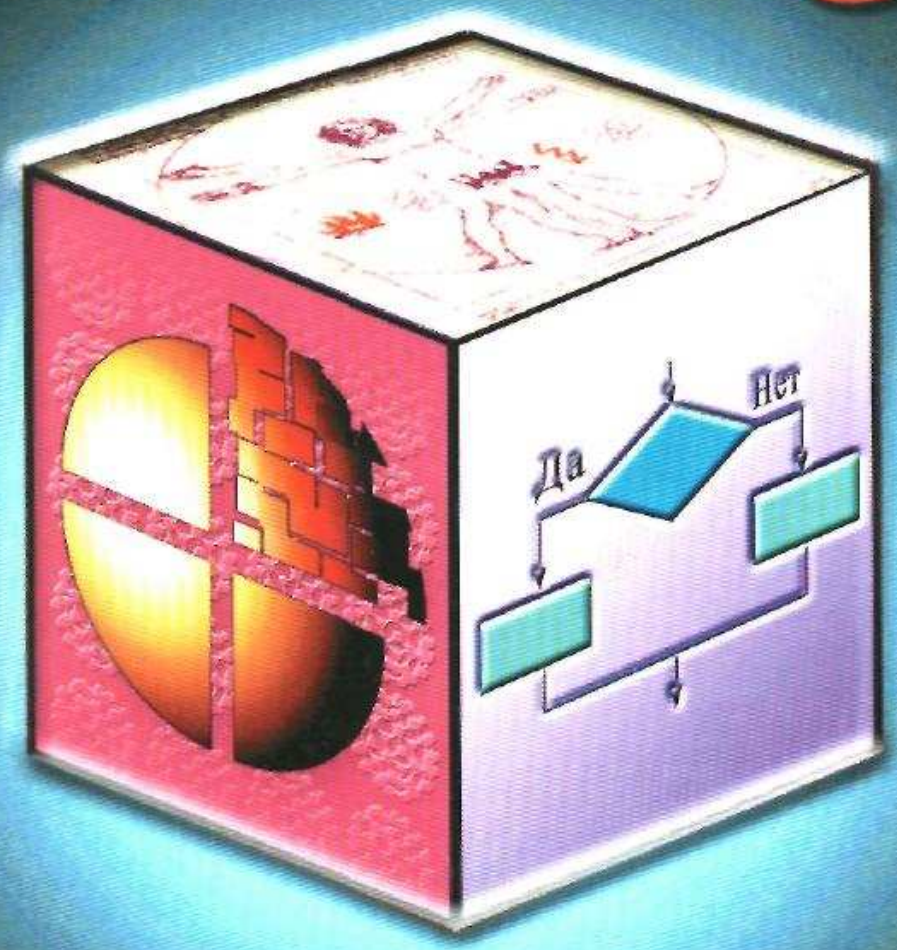
информатика

И. Семакин, Л. Залогова, С. Русаков, Л. Шестакова

БАЗОВЫЙ КУРС

9



БИНОМ

информатика

И. Г. Семакин, Л. А. Залогова
С. В. Русаков, Л. В. Шестакова

Информатика и ИКТ
БАЗОВЫЙ КУРС
Учебник для 9 класса

Допущено
Министерством образования и науки
Российской Федерации

Москва
БИНОМ. Лаборатория знаний
2005

УДК 004.9
ББК 32.97
С30

Семакин И. Г.

С30 Информатика и информационно-коммуникационные технологии. Базовый курс: Учебник для 9 класса / И. Г. Семакин, Л. А. Залогова, С. В. Русаков, Л. В. Шестакова. — М.: БИНОМ. Лаборатория знаний, 2005. — 371 с.: ил.

ISBN 5-94774-230-6

Учебник предназначен для изучения базового курса информатики и ИКТ в 9 классе общеобразовательных школ. Содержание учебника соответствует принятому стандарту по информатике и ИКТ.

Учебник разделен на две части. Первая часть обеспечивает обязательный минимальный уровень изучения предмета. Материал второй части ориентирован на углубленный курс информатики.

Учебник входит в комплект учебно-методической литературы по базовому курсу наряду с учебником для 8 класса, задачником и методическим пособием для учителя.

УДК 004.9
ББК 32.97

По вопросам приобретения обращаться:
«БИНОМ. Лаборатория знаний»
(095) 955-03-98, e-mail: Lbz@aha.ru
http://www.Lbz.ru

ISBN 5-94774-230-6

Оглавление

Введение

Изучив первую часть курса «Информатика и ИКТ», вы освоили лишь некоторые разделы этой большой научной и прикладной области. Данный учебник познакомит вас с новыми разделами предмета, поможет приобрести новые навыки использования информационно-коммуникационных технологий.

В этом учебном году вы познакомитесь с компьютерными сетями, которые используются для передачи информации. Технологии работы с информацией в компьютерных сетях называются коммуникационными технологиями. В результате вам в полной мере станет понятным смысл термина «информационно-коммуникационные технологии» — ИКТ.

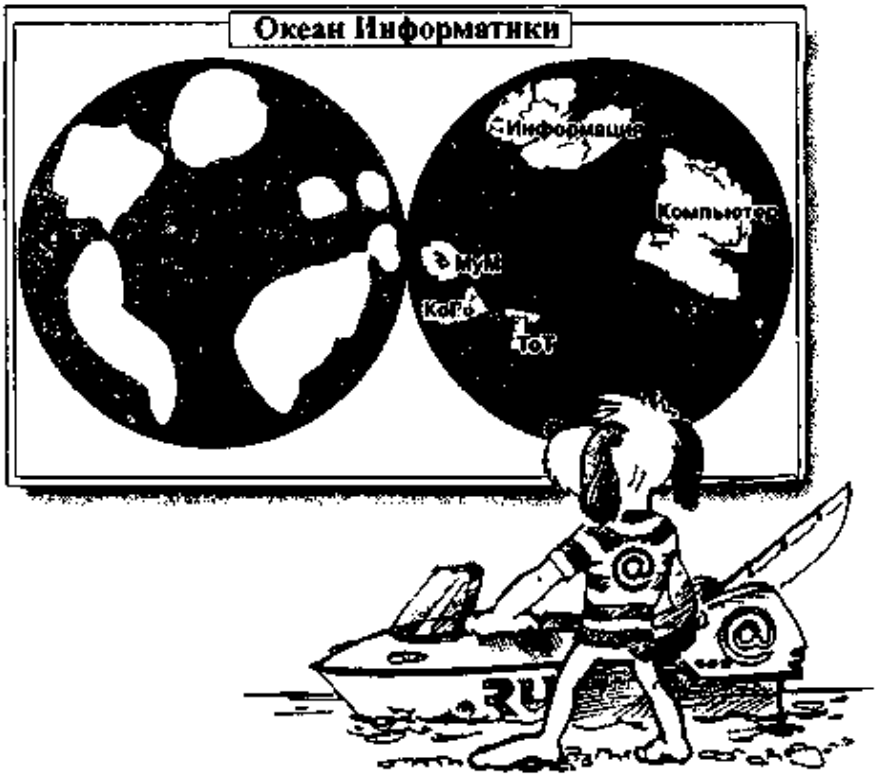
Очень важным приложением компьютерной техники является информационное моделирование. Скоро вы узнаете, что такое модель, как на компьютере создаются и для чего используются информационные модели, какие средства ИКТ применяются для информационного моделирования. В число таких средств входят системы управления базами данных (СУБД) и табличные процессоры. С помощью СУБД на компьютере создаются информационные хранилища, которые называются базами данных. Табличные процессоры применяются для организации вычислений в электронных таблицах. Если такие вычисления относятся к какому-то реальному процессу, то в этом случае можно говорить о том, что в электронных таблицах реализована математическая модель этого процесса.

Управление — еще одна обширная область использования компьютеров. С помощью компьютеров управляют самолетами, ракетами, промышленными установками, производственными коллективами, отраслями экономики и пр. Оказывается, что у таких разнообразных вариантов управления есть общие законы. Они были открыты в науке кибернетике. С основными понятиями этой науки вам предстоит познакомиться.

Всякое управление происходит по определенным правилам, которые называются алгоритмом управления. Алгоритм, записанный на языке программирования, «понятном» компьютеру, называется компьютерной программой. В этой части курса вы научитесь строить алгоритмы и писать несложные программы на языке программирования Паскаль.

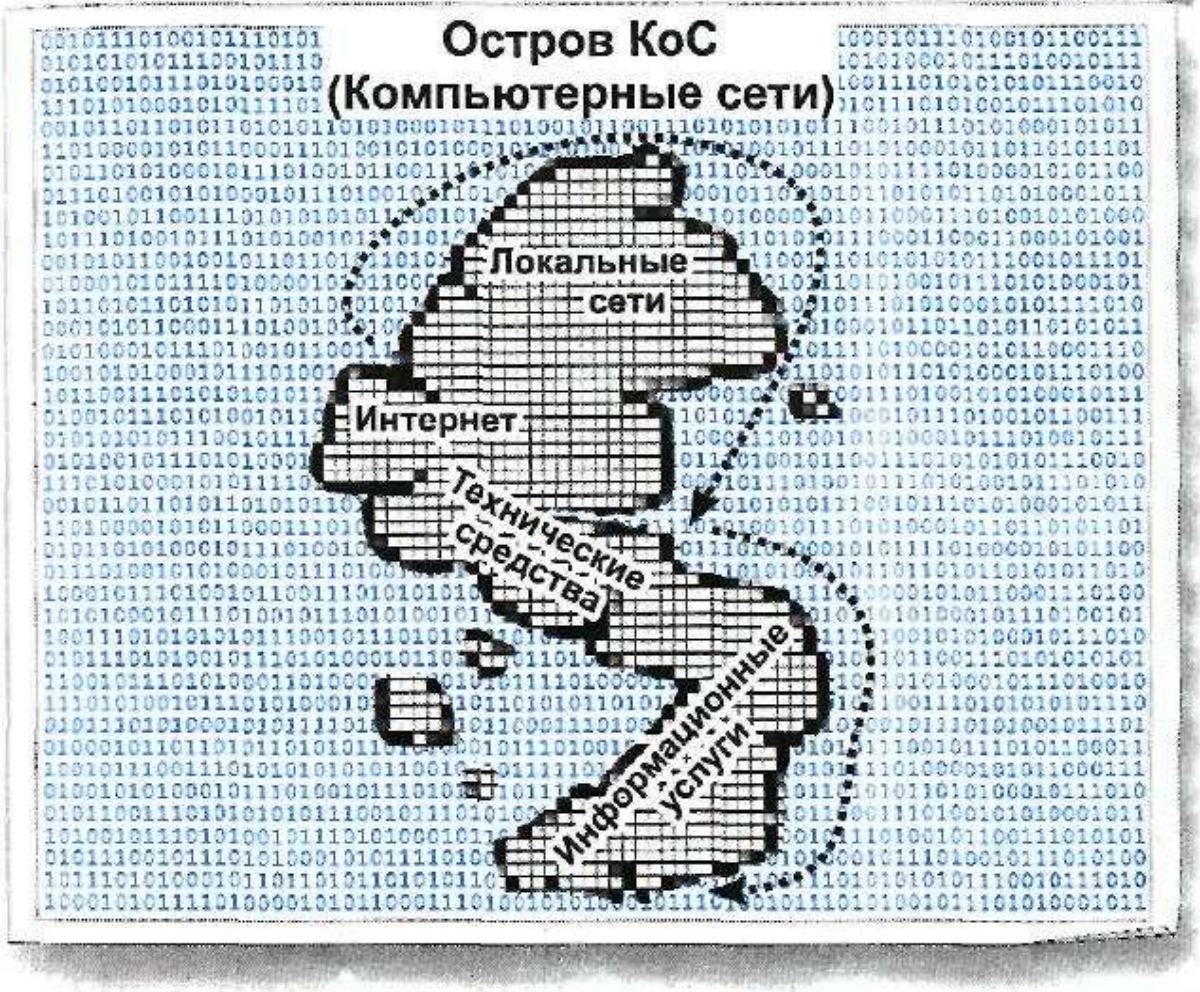
Изобретения различных средств и инструментов для хранения, передачи и обработки информации совершались с древних времен и до наших дней. С историей таких изобретений вы познакомитесь в последней главе учебника. История компьютеров и ИКТ берет начало с середины XX века. С этого времени начинается процесс информатизации различных областей деятельности людей, происходит формирование информационно-ориентированного общества. Проблемы информационного общества изучает раздел информатики, который называется социальной информатикой. О некоторых вопросах социальной информатики вы узнаете в конце нашего курса.

Итак, продолжим наше путешествие по океану Информатики. Некоторые материки и острова в этом океане вы посетили и исследовали в первой части путешествия. Но еще остались неизведанные земли, белые пятна на карте океана знаний. Вместе с нашим отважным капитаном — Точкой-РУ вам предстоит посетить эти земли, нанести их подробное описание на карту знаний. Успехов вам в этом нелегком, но очень интересном путешествии!



Глава 1

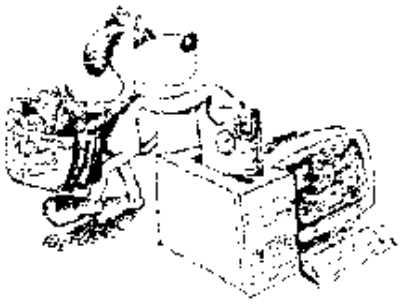
Передача информации в компьютерных сетях



Здесь вы узнаете

- как устроены компьютерные сети
- для чего используются сети
- что такое Интернет
- как не заблудиться в Интернете

§ 1 Как устроена компьютерная сеть



Основные темы параграфа:

- *что такое компьютерная сеть;*
- *локальные сети;*
- *глобальные сети.*

Что такое компьютерная сеть

Вы уже знаете, что при работе компьютера непрерывно происходит информационный обмен между составляющими его устройствами. Передача информации между пользователем и компьютером осуществляется через клавиатуру, дисплей, принтер и другие устройства ввода/вывода. А теперь вы узнаете, как компьютеры обмениваются информацией между собой.

Система компьютеров, связанных каналами передачи информации, называется **компьютерной сетью.**

Локальные сети

Небольшие компьютерные сети, работающие в пределах одного помещения, одного предприятия, называются ***локальными сетями*** (ЛС). Обычно компьютеры одной локальной сети удалены друг от друга на расстояние не более одного километра.

Локальная сеть дает возможность пользователям не только быстрее обмениваться данными друг с другом, но и более эффективно использовать ресурсы объединенных в сеть компьютеров. Такими ресурсами могут быть дисковая память, устройство печати, сканер и другие технические средства, а также программное обеспечение и любая информация в файлах.

С точки зрения организации взаимодействия отдельных элементов ЛС выделяют два типа таких систем:

- ***одноранговую сеть***; в ней все объединенные компьютеры равноправны;
- сеть ***с выделенным сервером***.

Пользователю одноранговой сети могут быть доступны ресурсы всех подключенных к ней компьютеров (в том случае, если эти ресурсы не защищены от постороннего доступа).

В школьных компьютерных классах чаще всего используется ЛС с выделенным сервером, организованная по следующему принципу: имеется одна центральная машина, которая называется ***сервером***, и множество подключенных к ней компьютеров — ***рабочих станций***. Центральная машина обычно имеет большую дисковую память, к ней подключены устройства, которых нет на рабочих станциях: принтер, сканер, модем для выхода в глобальную сеть и пр. На сервере хранится программное обеспечение и другая информация, к которой могут обращаться пользователи сети. Название «сервер» происходит от английского «server» и переводится как «обслуживающее устройство».

На многих предприятиях на базе локальных сетей работают информационные системы. Например, в крупном торговом центре на сервере хранится база данных, содержащая сведения о товарах, имеющихся на складе. Рабочие станции установлены в торговых отделах. На них по запросам продавцов с сервера поступает информация о наличии нужного вида товара. С рабочей станции на сервер передаются сведения о проданном товаре. После этого сервер вносит соответствующие изменения в базу данных.

Основой программного обеспечения ЛС является ***сетевая операционная система***. Важнейшая задача сетевой ОС — поддержка такого режима работы ЛС, чтобы работающие в ней пользователи могли использовать общие ресурсы сети и при этом не мешали бы друг другу.

Глобальные сети

Другой разновидностью компьютерных сетей являются глобальные сети. Дальше речь пойдет именно о них.

Глобальная сеть связывает между собой многие локальные сети, а также отдельные компьютеры, не входящие в локальные сети. Размеры глобальных сетей не ограничены: могут существовать сети от региональных до всемирных.

Глобальную компьютерную сеть называют ***телекоммуникационной сетью***, а процесс обмена информацией по такой сети называют ***телекоммуникацией*** (от греч. «tele» — «вдаль», «далеко» и лат. «comunicato» — «связь»).

Организация связи в глобальных сетях похожа на организацию телефонной связи. Телефон каждого абонента под-

ключен к определенному узлу-коммутатору. Связь между коммутаторами организована таким образом, чтобы любые два абонента, где бы они ни находились, могли бы поговорить друг с другом. И такая телефонная сеть «покрывает» весь мир. Аналогично работают компьютерные сети. Персональный компьютер пользователя сети (его также можно назвать абонентом) подключается к определенному узлу сети. Узлы связаны между собой, и эта связь действует постоянно. На рис. 1.1 узлы сети обозначены У1, У2 и т. д., а компьютеры абонентов — А11, А12 и т. д.

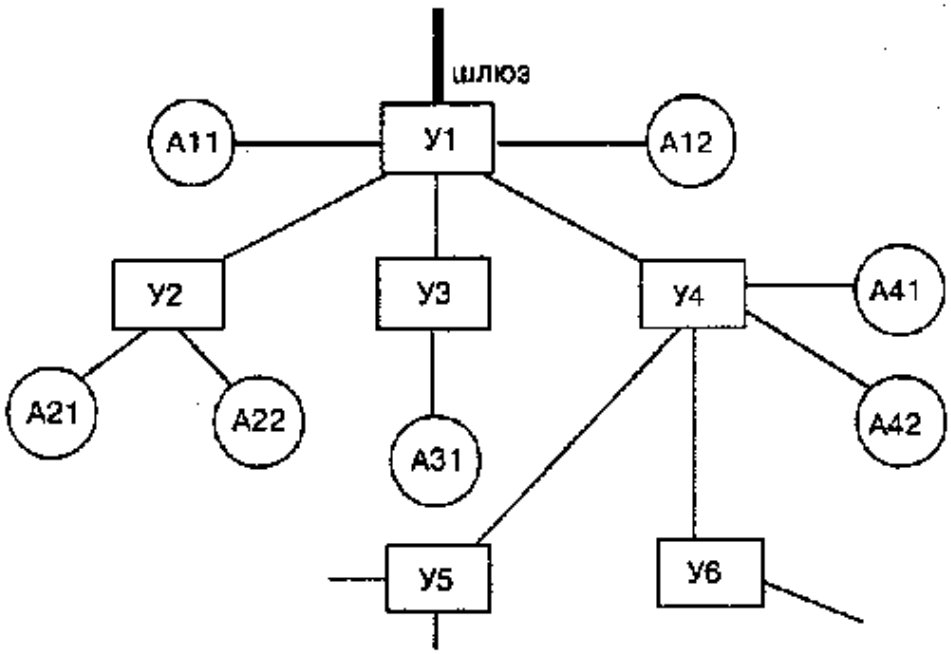


Рис. 1.1. Характерная архитектура глобальной сети

Сети, обслуживающие какую-то отрасль государства (образование, науку, оборону и т. п.), называются ***отраслевыми*** (корпоративными) сетями. Если сеть существует в пределах определенного региона, то она называется ***региональной***.

Каждая региональная или отраслевая компьютерная сеть обычно имеет связь с другими сетями. Для этого один из узлов сети выполняет функцию ***шлюза***. Он соединяется линией связи с аналогичными узлами других сетей*.

Существует *мировая система компьютерных сетей*, через которую можно установить связь с самыми далекими уголками планеты. Эта система называется «Интернет» (англ. «net» — сеть; «Internet» — объединение сетей). Об Интернете речь пойдет немного позже.

---

* Потребность в шлюзе существует лишь в том случае, если в связываемых сетях используются разные протоколы (соглашения).

Коротко о главном

Компьютерная сеть — это множество компьютеров, соединенных линиями передачи информации.

Компьютерные сети бывают локальными и глобальными.

В одноранговых локальных сетях все компьютеры равноправны.

Локальная сеть с выделенным компьютером включает в себя сервер и множество рабочих станций. Сервер используется как хранилище общих информационных ресурсов, а также содержит некоторые технические устройства общего доступа.

Глобальная (телекоммуникационная) сеть — это система связанных между собой локальных сетей и компьютеров отдельных пользователей. Персональные компьютеры пользователей (абонентов) подсоединяются к узлам глобальной сети.

Существуют региональные, отраслевые сети. В настоящее время большинство из них объединено в мировую систему, которая называется «Интернет».

Вопросы и задания

1. Что такое компьютерная сеть?
2. Как устроена локальная сеть? Какие функции она выполняет?
3. Что такое глобальная сеть?
4. Что такое отраслевая сеть; региональная сеть?
5. Как называется всемирная сеть, объединяющая в себе большинство существующих в мире сетей?
6. Придумайте различные способы соединения в сеть четырех компьютеров-серверов. Найдите способ, обеспечивающий самый короткий маршрут передачи информации между любыми двумя абонентами.

§ 2 Электронная почта и другие услуги сетей

Основные темы параграфа:
- *назначение электронной почты;*
- *почтовый ящик, электронный адрес;*
- *структура электронного письма;*
- *телеконференции;*
- *файловые архивы.*

Назначение электронной почты

Какая же информация передается по компьютерным сетям? Самая разнообразная. Это могут быть письма, объявления, реклама, программное обеспечение, компьютерные игры, деловая документация и многое другое. Вся эта информация в виде файлов хранится на магнитных дисках абонентских ПК и серверов.

Обмен письмами в компьютерных сетях называется **электронной почтой** (e-mail).

Электронное письмо — это файл, содержащий электронный адрес получателя и текст письма.

Электронная почта — один из самых популярных видов услуг компьютерных сетей.

Электронная почта работает гораздо быстрее обычной почты. В самый далекий уголок мира письмо может дойти за несколько минут. В течение дня можно несколько раз обменяться письмами со своим корреспондентом на другом континенте.

Почтовый ящик, электронный адрес

Зарегистрированный абонент сети получает на почтовом сервере так называемый ***почтовый ящик.***

Почтовый ящик — это раздел внешней памяти почтового сервера, отведенный для абонента.

Каждому почтовому ящику присваивается свое имя, отличное от других имен. В этот ящик почтовый сервер помещает письма, поступающие к абоненту.

Передать *письмо в почтовый ящик может любой абонент* сети, если он знает ***электронный адрес***. Извлечь (прочитать или переписать на свою машину) письмо из почтового ящика может только его владелец. Доступ к информации защищен паролем, который знает только хозяин ящика.

Что представляет собой электронный адрес? По своей структуре он похож на обычный почтовый адрес, когда письма посылаются на абонентский ящик адресата в почтовом отделении. Вот пример обычного почтового адреса:

Страна	Город	П/о	№ ящика абонента
Россия	Пермь	10	644

А вот пример электронного адреса:

somov@pgu.perm.ru

Он состоит из таких частей:

Страна	Город	Имя сервера	Имя почтового ящика
ru (Россия)	*perm* (Пермь)	*pgu* (ПГУ)	*somov* (Сомов)

Точки и символ @ — разделительные знаки. Разделенные точками части электронного адреса называются ***доменами***. Каждый домен уточняет местоположение в компьютерной сети почтового сервера, обслуживающего адресата. Количество *доменов может быть различным: два, три и более.*

Не нужно думать, что в адресах всегда используется административно-географический принцип (страна–город–район и т. д.). Вся часть адреса, расположенная справа от значка @, является ***доменным именем почтового сервера***, содержащего ящик абонента. Главный принцип состоит в том, чтобы это имя отличалось от имен всех прочих серверов в компьютерной сети.

Структура электронного письма

В целом электронное письмо состоит из «конверта» и текста. Обычно на «конверте» записывается адрес получателя,

адрес отправителя и краткая информация о назначении письма (два последних элемента — необязательные).

Вот пример электронного письма из Перми в Москву:

Куда:	*frolov@mgu.msk.ru*
Откуда:	*somov@pgu.perm.ru*
О чем:	*приглашение*

Приглашаю Вас принять участие в праздновании юбилея Пермского университета. С уважением, Е. Сомов

Отправитель на своем компьютере формирует файл с текстом письма, заполняет «конверт». Затем он выходит на связь с почтовым сервером. Сервер, приняв письмо, тут же пересылает его адресату.

К электронному письму могут быть присоединены самые разнообразные файлы: с графикой, звуком, программами и пр. Адресат их получит вместе с текстом письма (это похоже на вкладывание фотографии в конверт с письмом).

Существенно, что абоненты сети выходят на связь по своему желанию. Сервер же функционирует без перерывов и выходных. Связь абонента с сервером устанавливается лишь в тот момент, когда абонент подключился к сети. Именно тогда сервер и отправляет ему всю корреспонденцию, накопившуюся в почтовом ящике.

Телеконференции

Кроме электронной почты абоненты компьютерных сетей могут получать и другие информационные услуги.

Всем известно, что такое конференция: в одном помещении собираются люди и выступают, задают вопросы, спорят на какую-то общую тему. Бывают научные, производственные, профсоюзные, школьные и другие конференции.

Телеконференция — это тоже общение группы людей по объединяющей их теме. Но для участия в такой конференции не нужно собираться в одно и то же время в одном помещении. Кроме того, телеконференция не ограничена во времени, как традиционная конференция. Она может продолжаться месяцами и годами.

Участники такой конференции — абоненты компьютерной сети. *Телеконференция заключается в обмене электронными письмами между ее участниками.* Сначала в компью-

терной сети объявляется открытие конференции на определенную тему. Телеконференция получает свой электронный адрес. Затем проводится подписка на участие в конференции. После этого каждый абонент, подписавшийся на данную конференцию, будет получать все поступающие в нее материалы. В свою очередь, посылая письмо в адрес конференции, абонент знает, что оно дойдет до всех ее участников.

Существует множество телеконференций, посвященных самым разнообразным темам: науке, образованию, музыке, разведению рыб, компьютерным играм, политике, литературе и пр. Через телеконференции можно распространять какие-то свои авторские работы, договариваться о покупке или продаже. Участники таких конференций всегда имеют самую оперативную информацию в области своих интересов.

Файловые архивы

А можно ли через Интернет получать новое программное обеспечение для своего компьютера? Оказывается, можно! Для этого существует служба распространения файлов — файловые архивы. Серверы, которые поддерживают их работу, называются FTP-серверами. В файловых архивах можно найти не только программы, но и файлы с самыми разнообразными информационными объектами: рисунками, фотографиями, видеоклипами, музыкой и др.

Программное обеспечение, бесплатно распространяемое через FTP-серверы, нередко носит рекламную функцию. Например, время действия таких программ может оказаться ограниченным. Если вы захотите и дальше пользоваться данной программой, то вам будет рекомендовано оплатить приобретение ее рабочей версии.

Среди прочих услуг компьютерных сетей: доски объявлений, базы данных, форумы прямого общения (chat-конференции), Интернет-телефония и пр.

Коротко о главном

Электронная почта — это система обмена письмами между абонентами компьютерных сетей.

Каждый абонент имеет свой почтовый ящик — поименованную область дисковой памяти на почтовом сервере, куда помещается входящая корреспонденция.

Почтовый ящик имеет уникальное имя; владелец получает доступ к своему почтовому ящику через пароль.

Электронное письмо — это текстовый файл, содержащий «конверт» с электронным адресом получателя и текст письма.

Телеконференция — это система обмена информацией на определенную тему между абонентами сети. Абонент, подписавшийся на конференцию, получает все ее материалы в свой почтовый ящик.

Файловые архивы позволяют через Интернет пополнять программное обеспечение своего компьютера.

Среди других услуг сетей: доски объявлений, базы данных, форумы прямого общения, интернет-телефония.

? *Вопросы и задания*

1. Что такое электронная почта?
2. Из чего состоит электронное письмо?
3. Где располагается почтовый ящик абонента? Что в него заносится?
4. Что представляет собой электронный адрес?
5. Что такое телеконференция? Как стать участником телеконференции?
6. Какие еще услуги предоставляются абонентам компьютерных сетей?

§ 3 Аппаратное и программное обеспечение сети

Основные темы параграфа:

- *технические средства глобальной сети;*
- *что такое протоколы;*
- *программное обеспечение глобальной сети; технология «клиент—сервер».*

Для работы компьютерных сетей требуются определенные аппаратные (технические) и программные средства.

Технические средства глобальной сети

Компьютер-сервер — это высокопроизводительный компьютер, обеспечивающий информационные услуги в сети. Обычно сервер постоянно находится во включенном состоянии, занимаясь приемом/передачей информации по сети.

Линии связи. Для информационных связей в компьютерных сетях часто используются телефонные линии связи (коммутируемые линии). Это удобно и дешево, поскольку система телефонной связи уже давно организована, налажена и охватывает весь мир. Каждый раз для организации связи между абонентом и узлом сети с помощью коммутируемых линий нужно «дозваниваться» по соответствующему номеру. В другое время эта же линия используется для обычных телефонных разговоров. Для связи узлов коммутации между собой могут использоваться специально выделенные телефонные линии. В этом случае связь действует постоянно и не требуется набирать телефонный номер.

Самую высококачественную связь поддерживают оптико-волоконные линии цифровой связи. Для связи между удаленными узлами сети используется также беспроводная спутниковая связь, радиорелейные линии.

Терминал абонента. Это персональная ЭВМ, используемая абонентом для получения и передачи информации.

Модем. Информация в ЭВМ имеет дискретную двоичную форму, по линиям же телефонной связи передается непрерывный (аналоговый) электрический сигнал. Для того чтобы соединить персональный компьютер с телефонной сетью, необходимо специальное устройство, согласующее их работу. Такое устройство носит название «модем» (МОдулятор — ДЕМодулятор). Модуляция — это преобразование информации из дискретной цифровой формы в аналоговую, которое производится при передаче абонентом информации в сеть. В учебнике для 8-го класса (§ 24) такое преобразование было названо цифро-аналоговым — ЦАП. Демодуляция — это обратное, аналого-цифровое преобразование (АЦП), происходящее во время приема информации.

Схема связи между абонентом и сервером с помощью модема показана на рис. 1.2.

Модем может быть выполнен в виде отдельного устройства, подключаемого к компьютеру через стандартный последовательный порт связи, который имеется у каждого

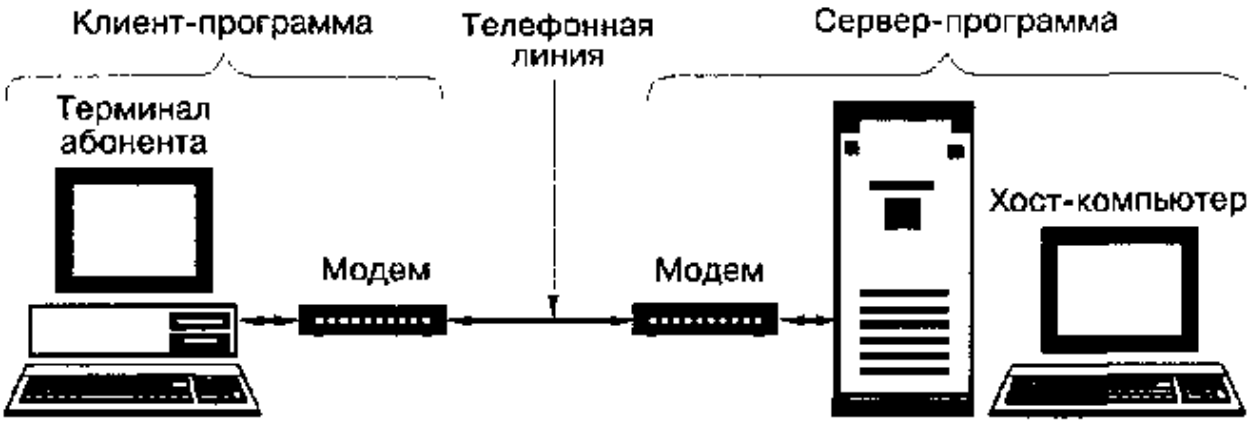


Рис. 1.2. Схема связи между абонентом и узлом сети по телефонной линии

компьютера. Бывают также встроенные модемы в виде электронной платы, устанавливаемой внутри компьютера.

Одной из важнейших характеристик модема является скорость передачи данных, измеряемая в битах в секунду. Вот характерные значения скорости передачи для современных модемов: 14 400 бит/с, 19 Кбит/с. Современные высокоскоростные модемы имеют скорости 28 Кбит/с, 56 Кбит/с.

Пусть используемый модем во время работы в сети может переслать 14 400 бит/с (1800 символов в секунду). Тогда передача полной страницы текста (около 2500 знаков) займет около полутора секунд. Переключение скорости модема на 28 Кбит/с удвоит скорость передачи. Модем, допускающий высокую скорость, как правило, позволяет работать и с низкой скоростью.

Серьезные проблемы при передаче данных часто возникают из-за плохого качества телефонных линий. Это ведет к искажению передаваемой информации. Иногда один искаженный бит может обесценить всю информацию. Многие типы модемов обладают способностью корректировать ошибки. Такие модемы называют интеллектуальными. Применение коррекции ошибок снижает скорость передачи данных, но зато увеличивает ее надежность.

Что такое протоколы

В компьютерных сетях абоненты могут использовать различные марки компьютеров, типы модемов, линии связи, коммуникационные программы. Чтобы все это оборудование работало согласованно, работа сетей подчиняется специальным техническим соглашениям, которые называются протоколами.

Протоколы работы сети — это стандарты, определяющие формы представления и способы пересылки сообщений, процедуры их интерпретации, правила совместной работы различного оборудования.

Программное обеспечение глобальной сети. Технология «клиент—сервер»

Обслуживанием сетевых информационных услуг занимается как компьютер абонента, так и узловой компьютер-сервер. Для каждой из услуг (электронная почта, передача файлов, базы данных и др.) должно существовать определенное программное обеспечение на машине пользователя и на сервере. Организация программного обеспечения, принятая в современных сетях, носит название ***«технология «клиент — сервер»***.

Любая сетевая услуга на машине абонента обслуживается программой, которая называется клиент-программой (или короче — клиентом); на узловом сервере эта услуга обеспечивается работой сервер-программы. Таким образом, слово «сервер» употребляется по отношению как к обслуживающему компьютеру, так и к его программному обеспечению.

Программы «клиент» и «сервер» устанавливают связь между собой, и каждая из них выполняет свою часть работы по обслуживанию абонента. ***Клиент-программа*** подготавливает запрос пользователя, передает его по сети, а затем принимает ответ. ***Сервер-программа*** принимает запрос, подготавливает ответную информацию и передает ее пользователю (см. рис. 1.2). При этом программы «клиент» и «сервер» используют общие протоколы — общаются между собой на одном и том же «сетевом языке». При предоставлении разных услуг могут использоваться разные протоколы.

Сервер-программа электронной почты организует рассылку по сети корреспонденции, передаваемой абонентом, а также прием в почтовый ящик поступающей информации.

Клиент-программу электронной почты обычно называют ***почтовой программой***. Ее назначение — подготовка и отправка писем абонента, получение поступающей корреспонденции из почтового ящика абонента и выполнение ряда сервисных услуг.

Почтовая программа создает на магнитном диске машины абонента следующие разделы:

- папки для хранения почтовой корреспонденции;
- адресный справочник.

Количество и названия папок, создаваемых разными почтовыми программами, могут быть разными. Практически всегда имеется следующий набор папок:

- «Входящие» — для хранения принятой корреспонденции;
- «Исходящие» — для хранения подготовленных, но еще не отправленных писем;
- «Отправленные» — для хранения отправленных писем;

В адресный справочник пользователь заносит электронные адреса своих постоянных корреспондентов.

Все клиент-программы обеспечивают пользователю электронной почты следующие режимы работы.

Настройка. В этом режиме устанавливаются необходимые параметры для правильной работы модема и почтовой программы. Обычно настройка производится во время подключения абонента к сети.

Просмотр почты. Во время просмотра можно отсортировать полученные письма (например, по дате отправления, по имени отправителя и т. д.) и выбрать письмо для просмотра. В этом режиме помимо визуального просмотра письма можно выполнить следующие действия над письмами:

- удаление из папки;
- переписывание в файл;
- пересылка другому адресату;
- печать на принтере.

Подготовка/редактирование писем. Письмо подготавливается в специальном рабочем поле — бланке письма, который содержит адресную часть, место для краткой информации о письме, место для указания имен файлов, отправляемых с этим письмом. Для записи на бланк используется встроенный текстовый редактор. Заполнение адресной части можно осуществить выбором из списка адресов. Прилагаемые к письму файлы выбираются из каталогов диска.

Отправка электронной корреспонденции. В этом режиме подготовленное письмо отправляется по сети адресату, при этом можно использовать дополнительные услуги, например уведомление о получении.

Коротко о главном

Техническими средствами сетей являются узловые компьютеры-серверы, персональные компьютеры абонентов (терминалы), линии связи, модемы.

В качестве линий связи используются коммутируемые телефонные линии или выделенные каналы — телефонные, кабельные, спутниковые, радиорелейные.

Модем используется в том случае, если каналом связи является телефонная линия. Модем преобразует двоичный код компьютера в аналоговый электрический сигнал телефонной сети при передаче информации (модуляция) и производит обратное преобразование (демодуляция) во время приема информации.

Основной характеристикой модема является скорость приема/передачи информации, которая измеряется в битах в секунду — бит/с.

Работа сетей подчиняется определенным протоколам — стандартам на представление и преобразование передаваемой по сетям информации.

Программное обеспечение компьютера-сервера, обеспечивающее предоставление информационных услуг пользователям сети, называется сервер-программой. На компьютере абонента работают соответствующие клиент-программы (технология «клиент — сервер»).

Клиент-программа электронной почты дает возможность абоненту принимать и отправлять письма, просматривать полученную корреспонденцию, формировать текст письма, вести адресный справочник, вести почтовый архив.

? *Вопросы и задания*

1. Что входит в технические средства компьютерных сетей?
2. Почему в качестве линий связи в компьютерных сетях чаще всего используются телефонные линии?
3. Что такое модем? Каково его назначение в сети?
4. Какая величина является основной характеристикой работы модема? В каких единицах она измеряется?
5. Сколько символов текста можно передать за 5 секунд, используя модем, работающий со скоростью 1200 бит/с; 14 400 бит/с?
6. Какими возможностями обладает интеллектуальный модем?

7. Что такое протокол сети?
8. Что такое технология «клиент — сервер»?
9. Какую работу выполняет сервер-программа электронной почты?
10. Перечислите режимы работы клиент-программы электронной почты.

§ 4 Интернет и Всемирная паутина

Основные темы параграфа:

- *Интернет — мировое содружество сетей;*
- *что такое World Wide Web;*
- *Web-сервер, Web-страница, Web-сайт;*
- *гиперструктура WWW;*
- *браузер — клиент-программа WWW; проблема поиска информации в Интернете.*

Интернет — мировое содружество сетей

Хотелось бы вам заглянуть в резиденцию президента США — Белый дом, или посетить Лувр — крупнейший художественный музей мира, или узнать, какая погода в Антарктиде, или получить сведения о спектаклях, идущих сегодня вечером в московских театрах? Всего этого и многого другого можно достичь, не выходя из-за стола, на котором установлен персональный компьютер, подключенный к мировой сети ***Интернет***.

Интернет объединяет в себе тысячи локальных, отраслевых, региональных компьютерных сетей всего мира. Отдельный пользователь, который не является абонентом какой-то из перечисленных сетей, также может подключиться к Интернету через ближайший узловой центр.

Все перечисленные выше услуги компьютерных сетей (электронная почта, телеконференции, файловые архивы и пр.) работают и в Интернете. При этом могут возникать лишь проблемы языка общения. Языком международного общения в мировой сети является английский. Вот вам еще один стимул старательно изучать английский язык!

Что такое World Wide Web

Самой интересной услугой, предоставляемой пользователям Интернета начиная с 1993 года, стала возможность работы с информационной системой ***World Wide Web*** (сокращенно — WWW). Это словосочетание можно перевести как «всемирная паутина». Именно работа с WWW имелась в виду, когда в начале этого параграфа вам предлагались всякие информационные чудеса.

Очень трудно дать точное определение, что такое WWW. Эту систему можно сравнить с огромной энциклопедией, страницы которой разбросаны по компьютерам-серверам, объединенным сетью Интернет. Чтобы получить нужную информацию, пользователь должен добраться до соответствующей страницы энциклопедии. Быть может, имея в виду такую аналогию, создатели WWW ввели понятие Web-страницы.

Web-сервер, Web-страница, Web-сайт

Web-страница — это основная информационная единица WWW. Она представляет собой отдельный документ, хранящийся на ***Web-сервере***. Страница имеет свое имя (подобно номеру страницы в энциклопедии), по которому к ней можно обратиться.

Информация на Web-странице может быть самой разной: текст, рисунок, фотография, мультимедиа. На Web-страницах помещают также рекламу, справочную информацию, научные статьи, последние новости, иллюстрированные издания, художественные каталоги, прогноз погоды и многое, многое другое. Проще сказать: на Web-страницах есть «всё».

Некоторое количество Web-страниц могут быть связаны тематически и образовывать ***Web-сайт***. У каждого сайта есть главная страница, которая называется домашней (Home page). Это своеобразный титульный лист, начиная с которого можно просматривать документы, хранящиеся на сервере. Обычно домашняя страница содержит оглавление — названия разделов. Чтобы обратиться к нужному разделу, достаточно подвести указатель мыши к названию раздела и щелкнуть кнопкой мыши.

Гиперструктура WWW

Однако просматривать Web-страницы совсем не обязательно подряд, перелистывая их, как в книге. Важнейшим свойством WWW является ***гипертекстовая организация***

связей между Web-страницами. Причем эти связи действуют не только между страницами на одном сервере, но и между разными серверами WWW.

Обычно ключевые слова, от которых идут гиперсвязи, выделяются на Web-странице цветом или подчеркиванием. Щелкнув мышью на таком слове, вы по скрытой ссылке перейдете к просмотру другого документа. Причем этот документ может находиться на другом сервере, в другой стране, на другом континенте. Чаще всего пользователь Интернета понятия не имеет, где находится сервер, с которым он в данный момент общается. Образно говоря, за один сеанс работы можно несколько раз «облететь» вокруг земного шара.

Роль ключа для связи может выполнять не только текст, но и рисунок, фотография, указатель на звуковой документ. В таком случае вместо термина «гипертекст» употребляется термин ***«гипермедиа»***.

На одну и ту же Web-страницу можно выйти самыми разными путями. Аналогия со страницами книги здесь уже не работает. В книге страницы имеют определенную последовательность. Web-страницы такой последовательности не имеют. Переход от одной страницы к другой происходит по гиперсвязям, образующим сеть, которая напоминает паутину. Отсюда и происходит название системы.

Обобщая сказанное, можно дать следующее определение:

World Wide Web — это распределенная по всему миру информационная система с гиперсвязями, существующая на технической базе всемирной сети **Интернет**.

Браузер — клиент-программа WWW. Проблема поиска информации в Интернете

Перемещаться по «паутине» пользователю помогает специальное программное обеспечение, которое называется ***Web-браузером*** от английского «browse» — «осматривать, изучать». С помощью браузера нужную информацию можно найти разными способами. Самый короткий путь — с помощью адреса Web-страницы. Вы набираете на клавиатуре этот адрес, нажимаете клавишу ввода и попадаете сразу на место.

Другой путь — поиск. Вы можете начать движение со своей домашней страницы по гиперсвязям. При этом есть

опасность не туда уйти, запутаться в «паутине», попасть в тупик. Впрочем браузер позволяет вернуться назад на любое количество шагов, продолжить поиск по другому маршруту. Такой поиск подобен блужданию в незнакомом лесу (правда, менее опасен).

Хорошими помощниками в навигации по WWW являются специальные ***поисковые программы***. Они «знают» всё или почти всё о WWW. Такой программе достаточно указать набор ключевых слов по интересующей вас теме, и она выдаст список ссылок на подходящие Web-документы. Если список окажется слишком длинным, нужно добавить еще какие-нибудь уточняющие термины.

Пользователь Интернета во время сеансов работы в сети оказывается погруженным в информационное пространство с неограниченными ресурсами. В последнее время стал распространенным термин ***«киберпространство»***, под которым понимается вся совокупность мировых систем телекоммуникаций и циркулирующей в них информации.

Система WWW очень быстро развивается. Уже сейчас все ее ресурсы плохо поддаются обзору. Выпускаются толстые справочники, каталоги, которые устаревают быстрее, чем телефонные книги. Поэтому одновременно с увеличением объема информации совершенствуется система поиска в World Wide Web.

Коротко о главном

Интернет — всемирная глобальная компьютерная сеть.

World Wide Web — Всемирная паутина: распределенная по всему миру информационная система с гиперсвязями, существующая на технической базе мировой сети Интернет.

Web-страница — отдельный документ WWW.

Web-сервер — компьютер в сети Интернет, хранящий Web-страницы и соответствующее программное обеспечение для работы с ними.

Web-сайт — совокупность тематически связанных страниц.

Гипермедиа — система гиперсвязей между мультимедиа документами.

Web-браузер — клиент-программа для работы пользователя с WWW.

Поиск нужного документа в WWW может происходить: путем указания его адреса; путем перемещения по «паутине» гиперсвязей; путем использования поисковых программ.

Киберпространство — совокупность мировых систем телекоммуникаций и циркулирующей в них информации.

Вопросы и задания

1. Что такое Интернет?
2. Как переводится словосочетание «World Wide Web»?
3. Что такое WWW?
4. Какую информацию можно извлечь из WWW?
5. Как организована связь между Web-страницами?
6. В чем аналогия между WWW и паутиной?
7. Что такое гипермедиа?
8. Что такое Web-сервер?
9. Какими методами в WWW можно найти нужную страницу?

§ 5 Способы поиска в Интернете

Основные темы параграфа:

- *три способа поиска в Интернете;*
- *поисковые серверы;*
- *язык запросов поисковой системы.*

Три способа поиска в Интернете

Интернет в целом и Всемирная паутина, в частности, предоставляют абоненту доступ к тысячам серверов и миллионам Web-страниц, на которых хранится невообразимый объем информации. Как не потеряться в этом «информационном океане»? Для этого необходимо научиться искать и находить нужную информацию в сети.

Как уже было сказано, существуют три основных способа поиска информации в Интернете.

1. *Указание адреса страницы.* Это самый быстрый способ поиска, но его можно использовать только в том случае, если точно известен адрес документа.

2. *Передвижение по гиперссылкам.* Это наименее удобный способ, так как с его помощью можно искать документы, только близкие по смыслу текущему документу. Если текущий документ посвящен, например, музыке, то, используя гиперссылки этого документа, вряд ли можно будет попасть на сайт, посвященный спорту.
3. *Обращение к поисковому серверу (поисковой системе).* Использование поисковых серверов — наиболее удобный способ поиска информации. В настоящее время в русскоязычной части Интернета популярны следующие поисковые серверы:

   Яndex;
   Rambler;
   Апорт.

Существуют и другие поисковые системы. Например, эффективная система поиска реализована на сервере почтовой службы mail.ru.

Поисковые серверы

Наиболее доступным и удобным способом поиска информации во Всемирной паутине является использование поисковых систем. При этом поиск информации можно осуществлять по каталогам, а также по набору ключевых слов, характеризующих отыскиваемый текстовый документ.

Рассмотрим использование поисковых серверов более подробно. ***Поисковый сервер*** содержит большое количество ссылок на самые различные документы, и все эти ссылки систематизированы в тематические каталоги. Например: спорт, кино, автомобили, игры, наука и др. Причем эти ссылки устанавливаются сервером самостоятельно, в автоматическом режиме путем регулярного просмотра всех появляющихся во Всемирной паутине Web-страниц. Кроме того, поисковые серверы предоставляют пользователю возможность поиска информации по ключевым словам. После ввода ключевых слов поисковый сервер начинает просматривать документы на других Web-серверах и выводить на экран ссылки на те документы, в которых встретились указанные слова. Обычно результаты поиска сортируются по убыванию специального рейтинга документов, который показывает, насколько полно заданный документ отвечает условиям поиска или насколько часто он запрашивается в сети.

Язык запросов поисковой системы

Группа ключевых слов, сформированная по определенным правилам — с помощью ***языка запросов***, называется запросом к поисковому серверу. Языки запросов к разным поисковым серверам очень похожи. Подробнее об этом можно узнать, посетив раздел «Помощь» нужного поискового сервера. Рассмотрим правила формирования запросов на примере поисковой системы Яndex.

Синтаксис оператора	Что означает оператор	Пример запроса
пробел или &	Логическое И (в пределах предложения)	лечебная физкультура
&&	Логическое И (в пределах документа)	рецепты && (плавленый сыр)
|	Логическое ИЛИ	фото | фотография | снимок | фотоизображение
+	Обязательное наличие слова в найденном документе	+быть или +не быть
( )	Группирование слов	(технология | изготовление) (сыра | творога)
~	Бинарный оператор И НЕ (в пределах предложения)	банки ~ закон
~~ или –	Бинарный оператор И НЕ (в пределах документа)	путеводитель по Парижу ~~ (агентство | тур)
/(n m)	Расстояние в словах (минус (–) — назад, плюс (+) — вперед)	поставщики /2 кофе музыкальное /(–2 4) образование вакансии ~ /+1 студентов
" "	Поиск фразы	"красная шапочка" Эквивалентно: красная /+1 шапочка
&&/(n m)	Расстояние в предложениях (минус (–) — назад, плюс (+) — вперед)	банк && /1 налоги

Чтобы получить лучшие результаты поиска, необходимо запомнить несколько простых правил:

1. Не искать информацию только по одному ключевому слову.

2. Лучше не вводить ключевые слова с прописной буквы, так как это может привести к тому, что не будут найдены те же слова, написанные со строчной буквы.
3. Если в итоге поиска вы не получили никаких результатов, проверьте, нет ли в ключевых словах орфографических ошибок.

Современные поисковые системы предоставляют возможность подключения к сформированному запросу семантического анализатора. С его помощью можно, введя какое-либо слово, выбрать документы, в которых встречаются производные от этого слова в различных падежах, временах и пр.

? *Вопросы и задания*

1. В чем состоят три основных способа поиска информации во Всемирной паутине?
2. *Каким образом ссылки на конкретные документы попадают в* поисковые системы?
3. Сформулируйте сложный запрос, состоящий из нескольких ключевых слов, используя язык запросов системы Яndex.

Чему вы должны научиться, изучив главу 1

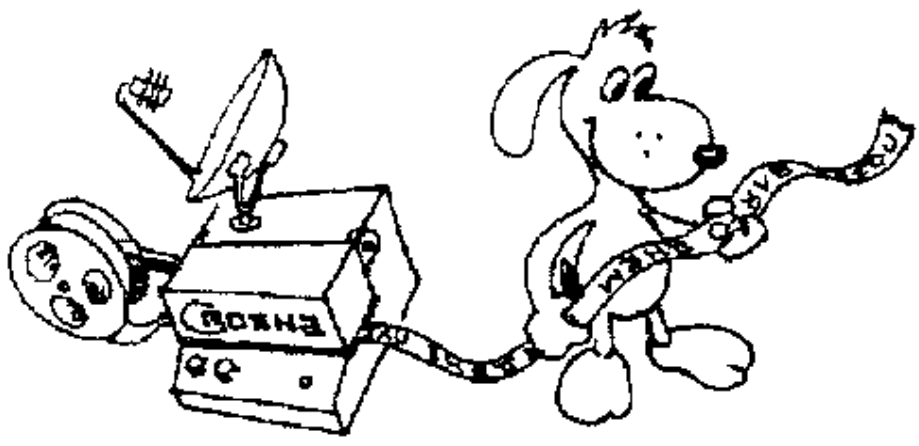
Осуществлять обмен информацией с сервером локальной сети школьного компьютерного класса.

Отправлять и получать письма по электронной почте.

Заказывать новости из телеконференций.

Получать информацию с Web-страницы, адрес которой вам известен.

Искать информацию в сети с помощью поисковых программ.



КОММУНИКАЦИОННЫЕ

Локальные сети

Назначение:
1) совместное использование общих аппаратных средств (накопителей, принтеров, модемов и пр.);
2) оперативный обмен данными;
3) информационная система предприятия

Организация:
- одноранговая сеть;
- сеть с выделенным сервером:
  сервер – рабочие станции

Программное обеспечение:
сетевая операционная система

Система основных понятий главы 1

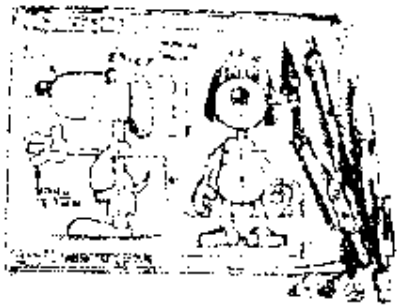
Глава 2

Информационное моделирование

Здесь вы узнаете

- что такое натурные и информационные модели
- какие существуют виды информационных моделей
- как они реализуются на компьютере

§ 6 Что такое моделирование



Основные темы параграфа:
- *натурные модели;*
- *информационные модели;*
- *формализация.*

Натурные модели

Сейчас речь пойдет об очень важном в науке понятии — понятии ***модели***. Это слово многим знакомо. Возможно, кто-то из вас занимается техническим моделированием — строит модели кораблей, автомобилей или самолетов. Такие модели воспроизводят некоторые свойства реальных устройств, например форму, способность плавать, ездить или летать. Можно привести и другие примеры моделей: глобус — это модель земного шара, манекен в магазине — модель человека, макет в мастерской архитектора — модель застройки города.

Выше перечислены примеры материальных моделей. Их еще называют ***натурными моделями***.

Как правило, моделируемый объект представляет собой сложную систему. Например, автомобиль состоит из корпуса, двигателя, колес, рулевого управления, салона и пр. Модель автомобиля, построенная школьником, много проще. В ней, например, может отсутствовать двигатель, электропитание, рулевое управление и другие части, размер ее меньше размера настоящего автомобиля.

Любая модель воспроизводит только те свойства оригинала, которые понадобятся человеку при ее использовании. Например, манекен и производственный робот можно назвать моделями человека. Манекен нужен для того, чтобы на него можно было надеть одежду для рекламы или для удобства работы портного, но способности ходить, мыслить или разговаривать от него не требуется. Поэтому манекен должен воспроизводить лишь форму и размер человеческого тела.

Цель создания производственного робота совсем другая. Робот должен воспроизводить некоторые физические действия человека: уметь брать и перемещать детали, закручивать и раскручивать болты и пр. Но для достижения этих целей внешнего сходства с человеком совсем не требуется.

Свойства модели зависят от цели моделирования. Модели одного и того же объекта будут разными, если они создаются для разных целей.

Информационные модели

Кроме натурных, существуют еще ***информационные модели***. Нетрудно понять, что для информатики именно они и представляют наибольший интерес*.

Если натурная модель объекта моделирования — это его физическое подобие, то информационная модель — это его описание. Способ описания может быть самым разным: вербальным, т. е. словесным описанием на естественном языке, математическим, графическим и др. Например, чертеж корабля является его графическим описанием, а стало быть, информационной моделью корабля (рис. 2.1).

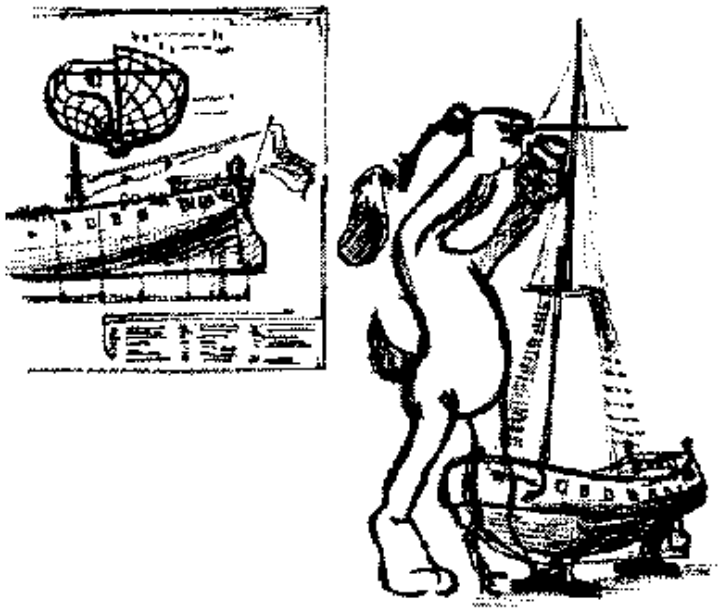


Рис. 2.1. Моделирование натурное и информационное

Слова ***«объект моделирования»*** надо понимать в самом широком смысле. Это может быть *материальный объект*:

* В науке существует еще одна разновидность моделей: воображаемые (идеальные) модели. Например, в физике: материальная точка, абсолютно твердое тело, идеальный газ; в математике: геометрическая точка, бесконечность и пр.

корабль, комета, живая клетка; *явление природы*: гроза, солнечное затмение; *процесс*: полет ракеты, ядерный взрыв, изменение стоимости акций на фондовой бирже.

Моделирование — это деятельность человека по созданию модели (натурной или информационной).

Так же как и натурные, информационные модели одного и того же объекта, предназначенные для разных целей, могут существенно различаться.

Вот пример. Нередко людям приходится заполнять всевозможные анкеты, личные карточки. Такие документы можно рассматривать как различные информационные модели человека. По форме они одинаковые (анкеты), а по содержанию — разные. Например, в личной карточке работника предприятия, которая хранится в отделе кадров, о нем имеются следующие сведения: фамилия, имя, отчество, пол, год рождения, место рождения, национальность, адрес проживания, образование, семейное положение. А в медицинскую карточку того же самого человека занесены следующие данные: фамилия, имя, отчество, пол, год рождения, группа крови, вес, рост, хронические заболевания. В обществе охотников, членом которого является этот же человек, о нем хранится третий набор сведений. Как видите, разное назначение — разные информационные модели.

Современным инструментом для информационного моделирования является компьютер. С его помощью воспроизводятся самые сложные объекты, процессы, явления. Такая модель обычно отображается на экране в виде статического (неподвижного) или анимированного (подвижного) изображения, может сопровождаться звуком, т. е. использовать технологию мультимедиа.

Модель — это упрощенное подобие реального объекта. Модель отражает лишь некоторые свойства объекта, существенные для достижения цели моделирования.

Для обозначения сложных объектов, состоящих из множества взаимосвязанных частей, в науке используется тер-

мин *«система»*. В большинстве случаев объектами моделирования являются сложные системы: природные, технические, общественные и др.

> Модель используется как заменитель реальной системы для воспроизведения отдельных ее функций, для прогноза ее поведения в определенных условиях.

Формализация

Что такое формализация? В этом слове заключается суть информационного моделирования. Информационная модель описывает объект моделирования в *форме* каких-либо знаков: букв, цифр, картографических элементов, математических или химических формул и т. п.

> **Формализация** есть результат перехода от реальных свойств объекта моделирования к их *формальному обозначению* в определенной знаковой системе.

Самой формализованной наукой является математика.

Коротко о главном

Модель — это упрощенное подобие реального объекта, отражающее существенные свойства объекта с точки зрения цели моделирования.

Модели бывают натурными и информационными.

Информационная модель представляет собой описание объекта моделирования.

Резным целям моделирования одного и того же объекта могут соответствовать разные модели.

Объект моделирования следует рассматривать как систему — целое, состоящее из взаимосвязанных частей.

Формализация есть результат перехода от реальных свойств моделируемой системы к их формальному обозначению в определенной знаковой системе.

? Вопросы и задания

1. Что такое модель?
2. Какие свойства реальных объектов воспроизводят следующие модели?
   - Муляжи продуктов в витрине магазина;
   - чучело птицы;
   - заводной игрушечный автомобиль.
3. Что такое информационная модель?
4. Поясните разницу между технической моделью самолета и информационной моделью самолета (чертежом).
5. Можно ли элементы следующего списка считать информационными моделями? Если «да», то что для них является объектом моделирования?
   - Расписание уроков;
   - программа телевидения;
   - рецепт на получение лекарства.
5. Придумайте несколько вариантов описания одного и того же объекта для разных целей.
6. Назовите процессы или явления, которые невозможно или очень сложно воспроизвести в натурных моделях.
7. Что такое формализация? Можно ли выставление учителем оценки за ваш ответ на уроке назвать формализацией?

§ 7 Графические информационные модели

Основные темы параграфа:
- *карта как информационная модель;*
- *чертежи и схемы;*
- *график — модель процесса.*

Карта как информационная модель

Можно ли назвать информационной моделью карту местности (рис. 2.2)? Безусловно, можно! Во-первых, ***карта*** описывает конкретную местность, которая является для нее *объектом моделирования*. Во-вторых, это *графическая информация*. Карта создается *с определенной целью*: с ее помощью можно добраться до нужного населенного пункта. Кроме того, используя линейку и учитывая масштаб карты, можно опре-

Рис. 2.2. Карта местности

делить расстояние между различными пунктами. Однако никаких более подробных сведений о населенных пунктах, кроме их положения, эта карта не дает.

Чертежи и схемы

Другими знакомыми вам примерами графических информационных моделей являются чертежи, схемы, графики.

Чертеж должен быть очень точным, на нем указываются все необходимые размеры. Например, чертеж болта нужен для того, чтобы, глядя на него, токарь мог выточить болт на станке (рис. 2.3).

У схемы электрической цепи нет никакого внешнего сходства с реальной электрической цепью (рис. 2.4). Электроприборы (лампочка, источник тока, конденсатор, сопротивление) изображены символическими значками, а линии — это соединяющие их проводники электрического тока. Электрическая

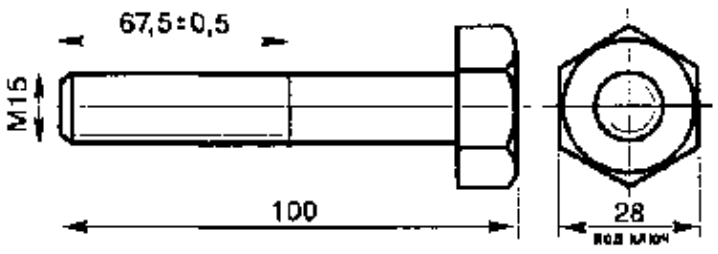


Рис. 2.3. Чертеж болта

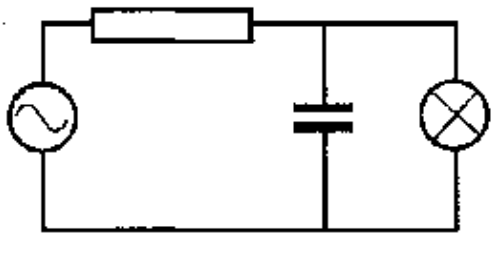


Рис. 2.4. Схема электрической цепи

схема нужна для того, чтобы понять принцип работы цепи, чтобы можно было рассчитать в ней токи и напряжения, чтобы при сборке цепи правильно соединить ее элементы.

На рис. 2.5 приведена схема.

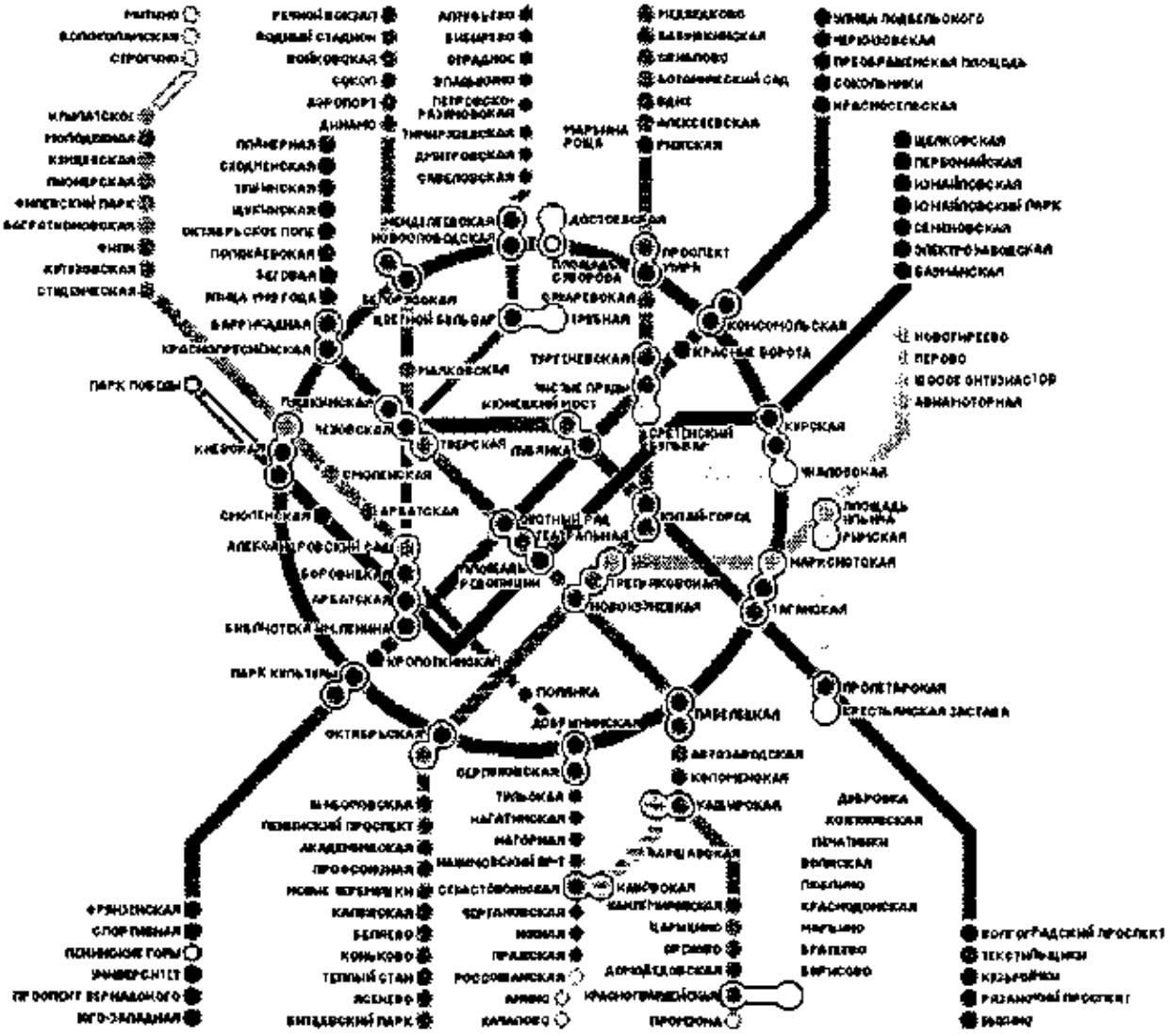


Рис. 2.5. Схема московского метрополитена

Схема — это графическое отображение состава и структуры сложной системы.

Структура — это определенный порядок объединения элементов системы в единое целое.

Структуру московского метрополитена называют радиально-кольцевой.

График — модель процесса

Для отображения различных процессов часто прибегают к построению *графиков*. На рис. 2.6 изображен график изменения температуры в течение некоторого периода.

Рис. 2.6. График изменения температуры

С картами, чертежами, схемами, графиками вы имели дело и раньше. Просто раньше вы их не связывали с понятием информационной модели.

Коротко о главном

Наглядными способами представления информационных моделей являются графические изображения: карты, чертежи, схемы, графики.

Вопросы и задания

1. Приведите различные примеры графических информационных моделей.
2. Постройте графическую модель вашей квартиры. Что это: карта, схема, чертеж?
3. Какая форма графической модели (карта, схема, чертеж, график) *применима для отображения процессов? Приведите* примеры.
4. Постройте графическую модель собственной успеваемости по двум различным дисциплинам школьной программы (самой любимой и самой «нелюбимой»). Спрогнозируйте по этой модели свой дальнейший процесс обучения данным предметам.

§ 8 Табличные модели

Основные темы параграфа:
- *таблицы типа «объект—свойство»;*
- *таблицы типа «объект—объект»;*
- *двоичные матрицы.*

Таблицы типа «объект—свойство»

Еще одной распространенной формой информационной модели является ***прямоугольная таблица***, состоящая из строк и столбцов. Использование таблиц настолько привычно, что для их понимания обычно не требуется дополнительных объяснений.

В качестве примера рассмотрим таблицу 2.1.

Таблица 2.1. Домашняя библиотека

Номер	Автор	Название	Год	Полка
0001	Беляев А.Р.	Человек-амфибия	1987	5
0002	Кервуд Д.	Бродяги севера	1991	7
0003	Тургенев И.С.	Повести и рассказы	1982	1
0004	Олеша Ю.К.	Избранное	1987	5
0005	Беляев А.Р.	Звезда КЭЦ	1990	5
0006	Тынянов Ю.Н.	Кюхля	1979	1
0007	Толстой Л.Н.	Повести и рассказы	1986	1
0008	Беляев А.Р.	Избранное	1994	7

При составлении таблицы в нее включается лишь та информация, которая интересует пользователя. Например, кроме тех сведений о книгах, которые включены в таблицу 2.1, существуют и другие: издательство, количество страниц, стоимость. Однако для составителя таблицы 2.1 было достаточно сведений, которые позволяют отличить одну книгу от другой (столбцы «Автор», «Название», «Год») и найти книгу на полках книжных стеллажей (столбец «Полка»). Предполагается, что все полки пронумерова-

ны и, кроме того, каждой книге присвоен свой инвентарный номер (столбец «Номер»).

Таблица 2.1 — это *информационная модель* книжного фонда домашней библиотеки.

Таблица может отражать некоторый процесс, происходящий во времени (табл. 2.2).

Таблица 2.2. Погода

День	Осадки	Температура (градусы С)	Давление (мм рт. ст.)	Влажность (проценты)
15.03.04	Снег	−3,5	746	67
16.03.04	Без осадков	0	750	62
17.03.04	Туман	1,0	740	100
18.03.04	Дождь	3,4	745	96
19.03.04	Без осадков	5,2	760	87

Показания снимались в течение пяти дней в одно и то же время суток. Глядя на таблицу, легко сравнить разные дни по температуре, влажности и пр. Данную таблицу можно рассматривать как *информационную модель процесса изменения состояния погоды.*

Таблицы 2.1 и 2.2 относятся к наиболее часто используемому типу таблиц. Их будем называть ***таблицами типа «объект–свойство»***. В одной строке такой таблицы содержится информация об одном объекте (книга в библиотеке или состояние погоды в 12-00 в данный день). Столбцы — отдельные характеристики (свойства) объектов.

Конечно, строки и столбцы в таблицах 2.1 и 2.2 можно поменять местами, повернуть их на 90°. Иногда так и делают. Тогда строки будут соответствовать свойствам, а столбцы — объектам. Но чаще всего таблицы строят так, что строк в них больше, чем столбцов. Как правило, объектов больше, чем свойств.

Таблицы типа «объект—объект»

Другим распространенным типом таблиц являются таблицы, отражающие взаимосвязи между разными объектами. Назовем их ***таблицами типа «объект–объект»***. Вот понятный каждому школьнику пример таблицы успеваемости (табл. 2.3).

Таблица 2.3. Успеваемость

Ученик	Русский	Алгебра	Химия	Физика	История	Музыка
Аликин Петр	4	5	5	4	4	5
Ботов Иван	3	3	3	3	3	4
Волков Илья	5	5	5	5	5	5
Галкина Нина	4	4	5	2	4	4

Строки относятся к ученикам — это первый вид объектов; столбцы — к школьным предметам — второй вид объектов. В каждой клетке на пересечении строки и столбца — оценка, полученная данным учеником по данному предмету.

Таблица 2.4 тоже имеет тип «объект–объект». Однако, в отличие от предыдущей таблицы, в ней строки и столбцы относятся к одному и тому же виду объектов. В этой таблице содержится информация о наличии дорог между населенными пунктами с карты из § 2.

Таблица 2.4. Дороги

	Дачи	Озерная	Подгорная	Елово	Бобры
Дачи	1	1	1	1	0
Озерная	1	1	0	1	0
Подгорная	1	0	1	0	1
Елово	1	1	0	1	1
Бобры	0	0	1	1	1

Двоичные матрицы

В математике прямоугольная таблица, составленная из чисел, называется ***матрицей***. Если матрица содержит только нули и единицы, то она называется ***двоичной матрицей***. Числовая часть таблицы 2.4 представляет собой двоичную матрицу.

Таблица 2.5 также содержит двоичную матрицу.

Таблица 2.5. Факультативы

Ученик	Геология	Цветоводство	Танцы
Русанов	1	0	1
Семенов	1	1	0
Зотова	0	1	1
Шляпина	0	0	1

В ней приведены сведения о посещении четырьмя учениками трех факультативов. Вам уже должно быть понятно, что единица обозначает посещение, ноль — непосещение. Из этой таблицы следует, например, что Русанов посещает геологию и танцы, Семенов — геологию и цветоводство и т. д.

В таблицах, представляющих собой *двоичные матрицы, отражается качественный характер связи между объектами* (есть дорога — нет дороги; посещает — не посещает и т. п.). Таблица 2.3 содержит количественные характеристики успеваемости учеников по предметам, выраженные оценками пятибалльной системы.

Мы рассмотрели только два типа таблиц: «объект–свойство» и «объект–объект». На практике используются и другие, гораздо более сложные таблицы.

Коротко о главном

Для представления информационных моделей широко используются прямоугольные таблицы.

В таблице типа «объект–свойство» одна строка содержит информацию об одном объекте. Столбцы — отдельные характеристики (свойства) объектов.

В таблице типа «объект–объект» отражается взаимосвязь между различными объектами.

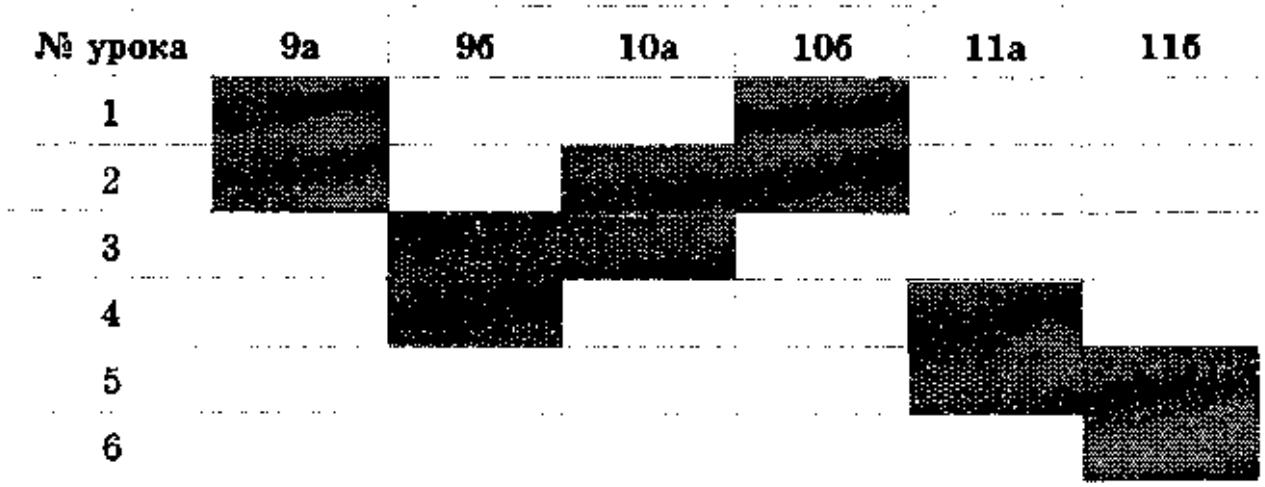
Числовая прямоугольная таблица называется матрицей. Матрица, составленная из нулей и единиц, называется двоичной матрицей.

Двоичная матрица отражает качественный характер связей между объектами.

Вопросы и задания

1. В чем состоит удобство табличного представления информации?
2. Приведите примеры таблиц, с которыми вам приходится иметь дело в школе и дома. Определите тип, к которому они относятся: «объект–свойство» или «объект–объект».
3. Что такое матрица? Что такое двоичная матрица?
4. Представьте в табличной форме сведения об увлечениях ваших одноклассников. Какой тип таблицы вы используете для этой цели?
5. Использование табличной модели часто облегчает решение информационной задачи. В следующей таблице закрашенные клетки в расписании занятий соответствуют урокам физкультуры в 9–11 классах школы.

Расписание занятий



Выполните следующие задания:

- определите, какое минимальное количество учителей физкультуры требуется при таком расписании;
- найдите один из вариантов расписания, при котором можно обойтись двумя учителями физкультуры;
- в школе три учителя физкультуры: Иванов, Петров, Сидоров; распределите между ними уроки в таблице так, чтобы ни у кого не было «окон» (пустых уроков);
- распределите между тремя учителями уроки так, чтобы нагрузка у всех была одинаковой.

6. В компьютерной сети узловым является сервер, с которым непосредственно связаны все остальные серверы. Дана следующая двоичная матрица. В ней С1, С2, С3, С4, С5 — обозначения серверов сети.

	C1	C2	C3	C4	C5
C1	1	0	0	1	0
C2	0	1	0	1	0
C3	0	0	1	1	0
C4	1	1	1	1	1
C5	0	0	0	1	1

Определите, какой сервер является узловым.

§ 9 Информационное моделирование на компьютере

Основные темы параграфа:

- *основное преимущество компьютера перед человеком;*
- *для чего нужны математические модели;*
- *компьютерная математическая модель;*
- *что такое вычислительный эксперимент;*
- *управление на основе моделей;*
- *имитационное моделирование.*

Основное преимущество компьютера перед человеком

Современным инструментом для информационного моделирования является компьютер. Конечно, на компьютере можно писать тексты (строить вербальные модели), рисовать карты и схемы (графические модели), строить таблицы (табличные модели). Но при таком использовании компьютера в моделировании его возможности проявляются не в полной мере.

Главное преимущество компьютера перед человеком — *способность к быстрому счету*. Современные компьютеры считают со скоростями в сотни тысяч, миллионы и даже миллиарды операций в секунду!

Учитывая, что расчеты производятся над многозначными числами (10–20 десятичных цифр), вычислительные способности человека нельзя даже сравнивать с компьютерными. Эти феноменальные вычислительные возможности проявляются, прежде всего, в ***компьютерном математическом моделировании***.

Для чего нужны математические модели

Многие процессы, происходящие в природе, в технике, в экономических и социальных системах, описываются сложными математическими соотношениями. Это могут быть уравнения, системы уравнений, системы неравенств и пр., которые являются ***математическими моделями*** описываемых процессов.

Математическая модель — это описание моделируемого процесса на языке математики.

В прежние времена, до появления ЭВМ, ученые стремились создавать такие математические модели, которые можно было бы просчитать вручную или с помощью несложных вычислительных механизмов. Поэтому математические модели были относительно простыми. Но простая модель не всегда хорошо описывает процесс. Ошибка расчетов по такой модели может быть слишком большой и полностью обесценить результат.

Еще в XVIII–XIX веках ученые-математики начали изобретать методы решения таких математических задач, которые не удавалось решить точно, аналитически. Например, вы знаете, что квадратное уравнение всегда можно решить точно, а вот кубическое — уже не всегда. Такие методы называются ***численными методами***. Они сводят решение любой задачи к последовательности арифметических операций. Но эта цепочка арифметических вычислений может быть очень длинной. И чем точнее мы хотим получить решение, тем она длиннее.

Может оказаться так, что для решения сложной задачи численным методом ученому потребуется вся жизнь. А может и этого не хватить! Например, какой смысл начинать расчет прогноза погоды на завтрашний день, если для этого потребуется несколько лет работы?

Компьютерная математическая модель

Появление компьютеров сняло эти проблемы. Стало возможным проводить расчеты сложных математических моделей за приемлемое время. Например, рассчитать погоду на завтрашний день до его наступления. Ученые перестали себя ограничивать в сложности создаваемых математических моделей, полагаясь на быстродействие компьютеров.

Компьютерная математическая модель — это программа, реализующая расчеты состояния моделируемой системы по ее математической модели.

Что такое вычислительный эксперимент

Использование компьютерной математической модели для исследования поведения объекта моделирования называется ***вычислительным экспериментом***. Говорят также: «численный эксперимент».

Вычислительный эксперимент в некоторых случаях может заменить реальный физический эксперимент.

Впечатляющий пример использования такой возможности — прекращение испытаний ядерного оружия, которые сопровождались значительным экологическим ущербом. Благодаря очень точным математическим моделям и мощным компьютерам стало возможно просчитать все последствия, к которым приводит изменение в конструкции ядерной бомбы. Образно говоря, удалось «взорвать бомбу» внутри компьютера, ничего не разрушив.

Важным свойством компьютерных математических моделей является возможность визуализации результатов расчетов. Этим целям служит использование компьютерной графики.

Представление результатов в наглядном виде — важнейшее условие для их лучшего понимания. Например, результаты расчетов распределения температуры в некотором объекте представляются в виде его разноцветного изображения: участки с самой высокой температурой окрашиваются в красный цвет, а с самой холодной — в синий. Участки с промежуточными значениями температуры окрашиваются в цвета спектра, равномерно переходящие от красного к синему (рис. 2.7).

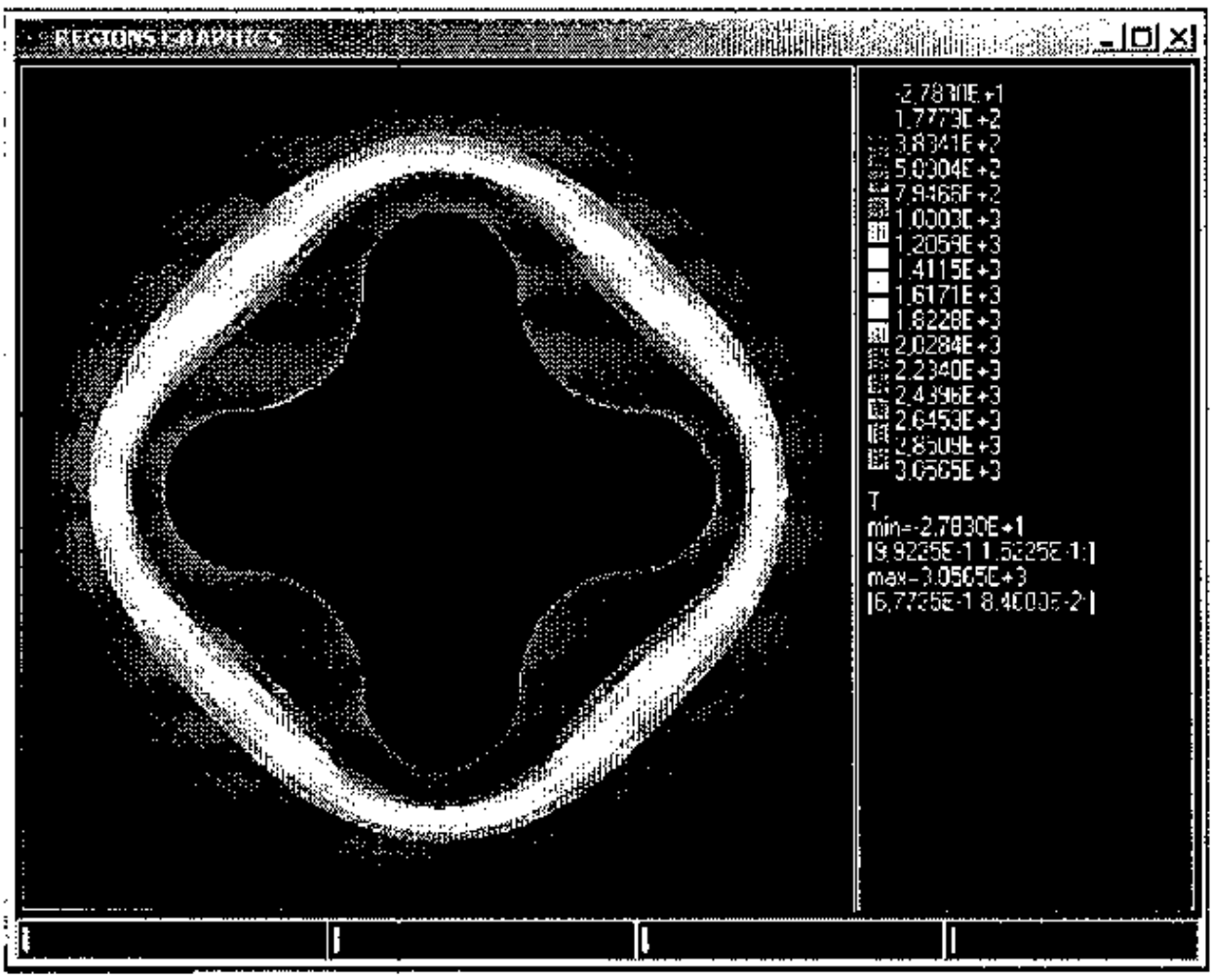


Рис. 2.7. Графическое представление результатов расчетов распределения температур по сечению твердотопливного ракетного двигателя

Для изображения изменяющихся со временем (динамических) результатов используют графическую анимацию.

Компьютерная графика позволяет человеку в процессе проведения численного эксперимента «заглянуть» в недоступные места исследуемого объекта. Можно получить изображение любого сечения объекта сложной формы с отображением рассчитываемых характеристик: температурных полей, давления и пр. В реальном физическом эксперименте такое можно сделать далеко не всегда. Например, невозможно выполнить измерения внутри работающей доменной печи или внутри звезды. А на модели это сделать можно.

Управление на основе моделей

Еще одно важное направление компьютерного математического моделирования связано с *использованием компьютеров в управлении*. Компьютеры используют для управления работой химических реакторов на заводах, атомных реакторов на электростанциях, ускорителей элементарных частиц в физических лабораториях, полета автоматических космических станций и т. д.

> Управляя производственной или лабораторной установкой, компьютер должен просчитывать ее характеристики для того, чтобы *вовремя снять показания с датчиков или оказать управляющее воздействие: включить реле, открыть клапан и т. п.*

Все расчеты производятся по заложенным в программу управления математическим моделям. Важно, чтобы результаты этих расчетов получались в режиме реального времени управляемого процесса.

Имитационное моделирование

Имитационное моделирование — особая разновидность моделирования на компьютере.

> **Имитационная модель воспроизводит поведение сложной системы, элементы которой могут вести себя случайным образом. Иначе говоря, поведение которых заранее предсказать нельзя.**

Такое поведение в математике называется ***стохастическим***. Из курса физики вам знакомо явление броуновского движения: хаотического перемещения легких частиц на поверхности жидкости из-за неравномерных ударов молекул с разных сторон. Нельзя точно рассчитать траекторию броуновской частицы, но ее можно сымитировать на экране компьютера. Отсюда и происходит название — имитационная модель.

К имитационным моделям относятся ***модели систем массового обслуживания***: например, системы торговли, автосервиса, скорой помощи, в которых появление заявок на обслуживание и длительность обслуживания одной заявки — события случайные.

> Задачи, решаемые с помощью имитационных моделей систем массового обслуживания, заключаются в *поиске режимов работы служб сервиса* (магазинов, автозаправок и пр.), *уменьшающих время ожидания клиентов*.

Еще одним популярным объектом для имитационного моделирования являются *транспортные системы*: сеть городских дорог, перекрестки, светофоры, автомобили. Модель имитирует движение транспортных потоков по городским улицам (рис. 2.8). Эксперименты на такой модели позволяют найти режимы управления движением (работа светофоров), уменьшающие возможность возникновения пробок. Работа имитационной модели всегда визуализируется на экране компьютера.

Коротко о главном

Компьютерная математическая модель — это программа, реализующая расчеты состояния моделируемой системы по ее математической модели.

Высокое быстродействие компьютеров позволяет быстро решать достаточно сложные математические задачи в процессе моделирования.

Вычислительный эксперимент — использование компьютерной математической модели для исследования поведения моделируемой системы.

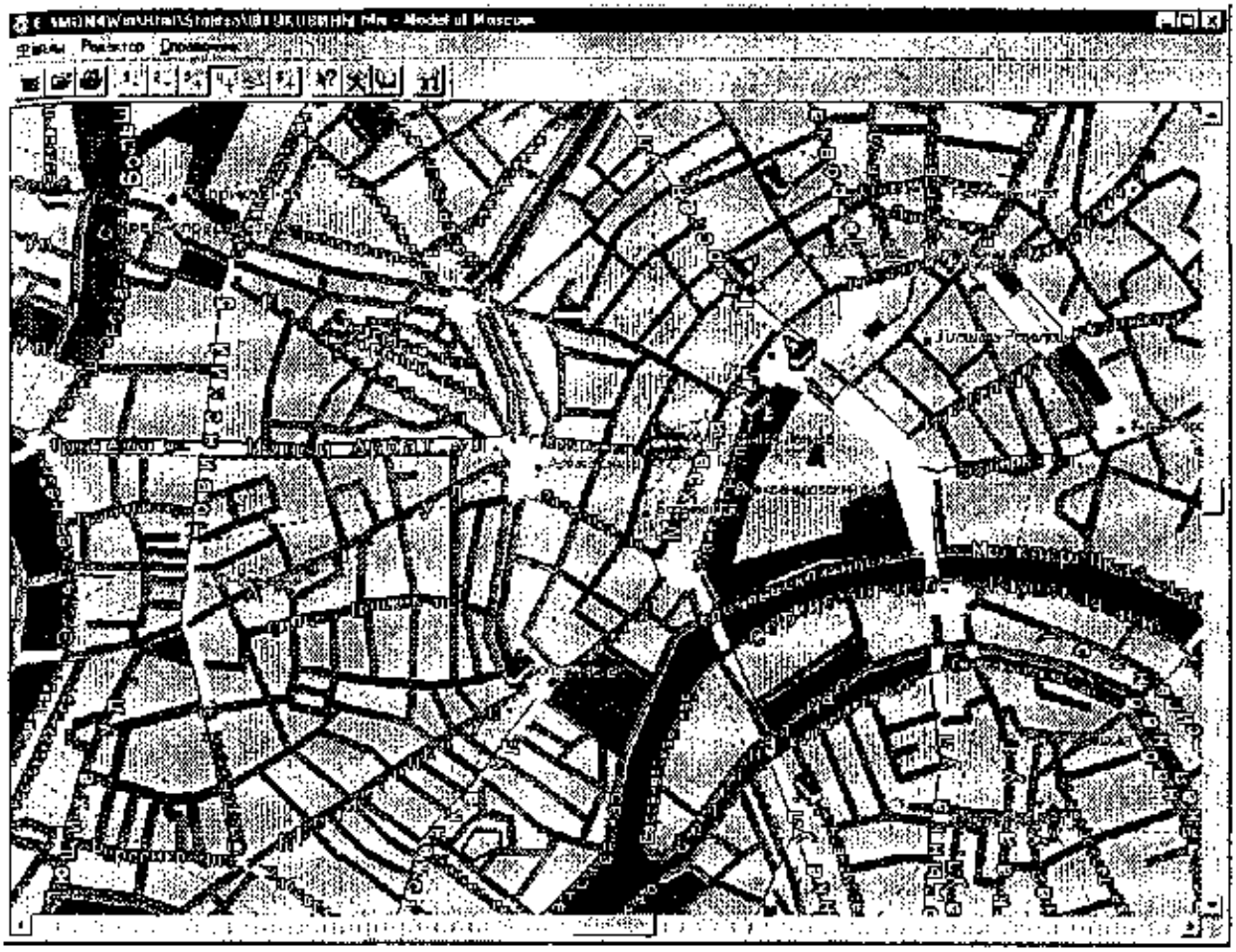


Рис. 2.8. Окно программы, имитирующей движение городского транспорта

Компьютерное управление техническими устройствами происходит в процессе расчетов по математическим моделям в режиме реального времени.

Имитационная модель воспроизводит поведение сложной системы, элементы которой могут вести себя случайным образом.

Вопросы и задания

1. Что общего и в чем различие понятий «математическая модель» и «компьютерная математическая модель»?
2. Расчет прогноза погоды на современном компьютере с быстродействием 1 млн операций в секунду длится в течение 1 часа. Оцените, сколько времени понадобилось бы для этого человеку, имеющему в своем распоряжении арифмометр (механический калькулятор)?
3. В чем состоит особенность компьютерного математического моделирования в процессе управления техническим устройством?

4. Самолет находится на высоте 5000 метров. Обнаружилась неисправность работы двигателя. Самолет начал падать. Бортовой компьютер производит диагностику неисправности и сообщает пилоту о необходимых действиях. Для решения этой задачи ему нужно выполнить 10^8 вычислительных операций. Быстродействие компьютера — 1 млн оп/сек. Успеет ли летчик спасти самолет, если минимальная высота, на которой самолет можно вывести из пике, — 2000 метров?
5. В каких ситуациях используется имитационное моделирование?
6. Придумайте по одному примеру формы использования компьютерной графики для вычислительного эксперимента, для компьютерного управления и для имитационной модели.

Чему вы должны научиться, изучив главу 2

Строить табличные информационные модели по словесному описанию объектов и их свойств.

ИНФОРМАЦИОННОЕ

Виды и типы моделей

Натурные модели

Технические: автомобиля, самолета и пр.
Глобус, манекен, муляж, макет здания и др.

Информационные модели

- Вербальные
- Графические
- Табличные
- Математические

Описание объекта моделирования на естественном языке

Карты, схемы, чертежи, графики

Таблицы типа О-С, О-О, двоичные матрицы

Количественные характеристики и связь между ними

Общие свойства моделей

Объекты моделирования:
- материальные объекты;
- явления природы;
- процессы

Ограниченность модели: отражает лишь часть свойств объекта моделирования

Неоднозначность модели: разные модели одного объекта, созданные для разных целей

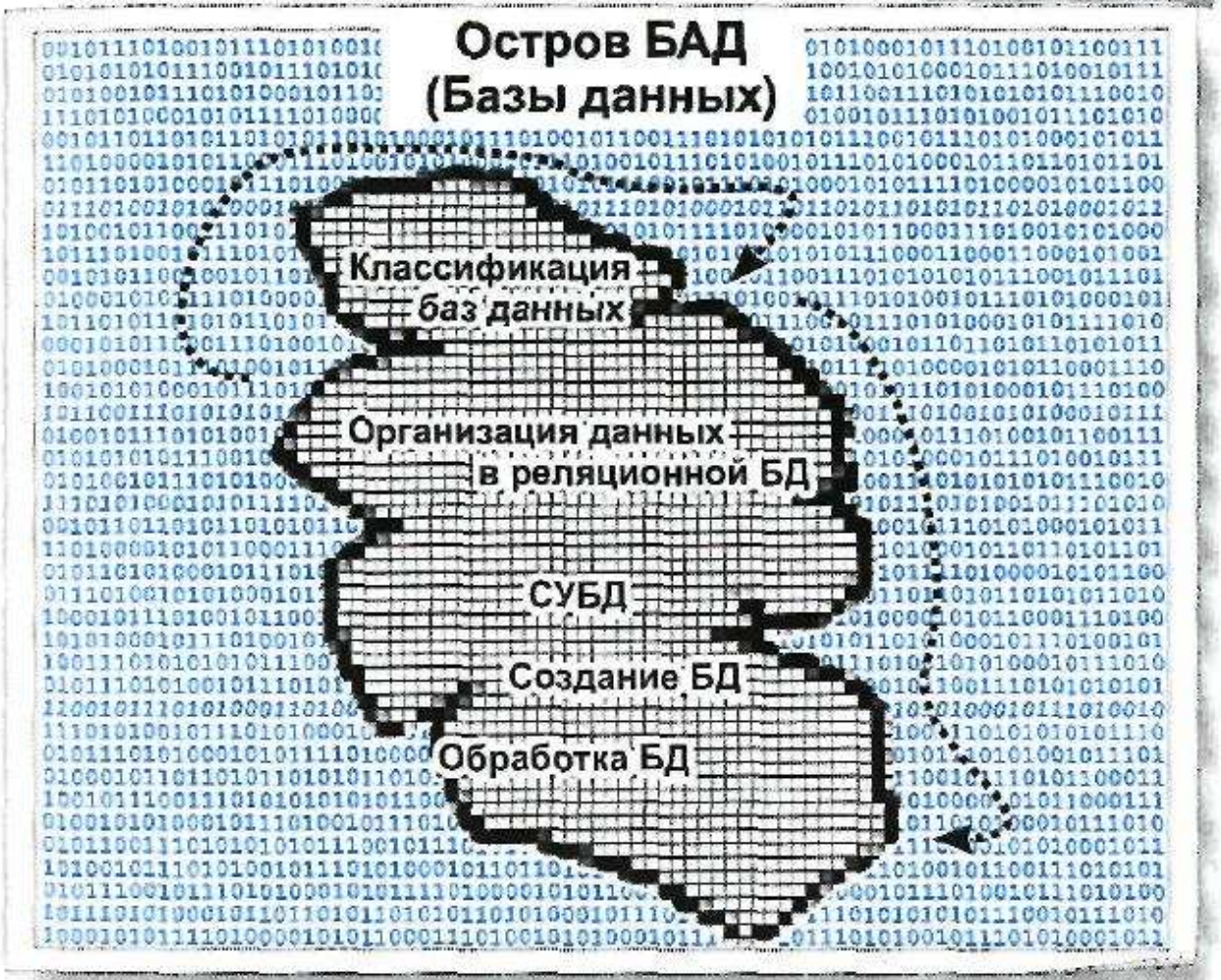
Назначение модели:
- ограниченная замена реального объекта;
- использование модели для прогнозирования поведения реального объекта

Система основных понятий главы 2

МОДЕЛИРОВАНИЕ

Глава 3

Хранение и обработка информации в базах данных



Здесь вы узнаете

- что такое база данных
- как расшифровать СУБД
- как создается база данных
- как извлечь информацию из базы данных

§ 10 Основные понятия

Основные темы параграфа:
- *что такое база данных и информационная система;*
- *реляционные базы данных;*
- *первичный ключ БД;*
- *типы полей.*

Что такое база данных и информационная система

База данных (БД) — совокупность определенным образом организованной информации на какую-то тему (в рамках некоторой предметной области). Примеры:
- база данных книжного фонда библиотеки;
- база данных кадрового состава учреждения;
- база данных законодательных актов в области уголовного права;
- база данных современных эстрадных песен.

Конечно, вся эта информация может храниться и на бумаге (например, книжный каталог библиотеки). Но современным средством хранения и обработки баз данных является, безусловно, компьютер. В дальнейшем мы будем иметь в виду только компьютерные БД.

Базы данных бывают фактографическими и документальными.

В ***фактографических БД*** содержатся краткие сведения об описываемых объектах, представленные в строго определенном формате. Из приведенных выше примеров две первые БД, скорее всего, будут организованы как фактографические. В БД библиотеки хранятся библиографические сведения о каждой книге: год издания, автор, название и пр. Разумеется, текст книги в ней содержаться не будет. В БД отдела кадров учреждения хранятся анкетные данные о сотрудниках: фамилия, имя, отчество, год и место рождения и т. д.

Базы данных в третьем и четвертом примерах наверняка будут организованы как документальные. Первая из них будет включать в себя тексты законов; вторая — тексты и ноты песен; биографическую и творческую справочную информацию о композиторах, поэтах, исполнителях; звуковые запи-

си и видеоклипы. Следовательно, ***документальная БД*** содержит обширную информацию самого разного типа: текстовую, графическую, звуковую, мультимедийную.

Современные информационные технологии постепенно стирают границу между фактографическими и документальными БД. Существуют средства, позволяющие легко подключать любой документ (текстовый, графический, звуковой) к фактографической базе данных.

Сама по себе база данных не может обслужить запросы пользователя на поиск и обработку информации. БД — это только «информационный склад». Обслуживание пользователя осуществляет информационная система.

Информационная система — это совокупность базы данных и всего комплекса аппаратно-программных средств для ее хранения, изменения и поиска информации, для взаимодействия с пользователем.

Примерами информационных систем являются системы продажи билетов на пассажирские поезда и самолеты. WWW — это тоже пример глобальной информационной системы.

Далее в нашей книге речь будет идти только о фактографических базах данных. Дадим более строгое определение компьютерной БД, чем то, что приводилось выше.

База данных — организованная совокупность данных, предназначенная для длительного хранения во внешней памяти компьютера и постоянного применения.

Для хранения БД может использоваться как один компьютер, так и множество взаимосвязанных компьютеров.

Если различные части одной базы данных хранятся на множестве компьютеров, объединенных между собой сетью, то такая БД называется ***распределенной базой данных***.

Очевидно, информацию в Интернете, объединенную «паутиной» WWW, можно рассматривать как распределенную

базу данных. Распределенные БД создаются также и в локальных сетях.

Реляционные базы данных

Информация в базах данных может быть организована по-разному. Чаще всего используется табличный способ.

Реляционные базы данных имеют табличную форму организации.

В чем же их преимущество?

Главное достоинство таблиц — в их понятности. С табличной информацией мы имеем дело практически каждый день. Загляните, например, в свой дневник: расписание занятий там представлено в виде таблицы. Когда мы приходим на вокзал, смотрим расписание электричек. Какой вид оно имеет? Это таблица! А еще есть таблица футбольного чемпионата. И журнал учителя, куда он выставляет вам оценки, — тоже таблица.

Видите, как много примеров, и их еще можно продолжить. Мы настолько привыкли к таблицам, что обычно не требуется никому объяснять, как ими пользоваться. Ну разве что маленькому ребенку, который только учится читать.

В реляционных БД строка таблицы называется ***записью***, а столбец — ***полем***. В общем виде это выглядит так:

	поле 1	поле 2	поле 3	поле 4	поле 5
запись 1	...	...	...	...	...
запись 2	...	...	...	...	...
...					

Таблицы 2.1–2.5 будем в дальнейшем рассматривать как примеры информации, пригодной для организации реляционных баз данных.

Каждое поле таблицы имеет имя. Например, в таблице 1.2 «Погода» имена полей такие: ДЕНЬ, ОСАДКИ, ТЕМПЕРАТУРА, ДАВЛЕНИЕ, ВЛАЖНОСТЬ.

Одна запись содержит информацию об одном объекте той реальной системы, модель которой представлена в таблице.

Например, домашняя библиотека — это множество книг. Значит, отдельный объект такой системы — это книга, и одна запись в базе данных *«Домашняя библиотека»* (табл. 2.1) — это информация об одной книге из библиотеки.

Поля — это различные характеристики (иногда говорят: атрибуты) объекта. Значения полей в одной строке относятся к одному объекту.

В реляционной базе данных не должно быть совпадающих записей.

Первичный ключ БД

Разные поля отличаются именами. А чем отличаются друг от друга разные записи? Записи различаются значениями ключей.

Первичным ключом в базе данных называют поле (или совокупность полей), значение которого не повторяется у разных записей.

В БД «Домашняя библиотека» разные книги могут иметь одного автора, могут совпадать названия книг, год издания, полка. Но инвентарный номер у каждой книги свой (поле НОМЕР). Он-то и является первичным ключом для записей в этой базе данных. Первичным ключом в БД «Погода» является поле ДЕНЬ, так как его значение не повторяется в разных записях.

Не всегда удается определить одно поле в качестве ключа. Пусть, например, в базе данных, которая хранится в компьютере управления образованием области, содержатся сведения о всех средних школах районных центров (табл. 3.1).

Таблица 3.1. Школы

Город	Номер школы	Директор	Адрес	Телефон
Крюков	1	Иванов А.П.	Пушкина, 5	12-35
Шадринск	1	Строев С.С.	Лесная, 14	4-23-11
Шадринск	2	Иванов А.П.	Мира, 34	4-33-24
.....	.....	.....	.....	.....

В такой таблице у разных записей не могут совпасть только одновременно два поля ГОРОД и НОМЕР ШКОЛЫ. Эти два поля вместе образуют ***составной ключ***: ГОРОД-НОМЕР ШКОЛЫ. Составной ключ может состоять и более чем из двух полей.

Типы полей

С каждым полем связано еще одно очень важное свойство — ***тип поля***.

Тип поля определяет множество значений, которые может принимать данное поле в различных записях.

В реляционных базах данных используются четыре основных типа поля:

- числовой;
- символьный;
- дата;
- логический.

Числовой тип имеют поля, значения в которых могут быть только числами. Например, в БД «Погода» три поля числового типа: ТЕМПЕРАТУРА, ДАВЛЕНИЕ, ВЛАЖНОСТЬ.

Символьный тип имеют поля, в которых будут храниться символьные последовательности (слова, тексты, коды и т. п.). Примерами символьных полей являются поля АВТОР и НАЗВАНИЕ в БД «Домашняя библиотека»; поле ТЕЛЕФОН в БД «Школы».

Тип «дата» имеют поля, содержащие календарные даты в форме «день/месяц/год» (в некоторых случаях используется американская форма: месяц/день/год). Тип «дата» имеет поле ДЕНЬ в БД «Погода».

Логический тип имеют поля, которые могут принимать всего два значения: «да», «нет» или «истина», «ложь», или (по-английски) «true», «false». Если двоичную матрицу представить в виде реляционной БД (табл. 2.4, 2.5), то ее полям, содержащим значения «0» или «1», удобно поставить в соответствие логический тип.

Итак, значения, находящиеся в полях, — это некоторые ***величины определенных типов.***

От типа величины зависят те действия, которые можно с ней производить.

Например, с числовыми величинами можно выполнять арифметические операции, а с символьными и логическими — нельзя.

Коротко о главном

База данных — организованная совокупность данных, предназначенная для длительного хранения во внешней памяти компьютера и постоянного применения.

Фактографическая БД содержит краткие сведения об описываемых объектах, представленные в строго определенном формате.

Документальная БД содержит обширную информацию самого разного типа: текстовую, графическую, звуковую, мультимедийную.

Распределенной называется база данных, разные части которой хранятся на различных компьютерах сети.

Информационная система — это совокупность базы данных и всего комплекса аппаратно-программных средств для ее хранения, изменения и поиска информации, для взаимодействия с пользователем.

Реляционные базы данных имеют табличную организацию. Строка таблицы называется записью, столбец — полем.

Таблица имеет первичный ключ, отличающий записи друг от друга. Ключом может быть одно поле (простой ключ) или несколько полей (составной ключ).

Каждое поле таблицы имеет свое уникальное имя и тип. Тип определяет, какого рода информация хранится в поле и какие действия с ней можно производить.

В БД используются четыре основных типа полей: числовой, символьный, логический, «дата».

? *Вопросы и задания*

1. Что такое база данных?
2. В чем различие между фактографическими и документальными БД?

3. Что такое распределенная БД?
4. Что такое информационная система? Приведите примеры информационных систем.
5. Что вы знаете о реляционной БД?
6. Что такое запись, поле? Какую информацию они содержат?
7. Определите имена полей в таблицах «Домашняя библиотека» (табл. 2.1), «Погода» (табл. 2.2), «Успеваемость» (табл. 2.3), «Факультативы» (табл. 2.5).
8. Что такое первичный ключ записи? Какие бывают ключи?
9. Назовите объекты, сведения о которых содержат записи баз данных «Погода», «Успеваемость», «Факультативы». Определите ключи записей в этих БД.
10. Определите следующие понятия: имя поля, значение поля, тип поля. Какие бывают типы полей? Что обозначает каждый из типов?
11. Определите типы всех полей в таблицах «Домашняя библиотека», «Погода», «Школы».
12. Определите структуру (состав полей), ключи и типы полей для реляционных баз данных под такими названиями:
    - «Страны мира»;
    - «Мои одноклассники»;
    - «Кинофильмы»;
    - «Телефонный справочник»;
    - «Мои посещения врача».

§ 11 Что такое система управления базами данных

Основные темы параграфа:
- *назначение СУБД;*
- *команда открытия БД;*
- *команда выборки.*

Назначение СУБД

Уже много раз говорилось о том, что любую работу компьютер выполняет под управлением программ. Значит, и для работы с базами данных требуется специальное программное обеспечение. Такое программное обеспечение называется системой управления базами данных или сокращенно СУБД.

Программное обеспечение, предназначенное для работы с базами данных, называется системой управления базами данных (СУБД).

Системы, работающие с реляционными базами данных, называются ***реляционными СУБД***. С помощью реляционной СУБД можно работать как с однотабличной базой данных, так и с базой, состоящей из множества связанных между собой таблиц. Здесь мы будем рассматривать только однотабличные базы данных.

С помощью СУБД вы будете создавать таблицы и сохранять их на магнитном диске в виде файлов. Каждый файл имеет свое имя. Если вы сами создаете таблицу и сохраняете ее в файле, то сами вы и придумываете имя для файла. Если же вы хотите работать с уже готовой базой, то вы должны узнать, в файле с каким именем, на каком диске и в каком каталоге она хранится.

Команда открытия БД

Для того чтобы начать работу с подготовленной базой данных, нужно открыть файл, в котором она хранится.

Делается это с помощью команды открытия файла.

Примечание 1. В нашей книге не будет описываться работа с реальной СУБД. Их много, постоянно появляются новые. В разных СУБД различный интерфейс, язык команд. Как правило, в них используются английские термины. Далее будет описываться работа с некоторой условной (гипотетической) реляционной СУБД, «понимающей» команды на русском языке. Однако эта система обладает всеми основными свойствами реальных СУБД. При выполнении заданий в компьютерном классе вам предстоит стать «переводчиками» с языка гипотетической СУБД на язык реальной системы.

Мы будем рассматривать работу с нашей гипотетической СУБД ***в режиме командного управления***. Система выводит на экран приглашение пользователю. Пусть в качестве такого приглашения выводится точка в начале командной строки (многие СУБД именно так и работают). Сразу после точки пользователь должен ввести команду с клавиатуры.

Команда открытия файла с базой данных имеет такой формат:

. открыть <имя файла>

Например, если файл имеет имя tabl.dbf, то открывается он по команде:

. открыть tabl.dbf

Примечание 2. Здесь и в дальнейшем в описаниях форматов команд будут использоваться угловые скобки <...>. Запись в угловых скобках указывает на смысл соответствующего элемента команды. Ее можно предварять при чтении наречием «некоторый». Например запись <имя файла> нужно читать так: «некоторое имя файла».

После открытия файла таблица стала доступна для работы с ней. Что можно делать с такой таблицей? Практически все, что угодно:

- добавлять новые записи;
- удалять записи, ставшие ненужными;
- изменять содержимое полей;
- изменять структуру таблицы: удалять или добавлять поля;
- сортировать записи по какому-нибудь принципу, например в алфавитном порядке фамилий авторов;
- получать справки, т. е. ответы на запросы.

Команда выборки

Очевидно, последнее — обслуживание запросов для получения справочной информации — это основная цель, ради которой создается база данных.

В большинстве случаев справка — это тоже таблица с интересующими пользователя сведениями, выбранными из базы данных. Она также состоит из строк и столбцов и может рассматриваться как результат «вырезания» и «склеивания» фрагментов исходной таблицы. Имитировать такую работу СУБД можно с помощью бумажного листа с расчерченной и заполненной таблицей, ножниц и клея.

Команда ***выборки*** информации из базы данных с целью получения справки имеет следующий формат:

.выбрать <список выводимых полей> **где** <условие выбора>

Примечание 3. Слова, входящие в формат команды (выделены жирным шрифтом), называют служебными словами.

Результат выполнения команды выводится на экран в виде таблицы. Если нужно получить на экране все строки и столбцы, то команда выглядит следующим образом:

. выбрать все

Слово «все» обозначает вывод всех полей таблицы; если условие выбора не указано, значит, выводятся все записи.

Обычно СУБД позволяют просмотреть всю базу данных, не прибегая к команде выборки. Для этого существует режим просмотра. Как правило, все записи базы не помещаются в одном кадре экрана, в таком случае используется прокрутка, т. е. последовательное перемещение строк таблицы по экрану.

Если требуется просмотреть лишь некоторые поля записей, то следует воспользоваться командой выборки. Например, из БД «Домашняя библиотека» нужно получить список всех книг, содержащий только фамилии авторов и названия. Для этого следует отдать команду

. выбрать АВТОР, НАЗВАНИЕ

Исполнение этой команды СУБД производит так: из табл. 2.1 вырезаются по очереди два столбца, соответствующие полям АВТОР и НАЗВАНИЕ. Затем они «склеиваются» в таком же порядке и в итоге получается табл. 3.2.

Таблица 3.2. Результат выборки двух полей из БД

АВТОР	НАЗВАНИЕ
Беляев А.Р.	Человек-амфибия
Кервуд Д.	Бродяги севера
Тургенев И.С.	Повести и рассказы
Олеша Ю.К.	Избранное
Беляев А.Р.	Звезда КЭЦ
Тынянов Ю.Н.	Кюхля
Толстой Л.Н.	Повести и рассказы
Беляев А.Р.	Избранное

Коротко о главном

Система управления базами данных (СУБД) — это программное обеспечение компьютера для работы с базами данных.

Таблицы БД хранятся в файлах.

Работа с базой данных начинается с открытия файлов.

Справка — это таблица, содержащая интересующие пользователя сведения, извлеченные из базы данных.

В команде получения запроса на выборку указываются выводимые поля и условие выбора (условие, которому должны удовлетворять выбираемые записи).

? *Вопросы и задания*

1. Как расшифровывается «СУБД»? Каково назначение этого вида программного обеспечения?
2. Какие СУБД называются реляционными?
3. На каком устройстве и в какой форме хранятся таблицы, созданные с помощью реляционной СУБД?
4. По какой команде (для рассмотренной здесь гипотетической СУБД) происходит получение справочной информации?
5. Как вывести на экран всю таблицу?
6. Как вывести на экран определенные столбцы таблицы? Как реализуется такая работа в терминах «вырезать», «склеить»?

§ 12 Создание и заполнение баз данных

Основные темы параграфа:

- *типы и форматы полей;*
- *создание новой БД;*
- *заполнение базы данных информацией.*

Создание базы данных связано с описанием структуры будущих таблиц. Этот этап работы выполняется в среде СУБД. *Пользователь должен указать имена всех полей таблицы, их типы и форматы.*

Типы и форматы полей

Типы полей. *Выше уже говорилось о четырех типах* полей: символьном, числовом, логическом и «дата». В некоторых СУБД используются и другие типы полей, например «Время», «День недели», «Адрес» и пр. Кроме того, многие СУБД позволяют создавать поля типа «Примечание». Дело в том, что размер символьного поля обычно ограничен величиной 255 символов. Текст большего размера в него уже не поместится. *Примечание позволяет* хранить практически неограниченный текст. Он будет храниться в отдельном файле и при необходимости может быть извлечен для чтения.

Форматы полей. *Формат символьного поля определяет* число символьных позиций, которое будет занимать поле в записи. Например, если символьное поле имеет формат 10, то его значения в различных записях могут содержать от 0 до 10 символов.

Формат числового поля обычно состоит из двух частей: длины и точности. Длина — это полное количество символьных позиций, выделяемых под запись числа: точность — это количество позиций, выделенных под дробную часть. *Следует иметь в виду, что десятичная точка тоже занимает позицию.* Например, формат записи числа 123.45 такой: длина — 6, точность — 2. Целое число, т. е. число без дробной части, имеет точность 0.

Формат логической величины стандартный — 1 символ. Чаще всего используются однобуквенные обозначения: T — true (истина), F — false (ложь). В учебнике для этих величин используются обозначения русскими буквами: И — истина, Л — ложь.

Формат даты обычно имеет длину 8 символов. Правда, бывают разные стандарты. Мы будем здесь использовать стандарт ДД/ММ/ГГ (или ДД.ММ.ГГ, или ДД-ММ-ГГ). Здесь ДД — двузначное обозначение числа, ММ — месяца, ГГ — года. Иногда используется стандарт ММ/ДД/ГГ. Бывают и другие обозначения.

Для примера в табл. 3.3 описаны типы и форматы полей из базы данных «Погода».

Таблица 3.3. Структура таблицы «Погода»

Поле	Тип	Длина	Точность
ДЕНЬ	Дата	8	
ОСАДКИ	Символьный	11	
ТЕМПЕРАТУРА	Числовой	5	1
ДАВЛЕНИЕ	Числовой	3	0
ВЛАЖНОСТЬ	Числовой	3	0

Создание новой БД

Создание новой базы данных начинается с описания структуры таблицы. По команде

. **создать** <имя файла>

пользователю предлагается заполнить таблицу типа таблицы 3.3. Затем необходимо указать первичный ключ таблицы. В данном примере первичным ключом является поле ДЕНЬ. Имя файла, в котором будет храниться база данных, пользователь задает сам.

Чтобы осмыслить этот этап работы, можно предложить следующую аналогию. Представьте себе, что строится овощная база. В ней монтируются отсеки, холодильники, контейнеры, ящики для хранения картофеля, моркови, лука, капусты и пр. Иначе говоря, готовится место для хранения, но овощи пока не завозятся. После того как овощная база создана, она готова к приему овощей.

В результате создания базы данных появляется файл с указанным именем, определяется структура данных, которые будут в ней храниться. Но база пустая, информации в ней пока нет.

Заполнение базы данных информацией

Теперь настало время заполнить базу данными (по аналогии — завезти овощи). Ввод данных производится по команде

. **добавить запись**

Ввод может происходить через форму, учитывающую структуру записей таблицы, которая была описана на этапе

создания. Например, ввод первой записи через форму в таблицу «Погода» будет происходить в таком виде:

ДЕНЬ	15/03/04
ОСАДКИ	Снег
ТЕМПЕРАТУРА	−3,5
ДАВЛЕНИЕ	746
ВЛАЖНОСТЬ	67

Добавление записей (ввод) повторяется до тех пор, пока не будет введена последняя запись. После сохранения файла создание базы данных завершено, и теперь к ней можно обращаться с запросами.

Любая СУБД дает возможность пользователю вносить изменения в уже готовую базу данных: изменять значения полей, изменять форматы полей, удалять одни поля и добавлять другие. О том, как это делается в СУБД вашего компьютерного класса, вы *узнаете на уроке.*

Коротко о главном

Этапы создания и заполнения БД происходят в среде СУБД.

На этапе создания БД создаются (открываются) файлы для хранения таблиц, сообщается информация о составе полей записей, их типах и форматах.

Основные типы полей, используемые в реляционных СУБД: числовой, символьный, логический, «дата».

Формат определяет количество позиций, отводимых в таблице для полей. Для числовых полей, кроме того, указывается количество знаков в дробной части (точность).

По команде «создать» открывается файл, определяется структура записей БД.

Ввод данных в БД начинается по команде «добавить запись».

? *Вопросы и задания*

1. Какая задача решается на этапе создания ВД? Какую информацию пользователь указывает СУБД на этапе создания?
2. Какие основные типы полей используются в базах данных?

3. Что определяется форматом для разных типов полей?
4. Составьте таблицы описания типов и форматов для всех полей баз данных «Домашняя библиотека», «Успеваемость», «Факультативы», «Школы».
5. Как происходит заполнение таблицы? Какие ошибки пользователя возможны на этом этапе?

§ 13 Условия выбора и простые логические выражения

Основные темы параграфа:

- *понятие логического выражения;*
- *операции отношения;*
- *запрос на выборку и простые логические выражения.*

Понятие логического выражения

Чаще всего для справки требуются не все записи, а только часть из них, удовлетворяющая какому-то условию. Это условие называется ***условием выбора***. Например, из таблицы «Погода» требуется узнать, в какие дни шел дождь; или из таблицы «Факультативы» — определить, кто занимается одновременно цветоводством и танцами; или из таблицы «Успеваемость» — получить список всех отличников по алгебре и физике.

В командах СУБД условие выбора записывается в форме ***логического выражения***.

Логическое выражение, подобно математическому выражению, выполняется (вычисляется), но в результате получается не число, а логическое значение: истина (true) или ложь (false). Логическая величина — это всегда ответ на вопрос, истинно ли данное высказывание.

Приведем логические значения некоторых высказываний, относящихся к трем рассмотренным выше БД (табл. 3.4).

Таблица 3.4. Высказывания и их логические значения

Высказывания	Номер записи	Значение
БД «Погода»		
1. Идет дождь.	1	Ложь
2. Давление больше 740 мм рт. ст.	2	Истина
3. Влажность не 100%.	3	Ложь
БД «Домашняя библиотека»		
4. Книга стоит ниже пятой полки.	3	Истина
5. Автор книги — Толстой Л.Н.	3	Ложь
БД «Факультативы»		
6. Фамилия ученика — Русанов.	1	Истина
7. Ученик занимается цветоводством.	1	Ложь
8. Ученик занимается танцами.	1	Истина

Вот как выглядят логические выражения, соответствующие *восьми высказываниям*, приведенным в табл. 3.4:

1. ОСАДКИ = "дождь"
2. ДАВЛЕНИЕ > 740
3. ВЛАЖНОСТЬ <> 100
4. ПОЛКА < 5
5. АВТОР = "Толстой Л.Н."
6. ФАМИЛИЯ = "Русанов"
7. ЦВЕТОВОДСТВО
8. ТАНЦЫ

Операции отношения

Шесть первых выражений называются ***отношениями***. В каждом из них имя поля базы данных связано с соответствующими значениями ***знаками отношений***. Вот все возможные знаки отношений:

=	равно;	<	меньше;
<>	не равно;	>=	больше или равно;
>	больше;	<=	меньше или равно.

Как выполняются отношения для числовых величин, вам должно быть понятно из математики. (В математике отношения называются неравенствами.) Для символьных величин требуется пояснение.

Отношение «равно» истинно для двух символьных величин, если их длины одинаковы и все соответствующие символы совпадают. Следует учитывать, что пробел — это тоже символ. Например, отношение

АВТОР="Беляев А.Р."

не будет истинным ни для одной записи нашей таблицы, поскольку в таблице везде между фамилией и инициалами стоит один пробел, а в данном отношении — два.

Символьные величины можно сопоставлять и в отношениях <, >, <=, >=. Здесь упорядоченность слов (последовательностей символов) определяется по алфавитному принципу. Вот фрагмент из орфографического словаря, содержащий последовательно расположенные в нем слова:

квартет, компонент, конверт, конвульсия.

Между этими словами истинны следующие отношения:

квартет < компонент;

компонент < конверт;

конверт < конвульсия.

Значения полей типа «дата» при выполнении отношений сравниваются в соответствии с календарной последовательностью. Например, истинны отношения:

3/12/1998 < 23/04/2001;

24/09/2004 > 23/09/2004.

В некоторых СУБД используется тип «время» со следующим форматом значений: ЧЧ:ММ:СС (часы, минуты, секунды). При выполнении отношений учитывается хронологическая последовательность. Например, истинны отношения:

12:53:08 > 03:40:00;

23:05:12 < 23:05:13.

А теперь вернемся к приведенным выше примерам логических выражений. В примерах 7 и 8 нет никаких знаков отношений. Дело в том, что поля с именами ЦВЕТОВОДСТВО и ТАНЦЫ имеют логический тип. Поэтому в каждой записи их значения — это логические величины «ложь», «истина».

Одна величина логического типа — простейшая форма логического выражения. Следовательно, условие выбора может содержать в своей записи в том числе имя логического поля.

Запрос на выборку и простые логические выражения

Запишем несколько команд для получения справки, используя условия выбора. Вот как выглядит команда запроса информации из БД «Погода» о датах всех дождливых дней:

. выбрать ДЕНЬ где ОСАДКИ = "дождь"

В итоговую справку попадут лишь те записи, для которых истинно условие поиска. Значит, получим:

ДЕНЬ
18/03/04

Следующая команда позволяет вывести даты и влажность, соответствующие тем дням, когда атмосферное давление было выше 745 мм рт. ст.:

.выбрать ДЕНЬ, ВЛАЖНОСТЬ где ДАВЛЕНИЕ >745.

ДЕНЬ	ВЛАЖНОСТЬ
15/03/04	67
16/03/04	62
19/03/04	87

Запишем команду запроса справки к БД «Домашняя библиотека»: вывести названия книг и фамилии и инициалы авторов, фамилии которых начинаются с буквы «О» и далее по алфавиту:

.выбрать АВТОР, НАЗВАНИЕ где АВТОР >= «О»

АВТОР	НАЗВАНИЕ
Тургенев И.С.	Повести и рассказы
Олеша Ю.К.	Избранное
Тынянов Ю.Н.	Кюхля
Толстой Л.Н.	Повести и рассказы

А теперь запрос к БД «Факультативы»: вывести список фамилий всех учеников, посещающих танцы:

.выбрать ФАМИЛИЯ где ТАНЦЫ.

ФАМИЛИЯ
Русанов
Зотова
Шляпина

Выражение, состоящее из имени поля логического типа или одного отношения, будем называть простым логическим выражением.

Многие СУБД позволяют в отношениях использовать арифметические выражения. Арифметические выражения могут включать в себя числа, имена полей числового типа, знаки арифметических операций, круглые скобки*.

Рассмотрим базу данных, содержащую таблицу успеваемости учеников (табл. 2.3).

Требуется получить список учеников, у которых сумма баллов по гуманитарным предметам больше, чем по естественным. Следует отдать команду:

.выбрать УЧЕНИК где РУССКИЙ + ИСТОРИЯ +
МУЗЫКА > АЛГЕБРА + ХИМИЯ + ФИЗИКА

В результате получим:

Ботов Иван;
Галкина Нина.

---

* В некоторых СУБД такая возможность реализуется через специально организуемые вычисляемые поля.

Следующая команда запрашивает фамилии учеников, у которых оценка по алгебре выше их среднего балла:

```
.выбрать УЧЕНИК где АЛГЕБРА > (РУССКИЙ +
        АЛГЕБРА + ХИМИЯ + ФИЗИКА + ИСТОРИЯ +
        МУЗЫКА)/6
```

Ответ: Аликин Петр;
Галкина Нина.

Коротко о главном

Логическое выражение вычисляется подобно математическому, но может принимать всего два значения: истина (true) или ложь (false).

Простейшая форма логического выражения — одна величина логического типа.

Отношение — форма логического выражения.

Существует шесть вариантов отношений: «равно», «не равно», «больше», «меньше», «больше или равно», «меньше или равно». Отношения применимы ко всем типам полей.

Условия выбора в командах СУБД записываются в виде логических выражений.

Вопросы и задания

1. Какую роль выполняет условие выбора? После какого служебного слова записывается это условие в команде **.выбрать**?
2. Что такое логическое выражение? Какие значения оно принимает?
3. Какое логическое выражение называется простым?
4. Какие виды отношений используются в логических выражениях? Как записываются знаки отношений?
5. Как сравниваются символьные величины, даты, логические величины?
6. В следующих простых логических выражениях поставьте вместо знаков вопроса такие знаки отношений, при которых эти выражения будут истинны в указанных записях баз данных.

   а) БД «Погода», запись номер 3.
   ВЛАЖНОСТЬ ? 90

ОСАДКИ ? "дождь"

б) БД «Домашняя библиотека», запись номер 1.
АВТОР ? "Толстой Л.Н."
ГОД ? 1990

в) БД «Успеваемость», запись номер 4.
ФИЗИКА ? 2

7. Данные высказывания запишите в форме простых логических выражений и определите результат их вычисления для указанных записей.

а) БД «Погода», запись номер 2.
Температура выше нуля.
Осадков нет.

б) БД «Домашняя библиотека», запись номер 3.
Книга издана в 1982 году.
Книга находится ниже пятой полки.

в) БД «Факультативы», запись номер 4.
Ученик занимается геологией.
Фамилия ученицы — Шляпина.

8. Запишите следующие высказывания в форме логических выражений:
а) фамилия ученика — не Семенов;
б) ученик занимается геологией;
в) дата — раньше 5 мая 1989 года;
г) дата — не позже 23 сентября 1996 года;
д) по алгебре — не отлично;
е) автор книги — Беляев А.Р.;
ж) книга издана до 1990 года;
з) книга находится не ниже третьей полки.

9. Запишите в форме команды **.выбрать** запросы, использующие в качестве условий простые логические выражения, полученные в результате выполнения предыдущего задания.

§ 14 Условия выбора и сложные логические выражения

Основные темы параграфа:
- *примеры сложных логических выражений;*
- *логическое умножение (и);*
- *логическое сложение (или);*
- *отрицание;*
- *приоритеты логических операций;*
- *запрос на выборку и сложные логические выражения.*

Примеры сложных логических выражений

Рассмотрим еще одну группу высказываний (табл. 3.5) и их логические значения.

Таблица 3.5. Высказывания и их логические значения

Высказывание	Значение
БД «Факультативы»	
1. Русанов занимается геологией.	Истина
2. Шляпина посещает факультативы.	Истина
БД «Успеваемость»	
3. У Аликина по физике то ли 4, то ли 5.	Истина
4. У Галкиной по алгебре не двойка.	Истина
БД «Погода»	
5. 15 марта 2004 года были осадки.	Истина
6. 17 марта 2004 года влажность была меньше 100%.	Ложь
БД «Домашняя библиотека»	
7. В библиотеке есть книги Беляева А.Р., изданные не ранее 1990 года.	Истина
8. В библиотеке есть книги Толстого Л.Н. или Тургенева И.С.	Истина

Каждое из этих высказываний объединяет в себе значения нескольких полей одновременно. Поэтому они не могут быть записаны в форме простых логических выражений.

Вот как записываются соответствующие логические выражения:

1. ФАМИЛИЯ="Русанов" и ГЕОЛОГИЯ
2. ФАМИЛИЯ="Шляпина" и (ЦВЕТОВОДСТВО или ГЕОЛОГИЯ или ТАНЦЫ)
3. УЧЕНИК="Аликин Петр" и (ФИЗИКА=4 или ФИЗИКА=5)
4. не АЛГЕБРА=2 и УЧЕНИК="Галкина Нина"
5. ДАТА=15/03/04 и (ОСАДКИ="дождь" или ОСАДКИ="снег")
6. ДАТА=17/03/04 и ВЛАЖНОСТЬ<100
7. АВТОР="Беляев А.Р." и ГОД>=1990
8. АВТОР="Толстой Л.Н." или АВТОР="Тургенев И.С."

Здесь кроме знакомых вам отношений и имен логических полей используются смысловые связки **«и»**, **«или»**, **«не»**. Это служебные слова, которые выполняют роль ***знаков логических операций***.

Познакомимся с тремя логическими операциями:

- операция логического умножения (конъюнкция); знак операции **«и»**;
- операция логического сложения (дизъюнкция); знак операции **«или»**;
- операция отрицания; знак операции **«не»**.

Выражения, содержащие логические операции, будем называть сложными логическими выражениями.

Операции логического умножения и сложения — двуместные. Это значит, что они связывают между собой две логические величины (два логических операнда).

Логическое умножение (и)

В результате логического умножения (конъюнкции) получается истина, если оба операнда истинны.

Пусть требуется получить справку о книгах Беляева А.Р., изданных не раньше 1990 года, с указанием названия книги, года издания и полки, на которой стоит книга. Соответствующая команда имеет вид:

.выбрать НАЗВАНИЕ, ГОД, ПОЛКА **где** АВТОР= "Беляев А.Р." **и** ГОД>=1990

Формирование справки происходит в такой последовательности: сначала вырезаются и склеиваются в одну таблицу все строки, удовлетворяющие первому отношению: АВТОР=Беляев А.Р. Получается следующее:

НОМЕР	АВТОР	НАЗВАНИЕ	ГОД	ПОЛКА
0001	Беляев А.Р.	Человек-амфибия	1987	5
0005	Беляев А.Р.	Звезда КЭЦ	1990	5
0008	Беляев А.Р.	Избранное	1994	7

Затем из этой таблицы вырезаются строки, удовлетворяющие второму отношению: ГОД>=1990. Получаем:

НОМЕР	АВТОР	НАЗВАНИЕ	ГОД	ПОЛКА
0005	Беляев А.Р.	Звезда КЭЦ	1990	5
0008	Беляев А.Р.	Избранное	1994	7

И наконец, вырезаются столбцы, указанные в списке полей команды. На экран выведется справка:

НАЗВАНИЕ	ГОД	ПОЛКА
Звезда КЭЦ	1990	5
Избранное	1994	7

Логическое сложение (или)

В результате логического сложения (дизъюнкции) получается истина, если значение хотя бы одного операнда истинно.

Пусть, например, мы хотим получить список всех книг Толстого Л.Н. и Тургенева И.С. Запрос на выборку пишется так:

.выбрать где АВТОР="Толстой Л.Н." или
АВТОР="Тургенев И.С."

В *этом случае строки, удовлетворяющие условиям* АВТОР="Толстой Л. Н." или АВТОР="Тургенев И. С.", *вырезаются одновременно* из исходной таблицы. После их склеивания получаем:

НОМЕР	АВТОР	НАЗВАНИЕ	ГОД	ПОЛКА
0003	Тургенев И.С.	Повести и рассказы	1982	1
0007	Толстой Л.Н.	Повести и рассказы	1986	1

Отрицание

Отрицание изменяет значение логической величины на противоположное: не *истина* = *ложь*; не *ложь* = *истина*.

Отрицание — одноместная операция. Это значит, что она применяется к одному логическому операнду.

Например, требуется получить список всех книг, кроме книг Беляева. Запрос такой:

.выбрать АВТОР, НАЗВАНИЕ **где не** АВТОР="Беляев А.Р."

В этом случае вырезаются все строки, в которых значение поля АВТОР не равно «Беляев А.Р.». Строки склеиваются, а из полученной таблицы вырезаются столбцы АВТОР и НАЗВАНИЕ. После их склеивания получим справку:

АВТОР	НАЗВАНИЕ
Кервуд Д.	Бродяги севера
Тургенев И.С.	Повести и рассказы
Олеша Ю.К.	Избранное
Тынянов Ю.Н.	Кюхля
Толстой Л.Н.	Повести и рассказы

Табл. 3.6 иллюстрирует результаты всех вариантов выполнения трех логических операций. Такую таблицу называют ***таблицей истинности***. В ней буквами *А* и *Б* обозначены логические операнды. Логическая величина «истина» обозначена буквой И, логическая величина «ложь» — буквой Л.

Таблица 3.6. Таблица истинности

А	*Б*	*А* и *Б*	*А* или *Б*	не *А*
И	И	И	И	Л
И	Л	Л	И	Л
Л	И	Л	И	И
Л	Л	Л	Л	И

Приоритеты логических операций

Если в сложном логическом выражении имеется несколько логических операций, то возникает вопрос, в каком порядке их выполнит компьютер. Это касается выражений под номерами 2, 3, 4, 5 в приведенном выше примере.

В логическом выражении можно использовать круглые скобки. Так же как и в математических формулах, скобки влияют на последовательность выполнения операций. Если нет скобок, то операции выполняются в порядке их старшинства. Среди логических операций, как и среди арифметических, есть различие по старшинству (еще говорят: по приоритету). По убыванию старшинства логические операции располагаются в таком порядке:

отрицание (не);
конъюнкция (и);
дизъюнкция (или).

Запрос на выборку и сложные логические выражения

Ниже приведены два логических выражения для БД «Домашняя библиотека», отличающиеся только скобками. Цифры над знаками логических операций указывают на последовательность их выполнения.

```
      1          3          2
(ГОД=1987 или ГОД=1986) и (ПОЛКА=5 или ПОЛКА=1)

      2          1          3
ГОД=1987 или ГОД=1986 и ПОЛКА=5 или ПОЛКА=1
```

Первое выражение будет истинным для записей с номерами 1, 4, 7. Второе истинно для записей с номерами 1, 3, 4, 6, 7. Постарайтесь понять, почему.

Коротко о главном

Выражение, содержащее логические операции, называется сложным логическим выражением.

Существуют три основные логические операции: отрицание (**не**); логическое умножение — конъюнкция (**и**); логическое сложение — дизъюнкция (**или**).

Отрицание изменяет значение логической величины на противоположное; результат логического умножения — истина только тогда, когда истинны оба операнда; результат логического сложения — истина, если значение хотя бы одного операнда истинно.

Порядок выполнения операций в логическом выражении определяется старшинством операций и расстановкой скобок. В первую очередь выполняются операции в скобках. Логические операции по убыванию старшинства расположены так: **не, и, или**.

Вопросы и задания

1. Какие логические операции употребляются в сложных логических выражениях? Сформулируйте правила их выполнения.
2. Как определяется порядок выполнения логических операций в сложных выражениях?

3. Пусть *a*, *b*, *c* — логические величины, которые имеют следующие значения: *a* = *истина*, *b* = *ложь*, *c* = *истина*. Определите результаты вычисления следующих логических выражений:

a **и** *b*;	*a* **и** *b* **или** *c*;	(*a* **или** *b*) **и** (*c* **или** *b*);
a **или** *b*;	*a* **или** *b* **и** *c*;	**не** (*a* **или** *b*) **и** (*c* **или** *b*);
не *a* **или** *b*;	**не** *a* **или** *b* **и** *c*;	**не** (*a* **и** *b* **и** *c*).

4. Напишите команды выборки с использованием сложных логических выражений:

- определить все даты до 17 марта, когда температура была выше нуля градусов;
- определить фамилии всех учеников, которые посещают танцы, но не посещают факультатив по геологии;
- получить список всех отличников по гуманитарным дисциплинам;
- определить полку, на которой стоит книга Толстого Л.Н. «Повести и рассказы»;
- определить фамилии авторов книг с названием «Повести и рассказы», выпущенных до 1985 года;
- получить инвентарные номера всех книг, стоящих на пятой и седьмой полках;
- получить фамилии авторов и названия книг, выпущенных в период с 1985 по 1990 год;
- получить инвентарные номера всех книг, стоящих ниже пятой полки и изданных после 1990 года.

§ 15 Сортировка, удаление и добавление записей

Основные темы параграфа:

- *команда выборки с параметром сортировки;*
- *ключи сортировки;*
- *сортировка по нескольким ключам;*
- *команды удаления и добавления записей.*

Команда выборки с параметром сортировки

Очень часто записи в таблицах бывают упорядочены по какому-то правилу. Например, в телефонных справочниках — в алфавитном порядке фамилий абонентов; в расписании движения поездов — в порядке времени отправления; в таблице футбольного чемпионата — по возрастанию номеров мест, которые занимают команды.

Процесс упорядочения записей в таблице называется ***сортировкой***. Для выполнения сортировки должна быть указана следующая информация:

а) по значению какого поля производить сортировку;

б) в каком порядке сортировать записи (по возрастанию или убыванию значений поля).

В команду выборки можно добавить параметры сортировки, в соответствии с которыми будут упорядочены строки в итоговой таблице. В таком случае формат команды выборки становится следующим:

.выбрать <список выводимых полей>
где <условия выбора>
сортировать <ключи сортировки>
по <порядок сортировки>

Ключи сортировки

Ключом сортировки называется поле, по значению которого производится сортировка. Возможны два варианта порядка сортировки: по возрастанию значений ключа и по убыванию значений ключа.

Отсортируем записи таблицы «Погода» по убыванию значений влажности. Для этого нужно отдать команду:

.выбрать все сортировать ВЛАЖНОСТЬ **по** убыванию

В результате выполнения этой команды будет получена таблица 3.7.

Таблица 3.7. Таблица «Погода», отсортированная по убыванию влажности

ДАТА	ОСАДКИ	ТЕМПЕРАТУРА	ДАВЛЕНИЕ	ВЛАЖНОСТЬ
17/03/03	Туман	1,0	740	100
18/03/03	Дождь	3,4	745	96
19/03/03	Без осадков	5,2	760	87
15/03/03	Снег	−3,5	746	67
16/03/03	Без осадков	0	750	62

А теперь отсортируем записи БД «Домашняя библиотека» в алфавитном порядке по фамилиям авторов. В итоговую таблицу выберем только сведения о книгах, изданных после 1985 года. Выведем три поля: АВТОР, НАЗВАНИЕ, ГОД. Для этого нужно выполнить команду:

. **выбрать** АВТОР, НАЗВАНИЕ, ГОД **где** ГОД>1985
сортировать АВТОР ***по возрастанию***

В итоге получим таблицу 3.8.

Таблица 3.8. Таблица «Домашняя библиотека», отсортированная в алфавитном порядке фамилий авторов

АВТОР	НАЗВАНИЕ	ГОД
Беляев А.Р.	Человек-амфибия	1987
Беляев А.Р.	Звезда КЭЦ	1990
Беляев А.Р.	Избранное	1994
Кервуд Д.	Бродяги севера	1991
Олеша Ю.К.	Избранное	1987
Толстой Л.Н.	Повести и рассказы	1986

Сортировка по нескольким ключам

Нередко приходится встречать таблицы, в которых строки отсортированы по значениям нескольких полей. Например, если мы хотим, чтобы в полученной таблице (см. табл. 3.8) книги одного автора были упорядочены в списке

в алфавитном порядке их названий, то команду выборки нужно записать так:

. **выбрать** АВТОР, НАЗВАНИЕ, ГОД **где** ГОД>1985
сортировать АВТОР **по возрастанию**, НАЗВАНИЕ
по возрастанию

Здесь указаны два ключа сортировки: поле АВТОР является первым ключом сортировки, поле НАЗВАНИЕ — вторым ключом сортировки. Сначала записи сортируются по возрастанию значений первого ключа (АВТОР), затем среди записей с одинаковыми значениями первого ключа происходит сортировка по значениям второго ключа (НАЗВАНИЕ). В результате получим таблицу (показана только часть таблицы, относящаяся к книгам Беляева А. Р. Порядок остальных строк не изменится):

АВТОР	НАЗВАНИЕ	ГОД
Беляев А.Р.	Звезда КЭЦ	1990
Беляев А.Р.	Избранное	1994
Беляев А.Р.	Человек-амфибия	1987

Команды удаления и добавления записей

Информация в базах данных часто подвергается изменениям. Например, БД «Погода» каждый день должна пополняться. Состав домашней библиотеки также со временем меняется. Мы покупаем книги, иногда дарим их друзьям. Все эти изменения должны сразу же отражаться в базе данных. Следовательно, в языке общения с СУБД должны присутствовать команды, позволяющие вносить такие изменения. В нашей гипотетической СУБД есть для этих целей две команды. Первая позволяет удалять строки из таблицы. Ее формат такой:

. **удалить где** <логическое выражение>

Чтобы удалить из БД одну конкретную запись, нужно указать значение ключа этой записи. Например, если применительно к БД «Домашняя библиотека» отдать команду

. **удалить где** НОМЕР="0003"

то сведения о книге под номером 3 будут исключены из таблицы. *Если по отношению к БД «Школа» выполнить команду*

. удалить для ГОРОД="Шадринск" и НОМЕР ШКОЛЫ=1

то из таблицы будет исключена вторая запись.

Вот еще пример. После выполнения команды

. удалить где ГОД<1985

из БД «Домашняя библиотека» исчезнут записи с номерами 3, 6, т. е. книги, выпущенные до 1985 года.

Если же нужно удалить все записи из таблицы, то это делается командой

. удалить все

Примечание. Часто в реальных СУБД по команде «удалить» лишь помечаются записи, предназначенные для удаления. *Исключение их из файла происходит после выполнения процедуры сжатия файла.*

Если к готовой базе данных требуется добавить новые записи, то это всегда можно сделать с помощью уже знакомой *вам команды:*

. добавить запись

По этой команде пользователю предоставляется возможность ввести значения полей новой записи, которая занесется в конец таблицы.

Коротко о главном

Сортировка БД — это упорядочение записей в таблице по возрастанию или убыванию значений какого-нибудь поля — ключа сортировки. Сортировка может производиться по нескольким ключам одновременно.

Изменение состава записей в БД происходит путем удаления ненужных записей и добавления новых. Добавленная запись помещается в конец таблицы.

Вопросы и задания

1. Что понимается под сортировкой базы данных?
2. Что такое ключ сортировки?
3. В каком случае и каким образом производится сортировка по нескольким ключам?
4. С помощью каких команд изменяется состав записей БД?
5. Запишите команды для выполнения следующих действий с БД «Домашняя библиотека»:
   а) сортировки в порядке возрастания годов издания книги;
   б) сортировки по двум ключам: АВТОР и ГОД, исключая книги Беляева А.Р.;
   в) удаления из БД «Домашняя библиотека» всех записей о книгах, стоящих на пятой полке и изданных до 1990 года.

Чему вы должны научиться, изучив главу 3

Освоить одну из СУБД, имеющихся в компьютерном классе.

Уметь открывать и просматривать готовую базу данных.

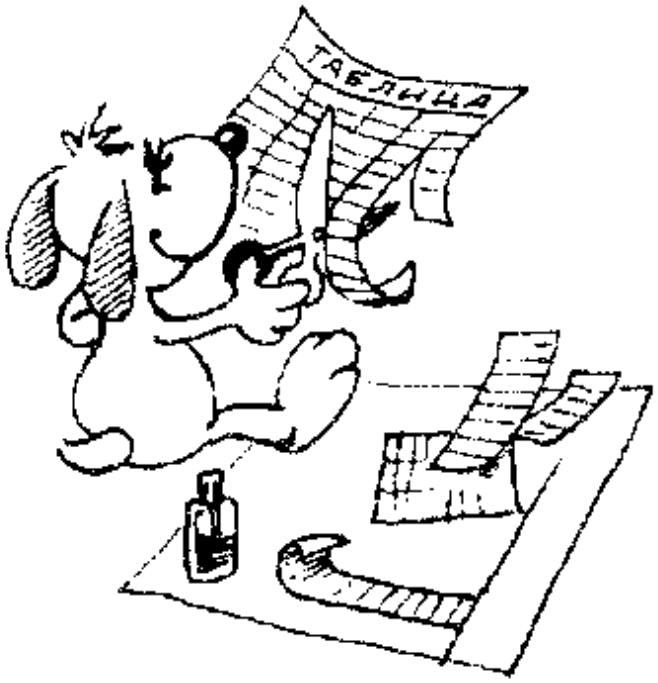
Уметь создавать однотабличную базу данных.

Записывать условия выбора в форме *логических* выражений.

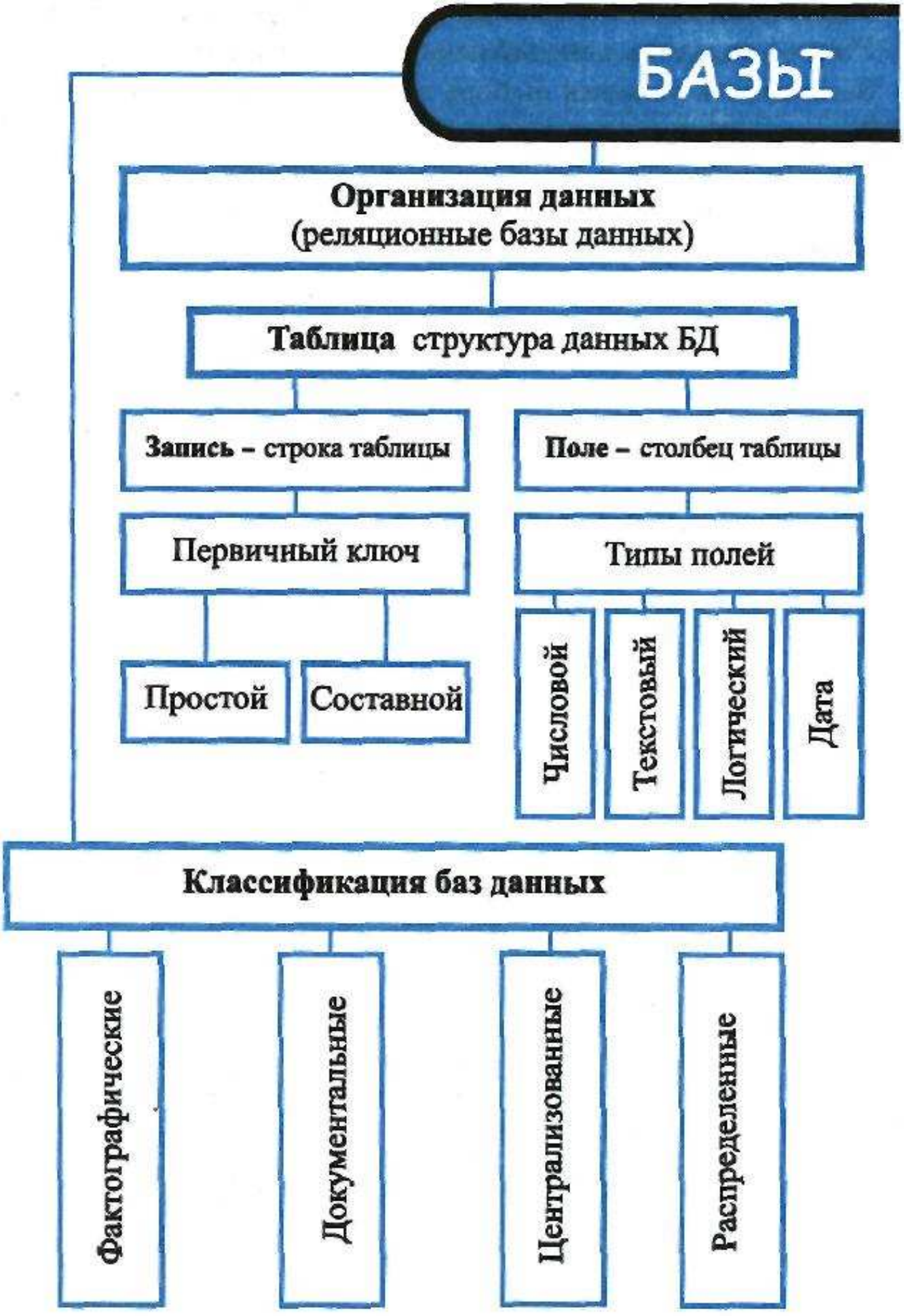
Организовывать в БД запросы на выборку.

Сортировать таблицы по заданному ключу.

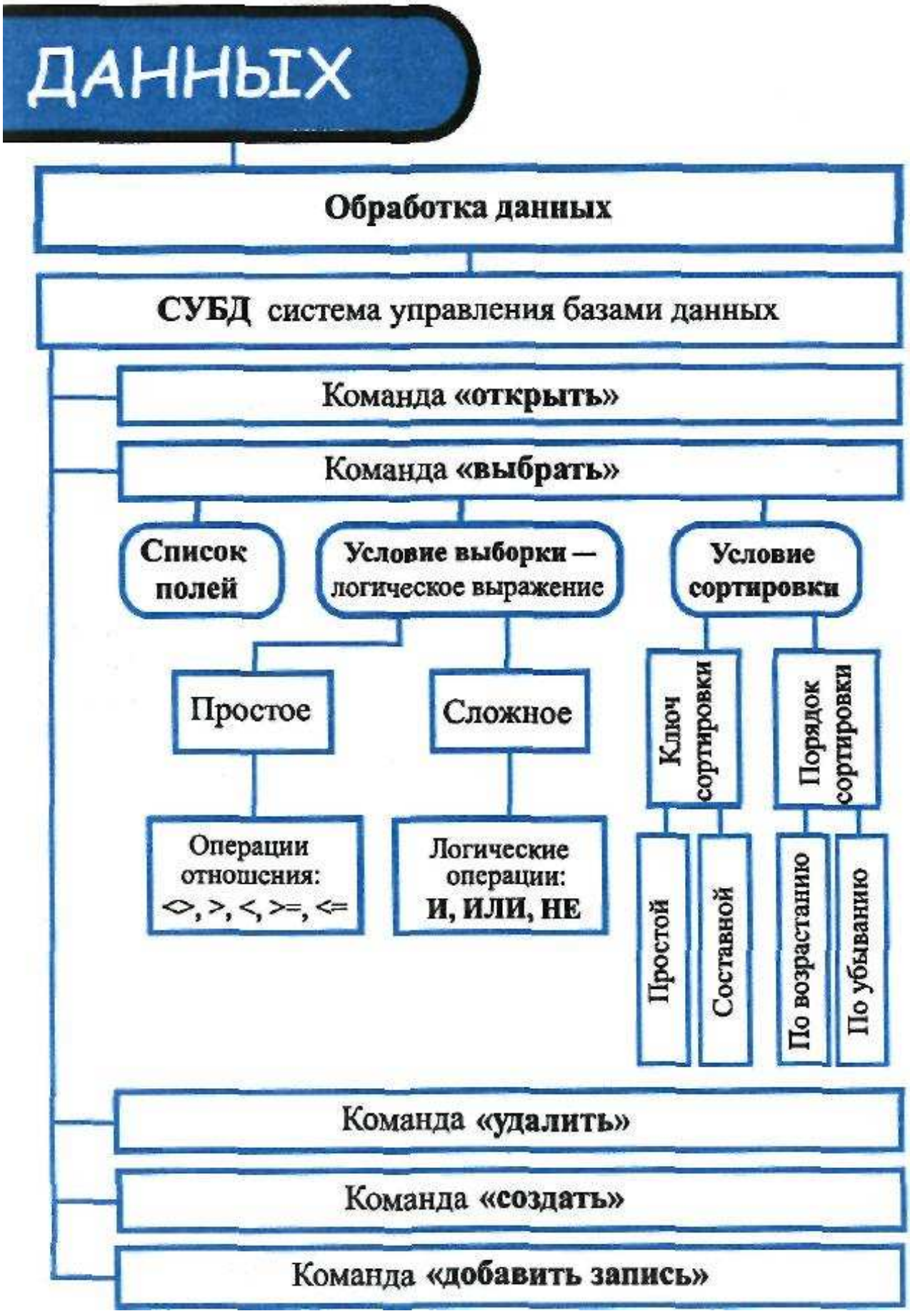
Добавлять и удалять записи в базе данных.



БАЗЫ

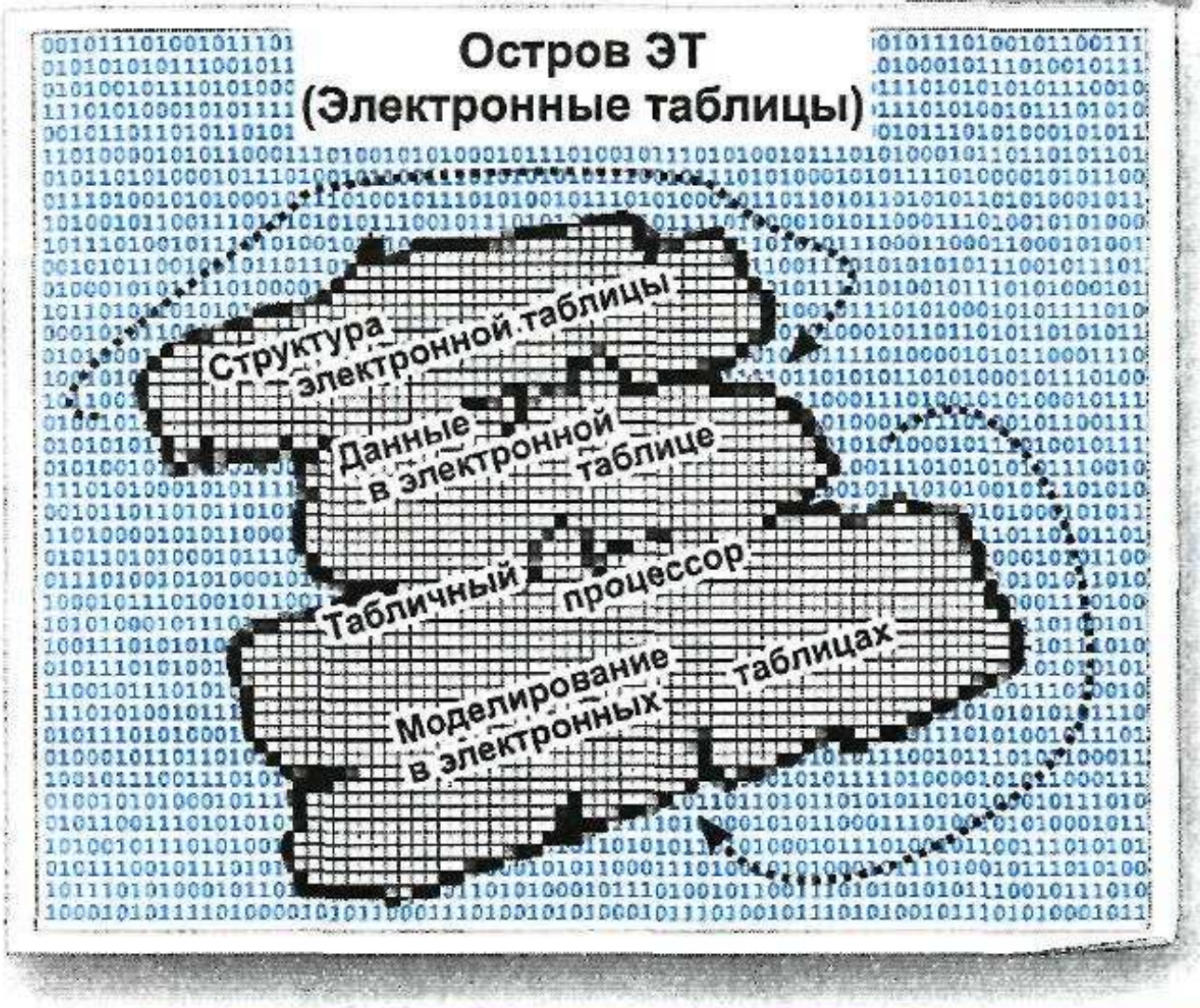

Организация данных
(реляционные базы данных)
Таблица структура данных БД
Запись – строка таблицы
Поле – столбец таблицы
Первичный ключ
Типы полей
Простой
Составной
Числовой
Текстовый
Логический
Дата
Классификация баз данных
Фактографические
Документальные
Централизованные
Распределенные

Система основных понятий главы 3



Глава 4

Табличные вычисления на компьютере



Здесь вы узнаете

- как компьютер работает с числами
- что такое электронная таблица
- как решаются вычислительные задачи с помощью электронных таблиц
- как можно использовать электронные таблицы для информационного моделирования

§ 16 Двоичная система счисления

Основные темы параграфа:

- *десятичная и двоичная системы счисления;*
- *развернутая форма записи числа;*
- *перевод двоичных чисел в десятичную систему;*
- *перевод десятичных чисел в двоичную систему;*
- *арифметика двоичных чисел.*

В данной главе речь пойдет об организации вычислений на компьютере. Вычисления связаны с хранением и обработкой чисел.

Компьютер работает с числами в двоичной системе счисления.

Эта идея принадлежит Джону фон Нейману, сформулировавшему в 1946 году принципы устройства и работы ЭВМ. Выясним, что такое система счисления.

Десятичная и двоичная системы счисления

Системой счисления называют определенные правила записи чисел и связанные с ними способы выполнения вычислений.

С историей систем счисления вы познакомитесь в главе 7 учебника. А пока нас будут интересовать двоичная и десятичная системы счисления.

Система счисления, к которой мы все привыкли, называется десятичной. Объясняется это название тем, что в ней используются десять цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Число цифр определяет основание системы счисления. Если число цифр — десять, то основание системы счисления равно десяти. В двоичной же системе существует всего две цифры: 0 и 1. Основание равно двум. Возникает вопрос, можно ли с помощью всего двух цифр представить любую величину. Оказывается, можно!

Развернутая форма записи числа

Вспомним принцип записи чисел в десятичной системе счисления. Значение цифры в записи числа зависит не только от самой цифры, но и от места расположения этой цифры в числе (говорят: от позиции цифры). Например, в числе 333 первая справа цифра обозначает три единицы, следующая — три десятка, следующая — три сотни. Этот факт можно выразить равенством:

$$333_{10} = 3 \cdot 10^2 + 3 \cdot 10^1 + 3 \cdot 10^0 = 300 + 30 + 3.$$

В данном равенстве выражение, стоящее справа от знака «равно», называется развернутой формой записи многозначного числа. Вот еще пример развернутой формы записи многозначного десятичного числа:

$$8257_{10} = 8 \cdot 10^3 + 2 \cdot 10^2 + 5 \cdot 10^1 + 7 \cdot 10^0 =$$
$$= 8000 + 200 + 50 + 7.$$

Таким образом, с продвижением от цифры к цифре справа налево «вес» каждой цифры увеличивается в 10 раз. Это связано с тем, что основание системы счисления равно десяти.

Перевод двоичных чисел в десятичную систему

А вот пример многозначного двоичного числа:

$$110101_2.$$

Двойка внизу справа указывает на основание системы счисления. Это нужно для того, чтобы не перепутать двоичное число с десятичным. Ведь существует же десятичное число 110101! Вес каждой следующей цифры в двоичном числе при продвижении справа налево возрастает в 2 раза. Развернутая форма записи данного двоичного числа выглядит так:

$$110101_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 53_{10}.$$

Таким способом мы перевели двоичное число в десятичную систему.

Переведем в десятичную систему еще несколько двоичных чисел.

$$10_2 = 2^1 = 2; \quad 100_2 = 2^2 = 4; \quad 1000_2 = 2^3 = 8;$$
$$10000_2 = 2^4 = 16; \; 100000_2 = 2^5 = 32 \text{ и т. д.}$$

Таким образом, получилось, что двузначному десятичному числу соответствует шестизначное двоичное! И это ха-

рактерно для двоичной системы: быстрый рост количества цифр с увеличением значения числа.

Вот как выглядит начало натурального ряда чисел в десятичной (A_{10}) и двоичной (A_2) системах счисления:

A_{10}	1	2	3	4	5	6	7	8	9	10
A_2	1	10	11	100	101	110	111	1000	1001	1010

A_{10}	11	12	13	14	15	16	17	18	19	20
A_2	1011	1100	1101	1110	1111	10000	10001	10010	10011	10100

Перевод десятичных чисел в двоичную систему

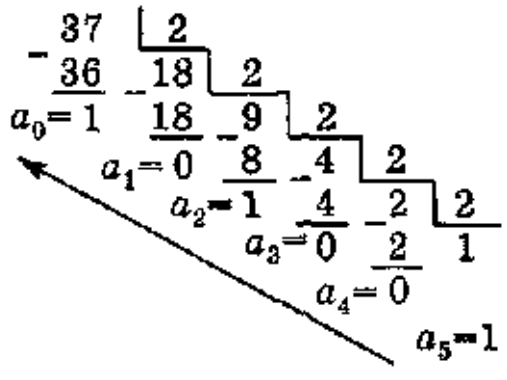
Как перевести двоичное число в равное ему десятичное, вам должно быть понятно из рассмотренных выше примеров. А как осуществить обратный перевод: из десятичной системы в двоичную? Для этого нужно суметь разложить десятичное число на слагаемые, представляющие собой степени двойки. Например:

$$15_{10} = 8 + 4 + 2 + 1 = 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1111_2.$$

Это сложно. Есть другой способ, с которым мы сейчас и познакомимся.

Существует процедура, позволяющая легко выполнить перевод десятичного числа в двоичную систему. Она состоит в том, что данное десятичное число делится на 2. Полученный остаток — это младший разряд искомого числа. Полученное частное снова делится на 2, полученный при этом остаток — это следующий разряд искомого числа. Так продолжается до тех пор, пока частное не станет меньше двойки (основания системы). Это частное — старшая цифра искомого числа.

Существуют два способа записи деления на 2. Продемонстрируем это на примере перевода числа 37 в двоичную систему.



37	2	1	a_0
18	2	0	a_1
9	2	1	a_2
4	2	0	a_3
2	2	0	a_4
1	→	1	a_5

Здесь a_5, a_4, a_3, a_2, a_1, a_0 — обозначения цифр в записи двоичного числа по порядку слева направо. В результате перевода получим:

$$37_{10} = 100101_2.$$

Арифметика двоичных чисел

Правила двоичной арифметики гораздо проще правил десятичной арифметики. Вот все возможные варианты сложения и умножения однозначных двоичных чисел:

0 + 0 = 0	0 × 0 = 0
0 + 1 = 1	0 × 1 = 0
1 + 0 = 1	1 × 0 = 0
1 + 1 = 10	1 × 1 = 1

Своей простотой и согласованностью с битовой структурой компьютерной памяти двоичная система счисления и привлекла изобретателей компьютера. Ее гораздо проще реализовать техническими средствами, чем десятичную систему.

Вот пример сложения столбиком двух многозначных двоичных чисел:

$$\begin{array}{r} 1\,0\,1\,1\,0\,1\,1\,1\,0\,1 \\ +\quad 1\,1\,1\,0\,1\,0\,1\,1\,0 \\ \hline 1\,0\,0\,1\,0\,1\,1\,0\,0\,1\,1 \end{array}$$

А теперь посмотрите внимательно на следующий пример умножения многозначных двоичных чисел:

$$\begin{array}{r} 1\,1\,0\,1\,1\,0\,1 \\ \times\qquad 1\,0\,1 \\ \hline 1\,1\,0\,1\,1\,0\,1 \\ 1\,1\,0\,1\,1\,0\,1 \\ \hline 1\,0\,0\,0\,1\,0\,0\,0\,0\,1 \end{array}$$

После небольшой тренировки любой из вас такие вычисления будет выполнять автоматически.

Коротко о главном

Система счисления — определенные правила записи чисел и связанные с этими правилами способы выполнения вычислений.

Основание системы счисления равно количеству используемых в ней цифр.

Двоичные числа — числа в двоичной системе счисления. В их записи используются две цифры: 0 и 1.

Развернутая форма записи двоичного числа — это его представление в виде суммы степеней двойки, умноженных на 0 или на 1.

Использование двоичных чисел в компьютере связано с битовой структурой компьютерной памяти и простотой двоичной арифметики.

? *Вопросы и задания*

1. Назовите преимущества и недостатки двоичной системы счисления по сравнению с десятичной.
2. Какие двоичные числа соответствуют следующим десятичным числам:

   128; 256; 512; 1024?
3. Чему в десятичной системе равны следующие двоичные числа:

   1000001; 10000001; 100000001; 1000000001?
4. Переведите в десятичную систему следующие двоичные числа:

   101; 11101; 101010; 100011; 10110111011.
5. Переведите в двоичную систему счисления следующие десятичные числа:

   2; 7; 17; 68; 315; 765; 2047.
6. Выполните сложение в двоичной системе счисления:

   11 + 1; 111 + 1; 1111 + 1; 11111 + 1.
7. Выполните умножение в двоичной системе счисления:

   111 · 10; 111 · 11; 1101 · 101; 1101 · 1000.

§ 17 Числа в памяти компьютера

Основные темы параграфа:

- *представление целых чисел;*
- *размер ячейки и диапазон значений чисел;*
- *особенности работы компьютера с целыми числами;*
- *представление вещественных чисел;*
- *особенности работы компьютера с вещественными числами.*

Любая информация в памяти компьютера представляется в двоичном виде: последовательностью нулей и единиц. Исторически первым типом данных, с которыми стали работать компьютеры, были числа. Теперь это и числа, и тексты, и изображение, и звук. Работа с данными любого типа в конечном счете сводится к обработке ***двоичных чисел*** — чисел, записываемых с помощью двух цифр — 0 и 1. Поэтому современные компьютерные технологии называют ***цифровыми технологиями.***

В компьютере различаются два типа числовых величин: целые числа и вещественные числа. Различаются способы их представления в памяти компьютера.

Представление целых чисел

Часть памяти, в которой хранится одно число, будем называть ячейкой. Минимальная ячейка, в которой может храниться *целое* число, имеет размер 8 битов — 1 байт. Получим представление десятичного числа 25 в такой ячейке. Для этого нужно перевести число в двоичную систему счисления. Как это делается, вы уже знаете. Результат перевода:

$$25_{10} = 11001_2.$$

Теперь осталось «вписать» его в восьмиразрядную ячейку (записать так называемое внутреннее представление числа). Делается это так:

$$00011001.$$

Число записывается «прижатым» к правому краю ячейки (в младших разрядах). Оставшиеся слева разряды (старшие) заполняются нулями.

Самый старший разряд — первый слева, хранит знак числа. Если число положительное, то в этом разряде ноль, если отрицательное — единица. *Самому большому положительному целому числу соответствует следующий код:*

$$01111111.$$

Чему он равен в десятичной системе? Можно расписать это число в развернутой форме и вычислить выражение. Но можно решить задачу быстрее. Если к младшему разряду этого числа прибавить единицу, то получится число 10000000. В десятичной системе оно равно $2^7 = 128$. Значит:

$$01111111_2 = 128 - 1 = 127.$$

Максимальное целое положительное число, помещающееся в 8-разрядную ячейку, равно 127.

Теперь рассмотрим представление *целых отрицательных чисел*. Как, например, в 8-разрядной ячейке памяти будет представлено число −25? Казалось бы, очевидным ответом является следующий: нужно в представлении числа 25 заменить старший разряд с 0 на 1. К сожалению, в компьютере все несколько сложнее.

Для представления отрицательных целых чисел используется **дополнительный код**.

Получить ***дополнительный код*** можно по следующему алгоритму:

1) записать внутреннее представление положительного числа X;
2) записать ***обратный код*** этого числа заменой во всех разрядах 0 на 1 и 1 на 0;
3) к полученному числу прибавить 1.

Определим по этим правилам внутреннее представление числа -25_{10} в 8-разрядной ячейке:

1) 00011001
2) 11100110
3) +1

11100111 — это и есть представление числа −25.

В результате выполнения такого алгоритма единица в старшем разряде получается автоматически. Она и является признаком отрицательного значения.

Проверим полученный результат. Очевидно, что при сложении чисел +25 и −25 должен получиться ноль.

$$\begin{array}{r} 00011001 \\ +\ 11100111 \\ \hline 1\ 00000000 \end{array}$$

Единица в старшем разряде, получаемая при сложении, выходит за границу ячейки и исчезает. *В ячейке остается ноль*!

Из этого примера теперь можно понять, почему представление отрицательного числа называется дополнительным кодом.

Представление восьмиразрядного отрицательного числа $-X$ дополняет представление соответствующего положительного числа $+X$ до значения 2^8.

Размер ячейки и диапазон значений чисел

Наибольшее по модулю отрицательное значение в 8-разрадной ячейке равно $-2^7 = -128$. Его внутреннее представление: 10000000. Таким образом, диапазон представления целых чисел в восьмиразрядной ячейке следующий:

$$-128 \le X \le 127, \text{ или } -2^7 \le X \le 2^7 - 1.$$

Восьмиразрядное представление целых чисел обеспечивает слишком узкий диапазон значений. Если требуется больший диапазон, нужно использовать ячейки большего размера. Для 16-разрядной ячейки диапазон значений будет следующим:

$$-2^{15} \le X \le 2^{15} - 1, \text{ или } -32\ 768 \le X \le 32\ 767.$$

Теперь становится очевидной обобщенная формула для диапазона целых чисел в зависимости от разрядности N ячейки:

$$-2^{N-1} \le X \le 2^{N-1} - 1.$$

Диапазон для 32-разрядной ячейки получается достаточно большим:

$$-2^{31} \le X \le 2^{31} - 1, \text{ или}$$
$$-2\ 147\ 483\ 648 \le X \le 2\ 147\ 483\ 647.$$

Особенности работы компьютера с целыми числами

Выполняя на компьютере вычисления с целыми числами, нужно помнить об ограниченности диапазона допустимых значений. Выход результатов вычислений за границы допус-

тимого диапазона называется ***переполнением***. Переполнение при вычислениях с целыми числами не вызывает прерывания работы процессора. Машина продолжает считать, но результаты могут оказаться неправильными.

Представление вещественных чисел

Целые и дробные числа в совокупности называются вещественными числами. В математике также используется термин «действительные числа». Решение большинства математических задач сводится к вычислениям с вещественными числами.

Всякое вещественное число (X) можно записать в виде произведения ***мантиссы*** m и основания системы счисления p в некоторой целой степени n, которую называют ***порядком***:

$$X = m \cdot p^n.$$

Например, число 25,324 можно записать в таком виде: $0{,}25324 \cdot 10^2$. Здесь $m = 0{,}25324$ — мантисса, $n = 2$ — порядок. Порядок указывает, на какое количество позиций и в каком направлении должна сместиться десятичная запятая в мантиссе.

Чаще всего для хранения вещественных чисел в памяти компьютера используется либо 32-разрядная, либо 64-разрядная ячейка. Первый вариант называется представлением с обычной точностью, второй — представлением с удвоенной точностью. В ячейке хранятся два числа в двоичной системе счисления: мантисса и порядок. Здесь мы не будем подробно рассматривать правила представления вещественных чисел. Отметим лишь основные следствия, вытекающие из этих правил, которые важно знать пользователю компьютера, занимающемуся математическими вычислениями.

Особенности работы компьютера с вещественными числами

1. Диапазон вещественных чисел ограничен. Но он значительно шире, чем для рассмотренного ранее способа представления целых чисел. Например, при использовании 32-разрядной ячейки этот диапазон следующий:

$$-3{,}4 \cdot 10^{38} \le X \le 3{,}4 \cdot 10^{38}.$$

2. Выход за диапазон (переполнение) – аварийная ситуация для процессора, который прерывает свою работу.
3. Результаты машинных вычислений с вещественными числами содержат погрешность. При использовании удвоенной точности эта погрешность уменьшается.

Коротко о главном

В памяти компьютера целые числа представляются в двоичной системе счисления и могут занимать ячейку размером 8, 16, 32 и т. д. битов.

Диапазон значений целых чисел ограничен. Чем больше размер ячейки, тем шире диапазон.

При выходе результатов вычислений с целыми числами за допустимый диапазон работа процессора не прерывается. При этом результаты могут оказаться неверными.

Вещественные числа представляются в виде совокупности мантиссы и порядка в двоичной системе счисления. Обычный размер ячейки — 32 или 64 бита.

Результаты вычислений с вещественными числами приближенные. Переполнение приводит к прерыванию работы процессора.

Вопросы и задания

1. Как в памяти компьютера представляются целые положительные и отрицательные числа?
2. Укажите, каков был бы диапазон значений целых чисел, если бы для их хранения использовалась 4-разрядная ячейка.
3. Запишите внутреннее представление следующих десятичных чисел, используя 8-разрядную ячейку:
   а) 32; б) –32; в) 102; г) –102; д) 126; е) –126.
4. Определите, каким десятичным числам соответствуют следующие двоичные коды 8-разрядного представления целых чисел.
   а) 00010101; б) 11111110; в) 00111111; г) 10101010.

§ 18 Что такое электронная таблица

Основные темы параграфа:
- *отличие электронной таблицы от базы данных;*
- *структура электронной таблицы;*
- *данные в электронной таблице;*
- *режимы отображения данных.*

Отличие электронной таблицы от базы данных

Теперь снова речь пойдет о таблицах. Но это особые таблицы, которые нельзя создать с помощью СУБД.

Представьте себе, что вы являетесь владельцем небольшого торгового павильона, в котором реализуется молочная продукция. Вам приходится вести самые различные формы учета товара. Пусть, например, один из учетных документов должен выглядеть так, как показано в табл. 4.1.

Таблица 4.1. Таблица учета продажи молочных продуктов

Продукт	Цена	Поставлено	Продано	Осталось	Выручка
Молоко	20	100	100	0	2000
Сметана	10,2	85	70	15	714
Творог	18,5	125	110	15	2035
Йогурт	5,4	250	225	25	1215
Сливки	15,2	50	45	5	684

Но это обычная прямоугольная таблица, — скажете вы, — и ее можно хранить в компьютере в виде реляционной базы данных!

Верно! Но обратите внимание на следующую особенность этой таблицы: *в ней есть поля, значения которых вычисляются через значения других полей.* Таким полем является поле «Выручка», значение этого поля равно произведению количества проданного товара на цену, а также поле «Оста-

лось», значение которого вычисляется как разность между количеством поставленного товара и количеством проданного. Такие поля будем называть ***вычисляемыми***, или ***зависимыми***.

Поля: «Продукт», «Цена», «Поставлено», «Продано» являются ***независимыми***. Независимые поля содержат ***исходные данные*** для расчетов.

Представьте, что вы создали на компьютере такую базу данных. Но ситуация в магазине постоянно меняется: продукты продаются, растет выручка, подвозятся новые партии товара. Если вы хотите постоянно поддерживать в базе данных достоверную информацию, то придется несколько раз в день вносить в нее изменения. При этом вы будете изменять не только исходные данные (поля «Поставлено» и «Продано»), но и пересчитывать вручную значения зависимых полей «Осталось» и «Выручка».

Возможно, что у кого-то из вас появилась такая мысль: вот если бы значения вычисляемых полей пересчитывались в таблице автоматически с изменением исходных данных! Но в базе данных такое невозможно.

Может быть именно так рассуждали авторы одной из самых замечательных идей в области информационных технологий: *идеи **электронной таблицы***.

Прикладные программы, предназначенные для работы с электронными таблицами, называются **табличными процессорами**.

Существует достаточно много разнообразных вариантов табличных процессоров. Однако с точки зрения пользователя они очень похожи друг на друга. Поняв принцип устройства электронной таблицы, легко освоить работу с любым конкретным табличным процессором.

Структура электронной таблицы

Что же представляет собой электронная таблица? Вот как выглядит заполненная электронная таблица с учетным документом (табл. 4.2).

Таблица 4.2. Электронная таблица в режиме отображения формул

	A	B	C	D	E	F
1	Продукт	Цена	Поставлено	Продано	Осталось	Выручка
2	*Молоко*	20	100	100	=C2–D2	=B2*D2
3	*Сметана*	10,2	85	70	=C3–D3	=B3*D3
4	*Творог*	18,5	125	110	=C4–D4	=B4*D4
5	*Йогурт*	5,4	250	225	=C5–D5	=B5*D5
6	*Сливки*	15,2	50	45	=C6–D6	=B6*D6

Электронная таблица, подобно шахматной доске, состоит из клеток, которые принято называть ***ячейками***. Строки и столбцы таблицы имеют обозначения. Чаще всего строки нумеруются числами, а столбцы обозначаются буквами (буквы латинского алфавита). Как и на шахматной доске, каждая клетка имеет свое ***имя (адрес)***, состоящее из имени столбца и номера строки. Например: A1, C13, F24 и т. п.

Но если на шахматной доске всего 8 × 8 = 64 клетки, то в электронной таблице ячеек значительно больше. Например, у табличного процессора MS Excel таблица максимального размера содержит 256 столбцов и 65 536 строк. Поскольку в латинском алфавите всего 26 букв, то, начиная с 27-го столбца, используются двухбуквенные обозначения также в алфавитном порядке:

AA, AB, AC, ..., AZ, BA, BB, BC, ..., BZ, CA ...

Последний, 256-й столбец имеет имя IV (не путайте с римской цифрой). Значит, существуют ячейки с такими, например, именами: DL67, HZ10234 и т. п.

Разумеется, столь большая таблица не может вся поместиться на экране. Экран дисплея — это окно, через которое пользователь видит только часть таблицы. Но это окно можно переместить в любое ее место.

Данные в электронной таблице

Все данные таблицы размещаются в ячейках. Содержимым ячейки может быть ***текст***, ***числовое значение*** или ***формула***. Табличный процессор должен «знать», данное какого типа хранится в конкретной ячейке таблицы, для того чтобы

правильно интерпретировать ее содержимое. Текст и числа рассматриваются как константы. Изменить их можно только путем редактирования соответствующих ячеек. Значения же формул автоматически пересчитываются, как только хотя бы один их операнд изменится.

Режимы отображения данных

Таблица 4.2 находится в режиме ***отображения формул***, который позволяет проследить алгоритм табличных вычислений. В режиме отображения значений на экране видны результаты вычислений по формулам. Таблица 4.3 — это та же самая электронная таблица, но переведенная в режим ***отображения значений***.

Таблица 4.3. Электронная таблица в режиме отображения значений

	A	B	C	D	E	F
1	Продукт	Цена	Поставлено	Продано	Осталось	Выручка
2	*Молоко*	20	100	100	0	2000
3	*Сметана*	10,2	85	70	15	714
4	*Творог*	18,5	125	110	15	2035
5	*Йогурт*	5,4	250	225	25	1215
6	*Сливки*	15,2	50	45	5	684

Коротко о главном

Электронные таблицы предназначены для организации табличных расчетов на компьютере. Прикладные программы, работающие с электронными таблицами, называются *табличными процессорами*.

Наименьшая структурная единица электронной таблицы — ячейка. Имя ячейки складывается из буквенного имени столбца и номера строки.

В ячейке может помещаться текст (символьная последовательность), число, формула.

Ячейки, в которые пользователь заносит числа, содержат исходные данные для вычислений. В ячейках с формулами получаются результаты вычислений.

Изменение исходных данных мгновенно приводит к пересчету формул, в которые эти данные входят.

Электронные таблицы (так же как и базы данных) можно рассматривать как информационные модели реальных объектов.

Вопросы и задания

1. В чем состоит существенное отличие электронной таблицы от таблицы реляционной базы данных?
2. Что такое табличный процессор?
3. Как именуются ячейки таблицы? Какая информация может храниться в ячейках?
4. В чем разница между режимом отображения формул и режимом отображения значений?
5. Что происходит в электронной таблице в результате замены *числа в ячейке на новое значение?*

§ 19 Правила заполнения таблицы

Основные темы параграфа:
- *тексты в ЭТ;*
- *правила записи чисел;*
- *правила записи формул;*
- *подготовка таблицы к расчетам.*

Тексты в ЭТ

При вводе в ячейку таблицы последовательности символов, которая не может быть воспринята как число или формула, табличный процессор воспринимает ее как текст, т. е. как символьную информацию. Кроме того, любая последовательность, ввод которой начинается с апострофа ('), воспринимается как текст (апостроф не вводится). В табл. 4.2 и 4.3 ячейки в первой строке и в столбце А заполнены текстами.

Правила записи чисел

В записях исходных данных, а также в математических формулах присутствуют числа — числовые константы, которые разделяются на *целые* и ***вещественные*** (действительные). Запись целых числовых констант не вызывает затруднений. Например:

25; −3456; +2134567.

Вещественные числа могут иметь любые значения: целые, дробные, смешанные. ***Вещественные*** константы можно записывать двумя способами: в форме ***с фиксированной запятой*** (обычная форма) и в форме ***с плавающей запятой***.

Запись числовой константы в форме с фиксированной запятой предполагает, что число содержит целую и дробную части, разделенные десятичной *запятой*. Например, числовая константа 3,1415 записывается как 3,1415.

> Числовая константа в форме с плавающей запятой трактуется как мантисса, умноженная на 10 в степени, равной порядку.

Например, в записи числа в виде $0{,}5 \times 10^9$ сомножитель 0,5 является мантиссой, а показатель степени 9 является порядком.

При записи в электронную таблицу числовой константы в форме с плавающей запятой сначала пишется ***мантисса***, затем — латинская буква E (прописная или строчная), после нее — ***порядок***. Мантисса может быть целой константой или константой с фиксированной запятой, а порядок — только целой константой. Порядок указывает, на какое количество позиций и в каком направлении должна сместиться запятая в мантиссе.

Например, математическая запись $0{,}5 \times 10^9$ в электронной таблице выглядит так: 0.5e9; а 1×10^{-2} запишется как 1e−2.

Обычно форма с плавающей запятой используется для представления очень больших или очень маленьких чисел. Например: 2e+25; 1e−30.

Правила записи формул

Запись формулы в ячейке начинается со знака «равно» (=). Формулы записываются по строго определенным прави-

лам. Их нетрудно освоить. Формулы содержат числа, имена ячеек, знаки операций, круглые скобки, имена функций. Вот как выглядят знаки операций:

+ (сложение);
− (вычитание);
* (умножение);
/ (деление);
^ (возведение в степень).

Вся формула пишется в строку, символы выстраиваются последовательно друг за другом.

Формулы в табл. 4.2 имеют следующий смысл:

C2 − D2 — из числа в ячейке C2 вычесть число в ячейке D2, результат будет помещен в ячейку E2, в которой записана эта формула;

B2*D2 — число в ячейке B2 умножить на число в ячейке D2, результат будет помещен в ячейку F2.

Вот еще примеры записи формул:

2.5*A1 + B2*C3

(B3 − C1)/(B3 + C1)

F7/2 + G7/3

(A5 − 1)^2

Нетрудно понять смысл этих математических выражений. Как всегда, в первую очередь выполняются операции в скобках. При отсутствии скобок последовательность операций определяется их старшинством. По порядку убывания старшинства операции располагаются так:

^ (возведение в степень);
*, / (умножение, деление);
+, − (сложение, вычитание).

Несколько подряд записанных операций одинакового старшинства выполняются в порядке их записи в формуле (слева направо). Например, формула M13/365*N4 будет соответствовать математической записи:

$$\frac{M13}{365} \cdot N4.$$

В формулах допускается употребление некоторых математических функций. Например математическое выражение

$$\sqrt{B5 + B6}$$

запишется так:

КОРЕНЬ(B5+B6).

Здесь КОРЕНЬ — имя функции «квадратный корень». Аргументы всегда пишутся после имени функции в круглых скобках.

Подготовка таблицы к расчетам

Совсем не обязательно при заполнении электронной таблицы сразу заносить в нее исходные данные. *Таблицу можно предварительно подготовить к вычислениям в виде бланка, не заполненного числами.* Для этого нужно заполнить все ячейки с текстовой информацией и записать в вычисляемые ячейки соответствующие формулы. В режиме отображения значений такая таблица выглядит почти пустой: в вычисляемых ячейках будут высвечиваться нулевые значения. Как только пользователь начнет заносить в нее числовые данные, в зависимых ячейках сразу же будут появляться вычисленные по формулам результаты. В табл. 4.4 приведен пример заготовки рассмотренного учетного документа в режиме отображения формул.

Таблица 4.4. Таблица, подготовленная к расчетам

	A	B	C	D	E	F
1	Продукт	Цена	Поставлено	Продано	Осталось	Выручка
2	*Молоко*				=C2–D2	=B2*D2
3	*Сметана*				=C3–D3	=B3*D3
4	*Творог*				=C4–D4	=B4*D4
5	*Йогурт*				=C5–D5	=B5*D5
6	*Сливки*				=C6–D6	=B6*D6

Коротко о главном

При вводе в таблицу любая последовательность символов, которая не может быть числом или формулой, а также вводимая после апострофа, воспринимается как текст.

Числа в электронной таблице представляются в форме с фиксированной запятой и в форме с плавающей запятой.

Формулы могут включать в себя числа, имена ячеек, функции, знаки операций, круглые скобки.

Предварительное занесение в таблицу лишь символьных данных и формул равносильно программированию таблицы для последующих расчетов.

Вопросы и задания

1. Как ввести текст в ячейку электронной таблицы?
2. В каких двух форматах представляются числа? В чем разница между ними?
3. Сформулируйте правила записи формул. Что произойдет, если при вводе формулы вы нарушите эти правила?
4. Как можно заранее подготовить таблицу для вычислений?
5. Запишите в традиционной математической форме следующие формулы из электронной таблицы, предварительно ответив на вопрос, в какой последовательности будут выполняться математические операции.

C2+A5/3Q;	(C2+A5)/3;	C2/(A5+3);
A1*A2/D12*D3Q;	A1*A2/D12/D3;	A1*A2/(D12*D3);
B2^2–D3^5Q;	F4+(A4*5)^3;	F4^3*A4.

6. Постройте электронную таблицу «Оплата электроэнергии» для расчета ежемесячной платы за расход электроэнергии в течение года. Исходной информацией являются показания счетчика в начале каждого месяца и стоимость одного киловатт-часа.

§ 20 Работа с диапазонами. Относительная адресация

Основные темы параграфа:
- *что такое диапазон (блок);*
- *функции обработки диапазона;*
- *принцип относительной адресации;*
- *сортировка таблицы.*

Что такое диапазон (блок)

Табличные процессоры позволяют выполнять некоторые вычисления с целой группой ячеек, называемой диапазоном.

Диапазон (блок, фрагмент) — любая прямоугольная часть таблицы.

Обычно диапазон обозначается именами верхней левой и нижней правой ячеек, разделенными двоеточием. Например, в табл. 4.4 диапазон, состоящий из вычисляемых ячеек, обозначается следующим образом: E2:F6 (в табл. 4.4 он выделен темным фоном). Минимальным диапазоном является одна ячейка таблицы.

Функции обработки диапазона

В каждом табличном процессоре имеется целый набор функций, применяемых к диапазонам. Это ***суммирование чисел*** (СУММ), входящих в диапазон, ***вычисление среднего значения*** (СРЗНАЧ), ***нахождение максимального*** (МАКС) и ***минимального*** (МИН) ***значения*** и некоторые другие. Такие функции называются *статистическими*.

Предположим, что в конце рабочего дня необходимо подсчитать выручку, полученную за день от продажи молочных продуктов. Для этого в таблице 4.3 нужно просуммировать все числа из диапазона F2:F6. Пусть функция суммирования обозначается словом СУММ. Тогда нужная нам формула запишется так: СУММ(F2:F6). Она обозначает следующее:

СУММ(F2:F6)=F2+F3+F4+F5+F6.

Запишем формулу суммирования в ячейку F7, а в ячейку E7 — текст«ВСЕГО:». Результат — в табл. 4.5.

Таблица 4.5. Таблица с вычислением суммарной выручки

	A	B	C	D	E	F
1	Продукт	Цена	Поставлено	Продано	Осталось	Выручка
2	*Молоко*	20	100	100	0	2000
3	*Сметана*	10,2	85	70	15	714
4	*Творог*	18,5	125	110	15	2035
5	*Йогурт*	5,4	250	225	25	1215
6	*Сливки*	15,2	50	45	5	684
7					ВСЕГО:	6648

Табличные процессоры позволяют манипулировать с диапазонами электронной таблицы. *К операциям манипулирования относятся: удаление, вставка, копирование, перенос, сортировка диапазонов таблицы.* Эти операции выполняются с помощью ***команд табличного процессора***. Обычно эти команды пользователь выбирает из ***меню команд***.

Принцип относительной адресации

Казалось бы, в результате таких манипуляций расчетные формулы могут стать неверными, поскольку изменятся адреса перемещенных на новое место ячеек. Чтобы такого не происходило, в электронной таблице реализован ***принцип относительной адресации.***

Согласно принципу относительной адресации, адреса ячеек, используемые в формулах, определены не абсолютно, а относительно ячейки, в которой располагается формула.

Следствием этого принципа является следующее правило:

Всякое изменение места расположения формулы ведет к автоматическому изменению адресов ячеек в этой формуле.

Поясним сказанное на примере. Пусть при подготовке таблицы для расчета продажи товара на следующий день владелец павильона знает, что в этот день не будут подвозиться сметана и творог. Поэтому две соответствующие строки из табл. 4.4 можно удалить. Это делается с помощью команды

УДАЛИТЬ A3:F4

На место удаленных строк сдвигаются строки снизу. В результате таблица преобразуется к виду, показанному в табл. 4.6.

Таблица 4.6. Таблица после удаления двух строк

	A	B	C	D	E	F
1	Продукт	Цена	Поставлено	Продано	Осталось	Выручка
2	*Молоко*				=C2–D2	=B2*D2
3	*Йогурт*				=C3–D3	=B3*D3
4	*Сливки*				=C4–D4	=B4*D4

Обратите внимание на две последние строки. В присутствующих в них формулах изменились адреса ячеек. Здесь был учтен сдвиг на две строки вверх; сработал принцип относительной адресации.

Сортировка таблицы

Допустим, владелец торгового павильона хочет узнать, какие товары пользуются наибольшим спросом. Для этого достаточно упорядочить строки таблицы по убыванию чисел в столбце «Продано». Большинство табличных процессоров позволяют производить сортировку (упорядочение) таблицы по какому-либо признаку. Для нашего примера формируется команда такого типа:

СОРТИРОВАТЬ СТОЛБЕЦ D ПО УБЫВАНИЮ

Применение этой команды к табл. 4.5 в режиме отображения значений даст результат, показанный в табл. 4.7.

Таблица 4.7. Результат сортировки таблицы по столбцу «Продано»

	A	B	C	D	E	F
1	Продукт	Цена	Поставлено	Продано	Осталось	Выручка
2	*Йогурт*	5,4	250	225	25	1215
3	*Творог*	18,5	125	110	15	2035
4	*Молоко*	20	100	100	0	2000
5	*Сметана*	10,2	85	70	15	714
6	*Сливки*	15,2	50	45	5	684
7					ВСЕГО:	6648

Отсюда видно, что наибольшим спросом пользуется йогурт, а меньше всего покупают сливки.

Эта же таблица в режиме отображения формул — табл. 4.8.

Таблица 4.8. Отсортированная таблица в режиме отображения формул

	A	B	C	D	E	F
1	Продукт	Цена	Поставлено	Продано	Осталось	Выручка
2	*Йогурт*	5,4	250	225	=C2–D2	=B2*D2
3	*Творог*	18,5	125	110	=C3–D3	=B3*D3
4	*Молоко*	20	100	100	=C4–D4	=B4*D4
5	*Сметана*	10,2	85	70	=C5–D5	=B5*D5
6	*Сливки*	15,2	50	45	=C6–D6	=B6*D6
7					ВСЕГО:	=СУММ(F2:F6)

Снова сработал принцип относительной адресации. Формулы изменились в соответствии с изменением месторасположения строк.

Коротко о главном

Диапазон (блок, фрагмент) таблицы — это любая ее прямоугольная часть (в том числе часть строки, часть столбца или одна ячейка).

Для выполнения расчетов с числовыми фрагментами используются статистические функции: суммирование, усреднение, нахождение наибольшего и наименьшего значений и др.

С фрагментами таблицы можно производить операции манипулирования: удаление, вставку, перенос, сортировку.

Принцип относительной адресации: адреса ячеек в формулах определены не абсолютно, а относительно местонахождения этой формулы. Следствие: при перемещении формулы в другую ячейку соответствующим образом изменяются адреса ячеек, содержащиеся в ней.

Вопросы и задания

1. Что такое диапазон? Как он обозначается?
2. Какие вычисления можно выполнять над целым диапазоном?
3. Что понимается под манипулированием диапазонами ЭТ?
4. Что такое принцип относительной адресации? В каких ситуациях он проявляется?
5. В ячейке D7 записана формула (C3+C5)/D6. Как она изменится при переносе этой формулы в ячейку:

   а) D8; б) E7; в) C6; г) F10 ?

6. В ячейке E4 находится формула СУММ(A4:D4). Куда она переместится и как изменится при:
   а) удалении строки 2; б) удалении строки 7; в) вставке пустой строки перед строкой 4; г) удалении столбца 3; д) вставке пустого столбца перед столбцом 6.
7. К таблице «Оплата электроэнергии», полученной при выполнении задания 6 из предыдущего параграфа, добавьте расчет всей выплаченной суммы за год.
8. К таблице из предыдущей задачи добавьте расчет выплаченной суммы денег за каждый квартал (квартал — 3 месяца).

§ 21 Деловая графика. Условная функция

Основные темы параграфа:

- *графические возможности табличного процессора;*
- *типы диаграмм;*
- *условная функция.*

Графические возможности табличного процессора

Замечательным свойством электронных таблиц является возможность графического представления числовой информации, содержащейся в таблице. Для этого существует специальный ***графический режим*** работы табличного процессора. Графики придают наглядность числовым зависимостям.

Типы диаграмм

Табличные процессоры дают возможность получать самые различные формы диаграмм и графиков. Ниже на рисунках показаны два типа диаграмм: ***круговая*** на рис. 4.1 и ***столбчатая*** на рис. 4.2. Исходные данные для этих диаграмм извлекаются из одинаковых диапазонов ячеек A2:A6 и D2:D6 таблиц из предыдущего параграфа. Первый диапазон содержит названия продуктов, второй — количество проданных единиц каждого продукта. Из диаграмм сразу видно, что наибольшим спросом у покупателей пользуется йогурт.

Рис. 4.1. Круговая диаграмма

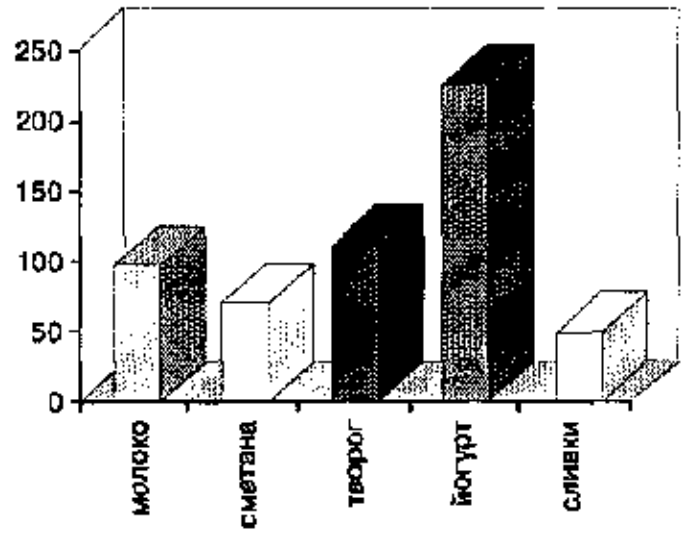


Рис. 4.2. Столбчатая диаграмма (гистограмма)

Круговую диаграмму обычно используют в тех случаях, когда нужно показать, какую часть от целого (круга) составляют отдельные величины (секторы). Столбчатая диаграмма (гистограмма) позволяет наглядно сопоставить между собой отдельные величины.

Условная функция

Продолжим обсуждение задачи об учете продажи молочных продуктов в торговом павильоне. В случае если тот или иной продукт продан полностью, необходимо организовать его подвоз в торговый павильон. Чтобы отразить это в электронной таблице, добавим в табл. 4.8 новый столбец с названием «Подвоз». В ячейках этого столбца будет высвечиваться слово «Да», если подвоз соответствующего продукта необходим, и «Нет», если продукт подвозить не надо. Разумеется, значения «Да» или «Нет» табличный процессор должен определить сам автоматически.

Для решения задачи воспользуемся ***условной функцией***. Общий вид условной функции следующий:

ЕСЛИ(<условие>; <выражение1>; <выражение2>)

<Условие> — это логическое выражение, которое может принимать значение ИСТИНА или ЛОЖЬ. С логическими выражениями вы познакомились в главе о базах данных. В электронных таблицах они имеют тот же смысл. <Выражение 1> и <выражение 2> могут быть числами, формулами или текстами.

Условная функция, записанная в ячейку таблицы, выполняется так:
если <условие> истинно, то значение данной ячейки определит <выражение 1>,
в противном случае — <выражение 2>.

В нашем случае условие означает проверку на равенство нулю количества оставшегося продукта. В качестве выражений 1 и 2 выступают текстовые константы «Да» и «Нет».

После внесенных изменений учетный документ примет вид, представленный в табл. 4.9 (в режиме отображения формул) и в табл. 4.10 (в режиме отображения значений).

Таблица 4.9. Таблица с условной функцией в режиме отображения формул

	A	B	C	D	E	F	G
1	Продукт	Цена	Поставле-но	Продано	Осталось	Выручка	Подвоз
2	*Йогурт*	5,4	250	225	=C2–D2	=B2*D2	=ЕСЛИ(E2=0;"Да";"Нет")
3	*Творог*	18,5	125	110	=C3–D3	=B3*D3	=ЕСЛИ(E3=0;"Да";"Нет")
4	*Молоко*	20	100	100	=C4–D4	=B4*D4	=ЕСЛИ(E4=0;"Да";"Нет")
5	*Сметана*	10,2	85	70	=C5–D5	=B5*D5	=ЕСЛИ(E5=0;"Да";"Нет")
6	*Сливки*	15,2	50	45	=C6–D6	=B6*D6	=ЕСЛИ(E6=0;"Да";"Нет")
7					ВСЕГО:	=СУММ(F2:F6)	

Таблица 4.10. Таблица с условной функцией в режиме отображения значений

	A	B	C	D	E	F	G
1	*Продукт*	*Цена*	*Поставлено*	*Продано*	*Осталось*	*Выручка*	*Подвоз*
2	*Йогурт*	5,4	250	225	25	54	Нет
3	*Творог*	18,5	125	110	15	274	Нет
4	*Молоко*	20	100	100	0	300	Да
5	*Сметана*	10,2	85	70	15	294	Нет
6	*Сливки*	15,2	*50*	*45*	*5*	*144*	Нет
7					ВСЕГО:	1552	

Коротко о главном

В современных табличных процессорах реализована деловая графика: возможность построения диаграмм и графиков по числовым данным в таблице.

Условная функция имеет следующий формат:

ЕСЛИ(<условие>; <выражение1>; <выражение2>)

Здесь <условие> — логическое выражение. Если <условие> истинно, то значение ячейки определяет <выражение1>, если ложно — <выражение2>.

Вопросы и задания

1. Что такое деловая графика?
2. Какой вид имеет условная функция? Как она выполняется?
3. На основании таблицы «Оплата электроэнергии» (задание 6 из § 19) постройте столбчатую диаграмму, отражающую ежемесячный расход электроэнергии в течение года.
4. По таблице «Оплата электроэнергии» с добавленным расчетом выплаченной суммы денег за каждый квартал (задание 8 из § 20) получите круговую диаграмму, отражающую относительный размер выплат в каждом квартале.
5. К таблице «Оплата электроэнергии» добавьте расчет среднемесячной платы, а также придумайте и реализуйте способ подсчета количества месяцев, плата за которые была выше среднемесячной.

§ 22 Логические функции и абсолютные адреса

Основные темы параграфа:

- *запись и выполнение логических функций;*
- *абсолютные адреса;*
- *функция времени.*

Запись и выполнение логических функций

Продолжим совершенствование таблицы учета продажи молочных продуктов. В условии подвоза товара желательно учесть следующее обстоятельство: подвозить товар не имеет смысла, если торговый павильон заканчивает работу. Это тоже можно предусмотреть в электронной таблице. Ячейку E9 будем использовать для хранения времени (в часах), оставшегося до конца рабочего дня. Условие подвоза товара сформулируем так: *товар подвозить, если оставшееся его количество равно нулю И до конца рабочего дня осталось больше двух часов.*

При записи сформулированного выше условия в форме логического выражения должна быть использована логическая операция И (конъюнкция, логическое умножение). Работая с базами данных, вы познакомились с логическими операциями. Однако в электронных таблицах несколько иные правила записи логических выражений, содержащих логические операции.

В электронных таблицах логические операции (И, ИЛИ, НЕ) рассматриваются как логические функции.

Например, логическое выражение, которое примет значение ИСТИНА, если выполнится сформулированное выше условие подвоза товара, пишется следующим образом (для второй строки, т. е. для йогурта):

И(E2=0; E9>2)

Перед скобками ставится имя логической операции (функции), а в скобках — логические операнды.

Следовательно, теперь условная функция в ячейке G2 должна выглядеть так:

ЕСЛИ(И(E2=0; E9>2); "Да"; "Нет")

Но в этой формуле таится опасность. Вам уже известно, что при любых манипуляциях с таблицей, связанных с переносом формул в другие ячейки, происходит изменение адресов переменных. Работает принцип относительной адресации. Однако в данном случае адрес ячейки E9 не должен изменяться в формуле. Иначе говоря, этот адрес должен быть не относительным, а ***абсолютным.***

Абсолютные адреса

В электронных таблицах существует способ «замораживания» адресов. На «замороженный» в формуле адрес ячейки не распространяется принцип относительности. Обычно для этой цели используется значок «$». Можно «заморозить» только номер строки или только имя столбца. Чтобы сделать абсолютным (неизменным при любом переносе формулы в таблице) адрес ячейки, нужно знак «$» писать дважды: E9.

Теперь должно быть понятно, что условную функцию, *решающую вопрос о подвозе товара, следует записать так:*

ЕСЛИ(И(E2=0;E9>2);"Да";"Нет")

Функция времени

Осталось обсудить формулу, вычисляющую количество времени, оставшееся до конца рабочего дня. Можно, посмотрев на часы, вручную вставить это время в ячейку E9. Но в современных табличных процессорах существуют специальные функции (функции времени), позволяющие получить текущее время. Это возможно благодаря тому, что в состав аппаратной части компьютеров входит *таймер* — внутренние часы компьютера. Если рабочий день заканчивается в 20 часов, то формула должна быть такой: 20 – ТЕКУЩИЙ ЧАС. *Пусть, например, функция определения текущего часа* записывается так: ЧАС(ТДАТА( )).

После внесения изменений таблица в режиме отображения формул примет вид табл. 4.11, а в режиме отображения значений — табл. 4.12.

Обратите внимание на то, что из табл. 4.12 следует, что молоко уже закончилось, но подвозить его не следует, так как до конца рабочего дня осталось 2 часа.

Коротко о главном

При записи логических выражений можно пользоваться логическими операциями: И, ИЛИ, НЕ. В электронных таблицах логические операции употребляются как функции.

Возможно «замораживание» адресов ячеек, используемых в формулах. «Замороженный» адрес становится абсолютным, т. е. на него не распространяется принцип относительной адресации.

Таблица 4.11. Таблица в режиме отображения формул

	A	B	C	D	E	F	G
1	*Продукт*	*Цена*	*Поставлено*	*Продано*	*Осталось*	Выручка	*Подвоз*
2	*Йогурт*	5,4	250	225	=C2 – D2	=B2*D2	=ЕСЛИ(И(E2=0;E9>2);"да";"нет")
3	*Творог*	18,5	125	110	=C3 – D3	=B3*D3	=ЕСЛИ(И(E3=0;E9>2);"да";"нет")
4	*Молоко*	20,0	100	100	=C4 – D4	=B4*D4	=ЕСЛИ(И(E4=0;E9>2);"да";"нет")
5	*Сметана*	10,2	85	70	=C3 – D3	=B5*D5	=ЕСЛИ(И(E5=0;E9>2);"да";"нет")
6	*Сливки*	15,2	50	45	=C6 – D6	=B6*D6	=ЕСЛИ(И(E6=0;E9>2);"да";"нет")
7					ВСЕГО:	=СУММ(F2:F6)	
8							
9	Осталось	до	конца	дня:	20–ЧАС(ТДАТА())	часа	

Таблица 4.12. Таблица в режиме отображения значений

	A	B	C	D	E	F	G
1	Продукт	Цена	Поставлено	Продано	Осталось	Выручка	Подвоз
2	*Йогурт*	5,4	250	225	25	1215	Нет
3	*Творог*	18,5	125	110	15	2035	Нет
4	*Молоко*	20,0	100	100	0	2000	Нет
5	*Сметана*	10,2	85	70	15	714	Нет
6	*Сливки*	15,2	50	45	5	684	Нет
7					ВСЕГО:	6648	
8							
9	Осталось	до	конца	дня:	2	часа	

? Вопросы и задания

1. Как в электронной таблице реализуются логические операции при записи условных функций?
2. Что такое абсолютный адрес?
3. В таблице «Оплата электроэнергии» (задание 6 из § 19) используйте абсолютный адрес для ячейки, хранящей стоимость 1 кВт · ч электроэнергии.
4. В таблице «Оплата электроэнергии» используйте следующее правило для подсчета суммы оплаты: если израсходовано не более 100 кВт · ч, то цена 1 кВт · ч равна 50 коп.; если израсходовано более 100, но менее 300 кВт · ч, то цена — 60 коп.; если израсходовано не менее 300 кВт · ч, то цена 1 кВт · ч равна 75 коп. Используйте логические функции.

§ 23 Электронные таблицы и математическое моделирование

Основные темы параграфа:

- *математическая модель;*
- *этапы математического моделирования на компьютере;*
- *пример математического моделирования в ЭТ.*

Математическая модель

Что такое компьютерное математическое моделирование? Снова вернемся к теме математического моделирования, обсуждение которой было начато в § 9. Реальную систему, для которой создается математическая модель, принято называть объектом моделирования. Объектами математического моделирования могут быть некоторые конструкции, например, железнодорожный мост или корабль; природные объек-

ты, например месторождение полезных ископаемых, водохранилище, а также процессы и явления, происходящие во времени, например взлет космической ракеты с космодрома, изменение погодных условий в определенной географической точке, изменение со временем численности определенных популяций.

Для людей могут оказаться жизненно важными многие вопросы, связанные с этими объектами и процессами. Например: на какой высоте ракета достигнет первой космической скорости и выйдет на орбиту спутника Земли; до какой предельной температуры нагреется ее оболочка? Какой может быть максимальная нагрузка на железнодорожный мост, при которой не будет происходить его разрушение? Каким будет уровень воды в водохранилище в тех погодных условиях, которые предсказывают метеорологи? Не вымрет ли данная популяция животных через сто лет?

На эти вопросы желательно получить ответы теоретическим путем, поскольку экспериментальный путь либо невозможен, либо он возможен, но опасен. Например, при перегрузке моста можно его разрушить, при перегреве корпуса ракеты ее можно сжечь; а экспериментально проверить, что будет с популяцией животных через сто лет, невозможно. В подобных ситуациях на помощь человеку приходят математическое моделирование и вычислительный эксперимент.

Этапы математического моделирования на компьютере

В математической модели используются количественные (числовые) характеристики объекта. Например, в математической модели полета ракеты учитываются масса и скорость ракеты, сила тяги двигателей, сопротивление атмосферного воздуха, теплоемкость обшивки ракеты, время полета, высота ракеты над поверхностью Земли, плотность атмосферы. Все эти величины связываются между собой через уравнения, отражающие физические законы движения тела в воздушной среде, нагревания тела в процессе трения. Из этих уравнений, зная одни величины — исходные данные, можно вычислить другие величины — результаты. Например, зная массу ракеты, силу тяги двигателей, скорость сгорания топлива, коэффициент трения воздуха о корпус, можно вычислить, какой будет высота и скорость ракеты в данный момент времени, а также температура обшивки ракеты. Часто такие расчеты бывает трудно осуществить

вручную, и тогда используются компьютерные методы решения задачи.

Реализованная на компьютере математическая модель называется **компьютерной математической моделью**, а проведение расчетов с помощью компьютерной модели с целью прогнозирования поведения моделируемой системы называется **вычислительным экспериментом.**

Таким образом, этапы компьютерного математического моделирования следующие:

1) выделение количественных характеристик моделируемой системы, существенных для решаемой задачи;
2) получение математических соотношений (формул, уравнений, систем уравнений и пр.), связывающих эти характеристики;
3) определение способа решения полученной математической задачи и реализация ее на компьютере с помощью прикладных программных средств или на языках программирования;
4) решение поставленной задачи путем проведения вычислительного эксперимента.

В результате вычислительного эксперимента можно получить прогноз поведения исследуемой системы; выяснить вопрос о том, как изменение одних характеристик системы отразится на других.

Одним из видов прикладных программных средств, пригодных для реализации математической модели на компьютере, являются электронные таблицы.

Пример математического моделирования в ЭТ

Чаще всего электронные таблицы используются в задачах такого типа, которые были рассмотрены в предыдущих параграфах: для получения расчетных ведомостей, смет, справок, списков, т. е. в области делопроизводства. Однако электронные таблицы могут оказаться полезными и для научных целей. С их помощью можно строить компьютерные математические модели, проводить вычислительные экспе-

рименты. Рассмотрим пример такого вычислительного эксперимента.

Ученые установили, что прирост какого-либо вида живых организмов за счет рождаемости прямо пропорционален их количеству, а убыль за счет смертности прямо пропорциональна квадрату их количества. Этот закон известен под названием *закона Мальтуса.*

Пусть в одном хозяйстве собираются разводить карпов. Прежде чем запускать мальков в пруд, решили провести расчеты. Согласно закону Мальтуса, изменение числа рыб за один год вычисляется по формуле

$$\Delta N = kN - qN^2.$$

Здесь N — число карпов в начале года, k — коэффициент прироста, q — коэффициент смертности. Экспериментально установлено, что для данного вида рыб (карпы) и в данных условиях (состояние водоема, наличие корма) $k = 1$, $q = 0{,}001$.

Если первоначально в пруд запущено N_0 рыб, то из закона следует, что количество карпов через год будет таким:

$$N_1 = N_0 + (kN_0 - qN_0^2).$$

Через два года:

$$N_2 = N_1 + (kN_1 - qN_1^2)$$

и так далее. Можно написать общую формулу для вычисления количества рыб в i-м году после их запуска:

$$N_i = N_{i-1} + (kN_{i-1} - qN_{i-1}^2) \text{ для } i = 1, 2, 3,\ldots$$

Эта формула является математической моделью процесса размножения рыб в водоеме.

Заполним электронную таблицу для проведения по этой формуле расчета рыбного «поголовья» в пруду в течение нескольких лет — табл. 4.13.

Не надо думать, что всю таблицу приходится вводить посимвольно с клавиатуры. Строки, начиная с 7-й, формируются путем копирования предыдущей строки. При этом относительные адреса изменяются автоматически.

Для получения результатов достаточно занести в ячейку F1 первоначальное число рыб.

Таблица 4.13. Расчет числа рыб в пруду с интервалом в год

	A	B	C	D	E	F
1	k =	1	q =	0,001	N=	
2						
3	Год		Число рыб			
4						
5	1		=F1 + B1*F1–D1*F1*F1			
6	=A5+1		=C5 + B1*C5–D1*C5*C5			
7	=A6+1		=C6 + B1*C6–D1*C6*C6			
8	=A7+1		=C7 + B1*C7–D1*C7*C7			
9	=A8+1		=C8 + B1*C8–D1*C8*C8			
..	.....		.....	.....	....	...

Теперь можно экспериментировать. Проследим, как за 10 лет будет меняться число карпов при разном количестве первоначально запущенных рыб. Вот несколько таблиц с результатами таких расчетов:

k=1 q=0,001 N=100

Год	Число рыб
1	190
2	343
3	569
4	814
5	965
6	998
7	1000
8	1000
9	1000
10	1000

k=1 q=0,001 N=1000

Год	Число рыб
1	1000
2	1000
3	1000
4	1000
5	1000
6	1000
7	1000
8	1000
9	1000
10	1000

k=1 q=0,001 N=1500		k=1 q=0,001 N=2000	
Год	Число рыб	Год	Число рыб
1	750	1	0
2	937	2	0
3	996	3	0
4	1000	4	0
5	1000	5	0
6	1000	6	0
7	1000	7	0
8	1000	8	0
9	1000	9	0
10	1000	10	0

Не правда ли, удивительные результаты? Из приведенных таблиц следует, что невозможно иметь в пруду 2000 карпов и более. Если начальное число рыб меньше 1000, то оно постепенно будет расти до 1000 штук и далее не будет меняться. Если сразу запустить 1000 рыб, то это количество останется неизменным и в последующие годы. Даже если запустить сначала 1500 рыб, то через год их численность в два раза сократится, а затем все равно дойдет до 1000. Если же запустить в пруд 2000 рыб, то через год все они вымрут.

Из полученных результатов рыбоводы могут сделать практические выводы. Приведенные выше таблицы автоматически получались после изменений значения всего лишь в одной ячейке F1.

Коротко о главном

Математической моделью называется информационная модель объекта, выраженная математическими средствами (формулами, уравнениями и т. п.).

Табличный процессор может применяться в качестве инструмента для математического моделирования.

Полученную математическую модель можно использовать для проведения вычислительного эксперимента. Вычисли-

тельный эксперимент — это расчеты с помощью компьютерной математической модели с целью прогноза поведения какой-то системы, с целью выяснения вопроса о том, как изменение одних характеристик системы отражается на других.

? *Вопросы и задания*

1. Что такое математическая модель?
2. Что такое вычислительный эксперимент?
3. Проведите вычислительный эксперимент в таблице расчета количества рыб в пруду, поставив следующую цель: подобрать такие значения параметров k и q, при которых количество рыб за 10 лет может быть доведено до 2000.
4. К решению предыдущей задачи добавьте графическую обработку результатов: график изменения численности рыб с течением времени.

§ 24 Имитационные модели в электронной таблице

Основные темы параграфа:

- *что такое имитационная модель;*
- *пример имитационного моделирования в электронных таблицах.*

Что такое имитационная модель

В предыдущем параграфе вы познакомились с примером реализации в электронной таблице математической модели. Сейчас рассмотрим пример реализации другого типа модели, которая называется ***имитационной моделью***.

Об имитационных моделях говорилось в § 9. Вспомним данное там определение:

Имитационная модель воспроизводит поведение сложной системы, элементы которой могут вести себя случайным образом. Иначе говоря, поведение которых заранее предсказать нельзя.

Пример имитационного моделирования в электронных таблицах

Как и в предыдущем параграфе, пример возьмем из класса моделей, описывающих эволюцию популяций.

Пусть на определенном пространстве случайным образом расселяются живые организмы. В дальнейшем происходит процесс смены поколений: в каких-то местах расселения жизнь сохраняется, в каких-то исчезает. Эти процессы протекают в соответствии с законами эволюции. Законы эволюции в описании модели представляются в виде формальных правил. *Цель моделирования* — проследить изменения в расселении живых организмов со сменой поколений.

Сначала рассмотрим простейший вариант задачи: жизненное пространство одномерное. Это значит, живые организмы расселяются вдоль линии. Будем считать жизненное пространство ограниченным, т. е. рассмотрим отрезок. Отрезок разделяется на ячейки, в пределах каждой из которых может поселиться один организм. Договоримся, что самые крайние ячейки не заселяются. Они определяют границу жизненного пространства.

На рис. 4.3 показано первоначальное расселение организмов на поле, состоящем из 20 ячеек. Организмы поселились в ячейках с номерами 5, 8 и 12. Ячейки 1 и 20 всегда должны быть пустыми.

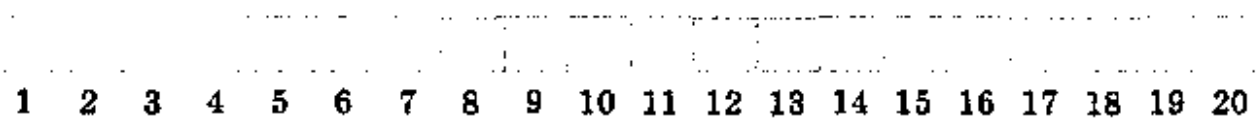


Рис. 4.3. Первоначальное расселение организмов

Теперь сформулируем законы эволюции. В следующем поколении в пустой ячейке жизнь может либо появиться, либо нет. В живой ячейке жизнь может либо сохраниться, либо исчезнуть. На состояние данной ячейки влияют ее ближайшие соседи: два соседа слева и два соседа справа. Если ячейка была живая, и число живых соседей не превышает двух, то в следующем поколении в этой ячейке жизнь сохранится, иначе жизнь исчезнет (погибнет от перенаселения). Если в ячейке жизни не было, но среди ее соседей есть 1, 2 или 3 живые ячейки, то в следующем поколении в этой ячейке появится жизнь. В противном случае ячейка останется пустой.

Следует учитывать, что у ячеек, расположенных у края, число соседей меньше других. У ячейки номер 2 соседи: 1, 3 и 4. Но ячейка 1 всегда пустая. У ячейки номер 3 из четырех соседей живыми могут быть не больше трех (2, 4, 5). Аналогичная ситуация у крайних правых ячеек.

То, что сказано выше, есть модельное описание процесса эволюции популяции. Формализуем это описание. Распределение живых организмов по ячейкам будем кодировать последовательностью из нулей и единиц. Ноль обозначает пустую ячейку, единица — живую. Например, расселение, отображенное на рис. 4.3, кодируется следующим образом:

00001001000100000000

Обозначим буквой n номер ячейки, а значение двоичного числа, соответствующего этой ячейке в текущем поколении, обозначим $R(n)$. В рассматриваемом примере $R(5) = R(8) = R(12) = 1$. Все остальные значения ячеек равны нулю.

Значения кода в n-й ячейке для следующего поколения будем обозначать $S(n)$. Внимательно проанализировав сформулированные выше правила эволюции, приходим к следующей формуле:

$$\text{Если } 1 \le R(n-2) + R(n-1) + R(n) + R(n+1) + R(n+2) \le 3,$$
$$\text{то } S(n) = 1, \text{ иначе } S(n) = 0.$$

Эта формула работает для значений n от 3 до 18. Всегда: $S(1) = S(20) = 0$. Для ячеек с номерами 2 и 19 в данной сумме нужно убрать по одному слагаемому. Но можно поступить иначе, чтобы оставить справедливой данную формулу для всех ячеек жизненного пространства. Для этого к отрезку добавим по одной фиктивной ячейке справа и слева. Их номера будут, соответственно, 0 и 21. В этих ячейках, как и в ячейках 1 и 20, всегда будут храниться нули. Тогда написанную формулу можно применять для n от 2 до 19.

Итак, модель построена и формализована. Однако имитационной моделью она станет только в результате реализации с помощью какого-то программного компьютерного средства. В качестве такого средства выберем табличный процессор.

Моделью жизненного пространства будет строка электронной таблицы. Первая строка — первое поколение, вторая строка — второе поколение и т. д. Тогда номера ячеек будут идентифицироваться именами столбцов таблицы. Ячейка номер 0 — столбец A, ячейка 1 — столбец B и т. д., ячейка 21 — столбец F.

В первой строке выставляем единицы в ячейках, заселенных в первом поколении. Это будут ячейки F1, I1, M1. Незаполненные ячейки по умолчанию приравниваются к нулю.

Теперь в ячейки второй строки нужно записать формулы. Сделать это достаточно один раз. Например, в ячейку C2 занести следующую формулу:

```
ЕСЛИ(И(A1 + B1 + C1 + D1 + E1 >= 1;
A1 + B1 + C1 + D1 + E1 <= 3); 1; 0)
```

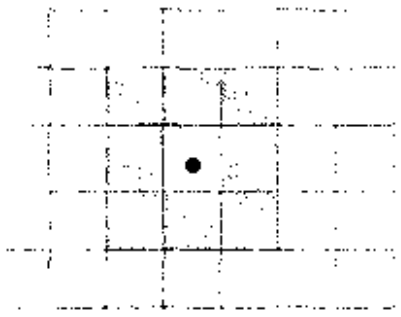
Далее, копируя эту формулу во все остальные ячейки второй строки с D2 по T2, получаем картину распределения живых организмов во втором поколении.

Чтобы получить третье поколение, достаточно скопировать вторую строку (блок C2:T2) в третью строку (блок C3:T3). Так можно продолжать сколько угодно.

На рис. 4.4 показаны результаты имитационного моделирования процесса эволюции исходного расселения живых организмов вплоть до 10-го поколения. Все очень наглядно. Обратите внимание, как драматично развивались события! В шестом поколении наступило состояние перенаселения. В результате в седьмом поколении вымерли все организмы, кроме крайних. Их спасло свободное пространство слева и справа. От них пошла новая волна жизни!

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	F
1						1			1				1									
2			0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0		
3			1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0		
4			1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0		
5			1	0	0	1	1	1	0	0	1	1	1	0	0	0	0	0	1	1		
6			1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1		
7			1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1		
8			1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1		
9			1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1		
10			1	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0	0	1		

Рис. 4.4. Имитация в электронной таблице эволюции десяти поколений популяции живых организмов



Рассмотренная задача является упрощенным (одномерным) вариантом известной модели Дж. Конуэя, которая называется «Жизнь». В этой модели эволюция популяции живых организмов происходит в двумерном пространстве. Рассматривается прямоугольная область, разделенная на квадратные ячейки. Тогда у каждой внутренней ячейки имеется 8 соседей. Судьба жизни в ячейке также зависит от состояния соседних клеток. Но теперь правила эволюции такие: если клетка живая и в ее окружении более трех живых клеток, то она погибает от перенаселения; если же живых соседей меньше двух, то она погибает от одиночества. В пустой клетке в следующем поколении зарождается жизнь, если у нее есть ровно три живых соседа.

Попробуйте самостоятельно получить имитационную модель для этой задачи в среде электронной таблицы. Последовательность действий будет аналогичной рассмотренной в примере. По-прежнему для перехода к новому поколению нужно использовать метод копирования диапазона. Но только теперь придется копировать не линейный диапазон, а прямоугольный.

Коротко о главном

Рассмотренная имитационная модель эволюционного типа позволяет проследить за изменениями в расселении живых организмов со сменой поколений.

Удобство применения электронной таблицы для имитационного моделирования заключается в простоте реализации вычислительного алгоритма и наглядности представления результатов.

Вопросы и задания

1. Для каких задач используется метод компьютерного имитационного моделирования?
2. В чем отличие эволюционной задачи, решавшейся методом математического моделирования в § 23, от задачи, решавшейся в данном параграфе методом имитационного моделирования?
3. Проведите вычислительный эксперимент на линейной имитационной модели для различных вариантов исходного расселе-

ния организмов. Ответьте на вопросы:

1) Возможно ли такое расселение, при котором все организмы в конце концов вымрут?

2) Не ведет ли любое расселение в конечном итоге к одной и той же последовательности поколений?

3) Что меняется с изменением размера жизненного пространства?

4*. Постройте в электронной таблице двумерную модель Дж. Конуэя «Жизнь». Проведите вычислительный эксперимент с разными вариантами первоначального расселения организмов.

5*. Решая предыдущую задачу, попробуйте найти такие первоначальные расселения, которые:

1) обречены на гибель;

2) не меняются со сменой поколений;

3) ведут к периодической смене повторяющихся конфигураций расселения.

Чему вы должны научиться, изучив главу 4

Освоить один из табличных процессоров (ТП), имеющихся в компьютерном классе.

Входить в программу, открывать файл с готовой электронной таблицей (ЭТ), сохранять ЭТ, выходить из программы.

Менять режимы отображения информации в ЭТ.

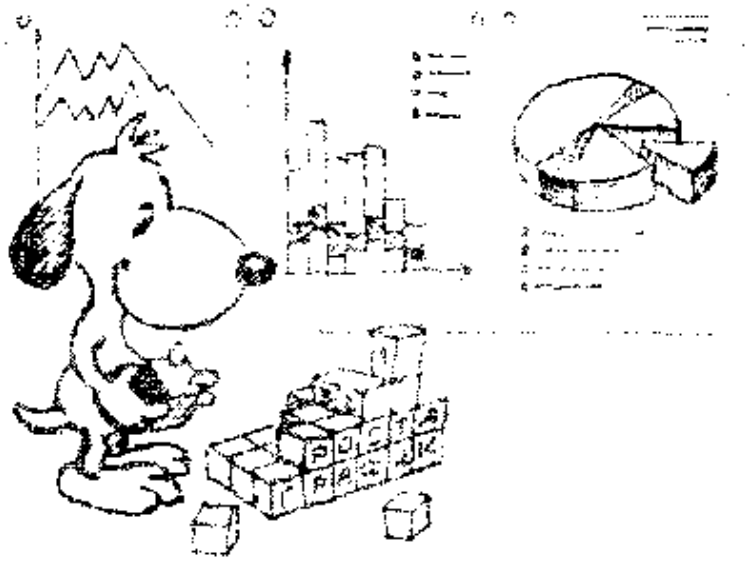
Перемещать табличный курсор, редактировать содержимое ячеек ЭТ.

Вводить в ячейки таблицы тексты, числа, формулы.

Выполнять основные операции манипулирования с диапазонами ЭТ: копирование, удаление, вставку, сортировку.

Получать диаграммы с помощью графических средств ТП.

Создавать собственную ЭТ для несложных табличных расчетов.



ТЕХНОЛОГИЯ ТАБЛИЧНЫХ

Информационная структура электронной таблицы (ЭТ)

- **Строка** — Идентификация: номер строки (1, 2, …, 65536)
- **Столбец** — Идентификация: буквенное имя по алфавиту (A, B,.., Z, AA, AB,.., IV)
- **Блок** — Идентификация: верх. лев. ячейка, ниж. прав. ячейка (A1:F10, B7:S34)
- **Ячейка**
  - Идентификация: столбец_строка (A1, B2, …, IV32767)
  - **Данные в ячейке**
    - **Текст** — Последовательность символов, не являющаяся числом или формулой
    - **Число**
      - Обычная форма. Примеры: 12 –321 24,45
      - Экспоненциальная форма. Примеры: 2e5 0,5e–7
    - **Формула**
      - Операции: +, -, *, /, ^
      - Функции
        - **Математические**: КОРЕНЬ, SIN, COS, LOG...
        - **Условная** ЕСЛИ(….)
        - **Логические** И, ИЛИ, НЕ
        - **Статистические**: СУММ, СРЗНАЧ, МАКС, МИН

Система основных понятий главы 4

РАСЧЕТОВ

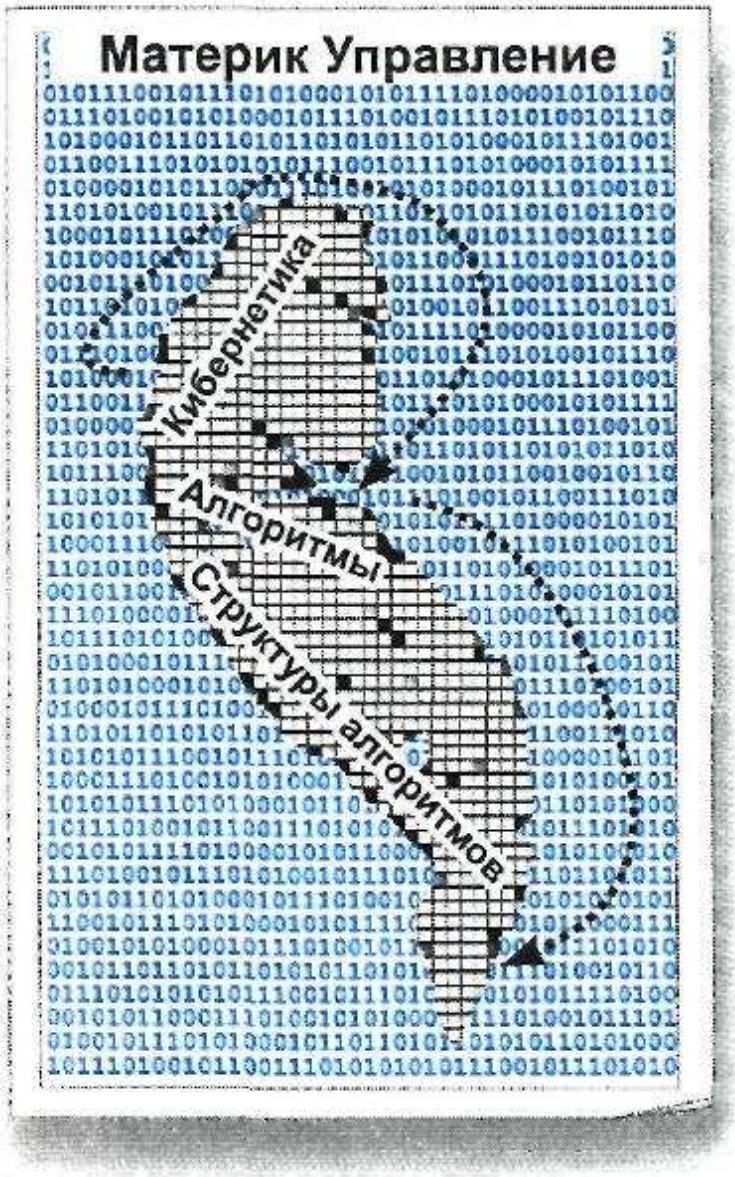
Обработка данных в ЭТ

Табличный процессор
ПО для работы с электронными таблицами

- Режимы отображения
  - Отображение значений
  - Отображение формул
- Пересчет формул
  - С изменением значений операндов формула мгновенно пересчитывается
- Принцип относительной адресации
  - Ссылки на ячейки в формулах определены относительно месторасположения формулы
- Деловая графика
  - Возможность построения диаграмм и графиков по табличным данным
- Команды манипулирования диапазонами
  - Копирование фрагментов; удаление строк и столбцов; *сортировка строк по столбцу*

Глава 5

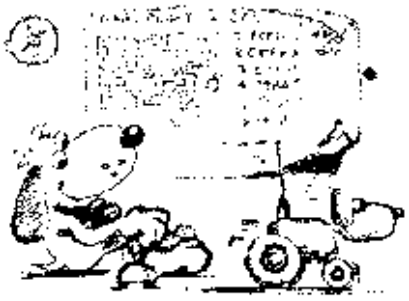
Управление и алгоритмы



Здесь вы узнаете

- что за наука кибернетика
- какие законы управления открыты кибернетикой
- что такое алгоритм управления
- какие бывают алгоритмы и как они описываются

§ 25 Управление и кибернетика



Основные темы параграфа:
- *возникновение кибернетики;*
- *что такое управление;*
- *алгоритм управления.*

Вы уже знакомы с некоторыми областями использования компьютеров. Знаете, что с помощью компьютера можно печатать книги, выполнять чертежи и рисунки; быстро передавать информацию на большие расстояния, создавать компьютерные справочники на любую тему; производить расчеты. Существует еще одно важное приложение компьютерной техники — использование компьютеров для управления.

Возникновение кибернетики

В 1948 году в США и Европе вышла книга американского математика **Норберта Винера** «Кибернетика, или управление и связь в животном и машине». Эта книга провозгласила рождение новой науки — кибернетики.

Не случайно время появления этого научного направления совпало с созданием первых ЭВМ. Н. Винер (рис. 5.1) предвидел, что использование ЭВМ для управления станет одним из важнейших их приложений, а для этого потребуется глубокий теоретический анализ самого процесса управления. Этому и посвящена наука кибернетика.

Рис. 5.1
Норберт Винер

Что такое управление

Управление есть целенаправленное воздействие одних объектов, которые являются управляющими, на другие объекты — управляемые.

Простейшая ситуация — два объекта: один — управляющий, второй — управляемый. Например: человек и телевизор, хозяин и собака, светофор и автомобиль. В первом приближении взаимодействие между такими объектами можно описать схемой, изображенной на рис. 5.2.

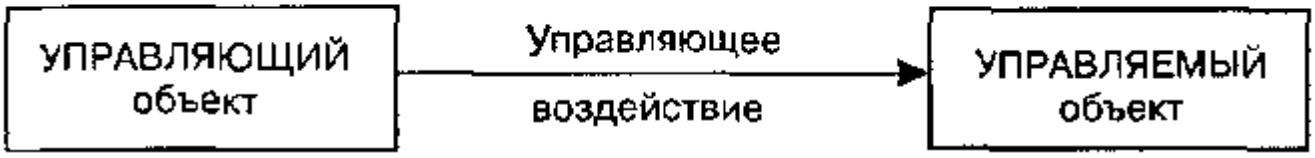


Рис. 5.2. Схема системы управления без обратной связи

В приведенных примерах управляющее воздействие производится в разных формах: человек нажимает клавишу или поворачивает ручку управления телевизором; хозяин голосом подает команду собаке; светофор разными цветами управляет движением автомобилей и пешеходов на перекрестке.

С кибернетической точки зрения все варианты управляющих воздействий следует *рассматривать как управляющую информацию, передаваемую в форме команд.*

В примере с телевизором через пульт управления передаются команды следующего типа: «включить/выключить», «переключить канал», «увеличить/уменьшить громкость». Хозяин передает собаке команды голосом: «Сидеть!», «Лежать!», «Взять!». Световые сигналы светофора шофер воспринимает как команды: красный — «стоять», зеленый — «ехать», желтый — «приготовиться».

Алгоритм управления

В данном выше определении сказано, что управление есть целенаправленный процесс, т. е. команды отдаются не случайным образом, а с вполне определенной целью. В простейшем случае цель может быть достигнута после выполнения одной команды. Для достижения более сложной цели бывает необходимо выполнить последовательность (серию) команд.

Последовательность команд по управлению объектом, выполнение которой приводит к достижению заранее поставленной цели, называется **алгоритмом управления.**

В таком случае объект управления можно назвать *исполнителем управляющего алгоритма*. Значит, в приведенных выше примерах телевизор, собака, автомобиль являются исполнителями управляющих алгоритмов, направленных на вполне конкретные цели (найти интересующую передачу, выполнить определенное задание хозяина, благополучно проехать перекресток).

С точки зрения кибернетики взаимодействие между управляющим и управляемым объектами рассматривается как *информационный процесс*. С этой позиции оказалось, что самые разнообразные процессы управления в природе, технике, обществе происходят сходным образом, подчиняются одним и тем же принципам.

Коротко о главном

Кибернетика — наука об общих свойствах процессов управления в живых и неживых системах.

Управление — это целенаправленное воздействие управляющего объекта на объект управления.

С точки зрения кибернетики управление происходит путем информационного взаимодействия между объектом управления и управляющим объектом.

Последовательность управляющих команд определяется алгоритмом управления, а исполнителем этого алгоритма является объект управления.

Вопросы и задания

1. Кто был основателем кибернетики? В каком году вышла первая книга по кибернетике?
2. Что такое управление?
3. Что представляет собой управляющее воздействие с точки зрения кибернетики?
4. Что такое алгоритм управления?

5. Определите, кто играет роль управляющего и кто (или что) играет роль объекта управления в следующих системах: школа, класс, самолет, стая волков, стадо коров.
6. Для систем управления, выявленных в предыдущей задаче, назовите некоторые команды управления и скажите, в какой форме они отдаются.

§ 26 Управление с обратной связью

Основные темы параграфа:
- *линейный алгоритм;*
- *обратная связь;*
- *модель управления с обратной связью;*
- *циклы и ветвления в алгоритмах;*
- *системы с программным управлением.*

Линейный алгоритм

Если внимательно обдумать рассмотренные в предыдущем параграфе примеры, то можно прийти к выводу, что строго в соответствии со схемой на рис. 5.2 работает только система «светофор — автомобили». Светофор «не глядя» управляет движением машин, не обращая внимания на обстановку на перекрестке. Вот алгоритм работы светофора:

КРАСНЫЙ–ЗЕЛЕНЫЙ–ЖЕЛТЫЙ–КРАСНЫЙ–
ЗЕЛЕНЫЙ–ЖЕЛТЫЙ–КРАСНЫЙ и т. д.

Такой алгоритм называется ***линейным*** или ***последовательным***.

Обратная связь

Совсем иначе протекает процесс управления телевизором или собакой. Прежде чем отдать очередную команду, человек смотрит на состояние объекта управления, на результат выполнения предыдущей команды. Если он не нашел нужную передачу на данном канале, то он переключит телевизор на следующий канал; если собака не выполнила команду «лежать!», хозяин повторит эту команду.

Из этих примеров можно сделать вывод, что управление происходит эффективнее, если управляющий не только отдает команды, т. е. работает ***прямая связь***, но и принимает информацию от объекта управления о его состоянии. Этот процесс называется ***обратной связью.***

Обратная связь — это процесс передачи информации о состоянии объекта управления управляющему объекту.

Модель управления с обратной связью

Управлению с обратной связью соответствует схема, изображенная на рис. 5.3.

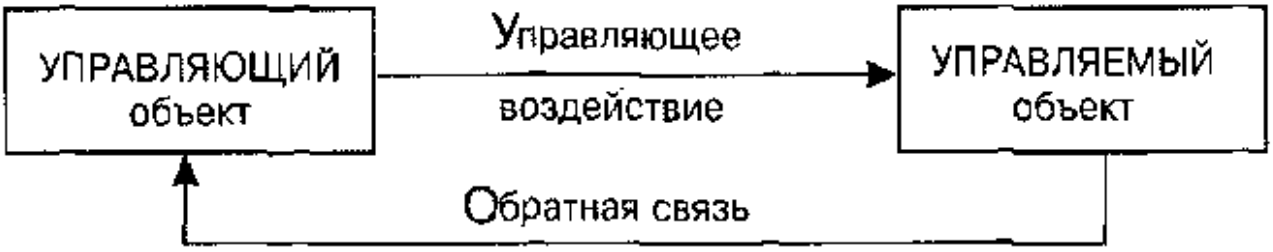


Рис. 5.3. Схема системы управления с обратной связью

Циклы и ветвления в алгоритмах

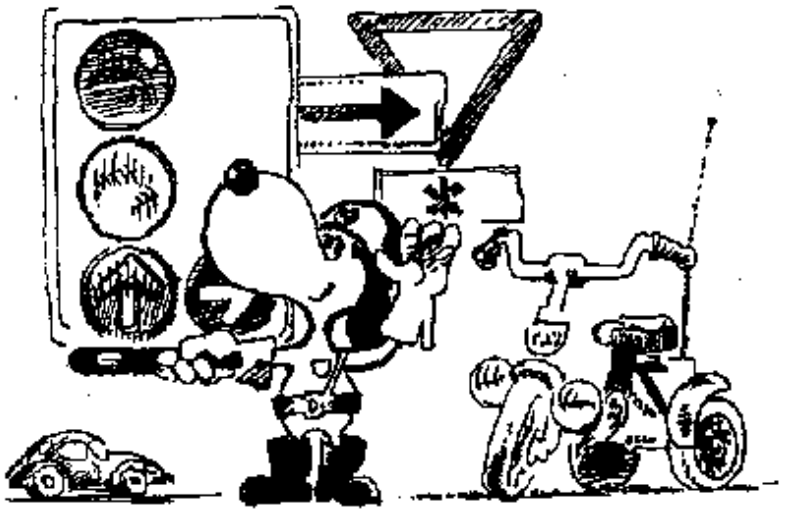
Вот как можно записать алгоритм поиска нужной передачи по телевизору:

ВКЛЮЧИТЬ ТЕЛЕВИЗОР НА 1-М КАНАЛЕ
ПОКА НЕ БУДЕТ НАЙДЕНА ИСКОМАЯ ПЕРЕДАЧА,
ПОВТОРЯТЬ:
    ПЕРЕКЛЮЧИТЬ ТЕЛЕВИЗОР НА СЛЕДУЮЩИЙ КАНАЛ

В этом алгоритме содержится указание на повторение одних и тех же действий (переключить канал) по некоторому условию (пока не найдем передачу). Такой алгоритм называется ***циклическим.***

Если вместо светофора на перекрестке дорог работает милиционер-регулировщик, то управление движением станет более рациональным. Регулировщик следит за скоплением машин на пересекающихся дорогах и дает «зеленую улицу»

в том направлении, в котором в данный момент это нужнее. Нередко из-за «безмозглого» управления светофора на дорогах возникают «пробки». И тут непременно приходит на помощь регулировщик.



Назовем пересекающиеся дороги: Дорога-1 и Дорога-2. Логика управления движением описывается следующим алгоритмом:

ЕСЛИ НА ДОРОГЕ-1 СКОПИЛОСЬ БОЛЬШЕ МАШИН
ТО ОТКРЫТЬ ДВИЖЕНИЕ ПО ДОРОГЕ-1
ИНАЧЕ ОТКРЫТЬ ДВИЖЕНИЕ ПО ДОРОГЕ-2

Здесь по определенному условию происходит выбор одного из двух действий. Такой алгоритм называется ***ветвящимся***. Проверка выполнения условия и в первом и во втором примере стала возможна благодаря обратной связи: телезритель наблюдает за состоянием телевизора, милиционер наблюдает за состоянием движения на дорогах.

В варианте управления *без обратной связи* алгоритм может представлять собой только *однозначную (линейную) последовательность команд*. При наличии обратной связи и «интеллектуального» управляющего объекта алгоритмы управления могут иметь сложную структуру, содержащую альтернативные команды (ветвления) и повторяющиеся команды (циклы).

При наличии обратной связи алгоритм может быть более гибким, допускающим проверку условий, ветвления и циклы.

Системы с программным управлением

Принцип управления с обратной связью и есть основной закон, открытый кибернетической наукой. Он действует в системах самой разной природы: технических, биологических, социальных.

Системы, в которых роль управляющего объекта поручается компьютеру, называются автоматическими системами с программным управлением.

Для функционирования такой системы, во-первых, между компьютером и объектом управления должна быть обеспечена прямая и обратная связь, во-вторых, в память компьютера должна быть заложена программа управления (алгоритм, записанный на языке программирования). Поэтому такой способ управления называют ***программным управлением.***

Программное управление широко используется в технических системах: автопилот в самолете, автоматическая линия на заводе, ускоритель элементарных частиц в физической лаборатории, атомный реактор на электростанции и пр.

Коротко о главном

Управляющая информация передается по линии прямой связи в виде команд управления; по линии обратной связи передается информация о состоянии объекта управления.

Без учета обратной связи алгоритм управления может быть только линейным, при наличии обратной связи алгоритм может иметь сложную структуру, содержащую ветвления и циклы.

Системы, в которых роль управляющего объекта выполняет компьютер, называются автоматическими системами с программным управлением.

? *Вопросы и задания*

1. Что такое обратная связь в процессе управления?
2. Какую структуру имеет управляющий алгоритм в системе без обратной связи?

3. Какую структуру может иметь управляющий алгоритм при наличии обратной связи?
4. Что такое система с программным управлением?
5. Проанализируйте систему «учитель—класс» как систему управления. Кто здесь управляющий объект, кто — объект управления? Какие действуют механизмы прямой и обратной связи?
6. Придумайте ситуации на уроке, когда учитель использует ветвление или цикл, принимая управляющие решения.
7. Назовите систему, в которой учитель является объектом управления. Проанализируйте ее.
8. Опишите систему обучения, в которой роль учителя выполняет компьютер. Какие механизмы прямой и обратной связи действуют в такой системе? В чем преимущества и в чем недостатки компьютерного обучения по сравнению с традиционным?

§ 27 Определение и свойства алгоритма

Основные темы параграфа:

- *происхождение понятия «алгоритм»;*
- *исполнитель алгоритма;*
- *алгоритмический язык;*
- *свойства алгоритма;*
- *определение алгоритма;*
- *формальное исполнение алгоритма;*
- *что такое программа.*

Понятие алгоритма так же фундаментально для информатики, как и понятие информации. Поэтому в нем очень важно как следует разобраться.

Происхождение понятия «алгоритм»

Само слово «алгоритм» происходит от имени выдающегося математика средневекового Востока Мухаммеда аль-Хорезми (787–850). Им были предложены приемы выполнения арифметических вычислений с многозначными числами (вам они хорошо знакомы из школьной математики). Позже в Европе эти приемы назвали алгоритмами, от Algorithmi — латинского написания имени аль-Хорезми. В наше время понятие алгоритма понимается шире, не ограничивается только арифметическими вычислениями.

Исполнитель алгоритма

Из предыдущего параграфа вы узнали, что алгоритм — это последовательность команд управления каким-либо объектом. Мы назвали его объектом управления или ***исполнителем алгоритма***. Им может быть как техническое устройство, так и живое существо.

Рассмотрим исполнителя-человека. Для него можно сформулировать множество алгоритмов, например алгоритмы арифметических вычислений. С таким же успехом можно назвать алгоритмами множество различных инструкций, предписывающих последовательность действий человека для выполнения какой-либо работы. Например, кулинарный рецепт — это алгоритм работы повара с целью приготовления блюда; инструкция по сборке машинки из деталей детского конструктора — алгоритм для ребенка; инструкция по использованию кухонного комбайна — алгоритм для домохозяйки.

Вы, наверное, никогда не задумывались над тем, какое количество алгоритмов вам известно. Жизненный опыт человека растет с увеличением числа освоенных им алгоритмов. Например, чтобы ребенок научился покупать в магазине хлеб, ему нужно сначала рассказать (а лучше показать), как это делается. Освоив «алгоритм покупки хлеба», он в дальнейшем будет успешно выполнять эту работу.

Поиск выигрышной тактики, а следовательно, и алгоритма несложной игры — интересная и полезная задача. Рассмотрим одну из таких игр, которая называется игрой Баше.

Играют двое. Перед ними 21 предмет, допустим, камни (также может быть 11, 16, 26 и т. д.). Игроки берут камни по очереди. За один ход можно взять 1, 2, 3, 4 камня. Проигрывает тот, кто забирает последний камень.

Имеется выигрышная тактика для игрока, берущего камни вторым. Она заключается в том, чтобы брать такое количество камней, которое дополняет число камней, взятых соперником на предыдущем ходе, до пяти. Этот алгоритм можно описать в виде последовательности команд:

алг Игра Баше
нач
1. Предоставить ход сопернику.
2. Взять столько камней, чтобы в сумме с предыдущим ходом соперника получилось 5.
3. Если остался один камень, то объявить о своем выигрыше, иначе вернуться к выполнению команды 1.

кон

Игрок, строго следующий этому алгоритму, будет всегда выигрывать, даже если он не понимает, почему так происходит.

Алгоритмический язык

В приведенном примере используется символика учебного ***Алгоритмического языка*** (***АЯ***).

Из примера видно, что при записи алгоритма на АЯ в начале пишется заголовок, начинающийся со ***служебного слова*** **алг** (сокращенное слово «алгоритм»). Затем указывается название алгоритма, которое составитель алгоритма придумывает сам. Следующая часть называется телом алгоритма. Она начинается со служебного слова **нач** (начало) и заканчивается словом **кон** (конец). Тело алгоритма представляет собой последовательность команд для исполнителя.

Здесь и в дальнейшем служебные слова в алгоритмах на алгоритмическом языке будут записываться жирным шрифтом. В языках программирования (как и в АЯ) служебными называются слова, которые всегда употребляются в одном и том же смысле.

Свойства алгоритма

Процесс решения задачи должен быть разбит на последовательность отдельно выполняемых шагов.

Это свойство алгоритма называется ***дискретностью***.

Всякий алгоритм составляется в расчете на конкретного исполнителя с учетом его возможностей. Для того чтобы алгоритм был выполним, нельзя включать в него команды, которые исполнитель не в состоянии выполнить. Нельзя повару поручать работу токаря, какая бы подробная инструкция ему ни давалась. У каждого исполнителя имеется свой перечень команд, которые он может исполнить. Такой перечень называется ***системой команд исполнителя алгоритмов*** (***СКИ***).

Алгоритм, составленный для конкретного исполнителя, должен включать только те команды, которые входят в систему команд исполнителя.

Это свойство алгоритма называется ***понятностью***.

Каждая команда алгоритма должна определять однозначное действие исполнителя.

Это свойство алгоритма называется ***точностью***.

Алгоритм не должен быть рассчитан на принятие каких-либо самостоятельных решений исполнителем, не предусмотренных составителем алгоритма.

Еще одно важное требование, предъявляемое к алгоритму — это свойство ***конечности*** (иногда говорят — ***результативности***) алгоритма. Это значит, что:

Исполнение алгоритма должно завершиться за конечное число шагов.

Для успешного выполнения любой работы мало иметь ее алгоритм. Всегда требуются еще какие-то ***исходные данные***, с которыми будет работать исполнитель (продукты для приготовления блюда, детали для сбора технического устройства и т. п.). Исполнителю, решающему математическую задачу, требуется исходная числовая информация. Задача всегда формулируется так: *дана исходная информация, требуется получить какой-то результат*. В математике вы привыкли в таком виде записывать условия задач. Например:

Дано: катеты прямоугольного
треугольника $a = 3$ см; $b = 4$ см.
Найти: гипотенузу c

Алгоритм решения этой задачи можно представить в таком виде:

алг Гипотенуза
нач
1. Возвести a в квадрат.
2. Возвести b в квадрат.
3. Сложить результаты действий 1 и 2.
4. Вычислить квадратный корень результата действия 3 и принять его за значение c.

кон.

Каждую из этих команд может выполнить любой человек, знающий основы математики, следовательно, они входят в его систему команд.

Еще пример: для поиска номера телефона нужного вам человека исходными данными являются: фамилия, инициалы человека и телефонная книга (точнее, информация, заключенная в телефонную книгу). Однако этого может оказаться недостаточно. Например, вы ищете номер телефона А. И. Смирнова и обнаруживаете, что в книге пять строк с фамилией «Смирнов А. И». Ваши исходные данные оказались *неполными* для точного решения задачи (вместо одного номера телефона вы получили пять). Оказалось, что нужно знать еще домашний адрес.

Набор: «фамилия — инициалы — телефонный справочник — адрес» является *полным набором данных* в этой ситуации.

Только имея полный набор данных, можно точно решить задачу.

Если исходные данные неполные, то задачу либо совсем нельзя решить (ничего нельзя узнать про гипотенузу по одному катету), либо получается неоднозначное решение (пять номеров телефонов).

В задачах управления физическими объектами (автомобиль, самолет, станок и т. п.) исходными данными является информация о состоянии объекта управления, об обстановке, его окружающей.

Определение алгоритма

Обобщая все сказанное, сформулируем определение алгоритма.

Алгоритм — *понятное и точное предписание* исполнителю выполнить конечную последовательность команд, приводящую от исходных данных к искомому результату.

Формальное исполнение алгоритма

Если алгоритм обладает перечисленными выше свойствами, то работа по нему будет производиться исполнителем

формально (то есть без всяких элементов творчества с его стороны). На этом основана работа программно управляемых исполнителей-автоматов, например промышленных роботов. Робот-манипулятор может выполнять работу токаря, если он умеет выполнять все операции токаря (включать станок, закреплять резец, перемещать резец, замерять изделие). От исполнителя не требуется понимания сущности алгоритма, он должен лишь точно выполнять команды, не нарушая их последовательности.

Что такое программа

А что такое программа? Отличается ли чем-то программа от алгоритма?

Программа — это алгоритм, записанный на языке исполнителя.

Иначе можно сказать так: алгоритм и программа не отличаются по содержанию, но могут отличаться по форме.

Для алгоритма строго не определяется форма его представления. Алгоритм можно изобразить графически, можно — словесно, можно — какими-нибудь специальными значками, понятными только его автору. Но программа должна быть записана на языке исполнителя.

Коротко о главном

Слово «алгоритм» происходит от имени Мухаммеда аль-Хорезми, первым предложившего приемы выполнения арифметических операций с многозначными числами.

Исполнитель алгоритма — это тот объект, для управления которым составлен алгоритм.

Процесс решения задачи должен быть разбит на последовательность отдельных шагов (свойство дискретности алгоритма).

Система команд исполнителя (СКИ) — это вся совокупность команд, которые исполнитель умеет выполнять (понимает). Алгоритм можно строить только из команд, входящих в СКИ исполнителя (свойство понятности).

Каждая команда алгоритма управления определяет однозначное действие исполнителя (свойство точности).

Выполнение алгоритма должно приводить к результату за конечное число шагов (свойство конечности).

Для успешного выполнения работы, решения задачи необходимо сообщить (передать) исполнителю полный набор исходных данных.

Выполнение алгоритма исполнителем производится формально.

Программа от алгоритма может отличаться по форме, но не по содержанию. Программа — это алгоритм, представленный на языке исполнителя.

? *Вопросы и задания*

1. Что такое алгоритм? Откуда произошло это слово?
2. Что такое исполнитель алгоритма?
3. Что такое система команд исполнителя?
4. В чем заключаются основные свойства алгоритма?
5. Назовите исполнителей следующих видов работы: уборка мусора во дворе; перевозка пассажиров; выдача заработной платы; прием экзаменов; сдача экзаменов; обучение детей в школе. Попробуйте сформулировать СКИ для каждого из этих исполнителей.
6. Определите полный набор данных для решения следующих задач обработки информации:
   - вычисление стоимости покупок в магазине;
   - вычисление суммы сдачи от данных вами продавцу денег;
   - определение времени показа по телевизору интересующего вас фильма;
   - вычисление площади треугольника;
   - определение времени падения кирпича с крыши дома;
   - определение месячной платы за расход электроэнергии;
   - перевод русского текста на итальянский язык;
   - перевод итальянского текста на русский язык.
7. Попробуйте сформулировать алгоритмы обработки информации для заданий из п. 6, если исполнителем являетесь вы сами. *Какие команды при этом вы должны уметь выполнять?*

§ 28 Графический учебный исполнитель

Основные темы параграфа:

- *назначение и возможности графического исполнителя (ГРИС);*
- *простые команды ГРИС;*
- *работа в программном режиме;*
- *линейные программы для ГРИС.*

Назначение и возможности графического исполнителя (ГРИС);

Учебные исполнители используются для обучения составлению управляющих алгоритмов.

Есть много учебных исполнителей, придуманных для занятий по информатике. У них разные, часто забавные названия: Черепашка, Робот, Чертежник, Кенгуренок, Пылесосик, Муравей, Кукарача и другие. Одни исполнители создают рисунки на экране компьютера, другие складывают слова из кубиков с буквами, третьи перетаскивают предметы из одного места в другое. Все эти исполнители управляются программным путем. Любому из них свойственна определенная ***среда деятельности, система команд управления, режимы работы.***

В предыдущих главах мы избегали детальных описаний работы с конкретными вариантами программ (редакторов, СУБД и пр.). И в этой главе мы не будем детально описывать работу с каким-то реальным исполнителем из вышеперечисленных (в компьютерных классах разных школ может быть разное программное обеспечение). Мы опишем условного исполнителя, который очень похож на некоторых из существующих в главном: системой команд, языком и приемами программирования.

Многие из учебных исполнителей занимаются рисованием на экране компьютера. Из названных выше это Черепашка, Кенгуренок, Чертежник. Эту группу можно назвать графическими исполнителями. Пусть наш гипотетический (придуманный) исполнитель тоже будет из этой «компании». Назовем его ГРИС, что значит «ГРафический ИСполнитель».

Что умеет делать ГРИС? Он может перемещаться по полю и своим хвостом рисовать на этом поле (предположим, что у него есть хвост, к которому привязан кусочек мела).

Обстановка, в которой действует исполнитель, называется *средой исполнителя*. Среда графического исполнителя показана на рис. 5.4. Это лист (страница экрана) для рисования. ГРИС может перемещаться в горизонтальном и вертикальном направлениях с постоянным шагом. На рис. 5.4 пунктиром показана сетка с шагом, равным шагу исполнителя. Исполнитель может двигаться только по линиям этой сетки. ГРИС не может выходить за границы поля.

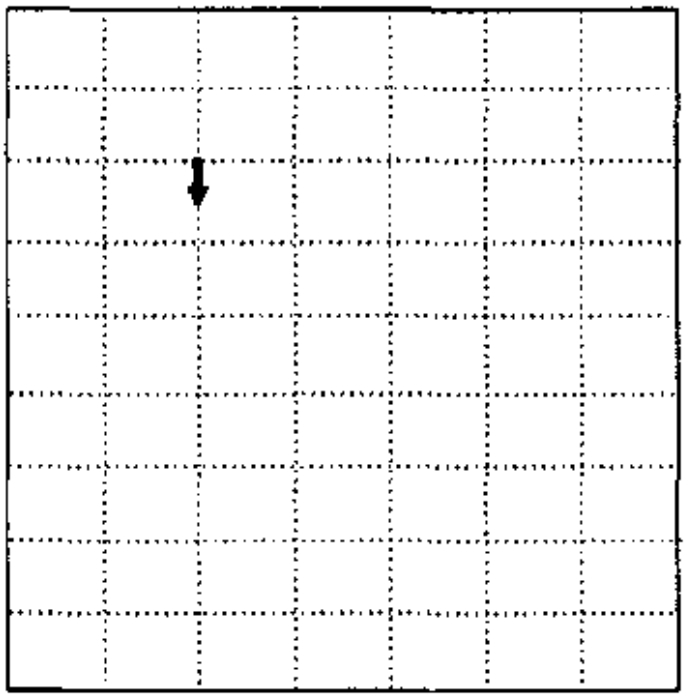


Рис. 5.4. Среда графического исполнителя. Стрелка указывает состояние исполнителя (местоположение и направление)

Состояние исполнителя на поле определяется, во-первых, его местоположением (в какой точке поля он находится), во-вторых, направлением (куда он смотрит). Направление будем определять, как на географической карте: вверх — на север, вниз — на юг, влево — на запад, вправо — на восток. ГРИС может шагать или прыгать по линиям сетки, а также поворачиваться. Поворачиваться он умеет только против часовой стрелки.

Графический исполнитель — это объект управления. А управлять им будем мы с вами. Целью управления является получение определенного рисунка. Понятно, что этот рисунок может состоять только из горизонтальных и вертикальных отрезков, в других направлениях ГРИС двигаться не умеет.

Задача обычно ставится так: исполнитель находится в данной точке поля, смотрит в данном направлении. Требуется получить определенный рисунок. Например: ГРИС находится в середине поля и смотрит на восток. Надо нарисовать букву «Т» с длиной каждой линии, равной четырем шагам.

Первоначально исполнителю придается исходное состояние. Это делается в специальном *режиме установки*.

Теперь перейдем к управлению графическим исполнителем. Здесь возможны два режима: ***режим прямого управления*** и ***режим программного управления***.

Простые команды ГРИС

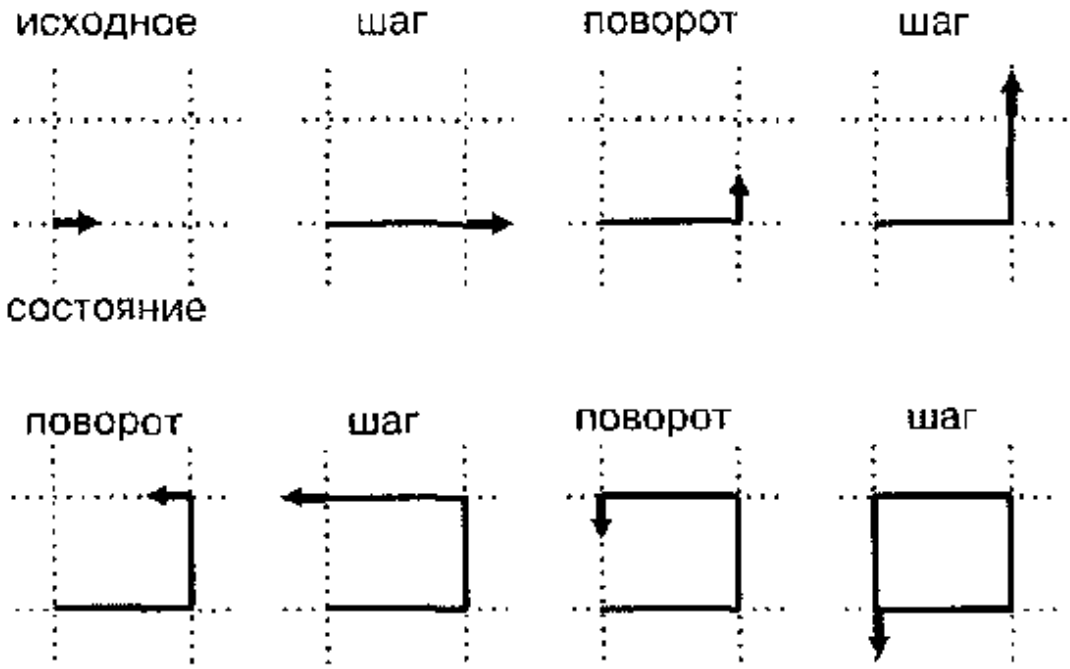
Работа в режиме прямого управления происходит так: человек отдает команду, ГРИС ее выполняет; затем отдается следующая команда и т. д. (как в примере с хозяином и собакой).

В режиме прямого управления система команд исполнителя следующая:

шаг	—	перемещение ГРИС на один шаг вперед с рисованием линии;
поворот	—	поворот на 90° против часовой стрелки;
прыжок	—	перемещение на один шаг вперед без рисования линии.

Эти команды будем называть ***простыми командами***.

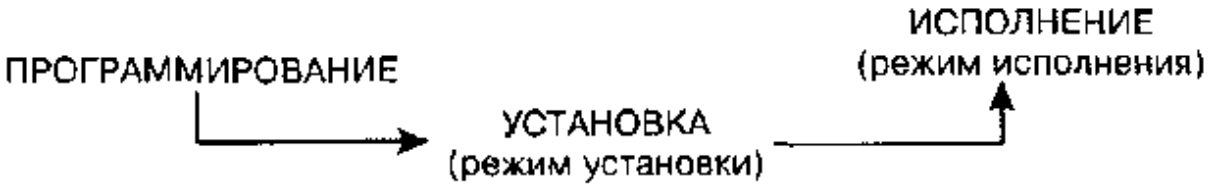
Например, пусть требуется нарисовать квадрат со стороной, равной одному шагу. Исходное положение ГРИС — в левом нижнем углу квадрата, направление — на восток. Будем отмечать состояние исполнителя маленькой стрелкой. Тогда последовательность команд и результаты их выполнения будут следующими:



Работа в программном режиме

Работа в программном режиме имитирует автоматическое управление исполнителем. Управляющая система (компьютер) обладает памятью, в которую заносится программа. *Человек составляет программу и вводит ее в память.* Затем ГРИС переводится в режим установки и человек вручную (с помощью определенных клавиш) устанавливает исходное состояние исполнителя. После этого производится переход в ***режим исполнения*** и ГРИС начинает работать по программе. Если возникает ситуация, при которой он не может выполнить очередную команду (выход за границу поля), то выполнение программы завершается аварийно. Если аварии не происходит, то работа исполнителя заканчивается на последней команде.

Таким образом, программное управление графическим исполнителем проходит этап подготовки (программирование и установка исходного состояния) и этап исполнения программы.



В режиме программного управления по-прежнему используются команды **шаг**, **поворот**, **прыжок**. Однако в этом режиме есть еще и другие команды. С ними вы познакомитесь позже.

Язык программирования для графического исполнителя — это учебный алгоритмический язык (АЯ). Поэтому алгоритмы управления ГРИСом, записанные на АЯ, являются для него одновременно и программами.

Линейные программы для ГРИС

Будем осваивать программирование на примерах решения конкретных задач. С новыми командами СКИ будем знакомиться по мере появления потребности в них.

Задача 1. Составим и выполним программу, по которой ГРИС нарисует на поле букву «Т». Пусть длина вертикального и горизонтального отрезков равна четырем шагам.

Исходное состояние — чистый лист. Исполнитель — в точке, где будет находиться левый конец горизонтального отрезка, направление — на восток.

Результат выполнения программы показан на рис. 5.5. Стрелка указывает конечное состояние исполнителя.

Программа

```
программа Буква Т
 нач
   шаг
   шаг
   шаг
   шаг
   поворот
   поворот
   прыжок
   прыжок
   поворот
   шаг
   шаг
   шаг
   шаг
кон
```

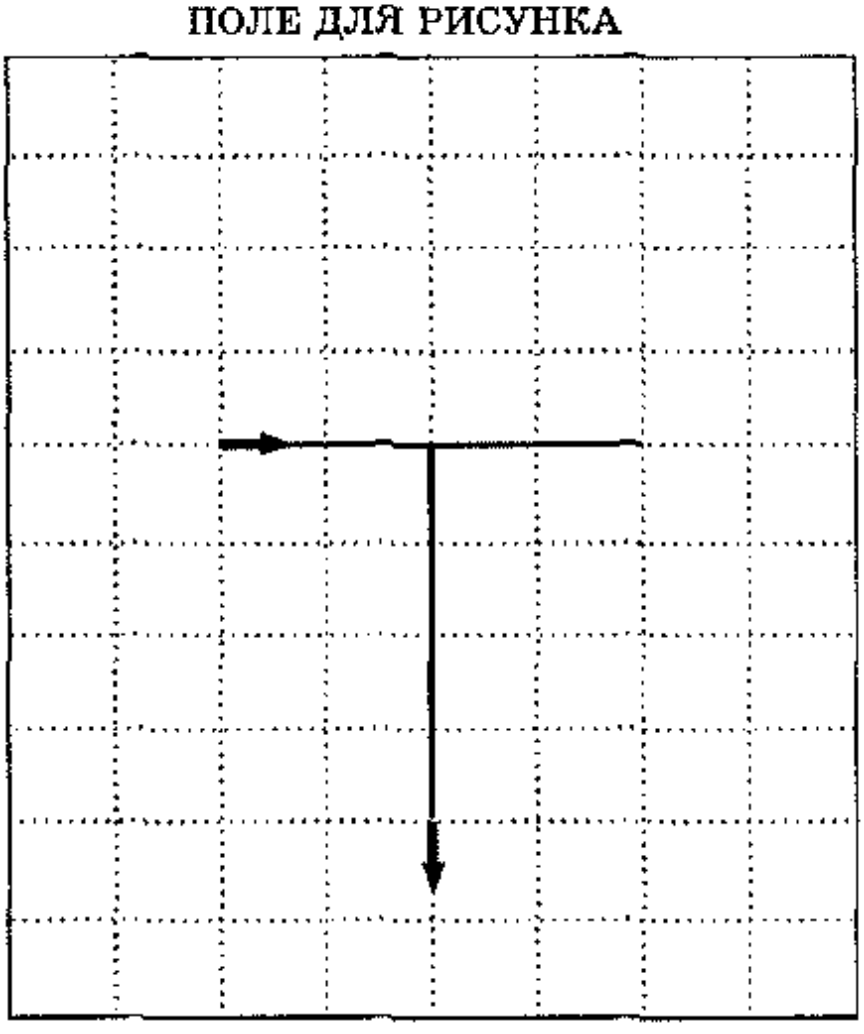


Рис. 5.5. Результат выполнения программы «Буква Т». Стрелками указаны начальное и конечное состояния ГРИС

Структура такой программы (алгоритма) называется линейной. Команды выполняются одна за другой, каждая только один раз.

Для решения этой задачи оказалось достаточно той части СКИ, которая используется в режиме прямого управления.

Коротко о главном

ГРИС — это графический исполнитель, назначение которого — получение чертежей, рисунков на экране дисплея.

Управление ГРИС может происходить в режиме прямого управления или в режиме программного управления.

С помощью команд **шаг**, **поворот**, **прыжок** в пределах рабочего поля можно построить любой рисунок, состоящий из вертикальных и горизонтальных отрезков. Структура управляющего алгоритма при этом будет линейной.

? *Вопросы и задания*

1. Какую работу может выполнять ГРИС?
2. Что представляет собой среда исполнителя ГРИС?
3. В чем разница между управлением в прямом режиме и в программном режиме?
4. Какие простые команды входят в СКИ ГРИС; как они выполняются?
5. В какой последовательности происходит выполнение команд в линейном алгоритме?
6. Может ли данный исполнитель нарисовать: прямоугольник, треугольник, пятиконечную звезду, буквы «Н», «Х», «Р», «М»?
7. Составьте программы рисования символов «Е», «П», «Б», «Ч», «Ц», «Ш», а также других фигур, состоящих из горизонтальных и вертикальных отрезков.

§ 29 Вспомогательные алгоритмы и подпрограммы

Основные темы параграфа:

- *что такое вспомогательный алгоритм;*
- *обращение к вспомогательному алгоритму (процедуре);*
- *описание вспомогательного алгоритма (процедуры);*
- *метод последовательной детализации;*
- *сборочный метод.*

Что такое вспомогательный алгоритм

А сейчас решим следующую задачу.

Задача 2. Пусть требуется составить программу, по которой ГРИС напишет на экране четырехзначное число 1919 (рис. 5.6).

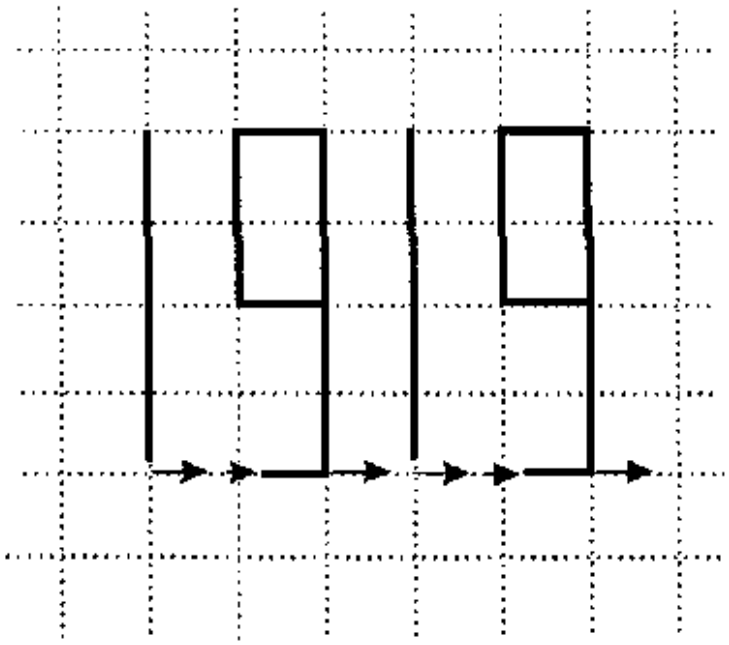


Рис. 5.6. Рисование числа 1919

Конечно, можно поступить так, как в предыдущей задаче, написав одну длинную программу, по которой исполнитель шаг за шагом нарисует эти цифры. Но с очевидностью возникает другая идея: поскольку здесь дважды повторяются цифры 1 и 9, нельзя ли сократить работу, написав программу рисования той и другой цифры только один раз? Это действительно можно сделать.

Алгоритм, по которому решается некоторая подзадача из основной задачи и который, как правило, выполняется многократно, называется **вспомогательным алгоритмом.**

Вспомогательный алгоритм, записанный на языке программирования, называется *подпрограммой* или *процедурой*.

Обращение к вспомогательному алгоритму (процедуре)

В таком случае программа решения поставленной задачи разделяется на основную программу (основной алгоритм) и процедуры (вспомогательные алгоритмы). Каждая процедура должна иметь свое уникальное имя. Для рассматриваемой задачи имена процедур выберем следующими: ЕДИНИЦА и ДЕВЯТЬ. Тогда в основной программе команды обращения к этим процедурам будут такими:

сделай ЕДИНИЦА
сделай ДЕВЯТЬ

По этим командам управление передается соответствующим процедурам, и после их выполнения управление вернется к следующей команде основной программы.

Договоримся, что начальное и конечное состояния ГРИС при вычерчивании каждой цифры будут такими, как показано стрелками на рис. 5.6 (внизу, на восток). У единицы начальное и конечное состояния совпадают. Основная программа:

программа Число 1919
нач
    сделай ЕДИНИЦА
    прыжок
    сделай ДЕВЯТЬ
    прыжок
    сделай ЕДИНИЦА
    прыжок
    сделай ДЕВЯТЬ
кон

Данный пример познакомил вас с новой командой из СКИ графического исполнителя — командой ***обращения к процедуре***. Ее формат, т. е. общий вид, следующий:

сделай <имя процедуры>

Описание вспомогательного алгоритма (процедуры)

Вот и все! Так просто! Но теперь надо «объяснить» исполнителю, что такое ЕДИНИЦА и что такое ДЕВЯТЬ. Это делается в ***описаниях процедур*** (здесь порядок выполнения — по столбцам):

```
процедура ЕДИНИЦА
нач
  поворот
  шаг
  шаг
  шаг
  шаг
  поворот
  поворот
  прыжок
  прыжок
  прыжок
  прыжок
  поворот
кон
```

```
процедура ДЕВЯТЬ
нач
  шаг
  поворот
  шаг
  шаг
  шаг
  шаг
  поворот
  поворот
  шаг
  поворот
  шаг
  шаг
  поворот
  шаг
```

```
  поворот
  поворот
  поворот
  прыжок
  прыжок
  поворот
кон
```

Определение процедуры в программе называется ее описанием. Формат описания процедуры:

```
процедура <имя процедуры>
нач
        <тело процедуры>
кон
```

Имя в описании и имя в обращении должны точно совпадать (никаких склонений по падежам!). Описание процедур располагается после основной программы.

Добавив к программе описание процедуры, мы тем самым расширили систему команд исполнителя. В данной программе стало возможным использование команды обращения к этой процедуре.

Метод последовательной детализации

Использованный нами подход облегчает программирование сложных задач. Задача разбивается на более простые подзадачи. Решение каждой оформляется в виде вспомогательного алгоритма, а основной алгоритм организует связку между ними.

Метод программирования, при котором сначала пишется основная программа, в ней записываются обращения к пока еще не составленным подпрограммам, а потом описываются эти подпрограммы, называется ***методом последовательной (пошаговой) детализации***. Причем количество шагов детализации может быть гораздо бóльшим, чем в нашем примере, поскольку сами подпрограммы могут содержать внутри себя обращения к другим подпрограммам.

Сборочный метод

Возможен и другой подход к построению сложных программ: первоначально составляется множество подпрограмм, которые могут понадобиться при решении задачи, а затем пишется основная программа, содержащая обращения к ним. Подпрограммы могут быть объединены в ***библиотеку подпрограмм*** и сохранены в долговременной памяти компьютера. Такую библиотеку можно постепенно пополнять новыми подпрограммами.

Например, если для управления графическим исполнителем создать библиотеку процедур рисования всех букв и цифр, то программа получения любого текста будет состоять из команд обращения к библиотечным процедурам.

Описанный метод называется ***сборочным программированием***. Часто в литературе по программированию используется такая терминология: метод последовательной детализации называют ***программированием сверху вниз***, а сборочный метод — ***программированием снизу вверх***.

Коротко о главном

Для упрощения программирования сложных задач используются вспомогательные алгоритмы.

Вспомогательный алгоритм — это алгоритм решения некоторой подзадачи из исходной (основной) задачи.

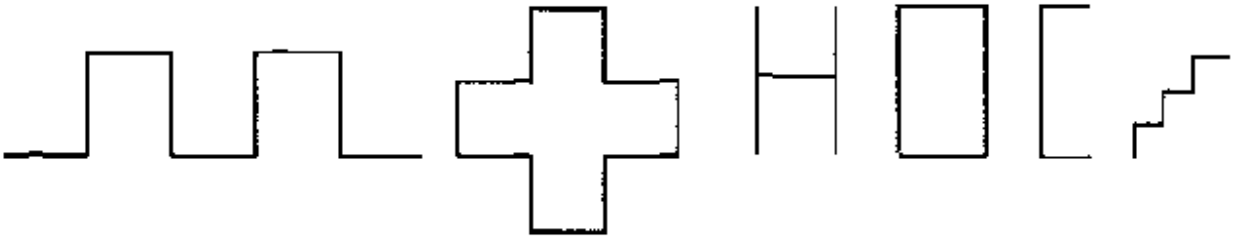
Вспомогательный алгоритм, записанный на языке программирования, называется процедурой.

Вспомогательный алгоритм должен быть описан. После этого в основном алгоритме можно использовать команду обращения к этому вспомогательному алгоритму.

Метод программирования, при котором сначала записывается основной алгоритм, а затем описываются использованные в нем вспомогательные алгоритмы, называется методом последовательной детализации, или программированием сверху вниз. Обратный порядок программирования называется программированием снизу вверх.

? *Вопросы и задания*

1. Что такое основной алгоритм; вспомогательный алгоритм?
2. Чем отличается описание вспомогательного алгоритма от обращения к вспомогательному алгоритму?
3. Каковы правила описания вспомогательных алгоритмов (процедур) для исполнителя ГРИС?
4. Как записывается команда обращения к процедуре в языке исполнителя ГРИС?
5. В чем суть метода последовательной детализации?
6. Что такое программирование снизу вверх; сверху вниз?
7. Используя вспомогательные алгоритмы, запрограммируйте рисование следующих фигур:



§ 30 Циклические алгоритмы

Основные темы параграфа:

- *команда цикла;*
- *цикл в процедуре;*
- *блок-схемы алгоритмов;*
- *цикл с предусловием.*

Команда цикла

Обсудим решение следующей задачи.

Задача 3. Исходное положение: ГРИС — у левого края поля, направление — на восток. Требуется нарисовать горизонтальную линию через весь экран.

Задачу можно решить, написав 15 раз команду **шаг** (если поперек поля рисунка 15 шагов). Но есть и более короткий вариант программы. Вот он:

```
пока впереди не край, повторять
нц
        шаг
кц
```

Здесь использована команда, которая называется *циклом*. Формат команды цикла следующий:

```
пока <условие>, повторять
нц
            <тело цикла>
кц
```

Служебное слово **нц** обозначает начало цикла, **кц** — конец цикла.

Это первая команда из СКИ, которая использует обратную связь между графическим исполнителем и управляющим им компьютером. Она заключается в том, что проверяется, не вышел ли ГРИС на край поля и не грозит ли ему следующий шаг или прыжок в этом направлении аварией. Проверяемые условия звучат так: *«впереди край?»* или *«впереди не край?»*. На что машина получает от исполнителя ответ *«да»* или *«нет»*.

В приведенном примере проверяется условие «впереди не край?». Если «да», то делается шаг (т. е. выполняется <тело цикла>). Затем происходит возврат на проверку условия, и все повторяется. Если проверка условия дает отрицательный результат (т. е. впереди край), то выполнение цикла завершается и исполняется следующая команда программы.

При программировании цикла важно думать о том, чтобы цикл был конечным. Цикл, записанный выше, — конечный. Двигаясь в одном направлении, исполнитель обязательно достигнет края, и на этом выполнение цикла закончится.

Ситуация, при которой выполнение цикла никогда не заканчивается, называется зацикливанием. Пусть ГРИС находится в середине поля. Исполнение следующего цикла:

```
пока впереди не край, повторять
нц
    шаг
    поворот
кц
```

никогда не закончится. ГРИС будет бесконечно рисовать квадратик, так как проверка условия «впереди не край?» всегда будет давать положительный ответ.

Цикл в процедуре

Задача 4. Теперь составим программу, по которой графический исполнитель нарисует прямоугольную рамку по краю поля (рис. 5.7). Исходное положение: ГРИС находится в левом верхнем углу, смотрит на юг.

Рамка состоит из четырех линий, поэтому разумно воспользоваться процедурой, проводящей линию от края до края поля. Опять будем действовать методом последовательной детализации. Напишем сначала основную программу.

```
программа Рамка
нач
    сделай ЛИНИЯ
    поворот
    сделай ЛИНИЯ
    поворот
    сделай ЛИНИЯ
    поворот
    сделай ЛИНИЯ
кон
```

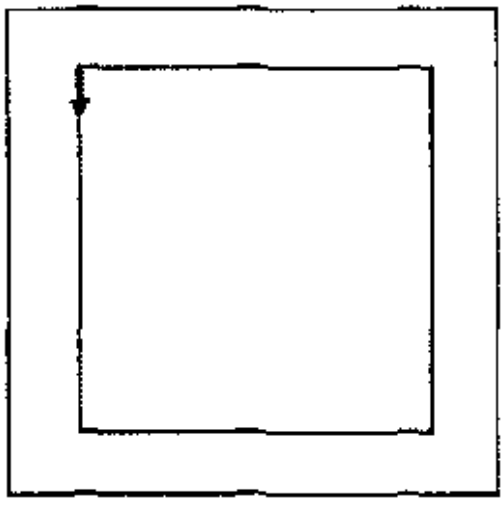


Рис. 5.7. Результат выполнения программы «Рамка». Стрелкой указано исходное положение

Программа проведения линии нами уже рассматривалась. Осталось оформить ее в виде процедуры.

```
процедура ЛИНИЯ
нач
    пока впереди не край, повторять
    нц
        шаг
    кц
кон
```

При составлении этой программы использовалась одношаговая детализация в такой последовательности:

ОСНОВНАЯ ПРОГРАММА
↓ шаг детализации
процедура ЛИНИЯ

Блок-схемы алгоритмов

Начиная с 50-х годов прошлого века, т. е. еще с эпохи ЭВМ первого поколения, программисты стали использовать графические схемы, изображающие алгоритмы, которые получили название блок-схем.

Блок-схема состоит из фигур (блоков), обозначающих указания на отдельные действия исполнителя, и стрелок, соединяющих эти блоки и указывающих на последовательность их выполнения. Внутри каждого блока записывается выполняемое действие. Сама форма блока подсказывает характер операции, которую он обозначает. Для придания наглядности и единообразия схемам алгоритмов все графические элементы стандартизированы.

Посмотрите на рис. 5.8, где показана блок-схема алгоритма рисования рамки. Она состоит из двух частей: блок-схемы основного алгоритма и блок-схемы вспомогательного алгоритма ЛИНИЯ.

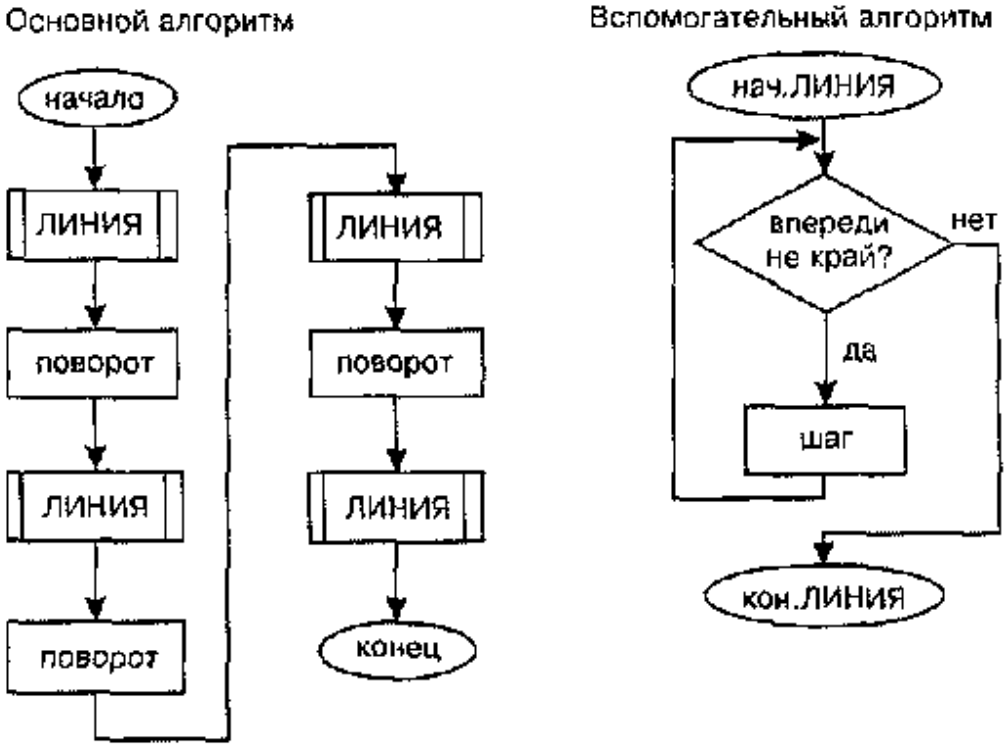


Рис. 5.8. Блок-схема алгоритма «Рамка»

Из этих схем видно назначение блоков различной формы (рис. 5.9).

Рис. 5.9. Элементы блок-схем и структура «цикл»

Цикл с предусловием

Команда цикла изображается не отдельным блоком, а целой структурой, показанной на рис. 5.9. Такую структуру называют ***циклом с предусловием*** (так как условие предшествует телу цикла). Есть и другой вариант названия: ***цикл-пока*** (пока условие истинно, повторяется выполнение тела цикла).

При решении следующей задачи снова будем использовать метод последовательной детализации.

Задача 5. Требуется расчертить экран горизонтальными линиями (рис. 5.10). Исходное состояние исполнителя: верхний левый угол, направление — на юг.

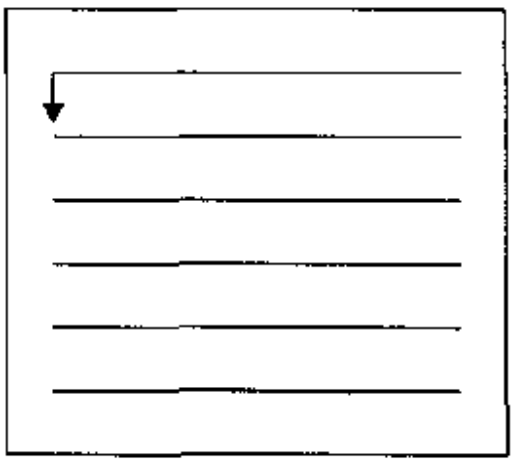


Рис. 5.10. «Разлиновка»

В программе для решения этой задачи используется та же процедура ЛИНИЯ. Другая процедура — ВОЗВРАТ — возвращает ГРИС к исходному положению для рисования следующей линии.

```
программа Разлиновка
нач
  пока впереди не край,
  повторять
  нц
        поворот
        сделай ЛИНИЯ
        сделай ВОЗВРАТ
        прыжок
  кц
  поворот
  сделай ЛИНИЯ
кон
```

```
процедура ВОЗВРАТ
нач
  поворот
  поворот
  пока впереди не край,
  повторять
  нц
        прыжок
  кц
  поворот
кон
```

Блок-схемы основного и вспомогательного алгоритмов представлены на рис. 5.11.

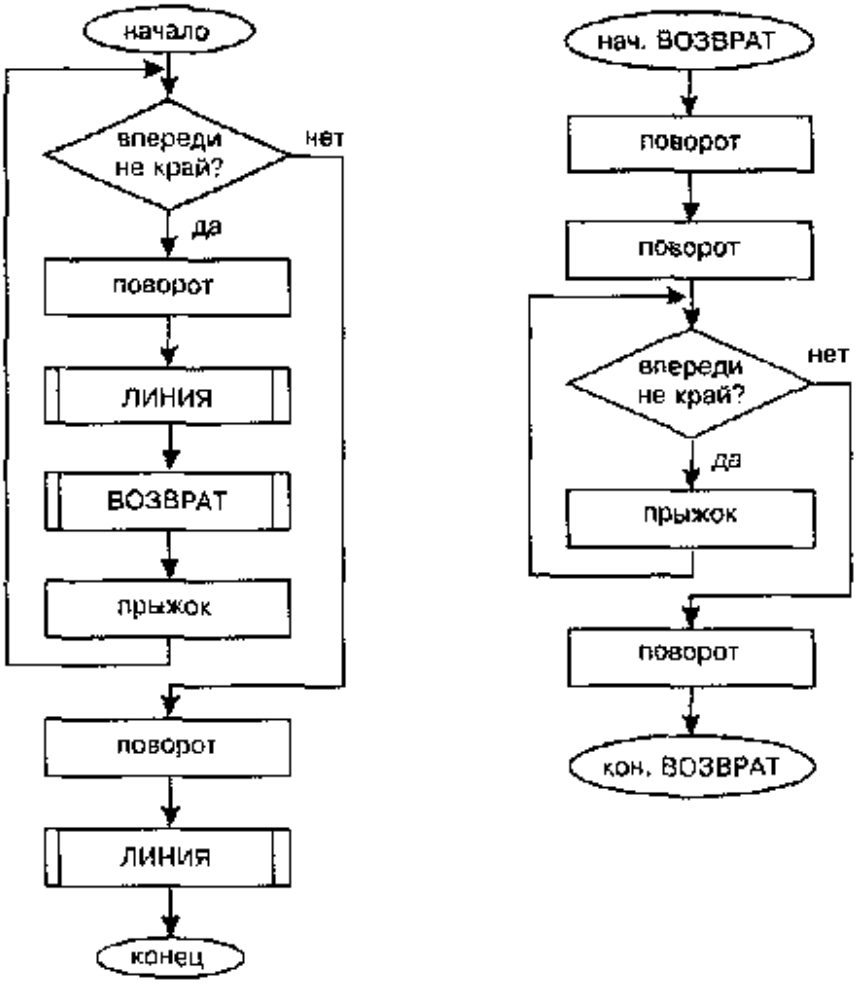


Рис. 5.11. Блок-схема алгоритма «Разлиновка»

Коротко о главном

Для программирования повторяющихся действий применяется команда цикла, которая имеет следующую структуру:

пока <условие>, **повторять**
нц
    <тело цикла>
кц

Команда цикла использует обратную связь между объектом управления и управляющей системой. Проверка условия дает информацию управляющей системе о состоянии объекта управления.

В цикле с предусловием если проверяемое условие выполняется (истинно), то выполняются команды, составляющие тело цикла. Если условие ложно, то происходит выход из цикла.

При программировании цикла необходимо следить за тем, чтобы не допускалось зацикливания.

Блок-схема — это графический способ описания алгоритма. Блоки обозначают действия исполнителя, а соединяющие их стрелки указывают на последовательность выполнения действий.

Вопросы и задания

1. Что такое цикл? Как записывается команда цикла?
2. Что такое условие цикла? Что такое тело цикла?
3. В каком случае происходит зацикливание алгоритма?
4. Что такое блок-схема?
5. Из каких блоков составляются блок-схемы (как они изображаются и что обозначают)?
6. Что обозначают стрелки на блок-схемах?
7. Составьте программу, переводящую ГРИС в угол поля из любого исходного состояния.
8. Составьте программу рисования прямоугольной рамки вдоль края поля, начиная рисование из любого начального состояния исполнителя.

§ 31 Ветвление и последовательная детализация алгоритма

Основные темы параграфа:

- *команда ветвления;*
- *неполная форма ветвления;*
- *пример задачи с двухшаговой детализацией.*

Команда ветвления

Познакомимся еще с одной командой ГРИС. Она называется ***командой ветвления***. Формат команды ветвления такой:

если <условие>
  то <серия 1>
  иначе <серия 2>
кв

Служебное слово **кв** обозначает конец ветвления.

По-прежнему ГРИС может проверять только два условия: *«впереди край?»* или *«впереди не край?»*. <Серия> — это одна или несколько следующих друг за другом команд. Если <условие> справедливо, то выполняется <серия 1>, в противном случае — <серия 2>. Пример показан на рис. 5.12.

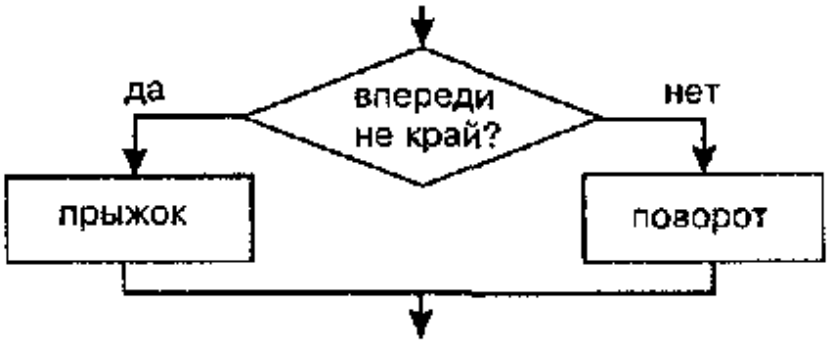


Рис. 5.12. Блок-схема полного ветвления

Такое ветвление называется ***полным***.

Неполная форма ветвления

В некоторых случаях используется ***неполная форма*** команды ветвления (рис. 5.13). Например:

если впереди край
  то поворот
кв

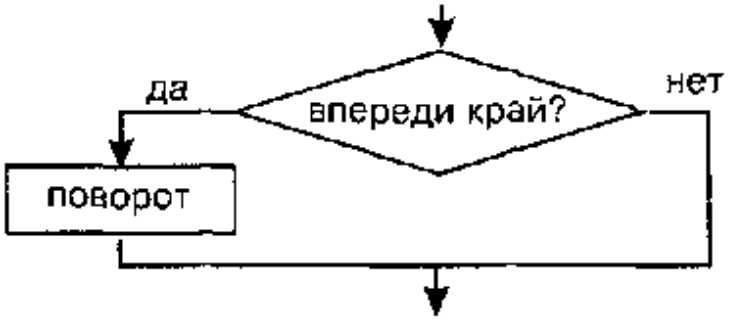


Рис. 5.13. Блок-схема неполного ветвления

Неполная команда ветвления имеет следующий формат:

если <условие>
  то <серия>
кв

Здесь <серия> выполняется, если <условие> справедливо.

Составим последнюю, сравнительно сложную программу для ГРИС. На этом примере вы увидите, что применение метода последовательной детализации облегчает решение некоторых «головоломных» задач.

Пример задачи с двухшаговой детализацией

Задача 6. Построить орнамент, состоящий из квадратов, расположенных по краю поля. Исходное положение ГРИС — в верхнем левом углу, направление на юг (рис. 5.14).

Процедуру, рисующую цепочку квадратов от края до края поля, назовем РЯД. Процедуру, рисующую один квадрат, назовем КВАДРАТ. Сначала напишем основную программу:

```
программа Орнамент
нач
        сделай РЯД
        поворот
        сделай РЯД
        поворот
        сделай РЯД
        поворот
        сделай РЯД
кон
```

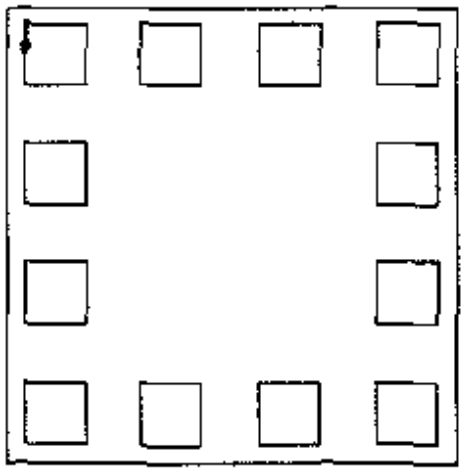


Рис. 5.14. Результат выполнения программы «Орнамент»

Теперь напишем процедуры РЯД и КВАДРАТ:

```
процедура РЯД
нач
  прыжок
  прыжок
  пока впереди не край, повторять
  нц
    сделай КВАДРАТ
    если впереди не край
    то прыжок
    кв
  кц
кон
```

```
процедура КВАДРАТ
нач
  шаг
  поворот
  шаг
  поворот
  шаг
  поворот
  шаг
  поворот
  прыжок
кон
```

В процедуре РЯД в теле цикла содержится неполное ветвление. Структуру такого алгоритма можно назвать так: ***цикл с вложенным ветвлением.***

На рис. 5.15 приведена блок-схема процедуры РЯД.

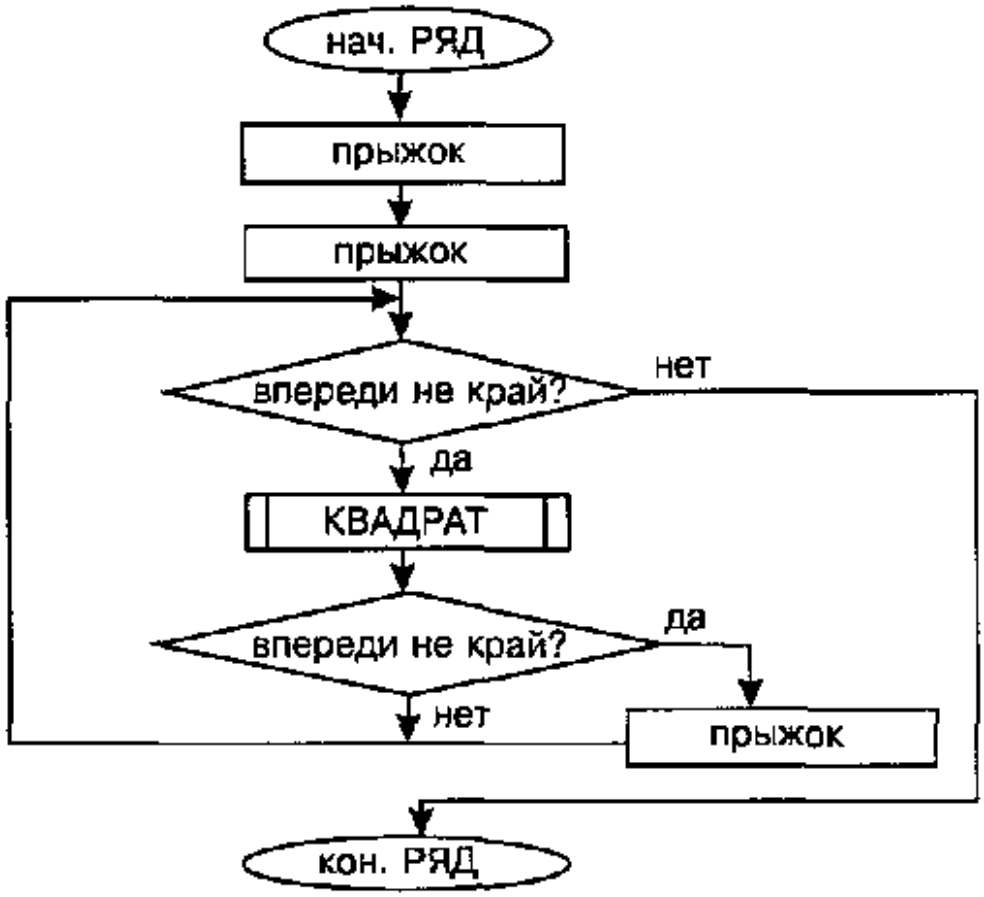


Рис. 5.15. Блок-схема процедуры РЯД

Составление этой программы потребовало двух шагов детализации алгоритма, которые выполнялись в такой последовательности:

ОСНОВНАЯ ПРОГРАММА
↓ 1-й шаг детализации
процедура РЯД
↓ 2-й шаг детализации
процедура КВАДРАТ

Теперь вам известны все команды управления графическим исполнителем. Их можно разделить на три группы: простые команды; команда обращения к процедуре; структурные команды. К третьей группе относятся команды цикла и ветвления.

СКИ графического исполнителя

Простые команды	Структурные команды
шаг	**пока** <условие>, **повторять**
поворот	**нц**
прыжок	<тело цикла>
	кц
Обращение к процедуре	
	если <условие>
сделай <имя процедуры>	**то** <серия 1>
	иначе <серия 2>
	кв

Коротко о главном

Команда ветвления имеет следующий формат:

```
если <условие>
      то <серия 1>
      иначе <серия 2>
 кв
```

Если <условие> истинно, то выполняются команды, составляющие <серию 1>, если ложно, то — <серию 2>.

Неполная команда ветвления имеет следующий формат:

```
если <условие>
      то <серия >
кв
```

Если условие истинно, то выполняется <серия>, если ложно, то сразу происходит переход к следующей команде алгоритма.

Сложные алгоритмы удобно строить путем пошаговой детализации.

? *Вопросы и задания*

1. Что такое пошаговая детализация?
2. Из каких команд могут состоять вспомогательные алгоритмы последнего уровня детализации?
3. Какой формат имеет команда ветвления? Какие действия исполнителя она определяет?
4. Чем отличается полное ветвление от неполного?
5. Путем пошаговой детализации составьте программы управления графическим исполнителем для решения следующих задач:
   - расчертить все поле горизонтальными пунктирными линиями;
   - нарисовать квадраты во всех четырех углах поля;
   - расчертить все поле в клетку со стороной, равной шагу.

Чему вы должны научиться, изучив главу 5

Освоить программное управление одним из учебных графических исполнителей.

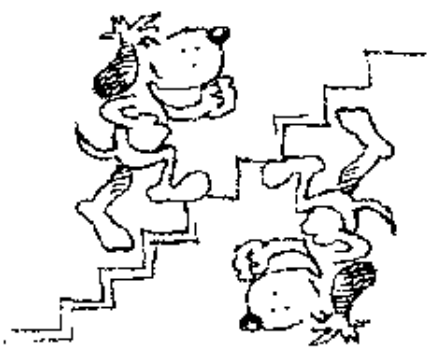
Составлять линейные программы.

Составлять циклические программы.

Составлять программы, содержащие ветвления.

Описывать и использовать вспомогательные алгоритмы (подпрограммы).

Применять метод последовательной детализации.



УПРАВЛЕНИЕ И

Кибернетическая модель управления

Управляющий объект	Объект или субъект, осуществляющий управление
Объект управления	Объект или субъект, выполняющий команды управления – *исполнитель алгоритма управления*
Прямая связь	Канал передачи команд управления
Обратная связь	Канал передачи данных о состоянии объекта управления
Алгоритм управления	Последовательность команд управления
Автоматические системы с программным управлением	Технические системы, в которых функции управляющего объекта выполняет компьютер

Система основных понятий главы 5

Глава 6

Программное управление работой компьютера

Здесь вы узнаете

- что такое программирование
- как строятся вычислительные алгоритмы
- как составляются программы на языке Паскаль

§ 32 Что такое программирование

Основные темы параграфа:

- *кто такие программисты;*
- *что такое язык программирования;*
- *что такое система программирования.*

Кто такие программисты

Теперь вам предстоит ближе познакомиться еще с одним разделом информатики, который называется «Программирование».

Назначение программирования — разработка программ управления компьютером с целью решения различных информационных задач.

Специалисты, профессионально занимающиеся программированием, называются ***программистами***. В первые годы существования ЭВМ для использования компьютера в любой области нужно было уметь программировать. В 1970-х – 80-х годах XX века начинает развиваться прикладное программное обеспечение. Бурное распространение прикладного ПО произошло с появлением персональных компьютеров. Стало совсем не обязательным уметь программировать для того, чтобы воспользоваться компьютером. Люди, работающие на компьютерах, разделились на ***пользователей*** и ***программистов***. В настоящее время пользователей гораздо больше, чем программистов.

Может возникнуть впечатление, что программисты теперь уже и не нужны! Но кто же тогда будет создавать все операционные системы, редакторы, графические пакеты, компьютерные игры и многое другое? Программисты, безусловно, нужны, причем задачи, которые им приходится решать, со временем становятся все сложнее.

Программирование принято разделять на системное и прикладное. ***Системные программисты*** занимаются разработкой системного программного обеспечения: операционных

систем, утилит и пр., а также систем программирования. ***Прикладные программисты*** создают прикладные программы: редакторы, табличные процессоры, игры, обучающие программы и многие другие. Спрос на высококвалифицированных программистов, как системных, так и прикладных, очень большой.

В данной главе вы познакомитесь с простейшими правилами и приемами программирования, заглянете в эту актуальную и престижную профессиональную область.

Что такое язык программирования

Для составления программ существуют разнообразные ***языки программирования.***

Язык программирования — это фиксированная система обозначений для описания алгоритмов и структур данных.

Популярными языками программирования сегодня являются Паскаль, Бейсик, Си, Фортран и др.

Что такое система программирования

Для создания и исполнения на компьютере программы, написанной на языке программирования, используются ***системы программирования.***

Система программирования — это программное обеспечение компьютера, предназначенное для разработки, отладки и исполнения программ, записанных на определенном языке программирования.

Существуют системы программирования на Паскале, Бейсике и других языках.

В данной главе речь будет идти о средствах и способах универсального программирования — не ориентированного на какую-то узкую прикладную область. Примером узкоспециализированного программирования является Web-программирование, ориентированное на создание Web-сайтов. Для этих целей, например, используются языки HTML, JavaScript. Языки Паскаль, Бейсик, Си относятся к числу ***универсальных языков программирования.***

Разработка любой программы начинается с построения алгоритма решения задачи. Ниже мы обсудим особенности алгоритмов решения задач обработки информации на компьютере. Такие алгоритмы называют алгоритмами работы с величинами.

Коротко о главном

Программирование — область информатики, посвященная разработке программ управления компьютером с целью решения различных информационных задач.

Программирование бывает системным и прикладным.

Паскаль, Бейсик, Си, Фортран — это универсальные языки программирования.

Система программирования — это программное обеспечение компьютера, предназначенное для разработки, отладки и исполнения программ, записанных на определенном языке программирования.

? *Вопросы и задания*

1. Что такое программирование?
2. Какие задачи решают системные и прикладные программисты?
3. Назовите наиболее распространенные языки программирования.
4. В чем состоит назначение систем программирования?

§ 33 Алгоритмы работы с величинами

Основные темы параграфа:

- *компьютер как исполнитель алгоритмов;*
- *величины: константы и переменные;*
- *система команд;*
- *команда присваивания;*
- *команда ввода;*
- *команда вывода.*

Компьютер как исполнитель алгоритмов

Вам уже известно, что всякий алгоритм составляется для конкретного исполнителя. *Теперь в качестве исполнителя мы будем рассматривать компьютер, оснащенный системой программирования на определенном языке.*

Компьютер-исполнитель работает с определенными *данными* по определенной *программе*. Данные — это множество величин.

Величины: константы и переменные

Компьютер работает с информацией, хранящейся в его памяти. Отдельный информационный объект (число, символ, строка, таблица и пр.) называется *величиной*.

Всякая обрабатываемая программой величина занимает свое место (поле) в памяти ЭВМ. Значение величины — это информация, хранимая в этом поле памяти.

Существуют *три основных типа величин*, с которыми работает компьютер: *числовой*, *символьный* и *логический*. Изучая базы данных и электронные таблицы, вы уже встречались с этими типами. В данной главе мы будем строить алгоритмы, работающие с числовыми величинами.

Числовые величины в программировании, так же как и математические величины, делятся на переменные и константы (постоянные). Например, в формуле $(a^2 - 2ab + b^2)$ a, b — переменные, 2 — константа.

Константы записываются в алгоритмах своими десятичными значениями, например: 23, 3.5, 34. Значение константы хранится в выделенной под нее ячейке памяти и остается неизменным в течение работы программы.

Переменные в программировании, как и в математике, обозначаются символическими именами. Эти имена называют *идентификаторами* (от глагола «идентифицировать», что значит «обозначать», «символизировать»). Идентификатор может быть одной буквой, множеством букв, сочетанием букв и цифр и т. д. Примеры идентификаторов: *A, X, B3, prim, r25* и т. п.

Система команд

Вам известно, что всякий алгоритм строится исходя из системы команд исполнителя, для которого он предназначен.

Независимо от того, на каком языке программирования будет написана программа, алгоритм работы с величинами составляется из следующих команд:

- ***присваивание***;
- ***ввод***;
- ***вывод***;
- ***обращение к вспомогательному алгоритму***;
- ***цикл***;
- ***ветвление***.

Команда присваивания

Команда присваивания — одна из основных команд в алгоритмах работы с величинами. Записывать ее мы будем так:

<переменная> := < выражение>

Значок «:=» читается «присвоить». Например:

$$Z := X + Y$$

Компьютер сначала вычисляет выражение, затем результат присваивает переменной, стоящей слева от знака «:=».

Если до выполнения этой команды содержимое ячеек, соответствующих переменным X, Y, Z, было таким:

X	2	Y	5	Z	—

то после выполнения команды оно станет следующим:

X	2	Y	5	Z	7

Прочерк в ячейке Z обозначает, что начальное число в ней может быть любым. Оно не имеет значения для результата данной команды.

Если слева от знака присваивания стоит числовая переменная, а справа — математическое выражение, то такую команду называют ***арифметической командой присваивания***, а выражение — арифметическим.

В частном случае арифметическое выражение может быть представлено одной переменной или одной константой. Например:

$X := 5$
$Y := X$

Команда ввода

Значения переменных, являющихся исходными данными решаемой задачи, как правило, задаются **вводом**.

Команда ввода в описаниях алгоритмов будет выглядеть так:

ввод <список переменных>.

Например:

ввод *A*, *B*, *C*

На современных компьютерах ввод чаще всего выполняется в режиме диалога с пользователем. По команде ввода компьютер прерывает выполнение программы и ждет действий пользователя. Пользователь должен набрать на клавиатуре вводимые значения переменных и нажать клавишу <ВВОД>. Введенные значения присвоятся соответствующим переменным из списка ввода, и выполнение программы продолжится.

Вот схема выполнения приведенной выше команды.

1. Память до выполнения команды:

A	—	*B*	—	*C*	—

2. Процессор компьютера получил команду *ввод A, B, C*, прервал свою работу и ждет действий пользователя.

3. Пользователь набирает на клавиатуре:

1 3 5

и нажимает клавишу <ВВОД> (<Enter>).

4. Память после выполнения команды:

A	1	*B*	3	*C*	5

5. Процессор переходит к выполнению следующей команды программы.

При выполнении пункта 3 вводимые числа должны быть отделены друг от друга какими-нибудь разделителями. Обычно это пробелы.

Из сказанного выше можно сделать вывод:

Переменные величины получают конкретные значения в результате выполнения команды присваивания или команды ввода.

Если переменной величине не присвоено никакого значения (или не введено), то она является неопределенной. Иначе говоря, ничего нельзя сказать, какое значение имеет эта переменная.

Команда вывода

Результаты решения задачи сообщаются компьютером пользователю путем выполнения команды вывода.

Команда вывода в алгоритмах будет записываться так:

вывод <список вывода>

Например:

вывод $X1$, $X2$

По этой команде значения переменных $X1$ и $X2$ будут вынесены на устройство вывода (чаще всего это экран).

О других командах, применяемых в вычислительных алгоритмах, вы узнаете позже.

Коротко о главном

Алгоритм решения любой задачи на компьютере составляется из следующих команд: присваивания; ввода; вывода; обращения к вспомогательному алгоритму; цикла; ветвления.

Программа для компьютера — это алгоритм, записанный на языке программирования.

Язык программирования — это фиксированная система обозначений для описания алгоритмов и структур данных.

Всякая обрабатываемая программой величина занимает определенное поле в памяти компьютера. Значение величины — это информация, хранимая в этом поле.

Переменная величина получает значение в результате выполнения команды присваивания или команды ввода.

Формат команды присваивания:

<переменная> := <выражение>

Сначала вычисляется выражение, затем полученное значение присваивается переменной.

Ввод — это занесение данных с внешних устройств в оперативную память компьютера. Исходные данные для решения задачи обычно задаются вводом.

Результаты решения задачи выносятся на устройства вывода (дисплей, принтер) по команде вывода.

? *Вопросы и задания*

1. Что такое величина? Чем отличаются переменные и постоянные величины?
2. Чем определяется значение величины?
3. Какие существуют основные типы величин в программировании?
4. Как записывается команда присваивания?
5. Что такое ввод? Как записывается команда ввода?
6. Что такое вывод? Как записывается команда вывода?
7. В схематическом виде (как это сделано в параграфе) отразите изменения значений в ячейках, соответствующих переменным A и B, в ходе последовательного выполнения команд присваивания:

1)	2)	3)
$A:=1$	$A:=1$	$A:=1$
$B:=2$	$B:=2$	$B:=2$
$A:=A+B$	$C:=A$	$A:=A+B$
$B:=2\times A$	$A:=B$	$B:=A-B$
	$B:=C$	$A:=A-B$

8. Вместо многоточия впишите в алгоритм несколько команд присваивания, в результате чего должен получиться алгоритм возведения в 4-ю степень введенного числа (дополнительные переменные, кроме A, не использовать):

ввод A . . . вывод A.

§ 34 Линейные вычислительные алгоритмы

Основные темы параграфа:

- *присваивание; свойства присваивания;*
- *обмен значениями двух переменных;*
- *описание линейного вычислительного алгоритма.*

Присваивание; свойства присваивания

Поскольку присваивание является важнейшей операцией в алгоритмах, работающих с величинами, то поговорим о ней более подробно.

Переменная величина получает значение в результате присваивания.

Присваивание производится компьютером при выполнении одной из двух команд из представленной выше системы: команды присваивания или команды ввода.

Рассмотрим последовательность выполнения четырех команд присваивания, в которых участвуют две переменные a и b. В приведенной ниже таблице против каждой команды указываются значения переменных, которые устанавливаются после ее выполнения. Такая таблица называется ***трассировочной таблицей***, а процесс ее заполнения называется ***трассировкой*** алгоритма. Компьютер выполняет команды в порядке их записи в алгоритме.

Команда	a	b
$a := 1$	1	—
$b := 2 \times a$	1	2
$a := b$	2	2
$b := a + b$	2	4

Прочерк в таблице обозначает неопределенное значение переменной. Конечные значения, которые получают переменные a и b, соответственно равны 2 и 4.

Этот пример иллюстрирует три основных свойства присваивания. Вот эти свойства:

1) пока переменной не присвоено значения, она остается неопределенной;
2) значение, присвоенное переменной, сохраняется вплоть до выполнения следующего присваивания этой переменной нового значения;
3) новое значение, присвоенное переменной, заменяет ее предыдущее значение.

Обмен значениями двух переменных

Рассмотрим еще один очень полезный алгоритм, с которым при программировании часто приходится встречаться. Даны две переменные величины X и Y. Требуется произвести между ними обмен значениями. Например, если первоначально было: $X = 1$; $Y = 2$, то после обмена должно стать: $X = 2$, $Y = 1$.

Хорошим аналогом для решения такой задачи является следующая: даны два стакана, в первом — молоко, во втором — вода; требуется произвести обмен их содержимым. Всякому ясно, что в этом случае нужен дополнительный третий пустой стакан. Последовательность действий будет следующей:

1) перелить из 1-го в 3-й;
2) перелить из 2-го в 1-й;
3) перелить из 3-го во 2-й.

Цель достигнута!

По аналогии для обмена значениями двух переменных нужна третья дополнительная переменная. Назовем ее Z. Тогда задача решается последовательным выполнением трех операторов присваивания (пусть начальные значения 1 и 2 для переменных X и Y задаются вводом):

Команда	X	Y	Z
ввод X, Y	1	2	–
$Z:=X$	1	2	1
$X:=Y$	2	2	1
$Y:=Z$	2	1	1
вывод X, Y	2	1	1

Действительно, в итоге переменные X и Y поменялись значениями. На экран будут выведены значения X и Y в таком порядке: 2, 1. В трассировочной таблице выводимые значения выделены жирным шрифтом.

Аналогия со стаканами не совсем точна в том смысле, что при переливании из одного стакана в другой первый становится пустым. В результате же присваивания (X:=Y) переменная, стоящая справа (Y), сохраняет свое значение.

Описание линейного вычислительного алгоритма

И наконец, рассмотрим пример составления алгоритма для решения следующей математической задачи: даны две простые дроби; получить дробь, являющуюся результатом их деления.

В школьном учебнике математики правила деления обыкновенных дробей описаны так:

1. Числитель первой дроби умножить на знаменатель второй.
2. Знаменатель первой дроби умножить на числитель второй.
3. Записать дробь, числителем которой является результат выполнения пункта 1, а знаменателем — результат выполнения пункта 2.

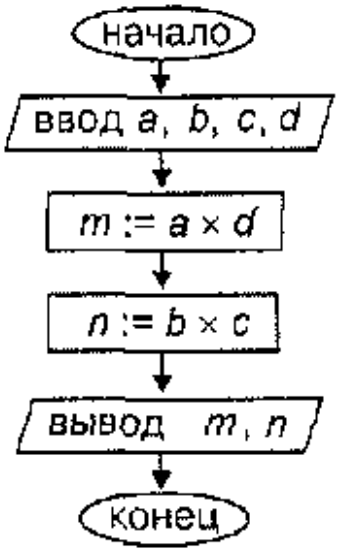
В алгебраической форме это выглядит следующим образом:

$$\frac{a}{b} : \frac{c}{d} = \frac{a \cdot d}{b \cdot c} = \frac{m}{n}.$$

Теперь построим алгоритм деления дробей для компьютера. В этом алгоритме сохраним те же обозначения для переменных, которые использованы в записанной выше формуле. Исходными данными являются целочисленные переменные a, b, c, d. Результатом — также целые величины m и n.

Ниже алгоритм представлен в двух формах: в виде блок-схемы и на Алгоритмическом языке (АЯ).

Раньше прямоугольник в схемах алгоритмов управления мы называли блоком простой команды. Для вычислительных алгоритмов такой простой командой является команда присваивания. Прямоугольник *будем называть блоком присваивания, или вычислительным блоком.* В форме параллелограмма рисуется блок ввода/вывода. Полученный алгоритм *имеет линейную структуру* (рис. 6.1).




```
алг   Деление дробей
цел   a, b, c, d, m, n
нач   ввод a, b, c, d
      m:=a×d
      n:=b×c
      вывод m, n
кон
```

Рис. 6.1. Алгоритм *деления дробей*

В алгоритме на АЯ строка, стоящая после заголовка алгоритма, называется ***описанием переменных.*** Служебное слово **цел** означает целый тип. Величины этого типа могут иметь только целочисленные значения.

Описание переменных имеет вид:

<тип переменных> <список переменных>

Список переменных включает все переменные величины данного типа, входящие в алгоритм.

В блок-схемах типы переменных не указываются, но подразумеваются. Запись алгоритма на АЯ ближе по форме к языкам программирования, чем блок-схемы.

Коротко о главном

Основные свойства присваивания:

- значение переменной не определено, если ей не присвоено никакого значения;
- новое значение, присваиваемое переменной, заменяет ее старое значение;
- присвоенное переменной значение сохраняется в ней вплоть до нового присваивания.

Обмен значениями двух переменных производится через третью дополнительную переменную.

Трассировочная таблица используется для «ручного» исполнения алгоритма с целью его проверки.

В алгоритмах на АЯ указываются типы всех переменных. Такое указание называется описанием переменных.

Числовые величины, принимающие только целочисленные значения, описываются с помощью служебного слова цел (целый).

Вопросы и задания

1. Из каких команд составляется линейный вычислительный алгоритм?
2. Что такое трассировка? Как она производится?
3. В каком случае значение переменной считается неопределенным?
4. Что происходит с предыдущим значением переменной после присваивания ей нового значения?
5. Как вы думаете, можно ли использовать в арифметическом выражении оператора присваивания неопределенную переменную? К каким последствиям это может привести?
6. Напишите на АЯ алгоритм сложения двух простых дробей (без сокращения дроби).
7. Напишите на АЯ алгоритм вычисления y по формуле

$$y = (1 - x^2 + 5x^4)^2,$$

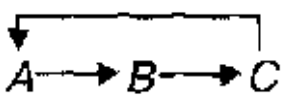
где x — заданное целое число. Учтите следующие ограничения: 1) в арифметических выражениях можно использовать только операции сложения, вычитания и умножения; 2) выражение может содержать только одну арифметическую операцию. Выполните трассировку алгоритма при $x = 2$.

8. Пользуясь ограничениями предыдущей задачи, напишите наиболее короткие алгоритмы вычисления выражений:

$$y = x^8; \quad y = x^{10}; \quad y = x^{15}; \quad y = x^{19}.$$

Постарайтесь использовать минимальное количество дополнительных переменных. Выполните трассировку алгоритмов.

9. Запишите алгоритм циклического обмена значениями трех переменных A, B, C. Схема циклического обмена:



Например, если до обмена было: $A = 1$, $B = 2$, $C = 3$, то после обмена должно стать: $A = 3$, $B = 1$, $C = 2$. Выполните трассировку.

§ 35 Знакомство с языком Паскаль

Основные темы параграфа:
возникновение и назначение Паскаля;
структура программы на Паскале;
операторы ввода, вывода, присваивания;
правила записи арифметических выражений;
пунктуация Паскаля.

Возникновение и назначение Паскаля

После того как построен алгоритм решения задачи, составляется программа на определенном языке программирования.

Среди современных языков программирования одним из самых популярных является язык ***Паскаль***. Этот язык разработан в 1971 году и назван в честь Блеза Паскаля — французского ученого, изобретателя механической вычислительной машины. Автор языка Паскаль — швейцарский профессор Никлаус Вирт.

> **Паскаль** — это универсальный язык программирования, позволяющий решать самые разнообразные задачи обработки информации.

Команду алгоритма, записанную на языке программирования, принято называть ***оператором***.

Программа на Паскале близка по своему виду к описанию алгоритма на Алгоритмическом языке. Сравните алгоритм решения уже знакомой вам задачи — деления простых дробей с соответствующей программой на Паскале:

```
алг Деление дробей
цел a, b, c, d, m, n
нач
  ввод a, b, c, d
  m:= a×d
  n:= b×c
  вывод m, n
кон
```

```
Program Division;
var a, b, c, d, m, n: integer;
  begin
      readln(a,b,c,d);    {Ввод}
       m:= a*d; {Числитель}
       n:= b*c; {Знаменатель}
       write(m, n)   {Вывод}
  end.
```

Структура программы на Паскале

Даже не заглядывая в учебник по Паскалю, в этой программе можно все понять (особенно помогает знание английского языка).

Заголовок программы начинается со слова **Program** (программа), за которым следует произвольное имя, придуманное программистом:

```
Program < имя программы>;
```

Раздел описания переменных начинается со слова **Var** (variables — переменные), за которым идет список имен переменных через запятую. Тип указывается после двоеточия. В стандарте языка Паскаль существуют два числовых типа величин: ***вещественный*** и ***целый***. Слово `integer` обозначает целый тип (является идентификатором целого типа). Вещественный тип обозначается словом `real`. Например, раздел описания переменных может быть таким:

```
var  a, b : integer; c, d : real;
```

Идентификаторы переменных составляются из латинских букв и цифр; первым символом обязательно должна быть буква.

Раздел операторов — основная часть программы. Начало и конец раздела операторов программы отмечаются служебными словами **begin** (начало) и **end** (конец). В самом конце программы ставится точка:

```
begin
    < операторы >
end.
```

Операторы ввода, вывода, присваивания

Ввод исходных данных с клавиатуры происходит по оператору `read` (read — читать) или `readln` (read line — читать строку):

```
read(<список переменных>);
или readln(<список переменных>);
```

При выполнении команды ввода компьютер ожидает действий пользователя. Пользователь набирает на клавиатуре значения переменных в том порядке, в каком они указаны в списке, отделяя их друг от друга пробелами. Одновременно с

набором данных на клавиатуре они появляются на экране. В конце нажимается клавиша <ВВОД> (<Enter>). Разница в выполнении операторов `readln` и `read` состоит в том, что после выполнения ввода по оператору `readln` экранный курсор перемещается в начало новой строки, а по оператору `read` этого не происходит.

Вывод результатов происходит по оператору `write` (write — писать) или `writeln` (write line — писать в строку):

```
write(<список вывода>);
или writeln(<список вывода>);
```

Результаты выводятся на экран компьютера в порядке их перечисления в списке. Элементами списка вывода могут быть константы, переменные, выражения.

Разница в выполнении операторов `writeln` и `write` состоит в том, что после выполнения вывода по оператору `writeln` экранный курсор перемещается в начало новой строки, а по оператору `write` этого не происходит.

Арифметический оператор присваивания на Паскале имеет следующий формат:

<числовая переменная> :=<арифметическое выражение>

Арифметическое выражение может содержать числовые константы и переменные, знаки арифметических операций, круглые скобки. Кроме того, в арифметических выражениях могут присутствовать функции.

Знаки основных арифметических операций записываются так:

\+ сложение,
\- вычитание,
* умножение,
/ деление.

Правила записи арифметических выражений

Запись арифметических выражений на Паскале похожа на обычную математическую запись. В отличие от математики, где часто пропускается знак умножения (например, пишут 2*A*), в Паскале этот знак пишется обязательно: 2*A. Например, математическое выражение

$$A^2 + B^2 - 12C$$

на Паскале записывается так:

```
A*A + B*B - 12*C
```

Это же выражение можно записать иначе:

```
SQR(A) + SQR(B) - 12*C
```

Здесь использована функция возведения в квадрат — SQR. Аргументы функций всегда пишутся в круглых скобках.

Последовательность выполнения операций определяется по их *приоритетам* (старшинству). К старшим операциям относятся умножение (*) и деление (/). Операции сложения и вычитания — младшие. В первую очередь выполняются старшие операции. Несколько операций одинакового старшинства, записанные подряд, выполняются в порядке их записи слева направо. Приведенное выше арифметическое выражение будет вычисляться в следующем порядке (порядок вычислений указан цифрами сверху):

```
  1   4   2   5   3
A * A + B * B - 12 * C
```

Круглые скобки в арифметических выражениях влияют на порядок выполнения операций. Как и в математике, в первую очередь выполняются операции в скобках. Если имеются несколько пар вложенных скобок, то сначала выполняются операции в самых внутренних скобках. Например:

```
  6     1     3     2     4   5
A + ((C - D) / (2 + K) - 1)*B
```

Пунктуация Паскаля

Необходимо строгое соблюдение правописания (синтаксиса) программы. В частности, в Паскале однозначно определено назначение знаков пунктуации.

Точка с запятой (;) ставится в конце заголовка программы, в конце раздела описания переменных, является разделителем операторов. Перед словом end точку с запятой можно не ставить.

Запятая (,) является разделителем элементов во всевозможных списках: списке переменных в разделе описания, списке вводимых и выводимых величин.

Строгий синтаксис в языке программирования необходим потому, что *компьютер является формальным исполнителем программы*. Если, допустим, разделителем в списке переменных должна быть запятая, то любой другой знак будет восприниматься как ошибка. Если точка с запятой является разделителем операторов, то в качестве оператора компьютер воспринимает всю часть текста программы от одной точки с запятой до другой. Если программист забыл поставить «;» между какими-то двумя операторами, то компьютер будет принимать их за один с неизбежной ошибкой.

В программу на Паскале можно вставлять комментарии. Комментарий — это пояснение к программе, которое записывается в фигурных скобках. В комментариях можно использовать русские буквы. На исполнение программы комментарий никак не влияет.

Заметим, что в Паскале нет различия между строчными и прописными буквами. Например, для Паскаля тождественны следующие варианты записи: `begin`, `Begin`, `BEGIN`, `BeGiN`. Использование строчных или прописных букв — дело вкуса программиста.

Коротко о главном

Паскаль — универсальный язык программирования.

Программа на Паскале состоит из заголовка, описаний и операторов.

Формат заголовка программы:

Program <имя программы>;

Формат описания переменных:

var <список однотипных переменных> : <тип>; ...

Раздел операторов:

```
begin
      <операторы>
end.
```

Операторы ввода данных с клавиатуры:

read(<список ввода>), readln(<список ввода>).

Операторы вывода на экран:

write(<список вывода>, writeln(<список вывода>).

Арифметический оператор присваивания:

<переменная>:=<арифметическое выражение>

Арифметическое выражение может содержать любое количество арифметических операций и функций.

Последовательность выполнения операций определяется расстановкой скобок и старшинством операций (приоритетами). Старшие операции: *, /; младшие операции: +, − .

Точка с запятой ставится в конце заголовка программы, в конце описаний, а также является разделителем операторов. Текст всей программы заканчивается точкой.

? Вопросы и задания

1. Когда появился язык Паскаль и кто его автор?
2. Как записывается заголовок программы на Паскале?
3. Как записывается раздел описания переменных?
4. С какими типами числовых величин работает Паскаль?
5. Как записываются операторы ввода и вывода в Паскале?
6. Что такое оператор присваивания?
7. Как записываются арифметические выражения?
8. По каким правилам определяется порядок выполнения операций в арифметическом выражении?
9. Какая задача решается по следующей программе?

```
Program Test;
var A, B, C: integer;
begin
      readln(A,B);
      C:=(A+B)*(B-A)
      writeln(C)
end.
```

Какой результат будет получен, если в качестве исходных значений *A* и *B* ввести соответственно 7 и 8?

10. Составьте программы на Паскале для решения задач № 6–9 из заданий к § 34. При этом отмените ограничения на количество операций в арифметическом выражении, сформулированные в условиях задач.

§ 36 Алгоритмы с ветвящейся структурой

Основные темы параграфа:

- *представление ветвлений на АЯ. Трассировка ветвящихся алгоритмов;*
- *сложные ветвящиеся алгоритмы.*

Представление ветвлений на АЯ. Трассировка ветвящихся алгоритмов

Рассмотрим несколько задач, решение которых на компьютере получается с помощью ветвящихся алгоритмов.

Первая задача: *даны два числа; выбрать большее из них.*

Пусть исходными данными являются переменные A и B. Их значения будут задаваться вводом. Значение большего из них должно быть присвоено переменной C и выведено на экран компьютера. Например, *если* $A = 5$, $B = 8$, *то должно получиться:* $C = 8$.

Блок-схема алгоритма решения этой задачи изображена на рис. 6.2.

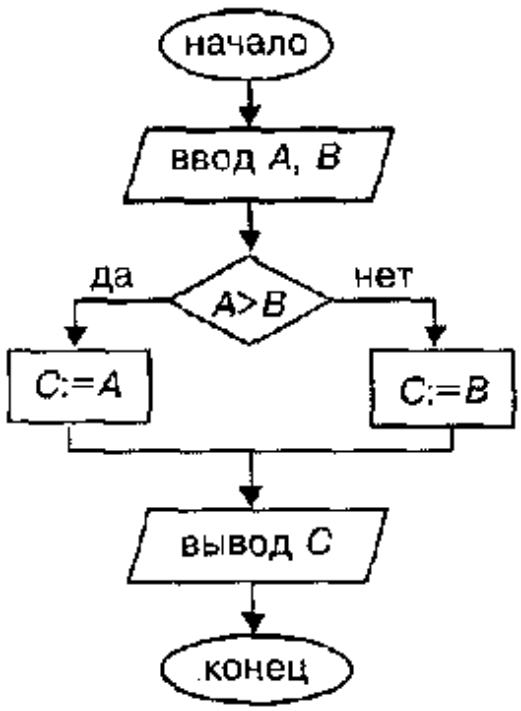


Рис. 6.2. Алгоритм выбора большего из двух чисел (с полным ветвлением)

Нетрудно понять смысл этого алгоритма. Если значение переменной A больше, чем B, то переменной C присвоится значение A. В противном случае, когда $A \le B$, переменной C присвоится значение B.

Условием, по которому разветвляется алгоритм, является *отношение неравенства* $A > B$. Изучая базы данных и электронные таблицы, вы узнали, что такое отношение является *логическим выражением*. Если оно справедливо, то результатом будет логическая величина «истина» и выполнение алгоритма продолжится по ветви «да»; в противном случае логическое выражение примет значение «ложь» и выполнение алгоритма пойдет по ветви «нет».

До выполнения на компьютере правильность алгоритма можно проверить путем заполнения трассировочной таблицы. Вот как будет выглядеть трассировка нашего алгоритма для исходных значений $A = 5$, $B = 8$.

Шаг	Операция	A	B	C	Проверка условия
1	ввод A, B	5	8		
2	A>B	5	8		5 > 8, нет (ложь)
3	C:=B	5	8	8	
4	вывод C	5	8	8	

Ветвление является ***структурной командой***. Его исполнение происходит в несколько шагов: проверка условия (выполнение логического выражения) и выполнение команд на одной из ветвей «да» или «нет». Поэтому в трассировочной таблице записываются не команды алгоритма, а отдельные операции, выполняемые компьютером на каждом шаге.

В алгоритме на рис. 6.2 используется ***полное ветвление***. Эту же самую задачу можно решить, применяя структурную команду ***неполного ветвления***. Блок-схема такого алгоритма изображена на рис. 6.3.

Выполните самостоятельно трассировку этого алгоритма для вариантов 1) $A = 0{,}2$, $B = 0{,}3$; 2) $A = 7$, $B = 4$; 3) $A = 5$, $B = 5$. Если вы все проделаете правильно, то убедитесь, что алгоритм верный.

А теперь запишем рассмотренные алгоритмы на Алгоритмическом языке (АЯ). Во-первых, нужно решить вопрос о том, как описать переменные в этом алгоритме. Вспомним, что для всех переменных в алгоритме на Алгоритмическом языке необходимо указать их тип.

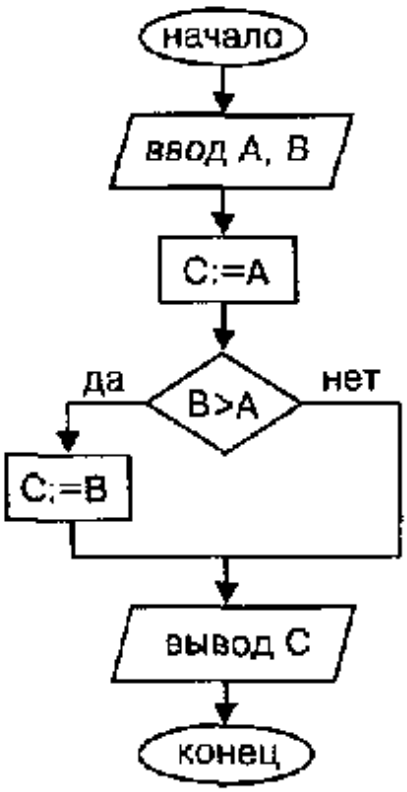


Рис. 6.3. Алгоритм выбора большего из двух значений (с неполным ветвлением)

Переменные *A*, *B*, *C* — числовые величины. В этой задаче они могут принимать любые значения. В программировании числовые величины, которые могут иметь любые значения — целые, дробные, — называются вещественными. Им ставится в соответствие ***вещественный тип.*** На Алгоритмическом языке этот тип указывается служебным словом **вещ**.

Как выглядит команда ветвления, вы уже знаете. Вот два алгоритма на АЯ, соответствующие блок-схемам на рис. 6.2 и 6.3:

```
алг БИД1
вещ A, B, C
нач   ввод A, B
      если A>B
      то C:=A
      иначе C:=B
      кв
      вывод C
кон
```

```
алг БИД2
вещ A, B, C
нач ввод A, B
    C:=A
    если B>A
    то C:=B
    кв
    вывод C
кон
```

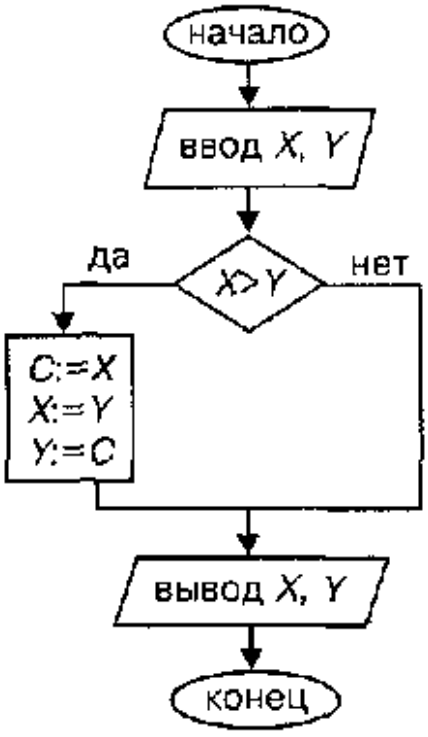
Под сокращенным названием алгоритмов БИД подразумевается «Большее из двух».

Для программирования характерно то, что одна и та же задача может быть решена с помощью разных алгоритмов. *И чем сложнее задача, тем больше можно придумать* различных алгоритмов ее решения. Для больших задач (производственных, научных) практически невозможно точное совпадение алгоритмов, составленных разными программистами.

Следующая задача: *упорядочить значения двух переменных X и Y по возрастанию.* Смысл этой задачи следующий: если для исходных значений переменных справедливо отношение $X \le Y$ (например, $X = 1$, $Y = 2$), то оставить их без изменения; если же $X > Y$ (например, $X = 2$, $Y = 1$), то выполнить обмен значениями.

Алгоритм обмена значениями двух переменных был рассмотрен в предыдущем параграфе. Вспомним, что для обмена *нужна третья вспомогательная переменная.*

В алгоритме решения данной задачи используется неполное ветвление. Приведем блок-схему (рис. 6.4) и алгоритм на АЯ.




```
алг СОРТИРОВКА
вещ X, Y, C
нач  ввод X, Y
     если X>Y
     то   C:=X
          X:=Y
          Y:=C
     кв
     вывод X, Y
кон
```

Рис. 6.4. Блок-схема алгоритма упорядочения двух величин

Здесь роль вспомогательной переменной для обмена выполняет C.

Сложные ветвящиеся алгоритмы

Получим алгоритм решения еще одной задачи: *найти наибольшее значение среди трех величин: A, B, C.*

Естественно, возникает следующая идея этого алгоритма: сначала нужно найти большее из значений A и B и присвоить его какой-то дополнительной переменной, например D; затем найти большее среди D и C. Это значение можно присвоить той же переменной D.

Решение задачи сводится к двукратному применению уже знакомого алгоритма нахождения большего из двух значений. Блок-схема алгоритма — на рис. 6.5.

```
алг БИТ1
вещ A, B, C, D
нач ввод A, B, C
    если A>B
    то D:=A
    иначе D:=B
    кв
    если C>D
    то D:=C
    кв
    вывод D
кон
```

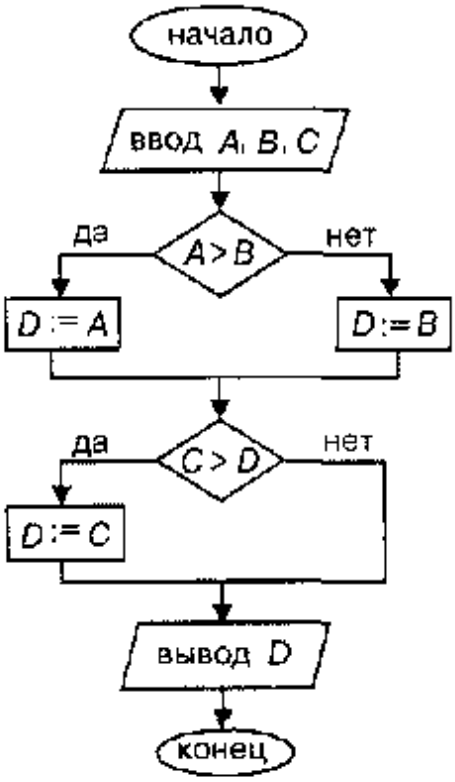


Рис. 6.5. Блок-схема алгоритма «БИТ» с последовательными ветвлениями

Нетрудно догадаться, что «БИТ» обозначает «Большее из трех». В структуре этого алгоритма содержатся *два последовательных ветвления*: первое — полное, второе — неполное.

Эту же задачу можно решить с помощью алгоритма, имеющего структуру *вложенных ветвлений*. Его блок-схема приведенная в следующем параграфе на рис. 6.6.

А вот как выглядят описание этого алгоритма на АЯ и трассировочная таблица при $A = 5$, $B = 7$, $C = 2$.

```
Алг БИТ2
вещ A, B, C, D
нач ввод A, B, C
    если A>B
    то    если A>C  то D:=A иначе D:=C кв
    иначе если B>C  то D:=B иначе D:=C кв
    кв
      вывод D
кон
```

Шаг	Операция	*A*	*B*	*C*	*D*	Проверка условия
1	ввод *A*, *B*, *C*	5	7	2	–	
2	*A*>*B*	5	7	2	–	5 > 7, нет
3	*B*>*C*	5	7	2	–	7 > 2, да
4	*D*:=*B*	5	7	2	7	
5	вывод *D*	5	7	2	7	

Коротко о главном

В команде ветвления в качестве условия может использоваться отношение неравенства между величинами.

Числовые величины, которые могут принимать любые значения (целые и дробные), имеют вещественный тип.

Для решения одной и той же задачи можно построить несколько вариантов алгоритмов.

Несколько ветвлений в одном алгоритме могут быть последовательными и вложенными.

Вопросы и задания

1. Какую структуру имеет алгоритм нахождения большего из двух значений?
2. Почему отношение неравенства можно назвать логическим выражением?
3. В каком случае для числовой переменной следует указывать целый тип, в каком — вещественный?

4. Составьте алгоритм (в виде блок-схемы и на АЯ) нахождения меньшего из двух значений.
5. Составьте алгоритм нахождения наименьшего из трех значений.
6. Для вывода на экран произвольной символьной строки нужно в команде вывода записать эту строку в апострофах. Например, по команде

вывод "ОТВЕТ"

на экран выведется слово ОТВЕТ.

Определите, какая задача решается по следующему алгоритму:

```
алг Задача-6
вещ X
нач ввод X
      если X<0
      то вывод "отрицательное число"
      иначе вывод "положительное число"
      кв
кон
```

7. Составьте алгоритм, по которому на компьютере будет происходить следующее: в переменную S вводится возраст Саши, в переменную M вводится возраст Маши. В качестве результата на экран выводится фраза «Саша старше Маши» или «Маша старше Саши» (предполагаем, что кто-нибудь из них обязательно старше).
8. *Решите предыдущую задачу, учитывая возможность одинакового возраста Саши и Маши.* В таком случае может быть получен ответ: «Саша и Маша — ровесники».
9. Составьте алгоритм упорядочения значений трех переменных по возрастанию, т. е. при любых исходных значениях A, B, C отсортируйте их так, чтобы стало: $A \le B \le C$. Проверьте алгоритм трассировкой при разных вариантах значений исходных данных.

§ 37 Программирование ветвлений на Паскале

Основные темы параграфа:
оператор ветвления на Паскале;
программирование полного и неполного ветвления;
программирование вложенных ветвлений;
логические операции;
сложные логические выражения.

Оператор ветвления на Паскале

В языке Паскаль имеется ***оператор ветвления***. Другое его название — условный оператор. Формат полного оператора ветвления следующий:

```
if <логическое выражение> then <оператор1>
                          else <оператор2>
```

Здесь `if` — «если», `then` — «то», `else` — «иначе».

Программирование полного и неполного ветвления

Сравните запись алгоритма БИД1 из предыдущего параграфа с соответствующей программой.

```
алг БИД1                    Program BID1;
вещ A, B, C                 var A, B, C : real;
нач ввод A, B               begin readln(A, B);
    если A>B                      if A>B
    то C:=A                       then C:=A
    иначе C:=B                    else C:=B;
    кв
    вывод C                       writeln(C)
кон                         end.
```

Очень похоже на перевод с русского языка на английский. Обратите внимание на следующее отличие: в программе нет специального служебного слова, обозначающего конец ветвления. Здесь признаком конца оператора ветвления является точка с запятой. (Разумеется, оставлять в программе пустую строку совсем не обязательно. Здесь это сделано только ради наглядности.)

Простой формой логического выражения является ***операция отношения***. Как и в АЯ, в Паскале допускаются все виды отношений (ниже указаны их знаки):

< (меньше);	<= (больше или равно);
> (больше);	= (равно);
<= (меньше или равно);	<> (не равно).

А теперь запрограммируем на Паскале алгоритм БИД2, в котором использовано неполное ветвление.

```
алг БИД2                    Program BID2;
вещ A, B, C                 var  A, B, C : real;
нач ввод A, B               begin readln(A, B);
    C:=A                          C:=A;
    если B > A                    if  B>A
    то C:=B                       then C:=B;
    кв
    вывод C                       write(C)
кон                         end.
```

Опять все очень похоже. Ветвь **else** в операторе ветвления может отсутствовать.

Программирование вложенных ветвлений

Запишем на Паскале программу определения большего из трех чисел, блок-схема которой показана на рис. 6.6. Структура этого алгоритма — вложенные ветвления. Алгоритм на АЯ (БИТ2) *приведен в предыдущем параграфе.*

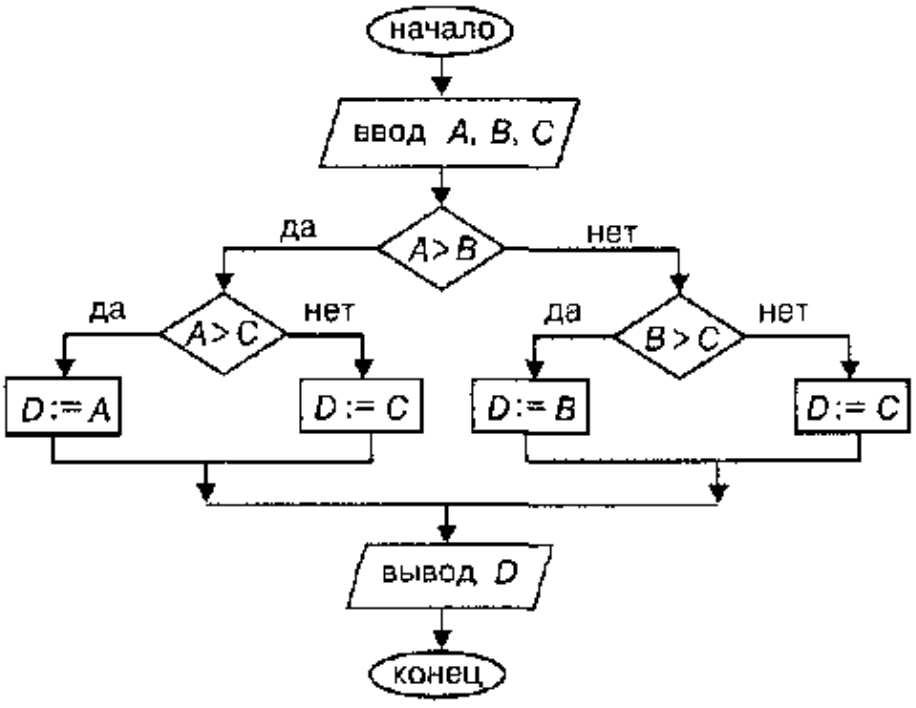


Рис. 6.6. Блок-схема алгоритма «БИТ» с вложенными ветвлениями

```
Program BIT2;
var A, B, C, D: real;
begin readln(A, B, C);
    if A>B
          then if A>C then D:=A else D:=B
          else if B>C then D:=B else D:=C;
    writeln(D)
end.
```

Обратите внимание на то, что перед **else** точка с запятой не ставится. Вся ветвящаяся часть структуры алгоритма заканчивается на точке с запятой после оператора D:=C.

Составим программу упорядочения значений двух переменных.

```
алг СОРТИРОВКА
вещ X, Y, C
нач ввод X, Y
    если X>Y
    то C:=X
       X:=Y
       Y:=C
    кв
    вывод X, Y
кон
```

```
Program SORTING;
var X, Y, C : real;
begin readln(X, Y);
  if  X>Y
   then begin C:=X;
                X:=Y;
                Y:=C
        end;
 write(X,Y)
end.
```

Этот пример иллюстрирует следующее правило Паскаля: если на какой-то из ветвей оператора ветвления находится несколько последовательных операторов, то их нужно записывать между служебными словами **begin** и **end**. Конструкция такого вида:

begin <последовательность операторов> **end**

называется ***составным оператором***. Следовательно, в описанной выше общей форме ветвления <оператор1> и <оператор2> могут быть простыми (один) и составными операторами.

Логические операции

Наконец, составим еще один, третий вариант программы определения большего числа из трех.

```
Program BIT3;
var A,B,C,D: real;
begin readln(A,B,C);
```

```
        if (A>=B) and (A>=C) then D:=A;
        if (B>=A) and (B>=C) then D:=B;
        if (C>=A) and (C>=B) then D:=C;
        writeln(D)
end.
```

Нетрудно понять смысл этой программы. Здесь использованы три последовательных неполных ветвления. А условия ветвлений представляют собой ***сложные логические выражения***, включающие ***логическую операцию*** **and** (И). С логическими операциями вы встречались, работая с базами данных и с электронными таблицами.

Напомним, что операция **and** называется логическим умножением или конъюнкцией. Ее результат — «истина», если значения обоих операндов — «истина». Очевидно, что если $A \geq B$ и $A \geq C$, то A имеет наибольшее значение и т. д. В Паскале присутствуют все три основные логические операции:

and — И (конъюнкция),
or — ИЛИ (дизъюнкция),
not — НЕ (отрицание).

Сложные логические выражения

Обратите внимание на то, что отношения, связываемые логическими операциями, заключаются в скобки. Так надо делать всегда! Например, требуется определить, есть ли среди чисел A, B, C хотя бы одно отрицательное. Эту задачу решает следующий оператор ветвления:

```
if (A<0)or(B<0)or(C<0)
  then write('YES') else write('NO');
```

Выражение, истинное для отрицательного числа, может быть записано еще и так:

```
not(A>=0)
```

Коротко о главном

Оператор ветвления (условный оператор) Паскаля имеет вид:

```
if <логическое выражение>
   then <оператор1> else <оператор2>
```

На ветвях условного оператора могут находиться простые или составные операторы. Составной оператор — это после-

довательность операторов, заключенная между служебными словами **begin** и **end**.

В сложных логических выражениях используются логические операции: **and**, **or**, **not**.

? *Вопросы и задания*

1. Как программируется на Паскале полное и неполное ветвление?
2. Что такое составной оператор? В каких случаях составной оператор используется в операторе ветвления?
3. Выполните на компьютере все программы, приведенные в данном параграфе.
4. Составьте не менее трех вариантов программы определения наименьшего из трех данных чисел.
5. Составьте программу сортировки по возрастанию значений в трех переменных: *A*, *B*, *C*.
6. Составьте программу вычисления корней квадратного уравнения по данным значениям его коэффициентов.

§ 38 Программирование диалога с компьютером

Основные темы параграфа:

- *что такое диалог с компьютером;*
- *пример программирования диалога.*

Что такое диалог с компьютером

Если вы исполняли рассмотренные выше программы на компьютере, то почувствовали определенное неудобство при работе с машиной. Во-первых, непонятно, когда машина начинает ожидать ввода данных, какие данные и в каком порядке нужно вводить (это ведь можно и забыть). Во-вторых, результаты получаются в виде чисел на экране, без всяких пояснений их смысла. Ясно, что люди между собой так не общаются.

> Любую программу составлять нужно так, чтобы ее исполнение имитировало диалог между *компьютером и пользователем в понятной* для человека форме.

Прежде чем начать составление программы, нужно продумать *сценарий* такого диалога.

Например, составим сценарий работы программы, вычисляющей сумму двух целых чисел. На экране компьютера последовательно должны появляться следующие строки (для примера предположим, что будем вводить числа 237 и 658):

Введите первое слагаемое: A = **237**
Введите второе слагаемое: B = **658**
A + B = 895
Пока!

Здесь курсивом записаны символы, которые выводит компьютер по программе, а прямым жирным шрифтом — символы, вводимые пользователем.

> Любой вывод на экран происходит по оператору вывода, записанному в программе.

Следовательно, с помощью оператора вывода на экран выносятся не только результаты решения задачи, но и все элементы диалога со стороны компьютера.

Вот программа, которая реализует наш сценарий:

```
Program Summa;
var A, B : integer;
begin  write ('Введите первое слагаемое: A =');
       readln(A);
       write('Введите второе слагаемое: B =');
       readln(B);
       writeln;
       writeln('A + B = ', A+B);
       writeln('Пока!')
end.
```

В этой программе используется возможность включать в список вывода символьные строки, заключенные в апостро-

фы, и арифметические выражения. Выражение A+B сначала вычисляется, а потом полученное число выводится на экран. Конечно, для вычисления суммы можно было написать отдельный оператор присваивания, но можно и так, как в этом примере.

Еще обратите внимание на оператор writeln без списка вывода. Он обеспечивает пропуск строки на экране.

Пример программирования диалога

Компьютерная программа совсем не обязательно должна иметь математическое содержание. Вот пример сценария, судя по которому компьютер выполняет роль электронной няньки, заботящейся о здоровье школьника. Приводятся два варианта развития сценария, в зависимости от ответа ребенка.

Вариант 1:

Ты вчера был болен. Измерь-ка температуру! Сообщи, какая у тебя температура: **36.5**
Ты здоров, дружок! Можешь идти в школу. Желаю успехов!

Вариант 2:

Ты вчера был болен. Измерь-ка температуру!
Сообщи, какая у тебя температура: **37.3**
Ты еще болен! Раздевайся и ложись в постель.
Поправляйся, дружок!

Алгоритм этой программы содержит ветвление. Идея алгоритма состоит в том, что значение температуры ребенка сравнивается с величиной нормальной температуры человека: 36,6 °С. И если у ребенка температура выше, то он нездоров. Вот соответствующий алгоритм на АЯ:

```
алг НЯНЬКА
вещ T
нач  вывод "Ты вчера был болен. Измерь-ка температуру!"
     вывод "Сообщи, какая у тебя температура: "
 ввод(T)
 если T> 36.6
 то  вывод "Ты еще болен! Раздевайся и ложись в постель."
     вывод "Поправляйся, дружок!"
 иначе вывод "Ты здоров, дружок!
```

Можешь идти в школу."
вывод "Желаю успехов!"
кв
кон

По этому алгоритму получается следующая программа на Паскале:

```
Program  NANNY;
Var T: real;
begin writeln('Ты вчера был болен.
                Измерь-ка температуру! ');
write('Сообщи, какая у тебя температура: ');
readln(T);
if T>36.6
then begin
       writeln('Ты еще болен! Раздевайся и
                ложись в постель.');
       writeln('Поправляйся, дружок! ')
     end
else begin
       writeln('Ты здоров, дружок! Можешь
                идти в школу.');
        writeln('Желаю успехов! ')
     end
end.
```

Обратите внимание на два момента: во-первых, *перед словом* **else** *ни в коем случае нельзя ставить точку с запятой*; во-вторых, в записи и при вводе вещественных чисел *целая и дробная части числа отделяются десятичной точкой.*

Составляя подобную программу, вы сами организуете интерфейс компьютера с пользователем вашей программы. Этот *интерфейс* обязательно *должен быть дружественным*. Содержание диалога должно быть понятным и удобным.

Коротко о главном

Сценарий работы программы — это описание ее общения с пользователем (*пользовательского интерфейса*). Интерфейс обязательно должен быть дружественным.

Любой символьный вывод на экран программируется с помощью оператора **write** или **writeln**.

Вопросы и задания

1. Что обозначает понятие «диалоговый характер программы»?
2. Какими средствами программируется диалог между пользователем и компьютером?
3. Что обозначает понятие «дружественный интерфейс»?
4. Выполните на компьютере все программы, приведенные в данном параграфе.
5. Постройте алгоритм и составьте программу, по которой будет реализован следующий сценарий: компьютер запрашивает номер дня недели, после ввода компьютер сообщает название этого дня. Например, если ввели 1, то выведется фраза «Это понедельник» и т. д.

§ 39 Программирование циклов

Основные темы параграфа:
этапы решения расчетной задачи на компьютере;
задача о перестановке букв. Программирование цикла на Паскале;
что такое отладка и тестирование программы.

Вы научились составлять линейные и ветвящиеся программы на Паскале. Теперь нужно освоить программирование циклов. Снова будем учиться на примере конкретной задачи. Но, в отличие от предыдущих примеров, подход к ее решению будет несколько другим.

Этапы решения расчетной задачи на компьютере

Часто задача, которую требуется решить, сформулирована не на математическом языке. Для решения на компьютере ее сначала нужно привести к форме математической задачи, а потом уже программировать.

Работа по решению таких задач с использованием компьютера проходит через следующие этапы:

1. Постановка задачи.
2. Математическая формализация.
3. Построение алгоритма.
4. Составление программы на языке программирования.

5. Отладка и тестирование программы.
6. Проведение расчетов и анализ полученных результатов.

Эту последовательность называют ***технологией решения задачи на компьютере.***

В чистом виде программированием, т. е. разработкой алгоритма и программы, здесь являются лишь 3-й, 4-й и 5-й этапы.

На этапе постановки задачи должно быть четко определено, ***что дано*** и ***что требуется найти.***

Второй этап — математическая формализация. Здесь задача переводится на язык математических формул, уравнений, отношений. Далеко не всегда эти формулы очевидны. Нередко их приходится выводить самому или отыскивать в специальной литературе. Если решение задачи требует математического описания какого-то реального объекта, явления или процесса, то формализация равносильна получению соответствующей ***математической модели.***

Третий этап — построение алгоритма. Вы знаете два способа описания алгоритмов: блок-схемы и Алгоритмический язык (АЯ).

Первые три этапа — это работа без компьютера. Дальше следует собственно программирование на определенном языке в определенной системе программирования. Последний (шестой) этап — это использование уже разработанной программы в практических целях.

Задача о перестановке букв. Программирование цикла на Паскале

Проследим все этапы технологии на примере конкретной задачи.

1. Постановка задачи. Дано N кубиков, на которых написаны разные буквы. Сколько различных N-буквенных слов можно составить из этих кубиков (слова не обязательно должны иметь смысл)?

Искомую целочисленную величину обозначим буквой F. Тогда постановка задачи выглядит так:

Дано: N

Найти: F.

2. Математическая формализация. Получим расчетную формулу. Сначала рассмотрим несколько конкретных примеров. Имеются два кубика с буквами «И» и «К». Ясно, что из них можно составить два слова:

ИК КИ.

Добавим к ним *третью* букву С. Теперь число разных слов будет в *три раза* больше предыдущего, т. е. равно 6:

ИКС КИС ИСК СКИ КСИ СИК.

Если добавить *четвертую* букву, например «А», то число слов возрастет *в четыре раза* и станет равным 24:

КИСА КИАС КСИА КСАИ КАИС КАСИ ИКСА ИКАС
ИСКА ИСАК ИАКС ИАСК СКИА СКАИ СИКА СИАК
САКИ САИК АКИС АКСИ АИКС АИСК АСКИ АСИК.

Попробуйте записать все варианты слов из пяти букв: И, К, С, А, У. Сделать это непросто. Ясно лишь, что количество таких слов будет *в пять раз больше* 24, т. е. равно 120. Из шести букв можно составить 720 различных слов. С ростом числа букв число слов быстро растет. Например, для 10 букв получается 3 628 800 слов.

Подобные задачи решает раздел математики, который называется ***комбинаторикой***.

Количество различных комбинаций из N предметов, получаемых изменением их порядка, называется числом ***перестановок****. Это число выражается функцией от N, которая называется* ***факториалом*** *и записывается так:*

$N!$

Читается: «***N факториал***». Для любого натурального N значение $N!$ вычисляется как произведение последовательности натуральных чисел от 1 до N. Например:

$1! = 1;$
$2! = 1 \cdot 2 = 2;$
$3! = 1 \cdot 2 \cdot 3 = 6;$
$4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24;$
$5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$
и т. д.

Теперь вернемся к формулировке задачи. Если N обозначает количество букв, а F — количество слов из этих букв, то расчетная формула такова:

$$F = N! = 1 \cdot 2 \cdot \ldots \cdot N.$$

3. Построение алгоритма. Поскольку алгоритм должен быть независимым от данного значения N, то его нельзя сделать линейным. Дело в том, что для разных N надо выполнить разное число умножений. В таком случае с изменением N линейная программа должна была бы менять длину.

Алгоритм решения данной задачи будет ***циклическим.*** С циклическими алгоритмами вы уже познакомились, работая с графическим исполнителем.

Цикл — это команда исполнителю многократно повторить указанную последовательность команд.

Рассмотрим блок-схему на рис. 6.7 и алгоритм на АЯ.

Здесь применена знакомая вам алгоритмическая структура ***«цикл с предусловием»***. Выполняется она так: *пока истинно условие цикла, повторяется выполнение тела цикла.*

```
алг СЛОВА
цел F, N, R
нач ввод N
    F:=1
    R:=1
    пока R<=N, повторять
    нц
        F:=F*R
        R:=R+1
    кц
    вывод F
кон
```

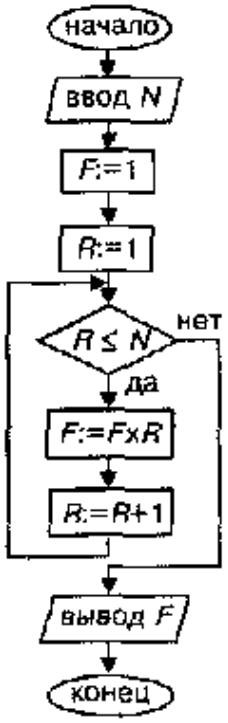


Рис. 6.7. Блок-схема алгоритма вычисления факториала

Тело цикла составляют две команды присваивания, заключенные между служебными словами **нц** и **кц**. Условие цикла — это отношение $R<=N$ (R меньше или равно N).

В данном алгоритме переменная R выполняет роль множителя, значение которого меняется от 1 до N через единицу. Произведение накапливается в переменной F, начальное значение которой равно 1. Цикл заканчивается, когда R становится равно $N + 1$. Это значение в произведение уже не попадет.

Для проверки правильности алгоритма построим трассировочную таблицу (для случая $N = 3$):

Шаг	Операция	N	F	R	Условие
1	ввод N	3	–	–	
2	F:=1		1	–	
3	R:=1			1	
4	$R<=N$				1<=3, да
5	$F:=F*R$		1		
6	$R:=R+1$			2	
7	$R<=N$				2<=3, да
8	$F:=F*R$		2		
9	$R:=R+1$			3	
10	$R<=N$				3<=3,да
11	$F:=F*R$		6		
12	$R:=R+1$			4	
13	$R<=N$				4<=3, нет
14	вывод F		6		
15	конец				

Из этой таблицы хорошо видно, как менялись значения переменных. Новое значение, присвоенное переменной, стирает ее старое значение (в данной таблице не повторяется запись значения переменной, если оно не изменяется; в таком виде таблица менее загромождена числами). Последнее значение F равно 6. Оно выводится в качестве результата. Очевидно, что результат верный: $3! = 6$.

4. Составление программы. Чтобы составить программу решения нашей задачи, нужно научиться программировать циклы на Паскале. Основной циклической структурой явля-

ется *цикл с предусловием* (*цикл-пока*). С помощью этой структуры можно построить любой циклический алгоритм. Оператор цикла с предусловием в Паскале имеет следующий формат:

while <логическое выражение> **do** <оператор>;

Служебное слово **while** означает «пока», **do** — «делать», «*выполнять*».

Оператор, стоящий после слова **do**, называется *телом цикла*. Тело цикла может быть простым или составным оператором, т. е. последовательностью операторов между служебными словами **begin** и **end**.

А теперь запрограммируем на Паскале алгоритм решения нашей задачи (добавим к нему организацию диалога).

```
Program Words;
var F, N, R: integer;
begin
write('Введите число букв');
    readln(N);
    F:=1;
    R:=1;
    while R<=N  do
    begin
         F:=F*R;
         R:=R+1
    end;
    write('Из ',N,' букв можно
          составить ',F,' слов'
end.
```

Снова бросается в глаза схожесть алгоритма на АЯ и программы на Паскале. Обратите внимание на то, что в Паскале нет специальных служебных слов для обозначения конца цикла (так же как и конца ветвления). Во всех случаях, где это необходимо, используются слова **begin** и **end**.

Что такое отладка и тестирование программы

5. Отладка и тестирование. *Под отладкой программы понимается процесс испытания работы программы и исправления обнаруженных при этом ошибок.* Обнаружить ошибки, связанные с нарушением правил записи программы на Паскале (синтаксические и семантические ошибки), помога-

ет используемая *система программирования*. Пользователь получает сообщение об ошибке, исправляет ее и снова повторяет попытку исполнить программу.

Проверка на компьютере правильности алгоритма производится с помощью тестов. ***Тест*** *— это конкретный вариант значений исходных данных, для которого известен ожидаемый результат.* Прохождение теста — необходимое условие правильности программы. На тестах проверяется правильность реализации программой запланированного сценария.

Нашу программу, например, можно протестировать на значении $N = 6$. На экране должно получиться:

Введите число букв: **6**
Из 6 букв можно составить 720 *слов.*

6. Проведение расчетов и анализ полученных результатов — этот этап технологической цепочки реализуется при разработке практически полезных (не учебных) программ. Например, программы «Расчет прогноза погоды». Ясно, что ею будут пользоваться длительное время, и правильность ее работы очень важна для практики. А поэтому в процессе эксплуатации эта программа может дорабатываться и совершенствоваться.

Коротко о главном

Последовательность этапов работы программиста при решении задачи на компьютере называется технологией решения задачи на компьютере. Таких этапов шесть: 1) постановка задачи; 2) математическая формализация; 3) построение алгоритма; 4) составление программы на языке программирования; 5) отладка и тестирование программы; 6) проведение расчетов и анализ полученных результатов.

Количество различных комбинаций из N предметов, получаемых изменением их порядка, называется числом перестановок. Число перестановок равно $N!$ (N-факториал):

$$N! = 1 \cdot 2 \cdot \ldots \cdot N.$$

Любой циклический алгоритм может быть построен с помощью команды «цикл-пока» (цикл с предусловием).

Оператор цикла с предусловием в Паскале:

while <логическое выражение> **do** <оператор>;

Оператор, составляющий тело цикла, может быть простым или составным.

? *Вопросы и задания*

1. Как блок-схемой и на алгоритмическом языке представляется команда цикла с предусловием?
2. Как программируется цикл с предусловием на Паскале?
3. Почему алгоритм вычисления $N!$ должен быть циклическим?
4. Из каких этапов состоит работа программиста по решению задачи на компьютере?
5. Что такое математическая формализация задачи?
6. Что такое отладка программы? Что называется тестом?
7. Составьте алгоритм вычисления суммы всех натуральных чисел, не превышающих заданного натурального числа N. Проверьте алгоритм трассировкой. Напишите программу на Паскале.
8. Дано целое число X и натуральное N. Составьте алгоритм вычисления X^N. Проверьте алгоритм трассировкой. Напишите программу на Паскале.

§ 40 Алгоритм Евклида

Основные темы параграфа:

- *наибольший общий делитель;*
- *идея алгоритма Евклида;*
- *описание алгоритма Евклида блок-схемой;*
- *программа на АЯ и на Паскале.*

Наибольший общий делитель

Рассмотрим следующую задачу: требуется составить программу определения наибольшего общего делителя (НОД) двух натуральных чисел.

Вспомним математику. Наибольший общий делитель двух натуральных чисел — это самое большое натуральное

число, на которое они делятся нацело. Например, у чисел 12 и 18 имеются общие делители: 2, 3, 6. Наибольшим общим делителем является число 6. Это записывается так:

$$\text{НОД}(12, 18) = 6.$$

Обозначим исходные данные как M и N. Постановка задачи выглядит следующим образом:

Дано: M, N
Найти: НОД(M, N).

В данном случае какой-то дополнительной математической формализации не требуется. Сама постановка задачи носит формальный математический характер. Не существует формулы для вычисления НОД(M, N) по значениям M и N. Но зато достаточно давно, задолго до появления ЭВМ, был известен алгоритмический способ решения этой задачи. Называется он ***алгоритмом Евклида***.

Идея алгоритма Евклида

Идея этого алгоритма основана на том свойстве, что если $M>N$, то

$$\text{НОД}(M, N) = \text{НОД}(M - N, N).$$

Иначе говоря, НОД двух натуральных чисел равен НОД их положительной разности (модуля их разности) и меньшего числа.

Легко доказать это свойство. Пусть K — общий делитель M и N ($M > N$). Это значит, что $M = mK$, $N = nK$, где m, n — натуральные числа, причем $m > n$. Тогда $M - N = K(m - n)$, откуда следует, что K — делитель числа $M - N$. Значит, все общие делители чисел M и N являются делителями их разности $M - N$, в том числе и наибольший общий делитель.

Второе очевидное свойство:

$$\text{НОД}(M, M) = M.$$

Для «ручного» счета алгоритм Евклида выглядит так:

1) если числа равны, то взять любое из них в качестве ответа, в противном случае продолжить выполнение алгоритма;

2) заменить большее число разностью большего и меньшего из чисел;
3) вернуться к выполнению п. 1.

Рассмотрим этот алгоритм на примере M=32, N=24:

M	32	8	8	8
N	24	24	16	8

Получили: НОД(32, 24) = НОД(8, 8) = 8, что верно.

Описание алгоритма Евклида блок-схемой

На рис. 6.8 приведена блок-схема алгоритма Евклида.

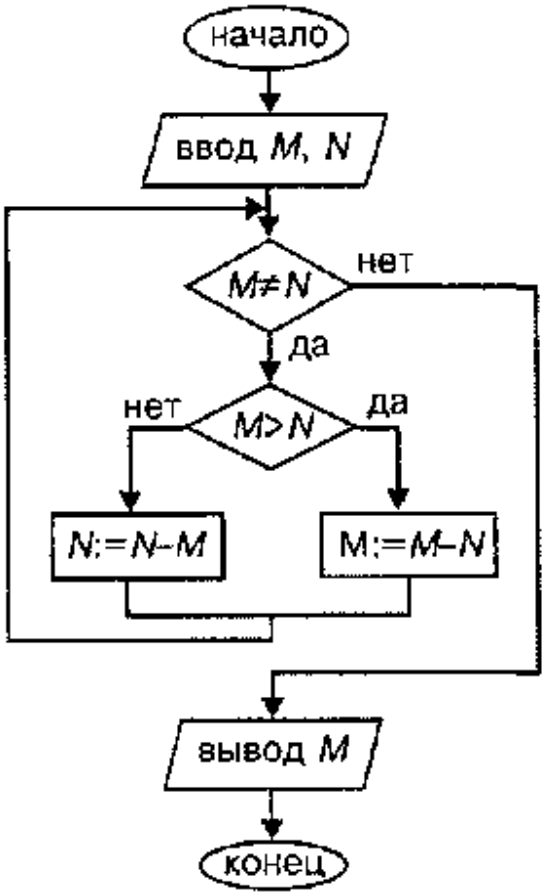


Рис. 6.8. Блок-схема алгоритма Евклида

Структура алгоритма — цикл-пока с вложенным ветвлением. Цикл повторяется, пока значения M и N не равны друг другу. В ветвлении большее из двух значений заменяется на их разность.

А теперь посмотрите на трассировочную таблицу алгоритма для исходных значений M = 32, N = 24.

Шаг	Операция	*M*	*N*	Условие
1	ввод *M*	32		
2	ввод *N*		24	
3	*M*≠*N*			32≠24, да
4	*M*>*N*			32>24, да
5	*M*:=*M*−*N*	8		
6	*M*≠*N*			8≠24, да
7	*M*>*N*			8>24, нет
8	*N*:=*N*−*M*		16	
9	*M*≠*N*			8 ≠16, да
10	*M*>*N*			8>16, нет
11	*N*:=*N*−*M*		8	
12	*M*≠*N*			8≠8, нет
13	вывод *M*	8		
14	конец			

В итоге получился верный результат.

Программа на АЯ и на Паскале

Запишем алгоритм на АЯ и программу на Паскале.

```
алг Евклид
цел M, N
нач
  вывод "Введите M и N"
  ввод M, N
  пока M ≠ N, повторять
   нц
        если M>N
        то M:=M−N
        иначе N:=N−M
        кв
   кц
  вывод "НОД=", M
кон
```

```
Program Evklid;
var  M, N: integer;
begin
writeln('Введите M и N');
     readln(M, N);
      while M<>N do
         begin
            if  M>N
            then  M:=M-N
            else  N:=N-M

         end;
       write('НОД=', M)
end.
```

Коротко о главном

Алгоритм Евклида предназначен для получения наибольшего общего делителя двух натуральных чисел. Структура алгоритма Евклида — цикл с вложенным ветвлением.

Ручная трассировка может использоваться для проверки правильности лишь сравнительно простых алгоритмов. Правильность программ проверяется путем тестирования на компьютере.

? *Вопросы и задания*

1. Выполните на компьютере программу `Evklid`. Протестируйте ее на значениях $M = 32$, $N = 24$; $M = 696$, $N = 234$.
2. Составьте программу нахождения наибольшего общего делителя трех чисел, используя следующую формулу:

$$\text{НОД}(A, B, C) = \text{НОД}(\text{НОД}(A, B), C).$$

3. Составьте программу нахождения наименьшего общего кратного (НОК) двух чисел, используя формулу:

$$A \cdot B = \text{НОД}(A, B) \cdot \text{НОК}(A, B).$$

§ 41 Таблицы и массивы

Основные темы параграфа:

- *что такое массив;*
- *описание и ввод значений в массив на Алгоритмическом языке;*
- *цикл с параметром в АЯ;*
- *расчет среднего значения элементов массива.*

Изучая базы данных, электронные таблицы, вы познакомились с табличным способом организации данных. Вы уже знаете, что большие наборы данных удобно представлять в табличном виде. В таблицах могут храниться данные разных типов. На практике чаще всего приходится встречаться с таблицами, содержащими числовые и символьные (текстовые) данные.

Что такое массив

Представление таблицы в языках программирования называется ***массивом***. Вот, например, таблица, содержащая среднемесячные значения температуры в Перми в 2000 году:

Месяц	1	2	3	4	5	6	7	8	9	10	11	12
Температура	−21	−18	−7,5	5,6	10	18	22,2	24	17	5,4	−7	−18

Такую таблицу называют ***линейной***. Она представляет собой последовательность упорядоченных чисел. Для обозначения этих чисел используют ***индексированные имена***. Например, через *T*[1] обозначается температура в январе (первом месяце года), *T*[5] — температура в мае и т. д.

В программировании линейная таблица называется ***одномерным массивом***. В нашем примере *T* — это имя массива. Элементы массива пронумерованы. Порядковый номер элемента называется его ***индексом***. Каждый элемент массива обозначается индексированным именем в следующей форме:

<имя массива> [<индекс>]

Индекс записывается в квадратных скобках: *T*[2], *T*[10], *T*[12]. Индексы могут представляться не только в виде констант, но и в виде целых переменных и даже выражений целого типа: *T*[*i*], *T*[*k*], *T*[*i*+*k*], *T*[2**k*]. Важно следить, чтобы значения индексов не выходили за допустимые границы. В примере с температурами они должны лежать в диапазоне от 1 до 12.

Все элементы массива должны иметь одинаковый тип. Если массив состоит только из целых чисел, то тип массива — целый. В нашем примере значения температур могут быть дробными, поэтому тип массива — вещественный.

Массив — это пронумерованная конечная последовательность однотипных величин.

Решение задач по обработке массива связано, как правило, с перебором элементов массива. Такой перебор происходит в цикле, в котором изменяется значение индекса от начальной до конечной величины. Для того чтобы организовать ввод исходных данных в массив, нужно также использовать цикл.

Описание и ввод значений в массив на Алгоритмическом языке

Запишем алгоритм ввода значений в массив температур. Сначала посмотрим, как это делается на АЯ. Рассмотрим два варианта алгоритмов на АЯ, использующих разные способы организации цикла.

```
алг Ввод массива, вариант 1
вещ таб T[1:12]
цел I
нач  I:=1
     пока I<=12, повторять
     нц
          вывод ("T[", I , "]=")
          ввод (T[I])
          I:=I+1
     кц
кон
алг Ввод массива, вариант 2
вещ таб T[1:12]
цел I
нач
для I от 1 до 12 шаг 1 повторять
нц
     вывод ("T[", I , "]=")
     ввод (T[I])
кц
кон
```

Обратите внимание на вторую строку алгоритмов. В ней присутствует описание массива температур. В Алгоритмическом языке массив называется таблицей. Запись

вещ таб T[1:12]

описывает таблицу (массив) вещественного типа, имя которого T и элементы пронумерованы от 1 до 12.

Цикл с параметром в АЯ

В первом варианте алгоритма используется уже знакомая вам алгоритмическая структура *цикла с предусловием*. Переменная I играет роль параметра цикла, изменяющегося от 1 до 12 с шагом 1. Внутри цикла она используется в качестве индекса в обозначении элементов цикла: $T[I]$.

Ввод организован в режиме диалога. Вы уже знаете, что это обязательное условие дружественности интерфейса программы. Перед вводом каждого очередного элемента таблицы на экран будет выводиться его имя. Это результат выпол-

нения команды вывод "*T*[", *I* , "]=". После этого программист должен ввести с клавиатуры соответствующее число (команда ввод *T*[*I*]):

T[1]= –21
T[2]= –18
T[3]= –7.5 и т. д.

Во втором варианте используется алгоритмическая структура, которая называется ***«цикл с параметром»***. Ее общая форма такая:

```
для <параметр цикла> от <начальное значение
параметра> до <конечное значение параметра>
шаг <величина приращения параметра> повторять
нц
  <тело цикла>
кц
```

Параметром цикла должна быть переменная целого типа. В нашем примере это переменная *I*. Выполнение тела цикла повторяется для всех последовательных значений параметра от начального до конечного значения включительно с изменением его значения при каждом повторении на величину шага. Следовательно, по второму варианту алгоритма будут выполняться те же самые действия, что и по первому.

Расчет среднего значения элементов массива

	A	B
1	*Месяц*	*Температура*
2	1	–21
3	2	–18
4	3	–7,5
5	4	5,6
6	5	10
7	6	18
8	7	22,2
9	8	24
10	9	17
11	10	5,4
12	11	–7
13	12	–18
14	**Среднее:**	2,56

Рис. 6.9. Таблица температур

Теперь сформулируем задачу обработки массива температур, которую будем решать дальше. Вычислим среднегодовую температуру. Для этого нужно сложить все 12 значений таблицы и разделить сумму на 12. Полученную величину выведем в качестве результата.

Эту задачу легко решить с помощью электронных таблиц. На рис. 6.9 показана такая таблица. В ячейки B2:B13 заносятся значения температур. В ячейку B14 помещается формула: =СРЗНАЧ(B2:B13). Результат вы видите в ячейке B14.

Табличный процессор — это программа, составленная программистами на некотором языке программирования. Вот мы и разберемся, как программируется вычисление среднего значения числового массива, реализованное в функции СРЗНАЧ.

Запишем алгоритм в полном виде (с вводом, вычислениями и выводом), используя в нем для организации циклов структуру цикла с параметром.

```
алг Средняя температура
вещ таб T[1:12]
цел I, вещ Tsred
нач
        {Цикл ввода}
    для I от 1 до 12 шаг 1 повторять
    нц
        вывод "T[", I , "]="
        ввод T[I]
    кц
    {Цикл суммирования}
    Tsred:=0
    для I от 1 до 12 шаг 1 повторять
    нц
        Tsred:=Tsred+T[I]
    кц
    {Вычисление среднего}
    Tsred:=Tsred/12
    вывод("Среднегодовая температура=", Tsred)
кон
```

Обратим внимание на следующие особенности алгоритма. Появилась новая переменная *Tsred*, в которой вычисляется среднее значение:

$$Tsred = (T[1] + T[2] + T[3] + \ldots + T[12])/12.$$

Переменная *Tsred* имеет вещественный тип. Перед циклом суммирования этой переменной присваивается нулевое значение. Так всегда следует поступать с переменной, в которой накапливается сумма какой-то последовательности слагаемых. При каждом повторении цикла к значению переменной *Tsred* добавляется очередное слагаемое. После окончания цикла полученная сумма делится на 12. Это искомый результат, который выводится на экран.

Коротко о главном

Массив — это пронумерованная конечная последовательность однотипных величин.

Линейная таблица в программировании называется одномерным массивом.

В описании массива указывается его тип, имя, границы индексов.

В алгоритмах, связанных с перебором элементов массива, удобно использовать структуру «цикл с параметром».

Вопросы и задания

1. Что такое массив?
2. Самостоятельно придумайте примеры данных, которые можно организовать в виде массива. В каждом примере отметьте: каким именем можно обозначить массив, как пронумеровать его элементы, какой тип будет иметь массив? Опишите массивы по правилам Алгоритмического языка.
3. Для тех же исходных данных, что рассматриваются в параграфе, составьте алгоритм, в котором вычисляются четыре величины: средние температуры зимних месяцев, весенних месяцев, летних месяцев, осенних месяцев.
4. Вы посетили магазин и купили 10 видов товара. В таблицу T[1:10] вы записали количество купленного товара каждого вида. В таблицу C[1:10] записали цены единиц каждого вида товара соответственно. Составьте алгоритм вычисления общей стоимости всех покупок.

§ 42 Массивы в Паскале

Основные темы параграфа:

- *описание и обработка массива на Паскале;*
- *цикл с параметром на Паскале;*
- *форматы вывода;*
- *программа с двумя массивами.*

А теперь посмотрим, как можно на Паскале запрограммировать алгоритм вычисления среднегодовой температуры.

Для этого сначала познакомимся с правилами описания массивов. Заметим, что в данном разделе учебника мы ограничиваемся только работой с одномерными массивами (линейными таблицами).

Описание и обработка массива на Паскале

Общая форма описания одномерного массива на Паскале такая:

```
var <имя массива>: array [<нижняя граница индекса
.. верхняя граница индекса>] of <тип массива>
```

Слово «array» буквально переводится как «массив». Границы индекса могут быть любыми целыми числами. Важно, чтобы нижняя граница была меньше верхней границы. Описание массива температур будет следующим:

```
var T: array [1..12] of real;
```

Цикл с параметром на Паскале

Рассмотрим полный текст программы на Паскале.

```
Program Temperature;
var T: array[1..12] of real;
I: integer;  Tsred: real;
begin
    {Цикл ввода}
    for I:=1 to 12 do
    begin
          write('T[',I:2,']=');
          readln(T[I])
    end;
    {Цикл суммирования}
    Tsred:=0;
    for I:=1 to 12 do
          Tsred:=Tsred+T[I];
    {Вычисление среднего}
    Tsred:=Tsred/12;
    writeln('Среднегодовая температура = ',
            Tsred:6:2, ' градусов')
end.
```

В этой программе дважды использован оператор цикла с параметром. Он имеет следующий формат:

```
for <параметр цикла> := <начальное значение
параметра> to <конечное значение параметра>
do <тело цикла>;
```

Если параметр цикла — целая переменная, то ее значение будет возрастать через единицу. Существует другой вариант этого оператора, в котором вместо слова **to** записывается **downto**. В этом случае значение переметра цикла убывает через единницу. Следовательно, начальное значение в этом случае должно быть больше конечного.

Так же как и для оператора цикла **while**, здесь *тело цикла может быть либо простым оператором, либо составным*. В первом случае тело цикла заканчивается на ближайшей точке с запятой. В нашем примере — это цикл суммирования. Во втором случае тело цикла заключается между словами **begin** и **end** (цикл ввода).

Форматы вывода

В программе присутствует еще один новый для вас элемент Паскаля: формат вывода. Это числа с двоеточиями, стоящие после переменных в операторе вывода `write`:

```
write('T[',I:2,']=');
```

В этой записи `I:2` обозначает, что значение переменной `I` выводится как целое число в две символьные позиции на экране. Для однозначного числа в первой позиции будет помещен пробел, например: `_5`.

В операторе вывода результата также используется формат: `Tsred:6:2`. Значение переменной `Tsred` выводится как смешанное число в 6 позиций, две последние из которых занимает дробная часть. В третьей справа позиции — точка. Лишние позиции для целой части занимаются пробелами. Например: `_34.25`.

Результат выполнения программы `Temperature` будет выведен на экран в следующем виде:

```
Среднегодовая температура = 2.56 градусов
```

Программа с двумя массивами

А теперь расширим условие задачи. Требуется для каждого месяца определить отклонение его средней температуры от среднегодовой величины.

Вернемся к электронной таблице на рис. 6.9. Добавим к ней еще один столбец С, в котором будут вычисляться искомые отклонения. В ячейку С2 занесем формулу =B2–B14. По этой формуле вычислится отклонение январской температуры от среднегодовой. Скопировав эту формулу в ячейки С3:С13, получим все остальные величины. Смысл «замора-

живания» адреса B14 вам должен быть понятен. Результаты приведены в таблице на рис. 6.10.

	A	B	C
1	*Месяц*	*Температура*	*Отклонения*
2	1	−21	−23,56
3	2	−18	−20,56
4	3	−7,5	−10,06
5	4	5,6	3,04
6	5	10	7,44
7	6	18	15,44
8	7	22,2	19,64
9	8	24	21,44
10	9	17	14,44
11	10	5,4	2,84
12	11	−7	−9,56
13	12	−18	−20,56
14	Среднее:	2,56	

Рис. 6.10. Температуры и отклонения от среднего

Реализуем вычисление отклонений в программе на Паскале. Очевидно, в программе должен появиться еще один массив для размещения в нем таблицы отклонений. Дадим этому массиву имя `Dt`. Как и массив температур, он состоит из 12 чисел: `Dt[1]`, `Dt[2]`, `Dt[3]`, ..., `Dt[12]`.

К предыдущей программе надо добавить описание массива `Dt` в следующем виде:

```
var Dt: array[1..12] of real;
```

Значение каждого элемента массива равно разности между температурой соответствующего месяца и среднегодовой температурой. Например, для января: `Dt[1] = T[1] - Tsred`. Такие вычисления повторяются в цикле 12 раз. Значения массива `Dt` выводятся на экран.

Запишем на Паскале фрагмент, который надо вставить в конец предыдущей программы, чтобы решить поставленную задачу.

```
for I:=1 to 12 do
begin
    Dt[I]:= T[I] - Tsred;
    writeln('Dt[',I:2,']=', Dt[I]:6:2)
end
```

Здесь вычисление значений массива Dt и вывод их на экран совмещены в одном цикле. Результат работы программы будет следующим:

```
Dt[ 1]= -23,56
Dt[ 2]= -20,56
Dt[ 3]= -10,06
...
Dt[12]= -20,56
```

Как и следовало ожидать, это те же самые числа, что получены в электронной таблице.

Коротко о главном

Формат описания массива на Паскале:

var <имя массива>: **array** [<нижняя граница индекса .. верхняя граница индекса>] **of** <тип массива>

Формат оператора цикла с параметром:

for <параметр цикла> := <начальное значение параметра> **to** <конечное значение параметра> **do** <тело цикла>;

В формате вывода указывается количество позиций на экране для вывода значения. Для вещественного числа указывается также количество цифр в дробной части.

В программе на Паскале должен быть описан каждый используемый в ней массив.

Вопросы и задания

1. Как можно описать на Паскале массив, в котором будут храниться значения численности населения Москвы к концу каждого года XX века?
2. Вы приобрели котенка. Каждый вечер вы определяете его вес с помощью весов. Как можно описать на Паскале массив, в котором будут храниться значения веса котенка в течение месяца (например, мая)?
3. Напишите фрагмент программы на Паскале ввода исходных данных для массивов, определенных в заданиях 1 и 2.
4. Введите в компьютер программу Temperature, добавив к ней обработку массива Dt. Выполните программу, получите результаты. Сравните их с приведенными в параграфе.
5. Составьте программы на Паскале по алгоритмам из заданий 3, 4 предыдущего параграфа. Выполните эти программы на компьютере.

§ 43 Одна задача обработки массива

Основные темы параграфа:
- *что такое случайные числа;*
- *датчик случайных чисел на Паскале;*
- *алгоритм поиска числа в массиве;*
- *программа поиска числа в массиве.*

Решим следующую задачу. Массив заполняется случайным набором целых чисел. Нужно определить, сколько раз данное целое число входит в этот массив.

Что такое случайные числа

Сначала несколько слов о случайных числах. Все себе представляют игральный кубик, имеющий шесть граней. При каждом бросании кубика выпадение какого-то числа есть случайное событие. С равной вероятностью может выпасть любое число от 1 до 6. Результат бросания кубика — это ***случайное число***. А теперь представьте себе кубик с 10-ю гранями. Правда, кубиком его можно назвать только условно. Это — десятигранник, на каждой грани которого нанесены числа от 1 до 10. Результат бросания такого «кубика» — случайное число в диапазоне от 1 до 10. При розыгрыше лотереи из вращающегося барабана достают пронумерованные шары. Выпавший номер шара — случайное число.

Датчик случайных чисел на Паскале

В языках программирования, как правило, имеется аналог подобного «кубика» или лототрона, позволяющий получать случайные числа. Он называется ***датчиком случайных чисел***. Это стандартная функция. В Паскале она записывается так: `random(X)`. Здесь `X` — целое число. При выполнении функции ее результатом становится целое число в диапазоне от 0 до `X`. Например, если `X = 50`, то в результате можем получить любое целое число от 0 до 50.

Приведем программу, которая демонстрирует работу датчика случайных чисел на Паскале:

```
Program Example;
var I: integer;
```

```
begin
for I:=1 to 10 do
    write(random(50):4)
end.
```

По этой программе на экран выводится десять случайных чисел из диапазона от 0 до 50. Вот результат тестового выполнения этой программы:

```
0   3   17  20  27  7   31  16  37  42
```

А теперь вернемся к условию задачи. Получающиеся с помощью датчика случайные числа «раскладываются» по элементам массива. Назовем массив Rand, а число элементов в нем пусть будет равно 20. Искомое число будет вводиться в переменную X.

Алгоритм поиска числа в массиве

На рис. 6.11 приведена блок-схема алгоритма поиска в массиве *Rand* величины *X* с подсчетом числа его вхождений в массив в переменной *NumberX*.

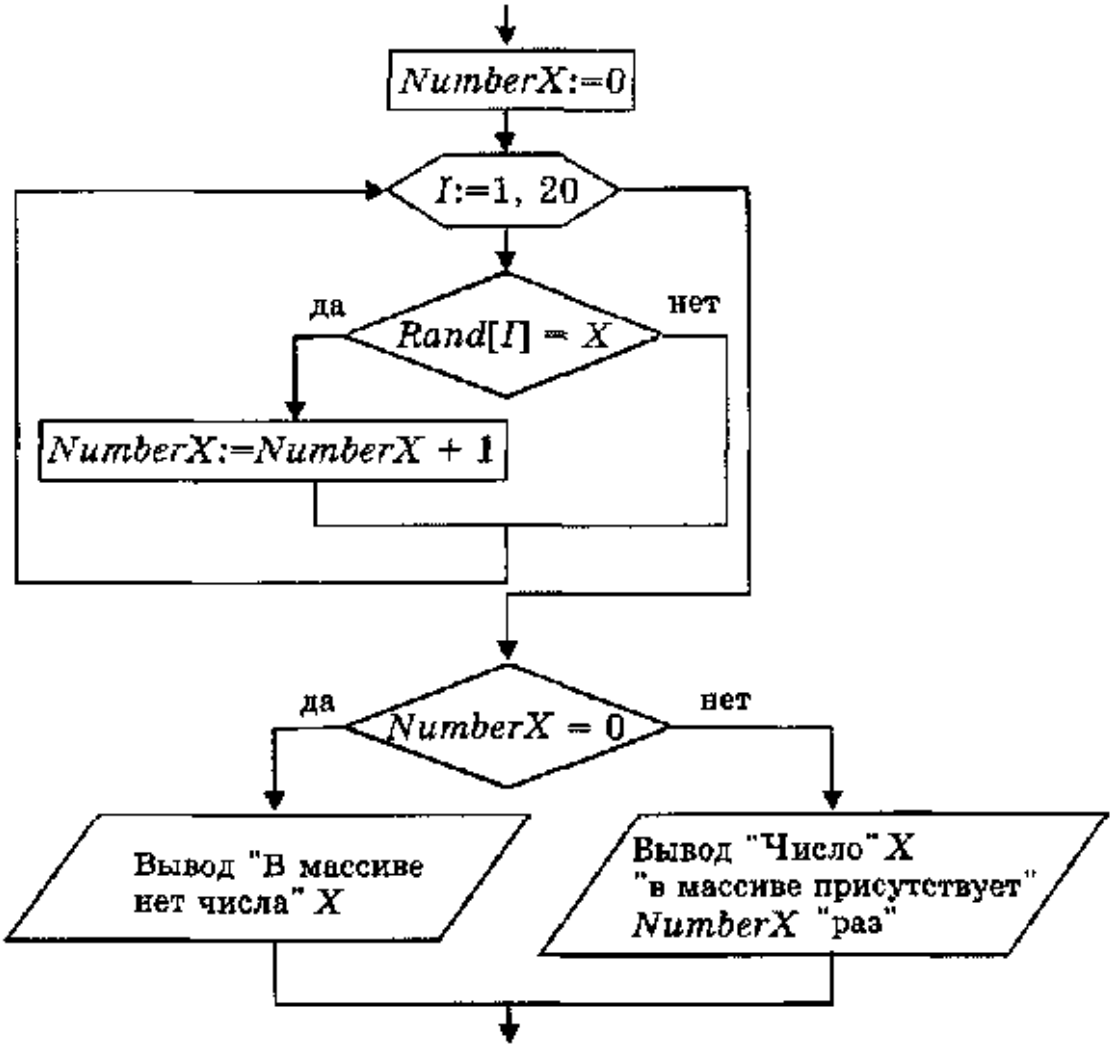


Рис. 6.11. Подсчет вхождений числа в массив

Обратите внимание на блок, отображающий цикл с параметром. Он имеет форму вытянутого шестиугольника. В блоке записывается параметр цикла (переменная *I*), начальное и конечное значения параметра через запятую (:=1, 20).

Переменная *NumberX* играет роль счетчика. Вначале ей присваивается ноль. Затем в цикле происходит перебор всех элементов массива, и при каждом выполнении условия равенства к счетчику добавляется единица. Так всегда организуются счетчики в программах! В результате выполнения программы на экран будет выведен один из двух вариантов ответа: либо сообщение, что в массиве нет искомого числа, либо сообщение о том, сколько раз это число присутствует в массиве, если оно там обнаружено.

Программа поиска числа в массиве

Напишем программу на Паскале, содержащую как заполнение массива случайными числами, так и алгоритм, описанный в блок-схеме на рис. 6.11.

```
Program Example2;
var Rand: array[1..20] of integer;
    I, X, NumberX : integer,
begin
    {Установка датчика случайных чисел}
    Randomize;
    {Заполнение массива случайными числами
    и вывод их на экран}
    writeln('Массив случайных чисел: ')
    for I:=1 to 20 do
    begin
       Rand[I]:=random(50); Write(Rand[I]:4)
    end;
    writeln;
    {Ввод X}
    write('Введите X:')); Readln(X);
    {Подсчет числа вхождений X в массив}
    NumberX:=0;
    for I:=1 to 20 do
    if Rand(I)=X then NumberX:=NumberX+1;
    {Анализ и вывод результата}
    if NumberX=0
    then writeln('В массиве нет числа ',X)
    else writeln('Число ',X,
    ' в массиве присутствует ',NumberX,' раз')
end.
```

В этой программе присутствует еще один новый для нас оператор: `Randomize`. Это стандартная процедура Паскаля, которая производит *установку начального состояния датчика случайных чисел*. Дело в том, что без этого оператора функция `random` при многократном повторении выполнения программы всегда будет выдавать одну и ту же последовательность чисел. Процедура `Randomize` *случайным образом* устанавливает начальное состояние датчика. Поэтому при повторном выполнении программы будут получаться разные наборы случайных чисел.

Посмотрите на результаты выполнения этой программы. Первое выполнение:

```
Массив случайных чисел:
5 44  0 41 29 12  1 39 32 25 17 31 44  5  5 16 29 10
13 34
Введите X: 5
Число 5 в массиве присутствует 3 раз
```

Жирным шрифтом обозначено вводимое с клавиатуры значение. Все остальные символы выводятся на экран по программе. Второе выполнение программы:

```
Массив случайных чисел:
41 33 26 26 33 11 19 38 29  4 31 20  6 32 39 21 28
40  7 19
Введите X: 2
В массиве нет числа 2
```

Коротко о главном

Случайные числа — результаты случайного выбора из конечного множества значений (игровой кубик, жребий, лотерея).

Функция `random(X)` — датчик случайных чисел в диапазоне от 0 до `X` на Паскале.

Для подсчета количества искомых величин используется переменная-счетчик.

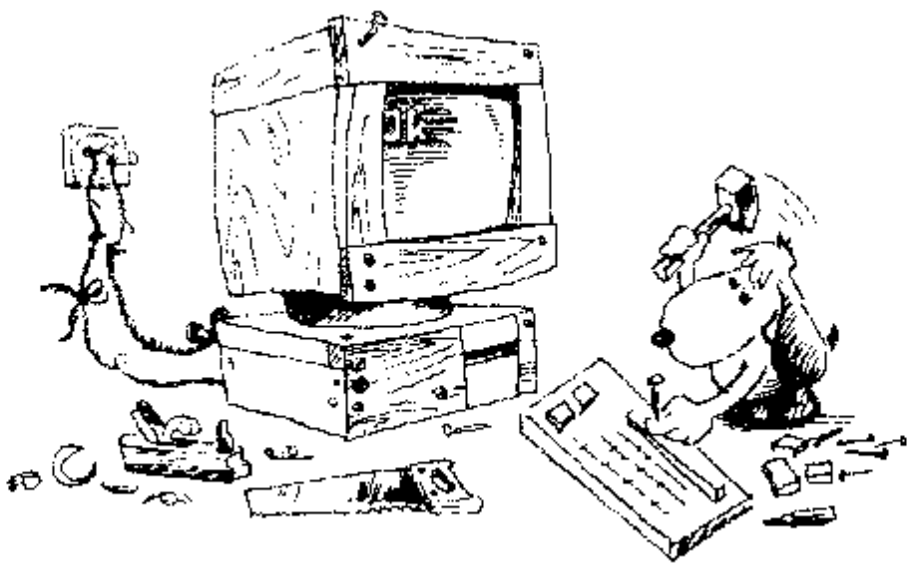
Вопросы и задания

1. Придумайте свои способы получения случайных чисел.
2. Какие значения может принимать целая переменная `Y`, если в программе записано: `Y:=10+random(5)`?

3. Как, используя функцию `random(X)`, можно получать числа в диапазонах: от 1 до 10, от −10 до +10, от 50 до 100?
4. Введите в компьютер программу `Example2`. Выполните программу, получите результаты.
5. Составьте программу заполнения массива из 100 чисел случайными значениями из диапазона от −20 до 20. Подсчитайте в этом массиве количество положительных и количество отрицательных значений.
6. Заполните случайными числами в диапазоне от 1 до 5 два массива: $A[1:20]$ и $B[1:20]$. Найдите и выведите на экран только те элементы этих массивов, значения которых совпадают. Например, если $A[2] = B[2] = 4$, то на экран надо вывести:
   Номер: 2 значение: 4
   Если таких совпадений нет, то вывести на экран сообщение об этом.

Чему вы должны научиться, изучив главу 6

1. Строить несложные вычислительные алгоритмы с использованием блок-схем и Алгоритмического языка.
2. Выполнять трассировку алгоритмов.
3. Составлять программу на Паскале по данному алгоритму.
4. Работать с системой программирования на Паскале: набирать текст программы; сохранять программу на диске и вызывать ее с диска; компилировать и исполнять программу; исправлять ошибки в программе.



ПРОГРАММИРОВАНИЕ

Алгоритмы работы с величинами

Данные

- **Константы:** *2,5 -0,1 345*
- **Переменные:** *цел M, N, K*
  вещ A, B, X
- **Таблицы:** *вещ таб T[1:12]*

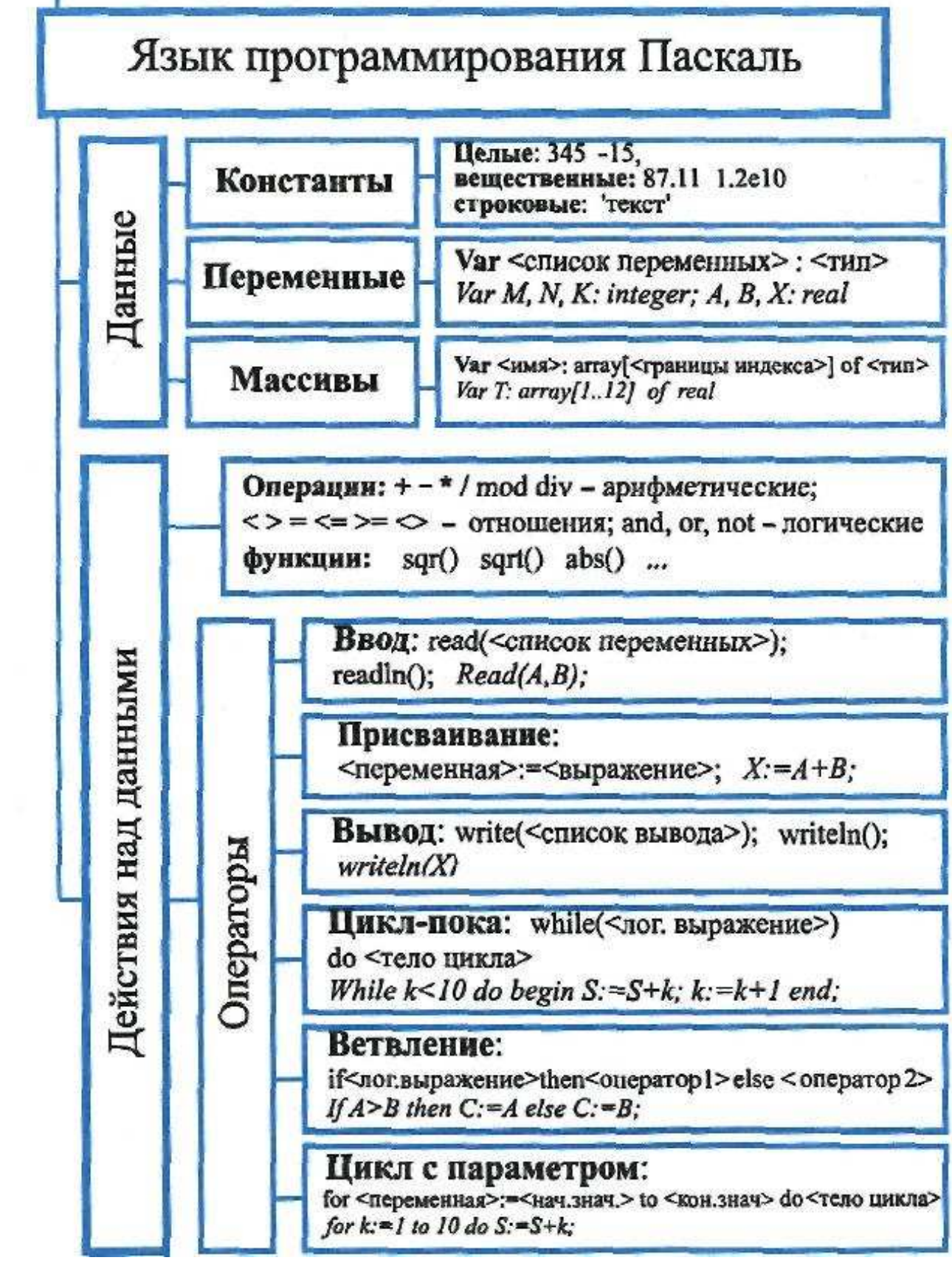
Действия над данными

- **Операции:** арифметические, отношений, логические
  Функции $\sqrt{\ }$, |x|

Команды

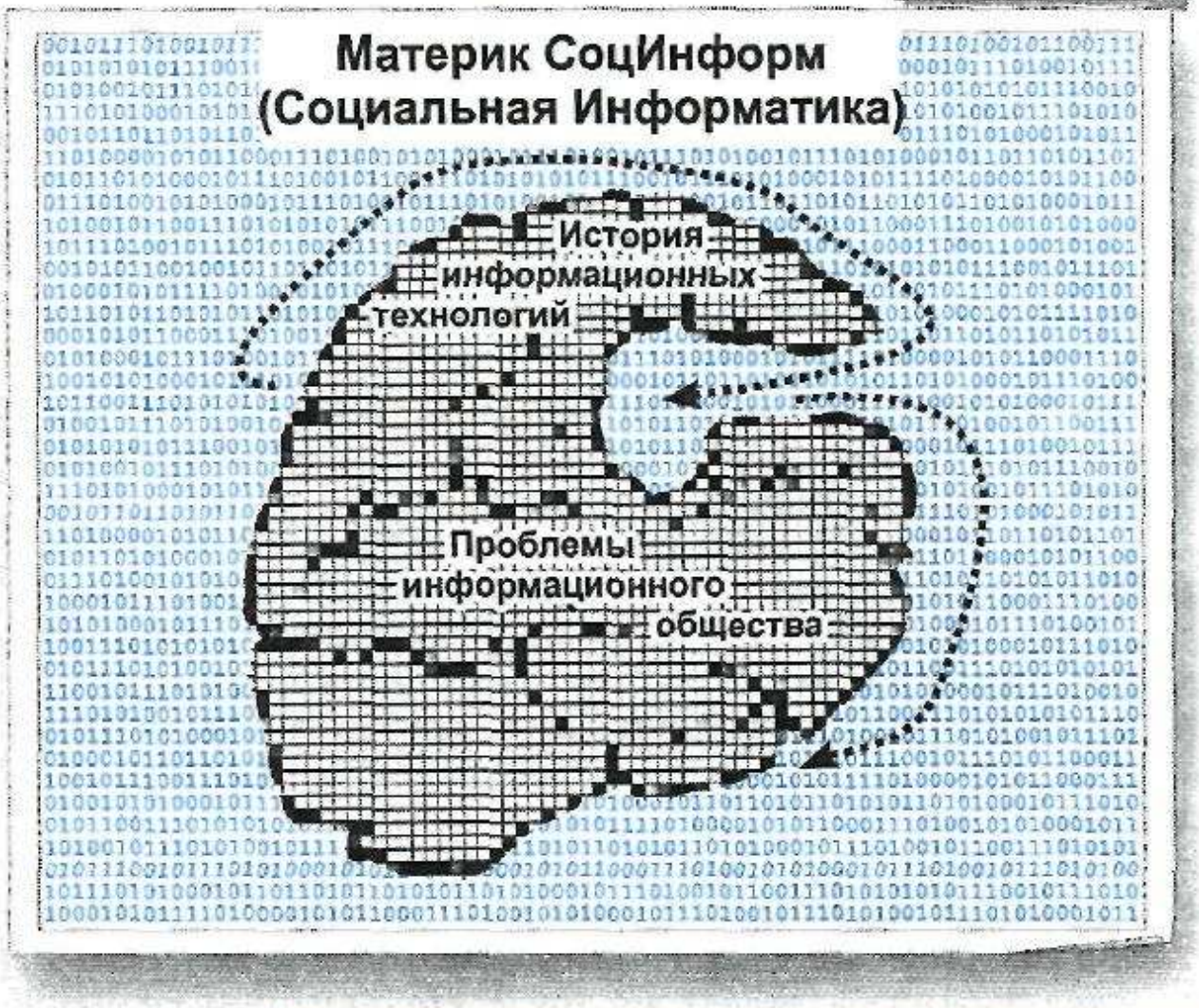
- **Ввод**: *ввод A, B*
- **Присваивание**: *X:=A+B*
- **Вывод**: *вывод X*
- **Цикл-пока**: *пока k<10 повторять*
  нц S:=S+k; k:=k+1 кц
- **Ветвление**: *если A>B то ... иначе*
- **Цикл с параметром**: *для k от 1 до 10*
  повторять нц S:=S+k кц

Система основных понятий главы 6



Глава 7

Информационные технологии и общество



Здесь вы узнаете

- историю информационных технологий и компьютерной техники
- что такое информационное общество
- что такое информационные ресурсы общества
- как решаются проблемы информационной безопасности

§ 44 Предыстория информатики

Основные темы параграфа:

- *история средств хранения информации;*
- *история средств передачи информации;*
- *история средств обработки информации;*
- *машина Бэббиджа — предшественница ЭВМ.*

В любой деятельности человек всегда придумывал и создавал самые разнообразные средства, приспособления, орудия труда. Все это облегчало труд, делало его производительнее, расширяло возможности людей. Известно, что история материального производства и мировой науки тесно связана с историей развития орудий труда.

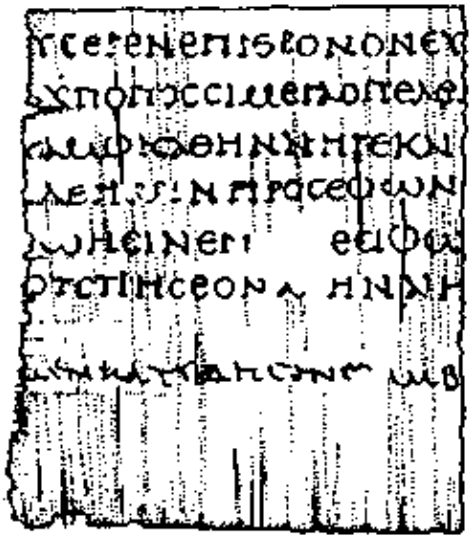
Первые вспомогательные средства для работы с информацией появились много позже первых орудий материального труда. Историки утверждают, что расстояние во времени между появлением первых инструментов для физического труда (топор, ловушка для охоты) и инструментов для регистрации информационных образов (на камне, кости) составляет около миллиона лет!

Следовательно, почти 99% времени существования человека на Земле труд носил только материальный характер.

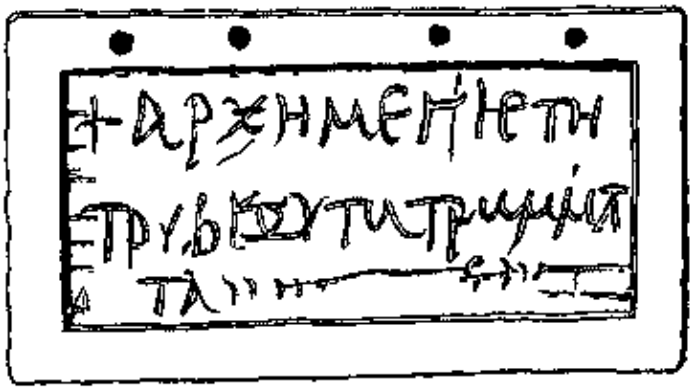
Уже говорилось о том, что информационную деятельность человека можно разделить на три составляющие: ***хранение, передачу*** и ***обработку***. Долгое время средства информационного труда развивались отдельно по этим трем направлениям.

История средств хранения информации

История хранения информации в письменной форме уходит в глубь веков. До наших дней в некоторых местах сохранились наскальные письмена древнего человека, выполненные 25–20 тысяч лет назад; лунный календарь, выгравированный на кости 20 тысяч лет назад. Для письма также использовались дерево, глина. Многие века письменные документы составлялись на пергаментных свитках. Это было «очень дорогим удовольствием». Пергамент делался из кожи животных. Ее растягивали, чтобы получить тонкие листы. Когда на востоке научились ткать шелк, его стали использовать не только для одежды, но и для письма.



Папирус. I–II века нашей эры. Отрывок из поэмы «Илиада» древнегреческого поэта Гомера



Школьная восковая табличка. Сохранившиеся таблички происходят в основном из Египта, Дакии, Помпеи и Геркуланума. Деревянные таблички покрывались цветным воском, надписи на нем процарапывались стилом – острым предметом

Глиняный диск. XVII век до нашей эры. Найден в городе Фест на Крите. По обеим сторонам диска спиралью «бегут» так и не расшифрованные письмена

«Апостол». 1564 год. Первая русская датированная печатная книга, выпущенная в Москве Иваном Федоровым

Во II веке нашей эры в Китае изобрели бумагу. Однако до Европы она дошла только в XI веке. Вплоть до XV века письма, документы, книги писались вручную. В качестве инструмента для письма использовались кисточки, перья птиц, позже — металлические перья; изобретались краски, чернила. Книг было очень мало, они считались предметами роскоши.

Письменный прибор древнеегипетского писца состоял из заостренных палочек для письма с расщепленными кончиками, которые складывались в пенал, тушницы для туши красного и черного цветов и мешочка с песком

В середине XV века немецкий типограф Иоганн Гутенберг изобрел первый печатный станок. С этого времени началось книгопечатание. На Руси книгопечатание основал Иван Федоров в середине XVI века. Книг стало значительно больше, быстро росло число грамотных людей.

До сегодняшнего дня лист бумаги остается основным носителем информации. Но у него появились серьезные «конкуренты».

В XIX веке была изобретена фотография. Носителями видеоинформации стали фотопленка и фотобумага.

В 1895 году французы братья Люмьер продемонстрировали в Париже первый в мире кинофильм, используя аппарат собственного изобретения. Этот год считается годом рождения ***кино***.

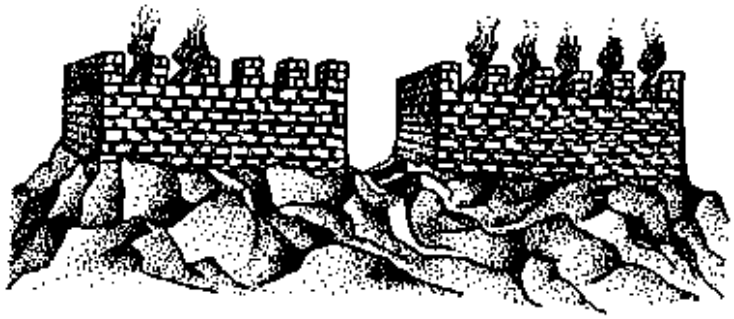
История технических средств хранения и воспроизведения звука начинается с 1877 года. В этом году в США Томасом Эдисоном был построен ***фонограф***. Звук механическим способом — с помощью записывающей иглы — наносился на поверхность вращающегося барабана, покрытого воском. Немного позднее был создан механический ***граммофон***, а затем его портативный вариант — ***патефон***, воспроизводящие звук, записанный на целлулоидной грампластинке. Электрический аналог патефона — ***электрофон*** был изобретен в XX веке. В XX веке был изобретен ***магнитофон***. И совсем недавно на магнитную ленту научились записывать не только звук, но и изображение: появился ***видеомагнитофон***.

История средств передачи информации

Первоначально люди пользовались лишь средствами ближней связи: речью, слухом, зрением. Развитие письменности породило первое средство дальней связи — *почту*.

Для быстрой передачи каких-то важных сведений часто использовались оригинальные идеи. Известно, например, применение на Кавказе ***костровой связи***. Два костровых сигнальщика находились на расстоянии прямой видимости на возвышенных местах или башнях. Когда приближалась опасность (нападали враги), сигнальщики, зажигая цепочку костров, предупреждали об этом население. В XVIII веке возник семафорный телеграф, это тоже световая связь, но технически более совершенная.

Почтовый курьер инков
Такой гонец пробегал около полутора километров и передавал устное послание. Чтобы оно не задерживалось, он еще издали оповещал о своем приближении, подавая звуковые сигналы голосом или трубя в раковину



Факельный телеграф. В Древней Греции пользовались оптической сигнальной связью: днем — дымом, ночью — огнем

Очень богатым на открытия в области связи был XIX век. В этом веке люди овладели электричеством, которое породило множество изобретений. Сначала П. Л. Шеллинг в России в 1832 году изобрел ***электрический телеграф***. А в 1837 году американец С. Морзе создал ***электромагнитный телеграфный аппарат*** и придумал специальный телеграфный код — азбуку, которая носит его имя.

В 1876 году американец А. Белл изобрел ***телефон***. И наконец, в 1895 году русский изобретатель А. С. Попов открыл эпоху ***радиосвязи***.

Самым замечательным изобретением XX века в области связи можно назвать ***телевидение***. Освоение космоса привело к созданию ***спутниковой связи***.

История средств обработки информации

Теперь познакомимся со средствами обработки информации. Важнейшим видом такой обработки являются вычисления. Появление и развитие счетных инструментов стимулировали развитие земледелия, торговли, мореплавания, астрономии и многих других областей практической и научной деятельности людей.

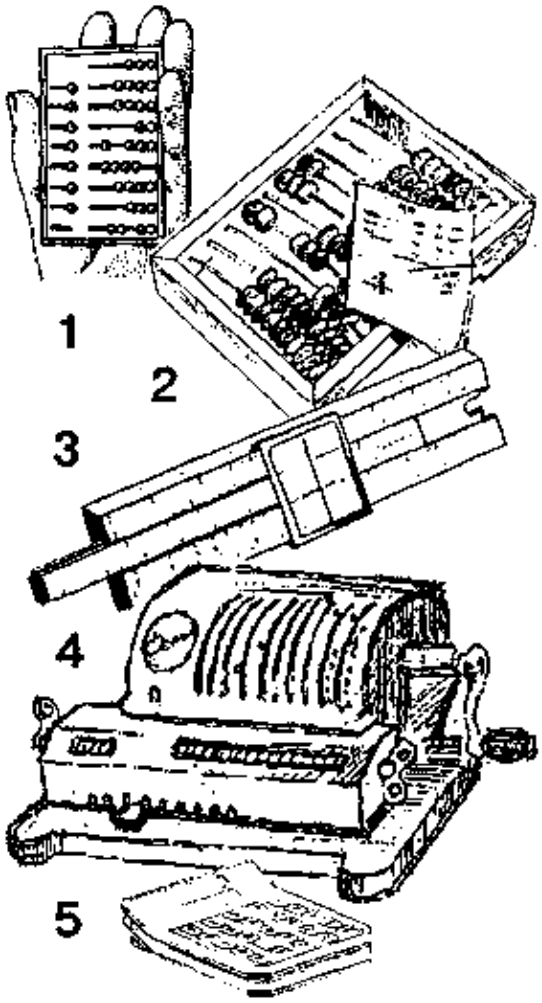
Нетрудно догадаться, что первым счетным средством для человека были его ***пальцы***. Этот инструмент всегда «под рукой»! Кто из вас им не пользовался?

Вот как описывает пальцевой счет туземцев Новой Гвинеи знаменитый русский путешественник Н. Н. Миклухо-Маклай: «...папуас загибает один за другим пальцы руки, причем издает определенный звук, например «бе, бе, бе»... Досчитав до пяти, он говорит «ибон-бе» (рука). Затем он загибает пальцы другой руки, снова повторяет «бе, бе»... пока не дойдет до «ибон али» (две руки). Затем он идет дальше, приговаривая «бе, бе»... пока не дойдет до «самба-бе» и «самба-али» (одна нога, две ноги). Если нужно считать дальше, папуас пользуется пальцами рук и ног кого-нибудь другого».

В V веке до нашей эры в Греции и Египте получил распространение ***абак***. «Абак» — это греческое слово, которое переводится как «счетная доска». Вычисления на абаке производились перемещением камешков по желобам на мраморной доске.

Подобные счетные инструменты распространялись и развивались по всему миру. Например, китайский вариант абака назывался суан-пан. «Потомком» абака можно назвать и русские счеты. В России они появились на рубеже XVI — XVII веков. И до сих пор в нашей стране счеты можно увидеть не только в музеях. До недавнего времени они активно использовались, преимущественно в торговле.

В начале XVII века шотландский математик Джон Непер ввел понятие логарифма, опубликовал таблицы логарифмов. Затем в течение двух веков развивались вычислительные инструменты, основанные на использовании этой математической функции. Логарифмы позволяют свести трудоемкие арифметические операции — умножение и деление — к более простым — сложению и вычитанию. В результате появилась ***логарифмическая линейка***. Этот инструмент до недавнего времени был вычислительным средством инженеров. И лишь ближе к концу XX столетия его вытеснили электронные калькуляторы.



1 — греческий абак;
2 — счеты;
3 — логарифмическая линейка;
4 — арифмометр;
5 — калькулятор

В 1645 году французский математик Блез Паскаль создал первую счетную машину. ***Машина Паскаля*** позволяла быстро выполнять сложение многозначных чисел.

Немецкий ученый Лейбниц, развив идею Паскаля, создал ***механический арифмометр***, на котором можно было выполнять все четыре арифметические операции с многозначными числами. Позднее арифмо-

метр многократно совершенствовался, в том числе и русскими изобретателями П. Л. Чебышевым и В. Т. Однером.

Арифмометр был предшественником современного *калькулятора* — маленького электронно-вычислительного устройства. Сейчас практически у каждого школьника есть калькулятор, который помещается в кармане. Любому академику начала XX века такое устройство показалось бы фантастическим.

Машина Бэббиджа — предшественница ЭВМ

Арифмометр, как и простой калькулятор, — это средство механизации вычислений. Человек, производя вычисления на таком устройстве, сам управляет его работой, определяет последовательность выполняемых операций. Мечтой изобретателей вычислительной техники было создание считающего автомата, который бы без вмешательства человека производил расчеты по заранее составленной программе.

Автором первого проекта вычислительного автомата был профессор Кембриджского университета Чарльз Бэббидж

В период между 1820 и 1856 годами Бэббидж работал над созданием программно управляемой *Аналитической машины*. Это было настолько сложное механическое устройство, что проект так и не был реализован.

Можно сказать, что Бэббидж опередил свое время. Для осуществления его проекта в ту пору еще не существовало подходящей технической базы. Некоторым ученым современникам Бэббиджа его труд казался бесплодным. Однако пророчески звучат сейчас слова самого Чарльза Бэббиджа: «Природа научных знаний такова, что малопонятные и совершенно бесполезные приобретения сегодняшнего дня становятся популярной пищей для будущих поколений».

Основные идеи, заложенные в проекте Аналитической машины, в нашем веке были использованы конструкторами ЭВМ. Все главные компоненты современного компьютера присутствовали в конструкции аналитической машины: это СКЛАД (в современной терминологии — память), где хранятся исходные числа и промежуточные результаты; МЕЛЬНИЦА (арифметическое устройство), в которой осуществляются операции над числами, взятыми из склада;

КОНТОРА (устройство управления), производящая управление последовательностью операций над числами соответственно заданной программе; БЛОКИ ВВОДА исходных данных и ПЕЧАТИ РЕЗУЛЬТАТОВ.

Для программного управления Аналитической машиной использовались перфокарты — картонные карточки с пробитыми в них отверстиями (перфорацией). Перфокарты были изобретены в начале XIX века во Франции Жозефом М. Жаккардом для управления работой автоматического ткацкого станка.

Интересным историческим фактом является то, что первую программу для машины Бэббиджа в 1846 году написала Ада Лавлейс — дочь великого английского поэта Джорджа Байрона.

Аналитическая машина Бэббиджа — это уже универсальное средство, объединяющее в себе обработку информации, хранение информации и обмен исходными данными и результатами с человеком.

? Вопросы и задания

1. Какие средства хранения информации были первыми?
2. Когда появилось книгопечатание, кто его изобретатель?
3. Какие средства хранения информации изобретены в XIX — XX вв.?
4. Назовите основные технические средства передачи информации в порядке их изобретения.
5. Перечислите основные вычислительные средства в хронологической последовательности их изобретения.
6. Кто, когда и где разработал первый проект автоматической вычислительной машины?
7. Какое влияние проект Аналитической машины оказал на дальнейшее развитие вычислительной техники?

§ 45 История чисел и систем счисления

Основные темы параграфа:
- *непозиционные системы древности;*
- *позиционные системы.*

О системах счисления (двоичной и десятичной) говорилось в § 16.

***Система счисления** — это способ изображения чисел и соответствующие ему правила действия над числами.*

Разнообразные системы счисления, которые существовали раньше и которые используются в наше время, можно разделить на непозиционные и позиционные.

Непозиционные системы древности

В древние времена, когда люди начали считать, появилась потребность в записи чисел. Первоначально количество предметов отображали равным количеством каких-нибудь значков: насечек, черточек, точек.

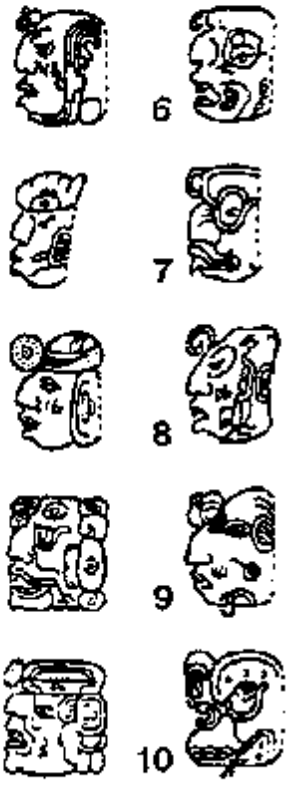
Изучение археологами «записок» времен палеолита на кости, камне, дереве показало, что люди стремились группировать отметки по 3, 5, 7, 10 штук. Такая группировка облегчала счет. Люди учились считать не только единицами, но и тройками, пятерками и пр. Поскольку первым вычислительным инструментом у человека были пальцы, поэтому и счет чаще всего вели группами по 5 или по 10 предметов.

В дальнейшем свое название получили десяток десятков (сотня), десяток сотен (тысяча) и т. д. Такие узловые числа для удобства записи стали обозначать особыми значками — цифрами. Если при подсчете предметов их оказывалось 2 сотни, 5 десятков и еще 4 предмета, то при записи этой величины дважды повторяли знак сотни, пять раз — знак десятков и четыре раза знак единицы.

В таких системах счисления от положения знака в записи числа не зависит величина, которую он обозначает; поэтому они называются *непозиционными системами счисления.*

Непозиционными системами пользовались древние египтяне, греки, римляне и некоторые другие народы древности.

Цифры майя



До нас дошла римская система записи чисел (римские цифры), которая в некоторых случаях применяется в нумерации (века, тома в собрании сочинений, главы книги). В римской системе в качестве цифр используются латинские буквы:

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

Например, число CCXXXII складывается из двух сотен, трех десятков и двух единиц и равно двумстам тридцати двум.

Если слева в записи римского числа стоит меньшая цифра, а справа — большая, то их значения вычитаются, в остальных случаях значения складываются.

$$VI = 5 + 1 = 6, \quad \text{а} \quad IV = 5 - 1 = 4.$$
$$MCMXCVII = 1000 + (-100 + 1000) + (-10 + 100) + 5 + 1 + 1 = 1997.$$

На Руси вплоть до XVIII века использовалась непозиционная система славянских цифр. Буквы кириллицы (славянского алфавита) имели цифровое значение, если над ними ставился специальный знак ~ (титло). Например: $\bar{А}$ — 1, $\bar{Д}$ — 4, $\bar{Р}$ — 100. Интересно, что существовали обозначения очень больших величин. Самая большая величина называлась «колода» и обозначалась знаком $\boxed{А}$. Это число равно 10^{50}. Считалось, что «боле сего несть человеческому уму разумевати».

Непозиционные системы счисления были более или менее пригодны для выполнения сложения и вычитания, но совсем не удобны при умножении и делении.

Позиционные системы

Впервые идея позиционной системы счисления возникла в Древнем Вавилоне.

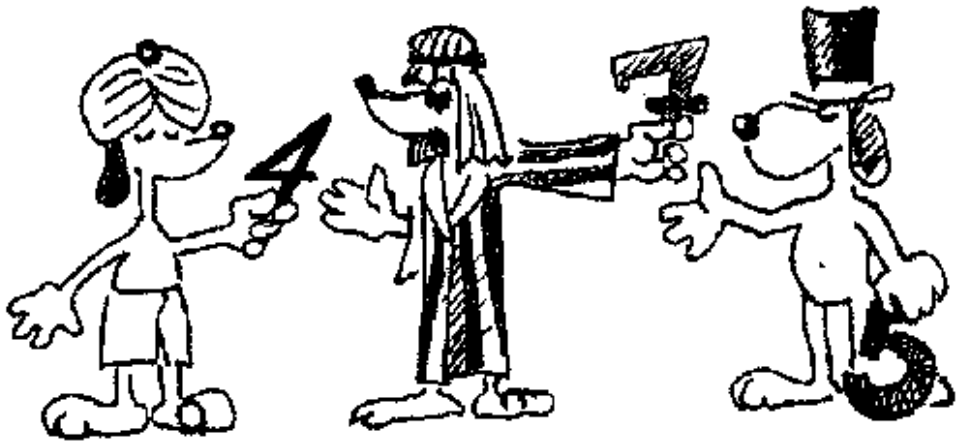
В позиционных системах счисления количественное значение, обозначаемое цифрой в записи числа, зависит от позиции цифры в числе.

Основание позиционной системы счисления равно количеству используемых в системе цифр.

Система счисления, применяемая в современной математике, является позиционной десятичной системой. Ее основание равно десяти, так как запись любых чисел производится с помощью десяти цифр:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Хотя десятичную систему принято называть арабской, но зародилась она в Индии в V веке. В Европе об этой системе узнали в XII веке из арабских научных трактатов, которые были переведены на латынь. Этим и объясняется название «арабские цифры». Однако широкое распространение в науке и в обиходе десятичная позиционная система получила только в XVI веке. Эта система позволяет легко выполнять любые арифметические вычисления, записывать сколь угод-



но большие числа. Распространение арабской системы дало мощный толчок развитию математики.

С позиционной десятичной системой счисления вы знакомы с раннего детства, только, возможно, не знали, что она так называется.

Что означает свойство позиционности системы счисления, легко понять на примере любого многозначного десятичного числа. Например, в числе 333 первая тройка означает три сотни, вторая — три десятка, третья — три единицы. Одна и та же цифра в зависимости от позиции в записи числа обозначает разные значения.

$$333 = 3 \cdot 100 + 3 \cdot 10 + 3.$$

Еще пример:

$$32\,478 = 3 \cdot 10\,000 + 2 \cdot 1000 + 4 \cdot 100 + 7 \cdot 10 + 8 =$$
$$= 3 \cdot 10^4 + 2 \cdot 10^3 + 4 \cdot 10^2 + 7 \cdot 10^1 + 8 \cdot 10^0.$$

Отсюда видно, что всякое десятичное число можно представить как сумму произведений составляющих его цифр на соответствующие степени десятки. То же самое относится и к десятичным дробям.

$$26{,}387 = 2 \cdot 10^1 + 6 \cdot 10^0 + 3 \cdot 10^{-1} + 8 \cdot 10^{-2} + 7 \cdot 10^{-3}.$$

Очевидно, число «десять» — не единственно возможное основание позиционной системы. Известный русский математик Н. Н. Лузин так выразился по этому поводу: «Преимущества десятичной системы не математические, а зоологические. Если бы у нас на руках было не десять пальцев, а восемь, то человечество пользовалось бы восьмеричной системой».

За основание позиционной системы счисления можно принять любое натуральное число, большее 1. Упомянутая выше вавилонская система имела основание 60. Следы этой системы сохранились до наших дней в порядке счета единиц времени (1 час = 60 минут, 1 минута = 60 секунд).

Для записи чисел в позиционной системе с основанием n нужно иметь *алфавит* из n цифр. Обычно для этого при $n \le 10$ используют n первых арабских цифр, а при $n > 10$ к десяти арабским цифрам добавляют буквы.

Вот примеры алфавитов нескольких систем:

Основание	Система	Алфавит
$n = 2$	Двоичная	0 1
$n = 3$	Троичная	0 1 2
$n = 8$	Восьмеричная	0 1 2 3 4 5 6 7
$n = 16$	Шестнадцатеричная	0 1 2 3 4 5 6 7 8 9 A B C D E F

Основание системы, к которой относится число, обычно обозначается подстрочным индексом к этому числу:

$$101101_2,\ 3671_8,\ 3B8F_{16}.$$

А как строится ряд натуральных чисел в разных позиционных системах счисления? Происходит это по тому же принципу, что и в десятичной системе. Сначала идут однозначные числа, потом двузначные, затем трехзначные и т. д. Самое большое однозначное число в десятичной системе — 9. Затем следуют двузначные числа — 10, 11, 12, ... Самое большое двузначное число — 99, далее идут 100, 101, 102 и т. д. до 999, затем 1000 и т. д.

Для примера рассмотрим пятеричную систему. В ней ряд натуральных чисел выглядит так:

1, 2, 3, 4, 10, 11, 12, 13, 14, 20, 21, 22, 23, 24, 30, 31, 32, 33, 34, 40, 41, 42, 43, 44, 100, 101,..., 444, 1000, ...

Видно, что здесь число цифр «нарастает» быстрее, чем в десятичной системе. Быстрее всего число цифр растет в двоичной системе счисления. В следующей таблице сопоставляются начала натуральных рядов десятичных и двоичных чисел:

10	1	2	3	4	5	6	7	8	9	10	11
2	1	10	11	100	101	110	111	1000	1001	1010	1011

Коротко о главном

Система счисления — это определенный способ записи чисел и соответствующие правила действия над числами.

Системы счисления бывают позиционными и непозиционными. Примером непозиционной системы является римская система записи чисел.

В позиционной системе счисления количественное значение каждой цифры зависит от позиции цифры в числе.

Алфавит системы счисления — множество цифр, используемых в ней. Основание системы счисления равно мощности алфавита (числу цифр).

Наименьшее возможное основание позиционной системы счисления — 2. Такая система называется двоичной.

Арабская система записи чисел является десятичной, позиционной.

? Вопросы и задания

1. Что такое система счисления?
2. В чем основное различие позиционных и непозиционных систем счисления?
3. Чему равно основание системы счисления?
4. Почему арабская система записи чисел называется десятичной позиционной?
5. Каково наименьшее основание для позиционной системы?
6. Чему в десятичной системе счисления равны следующие числа, записанные римскими цифрами:
   XI; IX; LX; CLX; MDCXLVIII?
7. Запишите римскими цифрами числа, равные следующим десятичным:
   13; 99; 666; 444; 1692.
8. Запишите последовательность двадцати чисел натурального ряда, начиная от единицы, для позиционных систем с основаниями 2, 3, 5, 8. Оформите результаты в виде таблицы:

$n = 10$	1	2	3	...	19	20
$n = 2$						
$n = 3$						
$n = 5$						
$n = 8$						

9. Постройте таблицы умножения для однозначных чисел в двоичной и троичной системах счисления.

§ 46 История ЭВМ

Основные темы параграфа:

- *счетно-перфорационные и релейные машины;*
- *начало эпохи ЭВМ;*
- *четыре поколения ЭВМ;*
- *перспективы пятого поколения.*

В § 44 рассказывалось о проекте Чарльза Бэббиджа — первой в истории попытке создания программно управляемого вычислительного автомата. Бэббиджу так и не удалось построить свою Аналитическую машину, используя техническую базу середины XIX столетия. К концу XIX — началу XX века с развитием электротехники появилась возможность снова вернуться к этой проблеме.

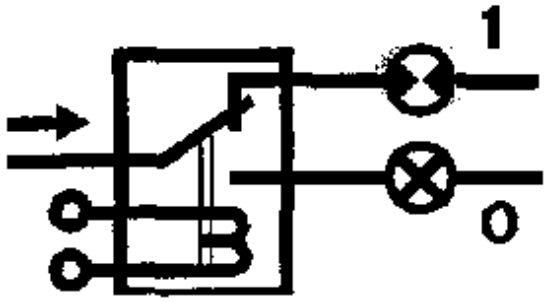
Счетно-перфорационные и релейные машины

В конце XIX века Герман Холлерит в Америке изобрел *счетно-перфорационные машины*. В них, так же как и в Аналитической машине Бэббиджа, использовались перфокарты, но только не для представления программы, а для хранения числовой информации. Каждая такая машина могла выполнять только одну определенную программу, манипулируя с перфокартами и числами, пробитыми на них. Счетно-перфорационные машины осуществляли перфорацию, сортировку, суммирование, вывод на печать числовых таблиц. На этих машинах удавалось решать многие типовые задачи статистической обработки, бухгалтерского учета и другие.

Г. Холлерит основал фирму по выпуску счетно-перфорационных машин, которая затем была преобразована в фирму IBM — ныне самого известного в мире производителя компьютеров.

Непосредственными предшественниками ЭВМ были ***релейные вычислительные машины***. К 30-м годам XX века получила большое развитие релейная автоматика.

В процессе работы релейной машины происходят переключения тысяч реле из одного состояния в другое.



Электромеханическое реле — это двухпозиционный переключатель, который имеет два состояния: включено и выключено. Это свойство позволяет использовать реле для кодирования информации в двоичном виде

Релейная машина «Марк-2», изготовленная в 1947 году, содержала около 13 000 реле. Одной из наиболее совершенных релейных машин была машина советского конструктора Н. И. Бессонова — РВМ-1. Она была построена в 1956 году и проработала почти 10 лет, конкурируя с существовавшими уже в то время ЭВМ. Поскольку реле — это механическое устройство, то его инерционность (задержка при переключении) достаточно велика, что сильно ограничивало скорость работы таких машин. Скорость РВМ-1 составляла 50 сложений или 20 умножений в секунду. Практически это был предел скорости для машин этого типа.

В первой половине XX века бурно развивалась радиотехника. Основным элементом радиоприемников и радиопередатчиков в то время были электронно-вакуумные лампы. Электронные лампы стали технической основой для первых ***электронно-вычислительных машин (ЭВМ)***.

Начало эпохи ЭВМ

Первая ЭВМ — универсальная машина на электронных лампах — была построена в США в 1945 году.

Эта машина называлась ENIAC (расшифровывается так: электронный цифровой интегратор и вычислитель). Конструкторами ENIAC были Дж. Моучли и Дж. Эккерт. Скорость счета этой машины превосходила скорость релейных машин того времени в тысячу раз.

Первый электронный компьютер ENIAC программировался с помощью штекерно-коммутационного способа, т. е. программа строилась путем соединения проводниками отдельных блоков машины на коммутационной доске. Эта сложная и утомительная процедура подготовки машины к работе делала ее неудобной в эксплуатации.

Основные идеи, по которым долгие годы развивалась вычислительная техника, были разработаны крупнейшим американским математиком **Джоном фон Нейманом**

В 1946 году в журнале «Nature» вышла статья Дж. фон Неймана, Г. Голдстайна и А. Беркса «Предварительное рассмотрение логической конструкции электронного вычислительного устройства». В этой статье были изложены принципы устройства и работы ЭВМ. Главный из них — ***принцип хранимой в памяти программы***, согласно которому данные и программа помещаются в общую память машины.

Принципиальное описание устройства и работы компьютера принято называть ***архитектурой ЭВМ***. Идеи, изложенные в упомянутой выше статье, получили название «*архитектура ЭВМ Дж. фон Неймана*».

В 1949 году была построена первая ЭВМ с архитектурой Неймана — английская машина EDSAC. Годом позже появилась американская ЭВМ EDVAC. Названные машины существовали в единственных экземплярах. Серийное производство ЭВМ началось в развитых странах мира в 50-х годах XX века.

В нашей стране первая ЭВМ была создана в 1951 году. Называлась она МЭСМ — малая электронная счетная машина. Конструктором МЭСМ был **Сергей Алексеевич Лебедев**

Велика роль академика С. А. Лебедева в создании отечественных компьютеров. Под его руководством в 50-х годах были построены серийные ламповые ЭВМ БЭСМ-1 (быстродействующая электронная счетная машина), БЭСМ-2, М-20. В то время эти машины были одними из лучших в мире.

В 60-х годах XX века С. А. Лебедев руководил разработкой полупроводниковых ЭВМ БЭСМ-3М, БЭСМ-4, М-220, М-222. Выдающимся достижением того периода была машина БЭСМ-6. Это первая отечественная и одна из первых в мире ЭВМ с быстродействием 1 миллион операций в секунду.

Последующие идеи и разработки С. А. Лебедева способствовали созданию более совершенных машин следующих поколений.

Четыре поколения ЭВМ

Электронно-вычислительную технику принято делить на поколения. Смены поколений чаще всего были связаны со сменой элементной базы ЭВМ, с прогрессом электронной техники. Это всегда приводило к росту вычислительной мощности ЭВМ, т. е. быстродействия и объема памяти. Но это не единственное следствие смены поколений. При таких переходах, как правило, происходили существенные изменения в архитектуре ЭВМ, расширялся круг задач, решаемых на ЭВМ, менялся способ взаимодействия между пользователем и компьютером.

Первое поколение ЭВМ — *ламповые машины 50-х годов XX века.* Скорость счета самых быстрых машин первого поколения доходила до 20 тысяч операций в секунду (ЭВМ М-20). Для ввода программ и данных использовались перфоленты и перфокарты. Поскольку внутренняя память этих машин была невелика (могла вместить в себя несколько тысяч чисел и команд программы), то они, главным образом, использовались для инженерных и научных расчетов, не связанных с переработкой больших объемов данных.

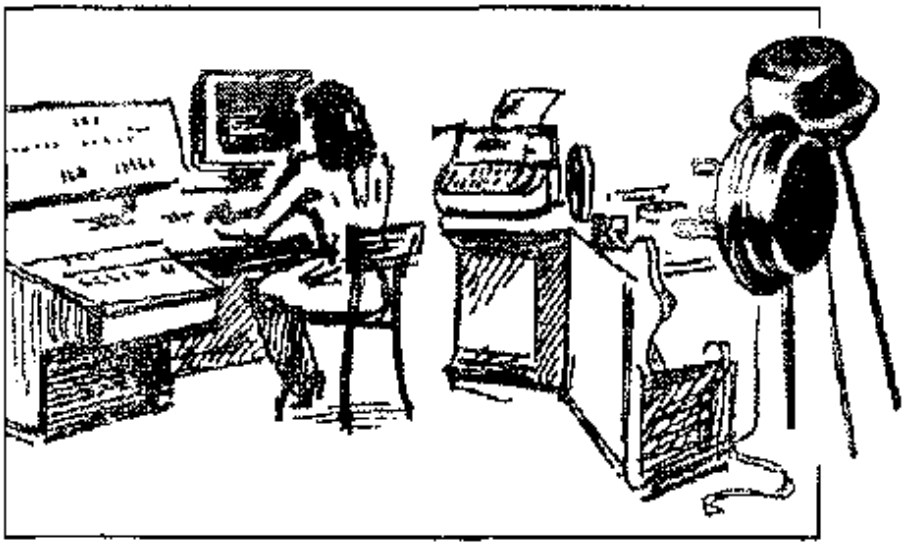
Это были довольно громоздкие сооружения, содержавшие в себе тысячи ламп, занимавшие иногда сотни квадратных метров, потреблявшие электроэнергию в сотни киловатт

Программы для таких машин составлялись на языках машинных команд. Это довольно трудоемкая работа. Поэтому программирование в те времена было доступно немногим.

В 1949 году в США был создан первый полупроводниковый прибор, заменяющий электронную лампу. Он получил название «транзистор». Транзисторы быстро внедрялись в радиотехнику.

В 60-х годах XX века транзисторы стали элементной базой для **ЭВМ второго поколения.**

Быстродействие большинства машин достигло десятков и сотен тысяч операций в секунду. Объем внутренней памяти возрос в сотни раз по сравнению с ЭВМ первого поколения.



Переход на полупроводниковые элементы улучшил качество ЭВМ по всем параметрам: они стали компактнее, надежнее, менее энергоемкими

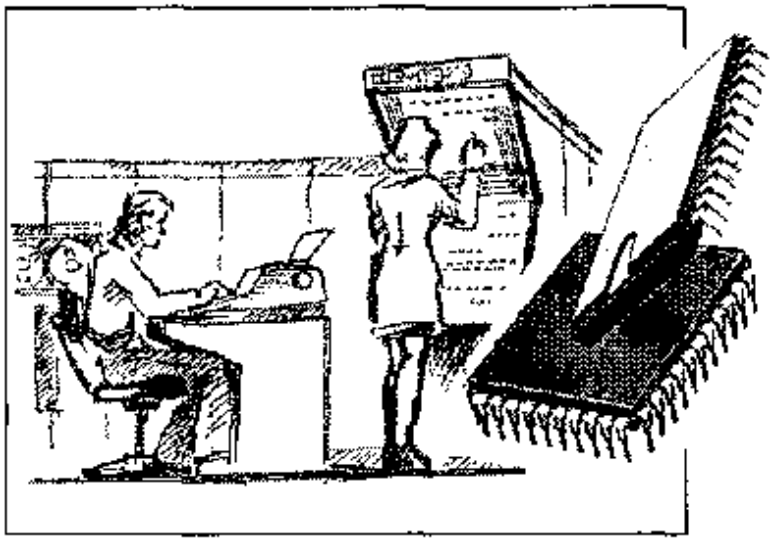
Большое развитие получили устройства внешней (магнитной) памяти: магнитные барабаны, накопители на магнитных лентах. Благодаря этому появилась возможность создавать на ЭВМ информационно-справочные, поисковые системы. Такие системы связаны с необходимостью длительно хранить на магнитных носителях большие объемы информации.

Во времена второго поколения активно стали развиваться языки программирования высокого уровня. Первыми из них были ФОРТРАН, АЛГОЛ, КОБОЛ. Составление программы перестало зависеть от модели машины, сделалось проще, понятнее, доступнее. Программирование как элемент грамотности стало широко распространяться, главным образом, среди людей с высшим образованием.

Появились мониторные системы, управляющие режимом трансляции и исполнением программ. В дальнейшем из мониторных систем выросли современные операционные системы.

Третье поколение ЭВМ *создавалось на новой элементной базе — интегральных схемах.*

Первые интегральные схемы (ИС) содержали в себе десятки, затем — сотни элементов (транзисторов, сопротивлений и др.). Когда степень интеграции (количество элементов) приблизилась к тысяче, их стали называть большими интегральными схемами — БИС; затем появились сверхбольшие интегральные схемы — СБИС.



С помощью очень сложной технологии специалисты научились монтировать на маленькой пластине из полупроводникового материала, площадью менее 1 см^2, достаточно сложные электронные схемы. Их назвали интегральными схемами

ЭВМ третьего поколения начали производиться во второй половине 60-х годов прошлого века, когда американская фирма IBM приступила к выпуску системы машин IBM-360. Это были машины на ИС. Немного позднее стали выпускаться машины серии IBM-370, построенные на БИС. В Советском Союзе в 70-х годах XX века начался выпуск машин серии ЕС ЭВМ (*Единая система ЭВМ*) по образцу IBM-360/370.

Переход к третьему поколению связан с существенными изменениями архитектуры ЭВМ. Появилась возможность выполнять одновременно несколько программ на одной машине. Такой режим работы называется мультипрограммным (*многопрограммным*) *режимом*.

Скорость работы наиболее мощных моделей ЭВМ достигла нескольких миллионов операций в секунду. На машинах третьего поколения появился новый тип внешних запоминающих устройств — магнитные диски. Как и на магнитных лентах, на дисках можно хранить неограниченное количество информации. Но накопители на магнитных дисках (НМД) работают гораздо быстрее, чем НМЛ. Широко используются новые типы устройств ввода/вывода: дисплеи, графопостроители.

В этот период существенно расширились области применения ЭВМ. Стали создаваться базы данных, первые системы искусственного интеллекта, системы автоматизированного проектирования (САПР) и управления (АСУ).

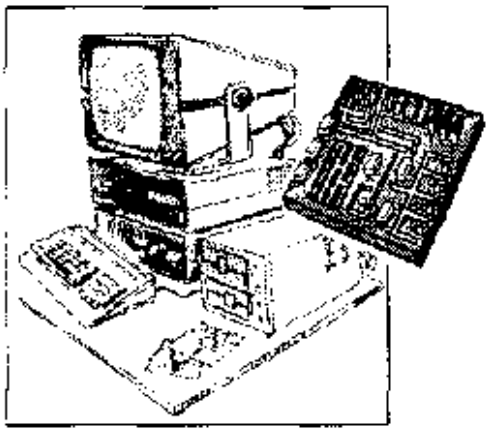
В 70-е годы XX века получила мощное развитие линия малых (мини) ЭВМ. Своеобразным эталоном здесь стали машины американской фирмы DEC серии PDP-11. В нашей стране по этому образцу создавалась серия машин СМ ЭВМ (Система малых ЭВМ). Они меньше, дешевле, надежнее больших машин. Машины этого типа хорошо приспособлены для целей управления различными техническими объектами: производственными установками, лабораторным оборудованием, транспортными средствами. По этой причине их называют управляющими машинами. Во второй половине 70-х годов XX века производство миниЭВМ превысило производство больших машин.

Очередное революционное событие в электронике произошло в 1971 году, когда американская фирма Intel объявила о создании микропроцессора.

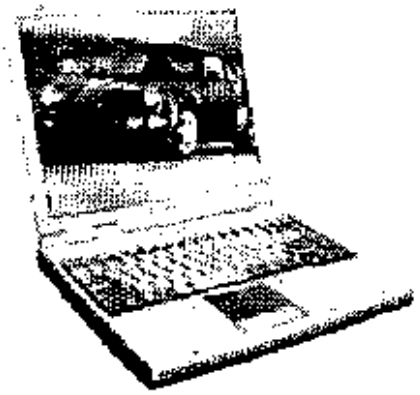
Микропроцессор — это сверхбольшая интегральная схема, способная выполнять функции основного блока компьютера — процессора

Микропроцессор — это миниатюрный «мозг», работающий по программе, заложенной в его память. Первоначально микропроцессоры стали встраивать в различные технические устройства: станки, автомобили, самолеты. Такие микропроцессоры осуществляют автоматическое управление работой этой техники.

Соединив микропроцессор с устройствами ввода/вывода, внешней памяти, получили новый тип компьютера: микроЭВМ



МикроЭВМ относятся к ЭВМ **четвертого поколения**. Существенным отличием микроЭВМ от своих предшественников являются их малые габариты (размеры бытового телевизора) и сравнительная дешевизна. Это первый тип компьютеров, который появился в розничной продаже.



Самой популярной разновидностью ЭВМ сегодня являются персональные компьютеры

Появление феномена персональных компьютеров связано с именами двух американских специалистов: Стива Джобса и Стива Возняка. В 1976 году на свет появился их первый серийный ПК Apple-1, а в 1977 году — Apple-2.

Сущность того, что такое персональный компьютер (ПК), кратко можно сформулировать так:

Персональный компьютер — это микроЭВМ с дружественным к пользователю аппаратным и программным обеспечением.

В аппаратном комплекте ПК используется цветной графический дисплей, манипуляторы типа «мышь», «джойстик», удобная клавиатура, удобные для пользователя компактные диски (магнитные и оптические). Программное обеспечение позволяет человеку легко общаться с машиной, быстро усваивать основные приемы работы с ней, получать пользу от компьютера, не прибегая к программированию. Общение человека и ПК может принимать форму игры с красочными картинками на экране, звуковым сопровождением.

Неудивительно, что машины с такими свойствами быстро приобрели популярность, причем не только среди специали-

стов. ПК становится такой же привычной бытовой техникой, как радиоприемник или телевизор. Их выпускают огромными тиражами, продают в магазинах.

С 1980 года «законодателем мод» на рынке ПК становится американская фирма IBM. Ее конструкторам удалось создать такую архитектуру, которая стала фактически международным стандартом на профессиональные ПК. Машины этой серии получили название IBM PC (Personal Computer).

В конце 80-х — начале 90-х годов XX века большую популярность приобрели машины фирмы Apple Corporation марки Macintosh. В США они широко используются в системе образования.

Появление и распространение ПК по своему значению для общественного развития сопоставимо с появлением книгопечатания. Именно ПК сделали компьютерную грамотность массовым явлением. С развитием этого типа машин появилось понятие «информационные технологии», без которых уже становится невозможным обойтись в большинстве областей деятельности человека.

Есть и другая линия в развитии ЭВМ четвертого поколения. Это **суперкомпьютер**. Машины этого класса имеют быстродействие в сотни миллионов и миллиарды операций в секунду. Только суперкомпьютеры могут справиться с обработкой больших объемов информации, например статистическими данными по переписи населения, результатами метеорологических наблюдений, финансовой информацией. Иногда скорость обработки информации имеет решающее значение. Примером может служить составление прогноза погоды, моделирование климатических изменений, позволяющих предсказать стихийное бедствие (цунами, тайфун, землетрясение и т. д.).

Суперкомпьютер — это многопроцессорный вычислительный комплекс. Высокое быстродействие достигается благодаря тому, что множество процессоров, его составляющих, осуществляют параллельную (одновременную) обработку данных.

Суперкомпьютеры являются дорогими машинами, стоимость которых может достигать десятков миллионов долларов. Поэтому возникает проблема доступности таких дорогих вычислительных ресурсов. Решение этой проблемы связано с созданием многопользовательских суперкомпьютерных центров.

В качестве альтернативы суперкомпьютерам создаются так называемые кластерные системы. Кластерная система это сеть из множества рабочих станций на базе ПК. Чтобы

рабочие станции функционировали как многопроцессорная вычислительная система, в такой сети используется специальное программное обеспечение. Оказалось, что можно построить многопроцессорный комплекс — кластер, который лишь в 2–3 раза уступает по быстродействию суперкомпьютеру, но дешевле его в сотни раз. В крупных российских университетах и научных центрах установлены и активно используются кластерные системы.

Перспективы пятого поколения

ЭВМ пятого поколения — это машины недалекого будущего. Основным их качеством должен быть высокий интеллектуальный уровень. *Машины пятого поколения — это реализованный искусственный интеллект.* В них будет возможен ввод с голоса, голосовое общение, машинное «зрение», машинное «осязание». Многое уже практически сделано в этом направлении.

Коротко о главном

Появлению ЭВМ предшествовали счетно-перфорационные и релейные машины.

ЭВМ первого поколения — ламповые машины (50-е годы XX века).

ЭВМ второго поколения — полупроводниковые машины (60-е годы XX века).

ЭВМ третьего поколения — машины на интегральных схемах (вторая половина 60-х годов).

Персональные компьютеры и суперкомпьютеры (многопроцессорные вычислительные комплексы) относятся к машинам четвертого поколения (70-е годы XX века).

ЭВМ пятого поколения — машины, основанные на искусственном интеллекте.

Вопросы и задания

1. Когда и кем были изобретены счетно-перфорационные машины? Какие задачи на них решались?
2. Что такое электромеханическое реле? Когда создавались релейные вычислительные машины? Каким быстродействием они обладали?
3. Где и когда была построена первая ЭВМ? Как она называлась?

4. Какова роль Джона фон Неймана в создании ЭВМ?
5. Кто был конструктором первых отечественных ЭВМ?
6. На какой элементной базе создавались машины первого поколения? Каковы были их основные характеристики?
7. На какой элементной базе создавались машины второго поколения? В чем их преимущества по сравнению с первым поколением ЭВМ?
8. Что такое интегральная схема? Когда были созданы первые ЭВМ на интегральных схемах? Как они назывались?
9. Какие новые области применения ЭВМ возникли с появлением машин третьего поколения?
10. Что такое микропроцессор? Когда и где был создан первый микропроцессор?
11. Что такое микроЭВМ и персональный компьютер?
12. Какие типы ПК наиболее распространены в мире?
13. Что такое суперкомпьютер?
14. Что такое кластерные системы ПК?
15. В чем особенность компьютеров пятого поколения?

§ 47 История программного обеспечения и ИКТ

Основные темы параграфа:
- *структура программного обеспечения;*
- *история систем программирования;*
- *история системного ПО;*
- *история прикладного ПО;*
- *ИКТ и их приложения.*

Вы уже хорошо знаете, что возможности современных информационных технологий определяются не только техническими характеристиками компьютеров, но и составом их программного обеспечения (ПО).

Структура программного обеспечения

Структура ПО современных персональных компьютеров схематически изображена на рис. 7.1.

Рис. 7.1. Структура программного обеспечения компьютера

Именно благодаря такому разнообразию программных средств персональный компьютер оказывается полезным и школьнику, и ученому, и домохозяйке. Представление об ИКТ — информационно-коммуникационных технологиях — связано с широким распространением всего этого множества программных продуктов.

Познакомимся с историей возникновения и развития программного обеспечения ЭВМ. Появление каждого нового типа программ связано с появлением новых областей приложения компьютеров, с расширением круга пользователей.

История систем программирования

Первые ЭВМ были доступны исключительно программистам. Поэтому исторически первым типом ПО стали системы программирования.

На машинах первого поколения языков программирования (в современном понимании) не существовало. Программисты работали на языке машинных кодов, что было весьма сложно. ЭВМ первого и второго поколений были приспособлены, прежде всего, для выполнения математических расчетов. А в таких расчетах часто приходится вычислять математические функции: квадратные корни, синусы, логарифмы и пр. Для вычисления этих функций программисты создавали стандартные программы, к которым производили обращения из своих рас-

четных программ. Стандартные программы хранились все вместе на внешнем носителе (тогда это преимущественно были магнитные ленты). Такое хранилище называлось библиотекой стандартных программ. ***Библиотеки стандартных программ*** (БСП) — первый вид программного обеспечения ЭВМ.

Затем в БСП стали включать стандартные программы решения типовых математических задач: вычисления корней уравнений, решения систем линейных уравнений и пр. Поскольку все эти программы носили математический характер, то в тот период чаще употреблялся термин ***«математическое обеспечение ЭВМ»***. Библиотеки стандартных программ используются и в современных системах программирования (см. рис. 7.1).

В эпоху второго поколения ЭВМ распространяются языки программирования высокого уровня (ЯПВУ). Об этом уже говорилось в предыдущем параграфе. ЯПВУ сделали программирование доступным не только для профессиональных программистов. Программировать стали многие научные работники, инженеры, студенты различных специальностей и даже школьники, проходящие специальную подготовку по программированию.

В программное обеспечение ЭВМ включаются ***трансляторы с ЯПВУ***. Подробнее о языках программирования и трансляторах читайте в разделе 6.3 второй части учебника. Понятие ***системы программирования*** в современном виде возникло в период третьего поколения ЭВМ, когда программисты для разработки программ стали пользоваться терминальным вводом (клавиатурой и дисплеем). В состав систем программирования были включены ***текстовые редакторы*** для ввода и редактирования программы и ***отладчики***, позволяющие программисту исправлять ошибки в программе в интерактивном режиме.

История системного ПО

Операционные системы (ОС). Первые версии ОС появились еще на ЭВМ второго поколения, но массовое распространение операционные системы получают, начиная с машин третьего поколения.

Основная проблема, которую решали разработчики ОС — повышение эффективности работы компьютера. На первых ЭВМ процессор — основное вычислительное устройство — нередко больше простаивал, чем работал во время выполнения программы. Такое происходило, если выполняемая про-

грамма часто обращалась к внешним устройствам: ввода, вывода, внешней памяти. Дело в том, что эти устройства работают в тысячи раз медленнее процессора.

Операционная система позволяет реализовать многопрограммный режим работы компьютера, при котором в состоянии выполнения находятся одновременно несколько программ. Когда одна программа обращается к внешнему устройству, процессор прерывает работу с ней (внешнее устройство продолжает работу без участия процессора) и переходит к обработке другой программы. Затем процессор может прервать работу со второй программой и продолжить выполнение первой. Таким образом, несколько программ «выстраивается в очередь» к процессору, а ОС управляет обслуживанием этой очереди. Точно так же ОС управляет обслуживанием очереди к внешним устройствам, например к принтеру. Управляют ОС и очередью к средствам ПО: трансляторам, библиотекам, прокладным программам и пр. *Управление ресурсами ЭВМ — это первая функция операционных систем.*

С появлением систем коллективного пользования ЭВМ операционные системы стали поддерживать многопользовательский режим работы. В таких системах с одной ЭВМ одновременно работают множество людей через терминальные устройства: клавиатуру и дисплей. *ОС обеспечивает режим диалога с пользователями — интерактивный режим общения.* При этом у каждого пользователя (программиста) создается впечатление, что он работает с компьютером один на один.

Еще одной важной функцией ОС стала организация работы с файлами. На ЭВМ третьего поколения появились магнитные диски, на которых информация хранится в файловой форме. *Файловая система — это компонента ОС, работающая с файлами.*

Операционные системы современных ПК также выполняют все эти функции. Отличительной особенностью их от первых ОС является дружественный графический интерфейс. А в последнее время – поддержка сетевого режима работы как в локальных, так и в глобальных сетях.

Сервисные программы. Этот тип ПО возникает и развивается в эпоху персональных компьютеров. Сюда входят разнообразные утилиты, антивирусные программы, программы-архиваторы.

Утилита — это небольшая программа, выполняющая действия, направленные на улучшение работы компьютера. Например, программа восстановления ошибочно удаленных

файлов, программа обслуживания жесткого диска: лечения, дефрагментации и т. д.

Компьютерным вирусом является программа, способная внедряться в другие программы. Программы-вирусы выполняют нежелательные, и даже опасные действия для работы компьютера: разрушают файловые структуры, «засоряют» диски, и даже выводят из строя устройства компьютера. Для защиты от вирусов используются специализированные ***антивирусные программы*** (антивирус Касперского AVP, Norton Antivirus и т.д.).

Потребность в ***программах-архиваторах*** первоначально возникала в 80–90-х годах XX века в связи с небольшими информационными объемами устройств внешней памяти — магнитных дисков. Программа-архиватор (WinRaR, ZipMagic и др.) позволяет сократить объем файла в несколько раз без потери содержащейся в нем информации. В последнее время большое значение приобрело использование архивированных файлов в сетевых технологиях: электронной почте, файловых архивах — FTP-службе Интернета.

История прикладного ПО

Именно благодаря этому типу ПО персональные компьютеры получили широкое распространение в большинстве областей деятельности человека: медицине, экономике, образовании, делопроизводстве, торговле и даже в быту.

Самым массовым спросом среди прикладных программ пользуются, конечно, ***текстовые редакторы*** и ***текстовые процессоры*** (например, MS Word). Ушли в прошлое пишущие машинки. Персональный компьютер, оснащенный текстовым редактором, и принтер стали основными инструментами для создания любых текстовых документов.

В 1979 году был создан первый ***табличный процессор*** — электронная таблица VisiCalc, ставшая самой популярной программой в среде предпринимателей, менеджеров и бухгалтеров. Идея электронной таблицы принадлежала Дэну Бринклину — студенту Гарвардской школы бизнеса. Начиная с 80-х годов табличные процессоры входят в число лидирующих категорий программного обеспечения.

В конце 70-х — начале 80-х годов XX века появились первые коммерческие ***системы управления базами данных (СУБД)*** — программное обеспечение, которое позволяет пользователям создавать и обслуживать компьютерную базу данных, а также управлять доступом к ней. В зависимости от области применения различают:

- настольные СУБД (Access, FoxPro, Paradox и т. д.), предназначенные для работы с небольшими базами данных, хранящимися на локальных дисках ПК или в небольших локальных сетях;
- СУБД серверного типа (Oracle, SQL Server, Informix и т. д.), ориентированные на работу с большими базами данных, расположенными на компьютерах-серверах.

В настоящее время все чаще приходится обрабатывать информацию (видео, звук, анимацию), которую невозможно хранить в традиционных базах данных. Jasmine является первой в мире СУБД, ориентированной на разработку баз данных, хранящих мультимедийную информацию.

Электронный офис — в последнее время часто используемое понятие. Обычно под этим понимают такой метод ведения делопроизводства, при котором всю циркулирующую информацию обрабатывают электронным способом с помощью определенных технических средств и программного обеспечения. Таким программным обеспечением являются ***интегрированные пакеты***, включающие набор приложений, каждое из которых ориентировано на выполнение определенных функций, создание документов определенного типа (текстовых документов, электронных таблиц и т. д.). В процессе работы может происходить обмен информацией между документами, создаваться составные документы, включающие в себя объекты разных типов (текст, рисунки, электронные таблицы).

Широко используемым сегодня интегрированным пакетом является офисная система Microsoft Office, базовыми компонентами которой принято считать текстовый редактор MS Word и табличный процессор MS Excel. В состав пакета также включены СУБД MS Access, система подготовки презентаций MS PowerPoint, программа обмена почтовыми сообщениями Outlook Express и Web-браузер Internet Explorer.

В 90-е годы XX века появляется термин ***мультимедиа***, относящийся к таким видам информации, как видео и звук. Для хранения мультимедиа файлов требуются большие объемы внешней памяти ПК, для обработки — большие процессорные мощности. Создание объемного реалистического изображения обеспечивается современными видеокартами, обработка звука — звуковой картой. Появляются программы редактирования и монтажа звука и видео, предназначенные для профессионалов в области музыки и видео. Наряду с этим создаются ***программы-проигрыватели мультимедиа***

файлов (Windows Media Player, Real Media Player и др.), ориентированные на широкий круг пользователей.

В 1991 году сотрудник Женевской лаборатории практической физики Тим Бернерс-Ли разрабатывает систему гипертекстовых страниц Internet, получившую название World Wide Web (WWW) — Всемирная паутина. Создание собственной Web-страницы и опубликование ее в сети под силу многим пользователям, благодаря специальным *программам-конструкторам Web-страниц*. Наиболее популярным сегодня являются Microsoft FrontPage, входящий в состав пакета Microsoft Office, и Macromedia DreamWeaver. Этими программами пользуются не только любители, но и профессионалы Web-дизайна.

Прикладное ПО специального назначения. Данный тип программного обеспечения служит информатизации различных профессиональных областей деятельности людей. Трудно дать исчерпывающий обзор для этой области. Сейчас практически в любой профессии, связанной с обработкой информации, существует свое специализированное ПО, свои средства информационных технологий.

Информационная технология — совокупность массовых способов и приемов накопления, передачи и обработки информации с использованием современных технических и программных средств.

ИКТ и их приложения

В последнее время в употребление вошел термин «информационно-коммуникационные технологии» — ИКТ. Рассмотрим лишь некоторые примеры профессионального использования ИКТ.

Технологии подготовки документов. Любая деловая сфера связана с подготовкой различной документации: отчетной, научной, справочной, сопроводительной, финансовой и т. д. Сегодня подготовка документа любой сложности немыслима без применения компьютера.

Для подготовки текстовых документов используются текстовые процессоры, которые прошли путь развития от простейших *редакторов*, не дающих возможность даже форматировать текст до *текстовых процессоров*, позволя-

ющих создавать документы, включающие в себя не только текст, но и таблицы, рисунки. Информационные технологии, связанные с созданием текстовых документов, широко используются в полиграфической промышленности. Там получили распространение ***издательские системы*** (например, Page Maker), позволяющие создавать макеты печатных изданий (газет, журналов, книг).

Большую роль в автоматизации подготовки финансовых документов сыграли ***электронные таблицы***. Первая электронная таблица под названием VisiCalc (Visible Calculator — «видимый калькулятор»), созданная Дэниелом Бриклином, появилась в 1979 году. Фактически в 80-х годах прошлого столетия электронные таблицы были лидирующей категорией программного обеспечения. И сейчас они широко применяются.

В настоящее время в финансовой сфере все больше используются ***бухгалтерские системы*** (1С-бухгалтерия и др.). Их широкое применение объясняется тем, что с помощью такой системы можно не только произвести финансовые расчеты, но и получить бумажные и электронные копии таких документов, как финансовый отчет, расчет заработной платы и пр. Электроные копии могут быть отправлены с помощью сетевых технологий в проверяющую организацию, например в налоговую инспекцию.

Для подготовки научных документов, содержащих математические расчеты, используются ***математические пакеты программ*** (MathCAD, Maple и пр.). Современные математические пакеты позволяют создавать документы, совмещающие текст с математическими расчетами и чертежами. С помощью такого документа можно получить результаты расчетов для разных исходных данных, изменяя их непосредственно в тексте документа. Большинство математических систем, используемых сегодня, было создано еще в середине 80-х годов прошлого столетия, т. е. вместе с появлением персональных компьютеров. Новые версии этих систем включают в себя новые возможности, например использование сетевых технологий: организацию доступа к ресурсам сети Интернет во время работы в среде математического пакета.

ИКТ в управлении предприятием. Эффективность работы компании (производственной, торговой, финансовой и пр.) зависит от того, как организованы хранение, сбор, обмен, обработка и защита информации. Для решения этих проблем

уже более двадцати лет назад стали внедряться ***автоматизированные системы управления*** (АСУ).

В настоящее время в этой области произошли большие перемены. Классическая АСУ включает в себя систему сбора информации, базу данных, систему обработки и анализа информации, систему формирования выходной информации. Блок обработки и анализа информации является центральным. Его работа основана на экономико-математической модели предприятия. Он решает задачи прогнозирования деятельности компании на основе финансово-бухгалтерских расчетов, реагирования на непредвиденные ситуации, т. е. оказывает помощь в принятии управленческих решений.

Как правило, *АСУ работают на базе локальной сети предприятия*, что обеспечивает оперативность и гибкость в принятии решений. С развитием глобальных сетей появилась ***коммуникационная технология Intranet***, которую называют корпоративной паутиной. Intranet обеспечивает информационное взаимодействие между отдельными сотрудниками и подразделениями компании, а также ее отдаленными внешними партнерами. Intranet помогает поддерживать оперативную связь центрального офиса с коммерческими представительствами компании, которые обычно располагаются далеко друг от друга.

ИКТ в проектной деятельности. Информатизация произвела на свет еще одну важную технологию — ***системы автоматизированного проектирования*** (***САПР***).

Проектирование включает в себя создание эскизов, чертежей, производство экономических и технических расчетов, работу с документацией.

Существуют САПРы двух видов: чертежные и специализированные. Чертежные САПРы универсальны и позволяют выполнить сложные чертежи в любой сфере технического проектирования (AutoCad). Специализированная САПР, например на проектирование жилых зданий, содержит в базе данных все необходимые сведения о строительных материалах, о стандартных строительных конструкциях, фундаментах. Инженер-проектировщик создает чертежи, производит технико-экономические расчеты с использованием таких систем. При этом повышается производительность труда конструктора, качество чертежей и расчетных работ.

Геоинформационные системы. Геоинформационные системы (ГИС) хранят данные, привязанные к географической карте местности (района, города, страны). Например, муни-

ципальная ГИС содержит в своих базах данных информацию, необходимую для всех служб, поддерживающих жизнедеятельность города: городских властей, энергетиков, связистов, медицинских служб, милиции, пожарной службы и пр. Вся эта разнородная информация привязана к карте города. Использование ГИС помогает соответствующим службам оперативно реагировать на чрезвычайные ситуации: стихийные бедствия, экологические катастрофы, технологические аварии и пр.

ИКТ в образовании. В наше время от уровня образованности людей существенно зависит уровень развития страны, качество жизни ее населения. Требования к качеству образования постоянно растут. Старые, традиционные методы обучения уже не успевают за этими требованиями. Возникает очевидное противоречие. Использование ИКТ в образовании может помочь в разрешении этого противоречия.

Технологии обучения мало изменились за последние 100 лет. Пока, в основном, действует метод коллективного обучения. Не всегда такой способ обучения дает высокие результаты. Причина заключается в разном уровне способностей разных учеников. Учителя хорошо понимают, что необходим индивидуальный подход в работе с учащимися. Решению этой проблемы может помочь использование в процессе обучения специальных программ (обучающих, контролирующих, тренажерных и т. д.), входящих в состав ***электронного учебника.***

Обучение — это процесс получения знаний. Традиционный источник знаний — учебник ограничен в своих информационных возможностях. Обучающимся на любой ступени образования всегда требовались дополнительные источники информации: библиотеки, музеи, архивы и пр. В этом отношении жители крупных городов находятся в более благоприятных условиях, чем сельские жители. Здесь можно говорить о существовании *информационного неравенства.* Решить эту проблему поможет широкое использование в обучении ***информационных ресурсов Интернета.*** В частности, *специализированных порталов учебной информации.*

Еще одна проблема системы образования связана с неравными возможностями получения качественного образования из-за географической отдаленности от образовательных центров. Например, для жителя Якутии проблематично получить диплом престижного московского вуза. В решении этой проблемы на помощь приходит новая форма обучения — ***дис-***

танционное образование, реализация которого стала возможна благодаря развитию компьютерных сетей.

Дистанционное образование приходит на смену старой форме заочного образования, при которой весь информационный обмен происходил в письменном виде через почтовую связь. Сетевое дистанционное образование позволяет вести обучение в режиме реального времени. Обучаемые могут не только читать учебный материал, но и видеть и слышать лекции крупных ученых, сдавать экзамены в прямом контакте с экзаменатором.

Коротко о главном

Программное обеспечение компьютера включает в себя системное ПО, прикладное ПО и системы программирования.

Исторически первым видом ПО стали системы программирования.

Ядро системного ПО — операционные системы, зародились в период второго поколения ЭВМ, но распространение получили, начиная с третьего поколения.

Сервисные программы (утилиты, архиваторы, антивирусные программы) получили распространение на персональных компьютерах.

Прикладное программное обеспечение общего назначения развивалось от внедрения отдельных программ (текстовых редакторов, табличных процессоров, СУБД и пр.) до интегрированных систем — офисных пакетов.

Информационная технология — совокупность массовых способов и приемов накопления, передачи и обработки информации с использованием современных технических и программных средств.

Информационно-коммуникационные технологии (ИКТ) в настоящее время используются в большинстве профессиональных областей, связанных с обработкой информации, в том числе все шире применяются в образовании.

Вопросы и задания

1. Какова структура программного обеспечения современного компьютера?
2. Почему первыми пользователями ЭВМ стали программисты?

3. Когда началось распространение операционных систем? С чем это связано?
4. Какие виды программ, кроме ОС, относятся к системному ПО?
5. Как классифицируется прикладное ПО?
6. Перечислите основные виды прикладных программ общего назначения и назовите информационные задачи, которые решаются с их помощью.
7. Приведите примеры профессионального использования прикладных программ.
8. Назовите формы использования ИКТ, с которыми вам приходилось иметь дело в школе. Какой эффект от их использования вы можете отметить?

§ 48 Информационные ресурсы современного общества

Основные темы параграфа:
- *понятие информационных ресурсов;*
- *национальные информационные ресурсы;*
- *виды национальных информационных ресурсов.*

Понятие информационных ресурсов

Ресурс — это запас или источник некоторых средств. Всякое общество имеет определенные ресурсы, необходимые для его жизнедеятельности. Традиционными видами общественных ресурсов являются материальные, сырьевые (природные), энергетические, трудовые, финансовые ресурсы. В дополнение к этому, одним из важнейших видов ресурсов современного общества являются ***информационные ресурсы***. К информационным ресурсам уместно отнести все научно-технические знания, произведения литературы и искусства.

Информационные ресурсы — это знания, идеи человечества и указания по их реализации, зафиксированные в любой форме, на любом носителе информации.

Между информационными ресурсами и всякими иными существует важное различие: *всякий ресурс, кроме информационного, после его использования исчезает* (например, иссякают нефтяные месторождения, расходуются финансовые ресурсы). Информационным ресурсом можно пользоваться многократно. Кроме того, использование информационных ресурсов влечет за собой создание новых ресурсов, в том числе и информационных. Информационные ресурсы тем быстрее растут, чем быстрее их расходуют.

Национальные информационные ресурсы

Применительно к отдельному государству говорят о ***национальных информационных ресурсах***, включающих библиотечные и архивные ресурсы, научно-техническую информацию, отраслевую информацию, информацию государственных (властных) структур, информационные ресурсы социальной сферы и пр.

Виды национальных информационных ресурсов

Огромные информационные ресурсы скрыты в ***библиотеках***. Библиотечная сеть России насчитывает 150 тысяч библиотек. Библиографическая регистрация и статистический учет выходящей в стране всей печатной продукции осуществляется Российской книжной палатой. Библиотечное дело в массовом порядке переводится на использование информационных технологий: создаются электронные каталоги, специализированные (библиографические) базы данных. В ряде центральных библиотек сформированы собственные электронные информационные ресурсы. Начался процесс публикации этих ресурсов через Интернет.

Архивы хранят материалы, связанные с историей и культурой страны. Объемы архивных материалов огромны и накапливаются зачастую быстрее, чем их удается обрабатывать. Архивный фонд Российской Федерации ежегодно пополняется на 1,6 млн документов. В государственных архивах создано и поддерживается более 400 баз данных.

Во всех развитых странах существуют специализированные ***центры научно-технической информации***, включающие специализированные издания, патентные службы и пр. Примером такого центра в нашей стране является ВИНИТИ РАН — Всероссийский институт научной и технической информации Российской академии наук. Этот информационный центр генерирует 60–70% всей научно-технической ин-

формации в России. Информация такого рода часто является дорогостоящим товаром.

Свои ***отраслевые информационные ресурсы*** имеются у любой промышленной, социальной и иной сферы общества.

Информационные ресурсы социальной сферы связаны с образованием, медициной, системой пенсионного обеспечения, службами занятости, медицинского и социального страхования. Например, основу информационных ресурсов в области образования составляют библиотеки вузов — высших учебных заведений (университетов, институтов, академий), общий фонд которых составляет более 300 млн экземпляров. Создана и функционирует федеральная телекоммуникационная университетская сеть. В сети Интернет отражена информация о большинстве российских вузов.

Следует отметить, что сегодня информационные ресурсы по своей значимости сопоставимы с другими ресурсами (материальными, сырьевыми, энергетическими, финансовыми). Об этом свидетельствует тот факт, что они становятся товаром, стоимость которого сопоставима со стоимостью других ресурсов. Для развитых стран характерно увеличение объема потребления из-за рубежа природных ресурсов (газ, нефть и пр.) в обмен на экспорт своих информационных ресурсов (научно-технических знаний, информационных технологий).

Коротко о главном

Информационные ресурсы — это знания, идеи человечества и указания по их реализации, зафиксированные в любой форме, на любом носителе информации.

Всякий ресурс, кроме информационного, после его использования исчезает. Информационным ресурсом можно пользоваться многократно.

К национальным информационным ресурсам относятся: фонды библиотек и архивов, центры научно-технической информации, отраслевые информационные ресурсы, информационные ресурсы социальной сферы, в том числе сферы образования.

Вопросы и задания

1. В чем основное отличие информационных ресурсов от материальных?

2. Перечислите основные виды национальных информационных ресурсов.
3. Назовите, какими видами информационных ресурсов вам приходится (или приходилось) пользоваться.

§ 49 Проблемы формирования информационного общества

Основные темы параграфа:

- *что такое информационное общество;*
- *что такое информатизация;*
- *цели информатизации;*
- *информационные преступления и информационная безопасность;*
- *меры обеспечения информационной безопасности.*

Что такое информационное общество

Человеческое общество вступает в период своего развития, который называют ***информационным обществом***.

В информационном обществе преимущественным видом трудовой деятельности людей станет информационная деятельность. Информационные ресурсы становятся важнейшими из всех видов ресурсов, влияющими на общественный прогресс.

Средствами информационной деятельности людей выступает компьютерная техника, информационно-коммуникационные технологии — ИКТ.

Что такое информатизация

Обсудим еще один часто употребляемый термин: информатизация. ***Информатизация*** — это процесс создания, развития и массового применения информационных средств и технологий. Этот процесс происходил в течение всей истории

человеческого общества и начался отнюдь не с появлением ЭВМ. Мы уже обсуждали этот вопрос в данной главе. Именно способность работать с информацией, создавать орудия информационного труда окончательно выделила человека из мира животных.

Однако в наше время смысл и значение процесса информатизации существенно изменились. Как никогда раньше возросла роль информационных ресурсов (об этом уже говорилось выше) для всего общества и для каждого отдельного человека. Доступность информационных ресурсов определяет степень информированности людей. Поэтому основной целью нынешнего этапа информатизации становится *обеспечение высокого уровня информированности населения, необходимого для кардинального улучшения условий труда и жизни каждого человека.*

Информатизация — это процесс создания, развития и массового применения информационных средств и технологий, обеспечивающий высокий уровень информированности населения, необходимый для кардинального улучшения условий труда и жизни каждого человека, повышения эффективности всех видов производства.

Цели информатизации

Всякий процесс направлен на достижение каких-то целей. В данном выше определении сформулирована *глобальная цель информатизации*, которую можно разделить на несколько составляющих. Их можно назвать основными целями информатизации. Таких основных целей четыре.

1. Информационное обеспечение всех видов человеческой деятельности. В качестве примеров видов деятельности можно назвать науку, образование, промышленное производство, сельское хозяйство, здравоохранение и многое другое. Информационное обеспечение складывается из специализированных фондов (баз данных, архивов, библиотек) и совокупности методов и средств их организации и использования.

2. Информационное обеспечение активного отдыха и досуга людей. В первую очередь эта цель достигается путем обеспечения населению возможности теледоступа ко всей со-

кровищнице мировой культуры, а также создание индустрии телеразвлечений.

3. Формирование и развитие информационных потребностей людей. Эта цель носит определенный воспитательный характер и является одной из задач общего образования. Учебная или производственная деятельность людей должна стимулировать их к повышению своего уровня информированности. Без решения этой задачи нельзя рассчитывать на успех информатизации, так как ее результаты могут оказаться просто невостребованными.

4. Формирование условий, обеспечивающих осуществление информатизации. К таким условиям относятся различные виды обеспечения информатизации: экономические, организационные, научно-технические, правовые. Создание этих условий — функция государства, органов, управляющих процессами информатизации.

Информационные преступления и информационная безопасность

Многие черты информационного общества уже присутствуют в современной жизни развитых стран. Компьютеры контролируют работу атомных реакторов, распределяют электроэнергию, управляют самолетами и космическими кораблями, определяют надежность систем обороны страны и банковских систем, т. е. используются в областях общественной жизни, обеспечивающих благополучие и даже жизнь множества людей.

Жизненно важной для общества становится проблема информационной безопасности действующих систем хранения, передачи и обработки информации.

О важности этой проблемы свидетельствуют многочисленные факты. Каждые двадцать секунд в США происходит преступление с использованием программных средств. Более 80% компьютерных преступлений осуществляется через глобальную сеть Интернет, которая обеспечивает широкие возможности злоумышленникам для нарушений в глобальном масштабе. Если компьютер, являющийся объектом атаки, подключен к Интернету, то для «взломщика» нет большой разницы, где он находится — в соседней комнате или на

другом континенте. Потери от хищения или повреждения компьютерных данных составляют более 100 млн долларов в год. Во многих случаях организации даже не подозревают, что «вторжение» имело место, хищение информации происходит незаметно.

Перечислим некоторые виды компьютерных преступлений, когда компьютер является инструментом для совершения преступления, а объектом преступления является информация:

1. *Несанкционированный (неправомерный) доступ к информации.* Лицо получает доступ к секретной информации, например, путем подбора шифра (пароля). Подавляющее большинство разработок в области информационной безопасности посвящено именно этому виду преступлений, что позволяет обеспечить охрану государственных и военных секретов.
2. *Нарушение работоспособности компьютерной системы.* В результате преднамеренных действий ресурсы вычислительной системы становятся недоступными, или снижается ее работоспособность. Примером такого рода преступлений является создание и распространение компьютерных вирусов.
3. *Подделка (искажение или изменение), т. е. нарушение целостности компьютерной информации.* Эта деятельность является разновидностью неправомерного доступа к информации. К подобного рода действиям можно отнести подтасовку результатов голосования на выборах, референдумах и т. д. путем внесения изменений в итоговые протоколы.

Меры обеспечения информационной безопасности

Эти меры применяются в основном на этапе эксплуатации информационной системы.

Разработчики системы, предназначенной для обработки важной информации, должны предусмотреть средства защиты уже на этапе ее создания. Существует даже специальный термин ***«защищенная система»*** — это информационная система, обеспечивающая безопасность обрабатываемой информации и поддерживающая свою работоспособность в условиях воздействия на нее заданного множества угроз (нарушение целостности информации, несанкционированный доступ, попытки нарушения работоспособности).

Средства защиты современных информационных систем должны учитывать современные формы представления информации (гипертекст, мультимедиа и т. д.). Развитие локальных сетей и Интернета диктует необходимость эффективной защиты при удаленном доступе к информации. Необходимо осуществлять защиту от автоматических (программных) средств нападения: компьютерных вирусов, автоматизированных средств взлома.

Технология создания защищенных автоматизированных систем обработки информации предполагает использование и следование стандартам информационной безопасности, которые изложены в специальных документах. В России — это документы Гостехкомиссии, в США — так называемая «Оранжевая книга». Существуют также «Единые критерии безопасности информационных технологий», являющиеся результатом совместных усилий авторов стандартов информационной безопасности из США, Канады и Европы. Это свидетельствует о том, что проблема информационной безопасности является межгосударственной.

Наряду с программно-техническими средствами защиты информации действуют правовые, юридические меры защиты. В 1996 году в России впервые в уголовный кодекс был внесен раздел «Преступления в сфере компьютерной информации». В нем определены меры наказания за перечисленные выше виды компьютерных преступлений.

К защите информации относится также и осуществление авторских и имущественных прав на интеллектуальную собственность, каковым является программное обеспечение.

Необходимой составляющей общей культуры современного человека становится информационная культура. Это понятие включает в себя не только умение использовать средства информационно-коммуникационных технологий, но также и соблюдение правовых норм в своей информационной деятельности.

Коротко о главном

В информационном обществе основным предметом трудовой деятельности людей становится информация.

Целью процесса информатизации является достижение высокого уровня информированности населения, что приведет к существенному улучшению условий жизни и труда, к повышению эффективности производства.

Проблема информационной безопасности связана с широким использованием компьютеров в жизненно важных областях.

Основные формы компьютерных преступлений: несанкционированный (неправомерный) доступ к информации, нарушение работоспособности компьютерной системы, нарушение целостности компьютерной информации.

Основные меры по защите от компьютерных преступлений: технические и аппаратно-программные, административные, юридические.

Важнейшая научно-техническая задача в области информатики: разработка технологий создания защищенных автоматизированных систем обработки информации.

Программное обеспечение является интеллектуальной собственностью разработчиков. Его использование должно оплачиваться.

Вопросы и задания

1. По каким признакам можно судить о наступлении эпохи информационного общества?
2. Попробуйте оценить, какая часть родителей ваших одноклассников трудится в сфере материального производства, а какая — в информационной сфере. Отсюда сделайте вывод: как далеко мы находимся от стадии информационного общества.
3. Что такое информатизация? Назовите основные цели информатизации. Какие из этих целей в большей или меньшей степени, по вашему мнению, достигнуты в нашей стране?
4. Какие действия относятся к области информационных преступлений?
5. Приведите примеры, когда человек бессознательно совершает информационное правонарушение.
6. Какие существуют меры предотвращения информационных преступлений?
7. Какие меры вы бы могли предложить сами?
8. Почему использование «пиратских» копий программного обеспечения является преступлением?
9. Как вы думаете, что привлекает хакеров (взломщиков информационных систем) в их преступной деятельности? Какими эффективными средствами можно пресечь такую деятельность?

СОЦИАЛЬНАЯ

История развития средств информатизации

Предыстория средств информатизации

- Средства **хранения** информации:
  камень, глина, папирус, бумага, книгопечатание, фотография, кино, магнитная запись
- Средства **передачи** информации:
  почта, телеграф, телефон, радио, телевидение
- Средства **обработки** информации (вычислений):
  системы счисления (непозиционные, позиционные);
  вычислительные механизмы: *абак, счеты, логарифмическая линейка, арифмометр, калькулятор*;
  прообраз ЭВМ: *аналитическая машина Беббиджа*

История ЭВМ

- 1-е поколение:
  ламповые машины 50-х годов XX века
- 2-е поколение:
  полупроводниковые машины 60-х годов XX века
- 3-е поколение:
  машины на интегральных схемах 70-х годов XX века
- 4-е поколение:
  современные ПК и супер-компьютеры

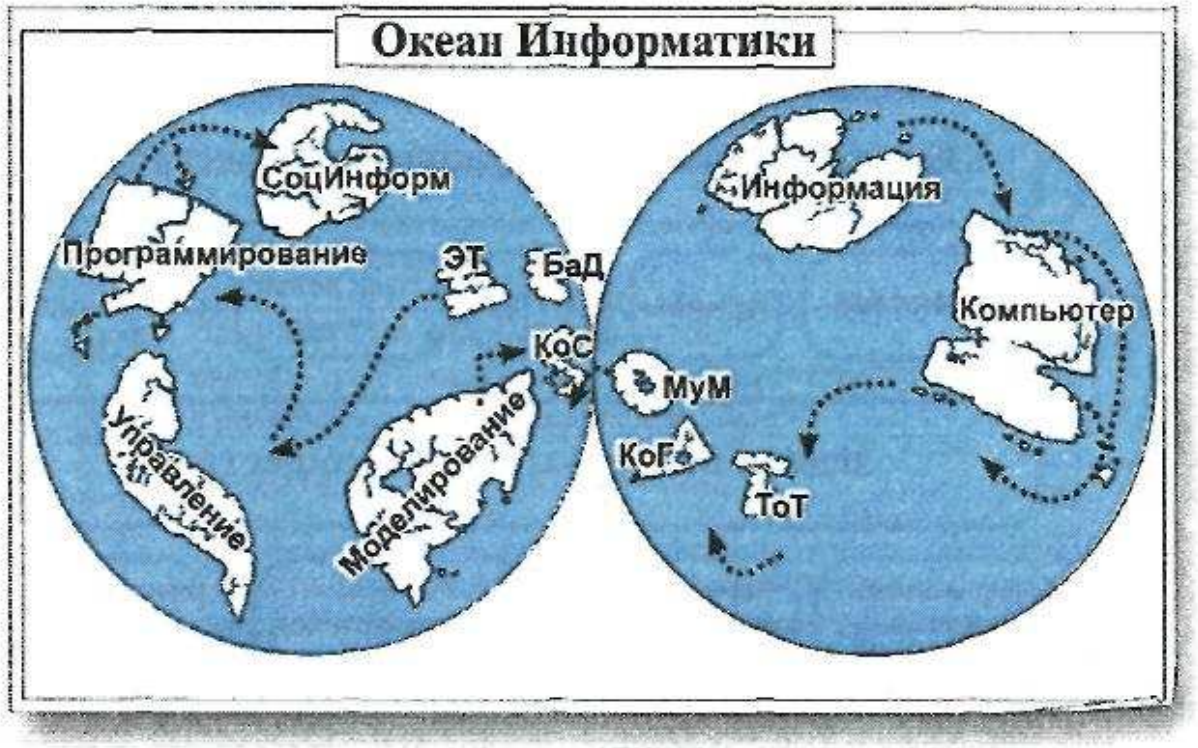
История ПО ЭВМ

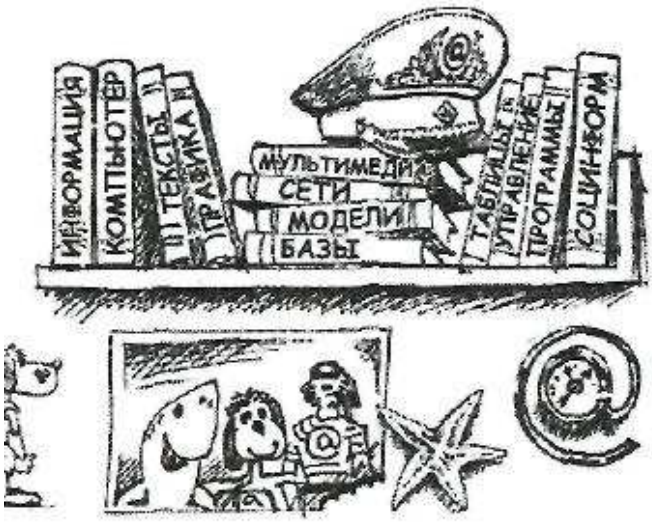
- 1-2-е поколения: Библиотеки стандартных программ; языки программирования высокого уровня
- 3-е поколение: операционные системы, СУБД, пакеты прикладных программ
- 4-е поколение: интегрированные офисные пакеты, языки и системы Web-программирования

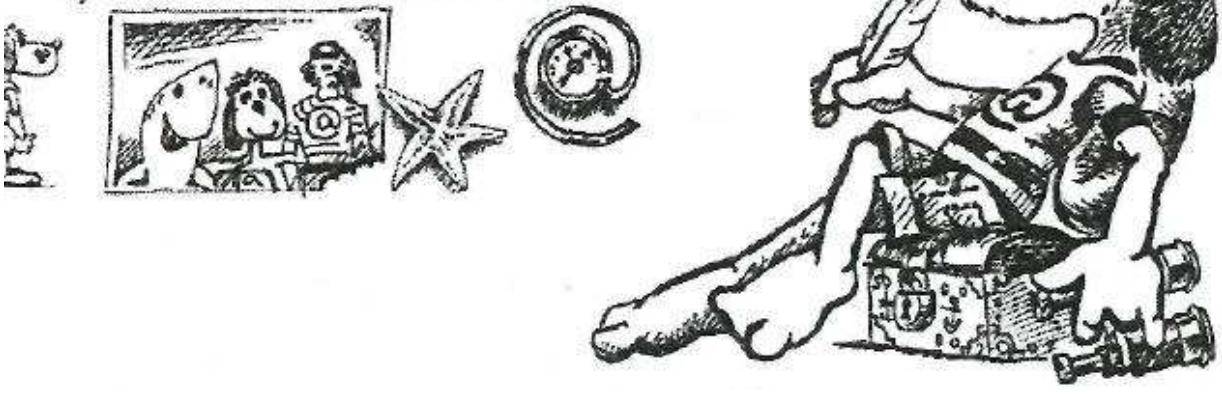
Система основных понятий главы 7

Заключение

Путешествие завершено!







Дорогие ученики!

Вот и завершилось ваше путешествие по океану знаний под названием «ИНФОРМАТИКА»! Заполнены все белые пятна на карте. Теперь вы можете еще раз внимательно рассмотреть карту, вспомнить, какие материки и острова вы посетили, что нового для себя узнали, какие приобрели практические навыки.

Знания и умения, полученные при изучении базового курса информатики, понадобятся вам в дальнейшей учебе. В современной школе все большее значение приобретает компьютерная техника и ИКТ в качестве средства обучения. Наряду с привычными бумажными учебниками используются электронные учебники. Неограниченным источником учебной информации становится Интернет. Все шире различные школьные предметы будут обращаться к компьютерному моделированию как средству обучения и развития школьников. Велика вероятность того, что после окончания школы ваша дальнейшая учеба и работа также будут связаны с использованием компьютеров и компьютерных технологий. Все это – проявление закономерного процесса общественного развития, связанного с переходом к стадии информационного общества. Вам быть членами этого общества! Желаем вам всяческих успехов!

Материал для углубленного изучения курса

Дополнение к главе 1

1.1. Передача информации по техническим каналам связи

Основные темы параграфа:
- *схема К. Шеннона;*
- *кодирование и декодирование информации;*
- *шум и защита от шума. Теория кодирования К. Шеннона.*

Схема К. Шеннона

Американским ученым, одним из основателей теории информации, Клодом Шенноном была предложена схема процесса передачи информации по техническим каналам связи, представленная на рис. 1.3.

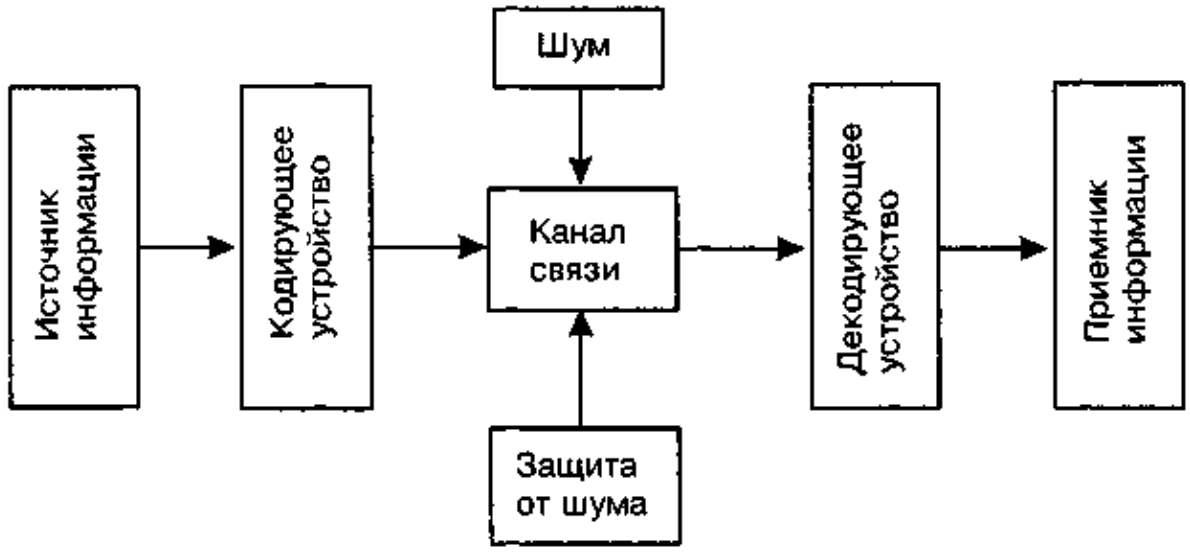


Рис. 1.3. Схема технической системы передачи информации

Работу такой схемы можно пояснить на знакомом всем процессе разговора по телефону. Источником информации является говорящий человек. Кодирующим устройством — микрофон телефонной трубки, с помощью которого звуковые волны (речь) преобразуются в электрические сигналы. Каналом связи является телефонная сеть (провода, коммутаторы

телефонных узлов, через которые проходит сигнал). Декодирующим устройством является телефонная трубка (наушник) слушающего человека — приемника информации. Здесь пришедший электрический сигнал превращается в звук.

Связь, при которой передача производится в форме непрерывного электрического сигнала, называется ***аналоговой связью***.

Кодирование и декодирование информации

Под ***кодированием*** понимается любое преобразование информации, идущей от источника, в форму, пригодную для ее передачи по каналу связи.

На заре эры радиосвязи применялся код азбуки Морзе. Текст преобразовывался в последовательность точек и тире (коротких и длинных сигналов) и передавался в эфир. Принимавший на слух такую передачу человек должен был суметь декодировать код обратно в текст. Еще раньше азбука Морзе использовалась в телеграфной связи. Передача информации с помощью азбуки Морзе — это пример ***дискретной связи***.

В настоящее время широко используется ***цифровая связь***, когда передаваемая информация кодируется в двоичную форму (0 и 1 — двоичные цифры), а затем декодируется в текст, изображение, звук. Цифровая связь, очевидно, тоже является дискретной.

Шум и защита от шума. Теория кодирования К. Шеннона

Термином «шум» называют разного рода помехи, искажающие передаваемый сигнал и приводящие к потере информации. Такие помехи прежде всего возникают по техническим причинам: плохое качество линий связи, незащищенность друг от друга различных потоков информации, передаваемых по одним и тем же каналам. Часто, беседуя по телефону, мы слышим шум, треск, мешающие понять собеседника, или на наш разговор накладывается разговор других людей. В таких случаях необходима защита от шума.

В первую очередь применяются технические способы защиты каналов связи от воздействия шумов. Такие способы бывают самыми разными, иногда — простыми, иногда — очень сложными. Например, использование экранированного кабеля вместо «голого» провода; применение разного

рода фильтров, отделяющих полезный сигнал от шума, и пр.

Клодом Шенноном была разработана специальная ***теория кодирования***, дающая методы борьбы с шумом. Одна из важных идей этой теории состоит в том, что передаваемый по линии связи код должен быть ***избыточным***. За счет этого потеря какой-то части информации при передаче может быть компенсирована. Например, если при разговоре по телефону вас плохо слышно, то, повторяя каждое слово дважды, вы имеете больше шансов на то, что собеседник поймет вас правильно.

Однако нельзя делать избыточность слишком большой. Это приведет к задержкам и удорожанию связи. Теория кодирования К. Шеннона как раз и позволяет получить такой код, который будет оптимальным. При этом избыточность передаваемой информации будет минимально возможной, а достоверность принятой информации — максимальной.

В современных системах цифровой связи часто применяется следующий прием борьбы с потерей информации при передаче. Все сообщение разбивается на порции — *пакеты*. Для каждого пакета *вычисляется контрольная сумма* (сумма двоичных цифр), которая передается вместе с данным пакетом. В месте приема заново вычисляется контрольная сумма принятого пакета, и если она не совпадает с первоначальной, то передача данного пакета повторяется. Так происходит до тех пор, пока исходная и конечная контрольные суммы не совпадут.

Коротко о главном

Любая техническая система передачи информации состоит из источника, приемника, устройств кодирования и декодирования и канала связи.

Под кодированием понимается преобразование информации, идущей от источника, в форму, пригодную для ее передачи по каналу связи. Декодирование — это обратное преобразование.

Шум — это помехи, приводящие к потере информации.

В теории кодирования разработаны методы представления передаваемой информации с целью уменьшения ее потерь под воздействием шума.

? Вопросы и задания

1. Назовите основные элементы схемы передачи информации, предложенной К. Шенноном.
2. Что такое кодирование и декодирование при передаче информации?
3. Что такое шум? Каковы его последствия при передаче информации?
4. Какие существуют способы борьбы с шумом?

1.2. Архивирование и разархивирование файлов

Основные темы параграфа:

- *проблема сжатия данных;*
- *алгоритм сжатия с использованием кода переменной длины;*
- *алгоритм сжатия с использованием коэффициента повторения;*
- *программы-архиваторы.*

Проблема сжатия данных

Вы уже знаете, что с помощью глобальной сети Интернет пользователь получает доступ к огромным информационным ресурсам. В сети можно найти редкую книгу, реферат практически по любой теме, фотографии и музыку, компьютерную игру и многое другое. При передаче этих данных по сети могут возникнуть проблемы из-за их большого объема. Пропускная способность каналов связи еще достаточно ограничена. Поэтому время передачи может быть слишком большим, а это связано с дополнительными финансовыми расходами. Кроме того, для файлов большого размера может оказаться недостаточно свободного места на диске.

Решение проблемы заключается ***в сжатии данных***, которое ведет к сокращению объема данных при сохранении закодированного в них содержания. Программы, осуществляющие такое сжатие, называются ***архиваторами***. Первые архиваторы появились в середине 1980-х годов XX века. Главной целью их использования была экономия места на дисках, информационный объем которых в те времена был значительно меньше объема современных дисков.

Сжатие данных (архивирование файлов) происходит по специальным алгоритмам. В этих алгоритмах чаще всего используются две принципиально различающиеся идеи.

Алгоритм сжатия с использованием кода переменной длины

Первая идея: использование кода переменной длины. Данные, подвергающиеся сжатию, специальным образом делят на части (цепочки символов, «слова»). Заметим, что «словом» может быть и отдельный символ (код ASCII). Для каждого «слова» находится частота встречаемости: отношение количества повторений данного «слова» к общему числу «слов» в массиве данных. Идея алгоритма сжатия информации: кодировать наиболее часто встречающиеся «слова» кодами меньшей длины, чем редко встречающиеся «слова». При этом можно существенно сократить объем файла.

Такой подход известен давно. Он используется в азбуке Морзе, где символы кодируются различными последовательностями точек и тире, причем чаще встречающиеся символы имеют более короткие коды. Например, часто используемая буква «А» кодируется так: • –. А редкая буква «Ж» кодируется: • • • –. В отличие от кодов одинаковой длины, в этом случае возникает проблема отделения кодов букв друг от друга. В азбуке Морзе эта проблема решается с помощью «паузы» (пробела), которая, по сути, является третьим символом алфавита Морзе, то есть алфавит Морзе не двух-, а трехсимвольный.

Информация в памяти ЭВМ хранится с использованием двухсимвольного алфавита. Специального символа-разделителя нет. И все же удалось придумать способ сжатия данных с переменной длиной кода «слов», не требующий символа-разделителя. Такой алгоритм называется алгоритмом Д. Хаффмена (впервые опубликован в 1952 году). Все универсальные архиваторы работают по алгоритмам, подобным алгоритму Хаффмена.

Алгоритм сжатия с использованием коэффициента повторения

Вторая идея: использование коэффициента повторения. Смысл алгоритма, основанного на этой идее, заключается в следующем: если в сжимаемом массиве данных встречается цепочка из повторяющихся групп символов, то ее заменя-

ют парой: число (коэффициент) повторений — группа символов. В этом случае для длинных повторяющихся цепочек выигрыш памяти при сжатии может быть очень большим. Данный метод наиболее эффективен при упаковке графической информации.

Программы-архиваторы

Программы-архиваторы создают архивные файлы (архивы). Архив представляет собой файл, в котором в сжатом виде хранятся один или несколько файлов. Для использования заархивированных файлов необходимо произвести их излечение из архива — ***разархивирование***. Все программы-архиваторы обычно предоставляют следующие возможности:

добавление файлов в архив;
извлечение файлов из архива;
удаление файлов из архива;
просмотр содержимого архива.

В настоящее время наиболее популярны архиваторы WinRar и WinZip. WinRAR обладает более широкими возможностями по сравнению с WinZip. В частности, он дает возможность создания многотомного архива (это удобно, если архив необходимо скопировать на дискету, а его размер превышает 1,44 Мбайт), а также возможность создания самораспаковывающегося архива (в этом случае для извлечения данных из архива не нужен сам архиватор).

Приведем пример выгоды использования архиваторов при передаче данных по сети. Размер текстового документа, содержащего параграф, который вы сейчас читаете, — 31 Кб. Если этот документ заархивировать с помощью WinRAR, то размер архивного файла составит всего 6 Кб. Как говорится, выгода налицо.

Пользоваться программами-архиваторами очень просто. Чтобы создать архив, нужно сначала выбрать файлы, которые необходимо в него включить, затем установить необходимые параметры (способ архивации, формат архива, размер тома, если архив многотомный), и, наконец, отдать команду СОЗДАТЬ АРХИВ. Похожим образом происходит обратное действие — извлечение файлов из архива (распаковка архива). Во-первых, нужно выбрать файлы, извлекаемые из архива, во-вторых, определить, куда должны быть помещены эти файлы, и, наконец, отдать команду ИЗВЛЕЧЬ ФАЙЛЫ

ИЗ АРХИВА. Подробнее с работой программ-архиваторов вы познакомитесь на практических занятиях.

Коротко о главном

Сжатие информации производится с помощью специальных программ-архиваторов.

Чаще всего в алгоритмах сжатия используются два метода: использование кода переменной длины и использование коэффициента повторения группы символов.

? *Вопросы и задания:*

1. В чем различие кодов постоянной и переменной длины?
2. Какими возможностями обладают программы-архиваторы?
3. Какова причина широкого применения программ-архиваторов?
4. Знаете ли вы другие программы-архиваторы, кроме перечисленных в этом параграфе?

Дополнение к главе 2

2.1. Системы, модели, графы

Основные темы параграфа:

- *понятие системы;*
- *граф системы;*
- *структура системы;*
- *виды графов;*
- *иерархические системы и деревья;*
- *сети.*

Понятие системы

Мы будем употреблять термин «*система*» для обозначения различных сложных объектов.

Система — это объект, состоящий из множества взаимосвязанных элементов и существующий как единое целое.

Наверняка вам приходилось слышать такие слова, как «система образования», «транспортная система», «система водоснабжения», «горная система». Действительно, слово «система» очень часто употребляется в речи. Под этим словом мы обычно понимаем что-то сложное, состоящее из множества элементов. Например, система городского транспорта включает в себя трамваи, автобусы, троллейбусы, трамвайные пути, линии электропередач, депо, службы технического обслуживания и пр.

Информационная модель всякой системы должна отражать ее состав и связи между составляющими ее элементами.

Граф системы

Посмотрите на следующий рисунок (рис. 2.9).

На нем в овалах записаны названия населенных пунктов с карты из § 7. Пункты, связанные на карте дорогами, соеди-

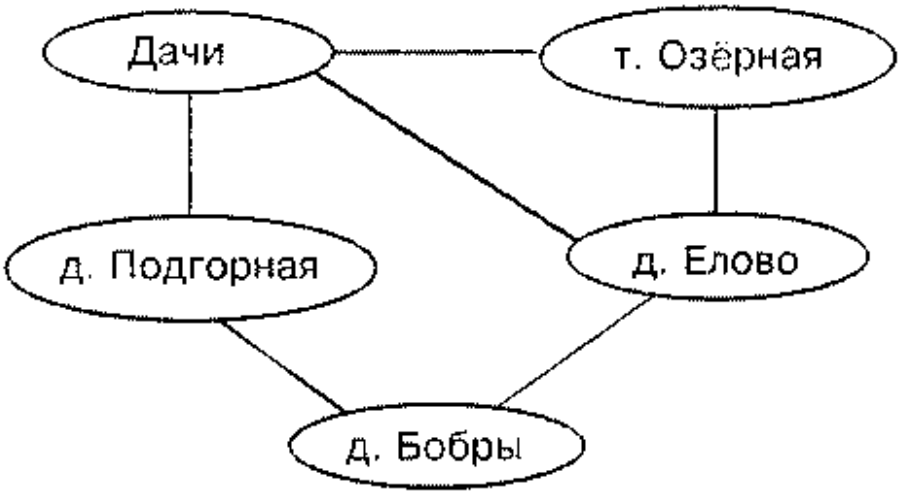


Рис. 2.9. Граф, отражающий связи между населенными пунктами

нены на рисунке линиями. Однако на карту этот рисунок не похож: относительное расположение поселков, форма и длина дорог здесь не отражены. Из рисунка можно лишь узнать, между какими населенными пунктами есть дороги. Такой рисунок является ***графом***.

Структура системы

В нашем примере мы рассматриваем данную местность *как систему взаимосвязанных населенных пунктов*. Элементами этой системы являются поселки. Расположение дорог между поселками определяет *структуру* данной системы.

Структура — это определенный порядок объединения элементов, составляющих систему.

Элементы системы (они изображены овалами) называются ***вершинами графа***. Связи между элементами изображаются на графе линиями. Если линия направленная (т. е. со стрелкой), то она называется ***дугой***. Если нет стрелки, то это ***ребро***. Две вершины, соединенные ребром или дугой, называются ***смежными***.

Разберемся, почему граф на рис. 2.9 содержит ненаправленные линии. Всякая связь имеет определенный смысл, ее можно как-то назвать. На нашем графе связи называются: «соединены дорогой». Понятно, что если поселок А соединен дорогой с поселком Б, то, значит, и Б соединен с А. Здесь не может быть односторонней связи.

Такие связи называются ***симметричными***. Симметричные связи на графе — это ребра.

Простейшей структурой системы является линейная структура. Если, например, населенные пункты А, Б, В, Г расположены вдоль одной дороги, то система дорожной связи между ними имеет линейную структуру (рис. 2.10).

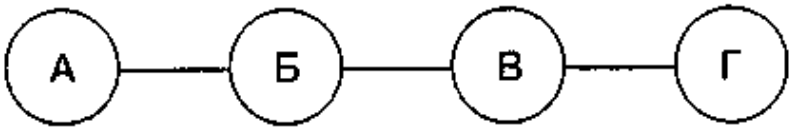
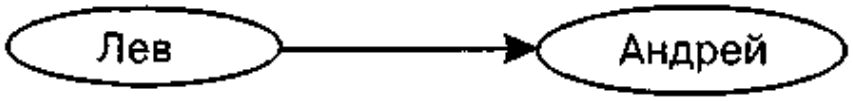


Рис. 2.10

А теперь рассмотрим пример системы с несимметричными связями. Изобразим в форме графа систему, состоящую из двух человек: отца (его зовут Лев) и сына (Андрей):



Стрелка (дуга) отражает связь «быть отцом». В таком случае ясно, что справедлив факт «Лев является отцом для Андрея», но не наоборот. Этот факт и представлен на графе.

Виды графов

Граф, в котором все связи изображены дугами, называется ***ориентированным графом***.

На рис. 2.11 изображен ориентированный граф, содержащий информацию о мужском составе некоторой семьи.

Здесь дуги обозначают связь «быть отцом», т. е. Лев является отцом для Андрея и Петра, Андрей — отец Алексея, а Петр — отец Михаила и Дмитрия. У каждого человека может быть только один отец, но несколько детей. Поэтому в

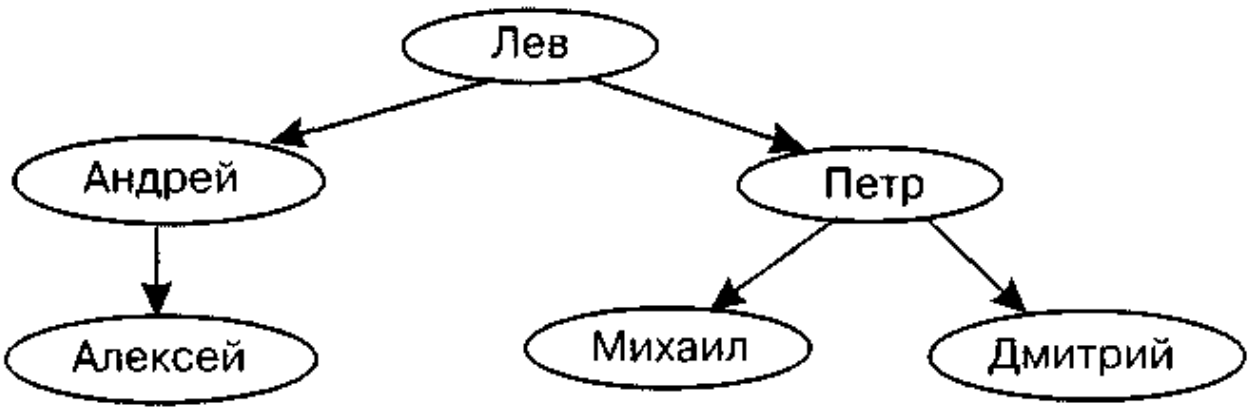


Рис. 2.11. Ориентированный граф родственных связей

каждую вершину графа может входить только одна стрелка (дуга), а выходить — несколько. Такой граф представляет собой ***генеалогическое дерево***.

Деревом называют граф, в котором нет петель, т. е. связанных по замкнутой линии вершин.

Граф на рис. 2.9 нельзя назвать деревом. В нем очевидны петли: Дачи — Подгорная — Бобры — Елово — Дачи и пр. Если бы, например, между Елово и Бобрами, Елово и Озёрной не было дорог, то такой граф был бы деревом.

У дерева на рис. 2.11 вершина «Лев» является ***корнем***. От корня идут ветви, по которым можно добраться до любой другой вершины дерева только по одному пути. Конечные вершины каждой ветви называются ***листьями***.

Иерархические системы и деревья

Название «дерево» выбрано не случайно, потому что очевидно некоторое внешнее сходство с деревом-растением. Правда, дерево-граф выглядит перевернутым, но это связано с нашей привычкой писать сверху вниз, а не наоборот. А строить дерево удобно, начиная с корня.

Система, информационная модель которой представляется в виде дерева, называется **иерархической системой**.

Как правило, *иерархическую структуру имеют общественные системы, между частями которых установлены отношения подчиненности* (например: директор — начальник цеха — начальник участка — бригадир — рабочий); *системы, между частями которых существуют отношения вхождения одних в другие* (например: федерация, республика, область, город, район). На рис. 2.12 вы видите «географическое дерево». Его корнем является вершина «Планета Земля», листьями — города.

Вершины дерева на рис. 2.12 четко разделены на пять уровней. Дерево на рис. 2.11 имеет три уровня.

Для дерева выполняется правило: *вершины верхнего уровня связаны с вершинами нижнего уровня как «один ко многим»*. Один континент содержит множество стран, одна страна — множество регионов, а не наоборот.

Иерархическими являются различные *системы классификации в науке*. Например, в биологии весь животный мир

Рис. 2.12. Граф иерархической системы («географическое дерево»)

Земли рассматривается как система, которая делится на типы животных, типы делятся на классы, классы состоят из отрядов, отряды — из семейств, семейства делятся на роды, роды — на виды. Следовательно, система животных имеет шестиуровневую иерархическую структуру.

Сети

А теперь рассмотрим систему, изображенную в виде графа на рис. 2.13. Этот граф содержит ту же информацию, что и табл. 2.4, о посещении четырьмя учениками школы различных факультативов. Русанов посещает геологию и танцы, Семенов — геологию и цветоводство, Зотова — цветоводство и танцы, Шляпина — танцы.

Здесь имеются два уровня вершин, но правило «один ко многим» не выполняется. Один ученик может посещать множество факультативов; один факультатив посещает множест-

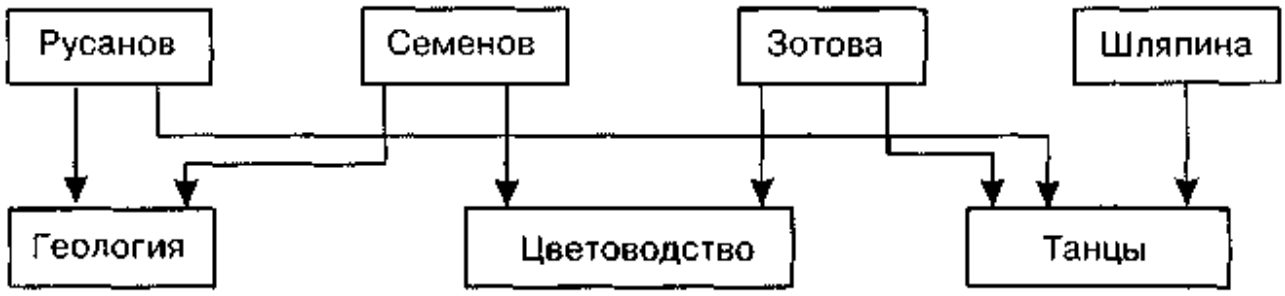


Рис. 2.13. Граф, имеющий структуру типа «сеть»

во учеников. Такой принцип связи называют «многие ко многим». Граф с такой структурой носит название *«сеть»*.

Элементы сети не всегда делятся по уровням. В сети возможно произвольное соединение элементов: каждый элемент может быть соединен с любым другим. Граф на рис. 2.11 — пример дорожной сети.

Коротко о главном

Система — это объект, состоящий из взаимосвязанных элементов и существующий как единое целое.

Структура — это определенный порядок объединения элементов, составляющих систему.

С помощью информационной модели-графа можно выразить информацию о составе и структуре системы.

Элементы системы изображаются овалами и называются вершинами графа; связи изображаются линиями, соединяющими вершины.

Две вершины, соединенные линией, называются смежными.

Несимметричное отношение изображается направленной линией (дугой); симметричное — ненаправленной линией (ребром).

Линейная структура — простейшая структура системы.

Граф, в котором все связи изображены дугами, называется ориентированным графом.

Деревом называют граф, в котором нет петель, т. е. связанных по замкнутой линии вершин. Между вершинами соседних уровней дерева в направлении сверху вниз выполняется принцип связи «один ко многим».

Система, информационная модель которой представляется в виде дерева, называется иерархической системой.

Сеть — это граф системы с произвольным принципом связи.

? *Вопросы и задания*

1. Что такое система; структура?
2. Назовите элементы, составляющие следующие системы: автомобиль, молекула воды, компьютер, магазин, Солнечная система, семья, футбольная команда, армия. Обоснуйте взаимозависимость элементов этих систем.

3. Что такое граф? Какую информацию он может нести в себе?
4. Как на графе изображаются элементы системы и отношения между ними?
5. Что значит «симметричное отношение», «несимметричное отношение»? Как они изображаются на графе? Приведите примеры.
6. Дайте имена возможным связям между следующими объектами и изобразите связи между ними в форме графа: брат и сестра; ученик и школа; Саша и Маша; Москва и Париж; министр, директор, рабочий; Пушкин и Дантес; компьютер и процессор.
7. Граф с какими свойствами называют деревом? Что такое корень дерева, ветви, листья?
8. Какие системы называют иерархическими?
9. Можно ли систему файлов в MS Windows (и ей подобных) назвать иерархической? Какой смысл имеют связи между ее элементами? Что в ней является листьями, ветвями, корнем?
10. Нарисуйте в виде графа систему, состоящую из четырех одноклассников, между которыми существуют следующие связи (взаимоотношения):
дружат: Саша и Маша, Саша и Даша, Маша и Гриша, Гриша и Саша.
Глядя на полученный граф, ответьте на вопрос: с кем Саша может поделиться секретом, не рискуя, что он станет известен кому-то другому?

2.2. Объектно-информационные модели

Основные темы параграфа:
что такое объект;
свойства объекта;
состояние объекта;
поведение объекта;
классы объектов;
наследование; иерархические системы классов.

Что такое объект

А сейчас рассмотрим еще один подход к информационному моделированию, который называется ***объектно-ориентированным подходом***. Главным понятием здесь является понятие «объект». Поясним его.

Объект — это некоторая часть окружающей нас действительности.

С точки зрения восприятия человеком объекты можно разделить на следующие группы:

- осязаемые или видимые объекты (например: кресло, автомобиль, мост);
- образы, созданные мышлением (например: стихотворение, музыкальное произведение, математическая теорема).

Свойства объекта

Объектно-информационная модель объекта должна отражать некоторый набор его свойств.

Свойства объекта отличают его от других объектов.

Рассмотрим примеры объектов и их свойств (табл. 2.6).

Таблица 2.6. Свойства объектов

Имя объекта	Свойства
Мой преподаватель	Имя Стаж работы Читаемый курс
Мой жесткий диск	Объем Количество занятой памяти
Важный документ	Имя Дата создания Объем занимаемой памяти Местоположение

У каждого конкретного объекта *свойства имеют определенные значения*. В нашем примере добавим значения свойств объектов (табл. 2.7).

Таблица 2.7. Свойства и значения объектов

Имя объекта	Свойства	Значения свойств
Мой преподаватель	Имя	Иванов Иван Иванович
	Стаж работы	10 лет
	Читаемый курс	Математика
Мой жесткий диск	Объем	10 Гб
	Количество занятой памяти	5 Гб
Важный документ	Имя	main.doc
	Дата создания	20 июня 2002 года
	Объем занимаемой памяти	50 Кб
	Местоположение	C:\Documents

Состояние объекта

Состояние объекта характеризуется перечнем всех возможных его свойств и текущими значениями каждого из этих свойств. Изменение состояния объекта отражается в его информационной модели изменением значений его свойств. Как правило, объекты не остаются неизменными. Например, растет стаж работы учителя И. И. Иванова; на жестком диске изменяется объем занятой памяти; документ может быть перенесен на другой диск, в другую папку и пр. Все эти процессы в информационной модели отражаются изменениями значений свойств.

Поведение объекта

В объектно-информационной модели отражаются не только свойства, но также и поведение объекта.

Поведение объекта — действия, которые могут выполняться над объектом или которые может выполнять сам объект.

Опишем поведение объектов из нашего примера (табл. 2.8).

Таблица 2.8. Поведение объектов

Имя объекта	Поведение (действия)
Мой преподаватель	Чтение лекции Прием экзамена Проведение консультации
Мой жесткий диск	Форматирование Копирование
Важный документ	Открытие Чтение Запись Копирование Переименование

Классы объектов

А сейчас введем еще одно очень важное понятие для объектно-информационного моделирования — понятие ***класса***.

> **Класс объектов** определяет множество объектов, обладающих одинаковыми свойствами и поведением.

Говорят, что объект является ***экземпляром*** какого-либо класса. Все преподаватели обладают одним и тем же набором свойств (имя, стаж работы, читаемый курс) и поэтому образуют класс. Присвоим этому классу имя *«Преподаватель»*. Каждый конкретный преподаватель — экземпляр этого класса (или объект). Следовательно, *«Мой преподаватель»* — экземпляр класса *«Преподаватель»*. Аналогично можно ввести класс *«Жесткий диск»*, объединив в нем все жесткие диски. Тогда *«Мой жесткий диск»* — экземпляр класса *«Жесткий диск»*. Если принять во внимание, что класс *«Документ»* описывает свойства и поведение всех документов, то *«Важный документ»* — экземпляр класса *«Документ»*.

Таким образом, экземпляр класса (объект) — это конкретный предмет или образ, а класс определяет множество объектов с одинаковыми свойствами и поведением. Класс может порождать произвольное число объектов, однако любой объект относится к строго фиксированному классу.

Объектно-информационные модели имеют иерархическую структуру (дерево). Иерархичность проявляется в том, что некоторый класс сам может быть подмножеством другого, более широкого класса. Вот пример иерархической классификации из биологии: вид «*Насекомые*» включает в себя два отряда: «*Крылатые*» и «*Бескрылые*»; в свою очередь «*Крылатые*» насекомые делятся на следующие подотряды: «*Мотыльки*», «*Бабочки*», «*Мухи*» и т. д. (рис. 2.14).

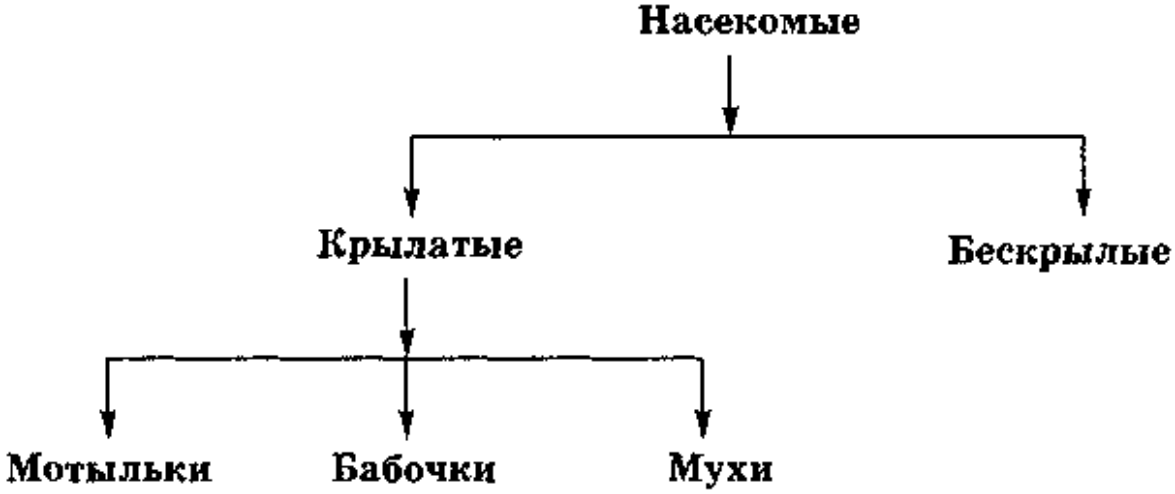


Рис. 2.14. Фрагмент классификации насекомых

Наследование. Иерархические системы классов

В такой иерархической структуре между классами определяется отношение наследования.

Наследование — это такое отношение между классами, когда один класс повторяет свойства и поведение другого класса.

Такой способ классификации, в частности, хорошо согласуется с механизмом биологического наследования в мире насекомых. Классы верхних уровней являются более общими по отношению к нижним. При спуске по дереву каждый следующий класс является более специфичным и в то же время наследует все свойства своих предшественников. Класс, свойства и поведение которого наследуются, называется ***суперклассом*** (или базовым классом). Производный от суперкласса класс называется ***подклассом***. В нашем примере «*Насекомые*» — суперкласс для подклассов «*Крылатые*», «*Бескрылые*», «*Мотыльки*», «*Бабочки*», «*Мухи*», а «*Крылатые*» — суперкласс для подклассов «*Мотыльки*», «*Бабоч-*

ки», *«Мухи»*. В подклассе дополняются свойства и уточняется поведение объектов суперкласса. При определении класса *«Мухи»* нет необходимости вводить свойство «наличие крыльев», так как это свойство наследуется из суперкласса *«Крылатые»*.

Вот еще один пример. Рассмотрим систему классов, отражающих сведения о различных видах транспорта (рис. 2.15).

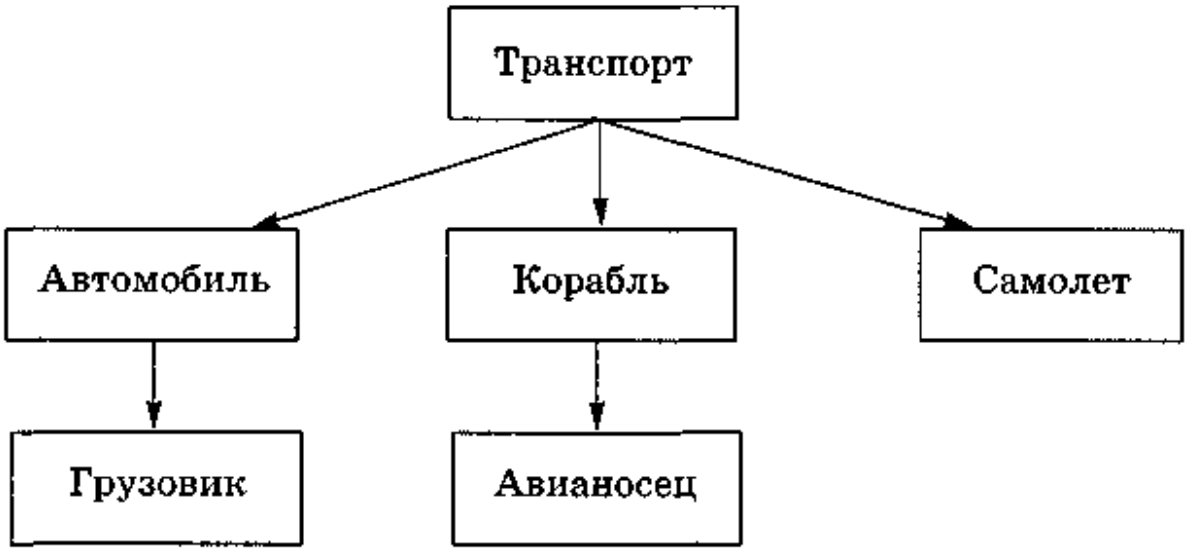


Рис. 2.15. Система классификации транспортных средств

Свойства и поведение, присущие каждому классу, отражены в табл. 2.9, где звездочками отмечены наследуемые свойства и действия.

Все самые общие свойства располагаются в суперклассе *«Транспорт»*. Эти свойства наследуются классами *«Автомобиль»*, *«Грузовик»*, *«Корабль»*, *«Авианосец»* и *«Самолет»*. Кроме того, свойства «марка» и «пробег» наследуются классом *«Грузовик»* от базового класса *«Автомобиль»*; а свойства «нахождение» и «водоизмещение» наследуются классом *«Авианосец»* от базового класса *«Корабль»*. В столбце «Поведение (действия)» отражено наследование действий.

А теперь определим экземпляры классов (объекты) и значения их свойств (табл. 2.10).

В табл. 2.10 мы определили три экземпляра класса *«Автомобиль»*. Для определения экземпляров (объектов) других классов необходимо аналогичным образом задать значения свойств.

Таблица 2.9. Сведения о видах транспорта

Имя класса	Свойства	Поведение (действия)
Транспорт	Скорость Мощность Цена	Движение вперед
Автомобиль	Скорость* Мощность* Цена* Марка Пробег	Движение вперед* Движение назад
Грузовик	Скорость* Мощность* Цена* Марка* Пробег* Грузоподъемность	Движение вперед* Движение назад
Корабль	Скорость* Мощность* Цена* Нахождение Водоизмещение	Движение вперед* Движение назад
Авианосец	Скорость* Мощность* Цена* Нахождение* Водоизмещение* Количество самолетов	Движение вперед* Движение назад* Запуск самолета
Самолет	Скорость* Мощность* Цена* Название Максимальная высота полета	Движение вперед* Движение вверх Движение вниз

Подводя итог, сделаем вывод о том, что такое объектно-информационная модель (ОИМ). ОИМ включает в себя описание иерархической системы классов, между которыми действуют отношения наследования. Для каждого класса определяется совокупность присущих ему свойств и действий (поведения), указывается, какие свойства и действия являются наследуемыми, а какие — специфическими. Для

каждого объекта, входящего в ОИМ, указывается класс, экземпляром которого он является, а также конкретные значения свойств.

Таблица 2.10. Экземпляры классов

Имя класса	Имя экземпляра класса (объекта)	Свойства	Значения свойств
Автомобиль	Мой автомобиль	Скорость*	130 км/ч
		Мощность*	85 л. с.
		Цена*	156 000 руб.
		Марка	Нива
		Пробег	10 000 км
Автомобиль	Автомобиль друга	Скорость*	200 км/ч
		Мощность*	95 л. с.
		Цена*	180 000 руб.
		Марка	ВАЗ 2110
		Пробег	15 000 км
Автомобиль	Машина соседа	Скорость*	250 км/ч
		Мощность*	408 л. с.
		Цена*	123 000 $
		Марка	Мерседес 600
		Пробег	20 000 км

Коротко о главном

Объект — часть окружающей действительности.

Информационная модель объекта включает в себя описание его свойств и поведения (действий).

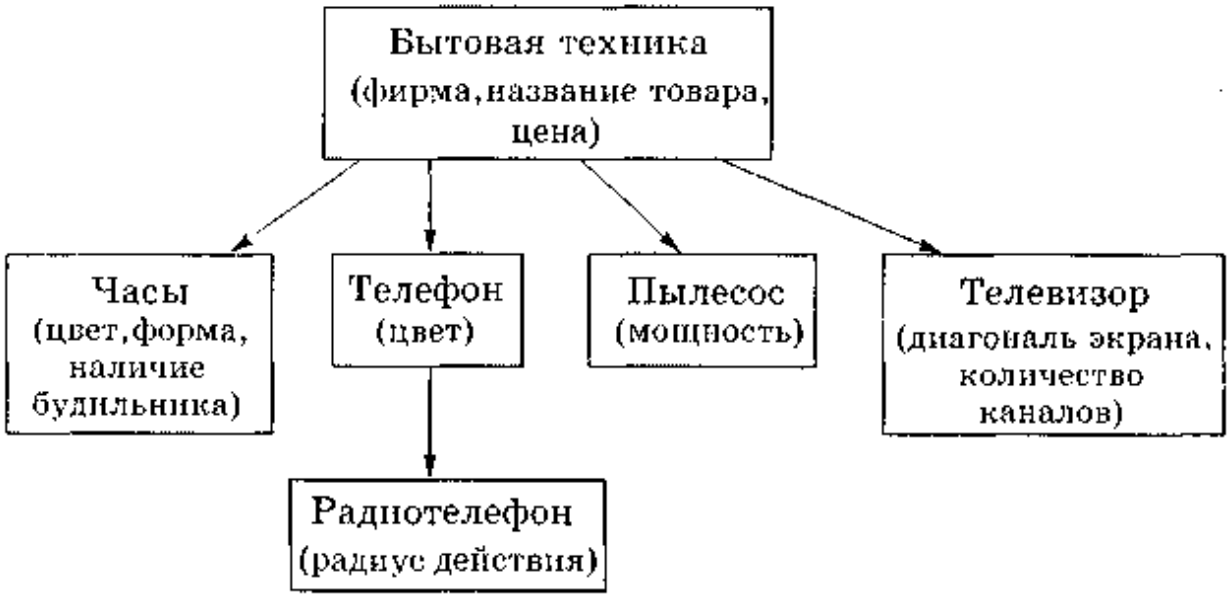
Класс объектов определяет множество объектов с одинаковым набором свойств и действий.

В иерархиях классов действует отношение наследования между суперклассами и подклассами.

Объектно-информационная модель включает в себя описание иерархии классов и отдельных объектов с конкретными значениями свойств.

Вопросы и задания

1. Приведите примеры объектов (не менее трех), их свойств, значений свойств и поведения.
2. Задан набор классов, которые содержат сведения о различных видах бытовой техники. В скобках указаны свойства.



Дополните каждый класс поведением. Опишите экземпляры классов и значения их свойств.

3. Заданы классы: «Фигура», «Эллипс», «Закрашенный квадрат», «Равносторонний треугольник», «Треугольник», «Круг, «Равнобедренный треугольник», «Прямоугольник, «Квадрат». Классифицируйте эти объекты, используя механизм наследования. Опишите свойства и поведение каждого класса.
4. Для предметной области «Туристические фирмы» опишите набор классов, которые должны содержать сведения о различных фирмах, а также об ассортименте предоставляемых туров и услуг.

Дополнение к главе 5

5.1. Автоматизированные и автоматические системы управления

Основные темы параграфа:

- *что такое АСУ и что такое САУ;*
- *простые автоматы;*
- *ЦАП – АЦП преобразование;*
- *схема САУ;*
- *управление в режиме реального времени;*
- *контроллеры и микропроцессоры в САУ.*

Что такое АСУ и что такое САУ

Компьютеры помогают решать задачи управления в самых разных масштабах: от управления станком или транспортным средством до управления производственным процессом на предприятии или даже целой отраслью экономики государства.

Конечно, поручать компьютеру полностью, без участия человека, руководить предприятием или отраслью экономики сложно, да и не безопасно. Для управления в таком масштабе создаются компьютерные системы, которые называются ***автоматизированными системами управления*** (***АСУ***). Такие системы работают вместе с человеком.

АСУ помогает руководителю получить необходимую информацию для принятия управляющего решения, а также может предложить наиболее оптимальные варианты таких решений. Однако окончательное решение принимает человек.

В АСУ используются самые современные средства информационных технологий: базы данных и экспертные системы, методы математического моделирования, машинная графика и пр.

С распространением персональных компьютеров технической основой АСУ стали компьютерные сети. В рамках одного предприятия это локальные компьютерные сети. Автоматизированные системы управления, работающие в масштабах отрасли, в государственных масштабах, используют глобальные компьютерные сети.

Другим вариантом применения компьютеров в управлении являются ***системы автоматического управления (САУ)***. Объектами управления в этом случае чаще всего выступают технические устройства (станок, ракета, химический реактор, ускоритель элементарных частиц).

В **САУ** все операции, связанные с процессами управления (сбор и обработка информации, формирование управляющих команд, воздействие на управляемый объект) происходят автоматически, без непосредственного участия человека.

Простые автоматы

Устройства автоматического управления стали создаваться задолго до появления первых ЭВМ. Как правило, они основаны на использовании каких-либо физических явлений. Например, автоматический регулятор уровня воды в баке основан на выталкивающем действии воды на поплавок регулятора; автоматические предохранители в электрических сетях основаны на тепловом действии электрического тока; система автоматического регулирования освещенности в помещении использует явление фотоэффекта. Существуют и более сложные примеры бескомпьютерного автоматического управления.

Преимущество компьютерных систем автоматического управления перед такими устройствами в их большей «интеллектуальности», в возможности осуществлять более сложное управление, чем простые автоматы.

ЦАП – АЦП преобразование

Рассмотрим ситуацию, в которой объектом управления является техническое устройство (лабораторная установка, бытовая техника, транспортное средство или промышленное оборудование), а управляющим объектом — система автоматического управления.

Компьютер работает с двоичной информацией, помещенной в его память. Управляющая команда, выработанная программой, в компьютере имеет форму двоичного кода. Чтобы она превратилась в физическое воздействие на управляемый объект, необходимо преобразование этого кода в электрический сигнал, который приведет в движение «рычаги» управления объектом. Такое преобразование из двоичного кода в электрический сигнал называют цифро-аналоговым преобразованием. Выполняющий такое преобразование прибор называется ***ЦАП*** (***цифро-аналоговый преобразователь***).

Приборы, которые дают информацию о состоянии объекта управления, называются ***датчиками***. Они могут показывать, например, температуру, давление, деформации, напряженности полей и пр. Эти данные необходимо передать компьютеру по линиям обратной связи. Если показания датчиков имеют аналоговую форму (электрический ток или потенциал), то они должны быть преобразованы в двоичную цифровую форму. Такое преобразование называется аналого-цифровым, а прибор, его выполняющий, — ***АЦП*** (***аналого-цифровой преобразователь***)*.

Схема САУ

Все сказанное отражается в схеме, приведенной на рис. 5.16. Такая система работает автоматически, без участия человека.

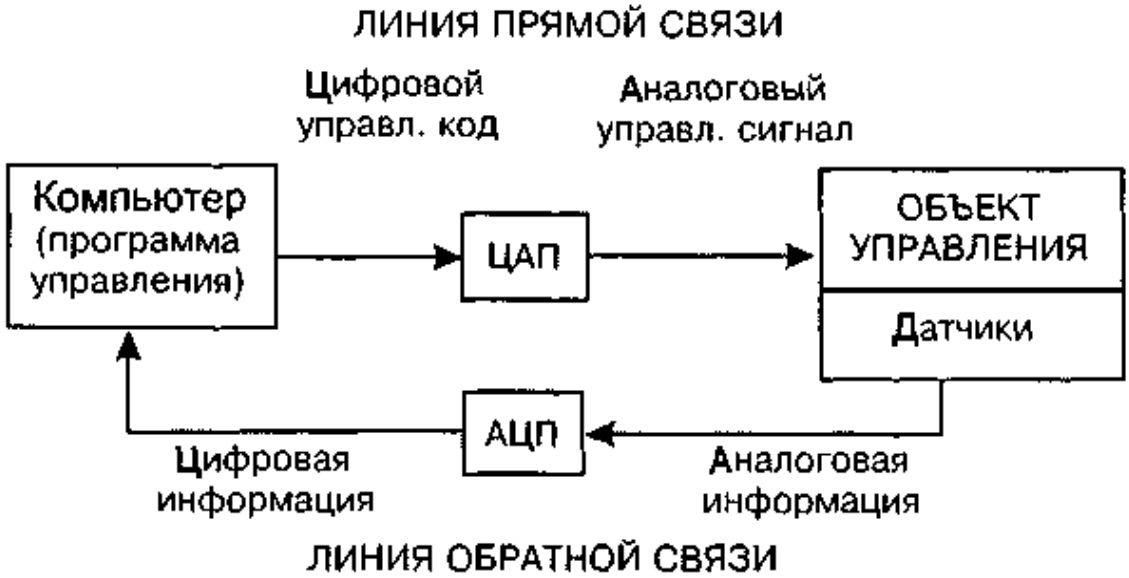


Рис. 5.16. Схема системы автоматического управления

---

* О ЦАП- и АЦП-преобразованиях речь уже шла в учебнике 8 класса (§ 24).

Управление в режиме реального времени

Системы автоматического управления работают в ***режиме реального времени***. Легко понять, что всякая управляющая команда должна быть отдана вовремя. Любой процесс происходит с какой-то скоростью, в каких-то временны́х рамках.

Режим, при котором управляющая система работает синхронно с объектом управления, называется **режимом реального времени**.

При составлении программ управления в реальном времени программистам приходится решать вопрос не только о том, в каком порядке отдавать команды, но и в какие моменты времени это делать. Значит, система управления должна взаимодействовать с прибором, отмеряющим время: часами, таймером.

Напомним, что в составе персонального компьютера есть устройство, называемое ***генератором тактовой частоты***. Работа всех узлов компьютера синхронизируется по тактовой частоте. Вот на эти «часы» и ориентируется программа управления в режиме реального времени.

Контроллеры и микропроцессоры в САУ

Не следует думать, что в системах автоматического управления всегда используется универсальный компьютер с полным комплектом всех устройств (клавиатура, дисплей и пр.). Конечно, бывает и такое, но очень часто для этих целей применяются специализированные устройства — ***контроллеры***. В их состав обязательно входят процессор, память и необходимые средства связи с объектом управления. Если управляющая система все время должна работать по одной и той же программе, то эта программа хранится в постоянной памяти (ПЗУ).

В простейших случаях для автоматического управления используются микропроцессоры, встроенные в управляемое устройство. Например, очень часто микропроцессоры применяются в транспортных средствах: автомобилях, самолетах, поездах. Каждый микропроцессор выполняет свою отдельную функцию, управляет работой определенного узла. Например, в автомобилях используется микропроцессор, управляющий работой карбюратора — устройства, регулиру-

ющего подачу топлива в двигатель. Такое автоматическое управление снижает расход горючего, повышает КПД двигателя.

Современные самолеты «нашпигованы» многочисленной электроникой: от микропроцессоров, управляющих отдельными приборами, до бортовых компьютеров, прокладывающих маршрут полета, т. е. выполняющих функции штурмана.

Коротко о главном

Автоматизированные системы управления (АСУ) помогают человеку в сборе информации и принятии управляющих решений.

В системах автоматического управления (САУ) все операции, связанные с процессами управления, происходят автоматически, без непосредственного участия человека, по заранее составленной программе.

В САУ на линии прямой связи для преобразования двоичной информации в аналоговый сигнал используется прибор ЦАП (цифро-аналоговый преобразователь); на линии обратной связи для преобразования аналогового сигнала в двоичный код используется прибор АЦП (аналого-цифровой преобразователь).

Управление в САУ происходит в режиме реального времени.

Вопросы и задания

1. В чем различие между автоматизированными системами управления (АСУ) и системами автоматического управления (САУ)?
2. Какие аппаратные компоненты входят в систему управления техническим устройством с помощью компьютера?
3. Для чего нужны устройства ЦАП и АЦП?
4. Что такое управление в режиме реального времени?
5. Приведите примеры использования встроенных в оборудование микропроцессоров.

Дополнение к главе 6

6.1. Поиск наибольшего и наименьшего элементов массива

Основные темы параграфа:

- *поиск максимума и минимума в электронной таблице;*
- *блок-схемы алгоритмов поиска максимума и минимума в массиве;*
- *программа на Паскале поиска максимума и минимума в массиве.*

Поиск максимума и минимума в электронной таблице

Одной из типовых задач обработки массивов является поиск наибольшего или наименьшего значения среди значений его элементов. Построим алгоритм решения этой задачи и составим программу на Паскале. Для примера возьмем итоговые данные чемпионата России по футболу в премьер-лиге за 2003 год.

На рис. 6.12 показана электронная таблица с итогами чемпионата. В столбце А расположены названия команд, в столбце В — количество очков, набранных каждой командой. Команды перечислены в алфавитном порядке. Победителем является команда, набравшая наибольшее количество очков. Команда, набравшая очков меньше всех других, в следующем сезоне покидает премьер-лигу.

Для определения максимального значения в электронной таблице существует функция МАКС( ), а для нахождения минимального значения — функция МИН( ). В ячейке В17 записана формула МАКС(B1:B16), а в ячейке В18 — формула МИН(B1:B16). Результаты вы видите в таблице. Отсюда делаем вывод: чемпионом России стала команда ЦСКА, а на последнем месте — Черноморец.

Блок-схемы алгоритмов поиска максимума и минимума в массиве

Разберемся, как же программируется определение максимального и минимального значения в числовом массиве.

Начнем с поиска максимума. Опишем алгоритм на Алгоритмическом языке.

	A	B
1	ДИНАМО	46
2	ЗЕНИТ	56
3	КРЫЛЬЯ СОВЕТОВ	42
4	ЛОКОМОТИВ	52
5	РОСТОВ	34
6	РОТОР	32
7	РУБИН	53
8	САТУРН	45
9	СПАРТАК	36
10	СПАРТАК-АЛАНИЯ	31
11	ТОРПЕДО	43
12	ТОРПЕДО-МЕТАЛЛУРГ	29
13	УРАЛАН	28
14	ЦСКА	59
15	ЧЕРНОМОРЕЦ	24
16	ШИННИК	47
17	Максимал. кол-во очков	59
18	Минимал. кол-во очков	24

Рис. 6.12

Пусть в целочисленный массив $B[1:16]$ заносятся очки команд в том порядке, в каком они расположены в таблице на рис. 6.12. Максимальное количество очков получим в переменной *MaxB*. Кроме того, найдем еще и номер команды, занявшей первое место. Для этого будет использоваться переменная *Nmax*.

Рассмотрим алгоритм решения задачи. Ниже приведен полный алгоритм на Алгоритмическом языке, а на рис. 6.13 — фрагмент блок-схемы, относящийся только к выбору максимального элемента (без ввода и вывода).

```
алг Премьер-лига
цел таб B[1:16]
цел I, MaxB, Nmax
нач
    {Цикл ввода}
    для I от 1 до 16 повторять
```

```
нц
      Вывод "B[", I , "]="
      Ввод B[I]
кц
{Цикл суммирования}
MaxB:=B[1]; Nmax:=1
для I от 2 до 16 повторять
нц
      если B[I]>MaxB
      то MaxB:=B[I];
         Nmax:=I
      кв
кц
Вывод ("Максимальное число очков:", MaxB)
Вывод("Номер команды-победителя:", Nmax)
кон
```

Идея алгоритма состоит в следующем. В переменную *MaxB* заносится значение первого элемента массива, в переменную *Nmax* — единица, т. е. номер первого элемента. Затем в цикле последовательно с *MaxB* сравниваются все остальные элементы массива *B*. Если очередной элемент оказывается больше текущего значения *MaxB*, то его значение заносится в *MaxB*, а его номер — в *Nmax*. Когда закончится

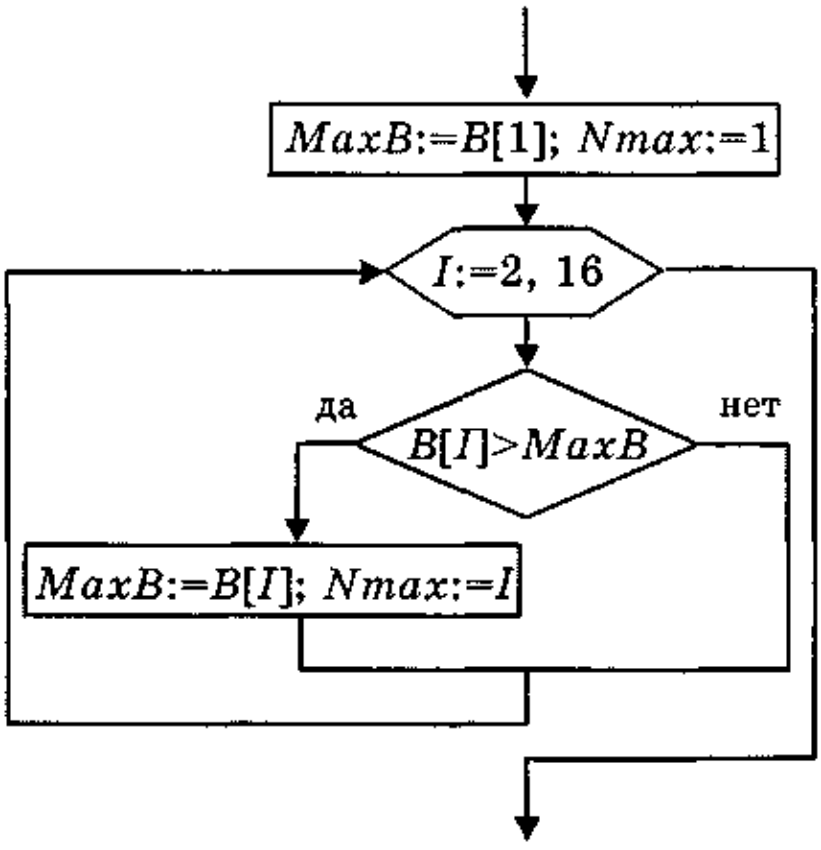


Рис. 6.13

цикл, в $MaxB$ останется наибольшее число из всего массива, а в $Nmax$ — его номер.

Теперь нетрудно догадаться, как искать минимальное значение в массиве и его номер. Если для этого использовать в программе переменные $MinB$ и $Nmin$ и в условии ветвления заменить знак отношения «больше» на «меньше», то получим нужный алгоритм. Он показан блок-схемой на рис. 6.14.

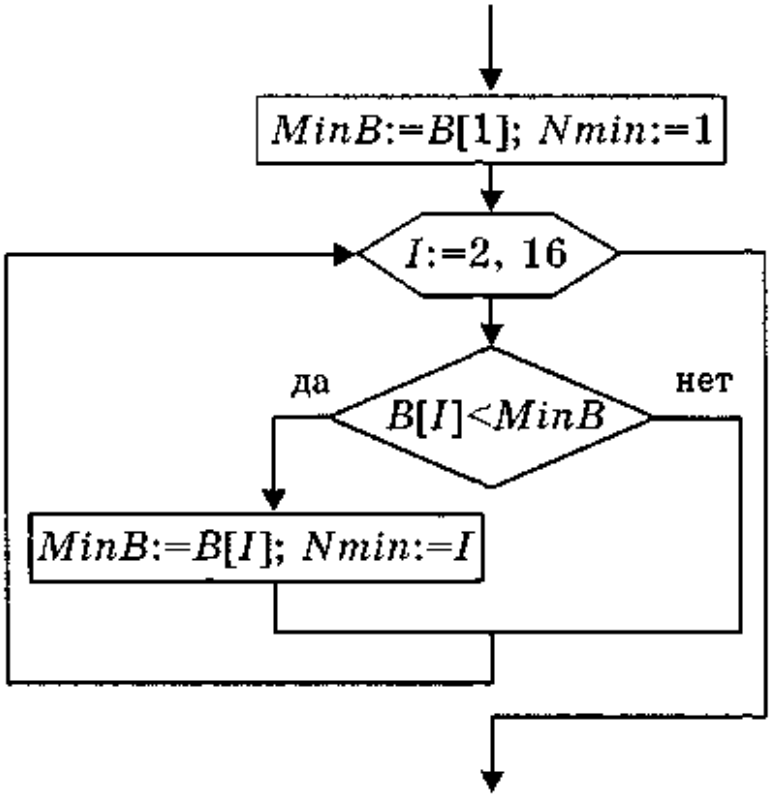


Рис. 6.14

Если в алгоритме нужно одновременно искать максимальное и минимальное значения, то соответствующие ветвления можно объединить в одном цикле. Именно так сделано в приведенной ниже программе на Паскале.

Программа на Паскале поиска максимума и минимума в массиве

Составим программу на Паскале. Но в эту программу мы внесем еще некоторые новые детали. Хотелось бы в итоге работы программы получить не номера, а названия команды-победителя и команды, занявшей последнее место. Но для этого названия всех команд чемпионата должны быть организованы в массив и введены как исходные данные. В программе такой массив назван `Team[1..16]` и тип его элементов объявлен как `string`.

String — это строковый тип данных Паскаля. Величина такого типа может принимать значение, представляющее собой произвольную символьную последовательность (в том числе и из русских букв), длина которой не должна превышать 255. Для названий команд это вполне подходящие условия.

```
Program Premier-liga;
var B: array[1..16] of integer;
    Team: array[1..16] of string;
    MaxB, MinB, Nmax, Nmin, I: integer,
begin
{Ввод названий команд и набранных ими очков}
writeln('Введите названия команд и полученные
        ими очки')
    for I:=1 to 16 do
    begin
      write(I,' Название: '); Readln(Team[I]);
      write('Очки: '); Readln(B[I])
    end;
{Поиск наибольшего и наименьшего значений и их
номеров}
MaxB:=B[1]; Nmax:=1; MinB:=B[1]; Nmin:=1;
for I:=2 to 16 do
begin
  {Выбор максимума}
  if B[I]>MaxB then
  begin
    MaxB:=B[I];
    Nmax:=I
  end;
   {Выбор минимума}
  if B[I]<MinB then
  begin
    MinB:=B[I];
    Nmin:=I
  end
end;
{Вывод результатов}
writeln('Команда-победитель чемпионата ',
        Team[Nmax], 'набрала ', MaxB, 'очков');
writeln('На последнем месте',Team[Nmin],
        ' с ',MinB,'очками')
end.
```

Обратите внимание на то, как определяется название команды-победителя и команды, занявшей последнее место. Это делается по значениям индексов максимального и минимального элементов массива B: Nmax и Nmin. В переменной Team[Nmax] находится название чемпиона, а в переменной Team[Nmin] — название последней команды в чемпионате.

При выполнении программы на экране будет отражено следующее:

```
Введите названия команд и полученные ими очки
1 Название: ДИНАМО
Очки:  46
2 Название: ЗЕНИТ
Очки: 56
3 Название: КРЫЛЬЯ СОВЕТОВ
Очки:  42
......
16 Название : ШИННИК
Очки: 47
Команда-победитель чемпионата ЦСКА набрала 59
очков
На последнем месте ЧЕРНОМОРЕЦ с 24 очками
```

Коротко о главном

Алгоритм выбора максимального (минимального) значения в массиве имеет структуру цикла с вложенным неполным ветвлением.

Для обработки последовательностей символов в Паскале имеется строковый тип данных: string.

? *Вопросы и задания*

1. Придумайте собственные примеры данных, которые можно было бы представить в виде строкового массива.
2. Представьте себе, что две команды набрали 59 очков. Например, ЦСКА и ЗЕНИТ. Номер какой команды был бы выведен в качестве результата?
3. При условии предыдущего задания определите, какие будут выведены результаты, если в операторе ветвления, где отбирается максимальное значение, заменить знак отношения «>» на «>=»?

4. Введите в компьютер программу `Premier-liga`. Выполните ее, получите результаты. Сравните с результатами, приведенными в параграфе.
5. По условиям чемпионата 2003 года из премьер-лиги выбывают две последние в турнирной таблице команды. Составьте программу, определяющую обе команды, выбывающие из премьер-лиги.

6.2. Сортировка массива

Основные темы параграфа:

- *алгоритм сортировки методом пузырька;*
- *программа на Паскале сортировки методом пузырька.*

Известно, что данные в электронной таблице можно сортировать по возрастанию или убыванию значений в столбцах. Для задачи с таблицей футбольного чемпионата естественным действием была бы сортировка по убыванию значений набранных очков. Тогда вверху таблицы останется победитель чемпионата, а в нижней строчке — команда, занявшая последнее место. На рис. 6.15 показана такая отсортированная таблица. Из нее мы получаем исчерпывающую информацию об итогах чемпионата: кто какое место занял.

	A	B
1	ЦСКА	59
2	ЗЕНИТ	56
3	РУБИН	53
4	ЛОКОМОТИВ	52
5	ШИННИК	47
6	ДИНАМО	46
7	САТУРН	45
8	ТОРПЕДО	43
9	КРЫЛЬЯ СОВЕТОВ	42
10	СПАРТАК	36
11	РОСТОВ	34
12	РОТОР	32
13	СПАРТАК-АЛАНИЯ	31
14	ТОРПЕДО-МЕТАЛЛУРГ	29
15	УРАЛАН	28
16	ЧЕРНОМОРЕЦ	24

Рис. 6.15

Алгоритм сортировки методом пузырька

Рассмотрим, как программируется сортировка массива. Для решения этой задачи существует целый класс алгоритмов. Мы рассмотрим здесь только один из них, известный под названием «метод пузырька». Откуда такое название, станет ясно немного позже.

Проиллюстрируем идею метода пузырька на маленьком массиве из пяти чисел. Пусть это будет массив B[1:5], исходные значения в котором распределены случайным образом (рис. 6.16). Требуется отсортировать числа по убыванию.

Первый этап (первый проход). Последовательно сравниваются пары соседних чисел и упорядочиваются по убыванию. Сначала сравниваются *B*[1] и *B*[2]. Поскольку на втором месте должно стоять меньшее число, то числа меняются местами. *B*[1] становится равным 3, *B*[2] — равным 1. Затем упорядочиваются *B*[2] и *B*[3]. Их значения тоже переставляются: *B*[2]=2, *B*[3]=1. Затем упорядочиваются *B*[3] и *B*[4]. И наконец, *B*[4] и *B*[5]. В результате первого прохода минимальное число попадает на свое место: *B*[5]=1.

	B[1]	*B*[2]	*B*[3]	*B*[4]	*B*[5]
Исходные значения	1	3	2	4	5
1-й проход	3	2	4	5	1
2-й проход	3	4	5	2	1
3-й проход	4	5	3	2	1
4-й проход	5	4	3	2	1

Рис. 6.16. Иллюстрация метода пузырька

Алгоритм первого прохода можно описать так:

```
для I от 1 до 4 повторять
нц
        если B[I]>B[I+1] то
        X:=B[I]; B[I]:=B[I+1]; B[I+1]:=X
        кв
кц
```

Обмен значениями *B*[*I*] и *B*[*I*+1] происходит через третью переменную *X*. Вот теперь можно понять смысл образа пузырька! В результате прохода минимальное значение «всплывает» в конец массива, как воздушный пузырек в

воде всплывает на поверхность, подталкиваемый архимедовой силой.

Далее нужно повторять такие проходы еще три раза. После второго прохода на своем месте окажется $B[4]$, после третьего прохода — $B[3]$. После четвертого упорядочатся $B[2]$ и $B[1]$. Нетрудно понять, что при втором проходе не надо трогать $B[5]$, т. е. цикл должен повторяться для I от 1 до 3. При третьем проходе — от 1 до 2. И наконец, на четвертом проходе — от 1 до 1, т. е. всего 1 раз.

Следовательно, циклы, реализующие проходы, сами должны циклически повторяться. Причем, при каждом следующем повторении длина цикла должна уменьшаться на единицу. Отсюда вывод: *структура алгоритма должна представлять собой два вложенных цикла.*

Вот полный алгоритм сортировки массива $B[1:16]$:

```
алг Сортировка методом пузырька
цел таб B[1:16]
цел I, K, X
нач
    {Цикл ввода}
    для I от 1 до 16 повторять
    нц
          Вывод ('B[', I , ']=')
          Ввод (B[I])
    кц
    {Циклы сортировки}
    для K от 1 до 15 повторять
    нц
          для I от 1 до 16-K повторять
          нц
          если B[I]>B[I+1] то
          X:=B[I]; B[I]:=B[I+1]; B[I+1]:=X
          кц
    кц
кц
    {Вывод отсортированного массива}
    для I от 1 до 16 повторять
    нц
          Вывод ('B[', I , ']=', B[I])
    кц
кон
```

Здесь переменная *K* играет роль номера прохода. Для массива длиной 16 такие проходы требуется повторить 15 раз. Длина каждого *K*-го прохода равна 16–*K*.

Программа на Паскале сортировки методом пузырька

Теперь запишем программу на Паскале. Но мы ее немного усложним по сравнению с построенным алгоритмом. По условию исходной задачи нам нужно получить список команд в порядке занятых ими мест и число очков, полученных каждой команды. Следовательно, сортировать нужно не только массив B, но и массив Team. Делается это очень просто: в массиве Team параллельно с массивом B производятся те же самые перестановки. В конце работы программы на экран выводятся одновременно элементы обоих отсортированных массивов.

```
Program Premier-liga-2;
var B: array[1..16] of integer;
    Team: array[1..16] of string
    I, K, X: integer;
    St: string;
begin
{Ввод названий команд и набранных ими очков}
writeln('Введите названия команд и полученные
         ими очки')
    for I:=1 to 16 do
    begin
       write(I,' Название: '); Readln(Team[I]);
       write('Очки: '); Readln(B[I])
    end;
    {Сортировка}
    for K:=1 to 15 do
    for I:=1 to 16-K do
    if (B[I] > B[I+1]) then
       begin
          X:=B[I]; B[I]:=B[I+1]; B[I+1]:=X;
          St:=Team[I]; Team[I]:=Team[I+1];
          Team[I+1]:=St
       end;
    {Вывод отсортированной таблицы}
    for I:=1 to 16 do
    begin
    {Присоединение пробелов к названиям команд}
       For K:=1 to 18-length(Team[I]) do
```

```
            Team[I]:=Team[I]+'';
        {Вывод: место, команда, очки}
        writeln(I:2,'',Team[I]:18,B[I]:2)
    end
end.
```

Поясним новые средства Паскаля, которые применены в этой программе. Обмен значениями между элементами строкового массива Team должен происходить через переменную строкового типа. Для этого в программе используется переменная St.

Вывод результатов на экран организован так, чтобы на экране номера мест, занятых командами, названия команд и набранные очки выводились в три ровных столбца. Названия разных команд имеют разную длину. Самое длинное название у команды ТОРПЕДО-МЕТАЛЛУРГ состоит из 17 символов. Для выравнивания длин строк каждое название дополняется пробелами до 18 символов. Число добавляемых пробелов вычисляется так:

```
18-length(Team[I])
```

Здесь length( ) — это стандартная функция, вычисляющая длину строки (число символов), указанной в скобках. Например, для ЦСКА длина строки равна 4, а для ТОРПЕДО-МЕТАЛЛУРГ длина равна 17. Значит к ЦСКА добавится 14 пробелов, а к ТОРПЕДО-МЕТАЛЛУРГ — 1 пробел.

В операторе Team[I]:= Team[I]+''; используется операция присоединения символов «+». В данном случае присоединяется пробел. К строке Team[I] добавится столько пробелов, сколько раз повторится присоединение. После этого по команде

```
Writeln(I:2,'',Team[I]:18, B[I]:2)
```

в ровные колонки выведутся места, названия команд и очки. Результаты будут иметь на экране следующий вид:

```
1 ЦСКА                  59
2 ЗЕНИТ                 56
3 РУБИН                 53
. . . . . .
14 ТОРПЕДО-МЕТАЛЛУРГ    29
15 УРАЛАН               28
16 ЧЕРНОМОРЕЦ           24
```

Коротко о главном

Метод пузырька — алгоритм сортировки числового массива.

Структура алгоритма метода пузырька – два вложенных цикла с переменной длиной внутреннего цикла.

length() — функция определения длины строковой переменной.

В Паскале существует операция присоединения строк. Ее знак — «+».

? *Вопросы и задания*

1. Как пояснить название метода сортировки массива – «метод пузырька»?
2. Сколько проходов с перестановками элементов потребуется при сортировке массива из 100 чисел?
3. Введите в компьютер программу `Premier-liga-2`. Выполните ее, получите результаты. Сравните с результатами, приведенными в параграфе.
4. Внесите изменения в программу для того, чтобы получить список в обратном порядке (по возрастанию очков). Выполните программу.
5. Возможно, что массив окажется отсортированным до завершения всех проходов. В таком случае число повторений внешнего цикла можно сократить, и программа будет выполняться быстрее. Попробуйте усовершенствовать приведенную программу с учетом этого факта. Проверьте результат на тестах.
6. Если несколько команд набрали одинаковое количество очков, то места между ними распределяются по разнице забитых и пропущенных мячей: чем разница больше, тем место выше. Попробуйте усовершенствовать программу, учитывая это правило. Для этого в программу надо добавить массив с разницами мячей. Придумайте тест, на котором можно проверить работу программы.
7. Условие то же, что и в предыдущем задании. Но в качестве исходных данных вводится еще два массива: с числом забитых мячей и с числом пропущенных мячей каждой командой.

6.3. О языках программирования и трансляторах

Основные темы параграфа:

- *системы программирования;*
- *уровни языков программирования;*
- *трансляция и трансляторы;*
- *о двух способах трансляции;*
- *работа компилятора;*
- *работа интерпретатора.*

Системы программирования

«Родным» языком ЭВМ является язык машинных команд (ЯМК). Самые первые ламповые ЭВМ понимали только этот язык. В программах на ЯМК данные обозначаются их адресами в памяти машины, выполняемые операции — числовыми кодами. Программист сам должен заботиться о расположении в памяти ЭВМ команд программы и данных.

Современные программисты так не работают. Для программирования на современных компьютерах применяются ***системы программирования*** (*СП*). В учебнике 8 класса (глава 2) говорилось о том, что программное обеспечение компьютера делится на три части:

- системное ПО;
- прикладное ПО;
- системы программирования.

С первыми двумя видами программного обеспечения вы уже знакомы. Системное ПО — это прежде всего операционные системы, сервисные программы. Прикладное ПО — это многочисленные редакторы, электронные таблицы, информационные системы, математические пакеты, экспертные системы и многое другое, с чем работает абсолютное большинство пользователей.

Системы программирования (СП) предназначены для создания программ управления компьютером.

Системы программирования позволяют разрабатывать и исполнять на компьютере программы, написанные на языках более высокого уровня, чем язык машинных команд.

Уровни языков программирования

Что понимается под уровнем языка? Понятие уровня языка программирования связано со степенью его удаленности от языка процессора компьютера и приближенности к естественному человеческому языку, к формальному языку предметной области (чаще всего — математики). *Чем выше уровень, тем дальше язык от компьютера и ближе к человеку.* Этот принцип схематически отражен на рис. 6.17.

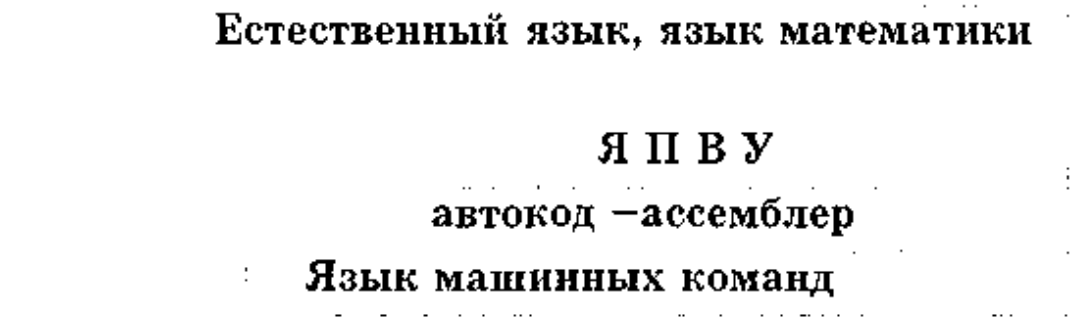


Рис. 6.17. Уровни языков программирования

Язык машинных команд — это язык самого низкого уровня. Первые языки программирования, отличные от ЯМК, появились на машинах первого поколения, и назывались они ***автокодами.***

Автокод — это машинно-ориентированный язык символического программирования.

Одна команда на автокоде соответствует одной машинной команде. Работая на автокоде, программист освобожден от необходимости распределять память под программу и величины; ему не приходится работать с адресами ячеек. Переменные величины и числовые константы обозначаются так же, как в математике, коды операций — мнемоническими (буквенными) обозначениями.

Начиная с машин третьего поколения, языки такого типа стали называть ассемблерами. В наше время на ассемблере программируют довольно редко. Это, как правило, делают системные программисты.

Сокращение ЯПВУ расшифровывается так: языки программирования высокого уровня. Сегодня большинство программистов работают именно на этих языках. Примеры языков высокого уровня: Паскаль, Бейсик, СИ, Фортран.

Вот пример записи одной и той же команды сложения двух чисел на трех языках разного уровня: ЯМК, автокоде и Паскале:

```
C:=A+B          Паскаль
ADD A, B, C     автокод
01 24 28 2C     ЯМК
```

Видно, как с повышением уровня языка повышается «понятность» команды (по-английски слово «ADD» означает «сложить»). Однако, чем понятнее язык для человека, тем непонятнее для процессора компьютера. Процессор понимает только ЯМК, это его «родной» язык. Человеку же легче писать программы на языках более высокого уровня. Как же быть?

Трансляция и трансляторы

Как сделать так, чтобы человек мог писать программы на автокоде или Паскале, а компьютер мог исполнять эти программы? Ответ на поставленный вопрос такой же, как на вопрос «Как мне общаться с японцем, если я не знаю японского языка?». Нужен переводчик! По-английски «переводчик» — «translator».

Программы-переводчики с автокода, Паскаля, Фортрана и других языков на язык машинных команд называются **трансляторами.**

Таким образом, компьютер сам производит перевод под управлением программы-транслятора. Процесс перевода программы на язык машинных команд называется *трансляцией.* Прежде чем выполнить, например, программу на Паскале, ее нужно *оттранслировать.* Трансляцию можно представить как спуск с верхней ступеньки языка на самую первую ступеньку — ЯМК (рис. 6.18).

Транслятор является обязательным элементом любой системы программирования. Первые СП включали в себя только транслятор. Затем к транслятору стали добавляться раз-

Рис. 6.18. Трансляция с автокода и Паскаля на ЯМК

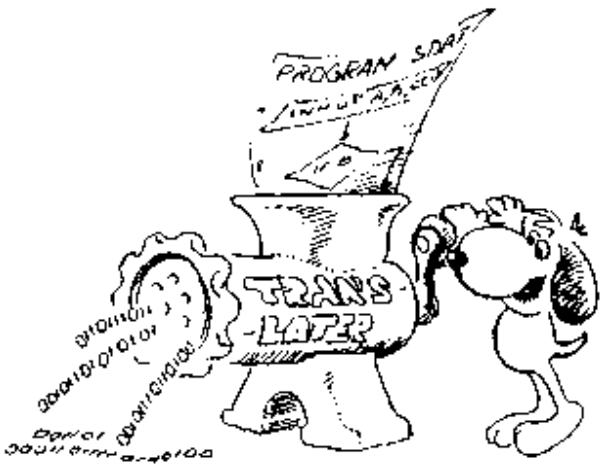
личные сервисные средства: текстовые редакторы, отладчики, системы обслуживания программных библиотек, средства организации дружественного интерфейса с пользователем.

Наиболее удобными для пользователя стали системы программирования, созданные на персональных компьютерах.

Язык программирования, с которым работает СП, называется ее *входным языком*. Системы программирования именуются по названию своего входного языка. Например: «Система Бейсик», «Система Паскаль», «Система Фортран». Иногда в название систем включаются префиксы, обозначающие, например, ее фирменное происхождение. Очень популярны системы с приставкой «Турбо»: Турбо Паскаль, Турбо С и др. Это системы программирования, разработанные фирмой Borland.

О двух способах трансляции

Реализовать тот или иной язык программирования на компьютере — это значит создать транслятор с этого языка для данного компьютера. Существуют два принципиально различных метода трансляции. Они называются ***«компиляция»*** и ***«интерпретация»***.



Для объяснения различия можно предложить такую аналогию: представьте себе, что иностранный лектор должен выступить перед аудиторией на незнакомом для слушателей языке. Требуется перевод, который можно организовать двумя способами:

1) полный предварительный перевод: лектор заранее передает текст выступления переводчику, тот записывает перевод, размножает его и раздает слушателям (после этого лектор может уже и не выступать);

2) синхронный перевод: лектор читает доклад, переводчик одновременно с ним, слово за словом, переводит выступление.

Компиляция является аналогом полного предварительного перевода; интерпретация — аналог синхронного перевода. Транслятор, работающий по принципу компиляции, называется ***компилятором***. Транслятор, работающий методом интерпретации, называется ***интерпретатором***.

Работа компилятора

При компиляции в память компьютера загружается программа-компилятор. Она воспринимает текст программы на ЯПВУ как исходную информацию. Компилятор производит синтаксический контроль программы и при обнаружении ошибок выводит диагностические сообщения. Если ошибок нет, то результатом компиляции является программа на языке машинных команд.

Затем компилятор удаляется из оперативной памяти. В памяти остается только программа на ЯМК, которая выполняется для получения результатов.

На рис. 6.19 схематически показан процесс выполнения программы на ЯПВУ с использованием компиляции. Прямоугольниками изображены программы в машинных кодах, овалами — обрабатываемая и конечная информация.

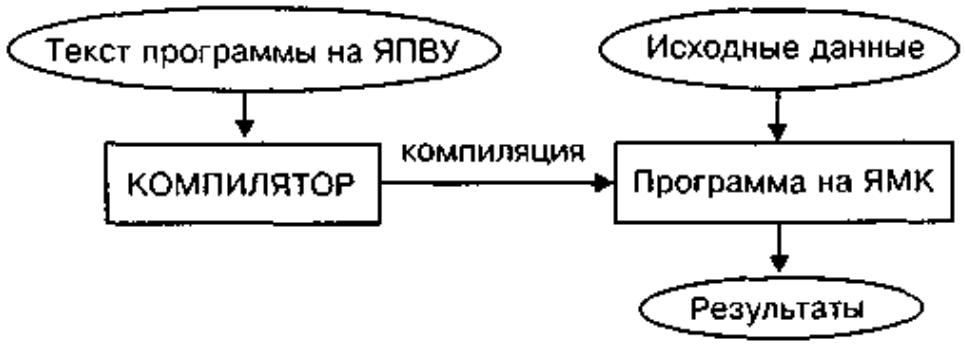


Рис. 6.19. Выполнение программы на ЯПВУ с использованием компилятора

Конечно, компиляция с автокода-ассемблера много проще, чем с языков высокого уровня. Для этой процедуры часто применяют специальный термин — *ассемблирование*. А под словом «ассемблер» понимается не только язык программирования, но и транслятор с него.

Работа интерпретатора

Интерпретатор в течение всего времени работы программы находится во внутренней памяти (иногда для этого используется ПЗУ). В ОЗУ помещается программа на ЯПВУ. Интерпретатор «читает» ее первый оператор, переводит его в машинные команды и тут же организует выполнение этих команд. Затем переходит к переводу и выполнению следующего оператора и так до конца программы. При этом результаты предыдущих переводов в памяти не сохраняются. При повторном выполнении одного и того же оператора в цикле он снова будет транслироваться. Перед трансляцией каждого оператора происходит его синтаксический анализ.

На рис. 6.20 схематически показан процесс выполнения программы на ЯПВУ с использованием интерпретатора.

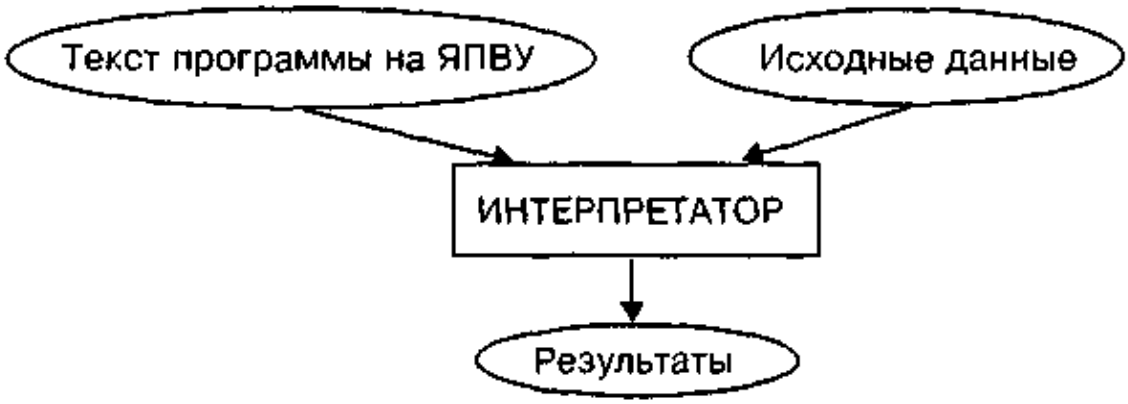


Рис. 6.20. Выполнение программы на ЯПВУ с использованием интерпретатора

Таким образом, при компиляции трансляция и исполнение программы идут последовательно друг за другом. При интерпретации — параллельно.

Один раз откомпилированная программа может быть сохранена во внешней памяти и затем многократно выполнена. На компиляцию машинное время тратиться больше не будет. Программа на интерпретируемом языке при каждом выполнении подвергается повторной трансляции. Кроме того, интерпретатор может занимать значительное место в оперативной памяти.

Из-за указанных причин использование компиляторов удобнее для больших программ, требующих быстрого счета и большого объема памяти. Программы на Паскале, Си, Фортране всегда компилируются. Язык Бейсик часто реализуется через интерпретатор.

Коротко о главном

Для разработки программ управления компьютером программисты используют системы программирования (СП).

Язык программирования, с которым позволяет работать данная СП, называется ее входным языком.

Язык процессора компьютера — это язык машинных команд — ЯМК.

Уровень языка программирования определяется степенью его удаленности от ЯМК (чем дальше, тем выше уровень).

Автокод (ассемблер) — это машинно-ориентированный язык символического программирования.

Наиболее удобным средством программирования являются языки высокого уровня (ЯПВУ). Сегодня с ними работает большинство программистов.

Трансляция — это процесс перевода текста программы на язык машинных команд. Программа-переводчик называется транслятором.

Существуют два способа трансляции: компиляция и интерпретация. При компиляции сначала весь текст программы переводится на ЯМК, затем производится ее исполнение. При интерпретации перевод и исполнение происходят параллельно.

Вопросы и задания

1. Что такое язык программирования?
2. Что обозначает понятие «уровень языка программирования»?
3. К какому уровню относятся языки типа «автокод-ассемблер»?
4. Почему языки программирования высокого уровня называют машинно-независимыми языками?
5. Какие из языков программирования высокого уровня вы знаете?
6. Что такое трансляция? Что такое транслятор?
7. В чем различие между компиляцией и интерпретацией?

Дополнение к главе 7

7.1. История языков программирования

Основные темы параграфа:

- *первые шаги автоматизации программирования;*
- *первые языки высокого уровня: Кобол и Фортран;*
- *языки процедурного программирования;*
- *языки искусственного интеллекта;*
- *современные языки объектно-ориентированного и визуального программирования.*

Первые шаги автоматизации программирования

Программы для первых ЭВМ программисты писали на языках машинных команд. Это очень трудоемкий и длительный процесс. Проходило значительное время между началом составления программы и началом ее использования. Решить эту проблему можно было лишь путем создания средств автоматизации программирования.

Первыми «инструментами», которые экономили труд программистов, стали подпрограммы. В августе 1944 года для релейной машины «Марк-I» под руководством Грейс Хоппер (женщина-программист, морской офицер ВМФ США) была написана первая подпрограмма для вычисления $\sin x$.

Не одну Грейс Хоппер волновала проблема облегчения труда программистов. В 1949 году Джон Моучли (один из создателей ЭВМ ENIAC) разработал систему Short Code, которую можно считать предшественницей языков программирования высокого уровня. Программист записывал решаемую задачу в виде математических формул, преобразовывал формулы в двухбуквенные коды. В дальнейшем специальная программа переводила эти коды в двоичный машинный код. Таким образом, Дж. Моучли разработал один из первых примитивных интерпретаторов. А в 1951 году Г. Хоппер создала первый компилятор А-0. Ею же впервые был введен этот термин.

Первые языки высокого уровня: Кобол и Фортран

В 50-е годы прошлого века группа под руководством Г. Хоппер приступила к разработке нового языка и компилятора B-0. Новый язык позволил бы программировать на языке, близком к обычному английскому. Разработчики языка выбрали около 30 английских слов, для распознавания которых Г. Хоппер придумала способ, сохранившийся в операторах будущих языков программирования: каждое слово содержит неповторимую комбинацию из первой и третьей букв. Благодаря этому компилятор при создании машинного кода программы может игнорировать все остальные буквы в слове.

Необходимость появления такой системы, язык которой приближен к разговорному, Г. Хоппер связывала с тем, что область применения ЭВМ будут расширяться, в связи с чем будет расти и круг пользователей. По словам Г. Хоппер, следует оставить попытки «превратить их всех в математиков».

В 1958 году система B-0 получила название FLOW–MATIC и была ориентирована на обработку коммерческих данных. В 1959 году был разработан язык COBOL (Common Business Oriented Language) (Кобол)— машинно независимый язык программирования высокого уровня для решения задач бизнеса. Одна и та же программа, написанная на машинно независимом языке, может быть выполнена на ЭВМ разных типов, оснащенных соответствующим транслятором с этого языка. Консультантом при создании языка COBOL вновь выступила Г. Хоппер.

В 1954 году публикуется сообщение о создании языка FORTRAN (FORmula TRANslation) (Фортран). Местом рождения языка стала штаб-квартира фирмы IBM в Нью-Йорке. Одним из главных разработчиков является Джон Бэкус. Он же стал автором НФБ (нормальная форма Бэкуса), которая используется для описания синтаксиса многих языков программирования.

Одним из основных достоинств первой версии Фортрана являлось то, что язык давал возможность автоматизировать организацию циклов. Этот язык стал незаменимым для разработки научных и инженерных приложений. Следует отметить долговечность языка Фортран. За полстолетия он сильно изменился. В настоящее время Фортран реализован на персональных компьютерах, супер-компьютерах и по-прежнему широко используется в научных исследованиях.

В тот же период в европейских странах и в СССР популярным становится язык ALGOL. Как и FORTRAN, он ориентировался на математические задачи. В нем была реализована передовая для того времени технология программирования — структурное программирование.

Большое количество новых языков стало появляться в 60-е, 70-е годы прошлого столетия, но не все из них выдержали испытание временем. К языкам-долгожителям следует отнести язык BASIC, разработанный в Дартмутском университете в 1964 году под руководством Джона Кемени и Томаса Курца. По замыслу авторов, это простой язык, легко изучаемый, предназначенный для программирования несложных расчетных задач. Наибольшее распространение BASIC получил на микро-ЭВМ и персональных компьютерах. Однако первоначально этот язык был неструктурным и с этой точки зрения плохо подходил для обучения качественному программированию. В 1985 году была создана версия языка True BASIC, которая по мнению разработчиков была совершеннее, чем PASCAL. В 1991 году появилась первая версия языка VISUAL BASIC.

Языки процедурного программирования

Для первых языков программирования характерной чертой была *предметная ориентация*. Это значит, что каждый язык предназначался для решения какого-то определенного класса задач. COBOL был ориентирован на решение задач бизнеса, FORTRAN — на проведение инженерных и научных расчетов. В эпоху ЭВМ третьего поколения большое распространение получил язык PL/1 (Program Language/1), разработанный фирмой IBM. Это был первый язык, претендовавший на универсальность, т. е. на возможность решать любые задачи: вычислительные, обработки текстов, накопления и поиска информации. PL/1 оказался слишком сложным языком. Транслятор с него недостаточно оптимальный, содержащий ряд невыявленных ошибок. По этой причине этот язык не получил распространения. Однако линия на универсализацию языков была продолжена. Старые языки были модернизированы в универсальные варианты. Примером тому стал FORTRAN 77.

Значительным событием в истории языков программирования стало создание в 1971 году языка PASCAL. Его автором является Никлаус Вирт, профессор из Швейцарии. Вирт назвал этот язык в честь французского математика и физика

Блэза Паскаля, который в 1642 году сконструировал вычислительный механизм. Первоначально PASCAL создавался как язык для обучения. В нем ярко выражена структурная линия программирования. Широкое практическое применение язык получил с появлением персональных компьютеров в версии Turbo PASCAL.

Язык программирования *C* («Си») был задуман как инструментальный язык для разработки операционных систем. Он создавался одновременно с операционной системой UNIX. Авторами этого языка и ОС UNIX являются американские программисты Деннис Ричи и Кеннет Томпсон. Первоначально К. Томпсон начал писать ОС UNIX на языке FORTRAN. В дальнейшем был создан язык С, и в 1973 году ядро операционной системы вместе с программами-утилитами было переписано на С. Этот язык является структурным языком высокого уровня. В настоящее время он применяется для разработки не только операционных систем, но и трансляторов, системных и прикладных программ.

Языки искусственного интеллекта

В 90-х годах прошлого столетия планировалось появление компьютеров пятого поколения, называемых машинами «искусственного интеллекта». В качестве основных языков программирования в этом, пока неосуществленном, проекте предполагались языки искусственного интеллекта LISP и PROLOG.

Создателем языка LISP (1956–1959 гг.) является Джон Маккарти, которого называют отцом искусственного интеллекта. Именно он первым ввел термин «искусственный интеллект». Основным в языке LISP является понятие рекурсивно определенных функций. Доказано, что любой алгоритм может быть описан с помощью некоторого набора рекурсивных функций. Основные идеи этого языка были позже использованы в языке программирования для детей LOGO, разработанном в 70-е годы в Массачусетском технологическом институте под руководством Сэймура Пейперта. Подмножество языка LOGO, включающее команды для Черепашки, применяется при раннем обучении программированию.

Язык PROLOG разработан во Франции в 1972 году также для решения проблем искусственного интеллекта. PROLOG позволяет в формальном виде описывать различные утверждения, логику рассуждений, заставляет компьютер давать ответы на заданные вопросы.

Современные языки объектно-ориентированного и визуального программирования

В последнее время одним из основных направлений в развитии программного обеспечения компьютера стал ***объектно-ориентированный подход***. Под словом «объект» понимается структура, объединяющая в единое целое данные и программы их обработки. Стали популярны объектно-ориентированные операционные системы (например, Windows), прикладные программы, а также ***системы объектно-ориентированного программирования*** (ООП).

Первым языком с элементами ООП был язык Симула-67. В Turbo PASCAL, начиная с версии 5.5, появились средства ООП. Итогом развития Turbo PASCAL в этом направлении стало создание фирмой Borland системы программирования DELPHI (Делфи). Использование этой системы, в частности, дает возможность легко и быстро программировать сложный графический интерфейс. В 1991 году появилась первая версия языка VISUAL BASIC. Начиная с пятой версии (1997 год) язык стал полностью объектно-ориентированным. По данным на конец 90-х годов прошлого столетия количество программистов, использующих для своих разработок VISUAL BASIC, не уступает числу сторонников VISUAL C++ и DELPHI.

В 1985 году лаборатория Bell Labs (США) сообщила о создании языка программирования C++ (СИ++). Этот язык является сегодня наиболее популярным среди языков объектно-ориентированного программирования. С его помощью возможно создание программных приложений, ориентированных на любые машины — от персональных до суперкомпьютеров. Создателем языка является Бьорн Страуструп.

Представителем языков объектно-ориентированного программирования является и язык JAVA, созданный в 1995 году под руководством Джеймса Гослинга группой инженеров компании Sun Microsystems. При его разработке была поставлена цель — создать простой язык, не требующий специального изучения. Язык JAVA был разработан так, чтобы быть максимально похожим на C++. JAVA является идеальным инструментом при создании приложений для Интернета.

Коротко о главном

Первые средства автоматизации программирования: Short Code (1949); компилятор A-0 (1951).

Первыми распространенными полноценными языками высокого уровня были: FORTRAN (Фортран) (1954), ориентированный на математические вычисления, COBOL (Кобол), ориентированный на задачи бизнеса.

Языки, распространившиеся в 60–70-х годах XX века: ALGOL, BASIC, PASCAL, PL/1; C — первый язык высокого уровня, применяемый в системном программировании; языки искусственного интеллекта: PROLOG, LISP.

В 1980–90-е годы XX века были созданы языки объектно-ориентированного программирования: C++, DELPHI, VISUAL BASIC; JAVA — язык Web-программирования.

Глоссарий

Компьютерная сеть	Система компьютеров, связанных каналами передачи информации	Глава 1 § 1
Локальная сеть	Небольшая компьютерная сеть, работающая в пределах одного помещения, одного предприятия	Глава 1 § 1
Локальная сеть одноранговая	Локальная сеть, в которой все объединенные в ней компьютеры равноправны	Глава 1 § 1
Локальная сеть с выделенным сервером	Локальная сеть, в которой имеется одна центральная машина, называемая сервером, и множество подключенных к ней компьютеров, называемых рабочими станциями	Глава 1 § 1
Сервер локальной сети	Компьютер, используемый как хранилище общих информационных ресурсов (данных и программ) и позволяющий подключаться к техническим устройствам общего доступа (принтерам, сканерам и т. д.)	Глава 1 § 1
Глобальная компьютерная сеть (телекоммуникационная сеть)	Сеть, связывающая между собой множество локальных сетей, а также отдельные компьютеры, не входящие в локальные сети. Размеры глобальных сетей не ограничены: существуют сети от региональных до всемирных	Глава 1 § 1
Телекоммуникация	Процесс обмена информацией по глобальной компьютерной сети	Глава 1 § 1
Узлы компьютерной сети	Компьютерное оборудование, осуществляющее связь между отдельными частями глобальной сети; абонент соединяется с сетью через узел определенного провайдера	Глава 1 § 1
Шлюз	Узел в региональной или отраслевой сети, связывающий ее с другими сетями	Глава 1 § 1

Интернет	Мировая система компьютерных сетей	Глава 1 § 1
Электронная почта	Служба обмена письмами в компьютерных сетях	Глава 1 § 2
Почтовый ящик	Поименованный раздел внешней памяти почтового сервера, отведенный для информации абонента	Глава 1 § 2
Электронный адрес	Уникальное имя почтового ящика абонента	Глава 1 § 2
Домены	Части электронного адреса, разделяемые точками, и уточняющие местоположение почтового сервера в сети	Глава 1 § 2
Доменное имя почтового сервера	Вся часть электронного адреса, расположенная справа от значка @	Глава 1 § 2
Электронное письмо	Текстовый файл, содержащий «конверт» с адресом (адресами) получателя (получателей) и текст письма	Глава 1 § 2
Телеконференция	Система обмена информацией на определенную тему между абонентами сети	Глава 1 § 2
Файловые архивы	Электронные хранилища, позволяющие через Интернет пополнять программное обеспечение пользователей персональных компьютеров. Серверы, поддерживающие работу файловых архивов, называются FTP-серверами	Глава 1 § 2
Аналоговая связь	Связь, при которой передача информации производится в форме непрерывного (электрического) сигнала	Глава 1 § 3
Цифровая связь	Связь, в которой любая информация передается в форме двоичного кода	Глава 1 § 3
Шум	Различного рода помехи, приводящие к потере (искажению) информации	Глава 1 § 3
Компьютер-сервер	Высокопроизводительный компьютер, обеспечивающий информационные услуги в сети	Глава 1 § 3

Линии связи в компьютерной сети	Коммутируемые телефонные линии или выделенные каналы — телефонные, кабельные, оптоволоконные, спутниковые и т.д.	Глава 1 § 3
Терминал абонента	ПК, используемый абонентом для получения и передачи информации	Глава 1 § 3
Модем	Электронное устройство, осуществляющее соединение компьютеров в сети через аналоговую телефонную линию. Модуляция — преобразование из цифровой формы в аналоговую, демодуляция — обратное преобразование	Глава 1 § 3
Протоколы работы сети	Стандарты, определяющие формы представления и способы пересылки сообщений, процедуры их интерпретации, правила совместной работы различного оборудования	Глава 1 § 3
Технология «клиент-сервер»	Организация программного обеспечения, принятая в современных сетях	Глава 1 § 3
Клиент-программа	Программа, подготавливающая запрос пользователя, передающая его по сети, а затем принимающая ответ	Глава 1 § 3
Сервер-программа	Программа, принимающая запрос пользователя, подготавливающая ответную информацию и передающая ее пользователю	Глава 1 § 3
Word Wide Web (WWW)	«Всемирная паутина»: распределенная по всему миру информационная система с гиперсвязями, существующая на технической базе мировой сети Интернет	Глава 1 § 4
Web-страница	Основная поименованная информационная единица, представляющая собой отдельный документ, хранящийся на Web-сервере	Глава 1 § 4
Web-сайт	Некоторое количество Web-страниц, связанных тематически. Имеет главную (домашнюю — Home page) страницу	Глава 1 § 4

Web-сервер	Компьютер в сети Интернет, хранящий Web-страницы и соответствующее программное обеспечение	Глава 1 § 4
Гипермедиа	Система гиперсвязей между мультимедиа документами	Глава 1 § 4
Web-браузер	Клиент-программа для работы пользователя с WWW	Глава 1 § 4
Поисковая система	ПО, позволяющее подбирать нужные документы в WWW по тематике или по ключевым словам	Глава 1 § 5
Модель	Упрощенное подобие реального объекта, отражающее свойства (характеристики) объекта, существенные для достижения цели моделирования	Глава 2 § 6
Объект моделирования	Материальные объекты, явления природы, процессы. В процессе моделирования объекты рассматриваются как системы	Глава 2 § 6
Система	Нечто целое, состоящее из множества взаимосвязанных частей	Глава 2 § 6
Материальная (натурная) модель	Объект-заменитель, физически подобный моделируемому объекту	Глава 2 § 6
Информационная модель	Описание объекта моделирования (словесное, математическое, графическое и т. д.)	Глава 2 § 6
Формализация	Результат перехода от реальных свойств моделируемой системы к их формальному обозначению в определенной знаковой системе	Глава 2 § 6
Структура системы	Определенный порядок объединения элементов, составляющих систему	Глава 2 § 7
Виды информационных моделей	Вербальные, графические, табличные, математические, имитационные, объектные	Глава 2 § 7, 8, 9
Численные методы	Методы, сводящие решение любой математической задачи к последовательности арифметических операций (используются в математическом моделировании)	Глава 2 § 9

Компьютерная математическая модель	Программа, реализующая расчеты состояния моделируемой системы по ее математической модели	Глава 2 § 9
Вычислительный эксперимент	Использование компьютерной математической модели для исследования поведения объекта	Глава 2 § 9
Имитационная модель	Воспроизведение на компьютере поведения сложной системы, элементы которой могут вести себя случайным образом (их поведение заранее предсказать нельзя)	Глава 2 § 9
База данных (БД)	Совокупность организованной информации, относящейся к определенной предметной области, предназначенная для длительного хранения во внешней памяти компьютера и постоянного применения	Глава 3 § 10
Информационная система	Совокупность базы данных и всего комплекса аппаратно-программных средств для ее хранения, изменения и поиска информации, для взаимодействия с пользователем	Глава 3 § 10
БД фактографическая	Содержит краткую информацию об объектах некоторой системы в строго фиксированном формате	Глава 3 § 10
БД документальная	Содержит документы самого разного типа: текстовые, графические, звуковые, мультимедийные	Глава 3 § 10
БД распределенная	База данных, разные части которой хранятся на различных компьютерах, объединенных в сеть	Глава 3 § 10
БД централизованная	База данных, хранящихся на одном компьютере	Глава 3 § 10
БД реляционная	База данных с табличной организацией данных (одна или несколько взаимосвязанных прямоугольных таблиц)	Глава 3 § 10
Запись	Строка таблицы реляционной БД	Глава 3 § 10
Поле записи	Именованный столбец таблицы реляционной БД	Глава 3 § 10

Первичный ключ	Одно поле (простой ключ) или совокупность полей записи (составной ключ), значения которых не повторяются у разных записей; идентификатор записи	Глава 3 § 10
Тип поля	Свойство поля, определяющее множество значений, которые может принимать данное поле в различных записях, а также действия, которые можно производить с этими значениями	Глава 3 § 10
Основные типы полей	– числовой; – символьный; – логический; – «дата»	Глава 3 § 10
Система управления базами данных (СУБД)	Программное обеспечение компьютера, предназначенное для работы с БД	Глава 3 § 11
Реляционная СУБД	Система управления реляционной базой данных	Глава 3 § 11
Открытие БД	Команда, с которой начинается работа с готовой БД	Глава 3 § 11
Запрос на выборку	Команда поиска записей в БД, удовлетворяющих некоторому условию. Параметры команды: выводимые поля, условие выбора, параметры сортировки	Глава 3 § 11, 13
Создание БД	Команда, по которой создаются (открываются) файлы для хранения таблиц, сообщается информация о составе полей записи, их типах и форматах	Глава 3 § 12
Формат	Свойство поля, определяющее число позиций, отводимых в таблице для поля. Для числовых полей, кроме того, может указываться количество знаков в дробной части (точность)	Глава 3 § 12
Условие выбора	Логическое выражение простое или составное (сложное)	Глава 3 § 13, 14
Логическое выражение	Выражение, принимающее логическое значение («истина» или «ложь»).	Глава 3 § 13, 14

Простое логическое выражение	Содержит одну величину логического типа или операцию отношения (сравнения)	Глава 3 § 13
Операции отношения (сравнения)	= (равно); <> (не равно); > (больше); < (меньше); >= (больше или равно); <= (меньше или равно)	Глава 3 § 13
Сложные логические выражения	Логические выражения, содержащие логические операции	Глава 3 § 14
Логические операции (основные)	– отрицание (НЕ); логическое умножение — – конъюнкция (И); логическое сложение — – дизъюнкция (ИЛИ)	Глава 3 § 14
Отрицание (НЕ)	Изменяет значение логической величины на противоположное («истина» на «ложь», а «ложь» на «истина»)	Глава 3 § 14
Конъюнкция (И)	Результат операции «истина» только тогда, когда оба операнда имеют значение «истина»	Глава 3 § 14
Дизъюнкция (ИЛИ)	Результат операции «ложь» только тогда, когда оба операнда имеют значение «ложь»	Глава 3 § 14
Старшинство логических операций	По убыванию старшинства: операции в скобках; отрицание (НЕ); конъюнкция (И); дизъюнкция (ИЛИ)	Глава 3 § 14
Сортировка БД	Упорядочение записей в таблице по возрастанию или убыванию значения какого-нибудь поля (или полей)	Глава 3 § 15
Ключ сортировки	Поле (поля), по значению которого (которых) производится сортировка	Глава 3 § 15
Целый тип	Тип представления целых чисел в памяти компьютера	Глава 4 § 17
Вещественный тип	Тип представления чисел, имеющих дробную часть, в памяти компьютера	Глава 4 § 17

Диапазон значений	Область изменения значений чисел (целых или вещественных), которые можно хранить в памяти компьютера. Всегда ограничен	Глава 4 § 17
Переполнение	Выход результатов вычислений за границы допустимого диапазона	Глава 4 § 17
Внутреннее представление чисел	Способ записи чисел в памяти компьютера в двоичной системе счисления	Глава 4 § 17
Представление вещественных чисел	$X = m \cdot p^n$, где m — мантисса числа; n — порядок числа; p — основание системы счисления, в которой представлено число	Глава 4 § 17
Погрешность вычислений	Ошибка машинных вычислений с вещественными числами, связанная с ограниченностью разрядности мантиссы	Глава 4 § 17
Электронная таблица (ЭТ)	Данные, представленные в табличном виде и предназначенные для организации табличных расчетов на компьютере	Глава 4 § 18
Табличный процессор (ТП)	Прикладная программа, работающая с электронными таблицами	Глава 4 § 18
Ячейка ЭТ	Наименьшая структурная единица электронной таблицы	Глава 4 § 18
Режимы отображения ЭТ	Режим отображения значений (основной); режим отображения формул	Глава 4 § 18
Имя (адрес) ячейки ЭТ	Складывается из буквенного обозначения столбца и номера строки	Глава 4 § 18
Содержимое ячейки ЭТ	– текст (последовательность символов); – числовое значение (целое или вещественное число); – формула	Глава 4 § 18
Текст	Любая последовательность символов, которая не может быть числом или формулой, а также начинающаяся с апострофа	Глава 4 § 19

Формула	Запись, определяющая порядок вычислений. Включает числа, имена ячеек, знаки операций, обращения к функциям, круглые скобки	Глава 4 § 19
Диапазон (блок, фрагмент) ЭТ	Прямоугольная часть таблицы, обычно обозначаемая именами верхней левой и нижней правой ячеек, разделенными двоеточием	Глава 4 § 20
Функции обработки диапазона	– суммирование чисел, входящих в диапазон; – нахождение минимального (или максимального) значения; – среднее значение и др.	Глава 4 § 20
Операции манипулирования диапазонами ЭТ	– удаление; – вставка; – копирование; – перенос; – сортировка и др.	Глава 4 § 20
Принцип относительной адресации	Адреса ячеек, используемые в формулах, определены не абсолютно, а относительно ячейки, в которой располагается формула	Глава 4 § 20
Условная функция	ЕСЛИ(<условие>;<выражение1>; <выражение2>), где <условие> — логическое выражение. Если значение этого выражение «истина», то значение ячейки определяет <выражение1>, если «ложь» — <выражение2>	Глава 4 § 21
Деловая графика в ЭТ	Построение диаграмм и графиков по данным ЭТ	Глава 4 § 21
Абсолютная адресация	Способ адресации ячеек ЭТ, при котором адрес «замораживается» и на него не распространяется принцип относительной адресации	Глава 4 § 22
Логические функции (И, ИЛИ, НЕ) в ЭТ	Способ реализации логических операций в ЭТ. Имя операции (<логическое выражение1>;<логическое выражение2>).	Глава 4 § 22
Кибернетика	Наука об общих свойствах управления в живых и неживых системах	Глава 5 § 25

Управление	Целенаправленное воздействие одних объектов, которые являются управляющими, на другие объекты – управляемые	Глава 5 § 25
Модель управления в кибернетике	Информационные процесс, протекающий между управляющим объектом и объектом управления путем обмена информацией по каналам (линиям) прямой и обратной связи	Глава 5 § 25
Алгоритм управления	Последовательность команд управления, приводящая к заранее поставленной цели. Информационная составляющая системы управления	Глава 5 § 25
Исполнитель алгоритма управления	Объект управления	Глава 5 § 25
Прямая связь	Процесс передачи команд управления от управляющего объекта к объекту управления по каналу прямой связи	Глава 5 § 26
Обратная связь	Процесс передачи информации о состоянии объекта управления управляющему объекту по каналу обратной связи	Глава 5 § 26
Структура алгоритма управления	В системах без обратной связи может быть только линейной. В системах с обратной связью может быть циклической и ветвящейся	Глава 5 § 26
Программное управление	Управление в автоматических системах, в которых функцию управляющего объекта выполняет компьютер	Глава 5 § 26
Алгоритм (определение)	Понятное и точное предписание исполнителю выполнить конечную последовательность команд, приводящую от исходных данных к искомому результату	Глава 5 § 27
Система команд исполнителя (СКИ)	Перечень команд, которые может выполнить конкретный исполнитель алгоритма	Глава 5 § 27
Дискретность алгоритма	Свойство алгоритма, в соответствии с которым процесс решения задачи должен быть разбит на последовательность отдельно выполняемых шагов	Глава 5 § 27

Понятность алгоритма	Свойство алгоритма, в соответствии с которым алгоритм, составленный для конкретного исполнителя, должен включать только те команды, которые входят в систему команд исполнителя	Глава 5 § 27
Точность алгоритма	Свойство алгоритма, в соответствии с которым каждая команда алгоритма должна определять однозначное действие исполнителя	Глава 5 § 27
Конечность (или результативность) алгоритма	Свойство алгоритма, в соответствии с которым исполнение алгоритма должно завершиться (привести к результату) за конечное число шагов	Глава 5 § 27
Программа	Алгоритм, представленный на языке исполнителя	Глава 5 § 27
Среда исполнителя	Обстановка, в которой действует исполнитель	Глава 5 § 28
ГРИС	Учебный графический исполнитель, назначение которого — получение чертежей, рисунков на экране дисплея	Глава 5 § 28
Алгоритмический язык (АЯ) (учебный)	Вербальный способ описания алгоритмов с русскими служебными словами	Глава 5 § 28
Вспомогательный алгоритм	Алгоритм, по которому решается некоторая подзадача из основной задачи и который, как правило, выполняется многократно	Глава 5 § 29
Подпрограмма (процедура)	Вспомогательный алгоритм в языках программирования	Глава 5 § 29
Последовательная (пошаговая) детализация алгоритма	Метод программирования, при котором сначала записывается основной алгоритм, а затем описываются используемые в нем вспомогательные алгоритмы	Глава 5 § 29
Команда цикла (повторение)	Команда многократного выполнения серии команд по некоторому условию	Глава 5 § 30

Зацикливание	Ситуация, при которой выполнение цикла никогда не заканчивается	Глава 5 § 30
Блок-схема	Графический способ описания алгоритма. Блоки обозначают указания на действия исполнителя, а соединяющие их стрелки указывают на последовательность выполнения действий	Глава 5 § 30
Команда ветвления (развилка)	Выбор по условию одного из двух вариантов продолжения выполнения алгоритма с последующим выходом на общее продолжение	Глава 5 § 31
Программирование	1. Раздел информатики, занимающийся вопросами разработки программ управления компьютером. 2. Процесс разработки программы для компьютера	Глава 6 § 32
Язык программирования	Фиксированная система обозначений для описания алгоритмов и структур данных	Глава 6 § 32
Система программирования	Программное обеспечение компьютера, предназначенное для разработки, отладки и исполнения программ на определенном языке программирования	Глава 6 § 32
Системные программисты	Занимаются разработкой системного программного обеспечения	Глава 6 § 32
Прикладные программисты	Занимаются разработкой прикладного программного обеспечения как общего, так и специального назначения	Глава 6 § 32
Величина	Отдельный информационный объект, имеющий имя, тип и значение, занимающий определенное место в памяти компьютера (ячейку памяти)	Глава 6 § 33
Константа	Постоянная величина, значение которой не может изменяться при выполнении программы	Глава 6 § 33
Переменная	Величина, обозначаемая символическим именем (идентификатором), значение которой может меняться в ходе исполнения программы	Глава 6 § 33

Тип величины	Свойство, определяющее множество значений, допустимые действия и форму внутреннего представления величины. Основные типы: целый, вещественный, символьный, логический	Глава 6 § 33
Ввод данных	Занесение данных с внешних устройств в оперативную память компьютера для их последующей обработки	Глава 6 § 33
Вывод данных	Передача данных из оперативной памяти на внешние устройства вывода (дисплей, принтер и т. д.)	Глава 6 § 33
Команда присваивания	<переменная>:=<выражение> Сначала вычисляется выражение, затем полученное значение присваивается переменной	Глава 6 § 33
Свойства присваивания	– значение переменной не определено, если ей не присвоено никакого значения; – новое значение, присваиваемое переменной, заменяет ее старое значение; – присвоенное переменной значение сохраняется в ней вплоть до нового присваивания	Глава 6 § 33
Паскаль	Универсальный язык программирования, позволяющий решать самые разнообразные задачи обработки информации	Глава 6 § 35
Оператор	Команда, записанная на языке программирования	Глава 6 § 35
Сценарий работы программы	Описание взаимодействия программы с пользователем (пользовательский интерфейс) в процессе ее выполнения	Глава 6 § 38
Этапы решения задачи путем программирования	1) постановка задачи; 2) формализация (математическая); 3) построение алгоритма; 4) составление программы на языке программирования; 5) отладка и тестирование программы; 6) проведение расчетов и анализ полученных результатов	Глава 6 § 39

Тест	Конкретный вариант значений исходных данных, для которого известен ожидаемый результат	Глава 6 § 39
Тестирование	Испытание работоспособности программы на серии тестов с целью обнаружения скрытых ошибок	Глава 6 § 39
Алгоритм Евклида	Алгоритм вычисления наибольшего общего делителя двух натуральных чисел. Имеет структуру цикла с вложенным ветвлением	Глава 6 § 40
Массив	Представление в языках программирования таблично организованных данных. Пронумерованная конечная последовательность однотипных величин	Глава 6 § 41, 42
Случайные числа	Числа, получающиеся в результате случайного выбора из конечного множества значений (игровой кубик, жребий, лотерея и т. п.).	Глава 6 § 43
Датчик случайных чисел	Программа получения случайных чисел	Глава 6 § 43
Счетчик	Переменная целого типа, в которой подсчитывается количество искомых значений (число выполнений некоторого события)	Глава 6 § 43
Печатный станок	Первое средство массового тиражирования книг. Изобрел Иоганн Гутенберг в середине XV века	Глава 7 § 44
Фонограф	Первое устройство звукозаписи. Изобрел Томас Эдисон в 1877 году	Глава 7 § 44
Электрический телеграф	Первое средство быстрой передачи информации на большие расстояния. Изобретатели: П. Л. Шеллинг (1832), С. Морзе (1837).	Глава 7 § 44
Азбука Морзе	Телеграфный код: язык кодирования телеграфных сообщений	Глава 7 § 44
Телефон	Первое средство передачи звука на расстояние. Изобрел А. Белл в 1876 году	Глава 7 § 44

Машина Паскаля	Первая механическая счетная машина. Изобрел Блез Паскаль в 1645 году	Глава 7 § 44
Аналитическая машина Бэббиджа	Первый проект программно управляемого вычислительного автомата. Разработал Чарльз Бэббидж в середине XIX века	Глава 7 § 44
Ада Лавлейс	Первый программист. Составляла программы для машины Бэббиджа	Глава 7 § 44
Система счисления	Способ изображения чисел и соответствующие ему правила действия над числами	Глава 7 § 45
Непозиционная система счисления	Система счисления, в которой количественное значение, обозначаемое цифрой, не зависит от позиции цифры в записи числа	Глава 7 § 45
Позиционная система счисления	Система счисления, в которой количественное значение, обозначаемая цифрой, зависит от позиции цифры в записи числа	Глава 7 § 45
Основание позиционной системы счисления	Равно количеству используемых в системе цифр (мощность алфавита системы счисления)	Глава 7 § 45
Арабские числа	Десятичная позиционная система счисления. Зародилась в Индии в V веке н. э.	Глава 7 § 45
Системы счисления, используемые для представления компьютерной информации	Двоичная, восьмеричная, шестнадцатеричная	Глава 7 § 45
Первая в мире ЭВМ	ENIAC. Создана в США в 1945 году	Глава 7 § 46
Первое поколение ЭВМ	Ламповые машины. Возникли в 50-х годах XX века	Глава 7 § 46
Второе поколение ЭВМ	Транзисторные машины. Возникли в 60-х годах XX века	Глава 7 § 46

Третье поколение ЭВМ	Машины на интегральных схемах. Возникли в 70-х годах XX века	Глава 7 § 46
Четвертое поколение ЭВМ	Компьютеры на микропроцессорах (микроЭВМ, персональные компьютеры). Многопроцессорные суперкомпьютеры. Возникли в 70-х, 80-х годах XX века	Глава 7 § 46
Персональный компьютер (ПК)	МикроЭВМ с дружественным к пользователю аппаратным и программным обеспечением. Первый ПК — Apple-1, 1976 г. Создатели: С. Джобс, С. Возняк	Глава 7 § 46
Кластерные системы	Сеть ПК, работающая как многопроцессорный вычислительный комплекс (альтернатива суперкомпьютеру). Зарождаются в 90-х годах XX века	Глава 7 § 46
Библиотеки стандартных программ	Первый вид программного обеспечения ЭВМ. Возникли на ЭВМ первого поколения.	Глава 7 § 47
Транслятор	Программа-переводчик с языка программирования на язык машинных кодов. Включаются в ПО ЭВМ второго поколения	Глава 7 § 47
Системы программирования	Развиваются на ЭВМ третьего поколения. Инструмент работы программиста. Современные СП включают: транслятор, текстовый редактор, библиотеки подпрограмм, отладчики и пр.	Глава 7 § 47
Системное ПО	Зарождаются на ЭВМ второго поколения. Основа ПО персонального компьютера. Включает в себя операционную систему и сервисные программы	Глава 7 § 47
Прикладное ПО	Основа программного обеспечения информационных технологий	Глава 7 § 47
Информационная технология	Совокупность массовых способов и приемов накопления, передачи и обработки информации с использованием современных технических и программных средств	Глава 7 § 47

Электронный офис	Возникает и развивается в 90-х годах XX века. Пример: Microsoft Office. Технология обработки деловой информации на базе интегрированных пакетов прикладных программ	Глава 7 § 47
Автоматизированные системы управления (АСУ)	Системы принятия управленческих решений на базе ИКТ	Глава 7 § 47
Системы автоматического проектирования (САПР)	Компьютерные технологии создания чертежей, осуществления экономических и технических расчетов, работы с конструкторской документацией	Глава 7 § 47
Геоинформационные системы (ГИС)	Технологии хранения, представления и обработки данных, привязанных к географической карте местности (района, города, страны)	Глава 7 § 47
ИКТ в образовании	Распространенные средства: электронные учебники; учебные ресурсы Интернета (образовательные порталы); дистанционное образование	Глава 7 § 47
Информационные ресурсы	Знания, идеи человечества и указания по их реализации, зафиксированные в любой форме, на любом носителе информации	Глава 7 § 48
Национальные информационные ресурсы	Фонды библиотек и архивов, центры научно-технической информации, отраслевые информационные ресурсы, информационные ресурсы социальной сферы, в том числе сферы образования	Глава 7 § 48
Информационное общество	Стадия развития общества, на которой основным предметом трудовой деятельности людей становится информация	Глава 7 § 49
Информационная безопасность	Гарантия защиты действующих систем хранения, передачи и обработки информации от компьютерных (информационных) преступлений	Глава 7 § 49

Информационные преступления	Основные формы: несанкционированный (неправомерный) доступ к информации, нарушение работоспособности компьютерной системы, нарушение целостности компьютерной информации	Глава 7 § 49
Защищенная система	Информационная система, обеспечивающая безопасность обрабатываемой информации и поддерживающая свою работоспособность в условиях воздействия на нее заданного множества угроз	Глава 7 § 49
Защита от информационных преступлений	Основные меры: технические и аппаратно-программные, административные, юридические	Глава 7 § 49

Учебное издание

Семакин Игорь Геннадьевич
Залогова Любовь Алексеевна
Русаков Сергей Владимирович
Шестакова Лидия Валентиновна

Информатика и информационно-коммуникационные технологии
Базовый курс: Учебник для 9 класса

Ведущий редактор *О. Полежаева*
Художник *Ф. Инфантэ*
Художественный редактор *О. Лапко*
Корректор *Е. Клитина*
Компьютерная верстка *В. Носенко*

Подписано в печать 12.04.05. Формат 60x90 1/16.
Бумага офсетная. Гарнитура Школьная. Печать офсетная.
Усл. печ. л. 23,5. Тираж 50000 экз. Заказ 3377

Издательство «БИНОМ. Лаборатория знаний»
Адрес для переписки: Москва, 119071, а/я 32
Телефон: (095)955-0398, e-mail: LBZ@aha.ru
http://www.LBZ.ru

Отпечатано с готовых диапозитивов
в полиграфической фирме «Полиграфист».
160001, г. Вологда, ул. Челюскинцев, 3.

информатика

ГОТОВЯТСЯ К ИЗДАНИЮ:

И. Г. Семакин, Е. К. Хеннер. Информационные системы и модели. Учебное пособие

Данное учебное пособие является частью УМК для старших классов наряду с практикумом и методическим пособием. Этот УМК реализует элективный курс «Информационные системы и модели».

Задача курса — научить создавать информационные системы, конструировать и исследовать информационные модели.

В первой главе учебного пособия в качестве информационной модели предметной области рассматривается база данных.

Во второй главе изучается математическое моделирование в его компьютерной реализации при максимальном использовании межпредметных связей и универсальной методологии моделирования.

Описанные в учебнике задачи решаются как с помощью специальных программных средств, не требующих от пользователя глубоких знаний сущности используемых методов, так и с помощью приложений, которые учащимся предлагается создавать самостоятельно, используя язык Visual Basic for Applications.

Для учащихся старших классов информационно-технологического и физико-математического профилей.

И. Г. Семакин, Е. К. Хеннер. Информационные системы и модели. Практикум

Данный практикум является частью УМК для старших классов наряду с учебным пособием и методическим пособием. Этот УМК реализует элективный курс «Информационные системы и модели».

Задача курса — научить создавать информационные системы, конструировать и исследовать информационные модели.

Практикум содержит контрольные вопросы, темы для рефератов, лабораторные работы, тесты по разделам курса.

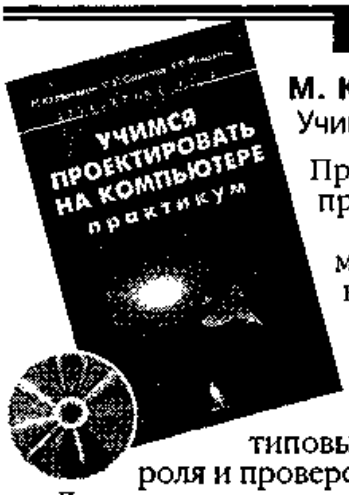
Для учащихся старших классов информационно-технологического и физико-математического профилей.

ИЗДАТЕЛЬСТВО
«БИНОМ.
Лаборатория знаний»

119071, Москва, а/я 32
Тел./факс (095) 955-0421
955-0398
955-0429
E-mail: Lbz@aha.ru
http://www.Lbz.ru

информатика

ИМЕЮТСЯ В ПРОДАЖЕ:



М. Ю. Монахов, С. Л. Солодов, Г. Е. Монахова. Учимся проектировать на компьютере. Практикум

Практикум реализует элективный курс «Учимся проектировать на компьютере».

Практикум позволяет освоить основы современных компьютерных технологий проектирования и дизайна. Рассмотрены компьютерные системы проектирования AutoCAD и 3D Studio MAX. Главы практикума представляют собой законченные учебные модули, каждый из которых включает краткую теорию по теме, типовые практические работы, вопросы для самоконтроля и проверочные задания.

Для учащихся старших классов естественно-научного, физико-математического, технологического профилей и универсального обучения.

М. Ю. Монахов, А. А. Воронин. Создаем школьный сайт в Интернете. Учебное пособие

Учебное пособие реализует элективный курс «Создаем школьный сайт в Интернете».

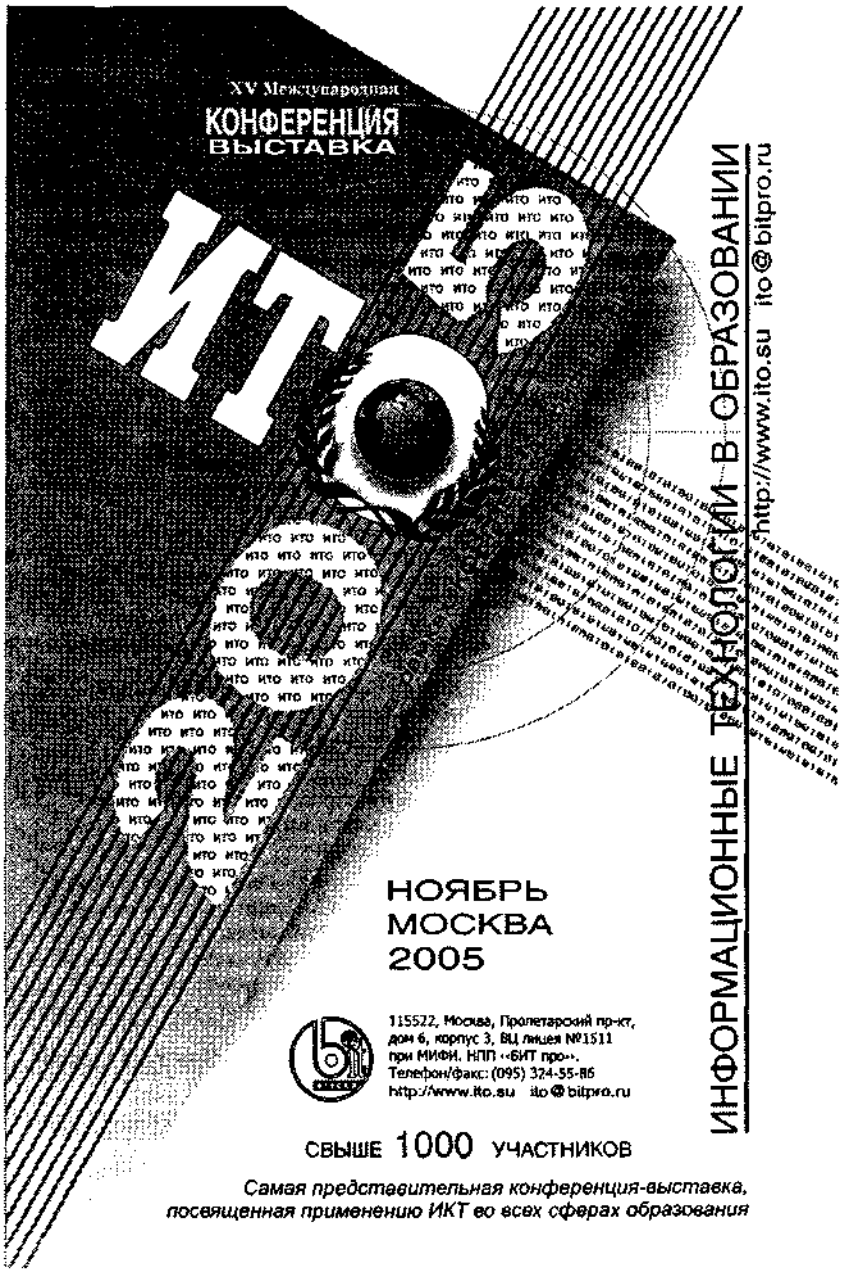
Учебное пособие позволяет получить профессиональные навыки создания сайтов в Интернете. Оно поможет сформировать у обучаемых творческий подход, способность к самостоятельному и инициативному решению проблем, умение использовать типовые инструментально-технологические средства и эффективно работать в неоднородных командах, что требуется для личностного развития и профессионального самоопределения.

Каждая тема учебного пособия представляет собой законченный учебный модуль, включающий теоретический материал, задания для самостоятельной работы, темы рефератов.

Для учащихся старших классов информационно-технологического, физико-математического, естественно-научного и гуманитарного профилей.

ИЗДАТЕЛЬСТВО
«БИНОМ.
Лаборатория знаний»

119071, Москва, а/я 32
Тел./факс (095) 955-0421
955-0398
955-0429
E-mail: Lbz@aha.ru
http://www.Lbz.ru


XV Международная
КОНФЕРЕНЦИЯ
ВЫСТАВКА
ИТО
ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ОБРАЗОВАНИИ
http://www.ito.su ito@bitpro.ru
НОЯБРЬ
МОСКВА
2005
115522, Москва, Пролетарский пр-кт,
дом 6, корпус 3, ВЦ лицея №1511
при МИФИ. НПП «БИТ про».
Телефон/факс: (095) 324-55-86
http://www.ito.su ito@bitpro.ru
СВЫШЕ 1000 УЧАСТНИКОВ
Самая представительная конференция-выставка,
посвященная применению ИКТ во всех сферах образования

информатика

ПОЛНЫЙ КУРС

УЧЕБНО-МЕТОДИЧЕСКИЕ КОМПЛЕКТЫ

для 2–4, 5–6, 7–9, 10–11 классов

ЭЛЕКТИВНЫЕ КУРСЫ

- Исследование информационных моделей
- Математические основы информатики
- Компьютерная графика
- Информационные системы и модели
- Учимся проектировать на компьютере
- Создаем школьный сайт в Интернете

ДЛЯ «ПРОФИ»

- Уроки Web-мастера
- Основы программирования
- Программирование в алгоритмах
- Практикум по объектно-ориентированному программированию
- Основы программирования в интегрированной среде Delphi. Практикум
- Сбои компьютера: диагностика, профилактика, лечение
- Устройство и настройка ПК
- Секреты компьютерного шпионажа

ISBN 5-94774-230-6

9785947742305