбцам. При этом ясно, что ответы также будут иметь форму таблиц.

Надо отметить, что база данных - это, собственно, хранилище информации и не более того. Однако, работа с базами данных трудоемкая и утомительная. Для создания, ведения и осуществления возможности коллективного пользования базами данных используются программные средства, называемые системами управления базами данных (СУБД).

## 6.4. СОСТАВ И ФУНКЦИИ СИСТЕМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

База данных предполагает наличие комплекса программных средств, обслуживающих эту базу данных и позволяющих использовать содержащуюся в ней информацию. Такие комплексы программ называют СУБД. СУБД - это программная система, поддерживающая наполнение и манипулирование данными, представляющими интерес для пользователей при решении прикладных задач. Иными словами, СУБД является интерфейсом между базой данных и прикладными задачами.

Ниже перечислены основные функции СУБД.

1. **Определение данных** - определить, какая именно информация будет храниться в базе данных, задать свойства данных, их тип (например, число цифр или символов), а также указать, как эти данные связаны между собой. В некоторых случаях есть возможность задавать форматы и критерии проверки данных.

2. **Обработка данных** - данные могут обрабатываться самыми различными способами. Можно выбирать любые поля, фильтровать и сортировать данные. Можно объединять данные с другой, связанной с ними, информацией и вычислять итоговые значения.

3. **Управление данными** - можно указать, кому разрешено знакомиться с данными, корректировать их или добавлять новую информацию. Можно также определять правила коллективного доступа.

Входящие в состав современных СУБД средства совместно выполняют следующие функции:

• *описание* данных, их структуры (обычно описание данных и их структуры происходит при инициировании новой базы данных или добавлении к существующей базе новых разделов (отношений); описание данных необходимо для контроля корректности использования данных, для поддержания целостности базы данных);

• *первичный ввод, пополнение* информации в базе данных;

• *удаление* устаревшей информации из базы данных;

• *корректировку* данных для поддержания их актуальности;

• *упорядочение (сортировку)* данных по некоторым признакам;

• *поиск информации* по некоторым признакам (для описания запросов имеется специальный язык запросов, он обеспечивает также интерфейс между базой данных и прикладными программами пользователей, позволяет этим программам использовать базы данных);

• *подготовку и генерацию отчетов* (средства подготовки отчетов позволяют создавать и распечатывать сводки по заданным формам на основе информации базы данных);

• *защиту информации и разграничение доступа* пользователей к ней (некоторые разделы базы данных могут быть закрыты для пользователя совсем, открыты только для чтения или открыты для изменения; кроме того, при многопользовательском режиме работы с базой данных необходимо, чтобы изменения вносились корректно; для сохранения целостности данных служит механизм трансакций при манипулировании данными - выполнение манипуляций небольшими пакетами, результаты каждого из которых в случае возникновения некорректности операций «откатываются» и данные возвращаются к исходному состоянию);

• *резервное сохранение и восстановление* базы данных, которое позволяет восстановить утраченную при сбоях и авариях аппаратуры информацию базы данных, а также накопить статистику работы пользователей с базой данных;

• *поддержку интерфейса с пользователями,* который обеспечивается средствами ведения диалога (по мере развития и совершенствования СУБД этот интерфейс становится все более дружественным; дружественность существующих средств интерфейса предполагает

• наличие развитой системы помощи (подсказки), к которой в любой момент может обратиться пользователь, не прерывая сеанса работы с компьютером и базой данных;

• защиту от необдуманных действий, предупреждающую пользователя и предотвращающую потерю информации в случае поспешных или ошибочных команд;

• наличие нескольких вариантов выполнения одних и тех же действий, из которых пользователь может выбрать наиболее удобные для себя, соответствующие его подготовке, квалификации, привычкам;

• тщательно продуманную систему ведения человеко-машинного диалога, отображение информации на дисплее, использование клавиш клавиатуры). В настоящее время выделяют пять уровней проблематики систем управления базами данных:

• реляционные базы данных, 1970 - 90 гг.;

• объектно-ориентированные базы данных, 1980 - 90 гг.;

• интеллектуальные базы данных, 1985 - 90 гг.;

• распределенные базы данных, начало 1990 гг.;

• базы данных мультимедиа и виртуальной реальности настоящего времени.

Архитектурно СУБД состоит из двух основных компонентов; **языка описания данных** (ЯОД), позволяющего создать схему описания данных в базе, и **языка манипулирования данными** (ЯМД), выполняющего операции с базой данных (наполнение, обновление, удаление, выборку информации). Данные языки могут быть реализованы в виде тренажеров или интерпретаторов. Помимо ЯОД и ЯМД к СУБД следует отнести **средства** (или языки) **подготовки отчетов** (СПО), позволяющие подготовить сводки (отчеты) на основе информации, найденной в базе данных, по заданным формам.

Язык описания данных (ЯОД) - это язык высокого уровня *декларативного* (непроцедурного) типа, предназначенный для формализованного описания типов данных, их структур и взаимосвязей. Исходные тексты описания данных на этом языке после трансляции отображаются в управляющие таблицы, задающие размещение в памяти ЭВМ и связи между собой рассматриваемых данных. В соответствии с этими описаниями СУБД находит в базе требуемые данные, правильно преобразует их и передает, например, в прикладную программу пользователя, которой они потребовались. При записи данных в базу СУБД по этим описаниям определяет место в памяти ЭВМ, куда их требуется поместить, преобразует к заданному виду и устанавливает необходимые связи.

Язык манипулирования данными (или язык запросов) представляет собой систему команд, например, следующего типа:

• произвести выборку данного, значение которого удовлетворяет заданным условиям;

• произвести выборку всех данных определенного типа, значения которых удовлетворяют заданным условиям;

• найти в базе позицию данного и поместить туда новое значение (или удалить данное) и т.д.

Широкое распространение имеют СУБД для персональных компьютеров типа DBASE (DBASE III, IV, FoxPro, Paradox), Clipper, Clarion. Эти СУБД ориентированы на однопользовательский режим работы с базой данных и имеют очень ограниченные возможности. Языки подобных СУБД представляют собой сочетание команд выборки, организации диалога, генерации отчетов. В связи с развитием компьютерных сетей, в которых персональные компьютеры выступают в качестве развитых (интеллектуальных) терминалов, новые версии СУБД все в большей степени включают в себя возможности описанного ниже языка манипулирования данными SQL.

В последнее время стали среди СУБД популярными ACCESS (входит в состав MS Office), Lotus, Oracle.

### Язык манипулирования данными SQL

Рассмотрим в качестве примера языка манипулирования данными некоторые команды языка SQL (от английских слов Structured Query Language), ставшего классическим языком реляционных баз данных.

Простейшая операция выборки представляется командой SELECT - FROM -WHERE (выбрать **-** из - где):

select <список атрибутов>
from <отношение>
**where <условие>.**

Например, если необходимо из отношения «Успеваемость», имеющего схему:

Успеваемость (ФПО_студента, Дисциплина, Оценка, Дата, Преподаватель)

произвести выборку данных о том, какие оценки студент Иванов И.И. получил и по каким предметам, надо задать команду:

```
select      Дисциплина, Оценка
from        Успеваемость
where       ФИО_студента = «Иванов И. И.».
```

Часть команды «where» не является обязательной. Например, можно получить список всех студентов из отношения «Успеваемость» с помощью следующей команды:

```
select      unique ФИО_студента
from        Успеваемость.
```

Ключевое слово unique позволяет исключить из результата дубликаты значений атрибута. Выбрать полностью информацию из таблицы можно с помощью команды

```
select      *
from        Успеваемость.
```

Условие, следующее за «where», может включать операторы сравнения =, **<>,** >=, **<, <=,** булевы операторы AND, OR, NOT, а также скобки для указания желаемого порядка операции. Например, выбрать из таблицы «Успеваемость» фамилии студентов, сдавших на "5" экзамен по информатике, можно с помощью команды

```
select      ФИО_студента
from        Успеваемость
where       Дисциплина = «Информатика» AND Оценка=5.
```

Выборка может быть и вложенной, когда необходимо использовать в условии результаты другой выборки. Например, если надо из отношения «Успеваемость» выбрать только студентов физико-математического факультета, пользуясь отношением «Студент», то команда select может выглядеть так:

```
select      ФИО_студента
from        Успеваемость
where       ФИО_студента is in
(select     Фамилия
from        Студент
where       Ф_т = «физмат»).
```

Здесь «is in» является представлением оператора принадлежности элемента множеству. Можно также использовать операторы «is not in» («не принадлежит множеству»), «contains» - содержит, «does not contains» - не содержит. Смысл выражения «A contains B» (A содержит B) тот же, что и выражения «B is in A» (B принадлежит множеству A). Помимо слов select, from, where в команде выборки можно использовать и другие служебные слова, например:

order by **<**атрибут**>** asc - определяет сортировку результата выборки в порядке возрастания (asc) или убывания (desc) значения атрибута;

group by <атрибут1> - группирует данные по значениям атрибута;
having set **<**атрибут2**>**

minus - операция вычитания множеств (данных выборок).

Помимо команды выборки select, язык SQL имеет команды, позволяющие обновлять данные (update), вставлять (insert) и удалять (delete). Например, если студенты переводятся со 2-го курса на третий, информацию можно обновить командой

```
update    Студент
set       Курс=3
where     Курс=2.
```

Если атрибут «Семенов С.С.» сдал экзамен по информатике на «5» 15 января 1996 г. преподавателю Петрову П.П., то информация об этом может быть добавлена в таблицу «Успеваемость» командой

```
insert inio Успеваемость:
<«Семенов С.С.», «Информатика», 5,15/01/96, Петров П.П.>.
```

Оператор insert может быть использован для включения одной строки (как в этом примере) или произвольного числа строк, определенных списком кортежей, заключенных в скобки, или операций выборки select из какой-либо другой таблицы. Команда delete используется для удаления информации из таблицы. Например,

```
delete Успеваемость
where Оценка=2
```

позволяет удалить информацию о студентах, получивших 2 (в случае их отчисления).

Существенно расширяют возможности языка библиотечные функции, такие как count (подсчет), sum (суммирование), avg (среднее), max и min.

Например, подсчитать число студентов в таблице «Студент»: select count (*) from Студент.

## *6.5.* ПРИМЕРЫ СИСТЕМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

### СУБД DBASE

СУБД типа DBASE позволяют работать с реляционными базами данных (БД), структура которых состоит из трех элементов:

- число полей БД;
- характеристика каждого поля**;**
- число записей в БД.

Каждое поле имеет следующие характеристики:

| Field name | Type | Width | Dec |
|---|---|---|---|
| (имя поля) | (тип) | (ширина) | (дес.знаки). |

**Field name** - может состоять из набора символов, но без пробелов.

**Type** - в системах типа DBASE имеется 5 типов полей:

- C (Character) - символьный (текстовый) тип;
- N (Numerical) - числовой тип;
- L (Logical) - логический тип;
- D (Date) - поле дат, содержит даты в виде dd/mm/yy;

M (Memo) - поле памяти, содержит большой текст (файл).

**Width** - обозначает допустимую ширину поля.

**Dec** - используется для числовых полей и определяет точность задаваемых чисел.

DBASE создает следующие типы файлов:

.dbf - файлы с записями БД;
.prg - файлы с текстами программ;
.frm - файлы структуры форматных отчетов;
.ndx - индексные файлы, сортирующие записи по определенному ключу;
.mem - файлы данных переменной Mem.

Запуск СУБД осуществляется **из** операционной системы exe-файлом (db.exe, foxdb и т.п.), выход-командой **.Quit.**

Теперь опишем кратко основные команды СУБД.

Создание БД осуществляется командой **Create.**

Create (например, «Абитуриент»)

После ввода этой команды на экране появится форма:

| Field name | Type | Width | Dec |
|---|---|---|---|
| (имя поля) | (тип) | (ширина) | (дес.знаки). |

.001

В соответствии с этой формой создадим структуру таблицы:

001 ФИО, C, 18
002 год_рожд, C, 7
003 район. C, 13
004 адрес. C, 100
005 группа, C, 3
006 оценка 1,N,3
007 оценка2,N, 3
008 оценка3,N, 3

Теперь можно начать заполнение таблицы записями.

В случае заполнения записями уже существующей базы данных, необходимо предварительно эту базу командой Use сделать активной:

.Use Абитуриент (use - использовать),
.Append (добавить)

Данные вводят в карточки, имеющие следующую форму:

Запись #00001
ФИО:
год_рожд:
район :
адрес:
группа:
оценка1:
оценка2:
оценка3:

Например,

Запись #00005

| | |
|---|---|
| ФИО: | Семенов Сергей Викторович |
| год_рожд: | 1980 |
| район: | Туруханский |
| адрес: | ул. Декабристов, д. 12, кв.23 |
| группа: | И2 |
| оценка!: | 5 |
| оценка2: | 4 |
| оценка3: | 4 |

Запись можно ввести в определенное место БД, введя одну из команд:

.**Insert** (вставить)

или

.**Insert before** .

Перемещение по таблице и просмотр записей БД осуществляется командами:

| | |
|---|---|
| **Go top** | - (идти наверх) установка указателя на первую запись, |
| **Go bottom** | - (идти вниз) установка указателя на последнюю запись; |
| **List** | - (список) просмотр всех записей БД; |
| **Display** | - (отобразить) просмотр записи, на которой находится указатель, |
| **Browse** | - (просмотреть) помимо просмотра позволяет редактировать записи БД. |

Редактирование записей позволяют проводить следующие команды:

| | |
|---|---|
| **Edit N** | - редактирование записи с номером *N;* |
| **Change** | - (поменять) изменения только в некоторых полях или записях, удовлетворяющих заданным условиям; |
| **Delete** | - (удалить) стирание ненужных записей; |
| **Copy** | -(копировать) копирование записей. |

Изменить структуру БД можно командой **Modify**. Ниже предложен перечень команд, осуществляющих обработку данных:

| | |
|---|---|
| **Report form** | - (отчет, форма) создание отчетов; |
| **Sort** | - (сортировка) упорядочение БД по какому-либо ключу; |
| **Index** | - (индекс) индексирование БД; |
| **Find** | - (найти) поиск в БД. |

Работу с несколькими БД помогают вести команды:

| | |
|---|---|
| **Select** | - (выбор) сделать активной какую-либо БД; |
| **Update** | - (расширить) передача данных из одной БД в другую; |
| **Join to** | - (присоединить) соединение целых БД. |

Для осуществления интерактивности БД используют команды ввода и вывода:

| | |
|---|---|
| **Wait** | - (ожидание) пауза, приостановка; |
| **Input** | - (вход) ввод данных; |
| **Say** | - (сказать) вывод информации; |
| **Read** | - (читать) ввод данных. |

**СУБД Microsoft Access**

Access - в переводе с английского означает «доступ». MS Access - это функционально полная реляционная СУБД. Кроме того, MS Access одна из самых мощных, гибких и простых в использовании СУБД. В ней можно создавать большинство приложений, не написав ни единой строки программы, но если нужно создать нечто очень сложное, то на этот случай MS Access предоставляет мощный язык программирования - Visual Basic Aplication.

Популярность СУБД Microsoft Access обусловлена следующими причинами:

• Access является одной из самых легкодоступных и понятных систем как для профессионалов, так и для начинающих пользователей, позволяющая быстро освоить основные принципы работы с базами данных;

• система имеет полностью русифицированную версию;

• полная интегрированность с пакетами Microsoft Office: Word, Excel, Power Point, Mail;

• идеология Windows позволяет представлять информацию красочно и наглядно;

• возможность использования OLE технологии, что позволяет установить связь с объектами другого приложения или внедрить какие-либо объекты в базу данных Access;

• технология WYSIWIG позволяет пользователю постоянно видеть все результаты своих действий;

• широко и наглядно представлена справочная система;

• существует набор «мастеров» по разработке объектов, облегчающий создание таблиц, форм и отчетов.

Запустить систему Access можно несколькими способами:

• запуск с помощью главного меню в WINDOWS 95;

• запуск с помощью ярлыка на панели инструментов.

После запуска системы появится главное окно Access, рис. 2.24. Здесь можно открывать другие окна, каждое из которых по-своему представляет обрабатываемые данные. Ниже приведены основные элементы главного окна Access, о которых необходимо иметь представление.

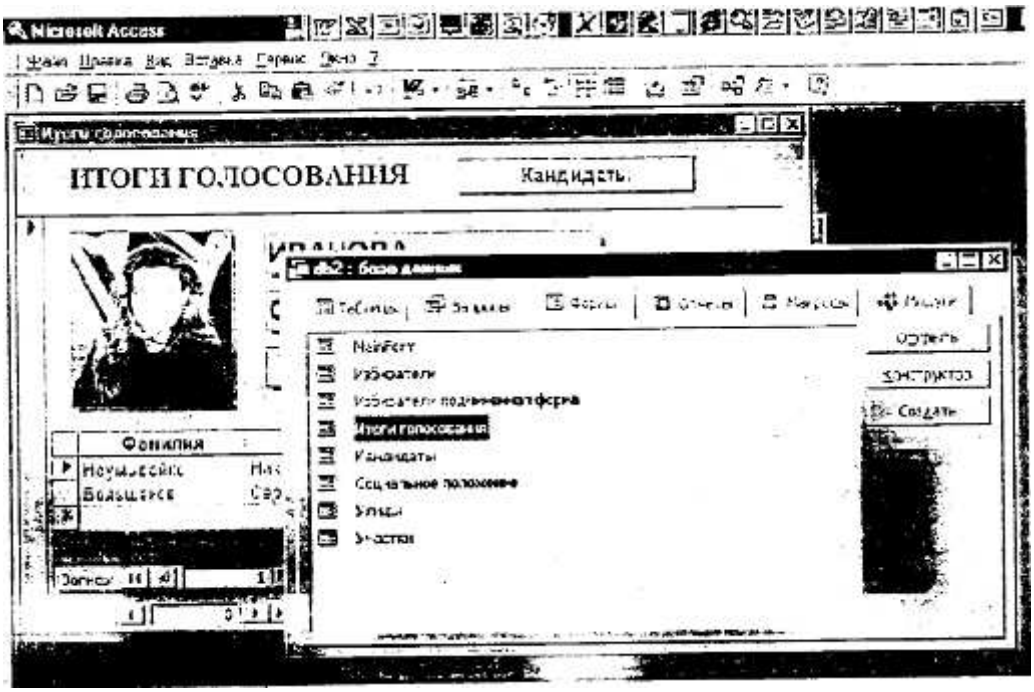



*Рис.2.24.* Экран СУБД Access

В строке заголовка отображается имя активной в данный момент программы. Строка заголовка главного окна Access всегда отображает имя программы MICROSOFT Access.

**Пиктограмма системного меню** - условная кнопка в верхнем левом углу главного окна практически любого приложения. После щелчка на этой пиктограмме появляется меню, ко.торое позволяет перемещать, разворачивать, сворачивать или закрывать окно текущего приложения и изменять его размеры. При двойном щелчке на пиктограмме системного меню работа приложения

завершается.

**Полоса меню** содержит названия нескольких подменю. Когда активизируется любое из этих названий, на экране появляется соответствующее подменю. Перечень подменю на полосе Access и их содержание изменяются в зависимости от режима работы системы.

**Панель инструментов** - это группа пиктограмм, расположенных непосредственно под полосой меню. Главное ее назначение - ускоренный вызов команд меню. Кнопки панели инструментов тоже могут изменяться в зависимости от выполняемых операций. Можно изменять размер панели инструментов и передвигать ее по экрану. Также можно отобразить, спрятать, создать новую панель инструментов или настроить любую панель инструментов.

В левой части **строки состояния** отображается информация о том, что вы делаете в настоящее время.

**Окно базы данных** появляется при открытой базе данных. В нем сосредоточены все «рычат управления» базой данных. Окно базы данных используется для открытия объектов, содержащихся в базе данных, таких как таблицы, запросы, отчеты, формы, макросы и модули. Кроме того, в строке заголовка окна базы данных всегда отображается имя открытой базы данных.

С помощью **вкладки объектов** можно выбрать тип нужного объекта (таблицу, запрос, отчет, форму, макрос, модуль). Необходимо сказать, что при открытии окна базы данных всегда активизируется вкладка-таблица и выводится список доступных таблиц базы данных. Для выбора вкладки других объектов базы данных нужно щелкнуть по ней мышью.

Условные кнопки, расположенные вдоль правого края окна базы данных, используются для работы с текущим объектом базы данных. Они позволяют создавать, открывать или изменять объекты базы данных.

К основным объектам Access относятся таблицы, запросы, формы, отчеты, макросы и модули.

**Таблица** - это объект, который определяется и используется для хранения данных. Каждая таблица включает информацию об объекте определенного типа. Как уже известно, таблица содержит поля (столбцы) и записи (строки). Работать с таблицей можно в двух основных режимах: в режиме конструктора и в режиме таблицы.

В режиме конструктора задается структура таблицы, т.е. определяются типы, свойства полей, их число и названия (заголовки столбцов). Он используется, если нужно изменить структуру таблицы, а не хранящиеся в ней данные. В этом режиме каждая строка верхней панели окна соответствует одному из полей определяемой таблицы.

Режим таблицы используется для просмотра, добавления, изменения, простейшей сортировки или удаления данных. Чтобы перейти в режим таблицы, надо дважды щелкнуть мышью по имени нужной таблицы в окне базы данных (или, выделив в окне БД имя нужной таблицы, воспользоваться кнопкой открытого окна БД).

Из режима конструктора перейти в режим таблицы можно, щелкнув по кнопке таблицы на панели инструментов.

В режиме конструктора и в режиме таблицы перемещение между полями осуществляется с помощью клавиши TAB, а также вверх или вниз по записям с помощью клавиш, но в большинстве случаев пользоваться мышью гораздо удобнее.

Вследствие того, что в таблицах, как правило, содержится большое количество записей, размещение всех их на экране невозможно. Поэтом) для перемещения по таблице используют полосы прокрутки, расположенные в нижней и правой части окна. Левее нижней полосы прокрутки выводится номер текущей записи и общее число записей таблицы. Для перехода к записям с нужным номером необходимо активизировать поле *Номера записи,* щелкнув по нему, или нажать клавишу F5, после чего набрать на клавиатуре новый номер записи и затем нажать клавишу <Enter>.

**Запрос** - это объект, который позволяет пользователю получить нужные данные из одной или нескольких таблиц. Можно создать запросы на выбор, обновление, удаление или на добавление данных. С помощью запросов можно создавать новые таблицы, используя данные уже существующих одной или нескольких таблиц.

По сути дела, запрос - это вопрос, который пользователь задает Access о хранящейся в базе данных информации. Работать с запросами можно в двух основных режимах: в режиме конструктора и в режиме таблицы.

Здесь надо вспомнить о том, что ответы на запросы получаются путем «разрезания» и

«склеивания» таблиц по строкам и столбцам, и что ответы будут также иметь форму таблиц. В режиме конструктора формируется вопрос к базе данных.

**Форма** - это объект, в основном, предназначенный для удобного ввода отображения данных. Надо отметить, что в отличие от таблиц, з формах не содержится информации баз данных (как это может показаться на первый взгляд). Форма - это всего лишь формат (бланк) показа данных на экране компьютера. Формы могут строиться только на основе таблиц или запросов. Построение форм на основе запросов позволяет представлять в них информацию из нескольких таблиц.

В форму могут быть внедрены рисунки, диаграммы, аудио (звук) и видео (изображение).

Режимы работы с формой:

•*режим формы* используется для просмотра и редактирования данных; предоставляет дружественную среду для работы с данными и удобный дизайн их представления на экране;

•*режим конструктора форм* необходим, если необходимо изменить определение

формы (структуру или шаблон формы, а не представленные в ней данные), надо открыть форму в режиме конструктора;

•*режим таблицы* позволяет увидеть таблицу, включающую все поля формы; чтобы переключиться в этот режим при работе с формой, надо нажать кнопку таблицы на панели инструментов.

**Отчет** - это объект, предназначенный для создания документа, который впоследствии может быть распечатан или включен в документ другого приложения. Отчеты, как и формы, могут создаваться на основе запросов и таблиц, но не позволяют вводить данные.

Режимы работы с отчетом:

*Режим предварительного просмотра* позволяет увидеть отчет таким, каким он будет воплощен при печати. Для того чтобы открыть отчет в режиме предварительного просмотра,надо

• щелкнуть по вкладке *Отчеты,*

• кнопкой выбрать необходимый отчет в окне базы данных;

• щелкнуть по кнопке *Просмотра.*

*Режим конструктора* предназначен для изменения шаблона (структуры отчета).

**Макрос** - это объект, представляющий собой структурированное описание одного или нескольких действий, которые должен выполнить Access в ответ на определенное событие. Например, можно определить макрос, который в ответ на выбор некоторого элемента в основной форме открывает другую форму. С помощью другого макроса можно осуществлять проверку значения некоторого поля при изменении его содержания. В макрос можно включить дополнительные условия для выполнения или невыполнения тех или иных включенных в него действии. Возможно также из одного макроса запустить другой макрос или функцию модуля.

Работа с формами и отчетами существенно облегчается за счет использования *макрокоманд.* В MS Access имеется свыше 40 макрокоманд, которые можно включать в макросы. Макрокоманды выполняют такие действия, как открытие таблиц и форм, выполнение запросов, запуск других макросов, выбор опций из меню, изменение размеров открытых окон и т.п. Макрокоманды позволяют нажатием одной (или нескольких одновременно) кнопки выполнять комплекс действий, который часто приходится выполнять в течение работы. С их помощью можно даже осуществлять запуск приложений, поддерживающих динамический обмен данных (DDE), например MS Excel, и производить обмен данными между вашей базой данных и этими приложениями. Один макрос может содержать несколько макрокоманд. Можно также задать условия выполнения отдельных макрокоманд или их набора.

**Модуль** - объект, содержащий программы на MS Access Basic, которые позволяют разбить процесс на более мелкие действия и обнаружить те ошибки, которые невозможно было бы найти с использованием макросов.

Завершив работу с Access (или с ее приложением), надо корректно закончить сеанс. Простое выключение компьютера - плохой метод, который может привести к возникновению проблем. При работе WINDOWS приложения используют множество файлов, о существовании которых пользователь может даже не подозревать. После выключения машины эти файлы останутся открытыми, что в будущем может сказаться на надежности файловой системы жесткого диска.

Безопасно выйти из Access можно несколькими способами:

• двойным щелчком мыши на пиктограмме системного меню в строке заголовка главного

окна Access;

• из меню Access выбором пункта *Файл Выход,*

• нажатием комбинации клавиш Alt + F4.

***Контрольные вопросы и задания***

1. Каково назначение программ, входящих в состав СУБД?
2. Какой интерфейс можно считать дружественным?
3. Какие компоненты можно выделить в составе СУБД?
4. В чем состоят функции языков описания и манипулирования данными?
5. Охарактеризуйте основные команды языка SQL.
6. Как с помощью команд SQL задать поиск в базе данных?
7. Как с помощью команд SQL модифицировать базу данных?
8. Используя СУБД типа DBase разработайте базы данных:

а) телефонный справочник;

б) каталог программного обеспечения персонального компьютера;

в) домашняя библиотека.

9. Используя СУБД Access разработайте БД «Музыкальная энциклопедия».

## § 7. ЭЛЕКТРОННЫЕ ТАБЛИЦЫ

### 7.1. НАЗНАЧЕНИЕ И ОСНОВНЫЕ ФУНКЦИИ ТАБЛИЧНЫХ ПРОЦЕССОРОВ

Как показала практика, решение многих задач экономического характера на языках высокого уровня с использованием всего арсенала приемов и методов профессионального программирования - сложное и громоздкое дело. Понадобился принципиально иной подход, и он был найден и воплощен в виде электронных таблиц - инструмента, доступного непрофессионалам. Основная область применения электронных таблиц - это те сферы человеческой деятельности, где информация предоставляется в виде прямоугольных таблиц (планово-финансовых и бухгалтерских документов, учета материальных ценностей и др.), требующих при обработке проведения математических расчетов, откуда, по-видимому, и возник термин «табличный процессор». Отметим, что реляционные базы данных, также представляемые с помощью таблиц, к расчетам, как правило, не приспособлены.

В настоящее время известно много вариантов электронных таблиц: АБАК, Варитаб-86. Суперплан, Multiplan, SuperCalk, QuattroPro, Excel, Lotus 1-2-3 и др. Принципиально все они представляют табличный процессор и разнятся лишь интерфейсом и сервисными возможностями.

Электронная таблица (ЭТ) - это прямоугольная матрица, состоящая из ячеек, кдая из которых имеет свой номер, рис. 2.25.

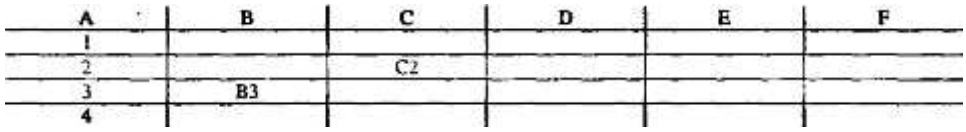



*Рис. 2.25.* Электронная таблица

Номер ячейки определяется обычным координатным способом, например, ячейка В3 и т.д.

Группа ячеек (диапазон) задается через двоеточие, например, B3:D4 (или B3..D4) и образует прямоугольник, включающий ячейки В3, С3, D3, В4, С4, D4.

В каждую из ячеек можно занести *число, формулу* (арифметическое выражение) или *текст.* Если в ячейку ЭТ записана формула, то в исходном состоянии на экране отображается значение этой формулы, а не она сама. Операндами формулы могут быть математические функции, константы, номера ячеек (содержимое ячейки с указанным номером). Ячейка ЭТ имеет сложную «многослойную» структуру, в ней может стоять ссылка на другую ячейку, значение которой является результатом вычислений по другой формуле и т.д.

Примеры функций:

sum(A2:A8) - сумма значений всех ячеек от A2 до A8;
sin(D5) - синус числа из ячейки D5;
cos(F3) - косинус числа из ячейки F3.

Пример формулы:

2.7. * A6 + cos (sum (D5:F7))

Приведенная формула означает, что мы хотим получить результат следующих вычислений: произведение числа из ячейки A6 на 2.7 сложить с косинусом угла, который является суммой чисел из ячеек D5, E5, F5, D6, E6, F6, D7, E7, F7.

Данные, входящие в таблицы, можно автоматически представлять в виде графиков, диаграмм, гистограмм и т.д.

Пользователь работает в диалоге со специальной программой, которая позволяет заполнять ячейки нужным ему содержимым (текстами, числами или формулами для расчетов); очищать их, копировать и удалять, сортировать (т.е. располагать клетки, а также строки и столбцы из них, в определенном порядке); производить вычисления над всей таблицей или ее частью, сохранять таблицу на диске и распечатывать частично или полностью на бумагу и т.д.

Приведем пример, иллюстрирующий возможности ЭТ.

Формирование зарплатной ведомости.

Так выглядят исходные данные для заполнения электронной таблицы:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Зарплатная ведомость фирмы «Рога и копыта» | | | | |
| 2 | ФИО | Оклад | Начисление | Налог | Всего |
| 3 | Балаганов А | 1500 | B3*l.6 | C3*0 12 | C3-D3 |
| 4 | Бендер О | 3000 | B4*l.6 | C4*0.12 | C5-D5 |
| 5 | Паниковскин М. | 1000 | B5*\|.6 | C5*0.12 | C5-D5 |

Здесь мы имеем дело с тремя типами содержимого ячеек: текст, число, формула. Ввод исходных данных происходит в командной строке. После заполнения таблицы мы увидим на экране:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Зарплатная ведомость фирмы «Рога и копыта» | | | | |
| 2 | ФИО | Оклад | Начисление | Налог | Всего |
| 3 | Балаганов А. | 1500 | 2400 | 288 | 2112 |
| 4 | Бендер О. | 3000 | 4800 | 576 | 4224 |
| 5 | Паниковскнй М. | 1000 | 1600 | 192 | 1408 |

Обычно работник бухгалтерии, поправив одну из цифр, вынужден был исправлять весь комплект взаимосвязанных документов, куда явно или неявно входил исправленный параметр. С помощью ЭТ такое изменение может быть учтено мгновенно и всюду.

## 7.2. ЭЛЕКТРОННЫЕ ТАБЛИЦЫ SUPERCALC

### Общие сведения

Одним из популярных табличных процессоров под DOS для компьютеров PC-286, -386 является SuperCalc-4 (SC-4).

SC4 позволяет работать с семью типами диаграмм и графиков, позволяет вводить различные обозначения, шкалы переменных, заголовки. В SC-4 можно создавать базы данных (БД),

имеются простейшие средства, характерные для систем управления БД. Кроме того, SC-4 располагает средствами для перевода информации к виду, доступному из текстовых редакторов, систем управления БД (например, семейства DBASE) и других программных средств.

После загрузки системы на экране появляется рекламная заставка фирмы-разработчика и далее после нажатия любой клавиши - пустая электронная таблица. ЭТ, созданные пользователем, записываются на диск в специальном оригинальном формате и имеют по умолчанию стандартное расширение .cal. Эти файлы в неизменном виде могут обрабатываться только системами SuperCalc-4 и SuperCalc-5.

Электронная таблица SC-4 состоит из клеток, образующих строки (rows) и столбцы (columns). Столбцы обозначены одно- и двухсимвольными буквами латинского алфавита: A, B, C, ..., Z, AB, ..., AZ, BA, ..., BZ, ..., IA, ..., IU. Максимальное число столбцов 255. Строки обозначены номерами от 1 до 9999. В обозначении каждой клетки указывают координаты столбца и строки. Например: Al, B20, IA1. Такое обозначение клетки еще называют адресом клетки. ! В каждый момент времени одна из клеток является активной (АК). Она высвечивается на экране при помощи указателя, которым можно управлять. Активная клетка доступна пользователю для чтения и записи данных. Для быстрого перемещения указателя к краю ЭТ используют одновременное нажатие клавиши END и стрелок (к верхнему краю, к нижнему, к левому и к правому). Для листания ЭТ по страницам используют клавиши PageUp (страница вверх), PageDown (страница вниз), Ctrl + <== , Ctrl + ==> (страницы влево или вправо).

ЭТ имеет обрамление (верхняя строка и левый столбец) с именами строк и столбцов. Сами клетки составляют рабочую область ЭТ. Напомним, что на экране видна . лишь часть таблицы. В нижней части экрана расположены четыре служебные строки:

• в первой строке отображается адрес и содержимое АК, а также направление ' движения указателя АК;

• во второй строке содержится информация об ЭТ;

• в третьей строке вводятся данные или команды;

• в четвертой строке содержатся подсказки и дополнительная информация о режиме работы ЭТ, назначении функциональных клавиш или пунктов меню команд.

Объекты, с которыми работает SC4: клетки, столбец, строка, диапазон столбцов (например A:C), диапазон строк (например 4:7) и блок клеток. Блок клеток задается адресами левой верхней и правой нижней клеток. В качестве разделителя используется двоеточие или точка, например, A3:B5 или A3.B5.

*Список*- один или более адресов объектов, разделенных запятыми.

*Ссылки* - адреса клеток, используемые в качестве имен переменных в формулах.

Обычное обозначение адресов является относительным, так как оно показывает расстояние от клетки, в которой содержится формула, до клетки, на которую в этой формуле есть ссылка. Например, пусть в клетке A3 хранится формула A2 + 1 (т.е. клетка A3 должна принять значение клетки A2, увеличенное на единицу). При копировании этой формулы в другие клетки, в них будет возникать не ссылка A2, а ссылка на клетку, расположенную выше, подобно тому как A2 есть клетка, расположенная над клеткой A3 - так называемое, «копирование с настройкой», значительно ускоряющее формирование ЭТ при решении многих задач.

Для того, чтобы ссылка не изменялась, используют абсолютный адрес. В этом адресе перед номером столбца и строки должен стоять знак $. Например, $A$2 - не перенастраивается ни номер столбца, ни номер строки; $A2 - не перенастраивается только номер столбца, номер строки остается относительным; A$2 - здесь относителен номер столбца.

Содержимым клетки может быть текст, повторяющийся текст и формула:

• формула используется для вычислений, строится из чисел, математических операторов и функций; длина формулы до 241 символа (частный вид формулы -число);

• текст - может содержать любой символ клавиатуры, его длина должна быть не более 241 символа, для введения текста набор следует начинать с символа S или кавычки (");

• повторяющийся текст - начинается с апострофа (') и распространяется на все свободные клетки строки (обычно используется для прочерчивания линий).

Важной особенностью ЭТ является тот факт, что как только в клетку внесено новое содержимое, автоматически происходит изменение значений всех клеток, содержащих на нее ссылку.

SC-4 поддерживает два вида математических операторов: арифметические и операторы от-

ношения. Арифметические операторы +, -, *, / имеют стандартное назначение, Операторы % (расчет процента), ** или ^ - возведение в степень. Операторы отношения: **<, >,** =, <=, >= (меньше, больше, равно, меньше или равно, больше или равно).

Имеются следующие виды функций: арифметические и тригонометрические, логические, календарные, статистические, специальные, финансовые, индексные.

После ввода символа "/" в первой и второй служебных строках появляется меню команд. Для выбора команды надо переместиться на ее имя и нажать клавишу ввода или ввести только первую букву ее имени. В ответ программа выводит полное имя команды:

/Blank -очистка клетки, группы клеток, всей таблицы или описания графиков;
/Insert - вставка пустых строк/колонок;
/View - визуализация данных в графическом режиме;
/Move - перенос-вставка существующих строк/колонок с указанной позиции;
/Global - задание общих режимов или режимов пересчета таблицы;
/eXecute - исполнение командного файла (xqt. файл);
/Copy - копирование содержания клеток или описания графиков;
/Zap -удаление таблицы и значения форматных характеристик из памяти**;**
/Load - загрузка таблицы или ее части с диска в рабочую область памяти;
/Window - установка режима «два окна»;
/Output - вывод отображения данных или содержимого клеток на экран, на диск или на печать;
/Edit - редактирование содержания клетки;
/Arrange - сортировка данных (строк, колонок);
/Title - фиксация заголовка и/или левых колонок таблицы,
/Delete - удаление колонки (строки), файла;
/Format - установление форматных характеристик отображения дан-ных на уровне клетки, строки, колонки или всей таблицы;
/Save - сохранение текущего содержания таблицы на диске;
/Quit - завершение сеанса работы с программой;
/Unprotect - снятие защиты клеток;
/Protect - установка защиты клеток;
/Name - задание имени для диапазона клеток;
//Data - предлагает дополнительные команды для работы с базой данных;
//Export - пересылка файлов из Суперкалка;
//Import - пересылка файлов в Суперкалк;
//Macro -создание макроопределений. Большинство команд имеют несколько уровней возможных ответов. После ввода буквы команды, вместо списка команд, появляется подсказка с вариантами ответов, допустимыми для этой команды.

**Пример: создание ЭТ «Штатное расписание»**

Общий вид создаваемой таблицы:

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | Школа №5 | | | | | | | |
| 2 | | | | | | | | | | | |
| 3 | ФИО | Стаж | Разряд | Разрядный коэффициент | Оклад. Руб. | Число часов | Всего начислено | Подоходный налог | Профсоюзные сборы | Всего удержано | К выдаче |
| 4 | | | | | | | | | | | |
| 5 | | | | | | | | | | | |
| 6 | Артемьева Т Н. | 5 | 10 | 3,30 | 86 | 20 | 124,22 | 14,91 | 0,12 | 15,03 | 109,19 |
| 7 | Бердышева А | 7 | 10 | 3,30 | 86 | 30 | 186,33 | 22,36 | 0,19 | 22,55 | 163,78 |
| 8 | Пришвина О.Н. | 9 | 11 | 3,40 | 88 | 21 | 133.47 | 16,02 | 0,13 | 16,15 | 117,32 |
| 9 | Веселова В А. | 16 | 12 | 3,45 | 89 | 24 | 154,27 | 18,51 | 0,15 | 18,67 | 135,60 |
| 10 | Николаева С.Ф. | 15 | 11 | 3,40 | 88 | 28 | 177,96 | 21,35 | 0,18 | 21,53 | 156,43 |
| 11 | Левина Е.А. | 10 | 12 | 3,45 | 89 | 18 | 115,70 | 13,88 | 0,12 | 14,00 | 101,70 |

Опишем порядок выполнения работы.

А. Оформление шапки таблицы.

1. Перемещаем курсор АК в клетку D1 и набираем с клавиатуры текст «Школа №5», нажимаем клавишу ввода.

2. В клетку А2 вводим повторяющийся текст:'_ . Он должен начинаться с апострофа, за ним указывается тот символ, который должен повторяться. В нашем случае это знак подчеркивания. Во второй строке проведена линия до правого края таблицы. Очистите клетки таблицы от линии, начиная со столбца L. Для этого введите в клетку L2 повторяющийся текст, содержащий пробел. Вернитесь к левому краю таблицы (Ctrl <== ).

3. В строки 3 и 4 введите заголовки столбцов, как показано в общем виде таблицы.

4. В пятой строке проведите линию, действуя аналогично п.2.

Б. Внесение данных.

1. Заполните столбец А. Так как по умолчанию ширина столбцов 9 символов, необходимо увеличить ширину столбца А. Выполните команду:

/F,CA,W,15.

Эта форма записи команды означает: нажатием клавиши «/» мы выходим в меню команд; клавишей F выбирается команда Format, нажатием С указывается область действия команды: Column (столбец); символ А автоматически появляется в командной строке, поскольку курсор АК находится в столбце А (или переведите его в этот столбец); далее выбирается опция Width (ширина) и указывается ширина столбца 15.

2. Внесите данные в столбцы B,C,D,F.

3. В клетку Е6 внесем формулу для расчета оклада: (D6+l)*20. Обратите внимание, что в этой клетке сразу появилось числовое значение этой формулы. Если же этого не произошло, значит при наборе была ошибка и формула воспринята как текст. Для исправления ошибки нажать F2, удалить признак текста - кавычки - и внести исправления в формулу.

4. Аналогично п.3 занесите формулу для расчета суммы начисления:

Е6*F6/18* 1.3 (эту формулу при желании можно сделать более точной).

5. В клетки Н6 и 16 введите формулы 12% G6 и 1 % G6, соответственно.

6. В клетку J6 внесите формулу Н6 + 16.

7. В клетку К6 внесите формулу G6 - J6.

8. Скопируйте формулу для расчета оклада из клетки Е6 в блок клеток командой

Е7:Е11 :/C,E6,E7:E11.

Далее процесс аналогичен.

Во всех клетках автоматически появились числовые значения формул. В первой служебной строке можно увидеть их вид. Заметьте, что при копировании произошла автоматическая настройка адреса D6 на D7, D8, D9 и т.д.

/C,G6:K6.G7:K11.

Рассмотрим на примере этой ЭТ применение команд Title, Global, Window, Arrange, а также запись и чтение с диска.

1. Вся таблица не вмещается на экран. Можно зафиксировать столбец А, тогда при движении к последним столбцам ЭТ, например к столбцу с суммой начисления, столбец с фамилиями будет служить границей и не исчезнет с экрана. Для этого сначала курсор АК поместите на столбец А, а затем выполните команду

/Title, Vertical.

Передвиньте курсор АК в столбец Н и убедитесь в правильности выполненных действии. Опция Clear снимает все титульные значки. Выполните команду /Title, Clear.

Для фиксации шапки таблицы, надо поместить курсор АК на 5-ю строку и выполнить команду

/Title. Horizontal.

Если поместить курсор АК в клетку А5 и выполнить команду

/Title,Both,

то фиксируется и шапка таблицы и столбец А.

2. Поместите курсор АК в клетку А12. Выполните команду Window, HorizontaI.

Экран разбился на два окна. Во втором окне при помощи, стрелок отобразите те же строки, что и в первом - в окнах можно высвечивать разные части одной и той же таблицы. Курсор АК сейчас находится во втором окне. Переход между окнами -. клавиша F6. Настройте второе окно на режим отображения формул:

/Global,Formula.

Обратите внимание, что некоторые опции команды высвечиваются желтым цветом - отключены соответствующие им режимы работы ЭТ. Выбор этих опций означает переключения с пассивного режима работы на активный и наоборот. В нашем случае включается режим отображения формул. По умолчанию установлено несинхронное перемещение информации на экране, т.е. информация, смещаемая в одном окне, остается неподвижной во втором. После выполнения команды

Window, Synchronize

установится синхронный режим смещения (Unsynchronize - несинхронный).

3. Запишите ЭТ на диск при помощи команды Save:/S . Во второй строке появляется запрос: Enter File Name (введите имя файла). Укажите имя файла, например PR1. Из опций A‖ (вся таблица). Values (без формул), Part (часть таблицы) выберите A‖. Назначение опций можно посмотреть при помощи клавиши-подсказки F1 в момент их высвечивания на экране. Команда записи на диск

/Save.PRlAll.

4. Команда /Zap удаляет всю таблицу из памяти. Загрузить таблицу с диска можно командой Load:

/Load.PRl, AH.

5. Сортировка данных в таблице производится командой Arrange. Опция Row означает, что по значениям указанной строки будут сортироваться столбцы, опция Column - сортировка производится между строками по значениям столбца.

Отсортируем строки таблицы по столбцу А, в котором находятся фамилии сотрудников, в соответствии с алфавитом. Выполните команду

/Arrange, Column, A,

но не нажимайте клавишу ввода. Обратите внимание на 2-ю служебную строку: "Enter Column; then <RETURN>, or **<,>** for Options" (введите колонку, затем ввод или запятая для опций). Вы должны нажать запятую. Это связано с тем, что не все строки нашей ЭТ должны сортироваться (в строках с 1-й по 5-ю находится шапка таблицы). На запрос "Enter Range" (введите область) укажите А6:К11 - можно воспользоваться удобствами режима Point. Далее из опций Ascending, Descend-

ing (по возрастанию, по убыванию) выберите Ascending. Из следующих опций Ajust, No-Ajust (с настройкой, без настройки формул) выберите опцию Ajust. Далее: Go. Общий вид команды

/Arrange, Column, A, A6:K 11, Ascending, Ajust, Go.

6. Отсортируем теперь таблицу по двум признакам: по убыванию разрядов, а внутри по возрастанию количества часов. Выполните команду:

/Arrange, Column, C, A6: K11, Descending, Adjust, Options, F, Ascending.

**Средства макропрограммирования**

Интегрированная система обработки электронных таблиц SC4 предоставляет пользователю средства макропрограммирования.

Макропрограммы хранятся, как правило, вместе с электронными таблицами и используются для автоматизации их обработки. Макропрограмма состоит из макросов. Макрос - самостоятельная структурная единица макропрограммы. Он обычно имеет имя и отделяется от остальных макросов пустыми клетками.

Макрос состоит из макрокоманд. С помощью макрокоманд можно записать любые действия пользователя, выполняемые в ручном режиме.

Работа с макросами включает в себя

• создание;
• поименование;
• запись на диск;
• отладку и исполнение.

Создавать макросы лучше всего непосредственно в ЭТ в обычном режиме ввода данных ENTRY.

Рекомендуется столбец А отводить для записи имен макросов, столбец В - для записи макрокоманд, а столбец С - для комментария, поясняющего действия макрокоманд.

Макрокоманды записывают в клетки как текст. Прописные и заглавные буквы не различают. Например,

/Ba1:a5~и/BA1:A5~

одна и та же команда. (Также как \a и \A - одно и то же имя или метка макроса). Значок ~ означает нажатие клавиши ввода. При вводе слэш-команд начинаем с символа ", причем вводим не полное название пункта меню, а выделенную букву.
Например,команда ввода

/Blank,al:a2

запишется в виде макрокоманды так:

"/Ba1:a2~.

*Пример.* Напишем макропрограмму, состоящую из трех макросов, которая будет выполнять следующие действия по обработке таблицы:

• очищать клетки a1:c1 таблицы;
• запрашивать значение переменной x в клетку a1;
• если значение x>0, то в клетке b1 выводить значение выражения x*0.25, а иначе в клетке c1 значение выражения x*.*0.25.

Наша макропрограмма в ЭТ будет выглядеть так:

| | A | B | C |
|---|---|---|---|
| 1 | | | |

| | | | |
|---|---|---|---|
| 2 | \a | /Bal:cl | -Очистка клеток a1:c1 |
| 3 | | {Getnumber "x=",al} | Ввод значения x в a1 |
| 4 | | {if al>0} {Branch \b} | По условию, переход в \b |
| 5 | | (Branch \c} | или в\c |
| б | | | |
| 7 | \b | {let blal *0.25} | b1=a1*0.25 |
| 8 | | | |
| 9 | \c | {letclal*al*0.25} | c1=a*a1*0.25 |
| 10 | | | |

В нашем примере три макроса \a, \b, \c отделяются друг от друга пустыми строками. Макрос с именем \a располагается в клетках b2:b5, макрос с именем \b в клетке b7, макрос с именем \c — в клетке b9. Выше описано создание макросов в режиме ENTRY. Кроме этого, макросы можно создавать в режимах LEARN и DIRECT. В режиме LEARN ваши действия автоматически записываются в виде макроса в LEARN — область, задаваемую командой

//Macro ,Learn, столбец.

Вход в этот режим осуществляется по нажатию клавиш Alt+F4. (Выход - повторное нажатие тех же клавиш.) Макрос, созданный в этом режиме, имеет большие размеры и сложен для восприятия и редактирования. Режим DIRECT (вход -Alt+F6, выход - повторное нажатие) является промежуточным между режимами ENTRY и LEARN.

Имена макросов лучше всего начинать с символа \ и далее одна из букв латинского алфавита. Макрос тогда очень просто запускается на выполнение: одновременное нажатие клавиш Alt+A запускает на выполнение макрос \a, Alt+B - макрос \b и т.д.

Чтобы текст в клетках столбца А воспринимался как имена (или метки) макросов, необходимо дать команду

- /Name, Labels, Right, A.

Эта команда назначает имена макросам, расположенным справа от столбца А. Макросы лучше всего записывать вместе с ЭТ по команде

/Save,имя ЭТ,A11.

Кроме этого, существует возможность записывать файлы с макросами в формате ASCII. В этом случае используется команда

//Macro,Write.

Эти файлы имеют стандартное расширение .xqt и могут создаваться в текстовых редакторах. Метки, макросы и комментарии записываются, тогда в один столбец. Первая строка файла должна содержать имя {Macro}. Эти файлы могут быть запущены на исполнение из SC по команде

//Macro.eXecute.имя.

Читаются такие файлы в SC по команде

//Macro,Read.

**Графическое представление данных**

Электронные таблицы имеют развитые возможности представления данных в графическом виде.

Рассмотрим принципы построения диаграмм на примере ЭТ «Показатели соревнования

между факультетами института».

Пусть пять факультетов соревнуются по следующим пунктам:

• % успеваемости (отношение числа студентов, сдавших сессию без двоек, к общему числу студентов);

• % качества знаний (сдавшие на 4 и 5 к общему числу студентов);

• участие в студенческих конференциях (0,2 балла за каждого студента);

• число печатных работ (0,5 балла за каждую работу).

Порядок работы.

1. Оформите «шапку» ЭТ.

2. Заполните данными столбцы B, C, D, E, F.

3. В клетку G5 внесите формулу C5/B5 и скопируйте ее в диапазон G5:G9 В клетку H5 внесите D5/B5 и скопируйте в H5:H9. В клетку 15 введите формулу (G5+H5+E5*0.2+F5*0.5) и скопируйте в 15:19.

4. Построим столбиковую диаграмм) по столбцу «Общий балл». Для этого введем команду /View. Появится основное меню команды:

Show Data Graph-Type Time-Labs Var-Labs Point-Labs Headings Options

Из этого меню производится настройка всех параметров диаграммы Обязательными параметрами являются тип диаграммы (Graph-Type) и данные (Data) Выберите тип диаграммы, войдя в режим Graph-Type и выбрав тип Bar (столбиковая). Определите данные, которые будут выводиться в диаграмме. Для этого войдите в режим Data - появится запрос на ввод первой переменной (Var A). Отметьте диапазон клеток 15:19. Выведите диаграмму на экран, выбрав пункт Show.

Нажатие клавиш Enter или Esc - возврат в ЭТ**.**

Без поясняющих меток диаграмма малопонятна. Вновь войдите в меню команды /View и выполните дополнительную настройку параметров:

| | |
|---|---|
| Headings MainAl | -задание общего заголовка диаграммы из клетки A1; |
| X-axis A3 | - задание заголовка для оси X из клетки A3; |
| Quit | - возврат в меню команды /View; |
| Time-Labs A5:A9 | - установка меток по оси X; |
| Var-Labs 15 | - задание метки для (первой) переменной. |

Выведите диаграмму на экран.

5. Построим столбиковую диаграмму по двум переменным (Var A и Var B). Например, по столбцам B и C.

Определим тип диаграммы и номер:

/View, 2, Graph-Type, Bar

Теперь определим данные, которые будут использоваться в диаграмме. Входим в Data, отмечаем диапазон клеток B5:B9 для первой переменной (Var A), после чего вводим (,), тем самым давая понять, что диаграмма будет строиться по двум переменным. Появляется запрос на ввод 2-й переменной (Var B). Укажите диапазон C5:C9. Выведите диаграмму на экран.

## 7.3. ЭЛЕКТРОННЫЕ ТАБЛИЦЫ EXCEL

Современные электронные таблицы типа Excel используют манипулятор «мышь»; в них реализован удобный и комфортный интерфейс.

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

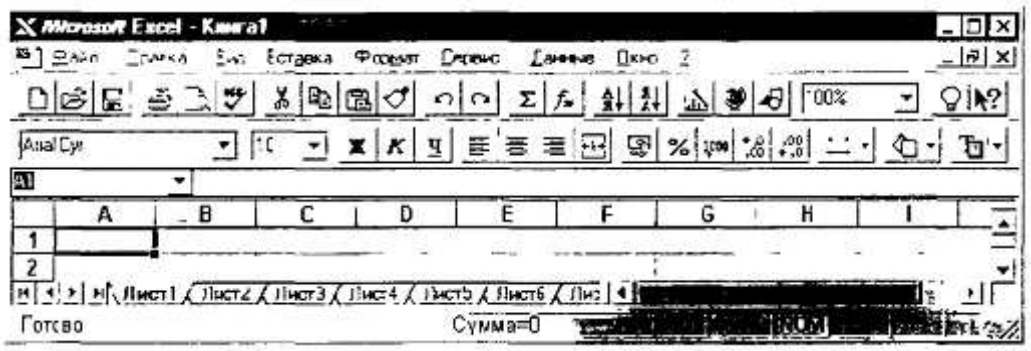



*Рис. 2.26.* Экран Excel

В окне Excel (рис. 2.26), как и в других программах пакета MicroSoft Office, под зоной заголовка находится область заголовков меню. Чуть ниже находится основная линейка инструментов.

Кнопки линейки инструментов позволяют быстро и легко вызывать различные функции Excel. Их можно вызывать также через меню.

Если читатель уяснил идеологию электронных таблиц, то он может смело приступать к работе с Excel.

Итак, данные подлежащие обработке размещаются на нескольких листах книги Excel. Введите числа и текст какой-нибудь сметы в ячейки первого листа книги, попытайтесь красиво оформить данные, а затем представьте их в виде диаграммы (попытайтесь!). Далее сохраните данные и напечатайте. У вас должно все получиться!

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

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

Создание формулы начинается с ввода знака равенства (=). Формула содержит встроенные функции, адреса ячеек, константы. В случае затруднений с формированием формулы используйте *Мастер функции.* Есть великолепная команда *Авто-суммирование.* Выделите столбец или строку данных (и вообще любой диапазон ячеек) и нажмите кнопку «Автосуммирование».

Подобный сервис есть и при оформлении дизайна таблицы. Вручную форматирование проводят стандартным способом. Выделяют ячейку или диапазон, а затем выбирают команды из контекстного меню форматирования. Автоформат позволяет оформить сразу весь текущий лист по шаблону, который выбирается командой *Автоформат* в меню *Формат.*

Перед печатью таблиц (кнопка *Печать)* удобно осуществить предварительный просмотр (соответствующая кнопка в меню *Сохранить).*

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

Для отображения числовых данных в графической форме используют линии, полосы, столбцы, сектора и другие маркеры, а также их объединенные вариации. Для размещения диаграммы рядом с данными создают *внедренную диаграмму* на том же листе. Можно создать диаграмму на отдельном *листе диаграммы.* Выделите нужный диапазон ячеек, содержащих данные, которые следует представить на диаграмме. Выберите команду *Диаграмма* в меню *Вставка,* а затем команду *На новом листе.* Далее следуйте указаниям *Мастера диаграмм.*

Если нет времени, желания и возможности для относительно сложных построений, используйте автоматическое оформление диаграмм с помощью команды *Автоформат* в меню *Формат.*

Помимо того, что имеется большая встроенная библиотека построения графических образов: графиков, диаграмм, гистограмм. Excel содержит мощный встроенный графический редактор. Принципы работы графического редактора аналогичны подобным системам, описанным в разд.

2.5.

Большое внимание в Excel уделено оформлению книги, ее листов. Для этих целей используют формат текстов и чисел, цвета н заливки, стиль, шрифты и т.д. По сути, графический редактор сопряжен с текстовым процессором, близким по возможностям с издательскими системами типа Word, о котором рассказано в разд. 2.4.

Excel не только «дружен» с текстовыми и графическими системами, но и поддерживает основные действия, характерные для систем управления базами данных (СУБД). В этом смысле современные электронные таблицы (Excel, QuattroPro, Lotus) являются интегрированными программными системами. Более того, у них развит аппарат импортирования и экспортирования данных из других программных систем.

Эти и многие другие приятные сервисные возможности, порой неожиданные, можно для себя открыть при работе с Excel.

***Контрольные вопросы и задания***

1. Создайте ЭТ «Стипендиальная ведомость».

2. Составьте смету расходов для организации турпохода, если известна общая сумма затрат. Постройте различные виды диаграмм и подготовьте отчет.

3. Составьте компьютерную модель Солнечной системы в электронных таблицах. Считая, что планеты движутся вокруг Солнца по окружностям с постоянной скоростью, определите скорость движения по орбите для каждой планеты. Например, для Марса V = 2·3,14-0,387/0,24 = 10,1(км/ч). Оцените в каких пределах может меняться расстояние от Земли до Марса. При каких расположениях планет достигаются наименьшее и наибольшее значения.

4. Пусть интервал движения автобуса составляет 10 мин. Среднее время ожидания автобуса можно оценить проведя *N* опытов, разыгрывая случайное число в интервале [0, 10]. Найдите среднее значение для серий из 10, 50, 100 опытов. В случае двух маршрутов найдите среднее значение ожидания, когда интервал движения первого автобуса составляет 10 минут, а второго 60 минут. Постройте вычислительную таблицу для трех маршрутов.

# §8. ИНТЕГРИРОВАННЫЕ ПРОГРАММНЫЕ СРЕДСТВА

## 8.1. ПРИНЦИПЫ ПОСТРОЕНИЯ ИНТЕГРИРОВАННЫХ ПРОГРАММНЫХ СИСТЕМ

Программные средства, подробно описанные в предыдущих разделах - системы подготовки текстов и машинной графики, базы данных и электронные таблицы -зачастую не могут удовлетворить запросов пользователей в силу того, что бывает необходимо использовать возможности каждого из них одновременно, в комплексе.

Типичной является ситуация, когда данные, полученные из базы данных, необходимо обработать средствами табличного процессора, представить графически, в виде диаграммы того или иного вида, а затем вставить в текст. Для выполнения работ такого типа существуют, так называемые, интегрированные пакеты - программные средства, совмещающие возможности, характерные в отдельности для текстовых редакторов, графических систем, электронных таблиц, баз данных (и других программных средств). Конечно, такое совмещение возможностей достигается за счет компромисса. Некоторые возможности оказываются в интегрированных пакетах ограниченными или реализованными не в полной мере. Это касается, в первую очередь, богатства команд обработки базы данных и электронной таблицы, их размеров, макроязыков. Однако преимущества, создаваемые единым интерфейсом объединенных в интегрированном пакете программных средств, неоспоримы. Многие ведущие мировые фирмы, выпускающие программное обеспечение, создали и продолжают развивать свои интегрированные пакеты. Так, фирма «Microsoft» развивает интегрированный пакет Works, известны пакеты Open Access фирмы «Open Access», FrameWork фирмы «Ashton-Tate», Lotus 1-2-3 и Symphony фирмы «Lotus Development Corporation».

Системы Symphony, KnowledgeMan позволяют рассматривать элементы записей в БД, тексты и ячейки электронной таблицы как единое целое: сохраняя на экране одни и те же данные, можно просто как бы менять на них точку зрения, переходя из электронной таблицы в редактор и т.п. В системах предусмотрен многооконный интерфейс: при работе пользователь, переходя из

одного окна в другое, меняет «среду», выполняет операции поиска в БД, использует редактор текста и т.д.

Необычно построение системы FrameWork. Все компоненты системы рассматриваются как фреймы - упорядоченные наборы информации, причем пользователь может дробить фреймы на набор фреймов более низкого порядка и т.п. Например, в поле текста можно вставить метку, указывающую на то, что в этом месте должен располагаться участок данных из электронной таблицы, график и пр. Эти фреймы можно будет описать позже, а затем «собрать» все фреймы в единый документ. В системе имеется свой язык программирования Fred.

В интегрированную систему «Мастер» отечественной разработки включены текстовый процессор «Лексикон», процессор электронных таблиц, БД. графический процессор. Кроме того, есть возможность писать программы на языке «Мастер». В этот язык включены функции работы с ячейками электронных таблиц, окнами экрана (рамками), функциями, определенными внутри подсистем. Таким образом, с помощью системы «Мастер» существует возможность сформировать свою версию интегрированной системы, поддерживающую необходимые в данном конкретном приложении функции. Система «Мастер» является примером разумно построенной интегрированной системы, работа с ней естественна и удобна. С помощью этой системы ведется разработка различных информационных систем.

Ограничимся рассмотрением одной из популярных в настоящее время интегрированных систем - Works. По мнению специалистов, этот пакет обладает наиболее наглядным и простым в освоении интерфейсом, полно реализующим основные функции обработки текстов, таблиц и баз данных, а также телекоммуникации по коммутируемым каналам.

## 8.2. ИНТЕГРИРОВАННЫЙ ПАКЕТ MS-WORKS

После запуска интегрированного пакета MS-Works вы попадаете в интегрированную среду этого программного средства. Опишем некоторые правила работы с составными компонентами пакета.

**Работа с текстовым редактором системы MS-Works.** Текстовый редактор MS-Works является составной частью интегрированного пакета MS-Works. Чтобы начать работу с текстовым редактором, выполняются следующие действия.

Нажатие на клавишу Alt - переход в главное меню. При этом опция *Файл* оказывается выделенной. Нажатие на клавишу Enter- подтверждение выбора и открытие подменю работы с файлами *Файл.* Если вы создаете новый текст, выберите в этом меню команду *Создать Фаил* и нажмите клавишу Enter. В этом случае программа выдаст запрос о том, какой файл следует создать. Выберите из нового открывшегося меню пункт, относящийся к текстовому процессору.

Если текст уже был создан ранее, выберите в меню *Фаил* команду *Открыть файл* и введите имя файла или выберите его из списка имеющихся на диске. Выбор файла из списка ничем не отличается от выбора опции из меню.

Для выбора файла можно использовать мышь или клавиши управления курсором, затем нужно нажать клавишу Enter. В этом случае на экране появляется интегрированная среда текстового редактора с уже загруженным текстом.

Подобным образом выбираются и другие команды меню.

Экран текстового редактора разделен на несколько частей. Вверху экрана находится меню. Оно содержит команды, которые используются для редактирования текста и работы с файлами. Внизу экрана - строка сообщений. Сверху над строкой сообщений - статусная строка. В верхней части экрана под меню имеется линейка. Она определяет границы текста и красную строку. Левая и правая границы текста обозначаются квадратными скобками. Остальная площадь экрана свободна. Это - рабочая область, в которой можно набирать и редактировать текст. На экране постоянно находятся три важных отметки - курсор, буква «п», указывающая начало страницы, и ромбик - метка конца файла. Он не может быть удален с экрана.

При вводе нового текста текстовый редактор сам разместит его в границах, определенных при помощи линейки. Когда текст доходит до конца строки, слово переносится на следующую строку.

Текстовый редактор системы Works использует стандартные для всех редакторов приемы работы с текстом. Если вы недовольны результатами редактирования, изменения легко отменить. Для этого перейдите в меню *Редактирование* (нажмите клавишу Alt, укажите выделенной строкой

команду *Редактирование* и нажмите Enter), выберите *Отмена.* Нажмите Enter (или «О»). После этого текст появится на экране в первоначальном виде.

Для того. чтобы сохранить текст на магнитном диске, надо сделать следующее:

• нажать одновременно Alt и Ф (так вы откроете меню работы с файлами *Файл);*

• нажать клавишу С (таким образом вы выберете команду *Сохранить);*

• придумать и ввести имя для вашего файла, в котором будет храниться подготовленный вами текст.

Для выхода из текстового редактора следует нажать Alt и открыть меню работы с файлами *Фаил* (нажав клавишу Ф), выбрать команду *Выход* и нажать Enter (можно нажать клавишу Х).

Текстовый редактор интегрированной системы Works имеет встроенные средства для проверки правописания - спеллинга. Чтобы проверить ошибки в тексте, необходимо поместить курсор в том месте, откуда надо начать проверку, или выделить текст, который надо проверить.

**Работа с электронными таблицами системы MS-Works.** Электронные таблицы MS-Works являются составной частью интегрированного пакета MS-Works. Для начала сеанса работы с электронной таблицей вызовите MS-Works. Чтобы начать работу с электронными таблицами, выполните следующие действия.

Нажмите клавишу Alt для перехода в главное меню. При этом пункт *Файл* оказывается выделенным. Нажмите клавишу Enter, чтобы открыть меню работы с файлами *Файл.*

Если вы создаете новую таблицу, выберите в этом меню команду *Создать Файл* с помощью клавиш со стрелками. В этом сл\чае программа выдаст запрос о том, какой файл следует создать. Выберите из открывшегося меню пункт, относящийся к электронным таблицам.

Если таблица уже была создана ранее, выберите в меню *Файл* команду *Открыть файл с* помощью клавиш со стрелками и введите имя файла или выберите его из списка имеющихся на диске. Выбор файла из списка ничем не отличается от выбора опции из меню. Для выбора файла вы также используете клавиши со стрелками, затем нажимаете клавишу Enter. В этом случае вы получите на экране электронную таблицу с уже загруженными данными.

Посмотрим, на какие части разделен экран при работе с электронными таблицами. Вверху экрана находится меню. Оно содержит команды, которые используются для работы с электронными таблицами. Для перехода в меню надо нажать клавишу Alt. Строкой ниже находится строка ввода, в которой отображается содержимое активной клетки. Внизу экрана - строка сообщений. Она дает подсказки и описание команд в меню. Сразу над строкой сообщений находится статусная строка, позволяющая следить за ходом работы. Например, она содержит имя активной клетки. Остальная площадь экрана занята электронной таблицей.

После загрузки пустой таблицы на экране видны 7 столбцов (с А до G) и строки с 1-й по 18-ю. Перемещение курсора по таблице осуществляется при помощи клавиш со стрелками. Для того, чтобы ввести в клетку информацию (текст, число, формулу), надо сделать следующее:

• указать клетку таблицы, установив на нее указатель с помощью клавиш управления курсором или мыши;

• набрать с помощью буквенно-цифровых клавиш текст, число, дату, формулу;

• нажать клавишу Enter, чтобы подтвердить ввод данных; если в клетке уже содержатся данные, они будут замещены введенными.

Для отмены ввода информации следует нажать Esc.

С отдельными клетками и с блоками клеток можно выполнять разнообразные операции (перемещение, копирование, очистку). В этих случаях требуется сначала выделить блок клеток, чтобы произвести с ним какие-то операции. Для того, чтобы выделить целый блок клеток, выполняется следующая последовательность
действий:

• курсор таблицы помещается в начало выделяемого блока (его левый верхний угол);

• из меню выбирается команда *Выделить,* а затем *Ячейки* (или нажимается клавиша F8);

• выделенная область расширяется клавишами со стрелками до необходимого размера.

После этого выделенные данные можно изменить, выбрав из меню *Редактирования* одну из команд: *Переместить, Копировать, Очистить.*

Вычисления в электронных таблицах выполняются с помощью формул. Формула **может** содержать обозначения клеток таблицы, числа, знаки арифметических действий, скобки, определяющие порядок действий, имена функций. Формула начинается со знака равенства (=).

Данные из электронной таблицы могут быть представлены в виде графиков, столбцовых

или круговых диаграмм. На диаграммах легче увидеть имеющиеся зависимости между данными. Изменение данных в электронной таблице отражается на диаграмме. Чаще всего ИСПОЛЬЗУЮТСЯ следующие виды диаграмм:

• линейный график - значения представляются в виде точек, соединенных линиями;

• столбцевая диаграмма - каждое число представлено на диаграмме столбиком.

• круговая диаграмма - значения представляются секторами круга.

**Работа с СУБД системы MS-Works.** Система управления базами данных MS-Works является составной частью интегрированного пакета MS-Works. Для начала работы с базами данных выполняются следующие действия. Нажатие клавиши Alt - переход в главное меню. При этом опция *Фаил* оказывается выделенной. Нажатие клавиши Enter - подтверждение выбора и открытие подменю *Файл* работы с файлами.

Если создается новая база данных, то в этом меню выбирается команда *Создать файл.* В этом случае программа выдаст запрос о том, какой файл следует создать. Из открывшегося меню следует выбрать пункт *Новый файл в формате базы данных.*

Если база данных уже была создана ранее, выберите в меню Файл команду Открыть *файл* и введите имя файла или выберите его из списка имеющихся на диске. Выбор файла из списка ничем не отличается от выбора опции из меню. Для выбора файла также используются клавиши со стрелками. После выбора на экране задействована интегрированная среда с уже загруженным файлом базы данных.

Экран при работе с базами данных разделен на зоны. Вверху экрана находится меню. Оно содержит команды, которые используются для работы с базами данных.

Для перехода в меню надо нажать клавишу Alt. Ниже расположена строка ввода, в которой отображается содержимое активного поля. Активным называется поле, в котором находится курсор. Оно выделено цветом. Перемещение курсора по полям осуществляется при помощи клавиш со стрелками.

Внизу экрана находится строка сообщений. Она содержит подсказки и описание команд в меню. Сразу над строкой сообщений - статусная строка. Она позволяет следить за ходом работы. Например, она содержит номер активной записи, имя активного поля этой записи, количество выведенных на экран записей из общего числа записей в базе данных.

Остальная площадь экрана свободна. Это рабочая область, в которой происходит работа с базой данных. Для того, чтобы создать базу данных, надо выполнить следующие действия:

• переместить курсор в то место экрана, где должно находится поле;

• ввести имя поля, после него поставить двоеточие (:);

• нажать клавишу Enter.

Вводятся также значения ширины (число символов) и высоты (число строк) поля. Например, если это поле, в которое будут вводиться фамилии учащихся класса, то его ширина выбирается такой чтобы поместилась самая длинная фамилия, а высота задается в одну строку.

После нажатия клавиши Enter имя поля появится на экране, а справа от имени создается само поле. Задав имена и размеры всех полей базы данных, определим форму базы данных.

После того, как создана форма базы данных, в каждое поле можно занести данные. Содержимым поля может быть текст, число или дата. Можно ввести также формулу. Для того, чтобы ввести данные, необходимо указать курсором нужное поле. Затем можно набирать текст, число или дату. Нажатие клавиши Enter завершает ввод.

Если необходимо ввести данные и одновременно переместиться в следующее поле, следует нажать Tab.

Если необходимо переместиться в предыдущее поле, нажмите одновременно Shift+Tab. Если после ввода данных в последнее поле записи нажать клавишу Tab, происходит переход к следующей записи. Для перехода к следующей записи также можно использовать сочетание клавиш Ctrl+PgDn.

Для того, чтобы отредактировать поле или имя поля, сделайте следующее:

• укажите курсором нужное поле;

• нажмите клавишу F2, при этом в строке ввода появится содержимое поля;

• внесите необходимые изменения с помощью буквенно-цифровых клавиш, клавиш управления курсором. Back Space и Delete;

• нажмите клавишу Enter.

Все действия с базой данных подобны действиям со специализированными системами управления базами данных.

Одной из наиболее важных возможностей, предоставляемых системой управления базами данных, является поиск записей, удовлетворяющих заданному условию. Для нахождения записи, поля которой содержат заданный текст, выполняются следующие действия:

• переход в меню *Просмотр* (вариант: нажатие Alt и «М»), выбор команды *Таблица;*

• переход в меню *Выделить* (вариант: нажатие Alt и «В») и выбор во вспомогательном меню команды *Поиск;*

• в окно *Что искать* вводится текст, который надо найти.

Часто бывает необходимо выделить из базы данных и наиболее наглядно представить необходимую информацию. Для этого можно создать отчет, сгруппировав данные и подведя итог. Действия при создании отчета таковы:

• переход в меню *Просмотр* (вариант: нажатие Alt и «М»), выполнение команды *Новый Отчет;*

• задание структуры отчета путем внесения необходимых изменений в стандартную форму с помощью буквенно-цифровых клавиш, клавиш управления курсором, Back Space и Delete.

Стандартная форма включает в себя две пустые строки для задания заголовка отчета, две пустые строки для задания заголовков столбцов; ниже располагаются строки записей. В нижней части стандартной формы находятся две пустые строки для вывода итога. В них можно ввести значения стандартных статистических формул СУММ (сумма), СРЧ (среднее значение), MAX (наибольшее значение), MIN (наименьшее значение).

Если в отчет включаются только записи, удовлетворяющие заданному условию поиска, то после задания структуры отчета выполняются следующие действия:

• возврат в меню *Просмотр* (вариант: нажатие Alt и «М»), выбор команды *Запрос;*

• ввод необходимых условий поиска в появившуюся на экране форму запроса;

• возврат в меню *Просмотр* (вариант: нажатие Alt и «М»), выбор команды *Отчет.*

Иногда бывает нужно вставить информацию из базы данных в документ текстового редактора. Например, если рассылается много одинаковых писем, то адреса можно вставлять из базы данных. Можно вставлять также и отчеты. Для того, чтобы перенести информацию в текст, следует сделать следующее:

• перейти в меню *Просмотр* (вариант: нажать Alt и «М»), выбрать команду *Таблица;*

• перейти в меню *Выделить* (вариант: нажать Alt и «В»), на экране появляется вспомогательное меню (первые четыре строки которого указывают возможные выделяемые области);

• выделить информацию, которую надо скопировать, перейти в меню *Редактирование* (вариант: нажать Alt и «Р») выполнить команду *Копировать;*

• выбрать из открывшегося списка файлов тот, в который необходимо скопировать информацию. После того, как этот файл появился на экране, следует поместить курсор в то место, где надо расположить копию.

Аналогичные действия справедливы при одновременном использовании данных в различных средствах интегрированной системы: текстовом, графическом редакторах, электронных таблицах и СУБД.

***Контрольные вопросы и задания***

1. Что называется интегрированным пакетом?

2. В чем достоинства и недостатки интегрированных пакетов по сравнению со специализированными инструментальными программными средствами?

3. Разработайте информационную систему «Каталог программных продуктов» и подготовьте следующие документы:

а) базы данных лицензионных и свободно распространяемых программных средств;

б) прайс-лист программных продуктов фирмы «Microsoft»;

в) рекламные буклеты по отдельным программным продуктам.

## § 9. ЭКСПЕРТНЫЕ СИСТЕМЫ

Систему искусственного интеллекта, построенную на основе глубоких специальных знаний о некоторой предметной области (полученных от экспертов-специалистов этой области), называют **экспертной системой.** Экспертные системы -один из немногих видов систем искусственного интеллекта (см. гл.1), которые получили широкое распространение и нашли практическое применение. Существуют экспертные системы по военному делу, геологии, инженерному делу, информатике, космической технике, математике, медицине, метеорологии, промышленности, сельскому хозяйству, управлению, физике, химии, электронике, юриспруденции и т.д. И только то, что экспертные системы остаются весьма сложными, дорогими, а главное, узкоспециализированными программами, сдерживает их еще более широкое распространение.

Особенности экспертных систем:

• компетентность - в конкретной предметной области экспертная система должна достигать того же уровня, что и специалисты-люди; при этом она должна пользоваться теми же эвристическими приемами, также глубоко и широко отражать предметную область;

• символьные рассуждения - знания, на которых основана экспертная система, представляют в символьном виде понятия реального мира, рассуждения также происходят в виде преобразовании символьных наборов;

• глубина - экспертиза должна решать серьезные, нетривиальные задачи, отличающиеся сложностью знаний, которые экспертная система использует, или обилием информации; это не позволяет использовать полный перебор вариантов как метод решения задачи и заставляет прибегать к эвристическим, творческим, неформальным методам;

• самосознание - экспертная система должна включать в себя механизм объяснения того, каким образом она приходит к решению задачи.

Экспертные системы создаются для решения разного рода проблем, но они имеют схожую структуру (рис. 2.27); основные типы их деятельности можно сгруппировать в категории, приведенные в табл. 2.5.


*Рис. 2.27.* Схема обобщенной экспертной системы

Экспертные системы, выполняющие *интерпретацию,* как правило, используют информацию от датчиков для описания ситуации. Например, это может быть интерпретация показаний измерительных приборов на химическом заводе для определения состояния процесса. Интерпретирующие системы имеют дело не с четкими символьными представлениями проблемной ситуации, а непосредственно с реальными данными. Они сталкиваются с затруднениями, которых нет у систем других типов, потому что им приходится обрабатывать информацию «зашумленную», недостаточную, неполную, ненадежную или ошибочную. Им необходимы специальные методы регистрации характеристик непрерывных потоков данных, сигналов или изображений и методы их символьного представления.

**Таблица 2.5 Типичные категории способов применения экспертных систем**

| Категория | Решаемая проблема |
|---|---|

| | |
|---|---|
| Интерпретация | Описание ситуации по информации, поступающей от датчиков |
| Прогноз | Определение вероятных последствий заданных ситуаций |
| Диагностика | Выявление причин неправильного функционирования системы по наблюдениям |
| Проектирование | Построение конфигурации объектов при заданных ограничениях |
| Планирование | Определение последовательности действий |
| Наблюдение | Сравнение результатов наблюдений с ожидаемыми результатами |
| Отладка | Составление рецептов исправления неправильного функционирования системы |
| Ремонт | Выполнение последовательности предписанных исправлений |
| Обучение | Диагностика и исправление поведения обучаемого |
| Управление | Управление поведением системы как целого |

Интерпретирующие экспертные системы могут обработать разнообразные виды данных. Например, система анализа сцен и распознавания речи, используя естественную информацию (в одном случае визуальные образы, в другом - звуковые сигналы), анализирует их характеристики и понимает их смысл. Интерпретация в области химии использует данные дифракции рентгеновских лучей, спектрального анализа или ядерного магнитного резонанса для вывода химической структуры веществ. Интерпретирующая система в геологии использует каротажное зондирование - измерение проводимости горных пород в буровых скважинах и вокруг них, чтобы определить подповерхностные геологические структуры. Медицинские интерпретирующие системы, основываясь на показаниях следящих систем (например, значениях температуры, пульса, кровяного давления), устанавливают диагноз или тяжесть заболевания. В военном деле интерпретирующие системы, получая данные от радаров, радиосвязи и сонарных устройств, оценивают ситуацию и идентифицируют цели.

Экспертные системы, осуществляющие *прогноз,* определяют вероятные последствия заданных ситуаций. Примерами служат прогноз ущерба урожаю от некоторого вида вредных насекомых, оценивание спроса на нефть на мировом рынке, прогнозирование места возникновения следующего вооруженного конфликта на основании данных разведки. Системы прогнозирования иногда используют имитационное моделирование, т.е. программы, которые отражают причинно-следственные взаимосвязи в реальном мире, чтобы сгенерировать ситуации или сценарии, которые могут возникнуть при тех или иных входных данных. Возможные ситуации вместе со знаниями о процессах, порождающих эти ситуации, образуют предпосылки для прогноза. Специалисты по искусственному интеллекту пока что разработали сравнительно мало прогнозирующих систем, возможно потому, что очень трудно взаимодействовать с имитационными моделями и создавать их.

Экспертные системы выполняют *диагностирование,* используя описания ситуаций, характеристики поведения или знания о конструкции компонентов, чтобы установить вероятные причины неправильно функционирующей диагностируемой системы. Примерами служат определение причин заболевания по симптомам, наблюдаемым у пациентов; локализация неисправностей в электронных схемах и определение неисправных компонентов в системе охлаждения ядерных реакторов. Диагностические системы часто являются консультантами, которые не только ставят диагноз, но и помогают в отладке. Они могут взаимодействовать с пользователем, чтобы оказать помощь при поиске неисправностей, а затем предложить порядок действий по их устранению. Медицина представляется вполне естественной областью для диагностирования, и действительно, в медицинской области было разработано больше диагностических систем, чем в любой другой отдельно взятой предметной области. Однако в настоящее время многие диагностические системы разрабатывают для приложений к инженерному делу и компьютерным системам.

Экспертные системы, выполняющие *проектирование,* разрабатывают конфигурации объектов с учетом набора ограничений, присущих проблеме. Примерами могут служить генная инженерия, разработка СБИС н синтез сложных органических молекул.

Экспертные системы, занятые *планированием,* проектируют действия; они определяют полную последовательность действий, прежде чем начнется их выполнение. Примерами могут служить создание плана применения последовательности химических реакций к группам атомов с целью синтеза сложных органических соединений или создание плана воздушного боя с целью нейтрализации определенного фактора боеспособности врага.

Экспертные системы, выполняющие *наблюдение,* сравнивают действительное поведение с ожидаемым поведением системы. Примерами могут служить слежение **за** показаниями измерительных приборов в ядерных реакторах с целью обнаружения аварийных ситуаций или оценка данных мониторинга больных, помещенных в блоки интенсивной терапии. Наблюдающие экспертные системы сравнивают наблюдаемое поведение с набором допустимых ситуаций нормального поведения. Наблюдающие экспертные системы по самой своей природе должны работать в режиме реального времени и осуществлять зависящую как от времени, так и от контекста интерпретацию поведения наблюдаемого объекта.

Экспертные системы, выполняющие *обучение,* подвергают диагностике, «отладке» и исправлению (коррекции) поведение обучаемого. В качестве примеров приведем обучение студентов отысканию неисправностей в электрических цепях, обучение военных моряков обращению с двигателем на корабле и обучение студентов-медиков выбору антимикробной терапии. Обучающие системы создают модель того, что обучающийся знает и как он эти знания применяет к решению проблемы. Системы диагностируют и указывают обучающемуся его ошибки, анализируя модель и строя планы исправлений указанных ошибок. Они исправляют поведение обучающихся, выполняя эти планы с помощью непосредственных указаний обучающимся.

Экспертные системы, осуществляющие *управление,* адаптивно руководят поведением системы в целом. Примерами служат управление производством и распределением компьютерных систем или контроль за состоянием больных при интенсивной терапии. Управляющие экспертные системы должны включать наблюдающие компоненты, чтобы отслеживать поведение объекта на протяжении времени, но они могут нуждаться и в других компонентах для выполнения любых или всех из уже рассмотренных типов задач: интерпретации, прогнозировании, диагностики, проектировании, планировании, отладки, ремонта и обучения. Типичная комбинация задач состоит из наблюдения, диагностики, отладки, планирования и прогноза.

Рассмотрим примеры наиболее известных классических экспертных систем, с которых началось создание и развитие этого типа программных средств.

**MYCIN** - это экспертная система, разработанная для медицинской диагностики. В частности, она предназначена для работы в области диагностики и лечения заражения крови и медицинских инфекций. Система ставит соответствующий диагноз, исходя из представленных ей симптомов, и рекомендует курс медикаментозного лечения любой из диагностированных инфекций. Она состоит в общей сложности из 450 правил, разработанных с помощью группы по инфекционным заболеваниям Стэнфордского университета. Ее основополагающим моментом является использование вероятностного подхода.

Система MYCIN справляется с задачей путем назначения показателя определенности каждому из своих 450 правил. Поэтому можно представлять MYCIN как систему, содержащую набор правил вида «ЕСЛИ... , ТО» с определенностью *P.* В случае MYCIN их предоставили люди-эксперты, которые изложили и правила, и указали свою степень доверия к каждому правилу по шкале от 1 до 10. Установив эти правила и связанные с ними показатели определенности, MYCIN идет по цепочке назад от возможного исхода, чтобы убедиться, можно ли верить такому исходу. Установив все необходимые исходные предпосылки, MYCIN формирует суждение по данному исходу, рассчитанное на основе показателей определенности, связанных со всеми правилами, которые нужно использовать.

Допустим, чтобы получить исход Z, требуется определить предпосылки *X* и *Y*, дающие возможность вывести Z. Но правила для определения *X* и *Y* могут иметь связанные с ними *Показатели определенности P* и *O* . Если значения *P* и *Q* были равны 1,0, то исход Z не вызывает сомнения. Если *P* и *Q* меньше 1,0 (как это обычно бывает), то исход Z не последует наверняка. Он может получиться лишь с некоторой степенью определенности.

MYCIN не ставит диагноз и не раскрывает его точный *Показатель неопределенности.* Система выдает целый список диагнозов, называя *Показатель определенности* для каждого из них. Все диагнозы с показателями выше определенного, специфического для каждого диагноза уровня, принимаются как в той или иной степени вероятные, и пользователю вручается список возможных исходов.

Стандартные фразы и грамматические формы были без труда приспособлены к программе, и в результате получился существенно вырожденный диалект английского языка, легко поддающийся программированию. Врачи оказались очень довольными таким результатом, потому что,

сами не сознавая того, говорили, используя очень небольшой набор слов английского языка (по крайней мере, когда сообщали о своей работе).

В некотором роде это имеет нечто общее с системой DENDRAL, в которой применяется графический язык, приспособленный к специфической деятельности химиков.

**DENDRAL -** это старейшая, самая разработанная экспертная система в мире. Или, по крайней мере, старейшая система, названная экспертной.

Химик, приготавливая вещество, часто хочет знать, какова его химическая структура. Для этого существуют различные способы. Во-первых, специалист может сделать определенные умозаключения на основе собственного опыта. Во-вторых, он может исследовать это вещество на спектрометре и, изучая структуру спектральных линий, уточнить свои первоначальные догадки. Во многих случаях это даст ему возможность точно определить структуру вещества. Проблема состоит в том, что все это требует времени и значительной экспертизы со стороны научного сообщества. Здесь-то и оказывается очень полезной система DENDRAL, автоматизирующая процесс определения химической структуры вещества.

В самых общих чертах процесс принятия решения следующий. Пользователь дает системе DENDRAL некоторую информацию о веществе, а также данные спектрометрии (инфракрасной, ядерного магнитного резонанса и масс-спектрометрии), и та, в свою очередь, выдает диагноз в виде соответствующей химической структуры. Можно для простоты представить систему DENDRAL состоящей из двух частей, как если бы в одной экспертной системе были две самостоятельные системы. Первая из них содержит набор правил для выработки возможных химических структур. Вводимая информация состоит из ряда заключений, сделанных химиком, и позволяет судить какие структуры вероятны в том или ином случае.

На выходе первой системы имеется не один простой ответ. Обычно это серия возможных структур - программа не в состоянии точно сказать, какая из них верна. Затем DENDRAL «берет» каждую из этих структур по очереди и использует вторую экспертную систему, чтобы определить для каждой из них, каковы были бы результаты спектрального анализа, если бы это вещество существовало и было на самом деле исследовано на спектрограмме. Процесс, часто именуемый «генерация и проверка», позволяет постоянно сокращать число возможных рассматриваемых вариантов, чтобы в любой момент оно было как можно меньше. В отличие от некоторых экспертных систем DENDRAL задумана не как «игрушка». Она используется не только для проверки теоретических основ экспертных систем, но и реально применяется для определения химических структур.

**PROSPECTOR** - это экспертная система, применяемая при поиске коммерчески оправданных месторождений полезных ископаемых.

Система PROSPECTOR, по аналогии с MYCIN, содержит большое число правил, относящихся к различным объектам, а также возможных исходов, выведенных на их основе. В этой системе используется также «движение по цепочке назад» и вероятности. Методы этой системы являются одними из лучших среди всех разработанных методов для любой из существующих ныне систем.

Самый простой случай - правила, выражающие логические отношения. Это правила типа «ЕСЛИ *X,* ТО Z», где событие Z непосредственно вытекает из *X*. Они остаются такими же простыми, если сопоставить *X* некоторую вероятность.

Если у *X* всего один аргумент, то это правило существенно упрощается. Обычно вместо *X* мы представляем более сложное логическое выражение, например *(X И Y)* или (*X ИЛИ Y*).

Если элементы отношения связаны с помощью логического И и отдельным элементам этого отношения сопоставлены определенные вероятности, то система PROSPECTOR выбирает минимальное из этих значений и присваивает эту минимальную вероятность рассматриваемому возможному исходу. Поэтому когда вероятность наступления события *X* равна 0,1 и вероятность наступления события *Y* равна 0,2, то вероятность исхода Z равна 0,1. Легко видеть, почему выбран такой метод: чтобы Z было истинным, и *X,* и *Y* должны быть истинными. Это является «жестким» ограничением, поэтому следует брать минимальное значение.

Система PROSPECTOR пользуется методом, основанным на применении формулы Байеса с целью оценки априорной и апостериорной вероятностей какого-либо события. В целом правила в системе PROSPECTOR записываются в виде ЕСЛИ ..., ТО (LS, LN), причем каждое правило устанавливается с отношением правдоподобия как для положительного, так и для отрицательного ответа.

Система PROSPECTOR предлагает пользователю шкалу ответов в диапазоне от -5 до + 5. Нижний предел - это определенно «Да», верхний - определенно «Нет».

Обычно ответ пользователя находится где-то между крайними значениями, и PROSPECTOR корректирует P(H), учитывая LS и LN с помощью линейной интерполяции. Это легко представить себе в виде линейной шкалы, где LN - крайнее левое, а LS - крайнее правое значения.

Кроме экспертных систем MYCIN, DENDRAL и PROSPECTOR существует большое количество других экспертных систем. В табл. 2.6 приводится список некоторых систем, отличительной особенностью которых является наличие большой базы знаний. Этот перечень, конечно, весьма неполный, потому что в последнее время происходит быстрое расширение сферы применения экспертных систем, и полный их перечень был бы огромным и устарел бы почти сразу после его опубликования.

В этом списке приведены также «пустые» экспертные системы (не содержащие конкретных правил предметных областей) и экспертные системы по построению других экспертных систем. Такие системы являются инструментальными средствами .гля создания прикладных экспертных систем. Они значительно облегчают задачи создания полномасштабной прикладной экспертной системы.

Вообще же инструментальные средства создания экспертных систем (ЭС) классифицируют следующим образом:

• символьные языки программирования, ориентированные на создание ЭС и систем искусственного интеллекта (например, LISP, INTERLISP, SMALLTALK);

•языки инженерии знаний, т.е. языки высокого уровня, ориентированные на построение ЭС (например, OPS-5, LOOPS, Пролог, KES);

• системы, автоматизирующие разработку (проектирование) ЭС (например, KEE, ART, TEIRESLAS, AGE, TIMM); их часто называют окружением (enviroment) для разработки систем искусственного интеллекта, ориентированных на знания;

• оболочки ЭС (или пустые ЭС) - ЭС, не содержащие знаний ни о какой проблемной области (например, ЭКСПЕРТИЗА, EMYCIN, ЭКО, ЭКСПЕРТ).

**Таблица *2.6* Список некоторых экспертных систем**

| Наименование системы | Назначение системы |
|---|---|
| MYCIN | Медицинская диагностика |
| PUFF | Тоже |
| PIP | » |
| CASNET | » |
| INTERNIST | » |
| SACON | Техническая диагностика |
| PROSPECTOR | Геологическая диагностика |
| DENDRAL | Определение химической структуры вещества |
| SECHS | Тоже |
| SYNCHEM | » |
| EL | Анализ цепей |
| MOLGEN | Генетика |
| MECHO | Механика |
| PECOS | Программирование |
| Rl | Конфигурирование компьютеров |
| SU/X | Машинная акустика |
| VM | Медицинские измерения |
| SOPHIE | Обучение электронике |
| GUIDON | Медицинское обучение |
| TE1RESIAS | Построение базы знаний |
| EMYCIN | Тоже |
| EXPERT | » |
| KAS | » |
| ROSIE | Построение экспертных систем |
| AGE | Тоже |
| HEARSAY Ill | » |

| | |
|---|---|
| AL/X | » |
| SAGE | » |
| Micro-Expert | » |

***Контрольные вопросы и задания***

1. Что отличает экспертные системы от других программ?
2. Какие категории различных типичных проблем решаются экспертными системами?
3. Охарактеризуйте экспертную систему MYCIN.
4. Охарактеризуйте экспертную систему DENDRAL.
5. Охарактеризуйте экспертную систему PROSPECTOR.
6. Какие виды инструментальных программных средств для создания экспертных систем существуют?

# § 10. ИНСТРУМЕНТАЛЬНЫЕ ПРОГРАММНЫЕ СРЕДСТВА ДЛЯ РЕШЕНИЯ ПРИКЛАДНЫХ МАТЕМАТИЧЕСКИХ ЗАДАЧ

## 10.1. НАЗНАЧЕНИЕ ПРОГРАММ

Описанные выше программные системы - текстовые редакторы и издательские системы, электронные таблицы и СУБД - являются инструментальными средствами общего назначения, т.е. могут использоваться для решения наиболее общих задач информационного характера в любой из сфер человеческой деятельности. Вместе с тем в отдельных сферах деятельности часто возникают задачи менее общего характера, такие, например, как проведение математических расчетов типа решения систем уравнений, интегрирования, статистической обработки информации и т.п., которые также требуют использования инструментальных программных средств. Таких более специальных инструментальных программ в настоящее время существует огромное количество. Одно их перечисление заняло бы многие страницы и все равно осталось бы неполным, так как новые «полуприкладные» системы появляются очень часто. Укажем лишь некоторые классы таких инструментальных средств: универсальные математические пакеты, пакеты статистической обработки данных, электронные «органаизеры» - программные средства, облегчающие планирование деятельности, хранение и поиск записей, отслеживающие заданные промежутки времени и т.д.

Ниже коротко описаны две широко используемых как в обучении, так и в практической деятельности системы, предназначенные для решения математических задач — пакет MATHCAD и система аналитических вычислений REDUCE.

## 10.2. ПАКЕТ MATHCAD

Одним из последних достижений в области инструментальных средств для решения прикладных задач является MATHCAD - физико-математический пакет с включенной в последнюю версию системой искусственного интеллекта SmartMath (разработка NASA), которая позволяет выполнять математические вычисления не только в числовой, но н в аналитической (символьной) форме.

На рисунке 2.28:

1 - палитра операторов;
2 - панель инструментов;
3 - главное меню;
4-рабочая область;
5 - панель форматирования;
6 - строка статуса.

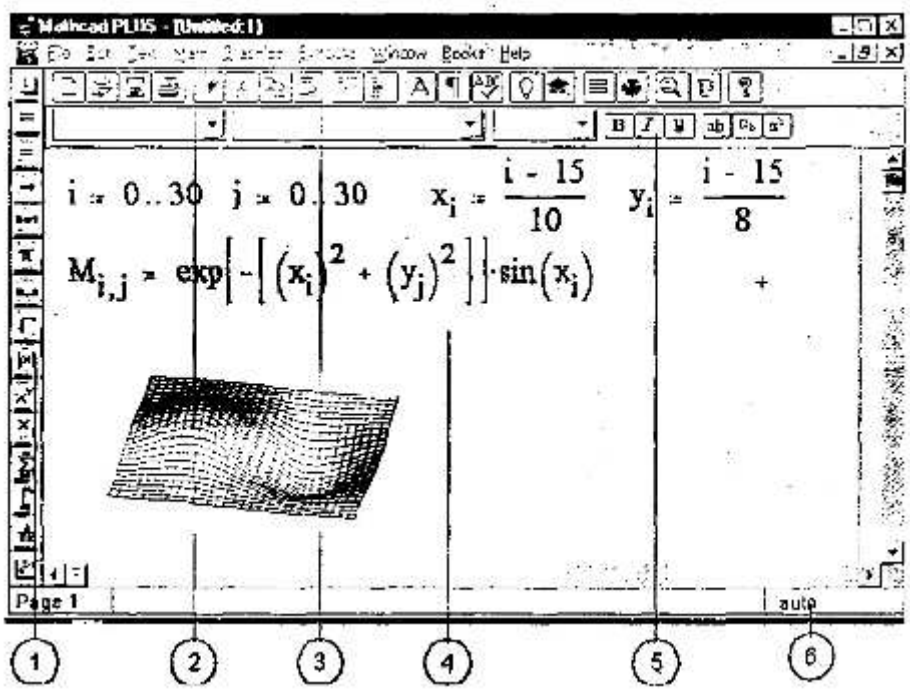



*Рис 2.28.* Пример экрана MATHCAD

Важное значение разработчики MATHCAD придавали удобству работы с ним и простоте освоения. Интерфейс MATHCAD прост и понятен, полностью отвечает стандартам среды Windows. Все графики и математические объекты могут быть введены щелчком «мыши» с перемещаемых палитр. Обучение пользователя происходит в процессе работы «на ходу» при помощи многочисленных сообщений системы.

Графическая среда MATHCAD позволяет записывать математические формулы в привычном виде, гибко и выразительно представлять данные графически.

Документ MATHCAD состоит из областей различного типа. Текстовые области создаются нажатием кнопки с буквой A на панели инструментов. Математические области возникают, если щелкнуть в свободном месте (появляется красный крестик -визир, фиксирующий место ввода формулы). Области на экране легко можно перетаскивать «мышью» или перемещать командами *Cut* и *Insert* меню *Edit.*

Большинство математических формул записывается в рабочем документе MATHCAD так же, как на листе бумаги. Знаки арифметических операций вводятся с помощью клавиш+,-,*,/.

Для ввода скобок, определяющих порядок выполнения арифметических операций используется клавиша <Space> (пробел). В большинстве случаев система тут же выдает ответ после ввода символа "=" с клавиатуры или с 1-й палитры операторов. В среде MATHCAD знак "=" означает числовой, а знак "стрелка вправо" символьный вывод значения переменной, функции, выражения.

Если последовательно вводить

$$\text{37/5 +9 = получится } \frac{37}{5+9} = 2{,}643,$$

$$\text{а если 37/5 <пробел> + 9= получится } \frac{37}{5} + 9 = 16{,}4.$$

При вводе более сложных операций используют кнопки палитр операторов MATHCAD, находящиеся на экране слева. Для перехода от одной палитры к другой надо щелкнуть на цифре над палитрой.

Стандартные математические функции, такие как cos, sin, arctan, log, exp можно вводить посимвольно, или вставлять из прокручивающегося списка. Чтобы вызвать прокручивающийся список встроенных функций MATHCAD, следует выбрать пункт *Insert Function* из меню *Math.*

Для редактирования выражения надо щелкнуть «мышью» правее элемента выражения, подлежащего изменению, а затем нажать клавишу <Backspace> и ввести нужный элемент. Для немедленного пересчета значения выражения следует щелкнуть «мышью» в стороне от выражения. Все вычисления могут производиться с высокой точностью - число значащих цифр задается из

меню системы и практически не ограниченно.

Символ определения «:=» (который можно ввести с 1-й палитры операторов или нажав клавишу «:» ) позволяет определять переменные и функции:

$$a := 6 \quad (a-7)\cdot(a+2) = -8$$

$$f(x) := \frac{\cos(x)}{a+\frac{x}{2}} \quad f(10) = -0{,}076$$

Важно следить за тем, чтобы все переменные и функции были определены левее и/или выше тех выражений, где они используются.

Вычислить (протабулировать) функции и выражения для параметров, пробегающих заданный диапазон значений, можно с помощью кнопки "m..n" 1-й палитры. Например, для табуляции функции *f(x),* приведенной выше, просто вводят z = *f(z)* = и т.д.

$z = 0,.5, 2$

| z | *F(z)* | exp(*f*(z)) • z |
|---|---|---|
| 0 | 0.167 | 0 |
| 0.5 | 0.14 | 0.575 |
| 1 | 0.083 | 1.087 |
| 1.5 | 0.01 | 1.516 |
| 2 | -0.059 | 1.885 |

Для создания этих таблиц просто вводят *z* = , *f(z)* = и т.д. Имеется ввиду, что функция определена в предыдущем примере (выше).

MATHCAD имеет широкие возможности визуализации числовых данных - 7 видов двумерных и трехмерных графиков. На каждом из двумерных графиков может одновременно находиться до 16 различных кривых, имеющих по 6 атрибутов. Можно создавать собственные библиотеки графических элементов, размещать в рабочем документе MATHCAD произвольные графические изображения.

Для построения графика надо определить с помощью кнопки "m..n" диапазон независимой переменной, а затем создать область графика с помощью кнопки внизу 1-й палитры. После этого вводятся выражения, откладываемые по осям X и Y (в средние поля ввода на соответствующих осях). Для каждой оси может быть введено несколько выражений.

Интегралы и суммы легко вычислять с помощью кнопок 1-й палитры. Для этого достаточно щелкнуть соответствующую кнопку и заполнить появившиеся позиции ввода.

Для выполнения вычислений с матрицами необходимо нажать кнопку с изображением матрицы на 2-й палитре, указать в диалоговом окне число столбцов и строк, нажать кнопку Create и заполнить пустые поля. Теперь, чтобы обратить матрицу *A,* надо напечатать *«A^-1=»,* а для вычисления определителя - «*A*».

Численное решение уравнения начинается с задания пробного значения корня и требует использования оператора root(.,,, ...). Его первый операнд - левая часть уравнения в виде *F(x)=0,* а второй - переменная, по которой ищется корень.

Например,

*t =1*
*root($t^2$- cosh(t),t)=1,621.*

MATHCAD корректно оперирует с единицами измерения выводимых числовых результатов и автоматически меняет числовое значение результата при изменении синицы измерения.

Например,

$$\frac{276\cdot\text{km}}{6\cdot\text{hr}} = 12.778\cdot\text{m}\cdot\text{sec}^{-1}.$$

Чтобы пересчитать ответ в других единицах, надо щелкнуть на ответе, затем на втором (дополнительном) поле ввода правее и ввести нужные единицы.

Документ MATHCAD, на котором совмещены текст, графика и формулы, выглядит как страница учебника или научной статьи, при этом формулы являются «живыми» - стоит внести изменения в любую из них, как MATHCAD пересчитает результаты, перерисует графики и т.д. Можно анимировать график, записав его эволюцию при изменяющемся значении параметра, а затем воспроизвести мультипликацию со звуковым сопровождением.

Документы MATHCAD могут быть особым образом «сшиты» в электронные книги. При этом они, сохраняя все свои свойства, оказываются организованными в структуру, обладающую гипертекстовыми ссылками, навигацией, контекстным поиском, открывающимися окнами и т.д.

Доступ к таким электронным книгам может осуществляться по локальным и глобальным сетям - MATHCAD имеет средства для выхода в Интернет и загрузки документов с помощью Интернет-протокола.

При поиске числового результата наряду с общеупотребительными математическими операциями и функциями может быть использовано большое количество встроенных функций, таких как функции отыскания собственных векторов матрицы, решения дифференциального уравнения, генерации последовательности случайных чисел с заданным законом распределения.

В среде MATHCAD имеются функции трех видов: встроенные, пользовательские и вложенные. Это виртуальные функции, производные, интегралы, корни, связанные с соответствующими вычислительными методами и алгоритмами. В меню Symbolic пакета Mathcad PLUS 5.0. включены следующие операции символьной математики:

- вычисление выражения в аналитическом виде;
- вычисление выражения в комплексном виде;
- вычисление числового значения выражения;
- упрощение выражений;
- развертывание выражения;
- разложение на множители;
- группировка выражения;
- вычисление коэффициентов полинома;
- поиск производной по переменной;.
- интегрирование по переменной;
- решение уравнения в аналитическом виде**;**
- подстановка в выражение;
- разложение в ряд;
- представление в виде смешанной дроби;
- транспонирование матрицы;
- инвертирование (обращение) матрицы;
- нахождение детерминанта (определителя) матрицы;
- преобразование Фурье;
- обратное преобразование Фурье;
- преобразование Лапласа;
- обратное преобразование Лапласа**;**
- Z-транспонирование;
- обратное Z-транспонирование;
- пределы (команд нет.есть кнопки-иконки).

В системе имеются разнообразные способы ввода числовых данных: с клавиатуры, из других приложений, например, электронных таблиц, с использованием технологии OLE или DDE или буфера обмена, непосредственно их файлов, с использованием разнообразных функций файлового доступа.

Интеллектуальная система SmartMath включается в работу двумя способами: одноименной командой из меню Math или нажатием и «притоплением» на панели инструментов кнопки-иконки с изображением кафедралки - головного убора средневековых ученых.

SmartMath позволяет работать не только в ручном, но и в автоматическом режиме. Режим автоматических символьных преобразований включается опцией . *Live Variable* в меню *Math.*

Искомое выражение появляется правее, ниже или вместо исходного, заданного пользователем. Место для результата задается установкой *Derivation Format...* и *Derive in Place* в меню *Symbolic.* Если в исходном выражении пользователь что-то поменяет, то ему придется все действия повторить, не забыв при этом стереть предыдущий вариант ответа. В автоматическом режиме за исходным выражением нажатием соответствующей кнопки-иконки ставится знак «стрелка вправо». А чтобы система SmartMath поняла, в каком направлении необходимо вести преобразования, введены 7 ключевых слов: factor, expand, series, simplify, complex, float и assume. Эти слова можно считать зачатками нового языка программирования, ориентированного не на вычислительный, а на аналитический процесс.

Второй режим системы SmartMath связан с оптимизацией численных расчетов.

Ключевое слово optimize, поставленное перед суммой (произведением, интегралом, пределом), заставляет систему SmartMath отойти от лобовой атаки. Если оптимальное решение найдено, то правее выражения появляется красная шестиугольная звезда.

Пользователь может просмотреть не только численный результат, но и аналитическое выражение, упростившее расчеты. Оно заносится в специальный буфер, отображенный на диске командой *Show SmartMath...меню Math* или щелчком по красной звездочке. Оптимизационное выражение можно записать в переменную, которая будет уже иметь, не числовой, а символьный тип. Оптимизировать расчеты можно и без слова optimize, включив опцию Optimize в меню Math. В этом случае оптимизироваться будут все выражения без особого на то приглашения.

Оптимизация не только ускоряет расчеты, но и повышает их точность. И не только количественно, но и качественно за счет исправления методологических ошибок (промахов) численных методов.

## 10.3. СИСТЕМА АНАЛИТИЧЕСКИХ ПРЕОБРАЗОВАНИЙ REDUCE

Развитие вычислительной техники начиналось с автоматизации выполнения арифметических действий. Вместе с тем известно, что компьютеры могут успешно оперировать математическими символами. Область вычислительной математики, связанная с аналитическими преобразованиями и получившая название компьютерной алгебры, в настоящее время развивается и получает широкое распространение в различных направлениях науки и образования. Основным объектом, над которым производит действие компьютер, является аналитическое (символьное) выражение, организованное и преобразуемое по заданным логическим правилам. Сегодня возможно компьютерное интегрирование и дифференцирование символьных выражений, перестановки и перегруппировки членов, приведение подобных членов, подстановки в выражения с последующим их преобразованием.

Очевидно, что известные системы программирования (Паскаль, СИ, Бейсик и т.п.) мало пригодны для анализа и преобразования символьной информации. Для этих целей созданы и развиваются специальные системы аналитических преобразований, которые можно разделить на универсальные, специализированные и общего назначения. Наибольшей популярностью пользуется универсальная система символьных вычислений REDUCE, автором которой является профессор А.Хиен. Система Reduce написана на языке высокого уровня ЛИСП.

Язык Reduce, составляющий ядро системы, трансляторы для которого разработаны для всех распространенных типов ЭВМ, предназначен прежде всего для проведения вычислении в аналитическом виде. Язык «знает» все операции алгебры с многочленами, приведением подобных членов, раскрытием скобок; все базовые элементарные функции, в том числе и в комплексной форме, ряд других функции; широкий набор операции над матрицами, включающий как входящие в обычные вузовские программы, так и выходящие за их пределы (например, функции от матриц); очень хорошо дифференцирует и несколько хуже вычисляет первообразные (но все же значительно лучше, чем большинство студентов, изучивших стандартный курс интегрального исчисления); умеет делать и ряд других действий.

Насколько это существенно для решения некоторых задач, показывает следующий пример. В одной из диссертаций по физике магнитных явлений диссертант (дело было в середине 50-х годов) потратил несколько лет на решение в принципе несложной задачи, требовавшей, однако, проведения совершенно фантастического (для человека) количества операторных коммутаций и последующего приведения подобных членов. Каждое отдельное неразрывное преобразование требовало, по-видимому, нескольких месяцев неустанной работы при максимальной аккуратности и на-

пряжении. Поручить же эту работу ЭВМ было невозможно, так как в те времена языков аналитических преобразований в практически пригодном виде не существовало. О независимой проверке работы не приходилось и думать - это потребовало бы от другого человека не менее года работы. Примерно через 15 лет все эти выкладки были проделаны на ЭВМ в системе Reduce за несколько дней; оказалось, что автор диссертации почти все выкладки сделал безошибочно.

Разумеется, язык «умеет» производить и численные операции, причем его арифметика имеет произвольную точность, не привязанную к способу представления чисел с плавающей запятой в регистрах процессора и ячейках ОЗУ. Получить при вычислении 20 или 50 значащих цифр в результате для Reduce вполне возможно.

В системе Reduce программа записывается и выполняется по предложениям, каждое из которых представляет собой последовательность символов. Предложение завершается одним из символов: ; (точка с запятой), п (кружок с черточками). Если предложение оканчивается знаком ";", то результат его выполнения выводится на экран дисплея или печатающее устройство. В случае знака ◘ вывод не происходит.

В системе Reduce каждая переменная имеет имя и значение. Если переменной не присвоено какое-либо значение, то имя переменной является ее значением. В этом заключается одно из принципиальных отличий подобных систем от традиционных языков программирования. Первоначально имя и значение переменной совпадают между собой, и такая переменная называется свободной.

Запуск программы на выполнение в системе осуществляется клавишей <Enter>.

Ниже приведем несколько примеров программ в системе Reduce, которые позволят получить первоначальные представления о системах аналитических преобразований символьной информации.

*Пример 1.*

A; XI; SS# ABCDIF; <Enter>

Листинг результата программы следующий:
A
XI
ABCDIF

Комментарий: все переменные являются свободными, т.е. их значения совпадают с именами.

*Пример 2.*

A:=123456789# B:= 123456789123456789#A*B; <Enter>

Листинг результата программы следующий:

152415787654320997501905 21

Комментарий: переменным А и В присваиваются целочисленные значения и вычисляется их произведение, причем результат вычисления точный без округления.

*Пример 3.*

A:=S# A; A:=X*Y# A; Q:=X:=Y# Q; X; <Enter>

Листинг результата программы следующий:

S
**X*У**

**Y**
**Y**

Комментарий: переменной A присваивается сначала значение S, затем - X*Y.

*Пример 4.*

13; 3+6; 2**64; 11-20; 25/(-125); 2*(3*A-6)/6; <Enter>

Листинг результата программы следующий:

139 18446744073709551616 (-9) (-1)/5A-2

Комментарий: при проведении алгебраических преобразований для записи сложных выражений используют имена переменных и знаки арифметических операций.

*Пример5.*

I**2; A:=X+I*Y# B:=X-I*Y# A*B; A**2; <Enter>

Листинг результата программы следующий:

-1
X-52-0+Y-52
' 2*I*X*Y + X-52-0-Y-52

Комментарий: для использования комплексных чисел за латинской буквой I закреплено значение мнимой единицы.

Пример 6.

OPERATOR F,W; W(X); F(5*X); (F(X)+A)**2; <Enter>

Листинг результата программы следующий:

W(X) F(5*X) F(X)-52-0 + 2*A*-F(x) + A-52

Комментарий: имена в скобках используют для обозначения операторов или функций, которые описываются предварительно командой OPERATOR.

Пример 7.

DF(X**2,X); DF(Y,Y,2); DF(X**3*Y*82*Z**3,X,3,Y,Z,2); DF(Y,X); <Enter>

Листинг результата программы следующий: 2*X 0 72*Y*Z O
Комментарий: встроенный оператор DF используется для вычисления частных производных по отношению к одной или нескольким переменным, первым аргументом в скобках является дифференцируемое выражение, далее - аргументы, по которым проводится дифференцирование, и числа, указывающие порядок производной.

Пример 8.

INT(X**2,X); INT(SIN(X),X); <Enter>
Листинг результата программы следующий:
X-53-0/3 - COS(X)

Комментарий: оператор INT используется для вычисления интегралов, на первом месте стоит интегрируемое алгебраическое выражение, на втором месте указывается переменная интегрирования.

Мы привели наиболее простые возможности системы Reduce. Подробнее ознакомиться с работой подобных систем читателю рекомендуется по специальным учебным пособиям и монографиям.

***Контрольные вопросы***

1. В чем основные отличия переменных в традиционных системах программирования от систем аналитических преобразований типа REDUCE?
2. В каких задачах предпочтительнее использовать методы компьютерной алгебры?

# § 11. КОМПЬЮТЕРНОЕ ТЕСТИРОВАНИЕ

## 11.1. ТЕХНОЛОГИЯ ПРОЕКТИРОВАНИЯ КОМПЬЮТЕРНЫХ ТЕСТОВ ПРЕДМЕТНОЙ ОБЛАСТИ

Компьютеры в обучении - вопрос, требующий отдельного рассмотрения. Отметим лишь, что различные варианты АОС (автоматизированных обучающих систем) вобрали в себя лучшие достижения компьютерных технологий и стали широко популярными не только в учебных заведениях, но и при подготовке персонала в промышленности, различных социальных сферах, военном деле и т д.

Обучение - многогранный процесс, и контроль знаний - лишь одна из его сторон. Однако именно в ней компьютерные технологии продвинулись максимально далеко, и среди них тестирование занимает ведущую роль. В ряде стран тестирование потеснило традиционные формы контроля - устные и письменные экзамены и собеседования.

По-видимому, многие преподаватели уже прошли через некоторую эйфорию при создании тестов и поняли, что это - весьма непростое дело. Куча бессистемно надерганных вопросов и ответов - далеко еще не тест. Оказывается, что для создания адекватного и эффективного теста нужно затратить много труда. Компьютер может оказать в этом деле немалую помощь.

Существует специальная *теория тестирования,* оперирующая понятиями *надежность, валидность, матрица покрытия* и т.д., не специфических именно для компьютерных тестов. Здесь мы не будем в нее углубляться, сосредоточившись в основном на технологических аспектах.

Широкое распространение в настоящее время получают инструментальные авторские системы по созданию педагогических средств: обучающих программ, электронных учебников, компьютерных тестов. Особую актуальность для преподавателей школ и вузов приобретают программы для создания компьютерных тестов - тестовые оболочки. Подобных программных средств существует множество и программисты-разработчики готовы строить новые варианты, так называемых, авторских систем. Однако широкое распространение этих программных средств сдерживается отсутствием простых и нетрудоемкнх методик составления тестовых заданий, с помощью которых можно «начинять» оболочки. В настоящем разделе представлены некоторые подходы к разработке компьютерных тестов.

**Проектирование модели знаний.** Эксперты чаще используют метод нисходящего проектирования модели знаний (технология «сверху - вниз»). Вначале строится генеральное содержание предметной области с разбивкой на укрупненные модули (разделы). Затем проводится детализация модулей на элементарные подмодули, которые, в свою очередь, наполняются педагогическим содержанием.

Другой метод проектирования «снизу - вверх» (от частного к общему) в большинстве случаев реализуется группой экспертов для разработки модели знаний сложной и объемной предметной области, или для нескольких, близких по структуре и содержанию, предметных областей.

Каждый модуль предполагает входящую информацию, состоящую из набора необходимых понятий из других модулей и предметных областей, а на выходе создает совокупность новых понятий, знаний, описанных в данном модуле.

Модуль может содержать подмодули. Элементарный подмодуль - неделимый элемент зна-

ния - может быть представлен в виде базы данных, базы знаний, информационной модели. Понятия и отношения между ними представляют семантический граф.

Приведем пример элемента модуля знаний по теме «Исследование графиков функций», рис**.** 2.29:


*Рис. 2.29.* Пример элемента модуля знаний

Модульное представление знаний помогает:

• организовать четкую систему контроля с помощью компьютерного тестирования, поскольку допускает промежуточный контроль (тестирование) каждого модуля, итоговый по всем модулям и их взаимосвязям, а также эффективно использовать методику «черного ящика»;

• осуществлять наполнение каждого модуля педагогическим содержанием;

• выявить и учитывать семантические связи модулей и их отношения с другими предметными областями**.**

**Этапы разработки компьютерных тестов.** Можно выделить два принципиальных способа контроля (тестирования) некоторой системы:

1) метод «белого ящика» - принцип тестирования экспертной модели знаний;

2) метод «черного ящика» - тестирование некоторой сложной системы по принципу контроля входных и выходных данных (наиболее подходит для компьютерного тестирования).

Введем ряд определений и понятий.

**Тестирование** - процесс оценки соответствия личностной модели знаний ученика экспертной модели знаний. Главная цель тестирования - обнаружение несоответствия этих моделей (а не измерение уровня знаний), оценка уровня их несоответствия.

Тестирование проводится с помощью специальных тестов, состоящих из заданного набора тестовых заданий.

**Тестовое задание** - четкое и ясное задание по предметной области, требующее однозначного ответа или выполнения определенного алгоритма действий.

**Тест** - набор взаимосвязанных тестовых заданий, позволяющих оценить соответствие знании ученика экспертной модели знаний предметной области.

**Тестовое пространство** - множество тестовых заданий по всем модулям экспертной модели знании.

**Класс эквивалентности** - множество тестовых заданий, таких, что выполнение одного из них учеником гарантирует выполнение других.

**Полный тест** - подмножество тестового пространства, обеспечивающего объективную оценку соответствия между личностной моделью и экспертной моделью знаний.

**Эффективный тест** - оптимальный по объему полный тест.

Самой сложной задачей эксперта по контролю является задача разработки тестов, которые позволяют максимально объективно оценить уровень соответствия или несоответствия личностной модели знании ученика и экспертной модели.

Подбор тестовых заданий осуществляется экспертами-педагогами методологией «белого ящика», а их пригодность оценивают с помощью «черного ящика», рис. 2.30.


*Рис. 2.30.* Схема создания тестовых заданий

Самый простой способ составления тестовых заданий - формирование вопросов к понятиям, составляющим узлы семантического графа, разработка упражнений, требующих для выполнения знания свойств выбранного понятия. Более сложным этапом является разработка тестовых заданий, определяющих отношения между понятиями. Еще более глубокий уровень заданий связан с подбором тестов, выявляющих связь понятий между отдельными модулями.

Множество тестовых заданий (тестовое пространство), вообще говоря, согласно принципу исчерпывающего тестирования, может быть бесконечным. Однако в каждом реальном случае существует конечное подмножество тестовых заданий, использование которых позволяет с большой вероятностной точностью оценить соответствие знаний ученика заданным критериям по экспертной модели знаний (полный тест).

Из полного теста можно выделить эффективный тест (оптимальный по объему набор тестовых заданий, гарантирующий оценку личностной модели ученика заданным критериям). Выбор эффективного теста зависит от удачного разбиения тестового пространства на классы эквивалентности, пограничные условия, создания тестов на покрытие путей и логических связей между понятиями и модулями.

В дальнейшем необходим тестовый эксперимент на группе учащихся, который позволит провести корректировку и доводку теста до вида эксплуатации (методика черного ящика).

Таким образом, построение компьютерных тестов можно осуществлять в следующей последовательности:

- формализация экспертной целевой модели знаний;
- нисходящее (или снизу - вверх) проектирование тестового пространства**;**
- формирование и наполнение тестовых заданий;
- формирование полного компьютерного теста;
- тестовый эксперимент;
- выбор эффективного теста;
- анализ, корректировка и доводка теста до вида эксплуатации.

## 11.2. ТИПЫ КОМПЬЮТЕРНЫХ ТЕСТОВ

В соответствии с моделью знаний, выделим три класса компьютерных тестов на знания, умения и навыки. Отметим, что типы компьютерных тестовых заданий определяются способами однозначного распознавания ответных действий тестируемого.

1.Типы тестовых заданий по блоку «знания» — вопросы альтернативные (требуют ответа да - нет);

- вопросы с выбором (ответ из набора вариантов);
- вопросы информативные на знание фактов (где, когда, сколько);
- вопросы на знание фактов, имеющих формализованную структуру (в виде информационной модели или схемы знаний);
- вопросы по темам, где имеются однозначные общепринятые знаковые модели: математические формулы, законы, предикатные представления, таблицы;
- вопросы, ответы на которые можно контролировать по набору ключевых слов;
- вопросы, ответы на которые можно распознавать каким-либо методом однозначно.

2. Типы тестовых заданий по блоку «навыки» (распознание деятельности: манипуляции с клавиатурой; по конечному результату):

• задания на стандартные алгоритмы (альтернативные да - нет, выбор из набора вариантов);

• выполнение действия.

3. Типы тестовых заданий по блоку «умения». Те же самые, что для навыков, но используют нестандартные алгоритмы и задачи предметной области при контроле времени их решения:

• задания на нестандартные алгоритмы (альтернативные да - нет, выбор из набора вариантов);

• выполнение действия.

Выбор типов тестов определяется:

• особенностями инструментальных тестовых программ (тестовыми оболочками); .

• особенностями предметной области;

• опытом и мастерством экспертов.

## 11.3. ИНСТРУМЕНТАЛЬНЫЕ ТЕСТОВЫЕ ОБОЛОЧКИ

Для создания тестов по предметной области разрабатываются специальные инструментальные программы-оболочки, позволяющие создавать компьютерные тесты путем формирования базы данных из набора тестовых заданий.

Инструментальные программы, позволяющие разрабатывать компьютерные тесты, можно разделить на два класса: универсальные и специализированные. Универсальные программы содержат тестовую оболочку как составную часть. Среди них Адонис (Москва), Linkway (Microsoft), Фея (Томск), Радуга (Москва) и т.п. Специализированные тестовые оболочки предназначены лишь для формирования тестов. Это - Аист (Москва), I_now (Иркутск), Тест (Красноярск) и др.

Для того, чтобы разработать компьютерный вариант теста с помощью одной из названных выше программ, необходимо уяснить, какие формы тестовых заданий они допускают.

Как правило, компьютерные формы представления тестовых заданий могут выглядеть следующим образом.

1. Вопросы с фасетом. Задание вопроса, в котором меняются признаки.

*Пример.*

Назовите столицу страны АНГЛИЯ : ? _____.

2. Вопросы с шаблоном ответа.

*Пример.*

В каком году произошла Октябрьская революция? В ___ году.

3. Вопросы с набором ключевых слов (изображений, обозначений), из которых можно конструировать ответ.

*Пример.*

Какие силы действуют на тело, движущееся по наклонной плоскости? (сила трения, сила упругости, сила тяжести, сила реакции опоры).

4. Закрытая форма вопроса: номер правильного ответа.

*Пример.*

Какой климат в Красноярском крае?

1. Континентальный.
2. Субтропики.
3. Умеренный.
4. Резко-континентальный.

5. Задание на соответствие: несколько вопросов и несколько ответов.

*Пример.*

| | |
|---|---|
| а) Кто автор планетарной модели? | а) Лермонтов М.Ю. |
| б) Кто автор закона тяготения? | б) Резерфорд |
| в) Кто автор поэмы «Мцыри»? | в) Ньютон |

6. Конструирование ответа (шаблонный и безшаблонный варианты): ответ формируется путем последовательного выбора элементов из инструментария по типу меню.

*Пример.*

Чему равна производная функции y = Sin(x) + Cos(x)?

y’ = (Sin(x), Cos(x), tg(x), +, -, /,*, log(x), 1, 2,3,5 и т.д.)

7. Задание на конструирование изображений: с помощью графредактора, меню изображении (аналогично предыдущему примеру).

8. Задание на демонстрацию с движущимися объектами. Ответ в виде действий тестируемого (определенный набор клавиш).

*Пример.*

Клавиатурный тренажер на время.

Перечисленные формы компьютерного представления тестовых заданий не исчерпывают их многообразия. Многое зависит от мастерства и изобретательности эксперта по тестированию. При создании тестов важно учитывать многие обстоятельства, например, личность тестируемого, вид контроля, методику использования тестов в учебном процессе и т.п.

Хорошим считается тест если

• он восприимчив к угадыванию тестируемым;

• он восприимчив к невнимательности и ошибочным действиям тестируемого;

• он положительно влияет на тестируемого и педагога.

При этом тест используется обучаемым как тренажер и орудие самоконтроля. Для учителя тест служит для корректировки учебного процесса, используется как вспомогательное средство текущего контроля знании, как дидактические средства обучения, для дистанционного обучения,

## 11.4. ПРИМЕР ТЕСТА ПО ШКОЛЬНОМУ КУРСУ ИНФОРМАТИКИ

В 1996 г. Республиканский центр тестирования использовал тесты по некоторым школьным предметам, в частности по информатике. Ниже приводится один из его вариантов (разработчики: Н.Г.Граве, И.А.Елисеев. Г.В.Тюрникова). Тесты построены на основе канонического принципа: вопрос и варианты ответа.

Разработчиками выбрана следующая модель знаний школьного курса информатики:

*N* Модуль 1. **Введение**

1. Измерение информации ,
2. Свойство информации
3. Измерение информации
4. Предмет информации. Фундаментальные понятия
5. История развития вычислительной техники

Модуль 2. **Устройство и работа ЭВМ**

6. Состав информационно-измерительного комплекса
7. Поколения ЭВМ
8. Арифметические основы ЭВМ
9. Состав информационно-измерительного комплекса
10. Арифметические основы ЭВМ
11. Физические основы ЭВМ
12. Состав информационно-измерительного комплекса

Модуль *3.* **Алгоритмизация**

13. Величины, тип, имя, значения, вид
14. Величины, тип, имя, значения, вид
15. Величины, тип, имя, значения, вид
16. Типы алгоритма
17. Способы описания
18. Способы описания
19. Алгоритм, свойства

20-24. Остальные вопросы как единый подраздел

Модуль 4. **Информационные технологии**

25-28. Операционные системы

29-30. Текстовый, графический, музыкальный редакторы

31-32. Базы данных

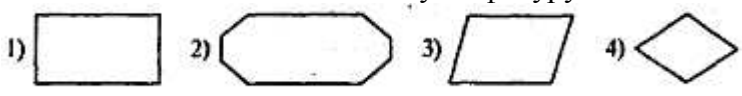
33. Электронные таблицы

Модуль 5. **Заключение**

34-36. Перспективы развития

## РОССИЙСКИЙ ТЕСТ ПО ИНФОРМАТИКЕ №01

01. Килобайт - это
    1) 1000 символов; 2) 1024 байт; 3) 8 бит; 4) 1000 байт.
02. Достоверность - это свойство
    1) алгоритма; 2) компьютера; 3) информации; 4) языка программирования.
03. Наибольший объем памяти требуется для хранения
    1)"10";2) 10; 3) "десять"; 4) (10).
04. Носителем информации является
    1) провода; 2) принтер; 3) классный журнал; 4) телефон.
05. Первая машина, автоматически выполняющая все 10 команд, была
    1) машина С.А.Лебедева; 2) машина Ч.Бэббиджа; 3) абак; 4) Pentium.
06. Минимально необходимый набор устройств для работы компьютера содержит
    1) принтер, системный блок, клавиатуру; 2) процессор, ОЗУ, монитор, клавиатуру; 3) монитор, винчестер, клавиатуру, процессор; 4) системный блок, дисководы, мышь.
07. Элементной базой ЭВМ третьего поколения являются
    1) ЭЛТ (электронно-лучевая трубка); 2) светодиоды;
    3) ИС (интегральные схемы); 4) транзисторы.
08. Число 3210-это
    1)1000002;2)358;3)2116;4)100001.
09. К внешним запоминающим устройствам относится
    1) процессор; 2) дискета; 3) монитор; 4) жесткий диск.
10. Определить сумму трех чисел: 0012 + 0178 + 1112
    1) 02310; 2) 00910; 3) 1112; 4) 10002.
11. Перевести число 3210 в двоичную систему счисления
    1)100000;2)111111;3)101010; 4) 100001.
12. К внутренним запоминающим устройствам относится
    1) монитор; 2) жесткий диск,,3) оперативная память (RAM); 4) флоппи-диск.
13. Неверно записанное выражение
    1) + 3; 2) tg(+3); 3)-tg(-3)+l; 4) -sin(-3)+(l)*(tg(+l).
14. По выполнении следующего алгоритма
    x:=7; y:=12+5, y:=y+y-x
    значение x будет
    1)7; 2) 89; 3)94; 4) 47.
15. Если исполнить
    X:=2; Y:=X+3; X:=X+1; Y:=X+3*Y,
    то значение Y равно
    1)0;2)-10;3) 18; 4) 6,5.
16. При t > 17 будет ложно
    1) t=17,01; 2) t>212 и t<1000; 3) t =17; 4) t>17 и t<20.
17. До какого числа должно изменится значение счетчика i в фрагменте алгоритма
    a:=1
    нц для i от 2 до <...>
    a: = a *i
    i: =i +1 a:= a* i
    кц
    чтобы a стало равно II?
    1)8; 2) 9; 3)10; 4) 11.
18. Для вывода данных в блок-схемах используют фигуру



19. Геометрическая фигура используется в блок-схемах для обозначения
1) условия; 2) останова; 3) любого действия; 4) цикла «для».
20. Не является свойством алгоритма
1) универсальность; 2) массовость; 3) результативность; 4) дискретность.
21. При составлении алгоритма для вычисления функции y=a*sin(x) аргументами являются 1) sin; 2) a, x; 3) x; 4) x, y.
22. Сколько раз выполнится цикл

```
i-1;
a:=10;
n:=2;
нц пока a>0
a:= a-n*i
кц
```

1)0; 2) 10; 3)5; 4) 4.
23. В качестве имени переменной может быть
1) 1996;2)a1996;3) 1996a;4)-1996.
24. Для описания циклического алгоритма используется конструкция
1) ПОКА; 2) ЕСЛИ; 3) ВЫБОР; 4) ПРОЦЕДУРА.
25. Какая программа является интерпретатором команд MS DOS?
1) AUTOEXEC.BAT; 2) MSDOS.SYS; 3) CONFIG.SYS; 4)COMMAND.COM
26. Минимально необходимый набор файлов для работы компьютера в MSDOS
1)IO.SYS, MSDOS.SYS; 2) IO.SYS. MSDOS.SYS, COMMAND.COM;
3)IO.SYS, MSDOS.SYS. COMMAND.COM, CONFIG.SYS;
4) IO.SYS, MSDOS.SYS, COMMAND.COM, AUTOEXEC.BAT.
27. Сколько символов в своем полном имени может содержать директория?
1)11; 2) 8; 3)7; 4) 12.
28. Неверным будет утверждение
1) файл с расширением .TXT может быть не текстовым;
2) системный диск может не содержать файл CONFIG.SYS;
3) файл AUTOEXEC.BAT может не содержать ни одной строки
(ни одного байта);
4) файл должен содержать в расширении не менее трех букв.
29. Текстовый редактор Лексикон - это
1) прикладная программа; 2) базовое программное обеспечение;
3) сервисная программа; 4) редактор шрифтов?
30. Под термином «интерфейс» понимается
1) внешний вид программной среды, служащий для обеспечения диалога с пользователем;
2) связь текстового редактора с устройством печати;
3) совокупность файлов, содержащихся в одном каталоге;
4) устройство хранения графической информации.
31. База данных - это 1) текстовый редактор; 2) совокупность связанных между собой сведений; 3) операционная оболочка; 4) утилиты NC?
32. Графический редактор нужен для
1) нормальной работы баз данных; 2) быстрого поиска информации;
3) проигрывания звуковых файлов; 4) создания рисунков.
33. В отличие от бумажных табличных документов, электронные таблицы обычно
1) имеют большую размерность; 2) позволяют быстрее производить расчеты;
3) обладают всеми свойствами, перечисленными в пунктах 1 -2;
4) стоят дороже.
34. Что делает невозможным подключение компьютера к глобальной сети?
1) тип компьютера; 2) состав периферийных устройств;
3) отсутствие винчестера; 4) отсутствие телефона.
35. Дан E-mail: artem@wremech.msk.ru. Слово msk означает
1) город назначения; 2) тип компьютера; 3) каталог; 4) имя пользователя.

36. Первый PHOTO CD был произведен фирмой
1) IBM; 2) APPLE; 3) KODAK; 4) POLAROID.

Ответы на тестовые задания

01 - 2); 02 - 3); 03 - 3); 04 - 3); 05 - 2); 06 - 2); 07 - 3); 08 - 4); 09 - 2); 10 -4); 11 - 4);
12 - 2); 13 - 4); 14 - 1); 15 - 3); 16-3); 17 - 4); 18 - 3); 19 - 2); 20 - 1); 21 - 3); 22 - 3); 23 - 2); 24 - 1); 25 - 4); 26 - 2); 27 - 2); 28 - 4); 29 - 1); 30 - 1); 31- 2); 32 - 4); 33 - 3); 34 -4); 35-1); 36-1).

***Контрольные вопросы и задания***

1. Разработайте модель знания по школьному разделу «действия с дробями», используя модульный принцип.
2. Разработайте тест на знание таблицы умножения чисел от 0 до 100.
3. Используя какую-либо инструментальную тестовую оболочку, разработайте тест по тестовым заданиям курса информатики, описанным в этой главе.

# §12. КОМПЬЮТЕРНЫЕ ВИРУСЫ

## 12.1. ЧТО ТАКОЕ КОМПЬЮТЕРНЫЙ ВИРУС

Среди огромного разнообразия видов компьютерных программ существует одна их разновидность, заслуживающая особого упоминания. Главное отличие этих программ от всех остальных состоит в том, что они вредны, т.е. предназначены для нанесения ущерба пользователям ЭВМ. Это - компьютерные вирусы.

Компьютерным вирусом называется программа, обычно малая по размеру (от 200 до 5000 байт), которая самостоятельно запускается, многократно копирует свой код, присоединяя его к кодам других программ («размножается») и мешает корректной работе компьютера и/или разрушает хранимую на магнитных дисках информацию (программы и данные).

Существуют вирусы и менее «злокачественные», вызывающие, например, переустановку даты в компьютере, музыкальные (проигрывающие какую-либо мелодию), приводящие к появлению на экране дисплея какого-либо изображения или к искажениям в отображении дисплеем информации, «осыпанию букв» и т.д.

Создание компьютерных вирусов можно квалифицировать с юридической точки зрения как преступление.

Интересны причины, заставляющие квалифицированных программистов создавать компьютерные вирусы, ведь эта работа не оплачивается и не может принести известности. По-видимому, для создателей вирусов это способ самоутверждения, способ доказать свою квалификацию и способности. Созданием компьютерных вирусов занимаются квалифицированные программисты, по тем или иным причинам не нашедшие себе места в полезной деятельности, в разработке прикладных программ, страдающие болезненным самомнением или комплексом неполноценности. Становятся создателями вирусов и те молодые программисты, которые испытывают. трудности в общении с окружающими людьми, не встречают признания со стороны специалистов, которым чужды понятие морали и этики компьютерной сферы деятельности.

Существуют и такие специалисты, которые отдают свои силы и талант делу борьбы с компьютерными вирусами. В России - это известные программисты Д.Лозинский, Д.Мостовой, П.А.Данилов, Н.Безруков и др. Ими исследованы многие компьютерные вирусы, разработаны антивирусные программы, рекомендации по мерам, предотвращающим уничтожение вирусами компьютерной информации и распространение эпидемий компьютерных вирусов.

Главную опасность, по их мнению, представляют не сами по себе компьютерные вирусы, а пользователи компьютеров и компьютерных программ, не подготовленные к встрече с вирусами, ведущие себя неквалифицированно при встрече с симптомами заражения компьютера, легко впадающие в панику, что парализует нормальную работу.

## 12.2. РАЗНОВИДНОСТИ КОМПЬЮТЕРНЫХ ВИРУСОВ

Рассмотрим подробнее основные особенности компьютерных вирусов, характеристики антивирусных программ и меры зашиты программ и данных от компьютерных вирусов в наиболее распространенной операционной системе MS DOS.

По приближенным оценкам к 1997 г. существовало около 7000 различных вирусов. Подсчет их осложняется тем, что многие вирусы мало отличаются друг от друга, являются вариантами одного и того же вируса и, наоборот, один и тот же вирус может менять свой облик, кодировать сам себя. На самом деле основных принципиальных идей, лежащих в основе вирусов, не очень много (несколько десятков).

Среди всего разнообразия вирусов следует выделить следующие группы:

•**загрузочные** (boot) **вирусы** заражают программу начальной загрузки компьютера, хранящуюся в загрузочном секторе дискеты или винчестера, и запускающиеся при загрузке компьютера;

• **файловые вирусы** в простейшем случае заражают пополняемые файлы, но могут распространяться и через файлы документов (системы Word for Windows) и даже вообще не модифицировать файлы, а лишь иметь к ним какое-то отношение;

• **загрузочно-файловые вирусы** имеют признаки как загрузочных, так и файловых вирусов;

• **драйверные вирусы** заражают драйверы устройств компьютера или запускают себя путем включения в файл конфигурации дополнительной строки.

Из вирусов, функционирующих не на персональных компьютерах под операционной системой MS DOS, следует упомянуть **сетевые вирусы,** распространяющиеся в сетях, объединяющих многие десятки и сотни тысяч компьютеров.

Рассмотрим принципы функционирования загрузочных вирусов. На каждой дискете или винчестере имеются служебные сектора, используемые операционной системой для собственных нужд, в том числе сектор начальной загрузки. В нем помимо информации о дискете (число дорожек, число секторов и пр.) хранится небольшая программа начальной загрузки, о которой уже рассказывалось в настоящей главе.

Простейшие загрузочные вирусы, резидентно находясь в памяти зараженного компьютера, обнаруживают в дисководе незараженную дискету и производят следующие действия:

• выделяют некоторую область дискеты и делают ее недоступной операционной системе (помечая, например, как сбойную - bad);

• замещают программу начальной загрузки в загрузочном секторе дискеты, копируя корректную программу загрузки, а также свой код, в выделенную область дискеты;

• организуют передачу управления так, чтобы вначале выполнялся бы код вируса и лишь затем - программа начальной загрузки.

Магнитные диски компьютеров винчестерского типа обычно бывают разбиты на несколько логических разделов. Программы начальной загрузки при этом имеются в MBR (Master Boot Record - главная загрузочная запись) и в загрузочном разделе винчестера, заражение которых может происходить аналогично заражению загрузочного сектора дискеты. Однако, программа начальной загрузки в MBR использует при переходе к программе загрузки загрузочного раздела винчестера, так называемую таблицу разбиения (Partition table), содержащую информацию о положении загрузочного раздела на диске. Вирус может исказить информацию Partition table и таким образом передать управление своему коду, записанному на диск, формально не меняя загрузочной программы.

Теперь рассмотрим принципы функционирования **файловых вирусов.** Файловый вирус не обязательно является резидентным, он может, например, внедриться в код исполняемого файла. При запуске зараженного файла вирус получает управление, выполняет некоторые действия и возвращает управление коду, в который он был внедрен. Действия, которые выполняет вирус, включают поиск подходящего для заражения файла, внедрение в него так, чтобы получить управление при запуске файла, произведение некоторого эффекта, например, звукового или графического. Если файловый вирус резидентный, то он устанавливается в памяти и получает возможность заражать файлы и проявляться независимо от первоначального зараженного файла.

Заражая файл, вирус всегда изменит его код, но далеко не всегда производит другие изменения. В частности, может не изменяться начало файла и его длина (что раньше считалось признаком заражения). Например, вирусы могут искажать информацию о файлах, хранящуюся в служеб-

ной области магнитных дисков -таблице размещения файлов (FAT - file allocation table), - и делать таким образом невозможной любую работу с файлами. Так ведут себя вирусы семейства «Dir».

**Загрузочно-файловые вирусы** используют принципы как загрузочных, так и файловых вирусов, и являются наиболее опасными.

## 12.3. АНТИВИРУСНЫЕ СРЕДСТВА

К настоящему времени накоплен значительный опыт борьбы с компьютерными вирусами, разработаны антивирусные программы, известны меры защиты программ и данных. Происходит постоянное совершенствование, развитие антивирусных средств, которые в короткий срок с момента обнаружения вируса -от недели до месяца - оказываются способными справиться с вновь появляющимися вирусами.

Создание антивирусных программ начинается с обнаружения вируса по аномалиям в работе компьютера. После этого вирус тщательно изучается, выделяется его сигнатура - последовательность байтов, которая полностью характеризует программу вируса (наиболее важные и характерные участки кода), выясняется механизм работы вируса, способы заражения. Полученная информация позволяет разработать способы обнаружения вируса в памяти компьютера и на магнитных -дисках, а также алгоритмы обезвреживания вируса (если возможно, удаления вирусного кода из файлов - «лечения»).

Известные ныне антивирусные программы можно разделить на несколько типов, перечисленных ниже.

• **Детекторы**. Пх назначение - лишь обнаружить вирус. Детекторы вирусов могут сравнивать загрузочные сектора дискет с известными загрузочными секторами, формируемыми операционными системами различных версий, и таким образом обнаруживать загрузочные вирусы или выполнять сканирование файлов на магнитных дисках с целью обнаружения сигнатур известных вирусов. Такие программы в чистом виде в настоящее время редки.

• **Фаги.** Фаг - это программа, которая способна не только обнаружить, но и уничтожить вирус, т.е. удалить его код из зараженных программ и восстановить их работоспособность (если возможно). Известнейшим в России фагом является Aidstest, созданный Д.Лозинским. К январю 1997 года эта программа была способна обнаружить и обезвредить около 1600 вирусов. Еженедельно появляются новые версии этой программы, рассчитанные на обезвреживание десятков новых вирусов.

Очень мощным и эффективным антивирусным средством является фаг Doctor Web (созданный И.Даниловым). Детектор этого фага не просто сканирует файлы в поисках одной из известных вирусных сигнатур. Doctor Web реализует эвристический метод поиска вирусов, может находить и обезвреживать, так называемые, полиморфные вирусы (не имеющие определенной сигнатуры), проверять файлы, находящиеся в архивах. Для нахождения вирусов Doctor Web использует программную эмуляцию процессора, т.е. он моделирует выполнение остальных файлов с помощью программной "модели микропроцессора 1-8086 и тем самым создает среду для проявления вирусов и их размножения. Таким образом, программа Doctor Web может бороться не только с полиморфными вирусами, но и с вирусами, которые только еще могут появиться в перспективе. Специалисты рекомендуют использовать Aidstest и Doctor Web в комплексе.

• **Ревизоры**. Программа-ревизор контролирует возможные пути распространения программ-вирусов и заражения компьютеров. Программы-ревизоры относятся к самым надежным средствам защиты от вирусов и должны входить в арсенал каждого пользователя. Ревизоры являются единственным средством, позволяющим следить за целостностью и изменениями файлов и системных областей магнитных дисков. Наиболее известна в России программа-ревизор ADinf, разработанная Д.Мостовым.

• **Сторожа**. Сторож - это резидентная программа, постоянно находящаяся в памяти компьютера, контролирующая операции компьютера, связанные с изменением информации на магнитных дисках, и предупреждающая пользователя о них. В состав операционной системы MS DOS, начиная с версии 6.0, входит сторож VSAFE. Однако, из-за того, что обычные программы выполняют операции, похожие на те, что делают вирусы, пользователи обычно не используют сторожа, так как постоянные предупреждения мешают работе.

• **Вакцины.** Так называются антивирусные программы, ведущие себя подобно вирусам, но не наносящие вреда. Вакцины предохраняют файлы от изменении и способны не только обнару-

жить факт заражения, но и в некоторых случаях «вылечить» пораженные вирусами файлы. В настоящее время антивирусные программы-вакцины широко не применяют, так как в прошлые годы некоторыми некорректно работающими вакцинами был нанесен ущерб многим пользователям.

Помимо программных средств защиты от вирусов существуют и специальные дополнительные устройства, обеспечивающие надежную защиту определенных разделов винчестера. Примером такого рода устройств является плата Sheriff (разработанная Ю.Фоминым). Несмотря на кажущееся обилие программных антивирусных средств, даже все вместе они не обеспечивают полной защиты программ и данных, не дают 100%-ной гарантии от воздействия вирусных программ. Только комплексные профилактические меры защиты обеспечивают надежную защиту от возможной потери информации. В комплекс таких мер входит:

• регулярное архивирование информации (создание резервных копий важных файлов и системных областей винчестера);

• избегание использования случайно полученных программ (старайтесь пользоваться только законными путями получения программ);

• входной контроль нового программного обеспечения, поступивших дискет;

•сегментация жесткого диска, т.е. разбиение его на логические разделы с разграничением доступа к ним;

• систематическое использование программ-ревизоров для контроля целостности информации;

• при поиске вирусов (который должен происходить регулярно!) старайтесь использовать заведомо чистую операционную систему, загруженную с дискеты. Защищайте дискеты от записи, если есть хоть малая вероятность заражения.

При неаккуратной работе с антивирусными программами можно не только переносить с ними вирусы, но и вместо лечения файлов безнадежно их испортить. Полезно иметь хотя бы общее представление о том, что могут и чего не могут компьютерные вирусы, об их жизненном цикле, о важнейших методах защиты.

### *Контрольные вопросы и задания*

1. Что называется компьютерным вирусом?
2. Какие типы компьютерных вирусов существуют?
3. Каковы принципы функционирования загрузочных вирусов?
4. Каковы принципы функционирования файловых вирусов?
5. Охарактеризуйте известные типы антивирусных программ.
6. Перечислите меры защиты информации от компьютерных вирусов.

## § 13. КОМПЬЮТЕРНЫЕ ИГРЫ

### 13.1. ВИДЫ И НАЗНАЧЕНИЕ КОМПЬЮТЕРНЫХ ИГР

Игры любят все. Игра является наилучшей средой для обучения любому виду деятельности.

Характерной приметой компьютерной эры стали компьютерные игры. К ним можно относиться по-разному. С одной стороны, они могут приносить пользу как учебные средства; с другой - отнимать время (иногда очень много), отвлекать от работы. Никуда не годится играть в рабочее время, применять для этого свои дискеты, рискуя занести вирус в систему коллективного пользования и т.п. Психологи считают, что игры с изобилием сцен насилия - пусть даже выраженного в предельно условной форме - способствуют формированию у детей не самых лучших качеств. Тем не менее, компьютерные игры широко распространены. Над их созданием трудятся высокопрофессиональные программисты, художники, мультипликаторы; это достаточно процветающая подотрасль индустрии программирования. Как и многие человеческие достижения, компьютерные игры можно использовать во благо и во зло.

Компьютерные игры ориентируются на развитие у игроков определенных знаний, навыков, способностей. Как правило, в компьютерных играх от игрока требуется

• владение средствами управления, быстрота и точность манипуляций;

• быстрая и правильная реакция на происходящие события;

• чувство времени, умение выдерживать заданные временные интервалы;
• способность следить за несколькими объектами одновременно;
• знание географии игрового поля, законов игрового мира;
• знание конкретной предметной области, которая моделируется в игре;
• умение искать закономерности;
• умение предугадывать действия противника;
• знание алгоритма и стратегии выигрыша;
• способность к быстрому и максимально полному перебору основных вариантов;
• память на текущие события;
• использование прошлого опыта, что происходило в предыдущих сеансах игры;
• способность интенсивно работать в течение всего сеанса игры.

В разных играх необходимы разные качества. Привлекательность компьютерных игр определяется следующими факторами:

• интересным сценарием;
• богатым внешним оформлением;
• кажущейся простотой;
• бесконечностью игры (недостижимостью поставленной цели);
• наличием большого числа стратегий;
• разнообразием игровых ситуаций.

В компьютерных играх можно выделить следующие категории:

• игры на мастерство;
• азартные игры;
• логические игры;
• обучающие (дидактические) игры.

Игры на мастерство основаны на управлении игровыми объектами. В азартных играх исход в большей степени зависит от случайности, везения. Логические игры содержат стратегию поведения игрока, зависящую от игровых ходов соперника или от игровой ситуации. В обучающих играх объектом управления становится ученик, а целью - отработка некоторых навыков и усвоение знаний.

По способам реализации игры можно классифицировать по признакам:

- дискретные и игры с режимом реального времени;
- антагонистические и неантагонистические;
- конечные и бесконечные;
- со случайными событиями или детерминированные;
- для одного или двух и более участников;
- игры с разным уровнем сложности.

Самые распространенные компьютерные игры - пошаговые, конечные, детерминированные для двух участников, один из которых компьютер.

По структуре в компьютерных играх можно выделить три блока и три уровня:

- блок игровой среды (правила игры);
- блок взаимодействия с играющим (интерфейс);
- блок оценки игровой ситуации (анализ);
- уровень оперативный (текущее управление клавишами);
- уровень тактический (локальные цели, усложнение игры);
- уровень стратегический (конец игры, фиксация результатов).

Общая структура компютерных игр представлена на рис. 2.31.

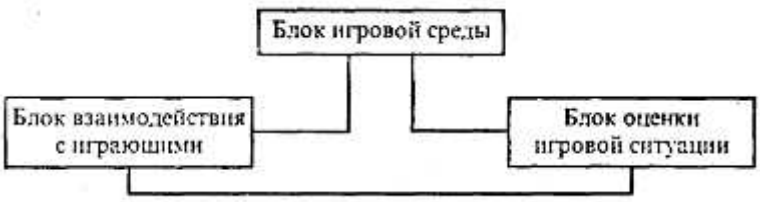



*Рис. 2.31.* Общая структура компьютерных игр

**Блок игровой среды** - это та сцена, тот трехмерный компьютерный мир, в котором есть все, что стоит, лежит, движется, появляется и исчезает в соответствии со смыслом и законами игры.

**Блок взаимодействия** - это все то в программе, что позволяет играющему изменять то, что предусмотрено блоком игровой среды.

**Блок оценки** - это условия для играющего и для объектов игры на игровой сцене. Это подсчет числа очков, описание или показ начальной и конечной игровой ситуации в игре.

Удается выделить три иерархических уровня, которые позволяют правильно построить схему игры: оперативный, тактический и стратегический.

**Оперативный уровень** - это изменение объектов на игровой сцене посредством нажатия клавиш или управляющего устройства (мышь, джойстик). Результатом действия оперативного уровня должно быть отображение всех перемещений и изменений на экране дисплея.

**Тактический уровень** включает и оперативный. Действия на этом уровне ведут к достижению некоторой вполне определенной локальной цели. Изменения сложности игры, темпа, уровня происходят на этом этапе.

**Стратегический уровень** включает тактический и содержит несколько самостоятельных блоков: ввод на игровую схему всех объектов для определения, задания и визуализации их начальных параметров, проверка критериев окончания игры, фиксации и визуализации результатов всей игры в целом и результатов прошлых игр.

## 13.2. ОБЗОР КОМПЬЮТЕРНЫХ ИГР

Компьютерные игры бывают разные и каждая из них требует разных ресурсов компьютера. Ниже приведен краткий обзор компьютерных игр.

*Игры на мастерство*

1. Имитационные игровые виды спорта: футбол, волейбол, теннис и т.д.
2. Военные игры: морские бои, воздушные бои, звездные воины, игры с вооружениями и др.
3. Рукопашные схватки: каратэ, кунг-фу, тэквондо.
4. Профессиональные игры: авторалли, пилотирование самолета и др.
5. Приключенческие игры: путешествия, подвиги, приключения.
6. Графические игрушки: тетрис, выбивалки, «живые картинки» и т.д.
7. Учебные тренажеры: азбука, клавиатурный тренажер и др.

*Азартные игры*

1. Карточные игры: пасьянсы, покер, преферанс, марьяж, бридж и т.п.
2. Имитационные азартные игры: кости, рулетка, «поле чудес» и др.

*Логические игры*

1. Шахматные игры: шашки, шахматы, реверси и т.п.
2. Логические учебные развивающие игры: крестики - нолики, минер, лабиринты, угадан число, слово, быки и коровы, поле чудес, ним и др.

*Обучающие игры*

Особая роль в мире компьютерных игр принадлежит обучающим и развивающим играм. Педагогам хорошо известна важность и высокая эффективность игровых форм обучения, особенно успешно применяемых при работе с детьми дошкольного и младшего школьного возраста. Существуют, например, специальные среды для начального обучения информатике с забавными исполнителями, которые в игровой форме отрабатывают простые пользовательские навыки, навыки алгоритмизации и т.д. Так, созданная под руководством Ю.А.Первина среда «Роботландия» с ее

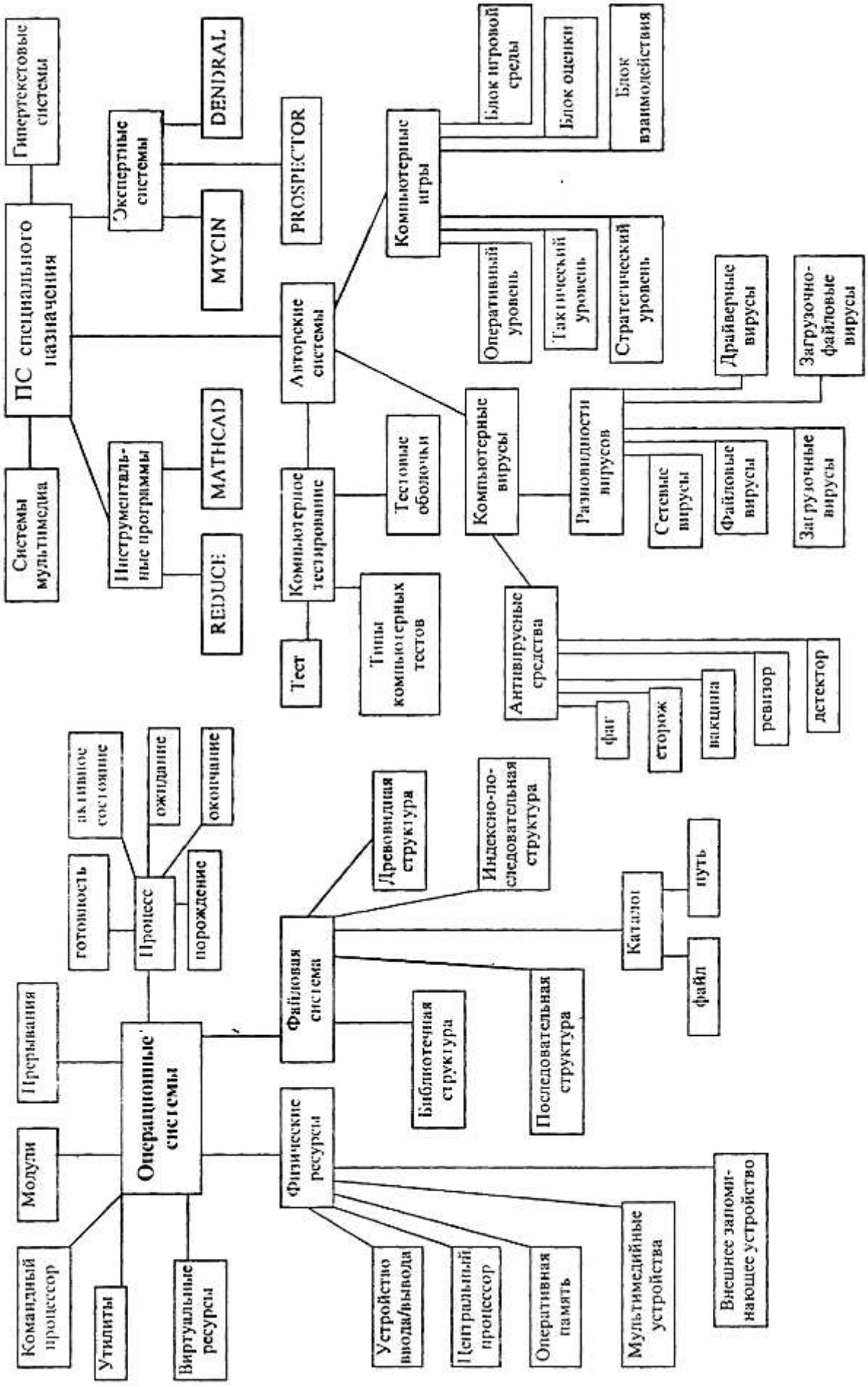
забавными исполнителями «Машинист», «Буквоед», «Кукарача» и др. с успехом используется во многих школах.

До недавнего времени основная масса игр требовала минимальных компьютерных ресурсов. Современная индустрия компьютерных игр нацелена на использование технологии мультимедиа и виртуальной реальности. Для них требуются высокие характеристики компьютера и его периферии.

### *Контрольные вопросы*

1. В чем полезность компьютерных игр? Может ли компьютерная игра приносить вред?
2. На какие основные группы можно разделить компьютерные игры и в чем их различия?

Программное обеспечение ЭВМ
Операционные системы
Прикладное ПО
ПС специального назначения
ПС профессионального уровня
ПС общего назначения
АРМ
АСУ
АСНИ
САПР
АСУТП
Педагогические комплексы
Системы телекоммуникаций
Инженерная графика
Научная графика
Деловая графика
Растровый способ
Векторный способ
Гистограммы
Столбчатая гистограмма
Круговая гистограмма
Трехмерная диаграмма
Линейный график
Структурная схема
Аналитические модели
Координатные модели
Упорядоченные модели
Приближенные модели
Системы программирования
Языки программирования
Процедурные языки
Непроцедурные языки
Функциональные языки
Логические языки
Языки высокого уровня
Макроассемблер
Компилятор
Интерпретатор
Отладчик
Средства редактирования, компоновки, загрузки
Абсолютный модуль
Объектный модуль
Исходный модуль
Синтаксический анализ
Лексический анализ
Семантический анализ
Инструментальные программные средства
Frame Works
Мастер
MS-Works
Текстовые редакторы
Шрифт
Полоса набора
Художественное редактирование
Техническое редактирование
Оформление формул
Оформление таблиц
Оформление заголовков
Формат
Типульные элементы
Word
Page Maker
Super Calk
Excel
Прикладные системы
Электронные таблицы
СУБД
DBASE
Microsoft Access
ЯОД
ЯМД
SQL
БД
Определение данных
Обработка данных
Управление данными
Информационная система
Структура данных
Атрибут
Табличная структура
Сетевые структуры
Иерархические структуры
Технология
бордюры
узоры
Изображение графика
Законы
гармония
мера
пропорции
композиция
перспектива
цвет




**Дополнительная литература к главе 2**

1. *Андриес ВанДааг.* Машинная графика // Современный компьютер. - М.: Мир, 1986.

2. *Ахметов К. С.* Курс молодого бойца. - М.: Компьютер-пресс.1995.

*3. Бабий А. А..Коновалова Н.И.,Пак Н.И.Пара.ией М.Г.* Методическое пособие по работе с IBM PC- Красноярск: КГПИ. «Демос», i991.

4. *Безрукое Н. Н.* Компьютерные вирусы. - М.: Наука, 1991.

5. *Брукс Ф П.* Как проектируются и создаются программные комплексы. /Очерки по системному программированию. - М.: Наука. 1979.

6. *Брябрин В. М.* Программное обеспечение персональных ЭВМ. - М.: Наука, 1990.

7. *Богумирский Б. С.* Руководство пользователя ПЭВМ. Части 1,2. - С.-Пб.: Печатный Двор, 1994.

*8. К.Герман О. В.* Введение в теорию экспертных систем и обработку знаний. -Минск: «Дизайн-ПРО», 1995.

*9. Глазко Т.П., Киреева Т. В., Киреев Н.В.* Работа с инструментальными средствами на IBM. - Красноярск, КГПУ, 1995.

10. *Илюшечкин В.М., Костин А.Е.* Системное программное обеспечение. - М.:
Высшая школа, 1991.

11. Как работать с Microsoft Office для Windows 95. Техническое описание Microsoft Corporation. 1995.

12. *Каратыгин С., Тихонов А., Долголаптев В.* Базы данных: простейшие средства обработки информации. Т. 1,2. Серия «Компьютер для носорога». - М.: ФИА, 1995.

13. *Келжек А. М.* и др. Работа на IBM. - М.:АО «Книга и бизнес». 1992.

14. *Кирмайер М.* Мультимедиа. - С.-Пб.: BHV-Санкт-Петербург, 1994.

15. *Климов Д. М., Руденко В.М.* Методы компьютерной алгебры в задачах механики. - М.: Наука, 1989.

16. Компьютерная вирусология. - Киев: Укр. сов. энцикл., 1991.

17. *Косневски Ч.* Занимательная математика и персональный компьютер. - М.:
Просвещение, 1989.

18. *Липаев В. В.* Проектирование программных средств. - М.: Высшая школа, 1990.

19. *Маейрс Г.* Надежность программного обеспечения. - М.: Мир, 1980.

20. *НаймертаимД.* Word 6.0 для Windows. - М.: Международные отношения, 1995.

21. Операционная система MS DOS. - М.: Радио и связь, 1992.

22. *Одинцова О. П.* Введение в Автокад. Методические указания. - Красноярск, КГПУ, 1996.

23. *Пак Н. И. Рогов В В* Компьютерная графика. - Омск, ОмГПУ, 1995.

24. Проектирование пользовательского интерфейса на персональных компьютерах. Стандарт фирмы IBM. - Вильнюс: DBS Ltd., 1992.

25. *Ратбон Э.* Windows 3.1 для «чайников». - Киев: Диалектика, 1994.

26. *Роджерс Д. Ф., Адамс Дж.А.* «Математические основы машинной графики». -М: Машиностроение, 1980.

27. *Скляров В. А.* Программное и лингвистическое обеспечение персональных ЭВМ. Системы общего назначения. - Минск: Высшая школа, 1992.

28. *Смирнов Н.Н.* Программные средства персональных ЭВМ. - Л.: Машиностроение, 1990.

29. *Смолянинова О. Г.* Основы компьютерной грамотности. - Красноярск, КГПУ, 1996.

30. *Смолянинова О. Г., Яшин А. В.* СУБД MS Access. Метод, указания. - Красноярск, КГПУ, 1997.

31. *Соловьев Г.Н., Никитин В. Д.* Операционные системы ЭВМ. - М.: Высшая школа,1989.

32. *Туранова Л. М.* Элементы компьютерной графики. Уч. пособие. - Красноярск, КГПУ, 1996.

33. *Фигурнов В.Э.* IBM PC для пользователя. Изд.б-е. - М.: ИНФРА, 1995.

34. *ФоксДж.* Программное обеспечение и его разработка. - М.: Мир, 1985.

35. *Фош Д., Ван Дем А.* Основы интерактивной машинной графики. - М.: Мир, 1989.

36. *Хаслер Р., Фанеништих К.* Word 6.0 для Windows. - М.: Экон, 1994.

37. *Шафрин Ю.* Основы компьютерной технологии. - М.: ABF, 1996.

38. *Шенен П., Каспар М.* и др. Математика и САПР, книга 1. - М.: Мир, 1988.

39. *Шикин Е. В.* Начала компьютерной графики. - М.: Диалог-МИФИ, 1994.

40. *Шкаев А. В.* Настольные издательские системы: Справочник. - М.: Радио и связь, 1994.

Олег Анатольевич Дмитриев

# ИНФОРМАТИКА:

курс лекций

для подготовки бакалавров очной и заочной форм обучения по направлению подготовки 23.03.03 «Эксплуатация транспортно-технологических машин и комплексов» - Димитровград: Технологический институт – филиал УлГАУ, 2019.- 205 с.