А.Г. Гейн А.И. Сенокосов Н.А. Юнерман

ИНФОРМАТИКА

Учебник для 10—11 классов
общеобразовательных учреждений

Допущено Министерством образования и науки Российской Федерации

6-е издание

Москва
«Просвещение»
2005

УДК 373.167.1:004
ББК 32.81я72
Г29

Гейн А. Г.

Г29 Информатика. учеб. для 10—11 кл. общеобразоват. учрежде-ний / А. Г. Гейн, А. И. Сенокосов, Н. А. Юнерман. — 6-е изд. — М.: Просвещение, 2005. — 288 с.: ил. — ISBN 5-09-014401-X.

УДК 373.167.1:004
ББК 32.81я72

ISBN 5-09-014401-X

Предисловие

Мир, окружающий каждого из нас, удивительно разнообразен. Природа и творения человеческих рук, куда ни взгляни, предъявляют нам вещественность объектов и энергетику процессов. Эти два понятия — «вещество» и «энергия» — уже несколько столетий освоены человечеством и в различных своих проявлениях служат объектом изучения таких наук, как физика, химия, биология, геология, и вообще любых естественных наук.

Но XX век выдвинул на передний край еще одно понятие — понятие информации. Наука, изучающая свойства информации, закономерности информационных процессов, называется **информатикой.** А теперь информатика появилась у вас и как учебная дисциплина. Вы будете ее изучать, а наш учебник, как мы надеемся, поможет вам в этом. В нем вы найдете теоретический материал и описания лабораторных работ, которые будете выполнять в компьютерном классе, разнообразные вопросы и задания, которые помогут оценить, насколько хорошо вы усвоили то, что изучали. Некоторые задания будут для вас простыми, другие окажутся посложнее. Самые трудные (разумеется, на наш взгляд) задания помечены звездочкой.

В нашей книге, как и в любом учебнике, вам встретятся новые **термины**. Для удобства они напечатаны жирным шрифтом. *Определения, свойства* и *правила* выделены курсивом. Самые важные выделены особо — заключены в рамочку. Заучивать свойства и правила наизусть совсем не обязательно, но важно понимать их смысл и уметь применять на практике.

Для выполнения лабораторных работ вам нужен будет не только компьютер, но и комплекс программных средств — так сказать компьютерная поддержка нашей книги. Их роль при изучении информатики можно сравнить с ролью циркуля и линейки при изучении геометрии или физических приборов при изучении физики.

Описания этих программных средств приведены в учебнике. Но в вашем классе могут использоваться программы, отличающиеся от описанных в учебнике. Тогда учитель объяснит вам, как выполнять лабораторные работы в этом случае.

Временами тон нашего рассказа вам покажется шутливым. Мы считаем, что и о серьезных вещах можно говорить в шутливой форме. Ведь шутка нередко позволяет увидеть неожиданные стороны того, что, казалось бы, уже хорошо знакомо. Надеемся, что читать наш учебник будет не только полезно, но и приятно.

А теперь — в путь! В мало знакомую вам пока еще страну Информатику.

1 Введение в информатику

Об информатике вы, наверно, уже слышали и теперь с любопытством ждете, каким этот предмет предстанет перед вами. На первый взгляд информатика — это кабинет с мерцающими экранами и щелкающими клавиатурами, жужжащими принтерами и шуршащими дисководами.

На самом деле информатика — это дверь в мир технологий, без которых немыслима современная цивилизация. Достаточно просто оглянуться вокруг. На столике стоит электронный будильник с заложенным в нем расписанием на неделю вперед, в прихожей — «умный» телефон с определителем номера, записной книжкой, системой автодозвона и с чем-то там еще, на телевизоре — видеомагнитофон, который сам знает, когда ему включиться, рядом — игровая приставка... И чудесные свойства этих вещей, а также многих-многих других заключены во внешне невзрачной коробочке, содержимое которой специалисты-электроники называют малопонятным словом «микропроцессор».

Именно микропроцессоры составляют основу компьютеров, управляющих атомными станциями и космическими кораблями, компьютеров, перерабатывающих необозримые объемы самой разнообразной информации, компьютеров, без которых сейчас не обходится проектирование и выпуск даже зубных щеток.

Словом, человек, не знакомый с компьютером, может оказаться совершенно неприспособленным к современной жизни. Почему же тогда изучаемый предмет не называется, скажем, «компьютерика» или «компьютерное дело»? Откуда взялось слово «информатика»? С ответов на эти вопросы мы и начнем наш рассказ.

§1. Информация

Представьте себе, что вы ничего не видите, ничего не слышите, не ощущаете ни запахов, ни холода, ни тепла, а вдобавок вся пища абсолютно безвкусная... Надо полагать, жизнь сразу бы потеряла

для вас свою привлекательность. Ни телевизора, ни компьютерных игр, ни музыки, ни чудесных цветочных ароматов... И все потому, что нас лишили постоянного притока **информации**, без которого немыслимо нормальное существование человека в окружающем мире.

Более того, можно смело сказать, что человек отличается от животного характером информации, которую он способен воспринимать и обрабатывать. Вот только один пример. Около сотни детей по разным причинам выросли в окружении зверей, лишенные всего лишь обычной человеческой речи — одного из главных источников информации. Результат печален: в отличие от легендарного Маугли из них получились существа, почти не отличающиеся от животных.

Попробуем поэтому разобраться подробнее с тем, что такое информация.

Информация — это очень странный объект. Вы можете посмотреть один и тот же фильм с видеокассеты, с видеодиска, с компьютерного CD-диска, просто в кинотеатре в конце концов... Информация, полученная вами при просмотре фильма, будет одна и та же, но находится она на разных **носителях**. И в принципе нам все равно, на каком носителе находится информация — лишь бы было качественное ее воспроизведение (в данном случае нужного фильма).

Из курса физики вы знаете, что все предметы вокруг нас характеризуются наличием в них вещества и энергии. Но, кроме всего прочего, любой предмет несет в себе еще и информацию. Даже самый маленький камешек, валяющийся на дороге, можно бесконечно изучать и изучать, получая информацию о его химическом составе, физических свойствах породы, особенностях его образования в земной коре и т. п.

И все же, говоря об информации, мы в первую очередь имеем в виду те сведения, которые предоставляют нам телевидение, радио, газеты, книги. Информацию несут нам карты местности и картины художников. Информация аккумулирована в тысячах научных трудов, составляющих основу нашей цивилизации. Информация составляет основу ваших школьных учебников, без которых трудно стать образованным человеком.

Оторвитесь на минуту от нашего учебника и выгляните в окно. Вы увидите дома и людей, солнце и небо, деревья и траву и автоматически переключитесь на анализ информации совсем другого сорта. Она очень сильно отличается от однообразных черных строчек с буквами, какие вы видите в книге.

Информация, получаемая из сообщений, записанных с помощью букв и цифр, называется **символьной**, а информация, получаемая с помощью зрительных образов окружающего мира, называется **видеоинформацией**. И если раньше человек имел дело в основном с видеоинформацией, то современная цивилизация окружила нас без-

донными морями символьной информации, способными поглотить любого в своих пучинах.

Человек создан природой как мощный «аппарат» по переработке видеоинформации. Вспомните хотя бы пословицу «Лучше один раз увидеть, чем сто раз услышать», ярко отражающую отличие в восприятии текстовой (т. е. символьной) информации от видеоинформации. В современных потоках символьной информации человек чувствует себя как щепка в водопаде. Чтобы покорить океаны символьной информации, нужны корабли, способные преодолеть океанские просторы. Ими и стали **компьютеры**, или, как их называли раньше, **электронно-вычислительные машины (ЭВМ)**. Но кроме кораблей, нужна наука об управлении этими кораблями, нужны люди, способные прокладывать курс корабля, и люди, знающие устройство этих кораблей. А прежде всего нужны капитаны.

Итак, информатика — это наука обо всех видах информации и инструментах ее обработки. Иными словами,

Информатика изучает процессы получения, хранения, переработки, передачи информации и разрабатывает технологии этих процессов.

И человек, и компьютер вместе с дополнительным оборудованием (профессионалы бы сказали: **периферийными устройствами**) — основные инструменты информационных технологий. Обратите внимание, что человек — это тоже мощнейший инструмент по переработке информации. Роль устройств, посредством которых он получает информацию, выполняют его органы чувств, сохраняет и перерабатывает информацию мозг, а передача осуществляется самыми разными способами, из которых важнейшие — устная и письменная речь.

ВОПРОСЫ И ЗАДАНИЯ

1. Что изучает информатика?
2. О каких двух основных видах представления информации рассказано в этом параграфе?
3. Какие способы передачи информации от человека к человеку вам известны?
4. Человек изобрел много различных способов хранения информации. Какие из них вы могли бы назвать?
5. Приведите примеры, когда, на ваш взгляд, компьютер не может заменить человека в деле обработки информации.

§ 2. Компьютер

При слове «компьютер» вам, конечно, представляется цветной экран монитора, красивый корпус, клавиатура... Но ведь есть еще и бортовые компьютеры автомобилей. А в начале главы, рассказывая о новых компьютерных технологиях, мы упомянули и говорящий будильник, и видеомагнитофон, и телефон с определителем номера... Что объединяет все эти вещи? В них во всех есть две наиглавнейшие части, без которых компьютер просто не существует. Это **центральный процессор** и **память**.

Процессор и память — это главные компоненты компьютера.

Чисто внешне процессор — это маленькая металлокерамическая плоская коробочка размером в 2–3 почтовые марки. Именно он руководит работой всех частей ЭВМ. Конечно, «главнокомандующий» процессор при этом сам исправно исполняет команды, отдаваемые человеком.

Но ЭВМ была бы бесполезна, если бы она не могла запоминать информацию, необходимую для решения задачи. Для хранения информации и предназначено особое устройство — память. Память может быть разная. В некоторых компьютерах есть только **постоянное запоминающее устройство** (или **ПЗУ**), в других — и ПЗУ, и оперативная память. Об этих тонкостях и пойдет сейчас речь.

Представьте себе, что вы читаете захватывающую книжку. Она настолько интересна, что вы буквально переселились в другой мир, переживая приключения главных героев. И тем не менее в данный конкретный момент времени перед вами всего одна страничка, информация из которой поступает вам в мозг и там анализируется. Оперативную память компьютера вполне можно уподобить вот этой самой страничке книжки, находящейся прямо перед вашими глазами в процессе чтения.

Итак, **оперативная память** — это запоминающее устройство, предназначенное для информации, непосредственно участвующей в процессе выполнения операций, выполняемых процессором.

Конечно, вам важна информация и из предыдущих страниц, но она уже обработана и отложилась в памяти. Так же и компьютер, обработав информацию из оперативной памяти, записывает ее во **внешнюю память** и «листает книжку» дальше, считывая в оперативную память очередную порцию информации.

Мы уже подчеркивали, что компьютер предназначен для работы с большими объемами информации. Никакой оперативной памяти никакого компьютера не хватит, чтобы удержать ее всю. Ведь и человек не может все упомнить. Поэтому люди пользуются записными книжками, магнитофонными лентами, видеокассетами и т. п. Подобные «записные книжки» имеются и у ЭВМ. Для **персонально-**

го компьютера (ПЭВМ) — это главным образом **жесткие** и **гибкие магнитные диски**, **магнитооптические диски**, **лазерные (оптические) диски**.

Гибкий магнитный диск называют еще **дискетой**. На одной дискете может храниться до 600 страниц текста — этого достаточно, чтобы поместить несколько школьных учебников (правда, без рисунков). Специальное устройство, называемое **дисководом**, позволяет записывать на дискету и считывать с нее информацию. Жесткие диски, как правило, несъемные. Оптические диски работают гораздо быстрее гибких, но медленнее жестких. Зато они съемные и очень объемные (на них размещается примерно в 500 раз больше информации, чем на гибких дисках).

Жесткие, гибкие, лазерные, магнитооптические диски, магнитные ленты и т. п. называются **внешними носителями информации**. И вовсе не потому, что их можно взять и вытащить, а потому, что процессор не имеет прямого доступа к информации, записанной на них.

Аналогия с человеком наверняка подсказывает вам, что ЭВМ должна иметь нечто заменяющее ей органы чувств (чтобы, например, вступать в контакт с человеком) и позволяющее ей связываться с внешним миром. И действительно, каждая ЭВМ снабжена такими устройствами. Они называются **устройствами ввода-вывода**, или периферийными устройствами. С помощью **клавиатуры** или **манипулятора мышь** человек дает ЭВМ задания; с помощью **сканера** в компьютер можно ввести графическое изображение; с помощью дисководов компьютер получает информацию из внешней памяти, а итоги своей работы ЭВМ выводит на экран **дисплея (монитора)** или на бумагу при помощи **принтера**.

Существуют и другие периферийные устройства, позволяющие компьютерам обмениваться информацией по телефону, воспринимать и обрабатывать звуковую и видеоинформацию, закодировав ее предварительно в символьном виде, управлять автоматизированными производствами, изготовлять чертежи...

ВОПРОСЫ И ЗАДАНИЯ

1. Каковы главные компоненты любого компьютера?
2. В чем разница между оперативной и внешней памятью?
3. Назовите устройства ввода-вывода, о которых рассказано в параграфе.
4. Разузнайте, какие еще существуют периферийные устройства компьютера и для чего они предназначены. Сделайте об этом сообщение на уроке информатики.
5. Вспомните, в каких областях человеческой деятельности используются компьютеры.

Первый раз в компьютерном классе

Вы пришли в компьютерный класс. Прежде всего нужно позаботиться о безопасности своей работы за компьютером.

Необходимо помнить: к каждому рабочему месту подведено опасное для жизни напряжение!

Техника, с которой вы будете работать, достаточно нежная, поэтому соблюдайте следующие правила:

1. Если вы обнаружили какую-либо неисправность, немедленно сообщите об этом преподавателю. Не работайте на неисправном оборудовании!
2. Не включайте и не выключайте компьютеры самостоятельно.
3. Не дергайте и вообще не трогайте различные провода.
4. Не стучите по клавиатуре и мышке.
5. Не садитесь за клавиатуру с грязными руками.

А теперь познакомимся поближе с клавиатурой ЭВМ. В этом вам поможет сам компьютер и обучающая программа «Клавиатурный тренажер».

Мы вам советуем сразу начать осваивать так называемый слепой десятипальцевый метод работы на клавиатуре. Это не так уж и сложно, самое главное — не торопиться и набраться терпения.

Рассмотрите внимательно рисунок 1.

Рис. 1. Зоны работы пальцев на клавиатуре

На нем указаны зоны ответственности каждого пальца:

- Левый мизинец постоянно находится над буквой **Ф**, а нажимает еще и на буквы Й и Я.
- Левый безымянный постоянно находится над буквой **Ы**, а нажимает еще и на буквы Ц и Ч.

- Левый средний постоянно находится над буквой **В**, а нажимает еще и на буквы У и С.
- Левый указательный постоянно находится над буквой **А**, а нажимает еще и на буквы К, Е, П, М и И. На большинстве клавиатур буква **А** отмечена риской.
- Правый указательный постоянно находится над буквой **О**, а нажимает еще и на буквы Н, Г, Р, Т и Ь. На большинстве клавиатур буква **О** тоже отмечена риской.
- Правый средний постоянно находится над буквой **Л**, а нажимает еще и на буквы Ш и Б.
- Правый безымянный постоянно находится над буквой **Д**, а нажимает еще и на буквы Щ и Ю.
- Правый мизинец постоянно находится над буквой **Ж**, а нажимает еще и на буквы З, Х, Ъ и Э.
- Большие пальцы нажимают на длинную клавишу пробела.

А теперь усаживайтесь поудобнее. В данном случае это означает сесть таким образом, чтобы можно было работать за клавиатурой быстро и без усталости. Для этого:

- Позвоночник должен быть вертикальным, спина опирается на спинку стула.
- Ступни удобно стоят на полу или специальной подставке.
- Расстояние до экрана не менее вытянутой руки.
- Верхний край экрана располагается примерно на уровне ваших глаз.
- Если вы посмотрите на центр экрана, то линия вашего взгляда должна быть перпендикулярна плоскости экрана, т. е. экран должен быть развернут немного вверх.
- Плечи развернуты и опущены, и вам не составляет труда удобно расположить четыре пальца каждой руки над основными клавишами среднего ряда, т. е. пальцы левой руки находятся над буквами Ф, Ы, В, А, правой — над буквами О, Л, Д, Ж.

Вот теперь запускайте клавиатурный тренажер (учитель объяснит вам, как это сделать), но не торопитесь осваивать сразу все буквы. Придерживайтесь того порядка, который предлагает вам клавиатурный тренажер. Самое главное на первых порах — при нажатии на клавиши не нарушать зону ответственности каждого пальца.

Многим поначалу кажется, что гораздо удобнее и быстрее работать одним пальцем одной лишь правой руки, но подумайте о будущем. Десятипальцевый слепой метод сэкономит вам уйму времени, с лихвой возместив затраты на его освоение.

Надеемся, за пару-другую уроков вы достигнете скорости 40—50 знаков в минуту, что вполне достаточно на начальном этапе.

§ 3. Символьное кодирование

Вы, конечно, помните о том, что информатика — наука о получении, преобразовании, хранении и передаче информации.

Информация... Без нее буквально как без воды. Недаром марафонский бегун пожертвовал жизнью всего лишь ради пары слов. Давайте же уделим свое внимание тому, ради чего не жалели жизни, — передаче информации.

Многие африканские племена до сих пор используют специальные барабаны — тамтамы, обмениваясь сообщениями со скоростью звука. На Руси, где леса гасят звук, применяли другой метод: для передачи срочных сообщений использовался дым костров. Ведь ни один гонец с пограничной заставы не успел бы вовремя предупредить горожан о набеге кочевников — их кони были не менее быстрыми. До сих пор на флоте используется семафорная азбука, когда каждой букве соответствует определенное положение рук сигнальщика, подчеркиваемое флажками.

Но удары барабана, столбы дыма, поднятые вверх руки — это вовсе не то же самое, что звуки человеческой речи или знакомые с первого класса буквы. Однако, подумав, можно обнаружить между ними много общего. Фактически речь идет о том, что каждое такое сообщение, несущее нам информацию, представляет собой последовательность сигналов. А для того чтобы сообщение было не только принято, но и понято, нужно предварительно договориться, что означают, например, два дыма, или последовательность из трех быстрых ударов, или разведенные в стороны руки.

Итак, кроме самого сообщения и физического способа его передачи, появился еще один компонент — **кодирование**. По мере развития цивилизации появлялись новые возможности передачи сообщений, а значит, и новые способы кодирования. Мы остановимся только на одном из них.

Майкл Фарадей в 1831 г. сделал открытие, буквально перевернувшее наш мир: он изобрел способ получения электрического тока. И чуть ли не сразу же электрический ток был использован для передачи сообщений — американский изобретатель Сэмюэл Морзе создает и широко внедряет в практику телеграфные аппараты и линии связи. В какой-то степени Морзе пришлось решать проблему, аналогичную проблеме передачи сообщений по африканскому барабанному телеграфу. И электрический ток, и барабан имеют весьма небогатые выразительные возможности. По барабану можно либо стучать, либо нет. Электрический ток или идет, или нет. Поэтому и кодировка, предложенная Морзе, использовала всего три своеобразные буквы: длинный сигнал (тире), короткий сигнал (точка), нет сигнала (пауза) — для разделения букв. Например, знаменитый сигнал бедствия SOS (Save Our Souls — спасите наши души) кодируется так:

• • •	<пауза>	— — —	<пауза>	• • •
S		O		S

Код Морзе вот уже полтора века служит человечеству. Он до сих пор используется на радиостанциях, потому что его сигналы пробиваются сквозь такие атмосферные помехи, которые глушат любую членораздельную речь.

С течением времени телеграф превратился в массовое средство передачи сообщений, доступное (в принципе) любому желающему, но все же достаточно дорогое. Требовался хорошо обученный оператор, виртуозно владеющий специальным ключом, замыкающим и размыкающим электрическую цепь. И тем не менее его скорость передачи сообщений не шла ни в какое сравнение со скоростью работы машинисток, набивающих текст с помощью клавиатуры, — ведь, чтобы передать одну букву, скажем О, надо трижды нажать на ключ, а машинистке достаточно один раз ударить по клавише.

Вот бы совместить пишущую машинку с телеграфным аппаратом! Но для этого нужно автоматизировать процесс кодирования-декодирования информации. **Автоматизировать** — это значит создать такое устройство, которое бы выполняло работу без вмешательства человека. В данном случае речь идет об устройстве, превращающем буквы человеческого алфавита в последовательности точек и тире. К сожалению, все попытки сделать машину, понимающую код Морзе, были безуспешными. Предложенные варианты оказывались излишне громоздкими, ненадежными и дорогими.

Конечно, техника начала века была еще не столь совершенна, как сейчас, но свою роль сыграло и то, что сам по себе код Морзе был весьма сложен для распознавания его автоматами. Более удачный код был предложен немецким изобретателем Бодо. Во-первых, в нем использовалось только два сигнала (например, точка и тире без паузы), а во-вторых, чтобы не возникала проблема отделения одной буквы от другой, все буквы кодировались последовательностью сигналов одинаковой длины. Аппараты Бодо были просты в производстве и надежны. С их помощью удалось сделать телеграф на самом деле массовым средством передачи срочных сообщений.

Давайте теперь встанем на место Бодо и подумаем, сколько же нужно сигналов, чтобы закодировать все буквы. Для удобства записи будем обозначать сигнал одного типа нулем (0), а другого типа единицей (1). Конечно, можно было бы договориться обозначать сигналы и какими-нибудь другими знаками, например ↑ и ↓, но, как вы позже увидите, это менее удобно.

Итак, последовательностью из одного сигнала можно закодировать всего две буквы (рис. 2, *а*).

Если бы наш язык состоял лишь из этих двух букв, нам бы этого и хватило. Но в русском языке букв несколько больше. Поэтому продолжим рассуждения. Последовательностью из двух сигналов можно закодировать уже четыре буквы (рис. 2, *б*). Это получше, но и с помощью этих букв тоже много не скажешь.

0	А
1	Б

а)

00	А
01	Б
10	В
11	Г

б)

000	А
001	Б
010	В
011	Г
100	Д
101	Е
110	Ё
111	Ж

в)

Рис. 2

Трехсигнальной последовательностью можно закодировать уже восемь букв (рис. 2, *в*). Это еще лучше. Можно, например, спросить «ГДЕВАЗА» или сообщить, что «ДЕДВЁЗЕЖА». Но хочется большего.

Думается, вы уже догадались, что с помощью последовательности из четырех сигналов можно закодировать шестнадцать букв, а пятисигнальной — тридцать две.

Возьмите какую-нибудь телеграмму. Вы увидите, что в ней все буквы только прописные. А вместо точек и запятых стоят слова ТЧК и ЗПТ. Поэтому, хотя пятисигнальных последовательностей достаточно, чтобы изъясняться на русском языке, мы продолжим ее наращивание.

С помощью последовательности из шести знаков (нулей и единиц) можно закодировать уже 64 символа. Но если хотеть, чтобы в сообщении были прописные и строчные буквы, а также цифры, этого недостаточно.

На числе «семь» можно остановиться. Этого хватает для того, чтобы закодировать сообщения на хорошем русском языке. Именно таков отечественный код КОИ-7. Сокращение КОИ родилось из первых букв словосочетания «код обмена информацией».

Чтобы не употреблять длинный оборот «последовательность из стольких-то знаков, каждый из которых нуль или единица», люди договорились появление одного такого знака в последовательности называть словом **бит** (от английского BInary digiT — двоичная цифра). Теперь можно сказать, что последовательность из шести нулей или единиц — это *шестибитная* последовательность, а КОИ-7 — это *семибитное* кодирование русскоязычных сообщений.

Теперь, наверно, уже не покажется совершенно неожиданным следующее заявление:

Всю информацию, циркулирующую внутри компьютера, можно рассматривать как сплошной поток всего лишь из двух символов: 0 (*нуля*) *и* 1 (*единицы*).

Каким образом при этом компьютер ухитряется обрабатывать и текст, и рисунки, вы узнаете из следующих параграфов.

В большинстве первых компьютеров использовался именно семибитный код. Однако с развитием техники это стало довольно неудобно. Новый код был уже восьмибитным и основывался на американском стандартном коде информационного обмена (**ASCII** — **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange). В частности, именно благодаря восьмибитному кодированию мы безо всяких проблем используем в тексте прописные и строчные буквы и русского, и латинского алфавитов, знаки препинания, цифры, специальные символы вроде № и конечно же **пробел**. Это очень важный символ — ведьбезнегочитатьтекстпростопротивно.

Последовательность восьми бит договорились называть словом **байт**. Вот и получается, что *один символ занимает в памяти компьютера ровно один байт*.

Но и один байт окажется маловат, если требуется оценить, сколько места в памяти компьютера занимает, скажем, десяток страниц текста. Поэтому были введены более крупные единицы — **килобайт** (обозначение Кбайт), **мегабайт** (Мбайт), **гигабайт** (Гбайт)... Соотношения между ними таковы:

1 Кбайт = 1024 байт,
1 Мбайт = 1024 Кбайт,
1 Гбайт = 1024 Мбайт.

Конечно, в современном мире, опутанном компьютерными сетями, даже восьмибитного кодирования недостаточно: есть же арабский алфавит, два японских, хинди, математическая символика и т. д. Поэтому не так давно был предложен новый стандарт символьного кодирования **UNICODE**, где каждый символ кодируется уже двумя байтами.

ВОПРОСЫ И ЗАДАНИЯ

1. Каков коэффициент пересчета байт в килобайты; килобайт в мегабайты? А коэффициент пересчета бит в байты? Сколько байт в одном мегабайте?
2. Посчитайте, сколько байт содержит одна страница вашего учебника. Выразите полученное число в килобайтах.
3. «Сколькобитное» кодирование вы бы предложили для языка племени мумбо-юмбо, в алфавите которого 16 букв, и все прописные, а цифр и знаков препинания и вовсе нет?
4. Если в предыдущем задании ваш ответ 4, то найдите ошибку. Без какого символа нельзя обойтись?
5. Сколько символов можно закодировать, используя UNICODE?
6. В объяснительном тексте приведен некоторый код, которым закодированы первые 8 букв русского алфавита.
   а) Запишите в этом коде фразы «ГДЕВАЗА» и «ДЕДВЁЗЕЖА». Объясните, почему в них нет пробелов.

б) Придумайте еще какую-нибудь фразу, которую можно закодировать данным кодом. Можете даже посоревноваться с одноклассниками, у кого фраза длиннее.

§ 4. Текстовый редактор

Конечно, каждый грамотный человек должен знать, что вся информация, так или иначе присутствующая в компьютере, закодирована последовательностями бит. Это знание позволяет определить, поместится на дискету или нет нужная вам информация, оценить время, необходимое для передачи данного сообщения в компьютерной сети, и т. д. Но во многих случаях, решая с помощью компьютера какую-либо задачу, вам вовсе не надо задумываться, как именно закодирован тот или иной символ. И мы не приводим кодов ни русских букв, ни цифр, ни знаков препинания... Вспомните свою работу на клавиатурном тренажере. Вы набирали заданный текст, еще не подозревая ни о каком битовом кодировании. Обо всем заботилась компьютерная программа, которую неукоснительно исполнял ваш компьютер.

Этот ваш самый первый опыт показывает, что компьютер можно применять для подготовки текстовых документов. При этом даже самый разборчиво написанный текст, созданный вами при помощи авторучки, конечно же уступает тому почти типографскому тексту, который печатает принтер.

Мы вовсе не хотим сказать, что поздравления своим родным и друзьям надо непременно печатать на принтере. Аккуратный рукописный текст на поздравлении выглядит живым, доверительным. По почерку можно судить об авторе текста. Существует даже целая наука, позволяющая определить основные черты характера человека по почерку. Когда же речь идет о деловой переписке и подготовке документов, тут, согласитесь, просто необходимо совместить разборчивость, грамотность и аккуратность, что очень даже непросто. Вспомните хотя бы, сколько неприятностей доставляет вам порой всего лишь одна неверно написанная буква!

Итак, давайте более пристально рассмотрим технологию написания писем, книг и документов. Конечно, можно сказать, что уж где-где, а здесь прогресс шел постоянно. Сначала надписи вырубали на камнях. Затем выдавливали стилом на глиняных дощечках. Царапали на бересте. Писали охрой на керамике и палочками на папирусах, кисточками на шелке и перьями на бумаге. Гусиные перья сменялись перьевыми ручками, те — авторучками, авторучки — пишущими машинками... А если вспомнить шариковые ручки, фломастеры, «мокрый шарик», особо тонкие автоматические карандаши!.. А сколько типов пишущих машинок!..

Однако не менялось главное: чтобы внести изменения в текст, его надо заново переписывать (если, конечно, стремиться к акку-

Рис. 3. Основные функциональные возможности текстового редактора

ратности). Вы по себе знаете, что такое черновик и сколько сил и времени отнимает переписывание начисто обычного школьного сочинения. Может быть, поэтому многие ученики больше думают не о содержании сочинения, а о том, как бы не допустить помарки.

А каково писателям! Лев Николаевич Толстой, например, переделывал «Войну и мир» более восьми раз, и каждый раз приходилось (целиком!) переписывать роман (правда, этим занималась главным образом его жена, Софья Андреевна).

Появление компьютеров коренным образом изменило технологию письма. С помощью специальной программы, которая называется **текстовым редактором**, на экране компьютера можно увидеть любой текст и внести в него (при необходимости) любые изменения. Хочешь букву заменяй, хочешь целую страницу. Современные текстовые редакторы обладают такими возможностями обработки текстов, что их стали называть **текстовыми процессорами**.

Текст можно раздвигать, вставляя новые слова. Можно стирать отдельные буквы и переставлять целые абзацы, автоматически заменять во всем тексте одно слово другим. Многие редакторы текстов умеют автоматически разбивать текст на страницы и нумеровать их.

Они могут следить за размером полей и выравнивать текст... Им можно даже поручить обнаружение и исправление орфографических ошибок!

Одним словом, текстовый редактор — это принципиально новый инструмент для работы, имеющий множество уникальных функций, о которых и мечтать-то не приходится при использовании традиционных методов письма. Эти функции иллюстрирует рисунок 3, и в следующем параграфе мы поговорим о них более подробно.

ВОПРОСЫ И ЗАДАНИЯ

1. Что такое текстовый редактор?
2. Какие изменения в тексте можно производить с помощью текстового редактора?

§5. Стандартные инструменты текстового редактора

При работе с редактором текстов роль бумаги играет специально выделенная часть дисплея, обычно называемая **окном**. Текст представляется написанным на длинном свитке, часть которого видна в этом окне. В окне располагается **курсор**: либо небольшой прямоугольник, либо вертикальная или горизонтальная мерцающая черточка, стоящие на экране там, где должен будет появиться очередной символ. С помощью клавиш со стрелками или мыши курсор можно переместить в любое место экрана.

Используя клавиши управления курсором и еще 4 клавиши, описанные ниже, можно перемещать (обычно говорят «прокручивать») текст в окне текстового редактора. Кстати, процесс просмотра информации (в частности, текста) на компьютере с помощью прокрутки называется **скроллингом** (от английского слова scroll — свиток).

При работе с текстовым редактором обычно используются следующие стандартные клавиши:

Вставка (<Insert>) переключает режим ввода символов. В режиме ***Вставка*** новый текст, который вы набираете, будет сдвигать вправо старый текст. В режиме ***Замена*** новый текст будет набираться прямо поверх старого.

Удаление (<Delete>) удаляет символ, отмеченный курсором или стоящий справа от него (если курсор — это вертикальная черточка). При этом весь текст, находящийся правее курсора, сдвигается на одну позицию влево.

Назад (<Back Space>) — это клавиша, расположенная в правом верхнем углу буквенной части клавиатуры, на ней, как правило, изображена стрелка, направленная влево. Клавиша

удаляет символ, стоящий слева от курсора. Весь текст справа от курсора сдвигается на одну позицию влево.

Смена регистра (<Shift>) служит для ввода прописных (больших) букв, а также различных специальных символов, расположенных над цифрами. Чтобы воспользоваться данной клавишей, нужно нажать ее и, не отпуская, нажать клавишу буквы (или цифры). Тем, кто сталкивался с пишущими машинками, можно сказать, что эта клавиша точно такая же, как и специальный рычажок на пишущей машинке, временно поднимающий литеры вверх.

Фиксация верхнего регистра (<Caps Lock>) — это все равно как если бы мы держали клавишу Смена регистра в постоянно нажатом положении. Требуется тогда, когда надо писать только прописными буквами. Клавиша имеет световой индикатор, позволяющий определить, включен этот режим или нет, без нажатия клавиши с буквой.

Табуляция (<Tab>) служит для перемещения курсора на заранее заданную строго определенную позицию в строке.

На страницу вверх (<Page Up>). Под словом «страница» в данном случае понимается та часть текста, которая видна на экране компьютера. При нажатии на эту клавишу текст прокручивается вверх как раз на размер экрана.

На страницу вниз (<Page Down>) — то же самое, что и предыдущая клавиша, только при нажатии на эту клавишу текст прокручивается вниз на размер экрана.

В начало строки / текста (<Home> / <Ctrl>+<Home>). Если просто нажать эту клавишу, то курсор перескочит в начало той строки, с которой вы работаете, а если нажать и удерживать клавишу <Ctrl> и нажать клавишу <Home>, то курсор перескочит в начало текста.

В конец строки / текста (<End> / <Ctrl>+<End>) — то же самое, что и предыдущая клавиша, но курсор перескакивает либо в конец текущей строки, либо в конец текста.

Конечно, это далеко не все возможные клавиши и сочетания клавиш, облегчающих создание нужного текста, но зато они работают так, как указано, практически во всех текстовых редакторах.

Используя клавиши вставки и удаления символов, необходимо иметь в виду следующее. В электронном тексте присутствуют *невидимые* символы, которые воспринимаются компьютером точно так же, как и самые обычные буквы.

С одним из таких символов вы уже очень хорошо знакомы — это *пробел*. А невидимый он потому, что, глядя на экран, сразу и не понять, сколько пробелов стоит между двумя словами. Определить, сколько же их в конце предложения, совсем сложно.

Невидимым символом является и символ **табуляции**.

Третий, самый коварный невидимый символ — **перевод строки**. Он вставляется в текст тогда, когда вы нажимаете клавишу <Ввод> (<Enter>). Многие редакторы самостоятельно делают перенос слов на новую строку и трактуют этот символ как начало нового абзаца.

Представьте теперь, что курсор стоит в конце абзаца и вы нажали клавишу <Delete>. Если в тексте стоял символ перевода строки, то, как и любой другой символ, он удалится и два абзаца объединятся в один.

Того же эффекта можно добиться, если поставить курсор в начало абзаца и нажать клавишу <Back Space>.

Одним словом, невидимые символы — «дело тонкое», и будем надеяться, они не сильно осложнят вашу работу. Кстати, в некоторых современных редакторах их при желании можно увидеть на экране (но, естественно, не в готовом напечатанном тексте).

Автоформатирование и автоперенос

Большинство редакторов текстов автоматически следит за длиной строки и в нужный момент либо делает перенос слова, либо перемещает его на новую строку целиком. Они следят за размером полей и выравнивают текст. Посмотрите-ка внимательно на текст любого вашего учебника. Он аккуратно выровнен и слева, и справа. И конечно же это было сделано компьютером в автоматическом режиме.

Обычно можно заказать три типа выравнивания: по левой границе (обычный тип для не очень больших деловых документов), по обеим границам (для больших текстов) и по правой границе (как правило, для специальных оформительских целей). В случае выравнивания по обеим границам текстовый редактор равномерно растягивает пустые места между словами (некоторые текстовые редакторы просто добавляют между ними пробелы), добиваясь красивого расположения текста.

Если вдруг возникла необходимость изменить параметры строки и/или страницы (например, расположить текст в две колонки или повернуть страничку на 90°), вовсе не требуется вводить текст заново. Редактор самостоятельно расположит текст в соответствии с изменившимися требованиями.

Кстати, полезно знать и об основных стандартных параметрах страниц, используемых при подготовке документов.

- Стандартная машинописная страница имеет размер 210 × 297 мм. Такой формат называется форматом А4.
- Страница может иметь книжную ориентацию или (после поворота на 90°) альбомную.
- Стандартная строка на такой странице в книжной ориентации содержит примерно 63—65 символов при использовании шрифта, похожего на шрифт пишущей машинки.

- Необходимо помнить и о полях: 2,5 — 3 см слева, 2 см справа и сверху, 1,5 — 2 см снизу.

Набор шрифтов

Возьмите любую книжку и откройте последнюю страницу. Там, где напечатаны данные об издательстве, тираже и т. п., вы почти наверняка найдете надпись вроде такой: «Гарнитура школьная».

Что означают эти слова — понять сложно, если не знать, что **гарнитурой** называется просто-напросто вид букв или, как говорят, **шрифт**, которым напечатана книга. Приведем несколько образцов шрифтов:

Этот шрифт называется Парсек.
А это шрифт Компакт.
Пример шрифта нижца.
Это шрифт для заголовков, называется Баухаус.
Изящный шрифт, имитирующий каллиграфию, называется Декор.

При оформлении документов следует обращать внимание на гарнитуру шрифта.

Так, довольно странно выглядит строгий деловой документ, набранный шрифтом Декор.
Ничуть не лучше смотрится и рассказ для детей, набранный шрифтом Компакт.

И таких шрифтов, которые могут использоваться в текстовом редакторе, несколько сотен.

Кроме названий, определяющих вид букв, шрифты имеют размер, называемый **кеглем**. Стандартный шрифт пишущей машинки, к которому стараются приблизиться, печатая деловые документы, очень похож на шрифт гарнитуры Times размером в 14 пунктов. (Здесь пунктом называют не раздел текста, а специальную единицу длины, используемую в издательском деле.) Вообще хорошо читаются шрифты с кеглем от 9 до 14 пунктов.

В полиграфии и издательском деле различают шрифты с засечками и без засечек (их еще называют рублеными). Например:

Этот текст набран шрифтом с засечками.

А этот текст набран рубленым шрифтом.

Шрифты с засечками как бы визуально объединяют слово в единое целое, и это увеличивает скорость чтения на 10–15%. Рубленые шрифты, как правило, используются в заголовках и подписях к рисункам.

Изменяя размер и форму шрифта, можно добиваться самых различных эффектов, но не надо впадать в крайности: большое количество шрифтов на одной странице ухудшает восприятие текста и вряд ли свидетельствует о хорошем вкусе. (Можете это проверить практически на любой газете, основу которой составляют рекламные объявления.)

Табличный редактор и калькулятор

Табличный редактор позволяет разграфить часть страницы документа горизонтальными и вертикальными линиями. При вводе текста в клетку получившейся таблицы (такую клетку нередко называют *ячейкой*) можно быть уверенным, что он будет строго ограничен левой и правой вертикальными линиями и не выйдет за их пределы. При необходимости горизонтальные и вертикальные линии можно сделать видимыми, а часть ячеек выделить штриховкой.

Если в ячейках таблицы находятся числа, то редактор может выполнить с ними различные арифметические действия: например, найти сумму по столбцу или строке, вычислить среднее или максимальное значение и т. п.

Кроме того, многие текстовые редакторы предоставляют такую услугу: если вы запишете арифметическое выражение, то по нажатии специальной клавиши компьютер вычислит его значение. Иными словами, можно считать, что в текстовый редактор встроен простенький калькулятор.

Проверка орфографии

К сожалению, проверка орфографии текстовым редактором сводится к поиску слова в электронном словаре. А почему «к сожалению» — сейчас объясним. Допустим, что в тексте встретилось предложение: «Карова дает молоко». Если включить режим орфографической проверки, то, скорее всего, компьютер предложит либо «карова» заменить словом «корова», либо пропустить (т. е. оставить все как есть), либо добавить в словарь, поскольку слова «карова» в словаре не оказалось. И если выбрать режим «Добавить», то в следующий раз слово «карова» уже не будет считаться ошибкой. Впрочем, в исходном словаре тоже встречаются ошибки — заполняли-то его тоже люди, а ошибиться может каждый.

Что касается знаков препинания — запятых, двоеточия, тире и т. д., то компьютер пока слишком часто в них путается. Так что на проверку надейся, а сам не плошай.

Вставка графических объектов

Если вы с помощью компьютера создали рисунок, то текстовый редактор позволяет вставить его в текст. (О том, как создавать такие рисунки или где взять готовые, мы расскажем позже.) Новый термин, с которым необходимо познакомиться, — это *обтекание*.

Давайте опять откроем любую книгу или журнал. Можно заметить, что большая часть иллюстраций «встроена» в текст, который их аккуратно огибает. Вот это и называется «обтекание графики текстом». Все современные редакторы умеют выполнять эту функцию автоматически.

Система работы с файлами

Документ, который вы подготовили, может быть не только распечатан на принтере, но и сохранен на каком-либо внешнем носителе информации для дальнейшего использования и правки. Сохранять можно, конечно, не только документы, подготовленные в текстовом редакторе, но и любую информацию, представленную в компьютере в электронном виде. Для этого такой информации присваивается некоторое имя. Информация, хранящаяся на внешнем носителе как единое целое и обозначенная одним именем, называется **файлом**. **Имя файла** позволяет различать файлы и дает возможность вызвать содержимое файла в оперативную память компьютера.

Группы файлов по желанию пользователя могут быть объединены в одну **папку** (или **директорию**). Такой папке также дается имя, обычно объявляющее общий признак, по которому файлы объединены именно в эту папку. Например, папку можно назвать PISMA или ПИСЬМА и «складывать» в нее всю свою электронную переписку.

К сожалению, до сих пор многие компьютерные программы, работающие с файлами, накладывают очень строгие ограничения на имена файлов и папок. Они не позволяют, например, назвать файл «Мое письмо Васе в день его рождения 12 мая 2000 года». А можно так: «PISMO12.DOC», что не слишком-то информативно.

Поэтому многие редакторы позволяют создать свою внутреннюю табличку вот такого типа:

Содержание файла	Имя файла
Мое письмо Васе в день его рождения 12 мая 2000 года	PISMO 12.DOC

И пользователям (т. е. нам с вами) вовсе не надо каждый раз ломать голову, пытаясь вспомнить содержание файла с данным именем или, наоборот, имя файла, в котором хранится потребовавшаяся информация. Все это легко сделать, имея под рукой табличку с описанием файлов.

Объединение документов, многооконность

Практически любой текстовый редактор позволяет одновременно работать с несколькими документами и с помощью **буфера обмена** переносить текст из одного документа в другой. При этом каждый документ открывается в своем окне, размеры которого можно менять. Легко добиться того, чтобы на экране были видны, хотя бы частично, вообще все документы, с которыми вы в данный момент работаете.

Что касается буфера обмена, то так называют специально выделенную область памяти компьютера, куда можно поместить выде-

ленный фрагмент текста, или рисунок, или таблицу, или какой-либо еще объект, работа с которым предусмотрена в данном редакторе. Информацию, записанную в буфер обмена, можно вставить в другое место того же документа или в другой документ. Записанный фрагмент сохраняется в буфере до тех пор, пока вы не запишете туда новую информацию. Поэтому одну и ту же информацию можно вставлять из буфера обмена столько раз, сколько потребуется.

Предварительный просмотр страницы перед печатью

Режим, при котором вид документа на экране соответствует тому, как он будет напечатан на бумаге, называется «Что вижу, то и получаю», или **WYSIWYG** (это далеко не очевидное, скажем прямо, сокращение происходит от английской фразы «What You See Is What You Get»). Далеко не все текстовые редакторы поддерживают этот режим в процессе работы, но почти все могут показать, как будет смотреться на бумаге уже готовый документ.

Конечно, мы рассказали только о наиболее важных функциях современного текстового редактора. Со многими дополнительными возможностями конкретного редактора текстов, который есть на вашем компьютере, вы познакомитесь в процессе выполнения лабораторных работ.

Но, думаем, вы уже поняли, что между гусиным пером и пишущей машинкой дистанция гораздо меньшая, чем между пишущей машинкой и хорошим текстовым редактором.

Простейшие функции текстового редактора

У всех текстовых редакторов существуют действия, которые выполняются одинаково, в частности вставка и удаление символов, вставка и удаление строки, соединение строк и разбиение строки на две. Напомним, что под символом понимается буква, цифра, знак препинания, специальный знак, а также пробел. Пробел — это такой же полноправный символ, его можно вставлять и удалять.

А теперь выполните задания:

❶ Наберите с помощью клавиатуры слова новогодней песенки «Елочка» (или какое-нибудь другое стихотворение, которое вы еще помните), причем заголовок наберите заглавными буквами и каждую строку начинайте тоже с заглавной буквы. Располагайте текст около левого края экрана. Используйте при работе вставку, удаление и т. д.

❷ Отделите все строки друг от друга, вставляя каждый раз пустую строку.

❸ Употребляя операцию разбиения строки, сделайте так, чтобы в каждой строке осталось только одно слово, и расположите эти слова лесенкой (как у Маяковского).

❹ Соедините строки так, чтобы они стали такими, как до разбиения. Что у вас получилось? Если это первоначально набранный текст, то ПОЗДРАВЛЯЕМ!

Работа с окнами и черчение

Вы уже знаете, что место на экране компьютера, выделенное для текста, называют окном. Современные текстовые редакторы имеют, как правило, несколько окон для работы с текстом. И раз вы собрались сделать эту лабораторную работу, значит, ваш текстовый редактор многооконный: используя его, можно открыть несколько окон и поместить туда разнообразную информацию. Окна могут располагаться в разных местах экрана и иметь разные размеры. Как именно менять конфигурацию окон, вам расскажет учитель или вы прочитаете в инструкции пользователю текстовым редактором.

Выполните следующие задания:

❶ Перед вами уже открытое пустое окно. Уменьшите его высоту и ширину вдвое.

❷ Переместите получившееся окно в правую нижнюю часть экрана.

❸ Создайте еще одно окно. Уменьшите вдвое его высоту и ширину, а потом переместите окно в правую верхнюю часть экрана.

❹ Создайте третье окно, измените размеры и поместите его в левую нижнюю часть экрана.

❺ Создайте четвертое окно, измените размеры и поместите его в левую верхнюю часть экрана.

А сейчас приятная неожиданность! Почти в каждом текстовом редакторе (надеемся, и в том, с которым работаете вы) имеется возможность не только создавать текст, но и рисовать, по крайней мере прямые линии. Может быть, вы думаете, что это так мало — проводить прямые линии? Тут вы ошибаетесь! Прямая линия — мощное изобразительное средство. Выполните следующее задание, и вы сами в этом убедитесь.

❻ Нарисуйте в созданных вами окнах такие же картинки (рис. 4).

Если ваш редактор позволяет, на рисунках в третьем и четвертом окнах у начерченных фигур сделайте «скругленные» углы.

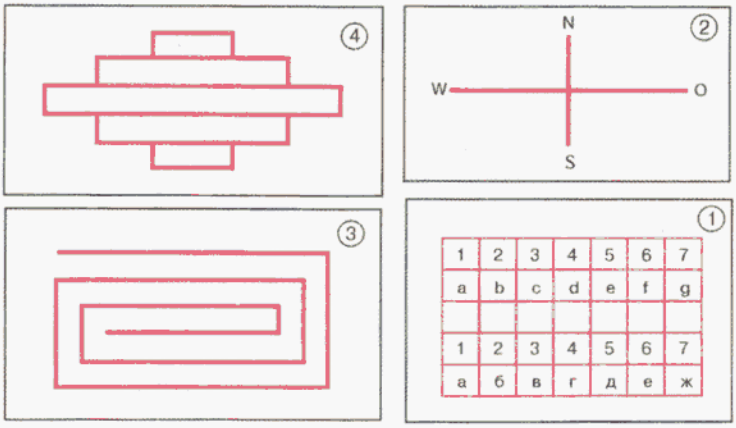


Рис. 4. Задание для выполнения в текстовом редакторе

Работа со шрифтами

Оформляем наклейку на тетрадь

Ваш текстовый редактор позволяет использовать разные шрифты: печатный и рукописный, подчеркнутый и увеличенный, наклонный и выделенный. Причем можно, например, использовать рукописный подчеркнутый шрифт, если выбрать одновременно рукописный шрифт и режим подчеркивания. Попробуйте, используя различные шрифты, оформить наклейку на тетрадь, а потом сделать рамочку.

Выполните задания:

❶ Наберите на клавиатуре следующий текст (указав, конечно, свои данные):

ТЕТРАДЬ
по математике
ученицы 10 А класса
школы № 104
г. Санкт-Петербурга
Степановой Насти

❷ Используя различные шрифты, преобразуйте каждую строку. Чтобы изменить шрифт, надо выделить фрагмент текста как ***блок***. Блок будет помечен другим цветом. Затем выберите шрифт

(как это сделать, вам объяснит учитель или вы узнаете из инструкции пользователю).

Ваш текст может стать, к примеру, таким:

ТЕТРАДЬ	печатный, подчеркнутый, увеличенный, наклонный, выделенный, гарнитура Pragmatika
по математике	печатный, наклонный, гарнитура Newton
ученицы $10^{А}$ класса	печатный, выделенный, использован верхний индекс для указания буквы класса, гарнитура Newton
школы № 104	печатный, наклонный, подчеркнутый, гарнитура Newton
г. Санкт-Петербурга	печатный, наклонный, подчеркнутый, выделенный, гарнитура Newton
Степановой Насти	печатный, выделенный, гарнитура SchoolBook

Можете, конечно, использовать другие комбинации шрифтов (те, которые вам нравятся).

❸ Поместите текст в рамочку так, как вам объясняли на уроке.

❹ Просматривая подготовленный текст перед печатью, обратите внимание на рамочку, подправьте ее при необходимости. Попробуйте (с разрешения учителя, разумеется) распечатать наклейку на принтере.

❺ Не забудьте записать ваш файл на диск. Можете указать на наклейке название другой учебной дисциплины и сделать еще одну распечатку.

Ваша работа завершена.

Работа с таблицами

Учимся начислять зарплату

Современный текстовый редактор, как правило, позволяет не только готовить тексты, вставлять рисунки, но и производить не очень сложные расчеты. Такие расчеты может выполнять, например, бухгалтер при начислении заработной платы. Представьте, что вы бухгалтер очень маленького предприятия, на котором работает всего пять человек, и вам надо решить подобную задачу. Предположим для простоты, что оплата труда у всех работников почасовая (т. е. стоимость одного часа работы сотрудника одинакова, но зависит от его квалификации).

Выполните задания:

❶ Создайте таблицу из восьми столбцов и семи строк. В первой строке в каждом столбце укажите его номер. Во второй строке будем указывать названия столбцов. Первый столбец предназначен у нас для фамилий сотрудников предприятия, поэтому во второй строке первого столбца наберите слово ФАМИЛИЯ. Название второго столбца: ОПЛАТА ЗА 1 ЧАС (в рублях). Он, очевидно, будет содержать числа. Третий столбец КОЛИЧЕСТВО ЧАСОВ тоже будет содержать числа (количество отработанных часов за месяц).

❷ Теперь наберите в первом столбце фамилии работников, во втором — соответствующую им стоимость одного часа работы, в третьем — количество отработанных часов.
На экране должно быть приблизительно следующее (табл. 1):

Таблица 1

1	2	3
ФАМИЛИЯ	ОПЛАТА ЗА 1 ЧАС (в рублях)	КОЛИЧЕСТВО ЧАСОВ
1. Иванов	15	82
2. Петров	23	75
3. Сидоров	18	94
4. Степанов	34	39
5. Михайлов	27	76

❸ Теперь нужно определить, какая сумма должна быть начислена каждому сотруднику за отработанное количество часов. Для этого нужно умножить каждое число из второго столбца на соответствующее число из третьего столбца. Заголовок четвертого столбца — ОПЛАТА ЗА ОТРАБОТАННОЕ ВРЕМЯ.

В вашем редакторе для работы со столбцами предусмотрены специальные операции: умножение, сложение, вычитание, а также умножение столбца на коэффициент. Как это делается в вашем редакторе, вам расскажет учитель или вы ознакомитесь с этим по инструкции пользователю.

❹ Заголовок пятого столбца — РАЙОННЫЙ КОЭФФИЦИЕНТ. Для тех, кто живет, например, на Урале, полагается надбавка 15%; для тех, кто живет на Севере, — другая надбавка: 20%; на Дальнем Востоке — третья и т. д. Узнайте, действует ли в вашей местности какой-либо районный коэффициент и если действует, то надо умножить на него весь четвертый столбец (надеемся, вы не забыли перейти от процентов к десятичным дробям).

❺ Заголовок шестого столбца — НАЧИСЛЕНО. Тут должна быть указана сумма четвертого и пятого столбцов — это будет полная начисленная сумма.

❻ Заголовок седьмого столбца — ПОДОХОДНЫЙ НАЛОГ. Подоходный налог составляет 12%, или 0,12. Здесь при расчете учитывается начисленная сумма. Умножьте столбец 6 на число 0,12.

❼ Заголовок восьмого столбца — ИТОГО. Это, наконец, та сумма, которая выдается человеку на руки, и вычисляется она как разность столбцов 6 и 7.

Ваша работа завершена. И хотя на самом деле заработная плата рассчитывается намного сложнее (а кроме зарплаты, бухгалтеру надо рассчитывать приход и расход материалов, вести учет готовой продукции, рассчитывать доход и прибыль и многое другое), вы получили первое представление о том, как может помочь компьютер работе бухгалтерии.

§6. Организация вычислений при помощи компьютера

В последней лабораторной работе с текстовым редактором вы воспользовались важным свойством компьютера — его умением быстро и безошибочно вычислять. Более того, вы, наверно, ощутили, чем он принципиально отличается от простого калькулятора. Одно дело — сидеть с калькулятором и вычислять заработную плату каждого из работников, и совсем другое — перемножать, складывать и вычитать целыми столбцами. Или просто вставить в ячейку таблицы необходимую для расчетов формулу и сразу получить нужный результат. Но, как обычно, хочется большего.

Предположим, мы ошиблись и оказалось, что Иванов Петр Сидорович отработал вовсе не 82 часа за месяц, а целых 105. Тогда придется или вручную пересчитать его зарплату, или повторить несложные, но немного утомительные операции по перемножению столбцов или вставке формул.

А нельзя ли сделать так, чтобы компьютер автоматически пересчитывал все данные в таблице при изменении некоторых из них? Бухгалтеру, например, частенько требуется пересчитывать весь квартальный отчет (а это несколько десятков показателей), если возникает необходимость заменить несколько исходных чисел. Или заново начислять заработную плату всем работникам при изменении всего одного числа — минимальной заработной платы.

Подобные проблемы возникают, конечно, не только у бухгалтеров. Диспетчерам при составлении графиков движения транспорта, экспериментаторам при проведении серий опытов и многим другим приходится решать задачи, в которых изменение значения какого-то одного параметра требует пересчета большого числа результатов.

И компьютер способен учесть огромный объем данных и сослужить добрую службу, если требуется неоднократно проводить однообразные вычисления. Для этого программистами созданы специальные программы — **электронные таблицы**. Но прежде чем познакомиться с ними, давайте подумаем, что требуется для расчета все той же заработной платы.

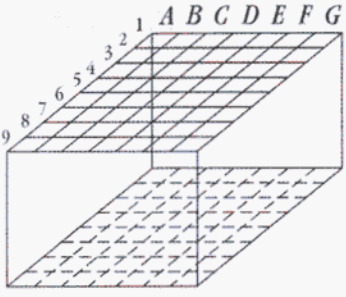


Рис. 5. Структура электронной таблицы

Во-первых, фамилии и инициалы сотрудников и все то, что к ним относится: ставки, количество отработанных ими часов, льготы по налогообложению и т. п. Эту информацию надо просто занести в компьютер для использования в его дальнейшей работе. Такого сорта информация называется **исходными данными**.

Во-вторых, необходимо знать формулы, по которым рассчитываются заработная плата, налоги, премии и т.п. Получающиеся при этом числа называют **рассчитываемыми данными** или **результатами**.

Электронная таблица позволяет хранить в табличной форме большое количество исходных и рассчитываемых данных, а также связи между ними (т. е. формулы). Но главное, при изменении исходных данных все результаты автоматически пересчитываются и изменяются прямо у вас на глазах. Конечно, чтобы это могло происходить, каждая клетка электронной таблицы должна представлять собой не просто кусочек экрана, а достаточно сложный объект, напоминающий ящик, в который можно «складывать» информацию. Поэтому клетки электронной таблицы обычно называют **ячейками**.

Электронную таблицу удобно представлять себе как одноэтажное здание с подвалом: на видимом этаже — привычные для нас числа, буквы и другие символы, т. е. исходные и рассчитываемые данные, записанные в ячейки обычной таблицы, а в подвале — невидимые с первого взгляда формулы, по которым получаются результаты (рис. 5).

Ячейки таблицы, у которой на верхнем слое число, а на нижнем ничего нет, — это, как легко понять, ячейки с исходными данными. Они, напомним, заносятся вручную и компьютером никак не изменяются.

Ячейка таблицы, у которой на нижнем слое формула, — это ячейка, в которую компьютер самостоятельно записывает результат. Вычислить его — долг и забота компьютера.

Идея проста, однако додумались до нее спустя целых двадцать лет после того, как стали применять компьютеры в бухгалтериях. Сделал это Дэниэл Бриклин в 1979 г. Вместе с программистом

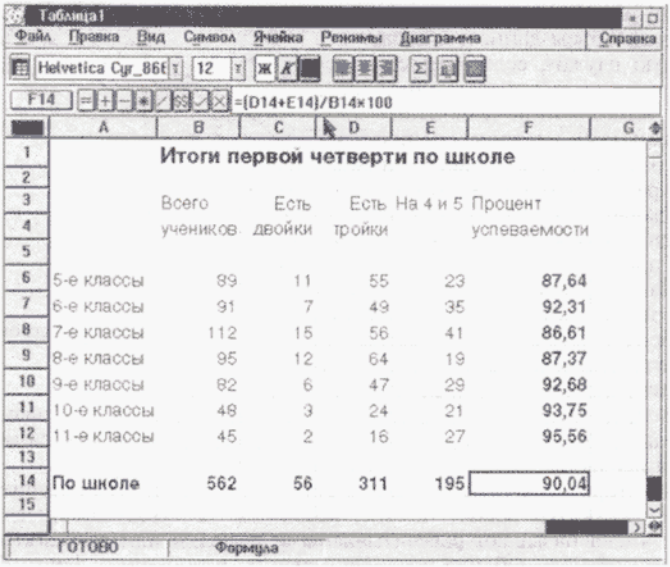


Рис. 6. Внешний вид электронной таблицы на экране компьютера

Робертом Фрэнкстоном они и создали первую в мире электронную таблицу с названием «Визикалк», что по замыслу создателей означало «Визуальный калькулятор».

Даже одна эта программа оправдывала в глазах пользователей приобретение персонального компьютера: более чем 100 тысяч человек купили персональный компьютер ради возможности работать с «Визикалком». Экономисты с восторгом писали, что с помощью этой программы можно почти мгновенно определить прибыль компании, если заработная плата вырастет на 6% при одновременном увеличении производительности труда на 3,5% и уменьшении цены на готовую продукцию на 7% с ожидаемым увеличением сбыта на 14%.

Посмотрите внимательно на электронную таблицу (рис. 6).

Несмотря на экзотическую структуру, видимый этаж электронной таблицы почти ничем не отличается от самого обычного документа, созданного в уже знакомом вам текстовом редакторе.

Правда, страничка разбита на *столбцы*, обозначенные буквами латинского алфавита, и на *строки*, пронумерованные целыми числами, но само это разбиение при выводе на принтер, как правило, делается невидимым, и по внешнему виду документа невозможно понять, сделан он с помощью текстового редактора или электронной таблицы. Благодаря разбиению на пронумерованные строки и

поименованные столбцы каждая ячейка электронной таблицы имеет свой собственный **адрес**.

Так, например, в ячейке B1 написан заголовок таблицы. Вообще-то столбец B не такой уж и широкий, чтобы в нем уместился весь заголовок целиком, но таблица устроена так, что если в правой соседней ячейке ничего нет, то текст перекрывает и ее. Если же в ячейку C1 что-нибудь поместить, то вместо слов «Итоги первой четверти по школе» мы бы увидели только «Итоги п».

Ширину столбцов можно произвольно менять, выбирая наиболее подходящую для различных данных, а некоторые строки — пропускать. Это позволяет красиво оформить документ.

К сожалению, далеко не во всех электронных таблицах есть возможность увидеть «подвальный этаж» целиком, но всегда виден «подвал» той ячейки, на которой стоит курсор в виде прямоугольника. На нашем рисунке он стоит на ячейке F14.

Содержимое «подвала» отображается в специально отведенном для этого месте (на рисунке 6 оно изображено в рамочке над таблицей, куда показывает черная стрелка). Кроме того, во многих электронных таблицах сообщается, что же находится в ячейке: результат вычисления (как в нашем случае, поскольку в нижней рамочке написано слово «Формула») или исходные данные (в той же рамочке будет написано «Текст» или «Число»). Слово «Готово» в левом нижнем углу означает, что произведен автопересчет и все результаты соответствуют формулам. Что же касается формул, то, взглянув на рисунок 6, вы сразу догадаетесь, как их вставлять в таблицу.

Для каждой таблицы существуют специальные правила ее заполнения, указанные в инструкции пользователю:

- Если среди символов, вводимых в ячейку, есть буквы или нечто, чего не может быть в числе, то это текст.

Например, текстом являются следующие последовательности символов: Василий, или 10.234.245, или a1234.

- Если вводится правильное число, то это число.

Например, 234 (разумеется, без кавычек; первый символ " сразу же указывает, что тип данных — текст).

- Чтобы электронная таблица распознавала, что вводится — просто текст или формула, в начале того и/или другого ставится заранее обусловленный знак. Текст обычно начинается с кавычек ", а формула — со знака =, или знака @, или еще какого-нибудь знака (о чем, разумеется, сказано в инструкции пользователю).

Как правило, электронная таблица имеет, помимо всего прочего, целый набор стандартных функций, облегчающих жизнь поль-

зователю. Как вы думаете, какая формула стоит в ячейке B14? Возможно, такая:

=B6+B7+B8+B9+B10+B11+B12

А если надо сложить не семь, а двадцать семь или сто семь чисел? Для этого имеется стандартная функция — *суммирование* содержимого *блока ячеек*:

SUM(B6:B12)

У электронной таблицы есть много разных операций над блоками ячеек. Это поиск максимального или минимального элемента, расчет среднего значения и т. д. Блок ячеек можно скопировать или перенести в другое место таблицы. Надо только помнить, что **блоком ячеек в электронной таблице** называется совокупность всех ячеек, заполняющих некоторый прямоугольник. Для того чтобы электронная таблица знала, с каким блоком ей иметь дело, указывают через двоеточие *адреса* ячеек, стоящих в левом верхнем и правом нижнем углах прямоугольника. Одним словом, современная электронная таблица вполне способна заменить целое бухгалтерское бюро прошлого.

ВОПРОСЫ И ЗАДАНИЯ

1. Для чего предназначены электронные таблицы?
2. Какая информация, имеющаяся в электронной таблице, называется исходными данными, а какая — результатами?
3. Что такое адрес ячейки электронной таблицы?
4. Определите, в каких ячейках таблицы, изображенной на рисунке 6, находятся результаты, а в каких — исходные данные.

§7. Как решать задачи с помощью электронной таблицы

Народная мудрость «Семь раз отмерь, один отрежь» универсальна. И, создавая документ в текстовом редакторе, тоже лучше сначала подумать, что именно вы собираетесь написать. Конечно, на то он и редактор, чтобы, написав какой-то вариант, потом править его до тех пор, пока не получится если не литературный шедевр, то по крайней мере нечто, что и не стыдно дать почитать.

Использование электронной таблицы для решения той или иной задачи требует еще более тщательного предварительного продумывания, в каких ячейках вы будете размещать исходные данные задачи, какие формулы, связывающие исходные данные с результатами, куда поместить и т. д. Чтобы всему этому научиться, в этом параграфе мы разберем две задачи. Вот первая из них.

Экскурсионная поездка

В конце учебного года администрация школы организовала для всех желающих экскурсию и, получив информацию из классов о числе экскурсантов, заказала соответствующее количество автобусов. Зная, что в каждый автобус входит ровно 45 пассажиров, завуч по внеклассной работе уже начала было заполнять таблицу 2, но с ней заспорила классный руководитель 10 В класса. Конечно, всем ребятам из одного класса хотелось бы ехать в одном автобусе. Завуч сказала, что все равно так не получится и кому-то придется ехать в разных автобусах...

Таблица 2

Класс	Едут на экскурсию	Первый автобус	Второй автобус	Третий автобус
10 А	23	23		
10 Б	17	17		
10 В	22	5	9	
10 Г	8		7	
11 А	18			
11 Б	6			
11 В	19			
11 Г	14			

Давайте попробуем с помощью электронной таблицы подобрать такой вариант, чтобы было как можно меньше недовольных. Эта задача отнюдь не простая. Как вы думаете, что должно стоять в ячейках таблицы на пересечении номеров автобусов и классов? Количество человек (как это сделано в таблице 1)? Но если класс целиком садится в один автобус, оно и так нам известно.

Немного отвлечемся от автобусной экскурсии и сформулируем одно из самых главных правил информатики:

Любая задача, связанная с обработкой информации, требует в первую очередь осмысления связи между исходными данными и результатами.

Итак, что общего имеют между собой первый автобус и, скажем, 10 А класс? Ну конечно же 10 А класс либо едет в этом автобусе,

либо нет. Стало быть, и в соответствующей ячейке должны стоять либо единица, либо нуль. А для контроля заполняемости автобуса необходимо добавить еще одну строку (табл. 3):

Таблица 3

Класс	Едут на экскурсию	Первый автобус	Второй автобус	Третий автобус
10 А	23	1		
10 Б	17		1	
10 В	22			
10 Г	8	1		
11 А	18			1
11 Б	6			
11 В	19			1
11 Г	14			
Итого в автобусе		31	17	37

Можете считать, что перед вами электронная таблица, предназначенная для решения задачи об организации экскурсии. Естественно, что пересчет количества людей в автобусе производится автоматически, т. е. у этих ячеек заполнен «подвальный» этаж. А как он заполнен, вы, наверно, уже сами догадались. Выполняя лабораторную работу, вы закончите работу по рассаживанию школьников в автобусы.

Перевозка грузов

Это вторая задача, которую мы разберем, прежде чем приступим к выполнению лабораторной работы.

Перед диспетчером компании «ПАНУРАЛТРАНССИБСЕРВИС» встала непростая задача. Три грузовика компании должны забрать с разных предприятий Новосибирска груз (табл. 4) и доставить его в Омск. Грузоподъемность каждой машины — 12 т, и хотелось бы распределить весь груз примерно поровну. Ну, быть может, допустив 100–150 кг перегрузки у какого-нибудь грузовика.

Таблица 4

Наименование оборудования	Количество	Вес одной упаковки (кг)	Первый грузовик	Второй грузовик	Третий грузовик
Станки (штуки)	11	850			
Трубы (упаковки)	4	1930			
Буровое оборудование (ящики)	2	1700			
Отделочный камень (ящики)	4	1250			
Промышленные электромоторы (штуки)	7	730			
Кабель (бухты)	5	1100			
Всего груза в машине (кг)					

Видимо, вы уже поняли, что и в этом случае диспетчеру поможет электронная таблица.

Как и в первой задаче, постараемся понять связь между строками и столбцами. Она похожа на то, что было в предыдущем задании, но есть и отличие. А именно вовсе необязательно грузить, например, весь кабель на одну машину. Поэтому в ячейках могут стоять не одни только нули и единицы. В данном случае связь заключается в том, сколько упаковок (штук, ящиков) груза берет конкретная автомашина.

И еще одно отличие. Решите-ка такую задачу:

На складе было 11 станков. Четыре увез первый грузовик, три — второй. Остальное увез третий. Спрашивается: сколько станков увез третий грузовик?

Что, задачка для первого класса? Так пусть компьютер сам ее и решает. А для этого необходимо заполнить «подвальный» этаж столбца, соответствующего третьему грузовику. Разумеется, требуется еще строка, которая контролирует загрузку каждой из машин. Ее «подвальный» этаж заполняется аналогично тому, как это было сделано в предыдущей задаче. Желаем успешной работы диспетчером!

Знакомство с электронной таблицей

Что надо знать для работы с электронной таблицей?

- Таблица содержит ячейки, адреса которых задаются с помощью обозначений для соответствующих столбцов и строк, например F5, A8.
- В каждую ячейку можно поместить либо число, либо текст, кроме того, число может быть вычислено по формуле, которая не видна (она как бы находится в «подвальном этаже»).
- При внесении в ячейку формулы или текста может потребоваться ввод сначала специального символа, например «=».
- Редактирование формулы осуществляется по правилам текстового редактора.
- Чтобы избавить себя от лишних манипуляций с клавиатурой, используйте мышь, отмечая с ее помощью нужные ячейки. Их адреса сами окажутся в формуле. Ваша задача — указать знаки операций и не забыть нажать клавишу <Enter>.

Рассаживаем учащихся по автобусам

Условие задачи подробно изложено в § 7. Вспомните его и выполните задания:

❶ Заполните таблицу 5.

Таблица 5

	A	B	C	D	E
1	Класс	Едут на экскурсию	Первый автобус	Второй автобус	Третий автобус
2	10 А	23			
3	10 Б	17			
4	10 В	22			
5	10 Г	8			
6	11 А	18			
7	11 Б	6			
8	11 В	19			
9	11 Г	14			
10	Итого в автобусе				

❷ В ячейках столбцов C, D и E должны находиться либо 1 (единица), либо 0 (нуль), либо ничего (что аналогично нулю). Единица ставится в ячейку тогда и только тогда, когда класс, указан-

ный в строке, целиком едет в автобусе, указанном в столбце. Обратите внимание, что при занесении чисел они прижимаются к правому краю столбца.

❸ Перед тем как начинать «рассаживание» классов по автобусам, необходимо заполнить формулами нижний этаж ячеек C10, D10, E10. Так, в ячейке C10 должна стоять формула

C2*B2+C3*B3+C4*B4+C5*B5+C6*B6+C7*B7+C8*B8+C9*B9.

Это позволит в дальнейшем контролировать загрузку автобусов. Запишите нужные формулы в соответствующие ячейки. Обратите внимание, что в качестве знака умножения в формулах, записываемых в ячейки электронной таблицы, употребляется «*».

❹ Ваша задача — расставить единицы и нули в таблице таким образом, чтобы в каждом автобусе ехало не более 45 человек. Выполните эту работу.

Работа с электронной таблицей

Диспетчер распределяет груз

Условие задачи изложено в § 7. Используя свои навыки работы с электронными таблицами, выполните задания:

❶ Заполните таблицу 6:

Таблица 6

	A	B	C	D	E	F
1	Наименование оборудования	Количество	Вес одной упаковки (кг)	Первый грузовик	Второй грузовик	Третий грузовик
2	Станки (штуки)	11	850			
3	Трубы (упаковки)	4	1930			
4	Буровое оборудование (ящики)	2	1700			
5	Отделочный камень (ящики)	4	1250			
6	Промышленные электромоторы (штуки)	7	730			
7	Кабель (бухты)	5	1100			
8	Итого вес груза			0	0	0

❷ В столбцах D, E и F должны находиться числа, определяющие, сколько упаковок или штук данного груза берет соответствующий грузовик.

❸ Перед тем как начинать распределение груза, необходимо заполнить формулами нижний этаж ячеек D8, E8, F8. Так, в ячейке D8 должна стоять формула

C2*D2+C3*D3+C4*D4+C5*D5+C6*D6+C7*D7.

Это позволит в дальнейшем контролировать загрузку грузовиков. Запишите нужные формулы в соответствующие ячейки.

❹ Кроме того, необходимо заполнить формулами «подвальный этаж» ячеек с F2 по F7, предоставляя компьютеру автоматически загружать третий грузовик теми товарами, которые не взяли первые два. Так, в ячейке F2 должна стоять формула B2–D2–E2. Воспользуйтесь возможностью копирования формулы, чтобы заполнить оставшиеся ячейки с F3 по F7.

❺ Ваша задача — манипулируя с числами в столбцах D и E, так распределить товары, чтобы загрузка каждой из автомашин не превышала 12 т более чем на 100–150 кг.

§8. Графическое представление информации. Монитор

Напомним, что символьная информация была именно тем видом информации, в котором начало «тонуть» человечество. Все эти бухгалтерские расчеты, экономические обоснования, проектирование самолетов, расчеты траекторий и разнообразных таблиц отнимали безумно много сил и средств.

Компьютеры оказались долгожданным спасательным кругом, позволившим резко увеличить эффективность управления производством и применить мощные математические методы, с помощью которых конструировали новые изделия — от одежды до спутников. И все это удавалось сделать с помощью одной только символьной информации. Как вы убедились, читая § 3, ее перевод на язык компьютеров оказался очень простым и вполне естественным.

Давайте теперь посмотрим на процесс получения и переработки информации с другой стороны. С человеческой. Пусть, к примеру, тренерскому совету спортивного клуба «Шустрые Шиповки» был представлен следующий документ (табл. 7):

Таблица 7

Итоги сдачи нормативов в средней группе на конец четверти

	Не сдали	Сдали плохо	Сдали хорошо	Сдали отлично
Первая четверть	12	12	5	1
Вторая четверть	9	15	4	2
Третья четверть	8	9	10	3
Четвертая четверть	4	9	10	7

Можете ли вы качественно оценить работу тренера по этой таблице? Конечно, если вдумчиво посидеть над ней минут 10 — 15, то можно сказать и нечто большее, чем «Положительные сдвиги налицо» или «Процесс пошел».

А теперь преобразуем этот документ (рис. 7).

Сразу видно: основной заботой тренера средней группы во вторую четверть было подтянуть отставших. Количество ребят, не выполнивших нормативы, уменьшилось. Но зато почти не выросло количество тех, кто сдал зачет хорошо и отлично. Видимо, тренером были сделаны правильные выводы, и он всю третью четверть занимался именно с теми ребятами, которые зачет сдали, но плохо. Результат не замедлил сказаться. Кстати, подтянулись и те, кто совсем не сдал зачет. И только в четвертую четверть достаточно много внимания было уделено тем юным спортсменам, кто сдал зачет хорошо, но не отлично. Правильно ли организовал учебный год тренер средней группы — решать тренерскому совету клуба «ШШ». Нам важно другое — весь этот анализ можно сделать буквально после первого взгляда на диаграмму.

Этот простой пример показывает, что человек тоже может очень быстро обрабатывать информацию, но информацию не символь-

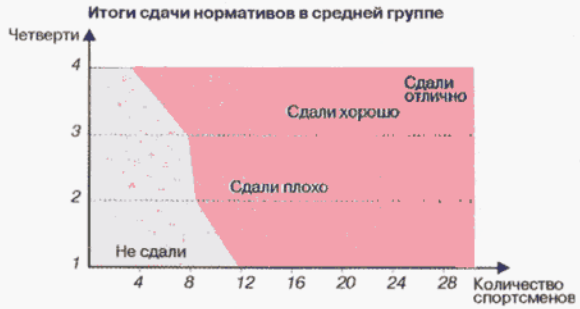


Рис. 7. Графическое представление информации

ную, а графическую и видеоинформацию. Достаточно сказать, что самому быстрому компьютеру требуется почти полгода, чтобы проанализировать всю ту информацию, которую человек получил и осознал, просто оглядевшись вокруг себя в течение одной секунды.

Впрочем, и человеку потребуется больше чем полгода, чтобы проанализировать всю ту символьную информацию, которую просматривает компьютер в течение одной секунды. И раз уж он начал претендовать на роль лучшего друга и помощника человека, то компьютеру следовало бы научиться подавать ему информацию в привычном человеческом виде. Ведь современный мир немыслим без игровых и обучающих программ, систем автоматического проектирования, баз данных фотографических и видеоизображений... Одним словом, без всего того, что мы называем компьютерной графикой.

Возьмите фломастеры, лучше всего 24 цвета, белый лист бумаги и нарисуйте хоть что-нибудь. Нарисовали? А теперь подумаем, как закодировать с помощью символьной информации то, что вы сотворили. Символьной — поскольку компьютер ничего, кроме нулей и единиц, не воспринимает.

Какие у вас появились предложения? Никаких? А между тем с компьютерной графикой мы сталкиваемся на каждом шагу. И не только на экранах компьютеров. Практически любая современная книжка готовится к печати с помощью компьютера. Многие фильмы, такие, как «Терминатор» или «Парк Юрского периода», без компьютера просто не снять...

Попробуем разобраться. Для этого возьмем увеличительное стекло и посмотрим для начала на цветной экран компьютера или телевизора, а чуть попозже — и на цветные иллюстрации в книге. В зависимости от марки видеотехники вы увидите либо множество разноцветных прямоугольничков, либо множество разноцветных кружочков. И те и другие группируются по три штуки, причем разного типа: одни из них всегда только синие (разной степени синевы — от ярчайшего синего до почти черного), другие — только красные, третьи — только зеленые. Группа из трех таких элементов образует **триаду**. Однако для каждой триады описывать ее положение на экране довольно сложно, поэтому поступают следующим образом. Экран делят на много рядов одинаковых квадратиков (такой ряд обычно называют строкой экрана), получается таблица, в которой уже легко указать положение каждого квадратика. Сам квадратик называется **пиксель** (от английского PICture'S ELement — элемент картинки). Разумеется, если ваш монитор черно-белый (или, выражаясь профессиональным языком, **монохромный**), мельчайший «элемент картинки» будет выглядеть совсем по-другому.

Количество пикселей на экране — одна из важнейших характеристик, определяющих качество изображения. Естественно при этом указывать не общее количество пикселей, а то, сколько их умеща-

ется в одной строке и сколько строк располагается на экране. Полученная характеристика называется **разрешением** данного **графического режима**. Применяются, например, CGA-разрешение, предусматривающее 320×200 пикселей на экране, VGA-разрешение, имеющее 640×480 пикселей, и т. д.

Но вернемся к цветному монитору. Как бы мы ни вглядывались в лупу, других цветов, кроме красного, зеленого и синего, на экране компьютера не различить. В чем же дело?

Те из вас, кто занимался рисованием, наверняка уже поняли, в чем дело: в смешении цветов. Действительно, все разнообразие цветов, которое мы видим на экране телевизора и компьютера, достигается смешением всего трех основных цветов. Любой серо-буро-малиновый цвет характеризуется тем, какая в нем доля красного, зеленого и синего цветов.

Давайте представим, что каждый кружочек в пикселе может либо гореть в полный накал (1), либо вообще не гореть (0). Сколько различных цветов мы сможем получить? Ответ вы найдете в таблице 8.

Таблица 8

Красный	Синий	Зеленый	Цвет
0	0	0	Черный
0	0	1	Зеленый
0	1	0	Синий
1	0	0	Красный
0	1	1	Голубой
1	0	1	Желтый
1	1	0	Пурпурный
1	1	1	Белый

А если различных типов яркости каждого элемента (или, как это называют, **градаций яркости**) будет не 2 — горит/не горит, а 64? Нетрудно сообразить, что тогда различных цветов будет $64^3 = 262\,144$.

Значит, любое видеоизображение на экране можно закодировать с помощью чисел, сообщив, сколько в каждом пикселе долей красного, сколько долей синего и сколько долей зеленого.

А теперь займемся привычным делом — расчетом количества байт, килобайт и мегабайт, необходимых для кодирования графической информации. Для этого перво-наперво нужно знать, сколько на экране пикселей. Вопрос этот непростой, и, по мере того как компьютеры становились все мощнее и мощнее, количество пикселей на мониторе увеличивалось.

Рассмотрим, к примеру, один из самых первых графических режимов. Он назывался CGA (Crayon Graphics Adaptor — графический адаптер «как цветной карандаш») и, как мы уже говорили, предусматривал 320×200 пикселей на экране. Каждый из пикселей мог светиться одним из восьми цветов — именно тех, которые указаны в таблице 8. Из нее же видно, что для кодирования этих восьми цветов нужно всего 3 нуля и / или единицы, т. е. 3 бита. Умножим теперь 3 на 320 и на 200 и получим 192 000. Вот сколько бит требуется, чтобы закодировать изображение в режиме CGA. Проведем заключительную операцию — разделим это число на 8 и получим количество байт, необходимых для хранения изображения на экране при таком разрешении — 24 000, или примерно 24 K (кстати, а почему примерно?).

Надеемся, вы теперь сможете самостоятельно рассчитать, сколько требуется памяти для хранения изображения в стандарте VGA (Video Graphics Adaptor): 640×480 пикселей. При этом в пикселе элемент каждого цвета имеет 64 градации яркости, поэтому стандарт VGA допускает те самые 262 144 цвета, количество которых мы подсчитали выше. К сожалению, из всего этого цветового разнообразия можно выбрать лишь 16 цветов, закодированных с помощью четырех бит. И для хранения такой картинки требуется 150 K памяти (проверьте, кто не верит).

Итак, кодирование шестнадцати цветов одного пикселя требует 4 бит памяти. В современных компьютерах предусмотрено одновременное использование и более широкой палитры цветов, например 65 536 цветов. Для этого требуется уже 16-битное кодирование (оно называется еще Hi-Color).

Наверняка среди вас найдутся те, кто доподлинно знает, что две разные картинки со стандартным VGA-разрешением занимают разное место на жестком или гибком диске, и уж никак не те безумные числа, которые мы насчитали.

Не будем забегать вперед (о сохранении графической информации на диске мы поговорим в § 9), а лучше посмотрим на экран монитора боковым зрением. Если он у вас не суперсовременный, вы наверняка заметите мерцание. Оно, конечно, гораздо меньше, чем на экране телевизора, но все же вполне ощутимо. Откуда оно берется?

Дело в том, что изображение на экране и телевизора, и компьютера создается тремя электронными лучами, каждый из которых отвечает за свой цвет. В считанные доли секунды лучи обегают весь экран. Где-то они почти гаснут, где-то горят в полную силу. Экран же обладает способностью гаснуть не сразу (в этом он немного похож на светосостав, которым раньше красили стрелки и цифры на светящихся часах). Все вместе это и создает иллюзию постоянного изображения.

Быть может, вы знаете, что изображение в телевизоре меняется 25 раз в секунду. На компьютере — в зависимости от его качества —

от 60 до 120 раз в секунду. Чем чаще меняется изображение, тем меньше мерцание, тем меньше устают глаза.

Откуда же лучи знают, какое менять изображение? Даже если на экране неподвижная картинка, они все равно бегают как угорелые. Так вот, информация о том, что сейчас надо вырисовывать на экране, хранится в **видеопамяти**, и именно ее размеры мы считали для различных видеорежимов.

Употребление термина «видеопамять» должно было вызвать у вас недоумение — ведь до этого речь шла только об оперативной памяти и памяти на внешних носителях. И действительно, для видеопамяти в компьютере имеется специальное устройство, называемое **видеокартой** или **графическим ускорителем**. Видеокарту можно рассматривать как самостоятельный специализированный компьютер: в нем есть и процессор, и оперативная память (та самая видеопамять, о которой шла речь), и ПЗУ с программой, управляющей работой процессора видеокарты.

Видеокарту не зря называют графическим ускорителем. Если бы выводом информации на экран занимался процессор самого компьютера, то вряд ли он мог бы делать еще что-то. Это ведь не шуточная работа — передавать по мегабайту информации чуть ли не 100 раз в секунду! Процессор современной видеокарты настолько интенсивно и жарко работает, что закрыт радиатором охлаждения, а в самых современных видеокартах даже используются вентиляторы. Этот «компьютер в компьютере» работает даже тогда, когда другие устройства компьютера простаивают — если на дисплее есть изображение, значит, видеокарта работает на полную мощность.

А теперь, когда вы так много узнали про изображение на мониторе, несколько советов владельцам домашнего компьютера:

- Прежде чем начать работу, поинтересуйтесь, с какой частотой обновляется изображение на вашем экране. Желательно, чтобы это происходило не реже 75 раз в секунду.
- Поинтересуйтесь также размером точек, из которых состоит пиксель. Чем они меньше, тем качественнее изображение. Желательно, чтобы они были не больше 0,28 мм.
- Качественный монитор с частотой обновления изображения не менее 75 раз в секунду не должен раздражать глаза при работе в текстовом редакторе черными буквами на белом фоне. Если это не так, отрегулируйте яркость и контрастность монитора.
- Нелишне знать: если расположить в поле зрения какой-нибудь ярко-красный или ярко-желтый объект и время от времени переводить на него взгляд, то это снизит утомляемость ваших глаз при долгой работе за компьютером.

ВОПРОСЫ И ЗАДАНИЯ

1. Что является пикселем в случае цветного монитора и почему он так называется?
2. Как получается ярко-белый цвет на экране цветного монитора?
3. а) Вы хотите работать с разрешением 800 × 600, используя одновременно 65 536 цветов (16-битное кодирование). В магазине продаются видеокарты с памятью 256 Кбайт, 512 Кбайт, 1 Мбайт, 2 Мбайт, 4 Мбайт. Какие карты можно покупать для вашей работы?
   б) А если вам для работы необходимо разрешение 1200 × 1600 пикселей и работа с 16 777 216 цветами (24-битное кодирование — режим True-Color; этого количества цветов достаточно для качественного воспроизведения обычной цветной фотографии), какие тогда видеокарты годятся для вашей работы?

§9. Графическое представление информации. Печать на бумаге и сохранение на диске

К этому времени у нас накопилось уже два вопроса, на которые необходимо дать ответ:

1. Как кодируется информация при печати на бумаге?
2. Почему одна и та же картинка занимает разное по величине место в видеопамяти и на магнитном диске?

Давайте по порядку. В предыдущем параграфе мы, вооружившись лупой, выяснили, что все цвета и в мониторе, и в телевизоре формируются всего тремя основными: красным, синим и зеленым. И все многообразие цветов объясняется степенью яркости каждого из них.

Посмотрите на рисунок куба (рис. 8). Каждому из цветов, которые мы видим на дисплее, соответствует точка внутри этого куба. В начале координат стоит черный цвет. На красной оси в вершине куба — ярко-красный цвет. Вершины, где смешиваются два цвета,

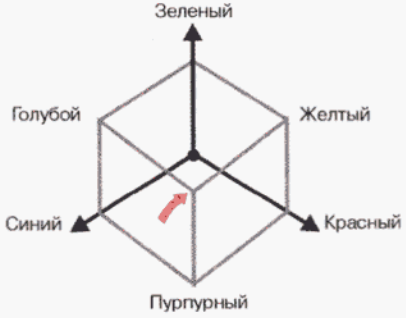


Рис. 8. Цветовой куб

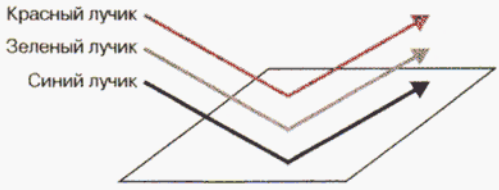


Рис. 9. Отражение лучей от листа белой бумаги

подписаны. Как вы думаете, какой цвет соответствует вершине, противоположной черной (на которую указывает цветная стрелка)?

Поскольку все цвета получаются смешением трех основных, то **модель цветопередачи** (т. е. способ получения цветного изображения), используемая в телевизоре и мониторе, называется **аддитивной** или **суммирующей**. А вариант, который мы рассмотрели, называется **RGB-кодировкой цвета** (от английского названия цветов — Red, Green, Blue).

Теперь снова вооружимся лупой и посмотрим... нет, не на тот шедевр, который вы создали фломастерами, изучая предыдущий параграф, а на цветную иллюстрацию, напечатанную в типографии.

Берем книжку с цветными картинками, рассматриваем и... Изображение распалось на мельчайшие кляксы всего лишь четырех цветов. Причем очень знакомых: черного, пурпурного, голубого и желтого. Не далее как 5 минут назад мы столкнулись с ними, разглядывая цветовой куб. Давайте порассуждаем и постараемся объяснить картину, представшую перед нами под лупой.

Итак, белый цвет можно рассматривать как смесь трех основных цветов. И когда мы видим белый лист бумаги, в наш глаз попадают лучи всех цветов, отраженные от ее поверхности, как это изображено на рисунке 9.

Стало быть, если мы видим на бумаге красный цвет, это означает, что:

1) либо из трех составляющих белого цвета, которым освещается лист бумаги, осталась только одна — красная;

2) либо краска, нанесенная на лист бумаги, отразила только красную составляющую.

Казалось бы, какая разница? Однако в этих двух случаях мы имеем дело с принципиально разными типами красок.

Первый тип, к которому и относится типографская краска, можно рассматривать как фильтр, задерживающий зеленый и синий лучики. Да и любую типографскую краску надо рассматривать как фильтр, задерживающий те или иные цвета. Нанесение типографской краски на черный лист оставит его черным.

Второй тип — это плотная отражающая краска типа гуаши или масляной. Ее можно наносить на бумагу любого цвета и получать при этом изображение необходимых цветов.

Рассмотрим более пристально типографскую краску. В случае соединения двух или более красок задерживается еще больше цветов. И если взять в качестве основных типографских красок красную, синюю и зеленую, то желтый цвет уже не получить — для него требуется две составляющие: красная и зеленая. А любая из этих трех красок не даст нам более одной составляющей. Если же соединить две из них, то получившийся «светофильтр» никаких лучей уже не пропустит и мы увидим... Все, наверно, догадались, что мы увидим. Поэтому в случае цветной печати типографскими красками изображение приходится формировать из таких красок, каждая из которых задерживает только *одну* составляющую.

Какой же цвет на бумаге поглощает красную составляющую? Конечно, голубой. Синий цвет поглощается желтым, а зеленый — пурпурным. Черный цвет мы получим, если нанесем на бумагу все три поглощающих цвета.

Взаимодействие основных цветов при печати отражается в таблице 9:

Таблица 9

Голубой (нет красного)	Желтый (нет синего)	Пурпурный (нет зеленого)	Цвет
0	0	0	Белый
0	0	1	Пурпурный
0	1	0	Желтый
1	0	0	Голубой
0	1	1	Красный
1	0	1	Синий
1	1	0	Зеленый
1	1	1	Черный

Модель цветопередачи, при которой основными являются не излучающие, а поглощающие цвета, называется **субтрактивной** или **вычитательной**.

В вычитательной модели цветовой куб как бы оказался перевернутым: отсчет в нем начинается не из прежнего начала координат, а из нового — как раз из той точки, куда направлена стрелка и которая соответствует белому цвету. Ну а кодировка, которую мы только что рассмотрели, называется **CMY-кодировкой** (по первым буквам английского названия основных цветов — Cyan, Magenta, Yellow).

Осталось объяснить, откуда же взялись черные кляксочки на цветной картинке, которую мы рассматривали. Если вы внимательно полистаете любое цветное издание, то заметите, что все же в ос-

новном оно печатается черной краской — за счет текста. Черного цвета хватает и в иллюстрациях. И получать его смешением трех основных цветов для бумаги неудобно по трем причинам:

— невозможно произвести идеально чистые пурпурные, голубые и желтые краски, и из-за этого получается не чисто черный, а темно-темно-коричневый цвет;

— на черный цвет при CMY-кодировке, сами понимаете, тратится в 3 раза больше краски;

— любые цветные чернила дороже обычных черных.

Поэтому на практике к базовому набору из трех красок добавляется — для качественной и более экономной печати — черный цвет. Такая кодировка называется **CMYK-кодировкой** (от слова blacK взяли последнюю букву, чтобы не путать с сокращением Blue). Естественно, необходимо учитывать особенности CMYK-кодировки и уметь переводить рисунки из RGB в CMYK, если вы занимаетесь подготовкой какой-либо печатной продукции.

Обсудим теперь второй вопрос: почему одна и та же иллюстрация занимает разное по величине место в видеопамяти и на магнитном диске? Как мы уже говорили, рисунок, выполненный на экране компьютера, занимает громадное количество видеопамяти, и от этого никуда не уйти — электронному лучу чуть ли не каждую сотую долю секунды нужна информация о том, как обегать экран.

Но вот вы решили записать информацию, соответствующую нашему рисунку, на жесткий или гибкий магнитный диск. Надо ли сохранять всю видеопамять? Ну, разумеется, нет. Во-первых, на рисунке практически все пиксели имеют один и тот же цвет — белый. А те, которые не белые, тоже имеют не так уж много вариантов раскраски (например, 16, если вы пользуетесь стандартом VGA). Сразу же возникает желание каким-то образом сэкономить память на диске, сообщив нечто вроде «Первые двадцать рядов пикселей — белые».

Такого сорта кодирование называется **сжатием информации**, и первыми, естественно, начали сжимать информацию разработчики программ для работы с графикой — графических редакторов. Поскольку сжатие информации — процесс творческий и далеко не очевидный, каждый из вновь создаваемых графических редакторов имел свою собственную систему сжатия и сохранения информации. А чтобы простые пользователи не запутались в этом разнообразии кодировок, к именам файлов стали приписывать расширения, по которым можно определить, в каком именно формате сохранено графическое изображение. Так, существуют расширения BMP, CDR, GIF, IFF, JPG, PCX, PIC, TIF, TGA... Впрочем, всех, наверно, и не перечислить. Как видите, с кодировкой графической информации дело обстоит отнюдь не так просто, как с символьной, где вполне хватает одного-двух стандартов, скажем ASCII и UNICODE.

ВОПРОСЫ И ЗАДАНИЯ

1. Какую модель цветопередачи (и какие краски) вы бы выбрали, если бы книги издавались на черной бумаге?
2. Можно ли использовать CMY-устройство для печати обычного черно-белого текста?
3*. Пусть в цветовом кубе (см. рис. 8) интенсивность каждого из цветов принимает значения от 0 до 1. А в системе координат вначале идет доля красного цвета, затем зеленого и, наконец, синего. Голубой цвет, таким образом, будет иметь координаты (0,1,1). Напишите формулы перехода из RGB-кодировки в CMY.
4. Почему одна и та же графическая информация имеет разный объем в видеопамяти и на магнитном диске?

§10. Компьютерная обработка графической информации

В двух предыдущих параграфах мы рассказали об особенностях представления графической информации при ее компьютерной обработке. Пора рассказать, как создается компьютерная графика. Ведь деловая графика, простейший пример которой вы видели в отчете тренера клуба «Шустрые Шиповки», становится все более необходимой. Информация, поданная в привычной человеку графической форме, позволяет быстро принимать качественные решения в самых разных областях деятельности. Даже перед операторами некоторых атомных электростанций на экране мониторов высвечиваются не только сотни цифровых показателей, но и сформированное компьютером человеческое лицо, каждый из элементов которого отражает какой-либо показатель работы станции. Скажем, поднимающиеся вверх брови говорят о повышении давления в трубопроводе. На протяжении тысячелетий человек научился почти мгновенно анализировать лицо собеседника, и даже мельчайшие изменения режима станции будут сразу же замечены.

Быть может, некоторые из вас имели дело с компьютерными обучающими программами и смогли оценить присутствие иллюстраций, выводимых на экран монитора. Согласитесь, без них изучать предмет было бы гораздо скучнее.

Компьютерная графика вовсю используется и тогда, когда различные фирмы представляют свою продукцию и рассказывают о своей работе на презентациях.

В общем, работа с графикой на компьютере все больше становится неотъемлемой частью компьютерной грамотности любого человека, и во многих объявлениях о приеме на работу, содержащих требование уметь работать на персональном компьютере, подразумевается умение работать не только с текстовыми документами, но и с компьютерной графикой.

Рис. 10. Рисунок, сделанный мышью в графическом редакторе IMAGE 72

Правда, для создания текста в вашем распоряжении имеется клавиатура с набором разнообразных символов. Что же касается графики, тут, как правило, нечего предложить, кроме манипулятора мышь. Легко понять, что им так же удобно рисовать, как и куском мела на доске. Конечно, существуют люди (и их не так уж и мало), способные даже и мышью нарисовать неплохую картину (см., например, рис. 10). Но все же обычные карандаши, фломастеры, кисточки для художников гораздо привычнее.

Чтобы одинокая мышка могла успешно справляться с компьютерной обработкой графической информации, создано много специальных программ для работы с графическим изображением на компьютере. Они делятся на несколько групп.

Графические редакторы. Незаменимы, когда требуется нарисовать или подправить рисунок.

Программы корректировки и преобразования фотографий. С их помощью можно добавить фотографии яркость или контрастность, отретушировать ее, создать те или иные эффекты (например, добиться эффекта того, что изображение находится на шаре или отчеканено на металле и т. д.).

Программы создания графиков и диаграмм по имеющимся числовым данным.

Программы, с помощью которых текст и иллюстрации объединяются в книгу, журнал, брошюру или газету. Их еще называют программами верстки.

Программы создания слайд-фильмов и мультфильмов.

Программы презентационной графики. Из названия ясно, где эти программы используются. Графическое изображение и звуковое сопровождение, объединенное с их помощью, просто незаменимы для наглядной иллюстрации любого выступления.

Конечно, возможны самые различные варианты объединения вышеназванных программ и друг с другом, и с текстовыми редакторами, и с электронными таблицами, и с базами данных, о которых речь пойдет чуть позже.

Свое знакомство с программами обработки графической информации вы начнете с изучения графического редактора. Отметим, что принципиально графические редакторы, как правило, не очень-то отличаются друг от друга; поэтому, освоив один, вы быстро разберетесь, как работать и с любым другим. То же самое можно сказать и про остальные виды программ. Поэтому не надо думать, что вы будете напрасно терять время, изучая относительно простые программные средства в следующих параграфах. Ведь и читать в первом классе вы учились не на романе Л. Н. Толстого «Война и мир», а на совсем простых текстах типа «Мама мыла раму».

ВОПРОСЫ И ЗАДАНИЯ

1. Какие имеются средства для работы с компьютерной графикой?
2. Вспомните, с помощью каких периферийных устройств можно преобразовать рисунок на бумаге в изображение на экране компьютера.

§11. Графический редактор. Общее описание

Те из вас, кто внимательно читал предыдущий параграф, не могли не задаться вопросом, зачем вообще нужен **графический редактор**, если рисовать в нем не очень-то удобно. Не проще ли всегда и во всех случаях пользоваться сканером, переводя в электронный вид рисунки, подготовленные самыми обычными карандашами, фломастерами, красками и кисточками?

Давайте представим себе художника, который долго трудился над каким-нибудь рисунком, полностью закончил его масляными красками, а на следующее утро с ужасом понял, что главного персонажа надо нарисовать заново. После такого решения почти наверняка не один час уйдет только на то, чтобы аккуратно его замазать, а уж потом рисовать нового. А если рисунок сделан акварелью, так и вообще придется переделывать все с самого начала. Вот именно поэтому так много времени уходит у художников на карандашные эскизы, которые легко править.

Но ведь ни по какому эскизу невозможно до конца понять, как будет выглядеть полноцветная картина. А в графическом редакторе можно запросто исправлять цветной готовый рисунок. И уже ради

Рис 11. Основные инструменты графического редактора

одного этого свойства он интересен. Конечно, речь не идет о том, чтобы создавать шедевры живописи, используя компьютер и графический редактор (хотя, быть может, и здесь они бы пригодились, например, для отработки композиции). Мы говорим прежде всего о подготовке деловых документов и видеоматериалов. Вот для их оформления графический редактор весьма полезен. При этом легкость корректирования только одно из многих его замечательных свойств.

Посмотрите на рисунок 11. На нем показаны основные компоненты графического редактора, которые превращают его в незаменимого помощника художника-оформителя.

Со всем, что изображено на рисунке, вы неоднократно встретитесь на практике, а пока ограничимся кратким описанием.

Стандартные инструменты

Стандартными эти инструменты называются потому, что позволяют делать то, что человек обычно делает с помощью карандаша, линейки, циркуля, ластика и тому подобных привычных инструментов. При этом нет никакой необходимости рисовать черно-белые эскизы. Всего одним нажатием на клавишу мыши можно выбрать любой цвет, представленный в палитре редактора. Если же подходящего цвета нет, его несложно подобрать, регулируя доли красного, синего и зеленого цветов (вспомнили, почему это так?).

Место на экране, где будет возникать очередной элемент рисунка, указывает **графический курсор**. Он может выглядеть по-разному:

в виде стрелки, крестика или точки. Курсор можно перемещать по экрану с помощью мыши (иногда клавиш со стрелками). При этом в специально отведенном месте экрана можно увидеть два изменяющихся числа. Это **координаты курсора**. Они показывают, на каком пикселе находится курсор. (Надеемся, вы еще помните, что такое пиксель на экране монитора.)

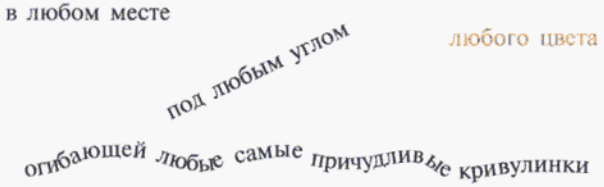
А теперь представим вам стандартные инструменты и возможности, которыми обладает практически любой графический редактор:

- ***Карандаш*** — смысл ясен из названия. При нажатой (обычно левой) клавише мыши курсор оставляет за собой линию заранее выбранного цвета.
- ***Отрезок*** — позволяет провести прямую линию. Отметьте мышкой начальную точку, растяните прямую линию до конечной точки и снова нажмите левую клавишу мыши.
- ***Прямоугольник*** — позволяет рисовать прямоугольники выбранного цвета любых размеров со сторонами, параллельными краям экрана. Выбрав этот инструмент, установите курсор в одну из вершин будущего прямоугольника и зафиксируйте ее, нажав нужную клавишу мыши (какую именно, указано в инструкции к конкретному графическому редактору; вам эту информацию сообщит учитель). Затем, сдвигая мышь, выберите нужный размер прямоугольника и зафиксируйте его.
- ***Овал*** — еще одна из стандартных фигур. Выбрав этот инструмент, поставьте курсор в центр будущего овала и зафиксируйте его (в соответствии с инструкцией или указаниями учителя). Затем, сдвигая мышь, выберите нужный размер овала и зафиксируйте его. Этим же инструментом легко рисуются и окружности.
- ***Выбор толщины линии*** — это необходимо не только для карандаша, но и при рисовании стандартных фигур.
- ***Резинка-ластик*** — вы можете выбрать размер резинки, а затем стирать ненужные фрагменты рисунка.
- ***Разбрызгиватель*** — разбрызгивает краску (как будто из аэрозольного баллончика).
- ***Заливка*** — заливает выбранным цветом часть рисунка, ограниченную замкнутым (необязательно одноцветным) контуром. Будьте осторожны! Если в контуре есть дырочка хотя бы в один пиксель, краска разольется по всему рисунку. К счастью, беду можно поправить кнопкой «Откатка», о которой мы расскажем ниже.
- ***Узор*** — залить можно не только сплошным цветом, но и некоторым рисунком. Можно, например, сделать заливку красными кирпичами или фиолетовыми цветочками.

- ***Редактирование узора*** — стандартных узоров довольно много. Если они вас они не устраивают, создайте свои.
- ***Лупа*** — после выбора этого инструмента появится небольшой прямоугольник. Наложите его на часть рисунка, которую хотелось бы рассмотреть получше, т. е. увидеть, какой цвет имеет каждый пиксель. Это особенно важно проделать перед заливкой. С помощью лупы легко отыскать малейшие дырочки в контуре.
- ***Откатка*** (иногда этот инструмент называют ***Отмена***) — дает возможность отменить только что выполненное действие, вернувшись к предыдущему изображению. Например, с его помощью можно тут же отменить заливку, если вам не понравился узор, которым вы закрасили нарисованный воздушный шарик.
- ***Файл*** — в этом разделе содержатся команды загрузки уже готовых рисунков и сохранение того, над которым вы работаете.

Работа с текстом

Обычно в графическом редакторе вы можете использовать все то многообразие шрифтов, которое имеется и в редакторе текстов. Кроме того, надпись в графическом редакторе очень легко сделать:



Понятно, что ни один оформитель, владеющий навыками работы на компьютере, не будет выполнять надписи вручную, если речь не идет о каких-либо совсем уж нестандартных шрифтах.

Спецэффекты

Спецэффекты графического редактора позволяют перевернуть фрагмент картинки, перекосить его, зеркально отразить, оставить от него только тень, изменить контур, получить копию объекта... Одним словом, изменить иногда до неузнаваемости картинку или ее фрагмент. С их помощью можно, например, нарисовав всего одно лишь деревце, создать за пару минут целую рощу. Результат применения некоторых эффектов к исходному рисунку приведен на рисунке 12.

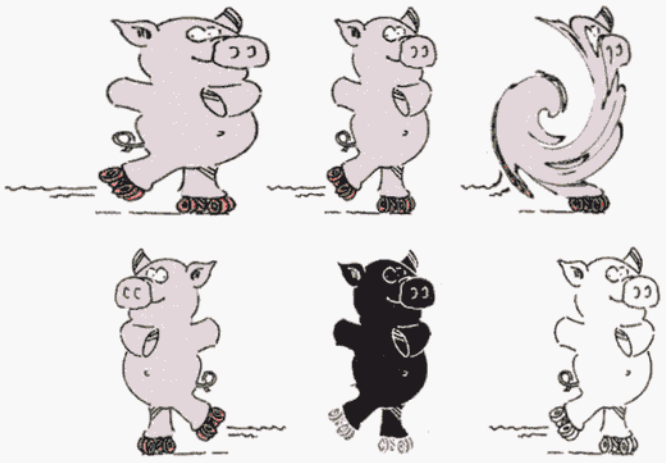


Рис. 12. Пример применения различных эффектов к исходному рисунку (центральный вверху)

Библиотека рисунков

Любой современный графический редактор снабжен обширной библиотекой рисунков (либо сразу в комплекте, либо за отдельную плату). Кроме того, многие фирмы занимаются тем, что разрабатывают и продают рисунки самой разнообразной тематики для графических редакторов.

Похоже, нет ни одного предмета в жизни, рисунка которого не было бы в графических библиотеках. Здесь и самые разнообразные здания, и множество видов животных и растений (как живых, так и вымерших), и все инструменты, которыми пользуется человек, начиная от швейной иголки и кончая громадными экскаваторами. О компьютерах и их деталях уж и говорить не приходится. В графических библиотеках можно найти изображение (карикатурное, схематическое, почти натуральное) любого известного спортсмена, артиста, политического деятеля...

Объединение рисунков

Графический редактор, как и текстовый, позволяет с помощью буфера обмена легко переносить фрагменты из одних рисунков в другие, компонуя новые изображения из старых. Умело это используя и не забывая о спецэффектах и библиотеках стандартных рисунков, можно создавать громадное количество оригинальных рисунков, даже совершенно не умея рисовать.

ВОПРОСЫ И ЗАДАНИЯ

1. Для чего может быть полезен графический редактор?
2. Что такое графический курсор?
3. Какие простейшие фигуры позволяют рисовать инструменты графического редактора?
4. Что представляет собой режим «Заливка»?
5. Для чего используют буфер обмена?
6. Что позволяет делать режим «Лупа»?
7. Можно ли в графическом редакторе выполнять текстовые надписи?
8. Какой режим позволяет вернуться к прежнему варианту рисунка, если результат очередного действия, выполненного в графическом редакторе, вам не понравился?
9. Назовите наиболее запомнившиеся вам спецэффекты графического редактора (из тех, которые перечислены в тексте параграфа).

Стандартные инструменты графического редактора

Прежде всего договоримся о терминологии.

ЩЕЛКНУТЬ клавишей мыши — быстро нажать и отпустить клавишу.

ПРИЖАТЬ клавишу мыши — держать, не отпуская (например, для растягивания изображения).

ЛЕВАЯ КЛАВИША мыши обычно используется для подтверждения выбора и фиксации объекта. Эта клавиша означает «ДА», она употребляется наиболее часто.

ПРАВАЯ КЛАВИША используется для отказа от чего-либо (это клавиша «НЕТ»), а в некоторых графических редакторах и для изменения размеров изображения.

ПИКТОГРАММА — схематичное изображение объекта или действия над объектом.

ВЫБОР ИНСТРУМЕНТА или РЕЖИМА, как правило, осуществляется установкой курсора на пиктограмме этого инструмента или режима, после чего надо щелкнуть левой клавишей мыши.

ПАЛИТРА — поле выбора цвета, располагающееся обычно снизу или сбоку экрана.

Напомним, как работать со стандартными инструментами графического редактора.

Отрезок:

- выберите этот инструмент;
- переведите графический курсор на палитру и выберите нужный цвет;

- отметьте один конец отрезка (для этого обычно устанавливают курсор в нужной точке и щелкают/нажимают нужную клавишу мыши);
- растяните отрезок до нужных размеров и отметьте его второй конец (щелкнув/отпустив нужную клавишу).

Прямоугольник:

- выберите этот инструмент;
- переведите графический курсор на палитру и выберите нужный цвет;
- переведите графический курсор на поле для рисования и установите его в точке, где будет располагаться одна из вершин прямоугольника;
- растяните прямоугольник до нужного размера;
- зафиксируйте прямоугольник на экране.

Овал:

- выберите этот инструмент;
- переведите графический курсор на палитру и выберите нужный цвет;
- переведите графический курсор на поле для рисования и установите его в точке, где будет располагаться центр овала;
- растяните овал до нужной формы и желаемого размера;
- зафиксируйте овал на экране.

Резинка-ластик:

- выберите этот инструмент;
- переведите графический курсор на поле для рисования и установите нужный размер прямоугольника, изображающего резинку;
- перемещая мышью прямоугольник-резинку, сотрите ненужную часть рисунка.

Заливка:

- выберите этот инструмент;
- переведите графический курсор на палитру и выберите нужный *узор*;
- переведите графический курсор на поле для рисования и установите его внутри той части рисунка, которую вы намерены закрасить (говорят еще *залить*) выбранным узором;
- подтвердите свой выбор.

Если при заливке краска разлилась по всему полю, то щелкните на пиктограмме ***Откатка***. Применив откатку, просмотрите с помощью *лупы* тот контур, внутри которого вы так неудачно выполнили заливку.

Лупа:

- выберите этот инструмент;
- перемещая мышью прямоугольник-лупу, установите ее на нужной части рисунка;

- подтвердите свой выбор;
- переведите курсор на палитру и выберите нужный цвет;
- переведите курсор на поле для рисования и установите его в квадратике, изображающем нужный вам пиксель;
- нажав соответствующую клавишу, либо закрасьте его выбранным цветом, либо, наоборот, отмените закраску.

Этот инструмент позволяет сделать особо тонкую корректировку вашего рисунка, в частности зарисовать все дырки в контуре, ограничивающем область экрана, которую вы намереваетесь целиком закрасить в режиме *Заливка*.

Карандаш и ***распылитель*** особых комментариев не требуют — выберите по очереди каждую из пиктограмм и попробуйте эти инструменты в деле.

Соблюдайте «правила хорошего тона» при работе с графическим редактором:

- Если вы случайно попали куда-нибудь не туда, подзовите учителя, и он поможет вам выбраться из сложной ситуации. Не пытайтесь щелкать клавишей мыши по всем подряд пиктограммам или панически нажимать клавиши на клавиатуре (тем более сразу несколько одновременно).
- Не выбирайте инструменты сканер и принтер, не получив предварительно разрешение учителя. Возможно, к вашему компьютеру не подключено ни то ни другое. Это может привести к зависанию машины, т. е. отказу компьютера работать с вами дальше.
- Не записывайте ничего на диск без разрешения учителя.
- Не заливайте все поле яркой краской — не будет заметен курсор!

А теперь выполните несколько заданий:

❶ Перемещая курсор, попробуйте выяснить, где находится начало системы координат и куда направлены оси.

❷ Изобразите, пожалуйста, домик. Используйте для этого инструменты ***прямоугольник*** и ***отрезок***.

❸ Пусть у этого домика будет труба, из которой идет дым, а рядом растет дерево (ель или тополь). Используйте инструменты ***распылитель*** и ***овал***.

❹ Раскрасьте дом и трубу, используя инструмент ***узор*** (для крыши — *черепицу*, для дома и трубы — *кирпичи*).

Работа с палитрой

Строим сказочный город

Выполните задания:

❶ Вы уже научились раскрашивать рисунки готовыми красками. Но настоящие художники любят смешивать краски. Мы уже рассказали, что новый цвет можно создать, смешивая три краски — красную (Red), зеленую (Green) и синюю (Blue). Следуя указаниям учителя или инструкции к вашему графическому редактору, создайте несколько новых цветов и подготовьте новую палитру.

❷ Вы уже довольно опытный художник, поэтому, используя вашу палитру, нарисуйте еще раз такой же дом, как и на предыдущей лабораторной работе. Но внимание! Дом не должен быть слишком большим или слишком маленьким: он должен занимать приблизительно $\frac{1}{3}$ экрана по высоте и $\frac{1}{3}$ по ширине. Это нужно для выполнения последующих заданий.

❸ Постройте сказочный город, в котором все дома как будто похожи и в то же время не похожи друг на друга, а вернее, на ваш исходный дом. Для этого поместите изображение дома в ***буфер обмена***. Как это сделать конкретно в вашем графическом редакторе, вам расскажет учитель или вы можете прочитать в инструкции пользователю. Затем щелкните нужной клавишей мыши на пиктограмме вставки буфера обмена (или выберите соответствующий пункт в меню «Редактировать») и «наклейте» домик, изменяя, если хотите, его размеры. Если дом сделать длинным и узким, то он станет похожим на башню. Если растянуть вширь, он скорее будет похож на амбар. Экспериментируйте!

❹ А теперь строим целую улицу домов, пользуясь правилами перспективы. У вас может получиться и средневековый город, и деревня, и поселение папуасов.

❺ Нарисуйте дерево. Выполняя действия в соответствии с заданием 4, украсьте ваш город деревьями, изменяя их размер и пользуясь правилами перспективы.

❻ В вашем сказочном городе пошел сказочный снег... Но у вас нет такой «снежной» заливки. Это новый ***узор***. Как его создать, вам расскажет учитель или вы узнаете из инструкции пользователю. Как только узор «снег» готов, выберите инструмент ***заливка***, подберите нужный цвет и устройте в своем городе снегопад. Можно выбрать, например, сиреневый цвет, тогда ваш снег будет совсем необычным.

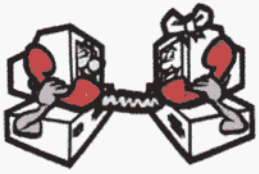
Спецэффекты графического редактора

Ралли в пустыне, или Жестокий спорт

Вы уже освоили работу с ***буфером обмена*** и умеете «вклеивать» его содержимое в любое место рисунка. Но этим отнюдь не ограничиваются возможности графического редактора. Их дальнейшему освоению и посвящена предлагаемая лабораторная работа. Для успешной работы вы должны знать из рассказа учителя или инструкции пользователю, как загрузить (считать) рисунок с диска на экран.

А теперь выполните следующие задания:

1. Используя режим ***файл***, загрузите рисунок «Джипа» (Jeep) или какого-нибудь другого автомобиля и с помощью вклейки создайте целую шеренгу из автомашин, уходящую вдаль. Надеемся, вы будете соблюдать при этом закон перспективы. (Вспомните предыдущую лабораторную работу, в частности задание 4.)
2. Теперь представьте, что машина на переднем плане попала в ужасную катастрофу. Какие-то ее части отлетели, какие-то сплющились. Используя эффект ***переноса*** и эффект ***перекоса***, создайте соответствующий рисунок.
3. В дополнение переверните одну из машин в шеренге, используя инструмент ***вертикальный переворот***.
4. Легко предположить, что организованное вами жестокое ралли далеко не все могут выдержать. С помощью инструмента ***зеркало*** или ***горизонтальный переворот*** разверните пару машин в другую сторону.
5. Две машины столкнулись лоб в лоб, так что их передние колеса поднялись. Используйте инструменты ***вращение*** и ***зеркало***.
6. А тем временем там, где идут наши соревнования, наступил вечер. Воспользуйтесь эффектом ***тени*** и измените контуры автомашин.
7. Напишите в правом верхнем углу рисунка слово «Rally» и примените к буквам все оставшиеся в вашем распоряжении спецэффекты. Изумительно смотрятся буквы с ***расслоенным контуром***, особенно если потом внутренний контур закрасить другим цветом. Не забудьте, что текст тоже можно ***наклонять***, ***поворачивать***...



2 Компьютерные телекоммуникации

В предыдущей главе вы познакомились с основными инструментами, позволяющими обрабатывать информацию с помощью компьютера. По-другому можно сказать, что вы осваивали основы **информационных компьютерных технологий**. Но представленная картина будет неполной, если не коснуться проблем *передачи* информации. Ведь именно на возможности передачи знаний от одного человека другому основывается прогресс человечества в целом и просто каждого человека.

Конечно, процессы, связанные с накапливанием, сохранением и передачей информации, присущи всему живому. Но именно человек начал целенаправленно создавать средства, специально предназначенные для обеспечения этих процессов, причем буквально с момента своего появления на Земле — ведь обмениваться информацией и накапливать ее стали еще первобытные люди в ходе совместного труда и охоты.

Первой информационной революцией было изобретение письменности. Именно письмо превратило информацию из ценности сиюминутной, разовой в ценность, которую можно без искажений передавать из поколения в поколение. Устный пересказ, которым пользовались для передачи информации в дописьменный период, опирался на такое ненадежное устройство хранения информации, как человеческая память.

Второй информационной революцией по праву считают изобретение книгопечатания (рис. 13). Ведь теперь накопленная человечеством информация, представленная в виде текстов, становилась доступной каждому грамотному человеку. Да и само обучение грамоте обретало массовый характер, поскольку появилась возможность создать и издать учебник для обучения не единиц людей, а тысяч людей.

Во второй половине XX в. выпуск научно-технической печатной продукции стал подобен все нарастающей лавине. Ни отдельно взя-

Рис. 13. Первый печатный станок и «Азбука» Федорова

тый человек, ни коллективы, ни специальные организации, созданные для обработки поступающей информации, не только не могли освоить весь информационный поток, но и оказались неспособными оперативно находить в нем то, что требовалось для осуществления тех или иных работ. Сложилась парадоксальная ситуация, когда для получения нужной информации легче и дешевле было провести исследования заново, чем разыскать ее в научной литературе. Информационная система, основанная на бумажных носителях информации, переросла свои возможности. Назрел кризис этой системы.

И трудно сейчас сказать, как долго еще компьютеры оставались бы просто мощными вычислительными комплексами, если бы разразившийся кризис не заставил осознать, что именно компьютеры способны помочь человеку преодолеть его. Правда, одних компьютеров здесь недостаточно — нужны еще средства коммуникаций, способные доставить информацию от одного компьютера к другому. Обо всем этом и пойдет речь в данной главе.

§12. Что такое компьютерная сеть

Изучая информационные технологии, вы довольно много занимались тем, что создавали и редактировали различные электронные документы. Это и графические изображения, и тексты, и электронные таблицы. Видимо, ни у кого не возникает сомнения, что электронные документы нужны не только их создателю. Отчет, деловое письмо или расчеты, произведенные на компьютере, как правило, необходимы и другим людям. Как же их передать? Конечно, можно напечатать электронный документ на бумаге и переслать его обычной почтой. Можно вывести его на видеопленку или слайды и в дальнейшем работать с ними. Но, утратив электронный вид, созданный вами документ теряет и чрезвычайно полезные свойства легкого редактирования и использования его при необходимости в других документах.

Итак, возникает проблема передачи документов в электронном виде с одного компьютера на другой. Самое простое решение — скопировать файл с документом на дискету (или несколько дискет) и перенести его на другой компьютер. На протяжении многих лет это и был наиболее популярный и доступный путь передачи электронных документов. Но дискета не самое удобное средство передачи информации, особенно если информации много или приходится пересылать дискету почтой, которая быстротой не отличается. И уж конечно никакие дискеты не помогут, если мы хотим, чтобы группа людей одновременно работала с одним и тем же документом или с одной и той же базой данных. А в современном мире это требуется постоянно. Достаточно заглянуть в крупный мага-

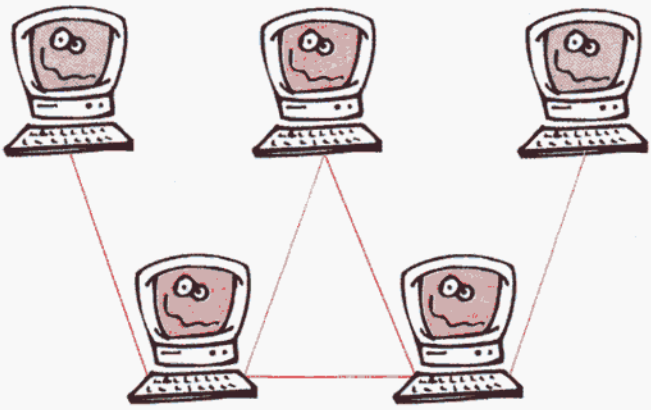


Рис. 14. Локальная компьютерная сеть

зин и посмотреть, как кассиры используют банковскую базу данных, проводя расчеты с покупателями по электронным кредитным карточкам.

Если компьютеры, между которыми надо организовать обмен документами, находятся недалеко друг от друга, их можно соединить, воспользовавшись либо кабелем, очень похожим на телевизионный, либо **оптоволоконным кабелем**, либо специальным проводом, называемым **витая пара**. Потребуются еще и специальные **сетевые платы**, и соответствующее программное обеспечение. Такое соединение называется **локальной компьютерной сетью** (рис. 14).

Основная цель создания локальной компьютерной сети — совместное использование информационных ресурсов и ресурсов компьютерной техники: памяти, процессоров, принтеров, сканеров и т. д.

Ведь данными, помещенными в память одного компьютера, можно теперь воспользоваться с любого другого, и его память уже не потребуется для размещения той же самой информации. Точно так же не потребуется теперь иметь и запускать автономно на каждом компьютере одну и ту же программу — к ней можно обратиться с любого компьютера сети на тот один компьютер, где она есть. Да и не только в экономии ресурсов дело. Может ли работать, скажем, банк без локальной сети его компьютеров? Конечно нет! Ведь сведения об операциях с клиентами банка, проводимыми в разных его подразделениях, тем не менее сразу должны быть отражены в единой базе данных, иначе возникнет полная неразбериха. Но невозможно же после каждой операции с клиентом, проведенной на компьютере служащего банка, записывать

Рис. 15. Схема связи двух компьютеров по телефонной линии

изменения на дискету и бежать с ней к главному компьютеру. Вместо работы служащие то и дело стояли бы в очереди у его дисковода.

А что делать, если необходимо подключить к тому же банковскому компьютеру далекий магазин? Неужели тянуть отдельный кабель? Давайте подумаем, нет ли какой-нибудь коммуникационной сети, доступной буквально каждому? Ну конечно же есть! Это телефонная сеть, которая более чем за 100 лет своего развития дотянулась практически до всех уголков Земли. Но телефонная сеть, хотя бы по причине своей древней природы, не приспособлена к тому, чтобы компьютеры пользовались ею напрямую. Необходимо специальное устройство, которое преобразует их цифровой язык в телефонные сигналы и наоборот.

Преобразование цифровых сигналов в телефонные называется **модуляцией**, а телефонных в цифровые — **демодуляцией**. Поэтому соответствующие устройства называют **модемами** (модуляция — демодуляция). Схема связи двух компьютеров по телефонной линии приведена на рисунке 15.

Внимательно приглядевшись к этой схеме, вы наверняка поймете, что оба компьютера в момент приема-передачи сообщения должны не просто быть включенными, но и работать с программой приема-посылки сообщений. Это не очень-то удобно. Нужно специально договариваться, одновременно запускать одну и ту же программу, ждать, пока закончится процесс передачи, повторять передачу, если произошел обрыв связи... Одним словом, прямое использование телефонных линий для связи двух компьютеров, особенно если они находятся в разных городах, вряд ли способно доставить удовольствие. И уж конечно не подходит для того, чтобы русский мальчик Ваня черкнул на компьютере своему заокеанскому другу Джонни пару-другую приветственных строчек.

Выходом явилось создание глобальных компьютерных сетей, буквально перевернувших наше представление об обмене информацией. Для того чтобы понять их принцип действия, забудем на время о компьютерах, модемах, телефонных линиях и заглянем на обычную почту. Здесь вот уже более столетия предлагают такую услугу, как абонентский ящик. Это означает, что вся ваша корреспонден-

ция будет аккуратно складываться в специальный почтовый ящик, находящийся прямо в здании почты, а не доставляться вам на дом. Вы же забираете корреспонденцию в удобное для вас время. Обычно абонентскими ящиками пользуются люди и организации, которые ведут большую переписку и почтовый обмен. Обратите хотя бы внимание на почтовые адреса газет, журналов, телевидения, многих фирм. На принципах абонентских почтовых ящиков и построена так называемая **электронная почта** или **E-mail**. Роль почтовых отделений в ней играют мощные круглосуточно работающие компьютеры, которые находятся во многих городах и соединены между собой не только обычными телефонными проводами, но и специально проложенными кабелями для цифровой связи и даже спутниковыми каналами связи (рис. 16). Такая система и называется **глобальной компьютерной сетью**.

Для того чтобы воспользоваться услугами электронной почты, необходимо заказать на одном из таких компьютеров (их еще называют **серверами**) абонентский ящик, куда и будет приходить вся адресованная вам электронная корреспонденция. Естественно, физически ваш ящик представляет собой область на жестком диске (т. е. «винчестере»), которая освобождается по мере того, как вы свою корреспонденцию забираете. Поскольку серверы компьютерной сети работают круглосуточно, нет никакой необходимости договариваться с кем-либо о приеме сообщений. Достаточно связаться по модему в удобное для вас время с сервером, получить свою почту и послать электронные документы.

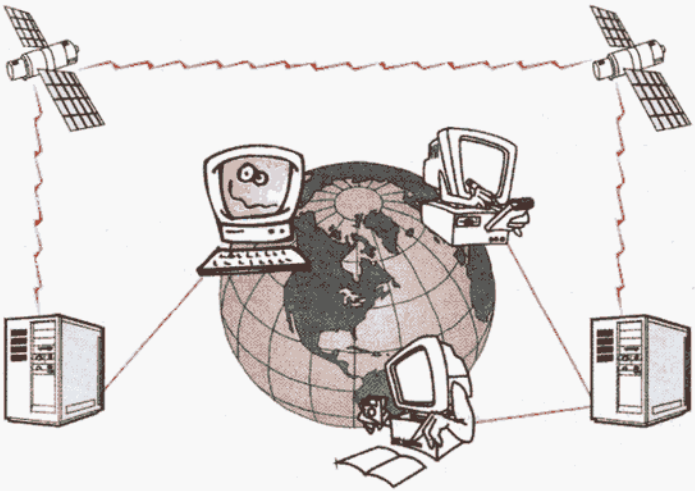


Рис. 16. Глобальная компьютерная сеть

Теперь вернемся к вопросу об адресе. Адресуя обычное письмо, необходимо указать страну, город, почтовое отделение, номер абонентского ящика. Первые три реквизита определяют не столько географическое положение получателя, сколько почтовые службы, которые займутся вашим письмом. Поначалу его получит почтовое ведомство, занимающееся международными отправлениями, затем оно попадет на общегородской почтамт, сортирующий корреспонденцию по отделениям связи внутри города, и лишь затем оно окажется на ближайшей к вам почте и его положат в ваш ящик. Такой принцип построения службы называется *иерархическим*. По иерархическому принципу построена и электронная почта. Но эта иерархия весьма отдаленно связана с географией.

Во-первых, глобальные сети изначально создавались отнюдь не для того, чтобы Ваня мог написать Джонни, а для стратегического управления вооруженными силами и крупными транснациональными компаниями. При таком назначении сети совершенно неважно, в какой стране находится абонент. Важно лишь то, какой сервер его обслуживает.

Во-вторых, как вы, наверно, уже поняли, глобальных компьютерных сетей много, созданы они на основе различного оборудования и программного обеспечения, каждая использует свою систему кодирования и пересылки информации, которая называется **протоколом информационного обмена**. Потребовались громадные усилия, для того чтобы объединить почти все глобальные сети в единое целое, называемое сейчас Интернетом.

Об Интернете мы подробнее поговорим в последующих параграфах этой главы. А сейчас продолжим обсуждение того, как адресуются электронные письма. Ясно, что прежде всего мы должны сообщить компьютеру, в какую глобальную сеть необходимо передать сообщение, во-вторых, какому серверу в этой сети, в-третьих, должен ли этот сервер отправить сообщение следующему, «менее главному» серверу, в-четвертых... — одним словом, указать все иерархические уровни абонента, совсем как и в случае обычного почтового отправления.

Этих иерархических уровней в электронной почте не так уж и много, как правило 3 – 5, и **электронный адрес** выглядит гораздо короче адреса обыкновенного. Например: Billt@tenet.edu, или Shaynes@monroe.lib.mi.us, или 2170@dialup.mplik.ru. Слева от значка @, как правило, указывается **идентификатор конечного пользователя**, т. е. присвоенный вам позывной на том сервере, где вы заказали «абонентский ящик».

ВОПРОСЫ И ЗАДАНИЯ

1. Для чего создаются локальные компьютерные сети?
2. Почему для связи компьютеров с помощью телефонной сети нужен модем?
3. Что такое идентификатор конечного пользователя?

4. Что такое протокол информационного обмена?
5. Для модема важной характеристикой является **скорость передачи информации**, которая показывает, сообщение какого информационного объема может быть передано за единицу времени. Скорость передачи информации измеряют в **бодах**:

$$1 \text{ бод} = 1 \text{ бит/с}.$$

В настоящее время применяются модемы, имеющие скорость передачи информации 14 Кбод, 28 Кбод и 56 Кбод.

а) Сколько времени требуется для передачи модемом в компьютерную сеть одной страницы текста данного учебника? Дайте ответ для каждого из трех указанных значений скорости передачи информации. Решить эту задачу вам поможет ответ к задаче 2 из § 3.

б) Каков информационный объем сообщения (в байтах), если для его приема потребовалось 1,5 с работы модема? Дайте ответ для каждого из трех указанных выше значений скорости передачи информации.

§13. Интернет — хранилище информации

Появление и ошеломляюще стремительное распространение Интернета является едва ли не самым заметным событием в современном компьютерном мире. И оно вовсе не ограничено рамками компьютерного мира — воздействие Интернета на общество намного шире. Без преувеличения можно сказать, что он влияет на судьбу человеческого сообщества. Причина в том, что глобальные компьютерные сети стали не только средством оперативного обмена информацией, но и огромным, к тому же чрезвычайно мобильным, хранилищем самой разнообразной информации. Объединение же глобальных сетей в Интернет знаменует собой **третью информационную революцию** (о двух первых говорилось в начале этой главы), когда практически вся накопленная человечеством информация оказалась переведенной на электронные носители, а мощные компьютерные станции, объединенные в глобальные сети и снабженные эффективными средствами поиска информации, способны оперативно доставлять эту информацию пользователю из любого уголка планеты.

Напомним, что в каждой компьютерной сети действует протокол информационного обмена, определяющий систему кодирования и пересылки информации в данной сети. Один из таких протоколов, названный сокращенно **TCP/IP**, и стали вводить у себя многие глобальные сети, чтобы обеспечить их взаимодействие. Аббревиатура TCP/IP расшифровывается как Transmission Control Protocol / Internet Protocol (т. е. протокол управления передачей / протокол Интернет). Его разработку связывают с именем Винтона Серфа (Vinton Cerf), которого сейчас нередко называют «отцом Интернета». Собственно говоря, **Интернет** — *это объединение глобальных сетей, поддерживающих протокол* TCP/IP.

К началу 1998 г. число компьютеров в Интернете достигло почти 30 млн. Винтон Серф считает, что к концу века это число увеличится еще в 10 раз. Рост числа пользователей, как и использование не только текстовых видов информации (вспомните-ка, о каких еще видах компьютерной информации шла речь в предыдущей главе?), резко увеличили нагрузку на Интернет. Сегодня в развитие Интернета вкладываются сотни миллионов долларов, однако его быстродействие снижается. Перед специалистами стоит задача найти такой способ построения архитектуры Интернета, который смог бы воплотить новейшие достижения сетевых технологий.

ВОПРОСЫ И ЗАДАНИЯ

1. Что такое Интернет?
2. Почему появление Интернета называют третьей информационной революцией?

§14. Что такое гипертекст

Как вы помните, электронные документы — это не только странички текста, но и рисунки, и базы данных, и электронные таблицы, и файлы с мультипликацией, и даже звуки. Представьте себе рекламный буклет, полный не только текста и рисунков, но и мультфильмов со звуками!

А чтобы такой электронный буклет можно было удобно листать, воспользовались довольно старой идеей **гипертекста**. По сути с той же идеей вы имеете дело, когда работаете с электронной таблицей. Гипертекстовая страничка, как и электронная таблица, тоже состоит из видимого и «подвального» этажей. На видимом располагаются самый обычный текст и самые обычные рисунки. Но только отдельные слова и рисунки особо выделены, и курсор мыши, оказавшись на них, принимает другую форму. Это означает, что под этими элементами страницы находятся ссылки на другие электронные документы или даже на целые программы, которые будут выполнены в случае нажатия на кнопку мыши при видоизмененном курсоре.

Что же располагается в «подвале», какие бывают ссылки?

- Самое простое — переход к другим страницам или к другому документу.
- Тоже не очень сложно — показ рисунка или мультфильма, прослушивание звукового фрагмента.
- Немного сложнее (с точки зрения аппаратуры, но не с точки зрения того, кто просматривает гипертекстовый документ) — переадресация пользователя к другому гипертекстовому документу на другом сервере, расположенном, быть может, за тысячи километров.
- Возможна, как уже говорилось, активизация специальных программ. Чаще всего это программы пересылки файлов. Так, практи-

чески все мировые производители комплектующих для компьютеров совершенно бесплатно распространяют новейшее программное обеспечение, необходимое для их работы (такие программы называются *драйверами*).

Систему гипертекстовых страниц, расположенную на серверах, позволяющих с легкостью странствовать по всем закоулкам глобальных компьютерных сетей, назвали **WWW** (World Wide Web) — **Всемирной паутиной**. Сами же страницы этой системы называют **Web-страницами**.

Для создания гипертекстовых страниц можно воспользоваться специальным **гипертекстовым редактором**. Таких редакторов существует довольно много, и у каждого свой стандарт форматов страниц. Понятно, что было необходимо выбрать один из них для обеспечения всем желающим беспрепятственного путешествия по электронной паутине. Таким стандартом стал **HTML-стандарт**. Сама аббревиатура HTML происходит от HyperText Markup Language (в переводе — язык разметки гипертекста).

Слово «разметка» употреблено здесь совсем не случайно. Дело в том, что словами этого языка описывается гипертекстовая структура документа: какой текст на какой странице и как разместить, из какого файла взять рисунок, каким кеглем напечатать текст и т. д. Эти управляющие слова HTML называют **тегами**. В тексте каждый тег заключен в угловые скобки.

Приведем пример текста, содержащего гипертекстовую разметку:

```
<HTML>
        <HEAD>
                <TITLE>
                        Моя страничка
                </TITLE>
        </HEAD>
<BODY BACKGROUND=«file_name»>
<CENTER>
        <H1> Привет всем, кто пришел на мою страничку! </H1>
<BR>
<IMG SRC=«file_name1»>
</CENTER>
</BODY>
</HTML>
```

Назначение тега HTML понятно — он сообщает компьютеру (и нам заодно), с чем тот имеет дело. Тег HEAD информирует, что дальше будет заголовок, а тег TITLE указывает, что после него идет название документа.

Уже разглядывая этот пример, вы сразу заметите, что некоторые теги «ходят» парами (как скобки). Например, HEAD и /HEAD, TITLE и /TITLE, H1 и /H1. Да и роль у них схожа со скобками: первый из них (без косой черты) указывает на то, как дальше долж-

ны выглядеть элементы гипертекста, второй (перед которым стоит косая черта) отменяет указанное оформление текста.

Есть, конечно, и непарные теги, например, тег BR, который указывает, что очередной элемент гипертекста начинается с новой строки. А вот тег BODY — парный. Он указывает на начало гипертекстового документа. Просто этот тег обладает еще и *атрибутами*, которые управляют оформлением документа в целом. Атрибут BACKGROUND объявляет, откуда надо взять рисунок, из которого как из мозаики будет составлен фон гипертекстового документа. В этом примере мы написали условное имя file_name; в реальном гипертекстовом документе надо указать полный путь к файлу, где хранится нужный рисунок. В теге, завершающем гипертекстовый документ, разумеется, никаких атрибутов указывать не нужно.

Тег IMG SRC тоже не является парным, он указывает, что в этом месте надо поместить рисунок из соответствующего файла. Вы, конечно, догадались, что вместо условного file_name1 здесь тоже надо указывать полный путь к желанному для вас графическому файлу. На лабораторных работах вы поближе познакомитесь с HTML, и мы надеемся, что вам станет ясно, как работают Web-странички и как они создаются.

Конечно, мы не рассказали даже о десятой доле возможностей глобальных сетей. Но, будем надеяться, вы уже поняли, что и здесь компьютерные информационные технологии «не ударили в грязь лицом», оставив далеко позади традиционные возможности обычной почты.

ВОПРОСЫ И ЗАДАНИЯ

1. В чем разница между обычной текстовой страницей и гипертекстовой страницей?
2. Что называют Всемирной паутиной?
3. Для чего предназначен HTML?
4. Что такое теги и для чего они предназначены?
5. В Приложении 1 к учебнику имеется краткий справочник по языку HTML. Пользуясь этим справочником, объясните назначение двух оставшихся неразъясненными тегов из примера гипертекстовой разметки, приведенного в объяснительном тексте параграфа.

§15. Как получить информацию

Когда говорят об Интернете, чаще всего имеют в виду систему World Wide Web, хотя ею вовсе не ограничивается перечень услуг, предоставляемых Интернетом. Вот и мы начнем именно с рассмотрения WWW.

В настоящее время систему Всемирной паутины составляют самые разные электронные документы: электронные библиотеки, виртуальные музеи и галереи, каталоги по продуктам и услугам, открытая правительственная информация, электронные журналы и газеты, публикации и программные продукты, коллекции музыкальных произведений и видео.

Как же добраться до всех этих богатств? Для этого прежде всего пользуются специальными программами просмотра гипертекстовых страниц. Такие программы называются **браузерами** (от английского browse — пролистывать, просматривать, читать без какого-либо определенного плана). Данные программы осуществляют следующие важные функции: просмотр, сохранение и печать документа, имеющегося в сети, формирование закладок, хранение истории навигации по WWW, настройка на различные форматы документов. Браузеры являются необходимыми и нередко достаточными инструментами поиска информации в Интернете. Впрочем, браузером можно пользоваться и не только применительно к гипертекстовым страницам Всемирной паутины; любой гипертекст, созданный по правилам HTML-стандарта, можно просмотреть браузером.

У браузеров тоже есть своя история, которая начинается с первых чисто текстовых браузеров. Браузер сегодня — это мощный графический интегрированный пакет программ, предназначенный для навигации в Интернете. Из современных браузеров наиболее популярными и достаточно мощными являются:

- **Microsoft Internet Explorer**;
- **Netscape Navigator**.

Внешний облик этих программ представлен на рисунках 17 и 18. Выполняя лабораторную работу № 11, вы на практике познакомитесь с их возможностями.

А пока поговорим об еще одном важном для Интернета понятии. Это понятие — **универсальный указатель ресурса**, или сокращенно **URL** (Uniform Resource Locator). Он представляет собой точное описание предоставляемого сервиса (так обычно называют услуги в глобальных сетях), включающее его местонахождение в Интернете, и имеет следующую структуру:

service://host:port/path/file.ext

(вид сервиса://имя узла:номер порта/путь/имя файла. расширение)

Например:

http://www.usu.ru/...

Сервис **http** означает, что мы намерены иметь дело со Всемирной паутиной; www.usu.ru — это имя того узлового компьютера, с информационными ресурсами которого мы собираемся работать; в данном случае речь идет об узловом компьютере сети WWW, обслуживающем Уральский госуниверситет.

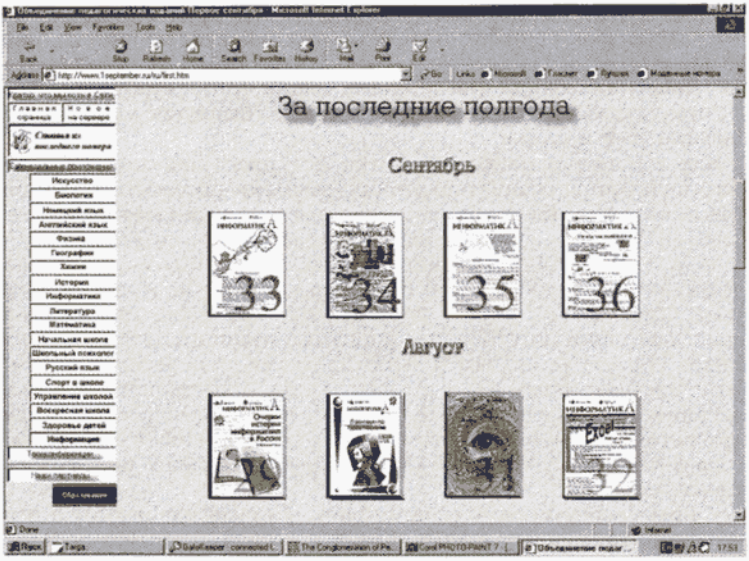


Рис. 17. Интерфейс Microsoft Internet Explorer

Каждый браузер позволяет обратиться к нужному вам ресурсу с помощью универсального указателя ресурса.

О других сервисах Интернета мы расскажем в § 16, а сейчас познакомимся с тем, как устроены имена узловых компьютеров.

В Интернете каждому компьютеру присвоен свой Интернет-адрес (IP-адрес). Он состоит из четырех полей, принимающих значение от 000 до 255. Например, 215.055.255.240 или 004.226.232.206.

Пользоваться такими адресами человеку крайне неудобно — и запоминать их трудно, и легко ошибиться при наборе... Поэтому существует специальный сервис, который определяет адрес через имя и называется Domain Name System — **доменная система имен** (сокращенно **DNS**).

Имя в доменной системе задается последовательностью **доменов**, разделенных между собой точками. Можно считать, что домен — это обозначение группы пользователей, и чем правее находится домен в имени, тем шире эта группа. Например:

www.virlib.usu.ru

Здесь www — имя конкретного компьютера; virlib — домен, который определяет конкретную организацию: в данном случае виртуальную библиотеку; usu — домен Уральского госуниверситета, ru — домен России.

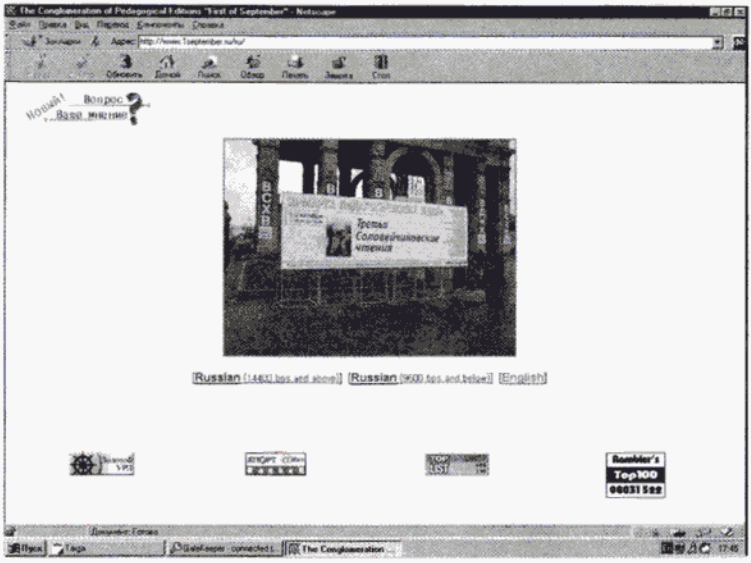


Рис. 18. Интерфейс Netscape Navigator

В принципе имя может содержать любое число доменов.

В Интернете имеется специальная служба, разрабатывающая и поддерживающая систему имен. К примеру, для всех стран мира придуманы двухбуквенные обозначения: .us — США, .ca — Канада, .ru — Россия и т. д.

В США с введением DNS были созданы шесть организационных доменов высшего уровня:

com — коммерческие организации;
edu — учебные заведения;
gov — правительственные учреждения;
mil — военные организации;
org — прочие организации;
net — сетевые ресурсы.

Понимая общий принцип формирования доменных имен, нередко по доменному имени можно догадаться, какой организации оно принадлежит.

Вернемся к поиску информации во Всемирной паутине. Как следует из сказанного выше, этот поиск фактически сводится к определению URL с нужной нам информацией. Для ускорения поиска в Интернете существуют *каталоги* и *справочники*, а также упорядоченные системы ссылок на различные URL.

Но искать «вручную» требуемый URL может оказаться долгим и нелегким делом. Есть, конечно, и другие способы поиска в WWW нужных документов. Можно, к примеру, прибегнуть к помощи Web-робота, которого нередко называют «пауком». **Web-робот** — это программа, которая, получив запрос пользователя, систематически исследует WWW, находя документы и оценивая их соответствие запросу, и возвращает пользователю список найденных документов, расположив их по степени близости к запросу.

Фактически же речь идет об использовании той или иной **информационно-поисковой системы** (сокращенно **ИПС**), располагающей информацией о документах сети. Правила работы с каждой из таких ИПС определены соответствующей инструкцией пользователю.

Расскажем подробнее о поисковых системах русскоязычной части Интернета. Сейчас в России создано несколько мощных ИПС, лидерами среди которых можно назвать Rambler, Апорт! и Яndex.

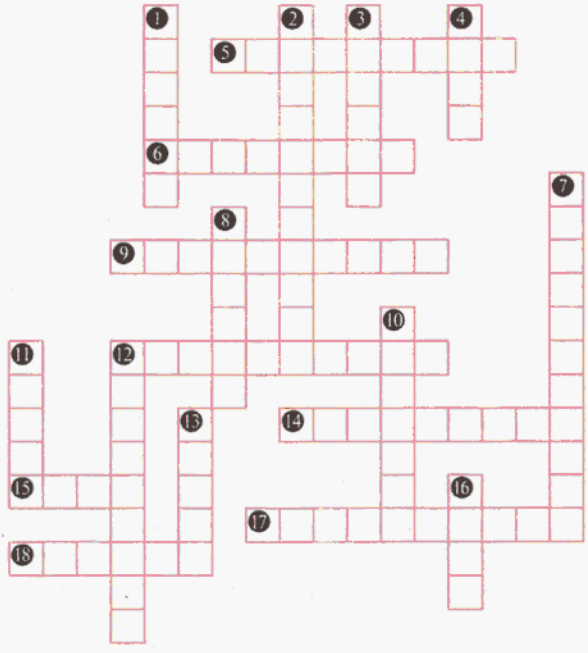
Rambler (http://www.rambler.ru) — первая по-настоящему профессиональная отечественная поисковая система. Она начала работать с конца 1996 г. Эта система обеспечивает поиск на 2 млн Web-страниц, расположенных в России и странах ближнего зарубежья. Система имеет удобный интерфейс, позволяющий легко составлять запрос.

Система Апорт! (http://www.aport.ru) вступила в действие на полгода позже ИПС Rambler. Одно из главных ее достоинств — удачные средства составления запроса. Кроме того, эта система предлагает возможность автоматического перевода запросов с русского на английский язык и наоборот. Найденные документы упорядочиваются в зависимости от частоты употребления в них искомых терминов и глубины их расположения в тексте.

Система Яndex (http://yandex.ru) появилась почти одновременно с ИПС Апорт! Эта система, помимо WWW-серверов с доменами .ru и .su, просматривает содержание зарубежных русскоязычных WWW-узлов.

ВОПРОСЫ И ЗАДАНИЯ

1. Как называются программы просмотра гипертекстовых страниц Всемирной паутины?
2. Для чего служит доменное имя?
3. Как устроен универсальный указатель ресурса в Интернете?
4. Какие средства поиска информации имеются в WWW?
5. В этом задании вам предлагается небольшой кроссворд. Разгадать его совсем нетрудно — все отгадки названы в тех четырех параграфах главы 2, которые вы уже изучили. Желаем успеха!



По горизонтали:

5. Жесткий диск. 6. Объединение глобальных сетей, поддерживающих протокол TCP/IP. 9. Документ со скрытыми связями своих элементов. 12. Одно из внешних устройств компьютера, предназначенное для ввода информации. 14. Преобразование цифрового сигнала в телефонный. 15. Манипулятор с кнопочным управлением, служащий для ввода информации. 17. Одно из центральных понятий науки XX века. 18. Имя одного из основателей Интернета.

По вертикали:

1. Страна, имеющая доменное имя ru. 2. Наука, изучающая процессы получения, хранения, передачи и обработки информации. 3. Компьютер, снабженный программами хранения и передачи информации в сети. 4. Система компьютеров, соединенных каналами связи. 7. Преобразование телефонного сигнала в цифровой. 8. Город, в котором была начата разработка Всемирной паутины. 10. Программа, позволяющая просматривать гипертекстовые страницы Интернета. 11. Устройство, преобразующее цифровой сигнал в телефонный и обратно. 12. Основной инструмент современных информационных технологий. 13. Минимальный элемент имени в электронном адресе. 16. Общее название для программ поиска информации во Всемирной паутине.

Поиск информации в Интернете

Начнем с простого: пусть вам известно, *где* взять нужную информацию. Это значит, вы знаете, на каком сервере хранится данная информация, в какой папке (или в каком каталоге) она располагается и каково имя файла. Иными словами, вы знаете универсальный указатель ресурса, предоставляющего нужную информацию. Возможно, у вас есть на примете какой-нибудь интересный для вас URL, но мы для учебных целей предлагаем воспользоваться следующим:

http://www. virlib.eunnet.net/mif

❶ Вызовите имеющийся на вашем компьютере браузер и запишите в строку вызова указанный URL. Вызовите искомый файл.

Перед вами научно-популярный журнал для старшеклассников «МИФ». Попробуйте определить, глядя на первую страницу, что означает это название.

Слева вы видите список номеров этого журнала. Каждый номер имеет гипертекстовую связь с оглавлением данного номера.

❷ Выберите какой-нибудь номер и просмотрите оглавление.

Возможно, вам понравилось название какой-нибудь статьи. Очередная гипертекстовая ссылка способна перенести вас к аннотации выбранной статьи.

❸ Ознакомьтесь с аннотацией.

Однако у вас нет сейчас времени читать всю статью. Вы можете ее скопировать на свой компьютер и прочитать потом, когда уроки закончатся.

❹ Выделите привлекшую ваше внимание статью и скопируйте ее в файл с подходящим именем.

Если вы чувствуете, что полностью разобрались с первыми четырьмя заданиями, то переходим к следующему заданию. Пусть, к примеру, вы хотите посмотреть фотографию Винтона Серфа — какникак отец Интернета!

❺ Вызовите с помощью браузера какую-либо ИПС Интернета и, следуя ее указаниям, сформируйте соответствующий запрос на имя Cerf.

Скорее всего, вас постигла неудача — слишком много документов предлагает вам ИПС на такой запрос. И работа ее продолжается — ведь это имя встречается в тысячах публикаций. Чтобы уменьшить объем поиска, в запрос можно добавить еще один признак «photo».

❻ Сформируйте нужный запрос.

Если и это не помогло (постарайтесь понять почему), подскажем, что есть в Интернете фотография Серфа с президентом США Клинтоном.

❼ Сформируйте нужный запрос.

Получилось? Если да, то вот задание посложнее: составьте список школ в своем городе (или области), имеющих в Интернете свои странички — Home pages.

❽ Сформируйте нужный запрос и полученные результаты сохраните в виде текстового документа.

Изготовление HTML-страницы: первые шаги

Выполняя предшествующую лабораторную работу, вы обнаружили немало школ, имеющих свои странички в Интернете.

❶ Выберите наудачу 2 – 3 адреса и посмотрите странички выбранных школ.

Мы не знаем, понравились они вам или нет, но думаем, что и у вас возникло желание создать свою собственную страничку. Этому вы и начнете учиться на данной лабораторной работе.

Прежде всего давайте посмотрим, как устроена такая страничка. Крайне неэтично брать для своих экспериментов первую попавшуюся вам страницу. Учитель назовет вам адрес страницы, на которой вы будете оттачивать свои навыки в работе с гипертекстом.

❷ Загрузите предложенный вам документ. Определите, сколько уровней в нем имеется.

А теперь посмотрим, как создан этот документ.

❸ Просмотрите документ в режиме источника. Вы видите текст странички и сопровождающие его команды языка HTML. Выделите эти команды. С помощью справочной системы учебника (раздел «Язык HTML») расшифруйте их.

Оставьте на время компьютер в покое и подумайте, как бы вы хотели, чтобы выглядела страничка вашего класса. Иными словами, разработайте план своей страницы в Интернете. Да так, чтобы всем хватило работы. На первом уровне такой страницы можно, к примеру, расположить:

название;

девиз;

«Наш класс» (переход на 2-й уровень);

«Наши интересы» (переход на 2-й уровень);
«Выдающиеся личности нашего класса» (переход на 3-й уровень; при этом каждая личность может считаться выдающейся);
фотографии класса;
«С кем мы дружим» (переход на 3-й уровень);
«Пишите нам»;
кнопки переключения кодировок.

Вы можете расширить этот список или убрать из него что-то по своему вкусу. Для удобства дальнейшей работы вы можете около каждого пункта пометить, какими командами языка HTML вы будете пользоваться для его создания, где располагаются файлы с нужным вам фоном и т. п.

А теперь займемся изготовлением первого уровня вашей гипертекстовой странички.

❹ Создайте в имеющемся на вашем компьютере HTML-редакторе новый документ. Пользуясь командами языка HTML, запишите структуру первого уровня страницы и оформите необходимые тексты. Создайте фон. Сохраните документ под именем klass1.htm. Проведите конкурс, у кого получилось лучше.

Первый шаг к созданию собственной страницы сделан. Впереди новые лабораторные работы.

Создание гипертекста

Создав в предыдущей лабораторной работе первый уровень своей гипертекстовой страницы, вы многому уже научились. Теперь разработайте план для каждой из страниц следующих уровней и реализуйте эти планы.

❶ Создайте с помощью HTML-редактора страницы второго уровня. Сохраните их, например, с именами klass2_1.htm, klass2_2.htm и т. д.

❷ Сделайте контекстные ссылки со страницы 1-го уровня на страницы 2-го уровня. Создайте ссылки со страниц 2-го уровня на страницу 1-го уровня с помощью кнопки Home. Сохраните получившиеся документы. Просмотрите свою работу с помощью браузера.

❸ Если у вас запланированы страницы последующих уровней, создайте их и установите нужные ссылки между страницами. Не забудьте сохранить созданные вами документы.

Согласитесь, что любой текст будет интереснее, если его сопровождают иллюстрации. Вот и ваша страничка только выиграет от

того, что вы поместите на ней рисунки. Подготовить рисунки надо, конечно, заранее в графическом редакторе. Если вы забыли, как пользоваться графическим редактором, обратитесь к § 11. Единственное ограничение — ваши графические файлы должны быть сохранены в формате gif или jpg.

❹ Вставьте графические объекты в уже существующие страницы или создайте новые страницы с компьютерной графикой. Установите связи с уже имеющимися страницами. Не забудьте сохранить созданные вами документы. Просмотрите проделанную вами работу браузером.

❺ Если рисунки вам не очень удаются, то вы можете поместить на своих страничках фотографии. Для этого отсканируйте выбранные вами фотографии и создайте соответствующие графические файлы (разумеется, с расширением gif или jpg). А затем поступайте с ними как указано выше.

§16. Что еще можно делать в Интернете

Надеемся, вы уже ощутили Интернет как бездонное хранилище информации и средство коммуникации с любыми уголками нашей планеты. Но информация информации рознь, и ваш интерес к ней тоже может быть весьма различным. Одно дело — просмотреть новости в той или иной газете и совсем другое — изучить большую научную статью. Для такого изучения потребуется время, и совсем ни к чему тратить на это довольно дорогие ресурсы сети. Гораздо эффективнее получить нужную информацию к себе на компьютер в виде соответствующего файла. Кроме того, Интернет наполнен не только текстовой, звуковой или видеоинформацией, но в нем имеется немало самого разнообразного программного обеспечения, распространяемого как солидными программистскими фирмами, так и просто программистами-любителями. Получить файлы по сети позволяет **FTP-сервис** (от **F**ile **T**ransfer **P**rotocol — протокол передачи файлов). Именно он предоставляет доступ к файловым системам и выполняет передачу файлов в сети. Одним из распространенных видов FTP-серверов является анонимный FTP-сервер. На таком сервере вы используете в качестве своего имени слово «anonymouse», а в качестве пароля — свой E-mail. Если же вы хотите установить соединение с сервером, не предоставляющим анонимного сервиса, вы должны иметь собственное имя пользователя и пароль, дающий вам право доступа к системе.

Но, конечно, польза от FTP-сервиса была бы не так значительна, если для получения информации, чем можно поживиться благодаря этому сервису, пришлось бы просматривать все имеющиеся в сети анонимные серверы. Конечно, Интернет и здесь идет навстречу пользователю. Для этого существуют ИПС, с помощью кото-

рых можно проводить поиск информации, расположенной на анонимных FTP-серверах. Система, поддерживающая этот вид услуг, регулярно собирает с таких серверов информацию о содержащихся на них файлах. К таким системам относится, например, ИПС Archie.

Перестроив на электронный манер почтовую связь, разработчики глобальных сетей не успокоились. Конечно, получать письма приятно, но еще приятнее диалог, когда на каждый свой вопрос или суждение вы тут же получаете реакцию собеседника. Ну совсем как по телефону. Только вместо голоса — письменное сообщение: вы строчку, вам в ответ строчку, снова вы и т. д. Есть сервис Интернета, который позволяет таким образом общаться на различные темы. Сокращенно его называют **IRC-сервис** (от **I**nternet **R**elay **C**hat; напомним, что chat в переводе означает «разговор, беседа, болтовня»). Он используется для общения в свободное время, но не для проведения длинных и серьезных обсуждений — дороговат этот сервис! Тем не менее любители компьютерных игр, например, могут, невзирая на расстояние, сыграть с несколькими (даже с несколькими сотнями) соперников-партнеров.

А для серьезного общения в Интернете существует система **телеконференций** UseNet, которая позволяет тем, кто к ней обращается, участвовать в различных дискуссионных группах. Телеконференции тоже родились не на пустом месте — они представляют собой сетевой вариант так называемых **досок объявлений** (**BBS** — от **B**ulletin **B**oard **S**ystem), изначально располагавшихся не на серверах, а на компьютерах с модемным доступом.

Тому, кто еще не участвовал в телеконференции, но желает этого, лучше всего начать с просмотра телеконференций, которые имеют в своем имени слово newuser (новый пользователь). Например, с телеконференции news.announce.newusers.

Система имен телеконференций, как обычно, построена по иерархическому принципу. Первый домен сообщает об общей тематике конференции.

Наиболее распространены следующие основные виды телеконференций:

biz — бизнес;
comp — компьютеры;
news — новости общего характера;
rec — развлечения (хобби и искусство);
sci — наука;
soc — социальные темы;
talk — с ориентацией на дискуссию;
misk — темы, не подходящие под вышеуказанные категории;
alt — альтернативные.

Как говорится, на любой вкус.

Конечно, мы рассказали далеко не обо всех возможностях Интернета. Так, относительно недавно начали работу электронные ма-

газины, в которых вы можете посмотреть на товар и заказать приглянувшуюся вам вещицу с доставкой на дом. И многое-многое другое, полезное для работы и быта.

ВОПРОСЫ И ЗАДАНИЯ

1. В чем суть FTP-сервиса?
2. Как узнать, имеется ли на каком-нибудь FTP-сервере нужная вам информация?
3. Для чего предназначены телеконференции?

§17. Этика Интернета. Опасности Интернета

Живя в любом обществе, человек подчиняется писаным и неписаным законам этого общества. Пока круг общения невелик, такие законы легко познаются и к ним легко привыкнуть. Но вот вы вошли в Интернет — и ваше общение распространилось на весь мир.

Поскольку Интернет — это объединение самых разных глобальных сетей, то каждая из этих сетей имеет свои собственные правила поведения и обычаи. Поэтому вопрос, что дозволено в Интернете, весьма непростой. К примеру, на Интернет распространяется действие международных законов, но в то же время это и «территория» национальных законов, которые непрерывно изменяются.

Скажем, посредством Интернета можно передавать информацию через любые государственные границы. Но при передаче чего бы то ни было через такую границу начинают действовать экспортные законы, которые в разных государствах могут быть весьма различными. В частности, эти законы, как правило, требуют лицензии на экспорт.

Для Интернета экспортную лицензию можно отнести к категории «общая лицензия», которая разрешает вывозить все, что не запрещено явно. Так что все, что мы получаем или передаем по Всемирной сети и на что не наложены ограничения из соображений безопасности, подпадает под общую лицензию. Правда, у разных государств разные мнения на счет безопасности.

Если же вы намерены воспользоваться чьим-то электронным продуктом (Web-страничкой или программным обеспечением, или еще чем-то, имеющимся в сети), то вам прежде необходимо ознакомиться с правами интеллектуальной собственности и лицензионными ограничениями. До того, как «вывезти» нечто по сети, узнайте, кто имеет права на это нечто. Проверьте, к какой категории относится программный продукт, свободного ли он распространения (free). А перед тем как высылать «за границу» не свой продукт, убедитесь, что вы имеете на то разрешение.

Что касается этики, то Интернет старается и в этом вопросе выработать некоторые общие положения. Вот только два принципа сетевой этики:

- проявление индивидуальности уважается и поощряется, гонения за взгляды и вкусы недопустимы;
- сеть следует защищать.

Но Интернет, предоставляя почти неограниченные возможности для общения людей, весьма слабо защищен от неэтичного поведения. К себе в дом вы вряд ли пустите любого незнакомца с улицы. А предоставляя свои ресурсы в Интернет, вы открываете дверь для каждого. Надо помнить об этом, когда в своих страничках вы указываете свой электронный адрес. Будьте тогда готовы, что на вас может хлынуть потоком реклама назойливых продавцов, неприятные вам шутки анонимных шутников и т. п. Есть пользователи, считающие вполне дозволенным взять себе любой файл, до которого они могут добраться, включая частную переписку. И все это вы должны учитывать, когда включаете свой компьютер в глобальную сеть. Уметь защищаться — это также ваша забота. Например, можно шифровать свои сообщения — такой сервис имеется сейчас для связи по электронной почте.

Но главное — вести себя этично самому и способствовать в этом своим партнерам.

3 Компьютерное моделирование

Мир, окружающий нас, огромен. Его разнообразие поражало человека уже много веков назад. Сейчас мы к этому привыкли. Но каждый раз, оставив в стороне повседневную суету, мы снова очарованы гармонией окружающего мира.

Гармонию мира старается передать в своих произведениях художник. Языком науки пытается описать гармонию мира ученый. И если им удается уловить самое существенное в том маленьком фрагменте мировой картины, который стал объектом их внимания, получается шедевр, происходит открытие.

Впрочем, умение выделить самое существенное небесполезно для каждого из нас. Ведь тогда не будут растрачены по пустякам силы и средства, отчетливо будет обозначена цель и намечены пути к ее достижению, появится возможность оценить перспективы и последствия. Информационные технологии и компьютер здесь помощники человека. О том, как их использовать, и рассказывается в этой главе.

§18. О задачах и моделях

Начнем с простого примера. Вы приехали в большой, незнакомый для вас город. Вам надо добраться до друга, который переехал недавно на новую квартиру, где вы еще не были. Выйдя из здания вокзала, вы, не думая (это самое главное предположение в нашем мысленном эксперименте!), садитесь в транспорт — трамвай, троллейбус, автобус или метро. Все так же не думая, проезжаете несколько остановок, выходите, садитесь на другой вид городского транспорта или идете пешком и т. д. При таких ваших действиях очень мало шансов, что друг вас дождется.

Мы, однако, хотим привлечь ваше внимание не к прописным истинам типа «Сначала думай — потом делай», «Семь раз отмерь — потом отрежь» и т. п., а к тому, как вы будете решать стоящую перед вами жизненную задачу. Самое легкое решение — попросить друга встретить вас на вокзале. Тогда больше и думать ни о чем не надо — друг сам предложит нужный маршрут. Что ж, такой метод — воспользоваться чьим-то готовым решением — достаточно распространен: от кулинарии до высших политических сфер. И в этом нет ничего зазорного — ведь потому и бесценен опыт человечества, что не нужно каждому заново, с нуля решать все жизненные задачи.

Если же друг не может вас встретить или вы хотите своим приездом преподнести ему сюрприз, то решать эту жизненную задачу придется иначе. Можно, к примеру, взять транспортную схему города (т. е. план города с нанесенными на него линиями движения транспорта) и прокладывать маршрут по этой схеме (рис. 19). Каждому ясно, что произошло: задача нахождения пути к дому приятеля заменена другой задачей — найти маршрут на транспортной схеме.

Транспортная схема — это, конечно, не сам город, а его упрощенное представление с помощью условных обозначений. Но в ней отражено то существенное, что позволит вам решить задачу — построить маршрут к дому вашего друга.

Рис. 19. Фрагменты схем метро Москвы и Санкт-Петербурга

Замена одного объекта (процесса или явления) другим, но сохраняющим все существенные свойства исходного объекта (процесса или явления), называется ***моделированием****, а сам заменяющий объект называется* ***моделью*** *исходного объекта.*

Да если вы и знаете, как добраться до друга, то все равно решение этой задачи присутствует в вашем сознании не в виде реальных улиц, трамвайных или троллейбусных вагонов, а в виде некоторого представления о том, какими улицами идти и номерами каких транспортных маршрутов вам придется воспользоваться. Иными словами, в своей памяти вы храните решение, но не исходной жизненной задачи, а ... ее модельной переформулировки!

Вообще, какую бы **жизненную задачу** ни взялся решать человек, первым делом он строит модель — иногда осознанно, а иногда и нет. Ведь бывает так — вы напряженно ищете выход из трудной ситуации, пытаясь нащупать, за что можно ухватиться. И вдруг приходит озарение... Что же произошло? Это сработало замечательное свойство нашего разума — умение безотчетно, словно по какому-то волшебству, уловить самое важное, превратить информационный хаос в стройную модель стоящей перед человеком задачи.

Как видите, с моделями вы имеете дело ежедневно, ежечасно и, может быть, даже ежеминутно. Вы, скорее всего, никогда об этом не задумывались, поскольку построение моделей для человека так же естественно, как ходьба или умение пользоваться ножом и вилкой. Разница же в том, что ходить, пользоваться ножом и вилкой вас довольно долго учили, и теперь вы хорошо умеете это делать. А вот искусством строить модели вы наверняка овладевали стихийно, сами того не подозревая. Потому и получаются у вас иногда удачные решения жизненных задач, а иногда не очень — ведь разные модели одной и той же жизненной задачи могут приводить к весьма различным результатам. Вы и сами в этом скоро убедитесь, выполняя лабораторную работу № 14.

ВОПРОСЫ И ЗАДАНИЯ

1. Что означает термин «моделирование»?
2. Что такое модель объекта, процесса или явления?
3. Укажите, какие модели вы обычно используете для решения следующих жизненных задач:
   а) купить билет в кино;
   б) скомплектовать волейбольную команду;
   в) испечь торт.
   Приведите свои примеры жизненных задач и моделей, используемых для их решения.

§19. Как устроены модели

Вряд ли со словом «модель» вы впервые встретились, читая предыдущий параграф. Любой самолет, прежде чем отправиться в свой первый испытательный полет, пройдет испытания в виде уменьшенной своей копии, т. е. модели, сохраняющей геометрическую форму, в аэродинамической трубе, чтобы подтвердить свои летные качества. Без карт, т. е. моделей земной поверхности, не отправится в плавание ни один капитан морского судна. В кабине космонавтов расположился маленький глобус — модель нашей планеты. В каждый момент времени он показывает, над какой точкой земной поверхности пролетает сейчас космический корабль. А с какими только моделями не сталкиваетесь вы на уроках в школе! Это карты и глобусы на уроках географии, модели атомного строения веществ на уроках химии, муляжи на уроках биологии и многое другое.

Общая черта, присущая этим моделям, состоит в том, что они *копируют* исходный объект. Они делаются из иного материала, чем исходный объект, зачастую более дешевого. Они, как правило, имеют другие размеры, чем моделируемый объект, — более удобные для человека, работающего с ними. Они вообще могут быть **виртуальными**, т. е. созданными как образ на экране компьютера или других специальных устройств, позволяющих не только увидеть объект, но и ощутить его теми или иными органами чувств. Но люди в своей работе и повседневной жизни часто пользуются совсем иными моделями.

Представьте, что вам предстоит поездка по железной дороге из одного города в другой. Вы пришли в билетные кассы выбрать подходящий поезд и купить билет. Вас, скорее всего, интересует время отправления, продолжительность пребывания в дороге, время прибытия и, быть может, стоимость поездки. Конечно, забавно понаблюдать за моделькой поезда, который на ваших глазах проедет по макету местности от вашего города до пункта назначения. Потом можно посмотреть, как другой поезд по тому же или другому маршруту пройдет до места назначения. (Скажем, от Москвы до Екатеринбурга курсирует около десятка поездов, причем одни из них идут через Казань, а другие — через Киров.) Но скорее всего, вы удовлетворитесь простым расписанием движения поездов, в котором найдете всю интересующую вас информацию.

Расписание движения поездов — это тоже модель. Но совсем иного вида. В ней просто указаны интересующие вас *характеристики объекта* — в данном случае поезда, следующего из некоторого пункта А в некоторый пункт Б. Вместо слова «характеристики» нередко употребляют слово «**параметры**».

Модель, представляющая объект, процесс или явление набором параметров и связей между ними, называется ***информационной моделью.***

Вскрыть связи между параметрами информационной модели — это зачастую едва ли не самая сложная часть в построении модели, возникающая после того, как определены ее параметры. Ведь, скажем, составление расписания движения поездов в том и состоит, чтобы связать воедино время прибытия и отправления каждого поезда с каждой станции так, чтобы не произошло столкновений, заторов и каждый поезд был обслужен (успели сойти и войти пассажиры, залита свежая вода, подвезены свежие продукты и т. д.). Мы и пытаться не будем описать все имеющиеся здесь связи между многочисленными параметрами.

Но чтобы вопрос о связях между параметрами модели стал яснее, давайте рассмотрим модель всем хорошо известного процесса — равномерного прямолинейного движения. Каковы параметры этого процесса? Все их знают — это постоянная скорость v, время t и путь S, пройденный телом за это время. Связь между этими параметрами напишет даже восьмиклассник:

$$S = v\,t.$$

Если же рассматривать другой процесс — равноускоренное прямолинейное движение, то параметров будет четыре: начальная скорость v_0, постоянное ускорение a, время t и путь S, пройденный телом за это время. Связь между этими параметрами тоже, наверно, многие помнят:

$$S = v_0\,t + \frac{at^2}{2}.$$

В этих информационных моделях есть важная особенность: значения параметров — это числа, а связь между ними представлена в виде равенства, иными словами, функциональной зависимостью.

Информационная модель, в которой параметры и зависимости между ними выражены в математической форме, называется ***математической моделью****.*

Рассмотренные два примера — это простейшие математические модели. Математика сегодня развилась настолько, что оперирует не только с числами, а зависимости выражает не только на языке равенств и неравенств. Но мы не будем в нашем курсе так далеко забираться в математические «дебри», и в рассматриваемых нами математических моделях параметры всегда будут числовыми, а связи между ними будут записываться равенствами и неравенствами.

Разумеется, информационная модель вовсе не обязана быть математической. Что, к примеру, представляет из себя информационная модель обычного школьного класса? Один из очевидных пара-

метров этой модели — фамилия и имя ученика. Его значение — это, конечно, не число, а пара слов (в данном случае собственных имен) языка той страны, где располагается школа. Другой естественный параметр — дата рождения. В нашей стране даты принято указывать тремя парами цифр, разделенными точкой: первая пара указывает число, вторая — номер месяца, третья пара — две последние цифры года рождения. Этот параметр тоже вряд ли можно назвать числовым. Более того, может быть выбрана и другая форма записи даты рождения, например 9 ноября 1979 г. Связь между этими параметрами такой модели довольно проста: каждому значению первого параметра соответствует в точности одно значение второго параметра. Эту связь удобнее всего представить в виде таблицы, например такой:

Параметр	*Фамилия, имя*	*Дата рождения*
	Алексеев Андрей	25 июля 1980 г.
Значения параметра	Борисова Белла	9 ноября 1979 г.
	Васильев Валерий	29 февраля 1980 г.
	...	...

Конечно, параметров, описывающих школьный класс, может быть довольно много. Если записывать их, как и выше, в виде таблицы, то такая таблица может иметь много разных столбцов. Да и таблица для такой информационной модели может потребоваться далеко не одна. Для того чтобы с такими таблицами, представленными в компьютере, можно было работать, существуют специальные программы: базы данных и системы управления базами данных. Рассказу о базах данных мы посвятим следующий параграф и лабораторную работу.

И вообще мы уделяем особое внимание информационным моделям потому, что для работы именно с этими моделями можно использовать компьютер. Ведь компьютер — основной инструмент современных информационных технологий, и значит, его сфера применения — решение задач на основе информационных моделей.

ВОПРОСЫ И ЗАДАНИЯ

1. Для чего используются модели человеком? Приведите примеры моделей и укажите, как они используются в деятельности человека.
2. Приведите примеры моделей, с которыми вам приходилось иметь дело на уроках в школе.
3. Что называют информационной моделью объекта, процесса или явления?

4. Как устроена информационная модель?
5. Какую модель называют математической?
6. Почему информационные модели играют особую роль в современных применениях компьютера?
7. Какие математические модели вы изучали на уроках физики, рассматривая объекты, процессы и явления, изображенные на следующих рисунках?

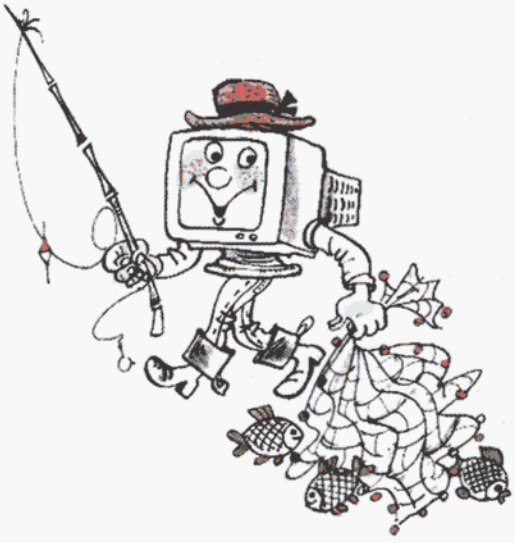
8. Часто моделью химической реакции выступает уравнение этой реакции. Например, $2KOH + H_2SO_4 = K_2SO_4 + 2H_2O$.

   Является ли эта модель информационной? Если да, то укажите параметры этой модели и связи между ними.
9. Какие параметры для информационной модели вашего класса вы бы предложили?

§ 20. Базы данных и информационно-поисковые системы

Представьте себе, что у вас есть небольшая фирма по выдаче напрокат видеофильмов. К вам постоянно обращаются клиенты в надежде получить интересующую их запись. Кому-то требуется самый популярный фильм сезона, у кого-то есть любимый актер, другие испытывают ностальгию по музыке давно прошедших лет. Дела в фирме идут хорошо, от клиентов нет отбоя. Одно удручает вас: все-таки они медленно обслуживаются, поскольку нелегко в длинных списках отыскать нужное название. Мало помогают и выложенные на прилавок списки записей. Одному клиенту надо подобрать фильмы-боевики. Другому — фильмы с любимой актрисой. Третий интересуется всеми мультфильмами с трансформерами...

Кстати, интересуются ведь не только видеофильмами. Наверняка почти каждый из вас пережил увлечение собирательством марок (этикеток, календариков и пр.). Заметим, что истинный коллекционер не станет складывать как попало дорогие ему экспонаты, а рассортирует их по тем или иным признакам. Например, для марок это страна, выпустившая марку, год выпуска, тема, гашеная марка или нет... Учитывается даже состояние (сохранность зубцов, наличие клея и др.). А календарики требуют другой классификации. Всю эту информацию заядлые коллекционеры хранят в специальных каталогах. Однако, даже имея под рукой каталог, порой приходится тратить много усилий, чтобы подобрать нужный вариант обмена.



Описанные ситуации имеют много общего: в большом объеме информации разыскивается та, которая необходима на данный момент. И здесь снова надежным помощником становится компьютер. Давайте вспомним предыдущую главу. Там мы отчетливо поняли, что компьютер — это мощный инструмент для переработки информации. По исходным данным и формулам он получает необходимые нам результаты.

Теперь же нам требуется, чтобы компьютер сыграл роль рыбака с сетью, который выловит из громадного косяка рыбы только ту добычу, которая нас интересует. Или, если это вам больше понравится, компьютер похож на обогатительную фабрику, которая из сотен тонн породы добывает нужный минерал. Роль косяка рыбы (или необогащенной руды) выполняют **базы данных (БД)**. А чтобы накапливать, изменять, хранить и искать нужную информацию, необходимо использовать специальные программы — **системы управления базами данных (СУБД)** и **информационно-поисковые системы (ИПС)**.

Различие между ними весьма условное. Пояснить его можно так. Если речь идет о каталоге библиотеки, то никому и в голову не придет разрешить любому посетителю изменять в этом каталоге информацию о книгах, хранящихся в ее фондах. А просмотреть и выбрать то, что необходимо, — пожалуйста. Можно сказать, что специалистам — сотрудникам библиотеки — доступны все возможности работы с базой данных, а посетители работают только с информационно-поисковой системой.

Как же организована база данных? В ней содержатся сведения о большом количестве однотипных объектов. При этом для каждого из объектов существенными являются значения лишь некоторых **признаков**. Впрочем, об этом мы говорили уже в конце предыдущего параграфа. Иногда некоторые из признаков объявляют **ключевыми**. Это полезно, потому что по ним, в частности, в дальнейшем можно сделать **сортировку**. Например, если мы хотим получить список всех фильмов с участием Шварценеггера в хронологическом порядке, необходимо год выпуска фильма объявить ключевым признаком. Некоторые из признаков могут быть объявлены *обязательно присутствующими*. Например, любой художественный фильм всегда имеет какое-нибудь название.

Каждая конкретная база данных предназначена для решения своего класса задач. В свою очередь для каждого класса задач характерен свой набор объектов и их признаков. Поэтому и соответствующие базы будут различными: одна нужна, чтобы разыскивать книгу по каталогу, и совсем другая — для облегчения работы директору школы. Впрочем, для разных классов задач объекты могут быть одинаковыми, а наборы нужных признаков — разными. Например, овощевода интересуют урожайность, время посева и сроки уборки овощей, повара — калорийность овощей и рецепты овощных блюд, а директора овощехранилища — условия хранения тех же самых овощей.

Для того чтобы заставить ЭВМ найти интересующие нас сведения, нужно составить **запрос**. Правила записи запросов для каждой ИПС свои. Эти правила устанавливаются теми, кто создает ИПС. Обычно самые распространенные запросы к базе данных уже заранее составлены. Чтобы получить информацию по такому запросу, надо просто выбрать соответствующий пункт в меню, расположенном на экране. Более того, в большинстве персональных баз данных вообще все запросы заранее составлены, и работа с ними доступна даже человеку, далекому от компьютерных премудростей. С одной из таких баз мы и познакомимся на очередной лабораторной работе.

Сейчас в мире созданы сотни тысяч баз данных. Они используются в библиотеках и больницах, в гидрометцентрах и на заводах, в магазинах и планирующих организациях, в банках и частных фирмах. Тематические базы данных для узких специалистов называют **банками данных**. Это может быть банк данных по микропроцессорам, банк по лекарственным средствам, банк публикаций в области ядерной физики и т. п.

В нашей стране действуют десятки банков данных. Пожалуй, самый большой из них — банк данных Российского института научной и технической информации. В нем содержится более 6 млн библиографических сведений о книгах и статьях практически по всем отраслям знаний. Тот, кто покупает авиабилеты, пользуется услугами другого крупного банка данных нашей страны — системой

«Сирена». С помощью компьютера кассир связывается с центральной большой ЭВМ, находящейся за тысячи километров от него, почти мгновенно получает сведения о наличии мест на данный рейс и печатает билет.

Теперь мы предлагаем вам познакомиться с несложной **базой данных «Ученик»**. Она вполне может быть частью автоматизированного рабочего места для директора или завуча школы. В ней содержатся следующие сведения:

- Класс.
- Фамилия.
- Имя.
- Дата рождения.
- Адрес.
- Телефон.
- Информация о братьях и сестрах: год рождения и имя.
- Спортивные результаты.
- Увлечения: чем увлекается.

Предусмотрены и разнообразные формы выдачи информации. Можно разыскивать и сортировать записи:

- по фамилиям;
- по именам;
- по старшинству (самый старший — в самом верху таблицы);
- по дням рождения (тот, кто родился в январе, — вверху таблицы);
- по адресам.

Можно задать интервал, скажем, с 1 по 10 ноября, и компьютер выберет всех, у кого в это время будет день рождения. Можно узнать рекорды школы или класса по различным видам спорта, выдать список умеющих вязать или, скажем, список собирающих баночки из-под сока.

Наконец, можно узнать:

- у кого есть братья;
- у кого есть сестры;
- у кого есть младшие братья или сестры;
- у кого есть старшие братья или сестры.

Ну вот, кажется, всё.

Перед тем как начать поиск, машина в большинстве случаев спросит вас, какого сорта информацию вы хотите получить: обо всех внесенных в базу или только об одном конкретном классе. Исключение составляет лишь поздравление с днем рождения. Здесь поиск всегда ведется по всей базе.

Конечно, можно сказать, что эта система несовершенна. А вдруг нам понадобятся фамилии лучших бегунов среди филателистов? Или всех, у кого есть не только брат или сестра, но еще и говоря-

щий попугайчик?.. А вот тут мы с приятелем поспорили, что на улице Полуночной все держат собак... Информация обо всем этом в компьютере есть, но получить ее одним запросом невозможно: для того чтобы компьютер понимал запрос о подобной информации, необходима специальная программа.

Такая программа потребует от вас умения не только правильно выбирать пункт меню, но еще и грамотно написать запрос. Впрочем, в большинстве случаев это настолько несложно, что, столкнувшись с ИПС, вы способны разобраться в ее языке запросов за считанные минуты. Мы надеемся, что так было и тогда, когда вы работали с ИПС при поиске информации во Всемирной паутине (лабораторная работа № 11).

ВОПРОСЫ И ЗАДАНИЯ

1. Что такое база данных?
2. Что такое СУБД? Что такое ИПС?
3. Что такое банк данных?
4. Какие операции можно производить с данными, хранящимися в базе данных?
5. Что является объектом в базе данных «Почтовые марки»? Какие признаки у объекта в этой базе данных?

Работа с учебной базой данных «Ученик»

Итак, перед вами несложная база данных, которой может пользоваться даже непрофессионал. Напомним, что в ней содержатся:

- Класс.
- Фамилия.
- Имя.
- Дата рождения.
- Адрес.
- Телефон.
- Информация о братьях и сестрах: год рождения и имя.
- Спортивные результаты.
- Увлечения: чем увлекается.

❶ Выберите пункт меню «Список с сортировкой и поиском по фамилиям». С помощью клавиш управления курсором сделайте так, чтобы рамка остановилась на этом пункте меню, и нажмите клавишу <Ввод>.

Перед тем как начать поиск, компьютер в большинстве случаев спросит вас, какой объем информации надо обработать: обо всех учениках, внесенных в базу, или только об учениках одного кон-

кретного класса. (Напомним, что исключение составляет лишь поиск тех, кого надо поздравить с днем рождения. В этом случае просматривается вся база.)

❷ Выберите режим ***класс***. ЭВМ предложит вам этот класс указать. Наберите номер класса и, перемещая курсор, выберите нужную букву.

Далее возможны варианты.

Если в базе данных уже есть ученики вашего класса (бывшие или нынешние), то вы сразу же получите их список.

Правила обращения с ним весьма просты:

- Чтобы перейти на нужную запись, воспользуйтесь клавишами со стрелками вверх или вниз — для перехода на одну запись, или
- клавишей <Page Up> — на страницу вверх, клавишей <Page Down> — на страницу вниз, или
- одновременно нажмите клавиши <Ctrl> и <Page Up> для перехода к самой первой записи, или
- одновременно нажмите клавиши <Ctrl> и <Page Down> для перехода к самой последней записи.
- Чтобы найти запись по первым буквам (или цифрам), надо просто набрать их на клавиатуре. Компьютер автоматически поместит искомые записи в самый верх экрана.
- Чтобы добавить новую запись, надо нажать клавишу <Insert> (при этом на экране появится **окно ввода информации** и вам будет предложено ввести в этом окне новую информацию).
- Чтобы убрать запись из базы данных, надо нажать клавишу <Delete> (при этом на экране снова появится окно ввода информации и вам будет предложено подтвердить удаление записи).
- Чтобы подправить старую запись, надо нажать клавишу <Enter>.
- В любой момент вы можете воспользоваться палочкой-выручалочкой — клавишей <F1>.

Если же в базе данных учеников вашего класса нет, вы сразу же попадете в режим ввода новой записи, как если бы вы нажали клавишу <Insert>.

При вводе новой информации компьютер вам время от времени будет помогать, предлагая уже готовый список видов спорта или увлечений. Вам останется лишь выбрать из него нужное и нажать клавишу <Enter>.

Эти списки меняются точно так же, как и списки учеников. Единственное отличие заключается в использовании клавиши <Enter>. Как вы уже поняли из предыдущего абзаца, она служит для переноса, скажем, наименования вида спорта или увлечения в запись какого-либо конкретного ученика. А что, если неправильно изначально введено наименование? Что же, теперь вся школа будет «прыгать в длену» или «увликаться фудболом»? Конечно, нет.

- Чтобы исправить эти наименования, недостаточно нажать только на клавишу <Enter>, обычно используемую для изменения данных, а нужно нажать две клавиши одновременно: <Ctrl> и <Enter>.
- Добавим, что для путешествия по различным пунктам меню вам потребуется еще клавиша <Выход> (как правило, на ней написано <Esc>), которая возвращает вас на шаг назад — к тому пункту меню, из которого вы сюда попали.

Наконец не осталось никаких препятствий для выполнения следующих заданий:

❸ Проверьте, совпадает ли база данных вашего класса с действительностью, и если нет — подправьте ее или создайте заново.

❹ Занесите в базу данных результаты последних спортивных соревнований в вашем классе. Если это сделают и остальные классы, легко узнать и чемпиона школы.

❺ Не забывайте с помощью базы данных поздравлять именинников.

❻ Если ваша база данных уже в меру заполнена, выясните, какие самые популярные имена в вашей школе.

❼ Придумайте еще несколько задач, которые можно решить с помощью этой базы данных.

§21. Рождение модели

В § 18 мы уже рассказывали, что к моделям человек обращается тогда, когда ему нужно решить какую-нибудь жизненную задачу. Это вовсе не те задачи, которые собраны в ваши школьные задачники по математике, физике, химии и т. д. Они потому и названы жизненными, что возникают в повседневной жизни человека. Что подарить другу на день рождения и как вывести страну из экономического кризиса? Чем лучше кормить рыбок в домашнем аквариуме и как уберечь планету от экологической катастрофы? Как организовать веселый школьный вечер и что надо сделать, чтобы улучшить жизнь в вашем городе, поселке, деревне? Разнообразие жизненных задач, возникающих перед человеком и обществом, огромно.

Что общего у всех таких задач? А то, что *их решение надо начинать с определения того, какие* ***факторы существенны*** *для задачи, а какими можно пренебречь, т. е. являются* ***несущественными***. Факторов, воздействующих на интересующий нас объект, процесс или явление, как правило, очень много, и человек не в состоянии даже перечислить их все. Поэтому и приходится выделять весьма неболь-

шое их количество — только те, которые представляются существенными.

Рассмотрим, к примеру, задачу о выборе подарка другу на день рождения. Желание друга на первый взгляд кажется здесь весьма существенным фактором. Но вы знаете, что он мечтает о собственном автомобиле. В действие вступает другой фактор — ваши финансовые возможности. И именно этот фактор, а не мечта друга может оказаться существенным при выборе для него подарка.

Итак, выбор существенных факторов — первый шаг к решению задачи. И значит, это первый шаг к созданию модели.

Поскольку нас интересуют в первую очередь информационные модели, то каждый выделенный нами *фактор нужно описать одним или несколькими параметрами*. Ясно, к примеру, что ваши финансовые возможности будут описываться одним числовым параметром — количеством денег, которые вы готовы выделить на приобретение подарка. А, скажем, финансовые возможности какого-нибудь банка, определяющие его жизнеспособность, будут описываться целым рядом параметров, в число которых входят и наличный фонд, и вложения в ценные бумаги, и предоставленные кредиты другим организациям и т. п.

Для фактора «Желание друга» соответствующий параметр может быть задан списком, содержащим названия этих желаний. Если же желание друга выражено в форме «Подарите мне что-нибудь красивое», то вряд ли удастся указать подходящий параметр. Понятие красоты, как и другие эстетические категории, *не формализуемо*, т. е. не может быть сведено к некой совокупности однозначно определенных параметров, действуя над которыми по заданному алгоритму можно дать ответ «Красиво» или «Некрасиво».

Вообще *процесс описания факторов с помощью параметров называется* ***формализацией***.

Что делать после того, как определены параметры, вы уже знаете: надо обнаруживать взаимосвязи между параметрами и описывать их на подходящем языке. Когда и эта работа проделана, у вас в руках новорожденная модель. Но несмотря на ее «малый возраст», она представляет собой мощный и единственный инструмент решения жизненной задачи. Задача, для которой построена модель, учитывающая существенные факторы, называется **хорошо поставленной**.

О задаче, для которой неизвестно заранее, какие факторы существенны, не выявлены параметры или не указаны связи между ними, иными словами, не построена соответствующая модель, говорят, что она **плохо поставлена**.

Умение хорошо поставить задачу — это *искусство* построения моделей. Ведь модель окажется удачной, если в ней будут учтены *все* существенные факторы и не будут присутствовать лишние, которые только усложняют модель, не добавляя никакой полезной информации.

ВОПРОСЫ И ЗАДАНИЯ

1. С чего начинается решение жизненной задачи?
2. Как представлены факторы в информационной модели?
3. Что такое формализация?
4. Приведите примеры факторов, которые могут быть формализованы, и факторов, которые формализовать нельзя.
5. Какую задачу называют плохо поставленной? Что нужно сделать, чтобы из плохо поставленной задачи получить хорошо поставленную?
6. Согласны ли вы с высказанным в объяснительном тексте параграфа тезисом, что построение модели для плохо поставленной задачи — это искусство, а не наука?
   Если да, то как бы вы его пояснили? Если нет, приведите аргументы, подтверждающие вашу точку зрения.
7. В математической модели равномерного прямолинейного движения фигурируют три параметра: скорость движения, время движения и пройденный путь. Приведите три примера жизненных задач, в которых два параметра по очереди были бы исходными данными, а третий — результатом.
8. Определите факторы, существенные для решения следующей задачи:
   Участок цеха по производству туристского снаряжения выпускает брезентовые палатки. Требуется определить количество брезента, нужное для выполнения участком месячного плана.
9. По заказу администрации парка культуры была изготовлена бронзовая статуя девушки с веслом. Определите те свойства статуи, которые существенны для решения каждой из следующих задач:
   а) перевезти статую из мастерской в городской парк;
   б) установить статую на площадке парка;
   в) увеличить посещаемость городского парка;
   г) продать статую с аукциона.
10. Завершите построение моделей, начатое при решении задачи из задания № 8.
11. Через иллюминатор затонувшего корабля требуется вытащить сундук с драгоценностями. Удастся ли это сделать? Постройте модель для решения этой задачи.
12. Укажите, какие факторы обычно считаются существенными при построении математических моделей следующих задач:
    а) К одной точке на нитях одинаковой длины подвешены два положительно заряженных шарика. Каков угол между нитями подвеса?
    б) В колбу с раствором едкого натра добавили раствор соляной кислоты. Какие продукты и в каком количестве оказались в колбе в результате реакции между этими веществами?
    Постройте соответствующие математические модели.

§ 22. Системный подход и информационные модели

Приведем две цитаты:

«...Как-то особенно тихо вдруг стало,
На небе солнце сквозь тучу играло.
Тучка была небольшая на нем,
А разразилась жестоким дождем!»

(Н. А. Некрасов. «Дедушка Мазай и зайцы».)

«...Ожидается малооблачная погода; возможен кратковременный дождь, гроза; ветер слабый, 1 — 2 м/с; температура воздуха 21 — 23 градуса тепла».

(Из сообщения метеослужбы.)

Если спросить, о чем идет речь в этих двух фрагментах текста, каждый ответит: о погоде. Более того, по всей видимости, об одном и том же состоянии погоды. Разница же заключается в способах описания данного явления: первое описание — художественное, второе — ... Если вы не догадались, как назвать второе описание, немного потерпите, мы все расскажем.

Но каким бы ни было описание погоды, это все равно не сама погода. В своей деятельности — художественной, научной, практической — человек всегда создает некий «слепок» того объекта, процесса или явления, с которым ему приходится иметь дело. Такую замену реального объекта, явления или процесса называют, как вы помните, моделированием.

Модели позволяют человеку сконцентрировать свое внимание на самом существенном в изучаемых объектах, процессах и явлениях. В этом главное назначение моделей вообще. Поэтому для человека, желающего посвятить себя какой-либо творческой деятельности, изучение уже известных человечеству моделей так же важно, как для спортсмена постоянные тренировки, накачивающие силу мышц. Можно без преувеличения сказать, что все образование — это изучение тех или иных моделей, а также приемов их использования.

Так, в школьном курсе физики рассматривается много разнообразных формул, выражающих зависимости между физическими величинами. Эти формулы представляют собой не что иное, как математические модели изучаемых явлений или процессов. Если вас просят решить физическую задачу, то вы начинаете, как правило, с поиска подходящей формулы, т. е. с подбора модели, которая отвечает требованиям вашей задачи. И значит, вы уже заранее предполагаете, что модель нужно искать в виде формулы.

Строгие правила построения моделей сформулировать невозможно. Однако человечество накопило богатый опыт в этой сфере деятельности. Один из эффективных подходов к построению мо-

делей был предложен в 1950 г. американским биологом Л. фон Берталанфи и затем получил развитие в самых различных направлениях.

Чтобы понять суть подхода, предложенного Берталанфи, вернемся еще раз к приведенным в начале параграфа двум описаниям погоды. Наверно, для каждой метеосводки при желании можно было бы подобрать подходящее художественное описание, однако, включая утром радио, вы, скорее всего, предпочтете услышать суховатое сообщение метеослужбы. Для решения ваших жизненных задач вторая модель более удобна: указание температуры подскажет вам, насколько тепло надо одеться, характер осадков — нужен зонт или дождь можно переждать, сила ветра — можно ли надеть шляпу с широкими полями... Удобство модели в данном случае определяется ее ярко выраженной *структурированностью*: в ней выделены основные элементы, которые необходимы для дальнейшего использования модели.

Другой пример. Вы собираетесь наметить маршрут будущего похода. Перед вами две модели местности — карта и набор фотографий. Скорее всего, вы предпочтете воспользоваться картой, хотя и небесполезно знать, как выглядят места, которые вам предстоит пройти. В чем различие между этими моделями? Фактически в том же самом, что и в описаниях погоды: на карте отчетливо выделены основные элементы местности — реки, дороги, просеки, овраги и т. п. Разглядывая фотографию, вам самим приходится выделять эти элементы.

Можно с уверенностью сказать, что большая часть моделей, которыми пользуется человек для решения жизненных задач, представляет собой некоторую совокупность элементов и связей между ними. Такие модели принято называть **системами**, а общие методы построения системных моделей — **системным подходом**. Основы системного подхода и заложил в своих трудах Л. фон Берталанфи.

Если теперь взглянуть на определение информационной модели (§ 19), то сразу станет ясно, что *всякая информационная модель обязательно является системной*. Элементами системы здесь выступают параметры, а связи между ними — это и есть связи системы. Поэтому построение информационной модели надо начинать с выделения существенных элементов и связей между ними, т. е. с построения подходящей (в рамках сделанных предположений) *системы*.

Приведем только один пример системного подхода при построении модели. Перед вами план г. Кенигсберга (сейчас он называется Калининград) в те времена, когда по нему гулял великий математик Леонард Эйлер (рис. 20, *а*). Его заинтересовал вопрос: можно ли, гуляя по городу, так выбрать маршрут, чтобы через каждый мост пройти ровно один раз? Л. Эйлер заметил, что в этой задаче речь фактически идет о множестве островов, между которыми ус-

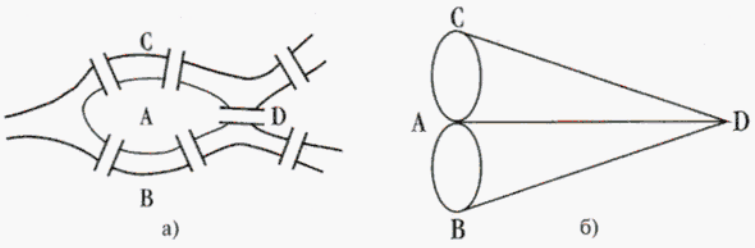


Рис. 20. Схема мостов Кенигсберга

тановлена связь — соединение мостом. Для простоты он изобразил острова и оба берега точками, а мосты — отрезками линий. Получился граф, соответствующий плану (рис. 20, *б*).

Решение задачи стало теперь почти очевидным — ведь, чтобы удовлетворить требованиям задачи, нужно иметь возможность, придя в какую-либо точку по одному отрезку, выйти из нее по другому отрезку (это не относится только к начальной и конечной точкам). Следовательно, из каждой точки, кроме, быть может, двух — начальной и конечной, должно выходить четное количество отрезков! Но для фигуры, изображенной на рисунке 20, *б* сформулированное условие, очевидно, не выполняется. Значит, и по мостам нельзя пройти, соблюдая требуемое условие.

Л. Эйлер

Если вдуматься в решение задачи, предложенное Л. Эйлером, то мы увидим здесь яркое проявление того, о чем шла речь выше, — модельного и системного подходов. Исходная задача заменена моделью, в которой существенно только то, соединены два острова мостом или нет. В самой модели ее составляющие четко структурированы: острова представлены как элементы (точки) и указано наличие связи между ними (отрезка) без какого-либо детального рассмотрения этих связей (железный мост или деревянный, на опорах или подвесной и т. п.). Иными словами, для решения задачи была построена системная модель.

ВОПРОСЫ И ЗАДАНИЯ

1. Что такое система? Что называют системным подходом?
2. Почему построение модели может оказаться полезным при решении конкретных задач?
3. Приведите примеры моделей, которые используются при изучении: а) физики; б) химии; в) биологии; г) географии; д) литературы.

4. Могут ли разные явления описываться одной и той же моделью? Если да, приведите пример.
5. Почему информационная модель всегда системна?
6. Можно ли утверждать, что всякая системная модель является информационной? Ответ «да» постарайтесь обосновать, ответ «нет» надо подтвердить примером системной, но не информационной модели.
7. а) Жизненная задача выбора жениха является плохо поставленной. Согласны ли вы с этим утверждением? Постарайтесь обосновать ответ.
   б) Агафья Тихоновна (персонаж пьесы Н. В. Гоголя «Женитьба»), решая жизненную задачу выбора жениха из числа представленных ей кандидатур, строит такую модель: «Если бы губы Никанора Ивановича да приставить к носу Ивана Кузьмича, да взять сколько-нибудь развязности, какая у Балтазар Балтазарыча, да, пожалуй, прибавить к этому еще дородности Ивана Павловича — я бы тогда тотчас же и решилась».
   Объясните, почему эта модель не может помочь в решении данной жизненной задачи.
   в) В каких случаях, на ваш взгляд, построение подобной модели может оказаться полезным при решении соответствующей жизненной задачи?
   г) Является ли модель, построенная Агафьей Тихоновной, системной?

§ 23. Динамические системы и черные ящики

Нередко при построении модели жизненной задачи одним из упрощающих предположений выступает неизменность выделяемой системы во времени, иными словами, **статичность** системы. Любая карта — типичная статичная системная модель: ведь она предполагает неизменность изображаемого ландшафта в течение некоторого времени.

Однако не меньший класс составляют жизненные задачи, в которых принципиально важен учет изменения во времени объектов или связей между ними. К таковым прежде всего относятся задачи, в которых фигурируют те или иные развивающиеся объекты, и задачи управления. Возникающие при постановке этих задач системы естественно назвать **динамическими**.

Изменение динамической системы во времени обычно называют ее **функционированием** или **эволюцией**. Рассматривая функционирование системы, нередко бывает достаточно сосредоточить свое внимание на внешних воздействиях, вызывающих изменения системы, и результатах этих воздействий. Внешние воздействия воспринимаются системой через ее **входы**, а результаты передаются системой во внешнюю среду через ее **выходы**. При этом для нас оказывается *несущественным внутреннее устройство системы*.

Рассмотрим, к примеру, действия человека, переходящего улицу на перекрестке. Он проверит наличие светофора, и если таковой имеется, то согласует свои действия с его сигналами. Кроме того, он учтет движение транспорта, который может подчиниться сигна-

лам светофора, а может и нет. Выберет необходимую скорость перемещения и направление. И наконец, перейдет улицу. Важно ли для нас в этом случае устройство органов зрения, слуха, опорно-двигательного аппарата и т. п.? Конечно, нет. Описывая переход улицы человеком, мы интересуемся только информацией, поступающей из внешней среды, т. е. подаваемой на входы, и действиями человека в соответствии с этой информацией, т. е. результатами на выходах.

Или вот другой пример, относящийся уже не к живой природе, а к техническим устройствам. Пусть перед нами автомат по продаже билетов на пригородные поезда. Мы опускаем в него монеты на нужную сумму (можно считать, что подаем ему на входы информацию), а он на выходе дает нам билет или сообщает, что сумма денег не соответствует стоимости билета. Интересует ли нас при этом, как он устроен внутри? Конечно, нет. Лишь бы он правильно работал, особенно если мы пришли на вокзал за несколько минут до отправления поезда.

Как видите, мы отказываемся от изучения и моделирования *внутренней* структуры объекта. Объект, внутреннее устройство которого принципиально скрыто от исследователя, был введен в кибернетике под названием **черный ящик**. Схематично черный ящик можно представлять себе так, как он изображен на рисунке 21.

Мы сейчас вступили в область информатики, общую с территорией другой науки — **кибернетики**. Эта наука, возникшая в XX в., изучает процессы управления в живой природе и системах, созданных человеком, а также разрабатывает методы построения эффективного управления при решении человеком тех или иных жизненных задач.

Мы не намерены разбирать пограничные споры, что именно принадлежит информатике, а что — кибернетике. С одной стороны, кибернетика как наука об управлении не может не изучать информационные процессы — какое же управление без информации? С другой стороны, далеко не всякий информационный процесс обязательно связан с каким-либо управлением. Вот и сейчас понятие черного ящика нас пока интересует не как средство для решения задач управления (речь об этом пойдет в главе 5), а исключительно как средство моделирования поведения тех или иных объектов.

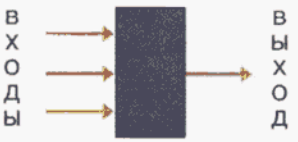


Рис. 21. Черный ящик

Почему же и здесь мы говорим о моделировании? Да потому, что, не зная, как устроен черный ящик, мы можем лишь *предполагать*, какую информацию он воспринимает на входах (иными словами, что для него существенно) и какой будет его реакция на те или иные входные сигналы. Догадку можно проверить, подавая на входы ту или иную информацию и наблюдая на выходах за реакцией черного ящика на эту информацию. Если наша догадка будет регулярно подтверждаться, то можно считать, что мы построили модель той динамической системы, которая представлена данным черным ящиком. На лабораторной работе № 15 вам предстоит потренироваться в таком разгадывании черных ящиков. Конечно, ящики эти носят учебный характер и придуманы нами, чтобы вы могли потренировать свою интуицию.

ВОПРОСЫ И ЗАДАНИЯ

1. Какая система называется динамической?
2. Как называют изменение динамической системы во времени?
3. Чем характеризуется черный ящик? Для моделирования каких динамических систем он используется?
4. Приведите известные вам примеры динамических моделей, используемых: а) в физике; б) в химии; в) в биологии.
5. Приведите пример использования метода черного ящика в исследованиях: а) по физике; б) по химии; в) по биологии.
6. Как вы думаете, может ли черный ящик не иметь входов, но иметь выходы?

Расшифровка черного ящика

Перед вами «коллекция» черных ящиков. Каждый из них имеет от одного до трех входов и один или два выхода. На каждый **вход** вы будете подавать какое-либо сообщение, т. е. последовательность символов — она может иметь смысл (например, быть числом или словом русского языка) или не иметь такового, а на **выходах** будете получать от черного ящика новые сообщения. Ваша задача — определить, что именно делает данный черный ящик.

Кроме количества входов и выходов, каждый черный ящик охарактеризован еще уровнем сложности. Конечно, уровень сложности — вещь субъективная, зависящая от того, кто разгадывает. Не исключено, что черный ящик, который мы оценили как сложный, для вас окажется совсем простым. Или наоборот.

Теперь опишем, как работать с программой «Черный ящик». Из заставки (рис. 22) вы переходите на страницу «Установка параметров» (рис. 23).

Рис. 22. Заставка программы «Черный ящик»

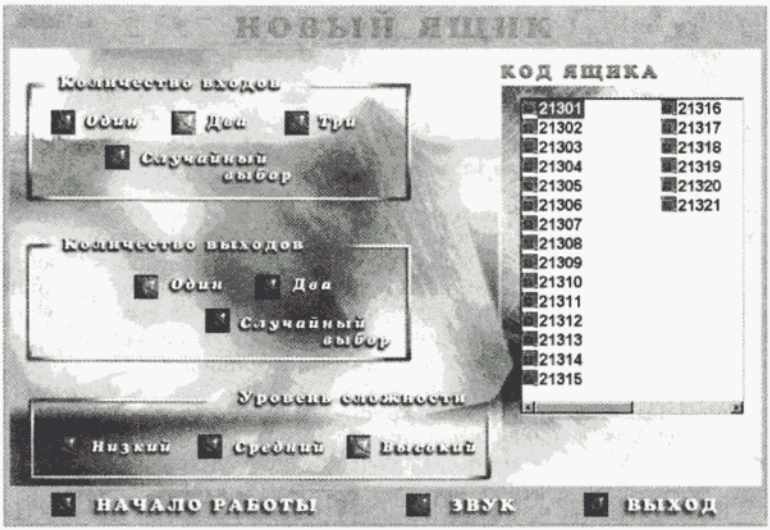


Рис. 23. Режим выбора параметров черного ящика

Рис. 24. Учебный черный ящик

Вы видите три группы кнопок, позволяющих выбрать количество входов и выходов, а также уровень сложности. Если вы щелкните по кнопке «Случайный выбор», то параметры черного ящика устанавливаются случайным образом. И потом не жалуйтесь, если вам попался какой-то слишком сложный вариант!

Каждый ящик имеет код, цифры которого означают количество входов и выходов данного ящика, его уровень сложности и порядковый номер среди всех ящиков данного типа. Вы можете выбрать себе ящик и по коду (если, например, на предыдущем занятии вы начали разгадывать какой-то черный ящик, не успели закончить и хотите продолжить работу).

Установив параметры, вы переходите на страницу, где располагается сам черный ящик (рис. 24). В верхней строке экрана вы видите два режима работы с черным ящиком: «Тестирование» и «Проверка гипотезы».

В режиме тестирования вы подаете сообщения на входы и нажимаете на кнопку «Обработать». Совсем не обязательно, что вы угадаете, какого вида информацию надо было подать на входы. Например, вы ввели последовательность русских букв, а надо было на этот вход ввести число. Черный ящик отреагирует на такую ситуацию словами «Не понимаю». Значит, ваша первая задача — *определить, какого вида информацию надо вводить на каждый из входов черного ящика*.

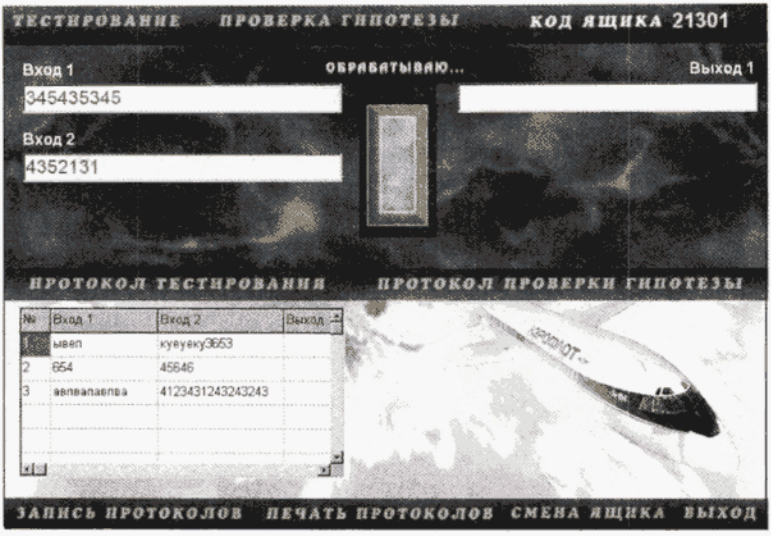


Рис. 25. Вид черного ящика с протоколом тестирования

Но вот вы определили, что, скажем, на первый вход надо вводить числа, а на второй — последовательности символов русского алфавита. Ввели нечто такое, а на выходе реакция «Не могу обработать». Это означает, что, хотя по отдельности информация на каждом входе черному ящику понятна, эти данные не совместимы для обработки. К примеру, черный ящик выбирает из последовательности букв русского алфавита букву, стоящую в этой последовательности на месте, указанном данным числом. Последовательность вы ввели МАМА, а число указали 8. Ясно, что в этой ситуации черный ящик просто не может дать никакого ответа. Так что не надо пугаться такой реакции черного ящика и думать, что он сломался. Вам надо попытаться определить, каковы ограничения на входную информацию.

Наконец, вы получили на выходах реакцию на введенную вами информацию. Теперь надо подумать, по какому принципу черный ящик обрабатывает входную информацию. Одного теста может оказаться недостаточно. Испытания придется повторить несколько раз, меняя информацию на входах. Чтобы результаты испытаний были у вас перед глазами, компьютер ведет протокол тестирования (рис. 25). Вы можете его в любую минуту посмотреть и даже распечатать или записать в файл, чтобы воспользоваться им в следующий раз.

Но вот вы, как вам кажется, догадались, что делает данный черный ящик. Говоря научным языком, у вас сформировалась *гипотеза*.

Ее надо проверить. Для этого и предусмотрен режим проверки гипотез. Включите этот режим.

Теперь компьютер сам будет подавать информацию на входы, а вы, приняв на себя роль черного ящика, должны записать на выходы результат ее обработки. Если ваша гипотеза верна, то черный ящик сообщит вам о совпадении ваших результатов с его результатами. Ясно, что гипотеза будет проверяться не на одном примере, а на нескольких. Чтобы видеть все результаты проверки сразу, предусмотрен протокол проверки гипотезы. Его тоже можно в любую минуту посмотреть, распечатать или сохранить в файле.

А теперь приступайте к работе. Желаем успеха! Программа содержит более 100 черных ящиков, так что заниматься их расшифровкой вы можете в течение не одного, а нескольких занятий.

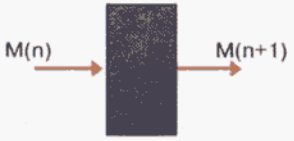
§24. Модель неограниченного роста

В 1937 г. на остров Протекшен завезли 8 фазанов. Никто на этих фазанов не охотился (ни люди, ни звери), корма для них было вдоволь, и через год фазанов стало 26. Прошел еще год, фазанов снова пересчитали — на этот раз их было 83. Интересно, сколько будет фазанов через n лет?

Мы намерены использовать компьютер для исследования того, как растет число фазанов на острове, значит, надо строить компьютерную модель. А построение любой модели начинается с выделения существенных факторов.

Конечно, факторов, влияющих на жизнь фазанов, много. Это и климатические условия, и даже конкретная погода в то или иное время года — скажем, холодная зима или засушливое лето отрицательно скажутся на росте популяции фазанов. Появление или быстрый рост вида животных, имеющих ту же кормовую базу, что и фазаны, тоже не будет способствовать процветанию фазаньего сообщества. Хорошо еще, что по условию на них никто не охотится.

Короче говоря, мы в принципе не можем учесть все существенные факторы, влияющие на жизнь фазаньего племени. Поэтому естественно рассмотреть воздействие окружающей среды на численность популяции фазанов как черный ящик. У него один вход — численность фазанов в некотором году и один выход — численность фазанов в следующем году. Если обозначить количество фазанов по истечении n лет через $M(n)$, то схематично данный черный ящик можно изобразить так:



Разгадать этот черный ящик, мы, конечно, не можем — на то он и черный. Но естественно предположить, что прирост числа фазанов за единицу времени пропорционален уже их имеющемуся количеству. Такое предположение согласуется с обычными представлениями о размножении: чем больше живых организмов, тем больше у них потомков. А окружающая среда (т. е. наш черный ящик) выступает как регулятор прироста количества фазанов.

Высказанные предположения показывают, что для нашей модели задачи существенна следующая информация: начальное количество (обозначим его $M(0)$) и коэффициент прироста за 1 год (обозначим его k).

Пусть по истечению n лет количество фазанов достигло, как мы договорились, величины $M(n)$. Тогда прирост за 1 год составит $M(n+1) - M(n)$.

Значит, высказанные нами предположения можно записать равенством

$$M(n+1) - M(n) = kM(n).$$

Преобразуя это равенство, получаем

$$M(n+1) = M(n) + kM(n)$$

или

$$M(n+1) = (1 + k)M(n).$$

По этой формуле, зная начальное количество $M(0)$ и коэффициент прироста k, можно сначала найти $M(1)$, т. е. число фазанов, которое будет через год, затем найти $M(2)$ (количество фазанов через два года), затем $M(3)$ и т. д.

Те из вас, кто дружен с математикой, без труда распознают в последовательности $M(n)$ геометрическую прогрессию. Обнаруженную закономерность можно сформулировать так: если действие окружающей среды сказывается лишь на скорости прироста, то живые организмы размножаются в геометрической прогрессии. И совершенно неважно, идет ли речь о фазанах или китах, о водорослях или гигантских секвойях. Мы построили модель некоторого природного процесса, и при этом не играет роли, какие именно живые организмы участвуют в этом процессе. Построенную модель называют **моделью неограниченного роста**.

ВОПРОСЫ И ЗАДАНИЯ

1. Какие предположения положены в основу модели неограниченного роста? Каковы параметры этой модели?
2. К какому виду информационных моделей относится модель, построенная в объяснительном тексте параграфа?
3. Как растет масса живых организмов, если действие окружающей среды сказывается лишь на скорости прироста массы?
4. Почему при построении модели неограниченного роста оказалось полезным использовать понятие черного ящика?

§25. Выбираем средство информационных технологий

Скорее всего, вы уже ответили на второй вопрос к § 24 — нами построена *математическая модель* задачи. Действительно, все параметры в ней числовые, а связи между параметрами выражаются функцией.

Теперь мы хотим «испытать» эту модель с помощью компьютера. Мы объявили, что построенная нами модель годится для любых живых организмов. Да и остров Протекшен вместе с живущими на нем фазанами вряд ли сильно волнует нас, живущих в российской действительности. Гораздо интереснее нам леса и степи, тундра и пустыни... Как, скажем, меняется масса растений в этих зонах? Ведь растения — это тоже живое, и они тоже размножаются. С растениями нам и рассуждать будет немного проще — мы можем говорить о массе растений (скажем, в тоннах), произрастающих на той или иной территории, а не об их количестве (в штуках, как с фазанами). Поэтому предусмотрительно запасемся значениями коэффициента размножения k, экспериментально полученными учеными-биологами для растений в разных природных зонах. Значение этого коэффициента для различных природных зон приведено в таблице 10.

Таблица 10

Природная зона	Тундра	Тайга	Степь	Пустыня
Коэффициент k	0,6	1,8	1,2	0,8

Теперь, глядя на эти числа, попытайтесь предсказать, через сколько лет масса растений превысит 100 т, если первоначально масса растений на некотором участке была всего 1 т. Запишите ваш прогноз, чтобы потом сравнить его с результатом, полученным при помощи компьютера.

Чтобы провести компьютерный эксперимент, нужно выбрать еще, с помощью какой конкретно информационной технологии он будет реализован. Поскольку все параметры модели числовые, то и компьютерную технологию надо выбрать так, чтобы она позволяла обрабатывать числа. Например, можно взять электронную таблицу.

Прежде всего надо в таблице 11 записать исходные данные. Для них мы отведем первые три строки (они выделены цветом).

Таблица 11

A	B	C	D	E	F
Год	Природная зона	Тундра	Тайга	Степь	Пустыня
	Коэффициент размножения k	0,6	1,8	1,2	0,8
0	Начальная масса $M(0)$	1	1	1	1
	Масса через 1 год				
	Масса через 2 года				

Теперь надо организовать вычисление массы растений через год, через два года, через три года и т. д. Будем эти вычисления производить в столбцах C, D, E и F. По две формулы в столбцах C и D мы записали в таблицу; кроме того, в столбец A записано вычисление года. Разберитесь с этими формулами и догадайтесь, как выглядят формулы в других клетках таблицы 12.

Таблица 12

A	B	C	D	E	F
Год	Природная зона	Тундра	Тайга	Степь	Пустыня
	Коэффициент размножения k	0,6	1,8	1,2	0,8
0	Начальная масса $M(0)$	1	1	1	1
A3+1	(Масса через 1 год)	C3*(1+C2)	D3*(1+D2)		
A4+1	(Масса через 2 года)	C4*(1+C2)	D4*(1+D2)		
A5+1					

Надеемся, что теперь вы готовы к проведению лабораторной работы. Но прежде оглянемся на путь, пройденный нами в этом параграфе. Чем мы занимались и что мы получили? Мы перестраивали математическую модель, созданную в предшествующем параграфе, в модель, к которой применимы компьютерные технологии. *Модели, которые реализованы посредством компьютера, мы будем называть* ***компьютерными моделями***.

ВОПРОСЫ И ЗАДАНИЯ

1. Какую модель называют компьютерной?
2. Запишите формулы, которые надо занести в клетки E4 и F4.

Неограниченный рост

❶ Загрузите электронную таблицу и занесите в нее необходимые данные так, как они даны в таблице 12, и то, что вы придумали, выполняя задание 2.

Как же узнать, через сколько лет масса растений превысит 100 т? Представим, что в таблице заполнены нужным образом все строки, начиная с 4-й. Тогда достаточно в столбце C найти место, когда первый раз встретится число, большее 100, и в той же строке в первом столбце мы увидим год, в котором масса растений превысит 100 т. Точно так же можно поступить с тремя последующими после C столбцами.

Конечно, заполнить *все* строки таблицы сразу — слишком трудоемкая, да и, наверно, ненужная работа. Проще поступить так: последовательно копировать блок ячеек A4:F4 в последующие строки. (Подумайте, адреса каких ячеек при таком копировании не должны меняться. Кроме того, нет смысла копировать один и тот же комментарий, записанный в ячейку B4; именно поэтому мы записали его в круглых скобках.) Как только во всех четырех столбцах C, D, E и F возникнут числа, большие 100, копирование можно прекратить.

❷ Проделайте указанную работу с таблицей. Для каждой природной зоны запишите год, когда произошло превышение числа 100. Сравните получившиеся данные с вашим прогнозом, записанным вами ранее.
Ну как? Удалось угадать?

Теперь после первого опыта попробуйте угадать, когда масса растений станет равной 1000 т.

❸ Составьте прогноз для этого случая и снова проверьте его с помощью электронной таблицы. (Интересно, улучшились ли ваши прогностические способности?)

Скорее всего, теперь ваш прогноз оказался завышенным (как и у авторов этой книги, проводивших такой же компьютерный эксперимент). Удивительный факт, не правда ли: масса увеличилась в 10 раз, а на ее «производство» потребовалось всего лишь два-три дополнительных года. Полученные результаты тоже запишите к себе в тетрадь.

На самом деле (убедитесь в этом сами и внесите результаты в таблицу) и на следующее «удесятерение» массы живых организмов понадобится тоже два-три года. Вот в чем сила геометрической прогрессии!

❹ Каждая современная электронная таблица позволяет автоматически построить график или диаграмму зависимости между величинами, вычисленными с ее помощью. Воспользуйтесь этой возможностью и посмотрите графически зависимость массы растений от числа прошедших лет (для каждой из природных зон). Как это делается в вашей таблице, вам расскажет учитель или вы узнаете из инструкции пользователю.

Столь стремительный рост массы растений заставляет задуматься: а что будет, когда пройдет 15, 20 и более лет? Не случится ли так, что эта масса превысит массу планеты? Конечно, масса планеты — это не 100, не 1000, даже не 100 000 т. Например, масса Земли 5 976 000 000 000 000 000 000 т!

❺ Попытайтесь с помощью той же электронной таблицы вычислить, через сколько лет масса растений превысит массу Земли.

От полученного результата волосы могут встать дыбом. В течение жизни одного поколения людей вся планета превратится в «зеленое море» растений! Есть над чем призадуматься... Видно, не все удачно в построенной нами модели.

§ 26. Модель ограниченного роста

Конечно, ни при каких, даже самых благоприятных, условиях масса растений не может превысить массу планеты — ведь это нарушало бы фундаментальный закон сохранения вещества. Модель неограниченного роста, которую мы построили и изучили, оказалась пригодной, как и следовало ожидать, лишь пока выполнено главное предположение — действие всех факторов выступает ограничителем только скорости прироста массы. Видимо, в нашем черном ящике, описывающем действие природных факторов, надо что-то подправить.

Поступим так. По-прежнему не вдаваясь в подробности, как именно факторы окружающей среды влияют на жизнь организмов (ящик-то все равно остается черным!), выдвинем предположение, что имеется некоторое предельное значение массы растений, «проживающих» на той или иной территории. Так, ученые показали, что запас массы растений не может превосходить 20 т на гектар в полярной зоне, 350 т на гектар в лесной зоне, 440 т на гектар в тропиках. А на всей Земле масса растений не может превысить $5 \cdot 10^{12}$ т.

И еще одно наше предположение будет таким: чем ближе масса живых организмов к своей максимально возможной, тем меньшим становится коэффициент прироста k. В качестве обоснования для такого предположения вы могли бы использовать свои собственные наблюдения (на пришкольном участке, в саду): вначале растения

быстро набирают массу, затем их рост становится все медленнее. Иными словами, коэффициент k не является неизменной величиной, а зависит от разности $L - M(n)$, где L — предельное значение массы растений на данной территории. Самая простая функция, как вы, конечно, знаете, — это прямая пропорциональность. Будем поэтому считать, что коэффициент прироста меняется по формуле

$$k(n) = a(L - M(n)) \text{ для } n = 0, 1, 2, 3, \ldots .$$

Подводя итог проделанной нами работе по уточнению модели, опишем получившуюся модель роста массы живых организмов.

Предположения:

- прирост массы живых организмов за единицу времени пропорционален уже имеющейся массе;
- существует некоторое предельное значение массы живых организмов;
- коэффициент прироста массы живых организмов за единицу времени пропорционален разности между максимально возможным значением массы и массой, имеющейся к данному моменту времени.

Параметры модели:

- начальная масса живых организмов $M(0)$;
- предельное значение массы живых организмов L;
- коэффициент пропорциональности a в формуле для коэффициента прироста;
- время n.

Связь между параметрами модели задается соотношением

$$M(n+1) = M(n) + aM(n)(L - M(n)),$$

где через $M(n)$ по-прежнему обозначена масса живых организмов по истечении n лет.

Эту модель принято называть **моделью ограниченного роста**.

ВОПРОСЫ И ЗАДАНИЯ

1. Какие факторы ограничивают массу живых организмов? Приведите примеры.
2. Какие предположения положены в основу модели ограниченного роста?

Ограниченный рост

Компьютерный эксперимент с моделью ограниченного роста будем проводить, также используя электронную таблицу. Для этого заполним таблицу 13 следующим образом:

Таблица 13

A	B	C	D	E	F
Природная зона	Год	Тундра	Тайга	Степь	Пустыня
Коэффициент размножения k		0,6	1,8	1,2	0,8
Предельное значение массы L					
Коэффициент a					
Начальная масса $M(0)$	0	1	1	1	1
Масса через 1 год	B5+1	C5+C4*C5*(C3–C5)			
Масса через 2 года	B6+1				
	B7+1				

Как обычно, цветом выделены ячейки таблицы, куда надо заносить исходные данные.

❶ Заполните электронную таблицу нужным образом.

Возможно, у вас возник вопрос: где взять значения для коэффициента a? Из формулы $k(n) = a(L - M(n))$ видно, что для нахождения коэффициента a достаточно разделить $k(0)$ на $(L - M(0))$.

Число $k(0)$ — это коэффициент прироста в самом начале процесса, и потому его нужно взять равным тому самому числу k, которое фигурировало в предыдущем параграфе. Поэтому заносить в таблицу в качестве исходного данного можно по-прежнему k (что и сделано в клетках второй строки), а коэффициент a вычислять по соответствующей формуле:

$$a = \frac{k}{L - M(0)}.$$

❷ Внесите в клетки C4, D4, E4 и F4 электронной таблицы формулы, по которым вычисляется значение коэффициента a для каждой из природных зон.

Значение массы $M(0)$ мы возьмем прежним (1 т). Значение предельной массы L возьмем, к примеру, 11 000 т.

❸ Введите в таблицу исходные данные и, подбирая значение n, найдите, через сколько лет масса растений станет больше 100 т. Для этого можно просто последовательно копировать блок B5:F5 в последующие строки.

Сравните получившиеся у вас четыре результата с результатами предыдущей лабораторной работы. Если все было сделано без оши-

бок, то вы увидите, что ее результаты совпадают с полученными результатами.

❹ А теперь по той же таблице найдите, через сколько лет масса растений станет больше 1000 т. Снова сравните получившиеся у вас четыре результата с результатами предыдущей лабораторной работы.

Изменятся ли существенно результаты, если мы возьмем, как и в предыдущей работе, массу S равной 10 000 т? Иными словами, останется ли по-прежнему время «удесятерения» равным двум-трем годам?

❺ Проведите соответствующие эксперименты, используя заполнение электронной таблицы, подготовленное вами при выполнении задания 4, и найдите, через сколько лет масса растений станет больше 10 000 т. Снова сравните получившиеся у вас четыре результата с результатами предыдущей лабораторной работы.

Как видите, ситуация изменилась: время «удесятерения» растет, причем быстрее для тех растений, где коэффициент k меньше. Это неудивительно: чем медленнее рост, тем большее время требуется для «удесятерения» массы.

❻ Подготовьте заполнение электронной таблицы и постройте с ее помощью графики зависимости массы растений от числа прошедших лет. Какие общие особенности этих графиков вы смогли обнаружить?

§ 27. Самостоятельная жизнь информационной модели

Если человеку каждый раз, столкнувшись с очередной жизненной задачей, приходилось бы с нуля строить модель, то едва ли прогресс человечества достиг бы сегодняшних высот. Разумеется, каждый человек и общество в целом опирается на опыт предшествующих поколений, который зафиксирован в моделях, описанных тем или иным способом. Легенды и мифы отражают существовавшие модели мироздания, притчи и поучения — модели общественного поведения. На смену этим моделям пришли естественно-научные теории, описывающие природные процессы, и теории общественного устройства, в которых их авторы пытались объяснить как существующие общества, так и гипотетические (различные утопии).

Почему же, однажды родившись, модели не живут вечно? Некоторые из них, как бабочки-поденки, исчезают, едва появившись на свет. Другие живут столетиями. Но даже модели, построенные лучшими умами человечества, все равно сменяются другими. Что управляет этой сложной жизнью моделей в человеческих знаниях?

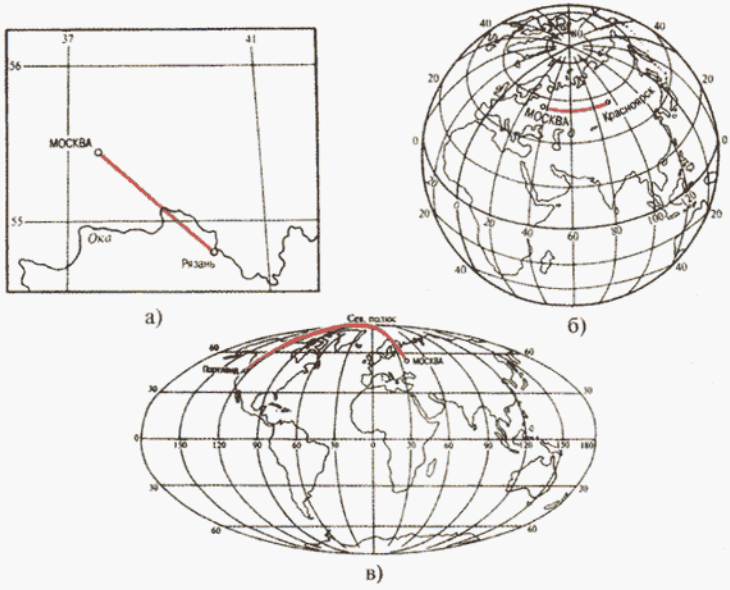


Рис. 26. Различные модели земной поверхности, используемые для измерения расстояния

Напрашивается ответ: растут знания человека об окружающем его мире, вот и меняются модели.

Это в некотором смысле верно. Мы ведь уже обсуждали, что именно в моделях представлено все, что человечество знает о себе и окружающем мире. Поэтому точнее было бы сказать, что изменение моделей обеспечивает рост знаний.

Чтобы приблизиться к ответу на поставленный вопрос, вспомним (см. § 7), что к моделям человек обращается тогда, когда ему нужно решить какую-нибудь жизненную задачу. Вот мы и начнем с разбора некоторой жизненной задачи.

Пусть вам нужно узнать расстояние между двумя не очень удаленными друг от друга населенными пунктами, скажем, от Москвы до Рязани. Вы берете карту, прикладываете линейку и затем измеренную длину отрезка умножаете на масштаб карты (рис. 26, *а*). Короче говоря, вы считаете поверхность Земли плоскостью. Если это два далеких города, скажем, Москва и Красноярск, то, проложив по карте прямолинейный маршрут, вы ошибетесь на сотни километров. Чтобы получить ответ, близкий к истинному, необходимо учесть, что Земля круглая, и измерять расстояние надо не по прямой, а по дуге большого круга (рис. 26, *б*). Наконец, если речь идет о трансконтинентальных путешествиях, то придется еще раз сменить мо-

дель — вместо шара рассматривать геоид (шар, сплюснутый у полюсов). Именно поэтому В. Чкалов свой перелет в Америку совершал через Северный полюс (рис. 26, *в*). Как вы видите, смена модели продиктована здесь требованиями практики.

Если построенная модель дает удовлетворительные результаты при решении задачи, то говорят, что модель **адекватна** рассматриваемому объекту (процессу или явлению).

Принцип адекватности говорит еще и о том, что *никакая модель не эквивалентна самому объекту (процессу или явлению)*. Но при решении конкретной задачи, когда нас интересуют сравнительно немногие свойства изучаемого объекта (процесса или явления), построение модели оказывается очень полезным, а подчас и единственным инструментом исследования.

Проблема адекватности — одна из самых трудных. И согласованность модели с практикой (в частности, экспериментом) — только один из критериев адекватности. Скажем, модель неограниченного роста, рассматривавшаяся в § 24, хорошо согласуется с практикой, пока масса живых организмов остается достаточно малой по сравнению с предельно допустимой массой. В некоторых случаях (когда коэффициент прироста невелик и мала начальная масса) это условие выполняется годами, так что экспериментально опровергнуть такую модель довольно трудно. И вообще модель неограниченного роста остается адекватной до тех пор, пока выполняется главное предположение, принятое при ее построении, — природные условия выступают фактором, определяющим *только* скорость прироста живых организмов. Отметим заодно, что введенный нами тогда же коэффициент k выступает параметром, описывающим действие этого фактора.

Мы заменили эту модель моделью ограниченного роста, когда в результате теоретических рассмотрений (в данном случае компьютерных вычислений, хотя неограниченный рост массы в этой модели можно было предвидеть, исходя из математических свойств геометрической прогрессии) выявилось нарушение фундаментального закона природы — закона сохранения массы.

Таким образом, *смена модели может происходить и в силу того, что она не согласуется с более общими законами, открытыми человеком при исследовании природы и общества.*

Чтобы еще раз проиллюстрировать этот тезис, расскажем историю одной модели — историю, длящуюся уже много столетий. Вопросы космического устройства мира всегда волновали человека. Космос притягивает внимание человека не только своей романтической безграничностью и загадочностью, но и чисто практическими проблемами: как происходит смена времен года, когда ожидать солнечные и лунные затмения, как по Солнцу и звездам определить свое местонахождение и т. д. Для этого необходимо было иметь модель пусть не всего космоса, но хотя бы того ближайшего к Земле окружения, которое сейчас именуется Солнечной системой. Оста-

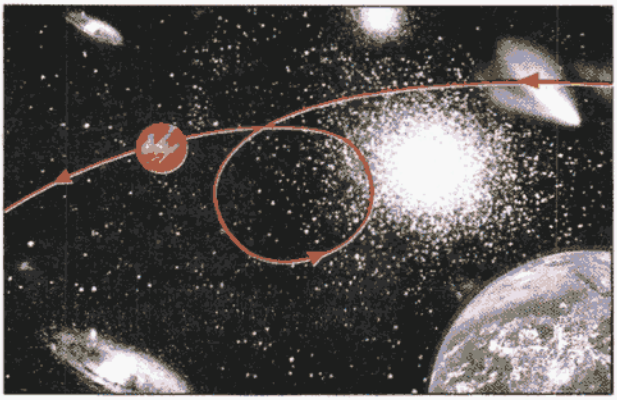


Рис. 27. «Попятное» движение планеты

вим в стороне представления древних народов о Земле, покоящейся на слонах и черепахах, и поговорим о планетарных моделях.

Одной из самых известных и продержавшихся в истории науки почти 14 столетий была модель, предложенная во II в. древнегреческим астрономом и математиком Клавдием Птолемеем. В этой модели, как вы, наверно, знаете из истории науки, в центре располагалась неподвижная Земля, а все планеты (открытые к тому времени) и Солнце вращались вокруг нее по круговым орбитам. Точнее, по круговым орбитам вращались вокруг Земли только Луна и Солнце, а для других планет движение было более сложным — ведь при круговом движении планет их перемещение среди звезд должно быть таким же, как у Солнца или Луны, только «вперед». А наблюдения показывали так называемое «попятное» движение планет. На рисунке 27 цветными стрелками показан видимый путь планеты среди звезд (видна петля «попятного» движения планеты).

И. Кеплер

Для объяснения такого движения Птолемей предположил, что каждая планета вращается по некоторой окружности, центр которой как раз и движется по круговой орбите вокруг Земли (рис. 28). Птолемею удалось так подобрать радиусы всех фигурирующих в этой модели окружностей, что она прекрасно согласовывалась с астрономической практикой вплоть до XVI в.

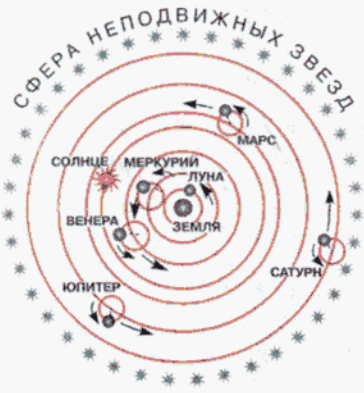


Рис. 28. Система Птолемея

Рис. 29. Система Коперника

Когда Николай Коперник в 1543 г. предложил свою гелиоцентрическую модель (рис. 29), то он мотивировал свою модель прежде всего тем, что в ней проще производить астрономические вычисления. Компьютеров в то время не было, и простота вычислений — важный аргумент в борьбе моделей.

Следующий шаг в истории моделей Солнечной системы принадлежит Иоганну Кеплеру. Сначала И. Кеплер, опираясь на наблюдения датского астронома Тихо-Браге, в 1609 г. формулирует первые два своих закона движения небесных тел, второй из которых как раз и гласит, что орбита такого тела представляет собой коническое сечение (т. е. эллипс, параболу или гиперболу), в одном из фокусов которого находится Солнце. Спустя 10 лет И. Кеплер сформулировал третий закон движения небесных тел.

И. Ньютон

Прошло еще почти 70 лет, и в 1687 г. Исаак Ньютон в своем сочинении «Philosophiae naturalis principia mathematica» на основании трех законов Кеплера выводит формулу для силы притяжения планеты Солнцем и затем, используя эту формулу, приходит к формулировке знаменитого закона всемирного тяготения. Но после того как этот закон (справедливый отнюдь не только для космических тел) был открыт, он стал основой всей небесной механики и теоретической астрономии. Теперь уже из закона всемирного тяготения стали выводить законы

Рис. 30. Искривление траектории луча вблизи массивного тела

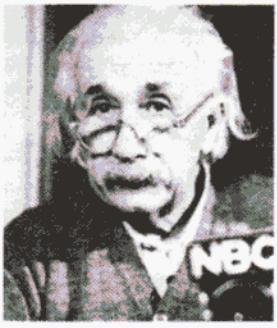
Кеплера. И что же оказалось? Выяснилось, что сами законы Кеплера, сыгравшие принципиальную роль в выводе формулы великого закона, справедливы только в том случае, если вокруг Солнца вращается ровно одна планета.

Эта история показывает, что эмпирически построенная модель (в данном случае три закона Кеплера), сомневаться в адекватности которой нет никаких оснований (ибо родилась она из практики астрономических наблюдений), способна после теоретической «обработки» породить модель, которая однозначно укажет на неадекватность исходной модели, или, точнее говоря, резко сузит ее область адекватности.

А что же дальше? В начале XX в. была сформулирована корпускулярно-волновая теория элементарных частиц, согласно которой свет — это не только волна, но и частица. Но тогда в соответствии с законом всемирного тяготения частицы света — их назвали фотонами, — проходя мимо космических тел, должны к ним притягиваться. Это значит, что траектория их движения вблизи таких тел заметно искривляется (рис. 30). Астрономические наблюдения этот вывод подтвердили.

Но мы привыкли, что луч света всегда движется по пути, кратчайшему между точками. Что же получается: прямая не является кратчайшим путем между двумя точками! Но так ли это удивитель-

но? Ведь, к примеру, на поверхности шара кратчайший путь тоже не отрезок между точками, а дуга окружности большого круга. Там по прямой нам мешает двигаться вещество шара. А в пространстве движению «напрямик» мешает сила притяжения между материальными телами, открытая еще Ньютоном. Получается, что геометрия пространства, т. е. вдоль каких линий надо измерять расстояние между двумя точками, зависит от распределения в этом пространстве материальных масс. Эта новая модель пространства, отличная от привычной нам геометрии Евклида, была построена А. Эйнштейном и получила название общей теории относительности. Видите: закон всемирного тяготения, выведенный Ньютоном в предположении, что наше пространство евклидово (неевклидовой геометрии во времена Ньютона не было!), привел к появлению модели, в которой геометрия пространства неевклидова. История повторяется...



А. Эйнштейн

ВОПРОСЫ И ЗАДАНИЯ

1. Что такое адекватность модели?
2. Какие причины могут вызывать смену модели?
3. Приведите известные вам из истории науки примеры смены одних моделей на другие, более адекватные.
4. а) Назовите параметры модели неограниченного роста. Какие существенные факторы описываются этими параметрами?
   б) Выполните задание а) для модели ограниченного роста.

§ 28. Границы адекватности модели

В предыдущем параграфе мы сформулировали важный принцип: никакая модель не эквивалентна исходному объекту, процессу или явлению. Ведь в любой модели мы можем учесть только часть информации об объекте, процессе или явлении. На наш взгляд, самую существенную, но все равно не всю.

Но тот же принцип можно сформулировать иначе: *всякая модель имеет ограниченную* ***область адекватности***, и за пределами этой области она перестает удовлетворительно отражать свойства моделируемого объекта. Поэтому и применять модель для решения той или иной жизненной задачи допустимо только тогда, когда мы убедились, что не вышли за границы области адекватности.

Как же проверить, что выбранная нами модель применима? Прежде всего надо убедиться, что все факторы, существенные для

данной задачи, присутствуют в модели. Затем надо проверить, что в исходных данных задачи значения параметров, описывающих действие факторов, не выходят за границы адекватности модели.

К сожалению, довольно часто эти требования игнорируются. Мы, к примеру, нередко слышим с самой высокой трибуны предложения использовать в России экономические модели, прекрасно работающие в западных странах. Как правило, таким предложениям не предшествует анализ, выполнены ли у нас те базовые предпосылки, на которых основываются предлагаемые модели. Ведь даже внешняя схожесть ситуаций еще не означает ни совпадения факторов, ни попадания в область адекватности.

О выделении существенных факторов мы уже говорили в § 21. Теперь обсудим, как находить границы адекватности. Как вы помните из предыдущего параграфа, адекватность модели определяется ее согласованностью с практикой и общетеоретическими положениями. Предложение вам провести какой-либо **натурный эксперимент** для проверки адекватности той или иной модели, конечно, выходит за рамки курса информатики. Поэтому наш эксперимент будет **компьютерным**, опирающимся на общетеоретические положения.

В § 26 мы упомянули, что модель неограниченного роста остается адекватной, пока масса живых организмов достаточно мала по сравнению с предельно допустимой массой этих организмов в данных природных условиях. Попытаемся определить, насколько мала должна быть исходная масса живых организмов по отношению к предельной массе, чтобы модель неограниченного роста оставалась адекватной в течение нескольких лет (напомним, что существование предельного значения массы — следствие общетеоретических положений). Иными словами, мы хотим найти границы адекватности модели неограниченного роста.

Выполнив задание 4 предыдущего параграфа, вы определили параметры, участвующие в моделях ограниченного и неограниченного роста. Вспомним заодно и связи между этими параметрами.

Модель неограниченного роста: начальная масса $M(0)$, коэффициент прироста k, число лет n, масса живых организмов через n лет $M(n)$; связь между параметрами определяется формулой

$$M(n+1) = (1 + k)M(n).$$

Модель ограниченного роста: начальная масса $M_O(0)$, коэффициент прироста k, предельное значение массы L, число лет n, масса живых организмов через n лет $M_O(n)$ — буква O внизу показывает, что вычисление массы идет в модели ограниченного роста; связь между параметрами, найденная в лабораторной работе 17, определяется формулой

$$M_O(n+1) = (1+k\,\frac{L-M_O(n)}{L-M(0)})M_O(n).$$

Поскольку $M_O(0) = M(0)$, то нетрудно подсчитать, что $M_O(1) = M(1)$, т. е. через год масса живых организмов, подсчитанная по обеим моделям, будет еще одинаковой. Но вот $M_O(2)$ будет уже меньше, чем $M(2)$. И чем дальше, тем больше будет различие между значениями M_O и M. Значит, надо договориться, какое расхождение между M_O и M мы будем считать еще допустимым. Пусть, к примеру, мы считаем модель неограниченного роста адекватной, если разница $M - M_O$ составляет не более 10% от M_O.

Чтобы найти границы адекватности, мы должны установить, в каких пределах и как по отношению друг к другу могут меняться параметры модели, чтобы она оставалась адекватной. Ответы на эти вопросы вы получите, выполнив очередную лабораторную работу. А сейчас подготовим для этой лабораторной работы заполнение электронной таблицы. Можно ее заполнить, например, как таблицу 14:

Таблица 14

A	B	C	D
k		*L*	
Год	Неограниченный рост	Ограниченный рост	Отклонение, в %
0	1	1	(B3–C3)/C3*100
A3+1	(1+B1)*B3	(1+B1*(D1–C3)/(D1–C3))*C3	(B4–C4)/C4*100
A4+1	(1+B1)*B4	(1+B1*(D1–C4)/(D1–C3))*C4	(B5–C5)/C5*100
A5+1			

В клетку B1 заносится коэффициент прироста k, а в клетку D1 — предельное значение L массы живых организмов. В клетки B3 и C3 мы занесли значение начальной массы живых организмов, взяв ее равной 1. Заполнение остальных клеток, скорее всего, ясно из написанных в них формул.

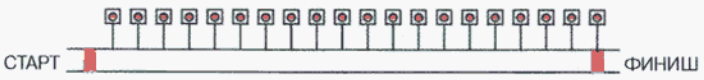
И снова немного истории. Первым модель неограниченного роста рассмотрел Т. Мальтус (1798 г.). Опираясь на эту модель, он пытался обосновать неизбежность войн и других кризисных явлений социально-политической жизни человеческого общества. Он ввел понятие демографического давления как показателя превышения численности населения, проживающего на данной территории, над возможностью данной территории обеспечить это население продовольствием. Из его рассмотрений делался вывод о необходимости постоянного расширения жизненного пространства, и этот вывод использовался для построения и оправдания расовых тео-

рий, националистических теорий борьбы за жизненное пространство и т. п.

Проанализируем эту ситуацию с позиций модельного подхода. В модели неограниченного роста в качестве существенных принимались только биологические факторы. Но если живые организмы образуют сообщества (стада, стаи и т. п.), то вступают в силу иные факторы, характерные для данного сообщества. Их можно было бы назвать социальными, хотя термин «социальные» обычно применяют к сообществам людей. Что касается человека, то для него одним из важнейших социальных факторов является развитие науки и производства. Скажем, за счет применения удобрений с той же сельскохозяйственной территории стали снимать в несколько раз больший урожай, а значит, та же территория способна прокормить большее население, чем прежде. Поэтому с точки зрения информатики несостоятельность применения модели неограниченного роста к человеческому обществу состоит уже в том, что учтены не все существенные факторы, а сама модель применяется не в той области, где она является адекватной, — в области социально-политической, а не чисто биологической.

ВОПРОСЫ И ЗАДАНИЯ

1. Сформулируйте принцип адекватности модели. Как вы могли бы его обосновать?
2. Выполнение каких условий надо проверить, прежде чем использовать какую-либо модель для решения жизненной задачи?
3. Что значит найти границы адекватности данной модели?
4. Для изучения графика бега на дистанции 100 м были установлены на одинаковом расстоянии 20 фотодатчиков, фиксирующих время, прошедшее от старта до пересечения спортсменом соответствующего светового луча.



Данные этих датчиков, округленные до десятых долей секунды:

Номер датчика	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Время, секунды	1,2	1,7	2,1	2,5	2,9	3,3	3,7	4,1	4,5	4,9	5,3	5,7	6,1	6,5	6,9	7,3	7,8	8,4	9,0	9,8

Попытайтесь по этим данным определить, на каких участках дистанции бег спортсмена адекватно описывается моделью равномерного движения, а на каких — равноускоренного.

Поиск границ адекватности модели

Исследовать модель неограниченного роста на адекватность мы будем при уже известных нам параметрах: значение k возьмем 1,8 (как для тайги), а L будем считать равным 11 000.

❶ Заполните электронную таблицу, как объяснено в § 28, и внесите исходные данные. Последовательно копируя блок A4:D4 в последующие строки, найдите, в какой год отклонение превзойдет границу 10%. (При этом не забудьте указать электронной таблице, адреса каких ячеек не должны меняться при копировании.)

Теперь исследуйте значение k, равное 1,2.

❷ Скопируйте блок B1:D... (здесь вы должны указать номер последней заполненной строки в таблице), начиная с клетки F1. Запишите в F1 коэффициент 1,2. В каком году отклонение превзойдет границу 10%? Пришлось ли вам еще копировать строки?

Уже эти два компьютерных эксперимента показывают, что с уменьшением k граница n отодвигается.

❸ Для подкрепления этого вывода проведите компьютерный эксперимент еще и при $k = 1$.

Теперь естественно исследовать, как граница адекватности зависит от величины L. Общие соображения подсказывают, что с ростом L граница n должна увеличиваться. Но каков характер этой зависимости? Давайте, к примеру, удвоим значение L.

❹ Введите удвоенное значение L при $k = 1$.

Посмотрите: граница отодвинулась на один год. А если еще раз удвоить?

❺ Проведите этот эксперимент.

Опять граница отодвинулась на один год. А если исходное значение L уменьшить вдвое?

❻ Проведите эксперимент при $L = 5500$ (но по-прежнему $k = 1$).

Эффект тот же самый — теперь граница n уменьшилась на 1. Напрашивается гипотеза, что L образует геометрическую прогрессию относительно границы адекватности n. Если вспомнить формулу общего члена геометрической прогрессии, то получим, что $L = b \cdot 2^{n-1}$, где b — некоторый коэффициент. Найти этот коэффициент нетрудно — достаточно разделить L на соответствующую степень 2.

Задумаемся, однако, над полученной формулой. Ведь в общей формуле для L, наверно, еще и k должно как-то участвовать. Учитывая, что при $k = 1$ выполнено соотношение $2=1+k$, можно предположить, что $L = b\ (1+k)^{n-1}$.

Осталось найти b и убедиться, что этот коэффициент практически не зависит от k.

❼ Найдите b при $L = 5000$ и различных k, равных: 1; 1,2; 1,5; 2. Убедитесь, что во всех случаях $b \approx 8$.

Итак, наши компьютерные эксперименты показали, что моделью неограниченного роста можно пользоваться с уровнем погрешности в 10% при выполнении условия $L \geqslant 8\ (1 + k)^{n-1}$. Тот, кто помнит, как решаются показательные неравенства, легко получит явное выражение для n, показывающее, как долго можно пользоваться моделью неограниченного роста при заданных L (предельный уровень массы живых организмов) и k (коэффициент ежегодного прироста). Эта формула такова:

$$n \leqslant 1 + \frac{\lg(0{,}125L)}{\lg(1+k)}.$$

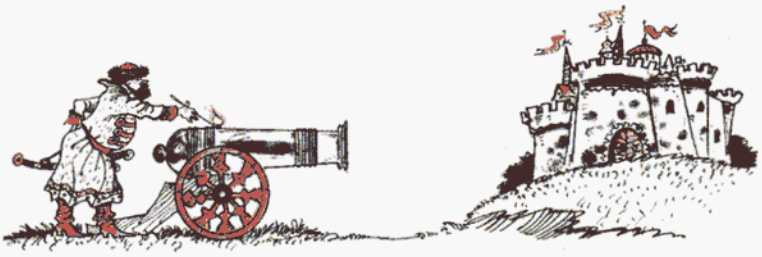
Цель, поставленная в начале лабораторной работы, достигнута. Но мы надеемся, что вы в ходе выполнения этой работы научились большему. Вы наблюдали, как с помощью средств информационных технологий исследуется характер зависимости между различными переменными, выдвигаются, а затем проверяются в компьютерном эксперименте гипотезы о формуле для этой зависимости. Позже, в других лабораторных работах, мы еще вернемся к вопросу, как с помощью компьютерных технологий разыскивать зависимость между величинами. Здесь и сейчас вы получили первый опыт этой интересной исследовательской деятельности.

§ 29. Из пушки по...

От задачи, которую мы будем решать в этом параграфе, веет средневековьем. Конечно, в те времена обходились без компьютеров, поэтому сначала приходилось проводить многочисленные испытания на полигонах. Ведь речь идет о... Впрочем, вот эта задача.

Идет осада неприятельской крепости. На некотором расстоянии от нее установили новую пушку. Под каким углом к горизонту надо стрелять из этой пушки, чтобы попасть в заданный участок крепостной стены?

Над моделью этой задачи физики изрядно поработали. Оно и понятно: ведь многие научные задачи, как и эта, возникали прежде всего в военном деле. И решение этих задач почти всегда считалось приоритетным.



Какие же факторы принять за существенные в этой задаче? Поскольку речь идет о средневековье, то скорость снаряда и дальность полета невелики. Значит, можно считать несущественным, что Земля круглая (помните обсуждение в § 27?), и пренебречь сопротивлением воздуха. Остается единственный фактор — сила земного притяжения. В этом случае, как вы знаете из физики, горизонтальное (x) и вертикальное (y) смещение снаряда за время t описывается формулами

$$x = (v\cos\alpha)t,\ y = (v\sin\alpha)t - \frac{gt^2}{2},$$

где g — ускорение свободного падения, v — начальная скорость снаряда, α — угол наклона пушки к горизонту. Эти формулы задают математическую модель полета снаряда. Нас же интересует, на какой высоте окажется снаряд, пролетев расстояние S.

Впрочем, это нетрудно найти. Выразим время полета снаряда на расстояние S из первой формулы:

$$t = \frac{S}{v\cos\alpha},$$

и подставим во вторую:

$$H = S\operatorname{tg}\alpha - \frac{gS^2}{2v^2\cos^2\alpha}.$$

Следуя нашей задаче, нам требуется найти такое значение угла наклона α, чтобы снаряд, пролетев заданное расстояние S, попал на нужную высоту H.

Математик тут бы сказал, что надо просто решить уравнение. Мы тоже будем его решать, только приближенно и очень похоже на то, как делают настоящие артиллеристы. Они же поступают следующим образом: производят несколько выстрелов, беря цель «в вилку», т. е. одно попадание выше цели, а другое ниже. Затем делят пополам угол между этими выстрелами, и при стрельбе под таким углом снаряд ложится к цели намного ближе. Но если все же не попали, то новую «вилку» снова делят пополам и т. д.

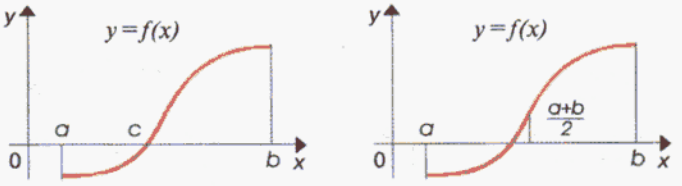


Рис. 31. Поиск корня методом деления пополам

Мы заранее можем указать «вилку» для угла: 0 и $\frac{\pi}{4}$ (мы надеемся, что вы помните, какой угол имеет радианную меру $\frac{\pi}{4}$ и чему приближенно равно π). А дальше будем делить пополам эту «вилку» и смотреть, куда попадает снаряд, пока не добьемся нужного результата.

Как же долго нам придется вести «пристрелку», чтобы получить угол α с нужной точностью? Чтобы ответить на этот вопрос, отвлечемся от нашей задачи и сформулируем на чисто математическом языке, что и как мы находили.

Нам даны некоторая функция $f(x)$ и отрезок $[a; b]$, причем на концах этого отрезка эта функция принимает значения противоположных знаков. Если функция непрерывна, т. е. ее график — непрерывная линия, то ясно, что график функции пересекает ось абсцисс в некоторой точке c отрезка $[a; b]$, как показано на рисунке 31, *а*. Иными словами, $f(c) = 0$, т. е. c — корень уравнения $f(x) = 0$.

Как же предлагается находить этот корень? А вот как. Делим отрезок $[a; b]$ пополам, т. е. берем середину отрезка $\frac{a+b}{2}$. В этой точке вычисляем значение функции $f(x)$ (рис. 31, *б*). Если это значение 0, то корень найден; если нет, то оно имеет тот же знак, что и значение на одном из концов отрезка $[a; b]$. Тогда этот конец заменяем точкой $\frac{a+b}{2}$. Новый отрезок тоже содержит корень уравнения $f(x) = 0$, поскольку на его концах функция $f(x)$ снова имеет разные знаки. Однако этот отрезок в 2 раза короче предыдущего. И самое главное — *с ним можно поступить точно так же*. Со следующим отрезком еще раз проделать то же самое и т. д. Поскольку длина отрезка каждый раз уменьшается вдвое, мы можем получить отрезок сколь угодно малой длины, внутри которого содержится корень уравнения $f(x) = 0$. Например, если исходный отрезок был [3; 4], т. е. имел длину 1, то через десять шагов мы получим отрезок длиной $\frac{1}{2^{10}} = \frac{1}{1024} < 0{,}001$. Это означает, что концы отрезка

дают нам приближенное значение корня с точностью, равной длине отрезка: левый конец отрезка — приближенное значение корня с недостатком, правый конец — приближенное значение корня с избытком.

Фактически мы сейчас сформулировали **метод приближенного решения уравнения** $f(x)=0$. Его можно было бы назвать методом артиллерийской пристрелки. Но математики называют его **методом половинного деления**.

ВОПРОСЫ И ЗАДАНИЯ

1. Для чего предназначен метод половинного деления? Сформулируйте его.
2. Почему метод половинного деления является приближенным, а не точным методом решения уравнения?
3. Важно ли для практических целей располагать точными методами решения уравнений? Свой ответ постарайтесь обосновать.
4. Попытайтесь объяснить границы для угла наклона пушки, указанные в объяснительном тексте параграфа.
5. а) Для уравнения $x^3 - 3x + 3 = 0$ определите два числа, образующие «вилку» для корня этого уравнения. Сколько раз придется выполнить деление пополам для найденного вами отрезка, чтобы получить корень с точностью 0,01? А с точностью 0,001?

   б) Выполните задание а) для уравнения $2^x = 3x$.

   в) Выполните задание а) для уравнения $\cos x = x$.
6. а) Постройте математическую модель следующей задачи:
   Пушка стреляет в направлении движения грозового облака в тот момент, когда оно проплывает над пушкой. Под каким углом к горизонту должна стрелять пушка, чтобы попасть в облако? Известны скорость антиградового снаряда, длина и скорость облака и высота, на которой оно движется.

   б) Какую «вилку» для угла наклона пушки к горизонту вы могли бы указать, чтобы решить получившееся в а) уравнение методом половинного деления?

Метод половинного деления

Снова компьютерным средством, с помощью которого мы будем решать задачу, служит электронная таблица. Подготовим ее заполнение (табл. 15).

Таблица 15

A	B	C	D
Расстояние S	3000	Точность	0.001
Высота H	1	Длина отрезка	C4–B4
Начальная скорость	200		B3^2
Угол	0	0	(B4+C4)/2
Отклонение от цели	B2–B1*(D5–9.8*B1*(1+D5^2)/(2*D3))		tg(D4)

В клетках B4 и C4 записаны значения угла (в радианах), составляющие «вилку»; в клетке D4 — значение угла, для которого будет вычисляться отклонение от цели. Кроме того, чтобы по нескольку раз не вычислялось одно и то же число (а на это уходит время), в клетке D5 записан тангенс очередного значения угла наклона пушки к горизонту, а в клетке D3 — квадрат начальной скорости (поскольку в электронной таблице все формулы записываются в «линейку», то и для показателя степени используется не верхний индекс, а специальный знак — ^). С той же целью — ускорения вычислений — мы в формуле отклонения заменили $\frac{1}{\cos^2\alpha}$ на $1+\operatorname{tg}^2\alpha$. Заполнение остальных клеток понятно из таблицы. Значение g взято 9,8 м/с^2, расстояние $S=3$ км, а высота $H=1$ м. Точность вычисления корня равна 0,001.

Сначала проверим, правильно ли мы выбрали отрезок для корня. В таблице в клетках B4 и C4 записаны нули, поэтому отклонение подсчитывается для $\alpha = 0$. Как видите, на левом конце отрезка отклонение положительно.

Запишите теперь в клетках B4 и C4 число 0,75 (это — приближенное значение для $\frac{\pi}{4}$). Теперь отклонение оказалось отрицательным.

❶ Приступим к нахождению нужного угла α. Запишите в клетке B4 число 0, и электронная таблица тут же вычислит значение отклонения в точке $\frac{0{,}75}{2}$.

Это значение оказалось положительным. Следовательно, значением $\frac{0{,}75}{2}$ надо заменить левый конец отрезка, записанный в клетке B4.

❷ Меняем 0 на значение клетки D4. Отклонение стало отрицательным.

Следовательно, надо поменять значение клетки C4 на значение клетки D4. Действуйте!

❸ Продолжайте поиск корня, пока не получится заданная точность (напоминаем, что индикатором точности служит клетка D2, в которой вычисляется длина текущего отрезка).

Так при каком же значении угла вы не промахнетесь?

§30. Как измерить количество информации

Название параграфа, возможно, вызовет у вас недоумение: ведь в учебнике уже рассказывалось о единицах измерения объема сообщения и вы, конечно, помните, что такое бит, байт, килобайт и т. п.

Все это верно. Но тогда вы должны помнить еще и то, что при введении указанных единиц оговаривалось, что в этих единицах объем информации измеряется чисто с технической точки зрения, никак не отражающей смысловое содержание информационного сообщения. К примеру, если на диск дважды записано одно и то же сообщение, то места на нем занято в 2 раза больше, чем в том случае, когда на диск сообщение записано однократно. Значит, и в байтах мы получим число, в 2 раза большее, чем для одного сообщения. Однако из повторного сообщения мы никакой новой (по смыслу) информации не имеем, так что количество информации остается тем же, что и после получения первого сообщения.

Чтобы разобраться в связи между техническим подходом к измерению **количества информации** и смысловым (по-научному *семантическим*), поиграем в игру «Угадайка». Правила игры таковы.

Один игрок задумывает целое число из заранее определенного диапазона, например от 1 до 16. Другой игрок, задавая такие вопросы, на которые первый может отвечать только «да» или «нет», должен выяснить, какое число было задумано. К примеру, можно спросить: «Верно ли, что задуманное число равно 7?» или «Больше ли задуманное число 3?»

Поиграйте с соседом по парте в эту игру и попытайтесь понять, какое наименьшее число вопросов гарантирует вам отгадывание числа.

Поиграли? Теперь порассуждаем. Конечно, число можно выяснить, задавая последовательно вопросы: «Верно ли, что задумано число 1?», «Верно ли, что задумано число 2?», «Верно ли, что задумано число 3?», «Верно ли, что задумано число 4?», ... , «Верно ли, что задумано число 16?» На какой-то из вопросов ответ будет «да». Но не исключено, что это будет шестнадцатый по счету вопрос.

Как же гарантированно получить ответ за меньшее число вопросов? Вспомним метод половинного деления. Разделим наши числа

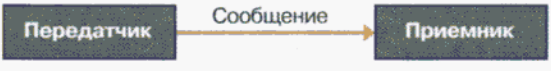
от 1 до 16 на две равные группы, например от 1 до 8 и от 9 до 16. Вопросом: «Верно ли, что задуманное число больше 8?» — мы заставим первого игрока сказать нам, в каком из этих двух интервалов находится задуманное им число. Вторым вопросом мы найденный нами интервал разделим еще раз пополам; если, скажем, ответ игрока на первый вопрос был утвердительным, то спросим: «Верно ли, что задуманное число больше 13?» Теперь уже задуманное число находится в интервале, состоящем всего лишь из четырех чисел. Следующий, третий по счету вопрос разделит этот интервал еще раз пополам. Из оставшихся двух чисел узнать задуманное можно, задав всего лишь один вопрос. Итак, четырех вопросов здесь оказалось достаточно, чтобы определить задуманное число.

Нетрудно понять, что если исходный диапазон чисел был бы от 1 до 32, то хватило бы пяти вопросов для гарантированного угадывания задуманного числа. Если исходный диапазон от 1 до 64, то хватило бы шести вопросов и т. д.

Ситуацию в игре «Угадайка» можно описать так. В начале игры неопределенность в выборе задуманного числа составляла 16 условных единиц — любое число из 16 могло оказаться задуманным. После первого вопроса неопределенность уменьшилась вдвое — задуманным теперь могло оказаться любое из 8 чисел. После второго вопроса неопределенность уменьшилась еще вдвое и т. д. Иными словами, ответ на каждый вопрос давал нам информацию, вдвое уменьшающую неопределенность. Поэтому в информатике договорились *принять количество информации, уменьшающей вдвое неопределенность исходной ситуации, равным одному биту*. Теперь ясно, что количество информации, уменьшающей неопределенность в 4 раза, равно двум битам, уменьшающей в 8 раз — трем битам и т. д. Вообще если полученная информация уменьшает неопределенность в n раз, то говорят, что количество этой информации равно $\log_2 n$. Иными словами, если необходимо выделить один объект из множества, содержащего n равноправных объектов, мы должны располагать $\log_2 n$ информации об этом объекте.

Почему же и при техническом подходе к измерению объема информации, и при семантическом подходе используются одни и те же единицы — биты, байты, килобайты?.. Попробуем в этом разобраться.

В главе 1, как вы, вероятно, помните, мы объясняли, что всю информацию, циркулирующую в компьютере и хранящуюся в его памяти или на магнитных носителях, можно представлять себе закодированной в двухсимвольном алфавите, например в виде последовательностей из 0 и 1. То же самое можно сказать и об информации, передаваемой по каналам связи. На рисунке схематично показано, как происходит такая передача информации:



Под **приемником информации** мы понимаем здесь человека или автоматическое устройство, которое просто фиксирует сигналы, поступающие к нему по каналу связи. Он не обязан при этом вникать в смысл передаваемой информации, да может и просто не знать код, которым закодировано исходное сообщение. Тем самым перед получением очередного сигнала приемник информации находится в состоянии неопределенности — придет сигнал, соответствующий символу 0, или сигнал, соответствующий символу 1. Когда сигнал приходит, неопределенность уменьшается вдвое (и, конечно, полностью исчезает), т. е. приемник получает, как мы договорились выше, 1 бит информации. Теперь уже ясно, что, получив последовательность из n символов двоичного алфавита, приемник получил n бит информации. Вот мы и получили тот же вывод, который ранее предлагался вам в качестве определения количества информации при чисто техническом подходе.

Первым понятие количества информации ввел американский ученый Клод Шеннон. Занимаясь проблемой передачи информации по каналам связи, он заложил основы теории информации, которая теперь является одним из краеугольных камней информатики.

К. Шеннон

ВОПРОСЫ И ЗАДАНИЯ

1. Как связано уменьшение неопределенности с количеством полученной информации?
2. Какое количество информации мы должны получить, чтобы угадать одно из 64 первых натуральных чисел; одно из 128 первых натуральных чисел; одно из 100 первых натуральных чисел? Сколько вопросов в игре «Угадайка» придется задать, чтобы гарантированно получить ответ в каждом из этих трех случаев?
3. а) Задумывается нечетное число от 1 до 63. Сколько вопросов в игре «Угадайка» придется задать, чтобы гарантированно угадать задуманное число?

   б) Известно, что задуманное число меньше 1000 и является квадратом некоторого целого числа. Сколько вопросов в игре «Угадайка» придется задать, чтобы гарантированно угадать задуманное число? Сколько бит информации вы получили дополнительно, располагая сведениями, что задуманное число является не просто целым неотрицательным числом, а квадратом некоторого целого числа?
4. Объясните, почему в игре «Угадайка» с первыми 16 натуральными числами нельзя гарантировать угадывание числа за три вопроса.

4 Основы алгоритмического управления

Вы многому уже научились в трех предшествующих главах. И компьютер наверняка стал вашим надежным помощником во многих делах. Но остается тайна: как он справляется со всеми поручениями, которые вы ему давали? Как создаются, на каких принципах основаны те чудесные программы, которые управляют его работой?

Скажем сразу: эти принципы относятся не только к компьютерам, но и к любой микропроцессорной технике, к любым роботам, какими бы сложными и человекоподобными они ни казались. И в основе управления всеми такими устройствами лежит важное понятие алгоритма. Об алгоритмах, их свойствах и применениях к решению задач и пойдет речь в этой главе.

§31. Бездумные Исполнители

Вспомним забавный диалог из мультфильма «Вовка в тридевятом царстве»:

— Эй, Двое-из-Ларца, Одинаковы-с-лица!!!

— Мы здесь!

— Замесить и нарубить!.. Эй, да вы что?!!

— А что?

— Наоборот!!!

Бедный Вовка! Ничего хорошего не получилось: почему-то рубилось тесто, а месились дрова. Все дело в том, что школьников в его время не учили информатике и он не распознал в лихих удальцах из ларца особый класс созданий, называемых **Бездумными Исполнителями** (БИ).

Но не торопитесь сетовать вместе с Вовкой на их несообразительность. Так ли уж они плохи? Все Вовкины приказания были выполнены быстро и четко. Разве нет? Стоит ли упрекать Двоих-из-Ларца за то, что они, получив не совсем понятную команду, шустренько кинулись ее выполнять? Вспомните, разве вам не приходилось «Принести вон то!», или «Купить чего-нибудь поесть», или делать нечто вроде «Пойди туда, не знаю куда, принеси то, не знаю что». Всегда ли вам удавалось выполнить эти команды ко всеобщему удовлетворению?

Вряд ли вы будете гордиться, если вас назовут Бездумным Исполнителем. Но всегда ли плохо им быть? Будет ли рад хозяин овчарки, когда по команде «Фас!» его четвероногий друг задумается, стоит ли связываться с бандитом? А самолет в ответ на движение штурвала пилота продолжал бы лететь дальше, потому что разворот делать не хочется. А оператор ядерного реактора, забросив инструкцию, начал бы управлять сложнейшим агрегатом по наитию... Согласитесь, даже человеку быть БИ иногда просто необходимо!

Вспомнив предыдущие уроки информатики, вы наверняка поймете, что все те средства компьютерных технологий, которые вами изучались, являются не чем иным, как Бездумными Исполнителями, готовыми по одному движению вашего пальца кинуться обрабатывать электронные документы. И если электронная таблица почему-то не хочет пересчитывать данные, то объясняется это либо неисправностью оборудования, либо (что бывает гораздо чаще) вашими собственными ошибками. БИ, разумеется, бездумный, но это вовсе не означает, что бездумным должен быть человек, отдающий ему команды.

Подумаем, какими должны быть команды, чтобы БИ нас не разочаровывал. Правильными? Но что это означает? Правильная команда — понятная команда?

А если вам скомандуют починить суперкомпьютер? Или сделать двойное сальто назад? Или графическому редактору скомандуют напечатать картинку, а принтер не подключен?

Значит, для БИ правильной командой будет та, которую он не только понял, но и способен выполнить. Во избежание недоразумений следовало бы поинтересоваться у БИ, как он понимает каждую **команду**. Собственно говоря, мы это и делаем, когда, например, внимательно читаем инструкцию к новому магнитофону, разбираясь, как он реагирует на нажатие различных кнопок. Этим же вы занимались и тогда, когда учились работать с текстовым редактором или базой данных.

Совокупность всех команд, которые может выполнить конкретный БИ, называется **системой команд** этого исполнителя. А совокупность всех **действий**, которые он может выполнить в ответ на эти команды, называется **системой допустимых действий** исполнителя.

Когда мы выбираем исполнителя, то надеемся с его помощью решить нужную нам задачу. Но тогда сразу встает вопрос, может ли исполнитель с помощью своих допустимых действий получить требуемый результат. Совокупность тех результатов, которые можно получить с помощью данного исполнителя, называется его **достижимыми целями**. Однако для конкретного исполнителя описание всех его достижимых целей — задача, как правило, довольно трудная. Некоторые простые случаи этой задачи мы предлагаем ниже в заданиях 11 и 12.

ВОПРОСЫ И ЗАДАНИЯ

1. Что такое система команд исполнителя? Что такое система допустимых действий данного исполнителя?
2. Приведите примеры Бездумных Исполнителей и опишите их системы команд. С какими БИ вы уже познакомились на уроках информатики?
3. Что называется достижимыми целями исполнителя?
4. Насколько можно говорить о том, плох БИ или хорош? Например, что означает фраза «Я недоволен своим текстовым редактором»? Означает ли это, что БИ не справляется со своей системой команд?
5. Часто ли вы бываете БИ? Кто и какую систему команд использует, чтобы вами управлять?
6. Какими способами можно подавать команды Бездумному Исполнителю?
7. Что, на ваш взгляд, ответит человек и что ответит БИ, если к ним обратиться со следующими вопросами:
   а) Не можете ли вы сказать, который час?
   б) Не знаете ли вы, который сейчас час?
   в) Не будете ли вы любезны сказать, который сейчас час?
   г) Который сейчас час?
8. Какую систему команд вы бы могли предложить для домашнего робота БИ-001?
9. Разведывательный дозор в составе двух человек подошел к реке. Мост был разрушен, а река слишком глубока и широка, чтобы переправиться через нее вброд или вплавь. К счастью, около берега в маленькой лодке проплывали два мальчика. Как переправиться на этой лодке через реку, если она может выдержать либо одного взрослого, либо двоих детей?

   Для решения этой задачи представьте, что и взрослые, и дети — это БИ, способные выполнить следующие действия:
   - сесть в лодку;
   - переправиться в лодке на противоположный берег;
   - выйти из лодки.

   Придумайте систему команд, соответствующую данным допустимым действиям, и составьте инструкцию для этого коллектива Бездумных

Исполнителей, позволяющую разведдозору переправиться на другой берег.

10. На полустанке одноколейной железной дороги остановился поезд в составе тепловоза и трех вагонов, доставивший бригаду рабочих для строительства второго пути. Пока же на этом полустанке имеется только небольшой тупик, в котором при необходимости может поместиться тепловоз с вагоном или два вагона. Вскоре следом за поездом со строительной бригадой к тому же полустанку подошел пассажирский поезд. Как пропустить пассажирский поезд?
Аналогично предыдущей задаче определите допустимые действия БИ — машинистов тепловозов, придумайте систему команд для них и составьте инструкцию по разводке поездов.
11. Даны три листа бумаги. Исполнитель берет лист, разрезает его на четыре части и кладет их обратно. Из нового набора листов он снова выбирает любой лист, опять разрезает на четыре части и кладет их обратно. С полученным набором листов он снова поступает так же и т.д., пока не поступит команда остановиться (такая команда всегда поступает, но заранее не известно, сколько разрезаний сделает к этому моменту исполнитель). Какое количество листов может получиться в результате его работы?
12*. На доске написаны числа 1, 2, 3, ..., n. Исполнитель может стереть два числа и записать вместо них абсолютную величину их разности. Через $n-1$ шаг на доске останется одно число. Цель — получить число 0. Для каких n эта цель достижима?
13. (Вопрос с уклоном в философию.) Может ли существовать Всемогущий Исполнитель, для которого любая цель является достижимой?

§ 32. Что такое алгоритм

Прекрасное время — каникулы! Сладко потягиваясь, вы просыпаетесь, когда все уже ушли на работу, и наталкиваетесь на записку: «Митя! Завтрак на столе. Сходи за хлебом и молоком. Вымой посуду. Надень новую футболку. Целую, мама». Значит, опять придется почти весь день быть БИ! Разве нет? Все команды понятны. Во всяком случае, совершенно очевидно, что завтрак должен быть съеден, сколько покупать хлеба и молока тоже ясно — не в первый раз...

Конечно, можно согласиться, что все команды в маминой записке входят в систему команд, но здесь, с точки зрения БИ, мы имеем нечто новое: нам предписано сразу несколько команд. В каком же порядке их должен выполнять Бездумный Исполнитель? «Ну, конечно, по порядку! — скажете вы. — Это же естественно!» Настоящий БИ именно так команды и выполняет. Но тогда посуда будет стоять грязной целый день, а в новой футболке вам придется щеголять только к вечеру. Вряд ли это имела в виду мама. Тогда нужно признать, что для БИ последовательность команд в маминой записке неверна.

Последовательность команд, полученная БИ для выполнения, называется **программой**. Таким образом, в этой программе сразу две ошибки. Об ошибках мы поговорим чуть позже, а пока прочтите внимательно две программы:

How to make a good tea	**Как приготовить хороший чай**
Bring fresh water to boil.	Вскипятите свежую воду.
Warm the teapot by rinsing out with hot water.	Ополосните заварочный чайник крутым кипятком.
Put one teaspoonful of tea per cup into the teapot and pour immediately the boiling water into the tea.	Положите чай в заварочный чайник из расчета одна чайная ложка на чашку и сразу же залейте кипятком.
Stir the tea after 3 minutes.	Через 3 минуты размешайте.
Add sugar.	Добавьте сахар.

Первая программа, очевидно, написана для Джонни-из-за-океана, о котором мы уже как-то упоминали. Вторая — для его русского друга Вани. Ясно, что каждый из них будет выполнять одни и те же действия, и поэтому, наблюдая за их работой, мы не сможем отличить одного от другого. Ведь фактически они последовательно выполняют один и тот же набор действий. Только записаны эти наборы для каждого на своем языке. Более того, можно заснять на видеокассету все манипуляции Джонни или Вани, а затем показать любому жителю Земли, и ему будет ясно, как заваривать чай. И совершенно неважно, какой язык понимает этот житель! (Вот, кстати, с точки зрения человека, еще одно преимущество видеоинформации перед информацией символьной.) Поэтому когда говорят о самой последовательности действий для достижения какой-либо цели, то используют термин **алгоритм**, а не программа. Чтобы алгоритм стал программой для конкретного БИ, нужно все действия, входящие в алгоритм, записать командами из системы команд этого исполнителя.

Приведем примеры двух алгоритмов:

Как открыть дверь	**Как проехать к другу**
Достать ключ.	Выйти из дома.
Вставить ключ в замочную скважину.	Повернуть направо.
Повернуть ключ 2 раза против часовой стрелки.	Пройти 2 квартала до автобусной остановки.
Вынуть ключ.	Сесть в автобус № 25, идущий к центру города.
	Проехать 3 остановки.
	Выйти из автобуса.

Давайте в первом алгоритме поменяем местами второе и третье действия, а во втором — третье и четвертое. Вот что получится:

Как «открыть» дверь

Достать ключ.
Повернуть ключ 2 раза против часовой стрелки.
Вставить ключ в замочную скважину.
Вынуть ключ.

Как «проехать» к другу

Выйти из дома.
Повернуть направо.
Сесть в автобус № 25, идущий к центру города.
Пройти 2 квартала до автобусной остановки.
Проехать 3 остановки.
Выйти из автобуса.

Перед нами снова два алгоритма. Но, следуя первому, едва ли удастся отпереть дверь — цель, для которой составлялся алгоритм, не будет достигнута. Что касается второго алгоритма, то его не удастся даже исполнить — невозможно, находясь в автобусе, идти 2 квартала пешком. Эти примеры показывают, что от порядка действий зависит не только результат, но и выполнимость алгоритма в целом (хотя каждое отдельно взятое действие является допустимым действием исполнителя).

Итак,

Алгоритм — это организованная последовательность допустимых для некоторого исполнителя действий, приводящая к определенному результату, а программа — это запись алгоритма на языке конкретного БИ.

Слово «алгоритм», несмотря на свою современную связь с компьютерной техникой, имеет очень древние корни. Оно произошло от латинского «algoritmi», представляющего собой запись арабского имени выдающегося математика IX в. Аль-Хорезми. В средневековой Европе алгоритмом называлось искусство счета в десятичной системе счисления, с которой европейцы познакомились в XII в. именно благодаря латинскому переводу математического трактата Аль-Хорезми. Имя этого математика было, конечно, намного длиннее: Аль-Хорезми в переводе на русский язык означает всего лишь «из Хорезма», города, который и сейчас является жемчужиной Узбекистана.

Различие между понятиями алгоритма и программы весьма тонкое, однако для поиска ошибок в программе даже есть специальный термин **отладка**. Отладка в качестве составной части обязательно включает в себя поиск ошибок, допущенных именно в *записи* команд. Такие ошибки принято называть **синтаксическими**. На них БИ дает реакцию «Не понимаю». Ошибки другого вида могут вызвать реакцию «Не могу исполнить»; такие ошибки называются **семантическими** (от слова «семантика» — смысл, значение). С семантической ошибкой мы встретились в ошибочном алгоритме поездки к другу. Тем не менее поиск ошибок в алгоритме никак по-особенному не называется.

Строго говоря, фразу, записанную в рамочке, нельзя считать определением алгоритма. В ней не объясняется, например, что означают слова «организованная», «действия», «результат». Скажем сразу — абсолютно строгого определения алгоритма мы не дадим и в будущем. Понятие алгоритма в информатике является фундаментальным. Таким же, какими являются понятия точки, прямой и плоскости в геометрии, пространства и времени в физике, вещества в химии и т. д. Поэтому мы не будем стремиться дать всеобъемлющее определение алгоритма, а будем уточнять смысл этого понятия в последующих параграфах.

Нам предстоит еще неоднократно записывать алгоритмы, поэтому давайте договоримся, выделяя порядок действий в алгоритме, записывать действия в столбик (как и в приведенных примерах).

Как правило, алгоритмы пишутся для человека. Поэтому мы будем записывать алгоритмы на обычном русском языке. При этом каждый из вас может применять какие-либо сокращения слов и вообще заменять одни слова другими. Важно только, чтобы за этими словами стояли действия, допустимые для данного вам исполнителя. При этом самого исполнителя удобно представлять себе как устройство управления, соединенное с набором инструментов. Устройство управления воспринимает и анализирует алгоритм, а затем организует его выполнение, командуя соответствующими инструментами. Инструменты же производят действия, выполняя команды управляющего устройства. От исполнителя требуется лишь четкое выполнение каждого действия, входящего в алгоритм. Мы не должны объяснять ему, для каких целей предназначается алгоритм.

ВОПРОСЫ И ЗАДАНИЯ

1. Любые ли действия могут присутствовать в алгоритме, предназначенном для данного исполнителя?
2. Почему нельзя дать строгого определения алгоритма?
3. Какие общие свойства алгоритмов вы можете назвать?
4. Приведите примеры неопределяемых понятий в следующих школьных предметах: а) литература; б) русский язык; в) история; г) биология. Перечислите фундаментальные понятия информатики, которые вы уже изучили к данному времени.
5. В чем различие между алгоритмом и программой?
6. Назовите книги — сборники алгоритмов. Кто является для них БИ и какие наиболее типичные команды входят в его систему команд?
7. С какими алгоритмами вы познакомились, занимаясь: а) русским языком; б) иностранным языком; в) математикой; г) литературой; д) химией; е) физкультурой; ж) информатикой?
8. *Старинная задача.* Некий БИ должен перевезти в лодке через реку волка, козу и капусту. Его допустимые действия таковы, что за один раз он может перевезти только что-нибудь одно: волка, козу или капусту. Ничем, кроме погрузочно-разгрузочных работ и перевозок, этот БИ не занимается.

Составьте для данного БИ алгоритм переправы, позволяющий избежать жертв.

9. а) Имеется два кувшина емкостью 3 л и 8 л. Исполнитель ДЖИНН может набирать воду из реки в каждый кувшин, выливать из него воду и определять, налита ли вода в кувшине доверху. Составьте алгоритм, выполнив который ДЖИНН наберет из реки 7 л воды.
   б) Пусть ДЖИННу подменили трехлитровый кувшин двухлитровым. Существует ли теперь алгоритм для этого исполнителя, позволяющий набрать из реки 7 л воды?
10. Исполнитель умеет заменять в слове ровно одну букву, причем из осмысленного слова должно получаться снова осмысленное слово (иначе исполнитель ломается). Составьте алгоритмы преобразования:
    а) слова САД в слово КОТ;
    б)* слова МУХА в слово СЛОН.
11. БИ умеет выполнять следующие действия:
    - взять *X*;
    - поджарить *X*;
    - смолоть *X* в мясорубке;
    - закатать *X* в *Y*;
    - сварить *X*;
    - нарезать *X*;
    - положить *X* на *Y*,

    где вместо букв *X* и *Y* можно подставить слова «мясо», «тесто», «сыр», «то, что получилось», «хлеб».
    Используя эти действия, составьте:
    а) алгоритм приготовления пельменей;
    б) алгоритм приготовления чего-либо еще съедобного;
    в) алгоритм приготовления чего-нибудь несъедобного;
    г) какой-нибудь неисполнимый алгоритм.
12. Составьте программу для выполнения какой-либо работы по дому роботом БИ-001 (о нем шла речь в задаче 7 из предыдущего параграфа).
13. Сыграйте в БИ. Пусть ваш товарищ станет БИ. Обсудите с ним его систему команд и попробуйте составить для него небольшую, команд в десять, программу. Посмотрите, сможет ли он ее выполнить. Если нет, то почему? Уточните систему команд или отладьте программу.
14. Представьте себе, что ваш младший братишка впервые пойдет в магазин за хлебом. Напишите для него алгоритм, объясняющий, как добраться до магазина и как обращаться с деньгами. (Совет: предварительно обсудите список его допустимых действий; например, является ли для него допустимым переход через дорогу.)
15. Что такое отладка программы?
16. а) Какую ошибку называют синтаксической? А какую семантической? Какие из этих видов ошибок относятся к программе, а какие — к алгоритмам?
    б) Могут ли быть в алгоритме или программе другие ошибки, кроме синтаксических и семантических? Если да, то как бы вы их назвали?

17. Вспомните свою работу с программой «Черный ящик». Там на выходах тоже встречались слова «Не понимаю» и «Не могу исполнить». Как вы думаете, это случайное совпадение с реакциями БИ на синтаксические и семантические ошибки?
18. Даны число x и набор действий: разделить полученное число на 3; умножить x на 2; сообщить результат; прибавить к полученному числу 4; вычесть из полученного числа 7.
    а) Составьте из этих действий какой-нибудь алгоритм. Любой ли алгоритм, составленный из этих действий, можно исполнить?
    б) Укажите две различные функции от x, значения которых вычисляются с помощью алгоритмов, использующих указанные действия, и два различных алгоритма, вычисляющих одну и ту же функцию.
19. Злоумышленник выдал следующий алгоритм за алгоритм получения кипятка:
    Налить в чайник воду.
    Открыть кран газовой горелки.
    Поставить чайник на плиту.
    Ждать, пока вода не закипит.
    Поднести спичку к горелке.
    Зажечь спичку.
    Выключить газ.
    Исправьте алгоритм, чтобы предотвратить несчастный случай.
20. Имеются цинк, 96%-ная серная кислота, вода, а также колба и пробирка. Исправьте ошибки в алгоритме получения водорода:
    Поставить колбу на стол.
    Налить в колбу кислоту.
    Налить в колбу воду.
    Собрать выделяющийся газ в пробирку.
    Бросить в колбу цинк.
21. Придумайте несколько коротких (не более 15 команд) алгоритмов с ошибками и дайте проанализировать их своему товарищу. Предварительно сообщите, с какой целью вы составляли каждый алгоритм.
22. а) Какие действия вы бы добавили, чтобы человеком был выполним следующий алгоритм переправы через Волгу в районе г. Саратова:
    Подойти к реке.
    Войти в реку.
    Идти по дну, пока не выйдешь на другой берег.
    б) Может ли, на ваш взгляд, существовать исполнитель, который способен исполнить алгоритм из пункта а) таким, какой он есть?
    в) Зависит ли исполнимость человеком алгоритма, приведенного в пункте а), от того, где совершается переход через Волгу?
23. Пусть дан отрезок AB. Определить, для решения какой задачи предназначен следующий алгоритм:
    Поставить ножку циркуля в точку A.
    Установить раствор циркуля равным длине отрезка AB.
    Провести окружность.
    Поставить ножку циркуля в точку B.

Провести окружность.
Провести прямую через точки пересечения окружностей.

24. а) Петя и Коля в роли БИ выполняют с помощью микрокалькулятора следующий алгоритм:
Сложить 83,2438 и 57,6847.
Полученный результат умножить на 10.
У Пети на табло калькулятора помещается 6 цифр, а у Коли — 8. Будут ли у них одинаковы результаты выполнения этого алгоритма?
б) Приведите другие примеры алгоритмов, результаты исполнения которых зависят от их исполнителей.

§33. Знакомьтесь: исполнитель Паркетчик

Как мы уже отмечали, осваивая средства информационных технологий, вы фактически изучали все новых и новых Бездумных Исполнителей и осваивали их системы команд. Но алгоритмов вы для них никаких не составляли, потому что понимают они только язык нажатий на кнопки мыши или клавиатуры.

А сегодня мы познакомим вас с исполнителем, которым можно управлять, составляя для него алгоритмы. Такие БИ, которые могут исполнять алгоритмы, называются **исполнителями алгоритмов**. Представляем вам одного из исполнителей алгоритмов. Его зовут **Паркетчик**, и, надеемся, он станет вашим хорошим другом. На первый взгляд он вам покажется совсем простым. Но с его помощью вы узнаете законы мира Бездумных Исполнителей, решите немало увлекательных головоломок, а иногда с ним можно будет и поиграть.

Говоря о Бездумных Исполнителях и их системах команд, мы неявно предполагали наличие еще одного важного компонента — среды обитания Бездумного Исполнителя, т. е. того окружения, над которым исполнитель может производить какие-либо действия. Такой средой для Перевозчика являются река с двумя берегами и пассажиры, для Джинна — два кувшина и вода в реке, для текстового редактора — символьный электронный документ и т. д.

Средой (или, если хотите, игровым полем) Паркетчика является лист клетчатой бумаги (не настоящий, разумеется, а изображенный на экране компьютера) и квадратные плитки двух цветов — красного и зеленого. Впрочем, желающие могут изготовить натурную модель Паркетчика, расчертив настоящий лист бумаги и вырезав достаточное количество красных и зеленых плиток из картона. Каждая плитка покрывает в точности одну клетку на бумаге. Паркетчик занимается тем, что, исполняя те или иные алгоритмы, выкладывает на бумаге орнаменты (паркеты) из этих плиток. Для простоты будем считать, что у Паркетчика неограниченный запас красных и зеленых плиток (хотя ясно, что ему не надо плиток каждого цвета больше, чем клеток на поле, где он работает).

Каждая клетка на поле имеет свой **адрес**, чтобы Паркетчик знал, где ему предстоит выполнять работу. На экране своего компьютера вы видите игровое поле сверху, поэтому мы будем говорить о горизонтальных и вертикальных рядах клеток.

Так, клетка, стоящая в 5-м столбце и 7-й строке, имеет адрес (5, 7). Вы, наверно, уже сталкивались с подобным, если играли в «морской бой». В отличие от игры в «морской бой» мы нумеруем строки снизу вверх и в адресе клетки вначале указываем столбец, а затем строку. (Кое-кто сразу понял, почему так сделано, а остальным рекомендуем вспомнить, что такое *система координат*. И почему номер столбца называют *абсциссой*, а номер строки — *ординатой*.)

Чтобы выложить тот или иной орнамент, Паркетчик может переходить с любой клетки на соседнюю. Иными словами, для Паркетчика допустимы следующие четыре действия:

- шаг вверх;
- шаг вниз;
- шаг вправо;
- шаг влево.

Но может Паркетчик и сразу прыгнуть в указанную ему клетку. Делает это он по команде «Перейти на (... , ...)», только вместо многоточий в круглых скобках, конечно, должны быть указаны координаты клетки.

Кроме того, как было уже сказано, у Паркетчика есть еще два допустимых действия:

- положить красную плитку;
- положить зеленую плитку.

Осталось сообщить, что в начале игры Паркетчик всегда находится в левом нижнем углу, т. е. в клетке (1, 1).

Теперь вы можете составлять разнообразные алгоритмы, инструктирующие Паркетчика, как выложить задуманные вами орнаменты.

Разумеется, когда вы будете записывать алгоритмы для Паркетчика в своих тетрадях, совсем не обязательно записывать действия буквально. Можно писать и так:

→ — вместо Шаг вправо;
К — вместо Положить красную плитку;
↑ — вместо Шаг вверх;
← — вместо Шаг влево;
З — вместо Положить зеленую плитку;
↓ — вместо Шаг вниз.

Можно записывать команды как-нибудь еще. Главное, чтобы было ясно, какое действие подразумевается. (Кстати, какой орнамент выложит Паркетчик, если написанную только что последовательность действий рассматривать как алгоритм?)

Другое дело, когда вы будете писать программу. Программа — это не алгоритм: в ней, как вы помните, должны стоять команды только из системы команд БИ. Так, команду Шаг вверх Паркетчик понимает как родную, а вот вашего сокращения не поймет.

Нельзя также, записывая команды для Паркетчика, делать грамматические ошибки. Увидев операторы «Шаг *верх*», «Пола*ж*ить (з)» и т. п., Паркетчик сообщит, что таких команд он не знает.

Не поймет Паркетчик и другого. Скажем, вы предложите ему из начальной позиции сделать шаг вниз. Или, к примеру, положить плитку на поле, где она уже лежит. И хотя команды написаны без ошибок, Паркетчик остановится, сообщив:

Ошибка во время выполнения программы.

Это надо понимать так: «Рад бы выполнить, да не могу!» А уж ваша задача определить, что же мешает Паркетчику выполнить команду.

Значит, как и в русском языке, ошибки в программе для Паркетчика могут быть двух типов: синтаксические и смысловые (или семантические).

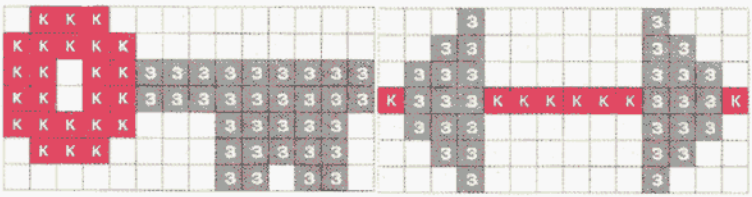
О том, как записываются команды Паркетчика, мы подробнее расскажем при подготовке к лабораторной работе.

ВОПРОСЫ И ЗАДАНИЯ

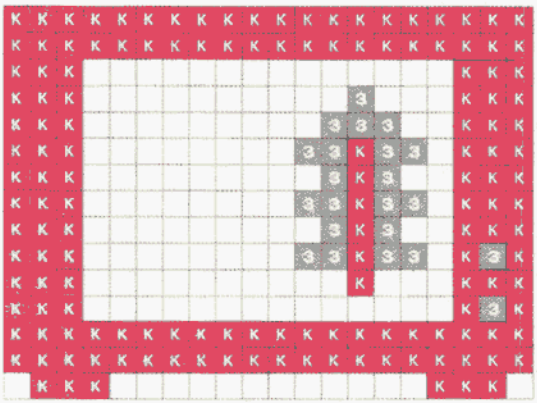
1. Какие допустимые действия Паркетчика вы знаете?
2. Что является средой для:
   системы управления базами данных;
   станка с числовым программным управлением;
   автопилота;
   графического редактора;
   программируемого видеомагнитофона?
3. Нарисуйте в тетради поле Паркетчика, имеющее 5 горизонтальных и 5 вертикальных рядов. Поработайте за Паркетчика и определите, какой рисунок он выложит, выполнив следующий алгоритм:
   Положить красную плитку;
   Шаг вверх;
   Положить красную плитку;
   Шаг вверх;
   Положить красную плитку;
   Шаг вверх;
   Положить красную плитку;
   Шаг вправо;
   Шаг вправо;
   Шаг вправо;
   Положить зеленую плитку;
   Шаг влево;
   Шаг вниз;
   Положить зеленую плитку;
   Шаг вниз;

Положить зеленую плитку;
Шаг вниз;
Положить зеленую плитку;
Шаг вправо;
Шаг вниз;
Положить зеленую плитку;
Шаг вправо;
Шаг вверх;
Положить зеленую плитку;
Шаг вверх;
Положить зеленую плитку;
Шаг влево;
Шаг влево;
Шаг влево;
Положить красную плитку.

4. Для каждого из рисунков 32, *а* — *в* составьте алгоритм выкладывания Паркетчиком этих орнаментов.



а) Ключ б) Штанга



в) Телевизор

Рис. 32. Орнаменты для Паркетчика

5. Нарисуйте какой-нибудь паркет с узором, приятным вашему глазу, и составьте алгоритм для Паркетчика, выполняя который он сможет так выложить паркет.
6. Красиво нарисуйте узор для паркета, составляющий ваши инициалы, и напишите алгоритм для Паркетчика по его выкладыванию.

Первая встреча с Паркетчиком

Пришла пора посмотреть, как вас слушается Паркетчик не на бумаге, а в жизни — на экране дисплея. Паркетчик относится к тем БИ, которым для работы требуется программа. И поэтому поначалу вы видите экран обычного, не очень сложного текстового редактора, в котором вам предстоит набирать эту программу.

Но набирать программу слово за словом по одной буковке слишком долго. Пока наберешь, урок может закончиться. Чтобы ускорить набор программы, все слова, которые понимает Паркетчик, занесены в меню. И выбор слова в меню обеспечивает его автоматическое появление в тексте программы. Само меню появляется на экране, как только вы нажмете функциональную клавишу <F4>.

Нажмите эту клавишу. Перед вами появилось шесть пунктов меню:

- Действия;
- Условия;
- Ветвления;
- Циклы;
- Подпрограммы;
- Описания.

С помощью клавиш управления курсором выберите нужный пункт и нажмите <Ввод>. Поначалу вам потребуются в основном **Действия**.

В появившемся списке команд выберите ту, которая необходима, и снова нажмите клавишу <Ввод>. Эта команда тут же появится в *программной строке* в том месте, где стоит курсор.

Если вы выбрали команду Положить( ), то нужно будет указать цвет плитки, нажав клавишу с буквой <з> или <к>, в зависимости от того, зеленую или красную плитку вы хотите положить на поле. А во всем остальном работа по набору команд ничем не отличается от работы в среде текстового редактора, с которым вы уже знакомы.

Правда, этот текстовый редактор «обучен» проверять правильность команд для Паркетчика. И если какая-то команда набрана неверно, справа от нее высвечивается сообщение об ошибке. Вы тут же можете внести исправления.

Для Паркетчика действует еще одно правило: каждая программа должна начинаться со слова Программа, а весь последующий текст заключается в фигурные скобки. Слово Программа, как и сами фигурные скобки, набирать вручную необязательно — вы можете найти его в подпункте меню «Подпрограммы».

Наберите, например, самую простенькую программу:

```
Программа
{
    Шаг вверх;
    Положить(к);
    Шаг вправо;
    Положить(з);
    Шаг вправо;
    Шаг вправо;
    Положить(к)
}
```

и запустите ее.

Чтобы поручить Паркетчику исполнить программу, выберите пункт меню **Выполнение** или просто нажмите функциональную клавишу <F9>. На экране появится поле Паркетчика, и он примется бойко бегать по полю, выполняя вашу программу.

Конечно, можно записывать и несколько операторов в одной строке, но это верный путь запутаться в собственной программе, когда она станет чуточку сложнее. Поэтому давайте сразу будем привыкать к правильному стилю оформления программ для БИ.

Если программа вам больше не нужна, ее легко стереть, последовательно выбрав в меню пункты **Файл** и **Новая программа**. Компьютер вас предусмотрительно спросит, не надо ли предварительно записать старую программу на диск — для дальнейшего использования.

Вы, конечно, помните, что Паркетчик живет и трудится на прямоугольном клетчатом листе бумаги. Значит, если вам вдруг захотелось изменить стандартные размеры игрового поля — 42×30 клеточек, выберите пункт меню **Установка** и задайте другие размеры.

Теперь вы можете создать на поле любой интересный для вас орнамент, положив плитки нужного цвета. Для этого выберите пункт меню **Паркет**, подпункт **Изменить** и воспользуйтесь клавишами с буквами <к> и <з>, нажимая их одновременно с клавишами управления курсором. Чтобы выйти из режима редактирования поля Паркетчика, достаточно нажать клавишу <Esc>.

А теперь для тренировки составьте программу, исполнив которую Паркетчик на чистом поле выложит по диагонали (начиная с левого нижнего угла) 4 зеленые плитки. *Не забудьте только поставить открывающую фигурную скобку { после слова* Программа, *а в конце текста программы — закрывающую скобку }*. Эти скобки принято называть **операторными**. Дело в том, что иногда слова «команда» и **«оператор»** считают синонимами. Раз в эти скобки заключают

команды, т. е. операторы программы, то и скобки стали называть не фигурными, а операторными, и мы тоже будем к этому привыкать.

Для целей отладки служит команда **Стоп**. Ее можно поставить в любое место программы, чтобы убедиться, что программа выполняется нормально вплоть до этого места.

Наконец, не забудьте после работы сохранить свою программу. Для этого служат пункты меню **Сохранить** и **Сохранить как**. Не сохраняйте программу со стандартным именем NONAME — вы можете оказаться неодиноки, и тогда вместо вашей программы будет записана чужая.

Мы не последний раз работаем с Паркетчиком. Постепенно вы запомните его режимы работы. Впрочем, это совсем не обязательно: нажав знакомую вам клавишу <F1>, вы тут же на экране дисплея получите подсказку.

А теперь напишите программу по алгоритму, который вы составили, выполняя задание 6 к этому параграфу. Введите эту программу. Надеемся, Паркетчик нарисовал именно то, что вам хотелось.

§34. Циклическое исполнение алгоритма. Оператор «Делать пока...»

Наверно, все вы наслышаны о замечательных приключениях Тома Сойера, описанных Марком Твеном. Конкретнее нас сегодня будет занимать история с покраской забора. Итак, слово Марку Твену:

«ВЕЛИКОЛЕПНЫЙ МАЛЯР

Том вышел на улицу с ведром известки и длинной кистью. Он окинул взглядом забор, и радость в одно мгновение улетела у него из души, и там воцарилась тоска... Со вздохом обмакнул он кисть в известку, провел ею по крайней доске, потом проделал то же самое снова и остановился: как ничтожна белая полоска по сравнению с огромным пространством некрашеного забора!..»

Остановимся. Попробуем представить, как могла бы выглядеть та же история в наши дни. Конечно, современная тетя Полли вряд ли огораживала бы свой дом тридцатью ярдами деревянного забора высотой девять футов! Скорее всего, ухоженная лужайка возле дома если и огорожена, то изящными решетками на каменных столбиках. Нынешние мальчишки, когда дело касается покраски забора, отнюдь не стали более прилежными. Зато они увлекаются компьютерами. Поэтому во избежание недоразумения тетушка Полли могла бы давать наставления племянничку в виде алгоритмов, так любимых Томом:

- Подойти к первой решетке;
- Покрасить первый прутик;

- Покрасить второй прутик;
- Покрасить третий прутик; ...

Здесь старушка поняла бы, что гораздо легче самой покрасить ограду или найти кого-нибудь попокладистей. Шутка ли: только в первой решетке то ли две, то ли три дюжины прутиков, а этих самых решеток!..

Пора прийти ей на помощь и ввести новый оператор:

```
Делать пока (условие)
  { оператор;
    оператор;
    оператор;
      ...
  } (* конец цикла *)
```

Этот замечательный оператор называется **циклом**. Запись Делать пока (условие) называется **заголовком цикла**. Идущая после заголовка цикла совокупность операторов, заключенных в операторные скобки, называется **телом цикла**.

В круглых скобках со звездочками стоит **комментарий**. Он совершенно не нужен БИ, но полезен тому человеку, который пишет программы, если он не хочет стать бездумным. С умом написанные комментарии помогают понять программу даже тому, кто ее не составлял. Кроме того, в программах с комментариями допускается гораздо меньше ошибок, чем в программах без них.

Программа покраски забора с использованием цикла будет выглядеть так:

```
Делать пока (есть неокрашенная решетка)
  { Подойти к неокрашенной
    решетке;
      Покрасить ее;
  } (* конец цикла *)
Сообщить о том, что работа выполнена;
Идти купаться;
```

Профессионалы, занимающиеся теорией программирования, вместо слова «условие» договорились употреблять термин **«высказывание»**. От высказывания требуется единственное — БИ должен уметь определять, истинно оно или ложно.

Думается, вам ясно, как работает этот оператор. Сначала БИ проверяет, истинно ли высказывание в скобках, и если да — выполняются операторы, стоящие в операторных скобках после заголовка цикла.

Дойдя до закрывающей операторной скобки, БИ снова проверяет, верно ли высказывание, и если да — снова выполняет те же самые операторы. Если же в момент проверки высказывание ложно (нет больше неокрашенных решеток), то БИ пропускает весь цикл

и начинает выполнение операторов, следующих за закрывающей операторной скобкой.

Поскольку слова Делать пока стандартные, мы *будем их подчеркивать*. Кроме того, при написании программы в тетради *будем выделять циклы стрелками* так, как показано в программе покраски забора.

Теперь вспомним, что решетка состоит из прутиков, и, следовательно, оператор «Покрасить решетку» требует дальнейшего уточнения:

```
(* Покрасить решетку — это: *)
      Делать пока (есть неокрашенный прутик)
      {Покрасить один неокрашенный прутик
      }
```

Это напоминает сбор детской игрушки «матрешки» — чтобы собрать программу покраски ограды в одно целое, нужно вставить этот фрагмент в основную программу:

```
Делать пока (есть неокрашенная решетка)
{Подойти к неокрашенной решетке;
    Делать пока (в решетке есть неокрашенный прутик)
    {Покрасить один неокрашенный прутик;
    }
}
Сообщить о том, что работа выполнена;
Идти купаться;
```

Интересно, что здесь мы получили **двойной цикл**. **Внутренний цикл** предписывает Тому красить по одному прутику в решетке, пока не выкрасит их все. **Внешний цикл** предписывает перейти к новой, не покрашенной решетке. Если, как здесь, один цикл располагается внутри другого, то говорят, что внутренний цикл является **вложенным** по отношению к внешнему.

Обратите внимание на порядок записи операторов: те из них, которые составляют тело цикла, *пишутся со сдвигом вправо*. Это делается для того, чтобы с первого взгляда увидеть в программе циклы.

ВОПРОСЫ И ЗАДАНИЯ

1. Когда целесообразно применять оператор цикла?
2. Что такое заголовок цикла? Что такое тело цикла?
3. Какие циклы называются вложенными?
4. Нужны ли комментарии БИ?
5. Если вы хотите оправдать репутацию не только знатока оператора цикла, но и литературы, объясните, почему историю с Томом нельзя использовать для иллюстрации циклической работы. (Совет: внимательно прочитайте исходный текст М. Твена, а не только приведенную в объяснительном тексте цитату.)

6. Придумайте программу для Тома с использованием цикла Делать пока и тетушкиного забора, позволяющую ему превратиться из жалкого бедняка в богача, буквально утопающего в роскоши.
7. Используя циклическую форму организации действий, запишите следующий алгоритм выполнения домашнего задания по переводу текста с иностранного языка:
   Прочитать первое предложение;
   Перевести его;
   Записать перевод в тетрадь;
   Найти следующее предложение;
   Перевести его;
   Записать перевод в тетрадь;
   Найти следующее предложение;
   Перевести его;
   Записать перевод в тетрадь;
   . . .
8. «Приключения Тома Сойера» начинаются с того, что тетя Полли зовет Тома:

   — Том!
   Нет ответа.
   — Том!
   Нет ответа.
   — Том!
   Нет ответа...
   Петя Торопыжкин составил следующий алгоритм вызова Тома:

   Делать пока (нет ответа)
   { Крикнуть: «Том!»
   } (* конец цикла *)

   Найдите ошибку в этом алгоритме.
9. Перечитайте задачу о мальчиках, переправляющих разведдозор из двоих солдат через реку (задание 9 из § 31). Представьте, что к реке подошел взвод солдат. Составьте алгоритм переправы для взвода.
10. Перечитайте задачу о разъезде рабочего и пассажирского поездов (задание 10 из § 31). Представьте, что рабочий поезд имеет не три, а больше вагонов. Составьте алгоритм для пропуска пассажирского поезда в этом случае.
11. Прочитайте внимательно стихотворение Григория Остера:

    Возьми густой вишневый сок
    И белый мамин плащ.
    Лей аккуратно сок на плащ —
    Появится пятно.
    Теперь, чтоб не было пятна
    На мамином плаще,
    Плащ надо сунуть целиком
    В густой вишневый сок.
    Возьми вишневый мамин плащ
    И кружку с молоком.

Лей аккуратно молоко —
Появится пятно.
Теперь, чтоб не было пятна
На мамином плаще,
Плащ надо сунуть целиком
В кастрюлю молока.
Возьми густой вишневый сок
И белый мамин плащ...

Неоспоримы поэтические достоинства этого произведения. Однако ни один программист такого бы не написал. Придумайте условия окончания цикла и запишите то же самое с помощью операторов цикла. И гораздо короче.

12. Работая в текстовом редакторе, школьник напечатал на экране компьютера текст: «Игра закончилась, а гол так и не был забит». Подошедший злоумышленник выполнил следующий алгоритм:
Выбрать режим «Замена».
Перейти к запросу «Что заменить».
Напечатать букву «г».
Перейти к запросу «Чем заменить».
Напечатать букву «к».
Установить режим «Во всем тексте».
Нажать на клавишу «ВВОД».

а) Какая фраза теперь написана на экране?
б)* С помощью какого алгоритма можно восстановить первоначальную фразу?

Оператор цикла в работе Паркетчика

Жизнь у Паркетчика сложная. И заказчики все как один привередливые. Вместо того чтобы ограничиться выкладыванием двух-трех плиток, подавай им большие красочные узоры. А попробуй-ка хотя бы выложить красной плиткой первый ряд! Вон какая длинная программа получается:

Программа
{
Положить (к);
Шаг вправо;
Положить (к);
Шаг вправо;
Положить (к);
Шаг вправо;
Положить (к);

```
    Шаг вправо;
    Положить (к);
    ...
(а там, где многоточие, еще более шести десятков операторов).
}
```

Но такую программу можно составить только в том случае, если заранее известны размеры поля. И для каждого поля приходится иметь уникальную программу! А хочется иметь универсальную программу, годную для полей любых размеров.

Не будем скрывать: Паркетчик прекрасно понимает конструкцию Делать пока, о которой мы так много говорили. Надо только знать, какие условия он умеет проверять. Вот эти условия:

- Справа стена,
- Слева стена,
- Снизу стена,
- Сверху стена.

Умеет Паркетчик проверять и **отрицание** этих высказываний:

- Не справа стена,
- Не слева стена,
- Не снизу стена,
- Не сверху стена.

Теперь уже не составляет труда написать короткую программу, с помощью которой Паркетчик выложит нижний ряд красными плитками при каком угодно размере паркета:

Программа

```
{Делать пока (не справа стена)
 {    Положить (к);
      Шаг вправо;
 }
```

Наберите эту программу и заставьте Паркетчика ее выполнить.

Все ли получилось так, как вы задумывали? Если нет, исправьте программу и запустите ее еще раз. Другими словами, займитесь отладкой программы.

Можно заказать Паркетчику одноцветную рамку вдоль границ поля, размеры которого Паркетчику неизвестны. Для этого достаточно слегка (раза в четыре) увеличить размер предыдущей программы. К счастью, увеличение довольно-таки однотипное и вряд ли вызовет у вас затруднения.

Теперь давайте попробуем выложить красными плитками все поле. Очевидно, можно было бы записать следующий алгоритм:

```
Выложить первый ряд;
Вернуться на ту плитку, с которой начинал;
```

```
Делать пока (не сверху стена)
{ Шаг вверх;
  Выложить ряд, на котором стоишь;
  Вернуться на ту плитку, с которой начинал;
  }
```

Но дело в том, что это алгоритм для человека, а не для Паркетчика: ведь у него нет допустимого действия Выложить ряд. Нам соответствующая строка в алгоритме понятна, а Паркетчику нет.

Воспользуемся тем приемом сборки программы, который уже помог нам организовать двойной цикл.

Ясно, что мы можем написать программу, с помощью которой Паркетчик выложит плитки в первом ряду. Поскольку мы это уже сделали, то с ходу приводим ее текст:

```
Делать пока (не справа стена)
{ Положить (к);
  Шаг вправо;
  }
Положить (к);
```

Внимательно присмотревшись, понимаем, что это программа замащивания не только первого ряда, но и любого, лишь бы к ее началу Паркетчик стоял в самой левой клетке этого ряда. Значит, ее вполне можно вставлять вместо сразу двух операторов алгоритма:

```
Выложить первый ряд;
Выложить ряд, на котором стоишь;
```

Получим следующий алгоритм, который уже гораздо ближе к программе на языке Паркетчика:

```
(* Выложить первый ряд; *)
Делать пока (не справа стена)
{ Положить (к);
  Шаг вправо;
  }
Положить (к);
(* конец закраски первого ряда *)
Вернуться на ту плитку, с которой начинал;
Делать пока (не сверху стена)
{ Шаг вверх;
  (* Закраска ряда, на котором стоишь; *)
  Делать пока (не справа стена)
  { Положить (к);
    Шаг вправо;
    }
  Положить (к);
  (* конец закраски ряда, на котором стоишь *)
  Вернуться на ту плитку, с которой начинал;
  }
```

В описании алгоритма осталось еще одно действие, не являющееся допустимым для Паркетчика:

Вернуться на ту плитку, с которой начинал;

Надеемся, для вас будет не очень сложно перевести его на язык Паркетчика и то, что получится, вставить в соответствующее место алгоритма.

Заставьте Паркетчика выполнить написанную вами программу.

Выполняя программу, Паркетчик бегает так, как если бы водил кистью в горизонтальном направлении. Переделайте вашу программу так, чтобы его «кисть» двигалась по вертикали, т. е. сначала выкладывался первый столбик, затем второй и т. д.

§ 35. Условные операторы

Вы уже видели, что в заголовке цикла записана проверка некоторого **условия**. И каждый раз, как закончится исполнение всех действий, входящих в тело цикла, это условие снова проверяется.

Но в жизни может быть так, что вовсе не надо много раз проверять условие, чтобы решить, следует или не следует выполнять какую-либо последовательность действий. Если на улице холодно, мы надеваем пальто; если идет дождь, раскрываем зонт; если перекопана улица, ищем объезд... И, выполнив действие, мы вовсе не спешим снова и снова проверять условие, которое вынудило нас предпринять действие.

Для составления алгоритмов, реализующих подобные ситуации, предусмотрен специальный вид команд, называемых **условными командами** или, по-другому, **условными операторами**.

Смысл условного оператора вполне ясен из его записи:

Если (условие), то {оператор; оператор; ...}

Заметим, что операторы, стоящие в фигурных скобках, будут исполняться только в том случае, если условие истинно. А если условие ложно, то исполнитель перейдет к исполнению тех операторов, которые стоят после данного условного оператора.

Слова Если... то... стандартные, поэтому мы их тоже будем подчеркивать.

Вот, например, какую замечательную программу написал для вас писатель Григорий Остер в книжке «Вредные советы»:

Если друг на день рожденья
Пригласил тебя к себе,
То оставь подарок дома —
Пригодится самому.
Сесть старайся рядом с тортом,
В разговоры не вступай,
(* Ты во время разговора *)
(* Вдвое меньше съешь конфет. *)

Выбирай куски помельче,
(* Чтоб быстрее проглотить. *)
Не хватай салат руками,
(* Ложкой больше зачерпнешь. *)
Если вдруг дадут орехи,
Ссыпь их бережно в карман,
Но не прячь туда варенье —
(* Трудно будет вынимать. *)

В этой программе, кроме операторов, присутствуют и очень небесполезные комментарии, которые мы, как обычно, заключили в скобки со звездочками.

Комментарии-то присутствуют, но отсутствует другое, нечто более важное. И если вы еще не догадались в чем дело, подумайте, как понимать в программе Г. Остера такой ряд операторов:

Если вдруг дадут орехи,
(то) Ссыпь их бережно в карман,
Но не прячь туда варенье.

Варенье не прятать в карман только в том случае, если дают орехи? А без орехов, значит, можно?

Каждому ясно: не хватает операторных скобок, которые показывают, насколько далеко распространяет свое влияние условие, стоящее после слова Если. Итак,

Необходимо заключать все операторы, к которым относится условие, в операторные скобки.

И не забудьте, конечно, о том, что условие надо помещать в круглые скобки, как это сделано в операторе Делать пока...

Условный оператор нередко называют **ветвлением**. Такое название возникло из-за существования еще одной формы условного оператора:

Если (условие), то {оператор; оператор; ...}
иначе {оператор; оператор; ...}

Такой оператор предписывает не только то, что надо исполнять, если условие истинно, но и действия исполнителя, если условие ложно. Этот вид называется **полной формой** условного оператора. Ну а вид Если... то... называют оператором ветвления в **неполной форме**.

Итак, в том случае, когда исполнение какого-либо действия зависит от проверки истинности некоторого условия, применяется условный оператор Если... то... или Если... то... иначе... . Ясно при этом, что

Проверка истинности условия должна быть допустимым действием исполнителя.

ВОПРОСЫ И ЗАДАНИЯ

1. Что такое условный оператор?
2. Как записывается условный оператор в полной форме? А в неполной?
3. Для чего служат операторные скобки в операторах Делать пока и Если... то...? Как вы думаете, почему они одинаковые для обоих операторов?
4. В стихотворении Г. Остера, процитированном в объяснительном тексте, не отмечен один комментарий.
   а) Найдите его.
   б) В том же стихотворении расставьте операторные скобки.
5. Напишите для себя программу проведения свободного вечера, используя условные операторы и операторные скобки.
6. Вспомните ДЖИННА, который переливает воду из одного сосуда в другой (см. задачу 9 из § 32). Предположим, что он умеет проверять, полный кувшин или нет (наполнен доверху) и есть ли в кувшине вода (по характерному бульканью). После нескольких переливаний в трехлитровом сосуде осталось A л воды (A — целое число). Известно, что сосуд неполный. Составьте алгоритм, после выполнения которого в трехлитровом сосуде будет $3 - A$ л воды.
7. Петя Торопыжкин решил позвонить приятелю по телефону и составил для себя такой алгоритм:

   Снять трубку;
   Если (есть гудок), то набрать номер;
   Если (длинные гудки), то подождать полминуты;
   Если (приятель ответил),
   то сказать: «Привет!»;
   Рассказать новости;
   Послушать новости приятеля;
   Сказать: «До свидания»;
   Повесить трубку;

   Петя, как обычно, торопился и забыл расставить операторные скобки. Объясните, к каким неприятностям может привести исполнение этого алгоритма. Расставьте в алгоритме операторные скобки.
8. Перепишите в той форме, о которой мы с вами условились, следующие алгоритмы, предлагаемые Г. Остером; не забудьте выделить комментарии:

   а) Если тебя родная мать
   Ведет к зубным врачам,
   Не жди пощады от нее,
   Напрасных слез не лей.
   Молчи, как пленный партизан,
   И стисни зубы так,
   Чтоб не сумела их разжать
   Толпа зубных врачей.

   б) Если всей семьей купаться
   Вы отправились к реке,
   Не мешайте папе с мамой
   Загорать на берегу.

Не устраивайте крика,
Дайте взрослым отдохнуть.
Ни к кому не приставая,
Постарайтесь утонуть.

в) Если друг твой самый лучший
Поскользнулся и упал,
Покажи на друга пальцем
И хватайся за живот.
Пусть он видит, лежа в луже:
Ты ничуть не огорчен.
Настоящий друг не любит
Огорчать своих друзей.

9. Приведите примеры каких-либо не проверяемых человеком условий.
10. (С математическим уклоном.) Однажды к Пете и Коле пришел их приятель шестиклассник Саша и сказал, что им задали трудную задачу: «Для двух чисел $\frac{9999}{10\,000}$ и $\frac{10\,000}{10\,000}$ установить, равны ли эти числа, и если равны, то их надо перемножить, а если нет, то из большего вычесть меньшее». Скандируя: «Для программиста нет трудных задач!» — Петя и Коля составили такой алгоритм решения сформулированной задачи:

Разделить 9999 на 10 000 и результат обозначить буквой b;
Разделить 10 000 на 10 001 и результат обозначить буквой c;
Если (b=c), то {умножить b на c};
Если (b>c), то {из b вычесть c};
Если (b<c), то {из c вычесть b};

а) Проверьте, правильно ли составлен алгоритм.
б) Напомним (см. задачу 24 из § 32), что Петя и Коля в роли БИ исполняют алгоритмы с помощью микрокалькулятора, при этом у Пети на табло калькулятора помещается 6 цифр, а у Коли — 8.
Будут ли у них одинаковыми результаты выполнения этого алгоритма?
в) Устроит ли Сашу хотя бы один из полученных результатов? Попытайтесь в уме решить Сашину задачу.

11. Придумайте несколько условий, которые, на ваш взгляд, должен уметь проверять робот БИ-001 (о нем шла речь в задачах 8 из § 31 и 12 из § 32). Составьте две-три программы для этого исполнителя с использованием условных операторов.

Условные операторы в работе Паркетчика

Как мы уже отмечали, жизнь у Паркетчика сложная, а заказчики привередливы: вчера был заказан орнамент в зеленых тонах, а сегодня нужен только в красных. Пришел, видите ли, домой и по-

нял, что зеленый орнамент ему не подходит. Хорошо, если бы это был какой-то единичный случай. Можно было бы просто переписать программу выкладывания орнамента, заменив все операторы Положить (з) на операторы Положить (к). Однако желающих поменять цвет паркета так много, что хочется иметь программу, выполнив которую Паркетчик заменил бы все зеленые плитки на красные. Но для этого Паркетчик должен уметь распознавать цвета плиток.

К счастью, в добавление к высказываниям:

- Справа стена,
- Слева стена,
- Снизу стена,
- Сверху стена —

Паркетчик умеет проверять, истинны ли такие два высказывания: «На клетке, где я нахожусь, лежит красная плитка» и «На клетке, где я нахожусь, лежит зеленая плитка».

Договоримся для краткости писать эти условия в алгоритмах так: красная плитка и зеленая плитка (а вы можете писать еще короче, например, как в системе команд Паркетчика: Если (к)...) — и попробуем разобраться с такой задачей:

Где-то на границах поля располагаются плитки разных цветов.

Требуется все красные плитки заменить на зеленые.

Алгоритм решения этой задачи довольно-таки прост:

Обежать границы поля;

В процессе обегания

Если (к) то {Снять плитку; Положить (з)};

Конечно, Паркетчик такого алгоритма не поймет. Если вы еще не забыли, БИ требуется программа, т. е. алгоритм, записанный с помощью системы команд БИ. Поэтому вспомним предыдущую работу и программу, с помощью которой мы раскрашивали границы поля в красный цвет. В ней Паркетчику как раз пришлось обежать границы и в процессе обегания выкладывать красные плитки:

Программа

```
{
   Делать пока (не справа стена)
      { Положить (к);
        Шаг вправо;
      }
   Делать пока (не сверху стена)
      { Положить (к);
        Шаг вверх;
      }
```

```
    Делать пока (не слева стена)
    ┌──→{ Положить (к);
    │     Шаг влево;
    └─────}
    Делать пока (не снизу стена)
    ┌──→{ Положить (к);
    │     Шаг вниз;
    └─────}
}
```

Для вас, умудренных опытом написания и отладки такой программы с четырьмя циклами, вряд ли будет сложным чуть-чуть ее подправить и, используя оператор Если ...то..., решить поставленную задачу замены плиток. Не забудьте нажимать клавишу <F4> и использовать текстовые шаблоны.

Еще немного измените программу замены плиток так, чтобы с ее помощью Паркетчик решал похожую, но чуть более сложную задачу:

Где-то на границах поля располагаются плитки разных цветов.

Требуется все красные плитки заменить на зеленые и *наоборот*.

Пользуясь тем, что вы уже научили Паркетчика обегать не только границы, но и вообще все поле, составьте для него программу смены цвета всех лежащих на поле плиток.

§36. Вспомогательный алгоритм

Надеемся, вы уже хорошо усвоили, что не имеет смысла говорить о БИ, не рассматривая жестко связанную с ним его систему команд. Система команд конкретного исполнителя называется **языком исполнителя**. Возникает вопрос: можно ли обогатить этот язык новыми словами и если да, то каким образом? Для человека это не проблема: ему достаточно объяснить, что за данными словами скрывается такая-то последовательность действий. Скажем, говорит мама: «Сходи за хлебом», — и каждому ясно, какой алгоритм надо исполнить. Конечно, когда-то, посылая вас в первый раз с таким поручением, она подробно объясняла, как его исполнить. Вот бы и для наших Бездумных Исполнителей предусмотреть такую возможность: один раз объяснил, дал этому объяснению название и затем пользуйся этим названием как командой.

Оказывается, большинство БИ такую возможность допускают, и называется она созданием **вспомогательного алгоритма**. Вспомогательный алгоритм, написанный с использованием исключительно системы команд конкретного БИ, называется **подпрограммой**. А чтобы лучше разобраться со вспомогательными алгоритмами, посмотрим сначала, как ими пользуются люди.

Откроем, например, полюбившуюся нам книгу кулинарных рецептов и обнаружим, что очень многие мясные блюда, скажем ту-

шеная баранина, просто немыслимы без, например, белого соуса. И совершенно ни к чему десяток раз писать, как его приготовить. Достаточно в нужном месте просто сослаться на страницу, где написан его рецепт. Иными словами, если в разных местах алгоритма есть совершенно одинаковые повторяющиеся последовательности действий, имеет смысл объявить одну такую последовательность вспомогательным алгоритмом и облегчить себе работу по написанию основного алгоритма.

Облегчится при этом не только создание, но и понимание алгоритма. Сравните, к примеру, два варианта одного и того же, в сущности, алгоритма:

АЛГОРИТМ РЕМОНТА КВАРТИРЫ

Освободить место для работы в первой комнате;
Отремонтировать в первой комнате потолок;
Отремонтировать в первой комнате стены;
Отремонтировать в первой комнате пол;
Освободить место для работы во второй комнате;
Отремонтировать во второй комнате потолок;
Отремонтировать во второй комнате стены;
Отремонтировать во второй комнате пол;
Освободить место для работы в третьей комнате;
Отремонтировать в третьей комнате потолок;
Отремонтировать в третьей комнате стены;
Отремонтировать в третьей комнате пол;
(*конец программы*)

Главный алгоритм:
Ремонт комнаты (первой);
Ремонт комнаты (второй);
Ремонт комнаты (третьей);
(*конец главной программы*)
Вспомогательный алгоритм:
Ремонт комнаты (номер);
Освободить место для работы в <номер> комнате;
Отремонтировать в <номер> комнате потолок;
Отремонтировать в <номер> комнате стены;
Отремонтировать в <номер> комнате пол;
(* конец вспомогательного алгоритма *)

Посмотрите внимательно на правую колонку текста. Все ли вам в ней понятно? Если вы внимательно читаете наш учебник, то у вас непременно возникли следующие вопросы:

- 1. Что такое «Ремонт комнаты (первой)»?

Допустим, что это команда и она как-то связана со вспомогательным алгоритмом, в котором есть похожая фраза «Ремонт комнаты (номер)». Как-никак совпадают два первых слова и скобки.

● 2. Является ли указанная фраза во вспомогательном алгоритме командой?

Ответ на второй вопрос: «Нет». Эта фраза называется **заголовком вспомогательного алгоритма**. Именно по нему БИ определяет, тот ли это алгоритм, выполнять который его послали (ведь в алгоритме может быть несколько вспомогательных алгоритмов).

Теперь отвечаем на первый вопрос. Написанная фраза — это и есть неверно оформленное задание для БИ на поиск и выполнение вспомогательного алгоритма. Правильно оформленный **вызов вспомогательного алгоритма** выглядит так:

Вызвать Ремонт комнаты (первой);

Отличие от того, что было, небольшое, но принципиальное. Ведь оператор Вызвать сообщает БИ, что он будет иметь дело со вспомогательным алгоритмом, а не с отдельным допустимым действием.

Исправим наш алгоритм, немного модернизировав и его внешний вид: текст алгоритма и текст вспомогательного алгоритма заключим в фигурные скобки. Эти скобки, как вы помните, называются операторными.

```
Алгоритм
{   Вызвать Ремонт комнаты (первой);
    Вызвать Ремонт комнаты (второй);
    Вызвать Ремонт комнаты (третьей)
}

Вспомогательный алгоритм Ремонт комнаты (номер)
{   Освободить место для работы в <номер> комнате;
    Отремонтировать в <номер> комнате потолок;
    Отремонтировать в <номер> комнате стены;
    Отремонтировать в <номер> комнате пол
}
```

Так что, с одной стороны, фигурные скобки окаймляют несколько операторов, например в операторе Если (и за одно это достойны называться операторными), а с другой стороны, показывают, что, скажем, весь вспомогательный алгоритм может рассматриваться как один большой оператор, состоящий из нескольких операторов. Совокупность операторов программы (подпрограммы), заключенная в операторные скобки, называется **телом программы (подпрограммы)**.

Как и раньше, договоримся подчеркивать оператор Вызвать, поскольку он присутствует всегда, когда есть хотя бы один вспомогательный алгоритм.

Теперь скажем несколько слов о том, как оформляются подпрограммы. После слова Подпрограмма обязательно записывается имя подпрограммы: ведь подпрограмм может быть несколько и БИ должен точно знать, какую подпрограмму его отправили исполнять.

Опять что-то неясно? Наверно, у вас возникли такие вопросы:

- 3. Что должен делать БИ после окончания подпрограммы? Прекратить свою деятельность или вернуться к своим делам в той программе, из которой его послали исполнять подпрограмму?
- 4. А если вернуться, то к какому оператору?
- 5. Откуда БИ знает номер комнаты, которую надо ремонтировать? БИ в силу своей бездумности требует четких инструкций на этот счет. Где они?

Разберемся с вашими очередными вопросами. Действительно, существует стандартный **оператор Возврат**, который дает БИ команду вернуться в вызывающий алгоритм. Поэтому наш вспомогательный алгоритм следовало бы закончить командой

Возврат;

Этот оператор, как и оператор Вызвать, мы будем подчеркивать.

Возврат из вспомогательного алгоритма (подпрограммы) осуществляется на следующий оператор после того оператора Вызвать, который отправил БИ во вспомогательный алгоритм (или подпрограмму). Во вспомогательном алгоритме (подпрограмме) разрешается иметь несколько операторов Возврат.

Введя пару новых операторов, перейдем к более сложной проблеме — ответу на пятый вопрос. Для этого вернемся к нашим тушеным баранам. Встретив в рецепте фразу «Приготовить 200 г белого соуса», хозяйка, как истинная БИ, бросится его готовить, быть может, даже забыв в предпраздничной суете, зачем этот соус нужен. Ничего страшного не случится. Главное, чтобы она помнила количество: 200 г. В лучшем случае она даже может перепоручить это дело кому-нибудь другому, написав на бумажке заветное число — 200 г.

Мы тоже будем считать, что БИ напрочь забывает все из основной программы, приступая к выполнению подпрограммы. Единственное, что он помнит, — это числа, заботливо положенные в основной программе в коробочку, называемую в нашем случае «номер». Иначе говоря, в подпрограмме должна быть заранее подготовлена коробочка для необходимой ей информации. Дело БИ — эту коробочку заполнить, прежде чем по уши уходить в подпрограмму. БИ уподобляется при этом рассеянному профессору, который сам себе писал письма с напоминанием, что требуется сделать. Ну что ж, пусть и бездумный, а хоть чем-то смахивает на профессора...

И тут поспевает очередной вопрос:

- 6. Куда девать белый соус?

Разумеется, раз есть коробочка с заданием, должно быть и блюдечко с голубой каемочкой, на котором БИ преподнесет **результат работы подпрограммы**.

Под конец приведем-таки правильно оформленный алгоритм с вызовом вспомогательного алгоритма:

Алгоритм
{ Вызвать Ремонт комнаты (первой);
Вызвать Ремонт комнаты (второй);
Вызвать Ремонт комнаты (третьей)
} (* конец главного алгоритма*)

Вспомогательный алгоритм Ремонт комнаты (коробочка: номер)
{ Освободить место для работы в <номер> комнате;
Отремонтировать в <номер> комнате потолок;
Отремонтировать в <номер> комнате стены;
Отремонтировать в <номер> комнате пол;
Возврат
} (* конец ремонта комнаты *)

На коробочке написано «номер». Именно туда БИ и помещает номер той комнаты, которую ему надо отремонтировать. Блюдечка в этом вспомогательном алгоритме не требуется.

ВОПРОСЫ И ЗАДАНИЯ

1. Как вы думаете, почему возникли именно такие названия: «вспомогательный алгоритм» и «подпрограмма»?
2. Приведите примеры определений новых операторов и создания вспомогательных алгоритмов для известных вам БИ.
3. БИ умеет:
   - брать бумажку с написанным на ней числом;
   - записывать одно число с бумажки на доску;
   - сравнивать два числа (определять, какое больше) — на бумажке или на доске;
   - стирать число с доски.

   а) Напишите для него программу поиска максимального числа из трех чисел.

   б) Используя механизм подпрограммы, напишите программу поиска максимального числа из пяти чисел.
4. БИ умеет:
   - порезать X на кусочки;
   - завернуть X в Y и положить в Z;
   - сварить X;
   - держать X на огне до готовности продукта;
   - вынуть X из Y.

Из этих действий составлен следующий вспомогательный алгоритм:

Подпрограмма Изготовление (коробочка: X,Y,Z, блюдечко: W)
{ Порезать X на кусочки;
Завернуть X в Y и положить в Z;

Держать *Z* на огне до готовности продукта;
Вынуть *W* из *Z*;
Возврат;
}

Используя этот вспомогательный алгоритм, составьте алгоритмы приготовления:
а) сибирских пельменей (исходные продукты: мясо, лук, тесто);
б) блинчиков с мясом (исходные продукты: мясо, лук, блинчики);
в) беляшей (исходные продукты: мясо, лук, тесто);
г) пирожков с капустой (исходные продукты: капуста, яйцо, лук, тесто);
д) голубцов (исходные продукты: капуста, лук, мясо);
е) чего-нибудь еще вкусненького.

5. а) Ниже приведена программа мытья посуды после чаепития:

Программа
{ Вызвать Мыть (чашки);
Вызвать Мыть (блюдца);
Вызвать Мыть (ложки)
} (* конец программы *)

Составьте подпрограммму Мыть (коробочка: посуда), если исполнитель понимает такие команды:
- открыть кран;
- закрыть кран;
- взять посуду (по этой команде исполнитель берет со стола 1 экземпляр посуды);
- вымыть посуду под краном (по этой команде исполнитель моет имеющийся у него экземпляр посуды);
- положить посуду в сушилку (исполнитель кладет имеющийся у него экземпляр посуды в сушилку).

Совет: чтобы исполнитель наверняка перемыл всю посуду, вспомните о циклической форме организации действий.

б) Составьте программу мытья посуды после обеда, использующую подпрограмму Мыть (коробочка: посуда).

в) Программы, составленные в пунктах а) и б), вполне годятся для домашнего робота БИ-001 (о нем шла речь в задаче 8 из § 31, задаче 12 из § 32 и задаче 11 из § 35). Составьте еще две-три программы для БИ-001, в которых использовалась бы циклическая форма организации действий и вспомогательный алгоритм «Мыть».

Подпрограммы в работе Паркетчика

В этой лабораторной работе мы не торопясь разберем задачу, поставленную перед Паркетчиком очередным заказчиком.

Задача
(закраска поля разноцветными прямоугольниками)

Поле произвольных размеров, но такое, что количество клеточек по горизонтали и вертикали делится на 2, требуется закрасить так, как это показано на рисунке 33.

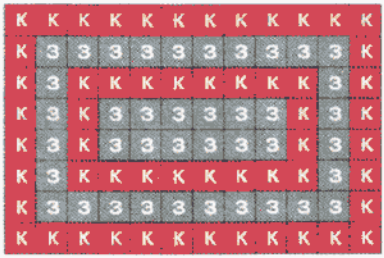


Рис. 33

Мы уже не раз обегали поле по периметру, выкладывая разноцветные бордюры, так что и эта задача не отличается какой-то особой новизной. Просто надо обеспечить, чтобы Паркетчик выкладывал плитки определенного цвета не только до стенки, а либо до стенки, либо до выложенных плиток какого-либо цвета.

Значит, вполне можно себе представлять Паркетчика стоящим в левом нижнем углу незакрашенного участка с задачей выложить бордюр (рис. 34).

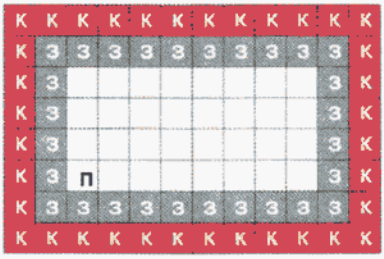


Рис. 34

Тогда проект первого уровня может выглядеть так:

```
Делать пока ((не к) и (не з))
{  Выложить внутри пустого пространства красный бордюр;
   Шаг вверх;
   Шаг вправо;
```

```
    Выложить внутри пустого пространства зеленый бордюр;
    Шаг вверх;
    Шаг вправо;
}
```

Тут существенно использовано условие задачи о четности ширины и высоты поля. Именно это дает возможность всегда выложить сразу пару бордюров — красный и зеленый.

Теперь заметим, что непонятные Паркетчику операторы:

```
Выложить внутри пустого пространства красный бордюр;
Выложить внутри пустого пространства зеленый бордюр;
```

легко заменяются обычным вызовом подпрограммы с параметром:

```
Вызвать Бордюр(к);
Вызвать Бордюр(з);
```

Таким образом, вместо проекта первого уровня получается готовая главная программа.

Давайте теперь четко сформулируем, что же должен сделать Паркетчик, выполнив подпрограмму Бордюр.

Уточняющая дополнительная задача

Паркетчик стоит на частично закрашенном поле произвольных размеров так, как это показано на рисунке 34. Требуется выложить очередной внутренний слой плиток заданным цветом (полученным из главной программы с помощью передачи параметров) и оказаться на исходной клеточке (потому что это необходимо для безошибочной работы главной программы).

Казалось бы, достаточно четырежды написать нечто вроде:

```
Делать пока ((не справа стена) и (не К) и (не З))
{ Положить (ц);
  Шаг вправо
}
```

— и дело сделано.

Однако все не так просто, как это кажется на первый взгляд. Дело в том, что Паркетчик умеет проверять, находится ли справа от него стена, однако не умеет впрямую проверять, находится ли справа от него красная плитка. Стало быть, первый бордюр будет выложен правильно, а вот с последующими выйдет накладка.

Посмотрите внимательно: выкладывая все бордюры, кроме первого, Паркетчик выйдет из написанного выше цикла тогда, когда будет стоять на зеленой или красной плитке. Значит, сразу бежать вверх и выкладывать очередную вертикальную полоску нельзя. Другими словами, если Паркетчик при выкладывании полоски наткнется не на стену, а на какие-нибудь цветные плитки, необходимо сделать еще кое-какие действия.

Будем надеяться, они не окажутся для вас тайной за семью печатями и вы сможете дописать подпрограмму до конца. А затем и отладить ее.

§ 37. Обзор вычислительных Бездумных Исполнителей

Мы неустанно твердили, твердим и будем твердить, что любой компьютер — Бездумный Исполнитель со своей специфичной системой команд. При этом подавляющее большинство компьютеров имеет совершенно «нечеловеческую» систему команд. И общаться с ними на их родном языке весьма затруднительно.

Поэтому программисты используют в своей работе какой-либо промежуточный язык, более понятный человеку. Один из таких языков и есть язык исполнителя Паркетчик. Это, пожалуй, самый «человечный» из промежуточных языков, потому что он в буквальном смысле слова реализует главный принцип:

Основной задачей программирования является разработка проекта до такого уровня, чтобы в нем остались только стандартные конструкции (те самые, которые мы всегда подчеркивали) и операторы из системы команд БИ, для которого пишется программа.

Для Паркетчика, как вы помните, мы без всяких изменений используем эти самые основные конструкции. Но профессиональные программисты работают с огромным количеством специализированных и универсальных промежуточных языков. И каждый из них по-своему пытается имитировать живую человеческую речь. К счастью, они не слишком далеко отстоят от языка Паркетчика.

И мы сейчас на примере двух достаточно распространенных языков покажем, как за весьма непродолжительное время можно выучить любой «настоящий» язык программирования. Конечно, после изучения этого параграфа вы не станете специалистами в этих языках, не постигнете все их тонкости. Но, ведь и изучая иностранный язык, вы не стремитесь сразу овладеть им так, чтобы писать поэмы и романы. Вам хочется, чтобы вас понимали. И здесь то же самое — мы расскажем, как овладеть языком настолько, чтобы компьютер вас понимал. А если уж захотите, можете совершенствовать свое знание языка самостоятельно — для этого всегда есть соответствующие руководства.

Изучать вы будете Бейсик (QBasic) и Паскаль (Pascal) (если два языка для вас многовато — изучайте один!). Что значит изучить язык? Для обычного иностранного языка это значит выучить перевод слов и правила их соединения в предложения. То же самое и для **языков программирования**. К счастью, в языках программирова-

ния используется очень мало слов (как правило, чуть больше двух десятков), а правил их соединения еще меньше. Правила эти, как вы, наверно, догадались, в точности те самые алгоритмические конструкции ветвлений, циклов и подпрограмм, с которыми вы познакомились, работая с Паркетчиком. Вот и посмотрим, как они переводятся на языки QBasic и Pascal.

Удобно себе представлять, что каждый из языков программирования — это язык некоторого Бездумного Исполнителя, который понимает команды, отдаваемые ему на этом языке. Тогда, как обычно, для исполнителя есть среда, состоящая из тех объектов, над которыми он может выполнять допустимые действия. Если вы еще раз прочтете заголовок параграфа, то вам станет ясно, что речь у нас пойдет об исполнителях, которые вычисляют. Значит, их объекты — числа, а допустимые действия — это действия над числами.

Конечно, пройдя с нами в этом учебнике уже такой длинный путь, вы понимаете, что вычисления — это всего лишь одна из многих разновидностей обработки информации (хотя именно ей компьютер обязан своим названием). Поэтому настоящие языки программирования, как и управляемые с их помощью исполнители, способны отнюдь не только к вычислениям. Чтобы описать все возможности какого-либо языка программирования, пишут специальное и довольно большое руководство. Мы же, как уже говорилось, не намерены погружать вас во всевозможные тонкости, поэтому ограничимся лишь простейшими вычислительными «способностями» двух указанных выше языков.

Итак, давайте вместе разберемся, что должен уметь делать вычисляющий исполнитель. Разумеется, ему необходимо уметь получать от нас числовую информацию, сохранять ее в памяти, выполнять над ней арифметические операции и сообщать нам о получившихся результатах. Получение информации Бездумным Исполнителем мы будем обозначать допустимым действием «Запросить», а выдачу им результатов — действием «Сообщить». Арифметические операции в языках программирования обозначаются, как правило, стандартными математическими знаками.

Что касается хранения информации, то об этом поговорим особо. Напомним, что физически хранение информации в компьютере представляет собой магнитную (или какую-нибудь другую) ее запись на носитель. В § 5, обсуждая понятие файловой системы, мы поясняли, что для различения таких записей каждой из них присваивается имя. В языках программирования, когда речь идет о хранении информации, нередко используется понятие **переменной**. Каждая переменная тоже обязательно имеет **имя**. Из переменных могут конструироваться другие *структуры данных*, например таблицы, стеки, графы. Но мы не будем обсуждать такие структуры в нашем довольно ограниченном по времени курсе, считая, что заинтересовавшиеся могут изучить их по соответствующим руководствам по языкам

программирования. Нам сейчас важнее разобраться с понятием переменной.

Мы уже сказали, что каждая переменная имеет имя. Информация, которая хранится в переменной, называется ее **значением**. Чтобы переменная получила некоторое значение, используют **оператор присваивания**. Этот оператор мы будем обозначать, так же как и в большинстве языков программирования, символом «:=». Например, если у нас имеется переменная с именем S, то присвоить ей значение 3 можно командой

S:= 3

Если после этого мы предложим выполнить команду

S:= S + 5

то значение переменной S изменится и станет равным 8.

Заставив компьютер выполнить затем команду

S:= S * S

мы для переменной S получим значение 64. При этом каждый раз предыдущее значение переменной будет компьютером бесследно забыто.

Если слева от знака присваивания всегда указывается имя подходящей переменной, то справа пишется выражение, состоящее из констант и имен переменных, связанных знаками операций, допустимых для данного исполнителя. Если это числовые переменные и константы (т. е. просто числа), то мы будем для краткости называть такое выражение **арифметическим**.

В каждом языке программирования свои правила образования имен переменных. Паркетчик, например, понимает имена, написанные только русскими буквами и цифрами, притом содержащими не более восьми знаков. Другие языки, наоборот, требуют в именах только латинские буквы (в сочетании с цифрами) и довольно часто весьма короткие. Каждый раз об этих правилах надо справиться в соответствующем руководстве.

Но кроме имени и значения, у переменной еще есть **тип**. Он указывает компьютеру, как можно обрабатывать ту информацию, которая хранится в переменной. Ведь каждому ясно, что, например, сложение натуральных чисел выполняется совсем не так, как обыкновенных дробей (хотя для сложения дробей нужно, конечно, уметь складывать натуральные числа). Да и значение переменной совсем не обязательно является числом — это может быть текст, или символ, или файл, или еще что-нибудь, что и складывать-то нельзя. Образно говоря, переменная — это тара для данных, которые обрабатывает исполнитель. И так же как вещество той или иной природы нужно хранить в специально приспособленной для этого таре — никто ведь не носит воду в решете, а яблоки в бутылках, — для каждого типа данных требуется свой тип переменных.

Для числовой информации обычно рассматриваются два **типа данных**: **целый** и **вещественный**. Намереваясь использовать в задаче те или иные числовые данные, вы должны каждый раз понять и указать компьютеру, какого типа будут переменные для хранения чисел, возникающих в ходе решения.

Разумеется, различные Бездумные Исполнители имеют дело не только с числовой информацией. Поэтому и типы данных, с которыми они умеют работать, могут быть весьма разнообразными. Например, Паркетчик, кроме типа «целый», знает еще тип «цветной». Зато он не знает тип «вещественный», поскольку плитки и клетки на поле не бывают дробными.

Узнав все, что вам нужно, про переменные, познакомьтесь теперь с двумя уже упоминавшимися языками программирования. Изучите для этого таблицу 16.

Таблица 16

Паркетчик	QBasic	Pascal
1. Начальные операторы		
Программа { ..... }	Можно не писать ничего	Program <имя>; begin end
2. Описание числовых переменных		
цел: <список переменных>— целочисленные переменные	Если описание переменной отсутствует, то она — вещественная; Defint<список переменных> — целочисленные переменные	Var <список переменных>: integer; — целочисленные переменные Var <список переменных>: real; — вещественные
3. Оператор условия		

Если (<условие>) то
{<оператор>; ... }
иначе {<оператор>;...}
	IF (<условие>) THEN
	<оператор>
	<оператор>
	...
	ELSE
	<оператор>
	...
	END IF
		IF (<условие>) THEN
		begin
		<оператор>;
		...
		end
		ELSE
		begin
		<оператор>
		...
		end;

Во всех трех языках конструкция ИНАЧЕ (ELSE) может отсутствовать. В этом случае в языке Pascal необходимо после последнего оператора END поставить точку с запятой.		

Продолжение

Паркетчик	QBasic	Pascal
4. Оператор цикла по условию		

Делать пока (<условие>)
{<оператор>;... }
	WHILE (<условие>)
	<оператор>
	<оператор>
	...
	WEND
		WHILE(<условие>) DO
		begin
		<оператор>
		...
		end;

5. Оператор цикла с параметром		

Делать от <переменная>:
=<арифм. выражение>
до <арифм. выражение>
с шагом
<арифм. выражение>
{<оператор>;...}
	FOR <переменная> =
	<арифм. выражение>
	TO <арифм. выражение>
	STEP <арифм. выражение>
	
	<оператор>
	...
	NEXT <переменная>
	Если нужен шаг не 1,
	а –1, вместо TO пишем
	DOWNTO
		FOR <переменная> =
		<арифм. выражение>
		TO <арифм. выражение>
		DO
		begin
		<оператор>
		...
		end;

6. Работа с подпрограммами		

Подпрограмма
<имя подпрограммы>
(короб:
<список параметров>,
блюд:
<список параметров>)
{<оператор>; ... }
	SUB <имя>
	(<список параметров>)
	<оператор>
	...
	ENDSUB
	...
		Procedure <имя>
		(<список параметров
		с указанием их типов>)
		begin
		<оператор>
		
		end;

Обращение к подпрограмме производится оператором:		
Вызвать <имя подпрограммы> (короб: <список параметров>; блюд: <список параметров>);	CALL <имя> (<список параметров>)	<имя> (<список параметров>)
Для возврата в главную программу служит оператор:		
Возврат;	EXITSUB или ничего	EXIT;
7. Операторы ввода и вывода информации		
Ввести <имя переменной>; Сообщить <имя переменной или/и сообщение в кавычках>;	INPUT"<уточняющее сообщение>"; <список переменных через запятую> PRINT "<сообщение>"; <список переменных через запятую или точку с запятой>	READLN (<имя переменной>); WRITE('<сообщение>', <имя переменной>;

И наконец приведем запись одного и того же алгоритма сразу на трех языках. Это вычисление члена ряда Фибоначчи по его номеру, который задается пользователем. Название для вас, безусловно, новое, но, если вы проанализируете результаты, все станет ясно.

ПАРКЕТЧИК

```
Программа
цел: a1, a2, Раб, Номер, Счетчик;
{Запросить Номер;
a1:=0; a2:=1;
Делать от Счетчик :=1 до Номер
   {Раб:=a1+a2;
    a1:=a2;
    a2:=Раб;
   }
Сообщить «Член с номером», Номер;
Сообщить «равен», a2;
}
```

QBASIC

```
INPUT «Введи номер члена», N
A1 = 0
A2 = 1
FOR i = 1  TO  N
   R = A1 + A2
   A1 = A2
   A2 = R
NEXT i
PRINT N,«-й член равен», A2
```

PASCAL

```
PROGRAM Fib;
VAR A1, A2, N, R, I : INTEGER;
BEGIN
   READLN(N);
   A1:= 0;
   A2:= 1;
   FOR I: = 1  TO  N DO
      BEGIN
         R:= A1 + A2;
         A1:= A2;
         A2:= R;
      END;
   WRITE (N ,'-й член равен ', A2)
END.
```

Ни один из трех языков в написании своих операторов не различает маленькие и большие буквы, поэтому пишите так, чтобы программа для вас выглядела красиво.

Подчеркнем еще раз, что оба языка (QBasic и Pascal) отнюдь не исчерпываются приведенными выше конструкциями. Но мы и не ставим перед вами цель овладеть ими в совершенстве. Того «ядра», которое описано, вам с лихвой хватит для работы с учебником.

ВОПРОСЫ И ЗАДАНИЯ

1. Что значит изучить язык программирования? Перечислите основные алгоритмические конструкции и запишите их на том языке, который вы изучаете.
2. Чем характеризуется переменная в языке программирования? Что называют значением переменной?
3. Почему бывают переменные разных типов? Зачем нужно указывать тип переменной? Какие типы числовых переменных вам известны?
4. С помощью какой операции можно поменять значение переменной?
5. Что в языке программирования называют арифметическим выражением?
6. Напишите какую-нибудь несложную программу (скажем, решения квадратного уравнения) на одном из доступных вам языков программирования.

Первая работа с «настоящим» языком программирования

Мы попали в сложное положение. Очень может быть, что в вашем распоряжении такой вычислительный БИ, о котором мы даже и не подозреваем. Поэтому засучите рукава и:

❶ Попробуйте, используя текстовый редактор вашего БИ, ввести программу вычисления члена ряда Фибоначчи.

Если все прошло нормально,

❷ Оттранслируйте программу (аналогично процессу трансляции в Паркетчике) и запустите ее.

❸ Если обнаружились ошибки — исправьте их и добейтесь правильной работы программы.

И наконец, если ваша первая программа удачно завершилась,

❹ Введите программу, написанную вами при выполнении задания 6 и отладьте ее.

§ 38. Рекуррентные соотношения

Давайте рассмотрим следующую задачу:

Петя и Коля прочитали в одной книге по математике, что сумма

$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\frac{1}{6}+\frac{1}{7}+\ldots+\frac{1}{N}$$

может стать больше любого числа, если только выбрать N достаточно большим.

Ребята очень удивились этому — ведь каждый раз добавляется все меньшее и меньшее число. И они решили попытаться найти такое N, чтобы указанная сумма была не меньше 20. Для этого Петя и Коля составили следующий алгоритм, который затем каждый из них попытался исполнить на своем калькуляторе:

```
S = 1;
N = 1;
Делать пока (S<20)
   {   N = N + 1;
       S = S + 1/N;
   } (*конец цикла*)
Сообщить N;
```

Убедитесь, что этот алгоритм действительно решает поставленную задачу.

Сначала разберемся, что же делается в приведенном алгоритме. Нам требуется найти сумму N чисел. Но доступный нам исполнитель алгоритмов (калькулятор) умеет складывать только два числа. Поэтому нахождение суммы большего числа слагаемых мы заменяем на несколько последовательных сложений двух чисел, постепенно накапливая сумму в переменной S. Если через S_K обозначить сумму первых K слагаемых, то можно записать такое равенство:

$$S_N = 1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\ldots+\frac{1}{N-1}+\frac{1}{N}.$$

Из него нетрудно усмотреть, что

$$S_N = S_{N-1}+\frac{1}{N}.$$

Посмотрите внимательно: полученная формула показывает, что N-е значение переменной S получается из $(N-1)$-го значения прибавлением $\frac{1}{N}$.

Когда существует правило, позволяющее находить N-е значение переменной, используя ее предыдущие значения, говорят, что *переменная задана рекуррентно*. Само правило при этом называют **рекуррентным соотношением**.

Рекуррентные соотношения очень часто применяют для вычисления тех или иных длинных сумм. Как мы видим, Петя и Коля тоже воспользовались этим приемом, даже о нем не подозревая. Но рекуррентные соотношения полезны отнюдь не только для суммирования.

Вот еще одна история, приключившаяся с Петей и Колей. Однажды Петя обнаружил, что на его микрокалькуляторе нет операции извлечения квадратного корня. И конечно, очень расстроился — ведь на таком калькуляторе даже квадратное уравнение не решишь. Но Коля ему сказал, что в одной умной книге по высшей математике (как известно, математика делится на школьную и высшую) он нашел вот какое рекуррентное соотношение:

$$X_N = \frac{1}{2}(X_{N-1} + \frac{A}{X_{N-1}}).$$

Там же утверждалось, что, какое бы положительное X_1 мы ни взяли, значение X_N с ростом N все ближе и ближе становится к числу A.

Поскольку на калькуляторе все равно лишь ограниченное число знаков, то можно получить A с максимально возможной точностью. Петя очень обрадовался и тут же составил вот какую программу:

```
Программа
   Вещ A, X;
   Цел N;
   {Ввести A;
   Если (A<0) то {Сообщить "Корень не извлекается";}
      Иначе
         {X=1; (* начальное приближенное значение корня *)
         N=0; (* N — счетчик числа выполнений тела цикла *)
         Делать пока ( |A – X*X| >0.0001)
            {X=0.5*(X+A/X);
            N=N+1;
            Сообщить N;
            Сообщить X (* N-е приближение к корню *)
            }(*конец цикла *)
         } (* конец ветвления *)
```

Вот какие результаты получил Петя на своем калькуляторе для $A = 25$:

N	1	2	3	4	5
X_N	13	7.4615	5.4060	5.0152	5.0000

Как видите, достаточно хорошее значение корня получилось уже на 5-м шаге.

А теперь расскажем еще одну историю, в которой главным действующим лицом будет уже не Петя, а Паркетчик.

Однажды поступил заказ на изготовление нескольких дорожек длины N (и ширины 1) из красных и зеленых плиток. При этом заказчик выдвинул условие, чтобы все дорожки были различны и в каждой дорожке не было двух красных плиток рядом. Естественно,

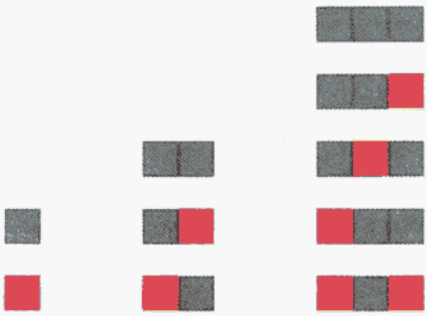


Рис. 35

возникает вопрос: сколько таких дорожек фиксированной длины N может изготовить Паркетчик?

Каждому ясно, что дорожек длины 1 можно сделать только две — одну зеленую и одну красную (рис. 35, *а*). Дорожек длины 2 существует 3 (рис. 35, *б*). Дорожек длины 3 — уже 5 штук (рис. 35, *в*).

Чтобы получить все дорожки длины 4, можно поступить так:

Если последняя плитка зеленая, то перед ней может лежать плитка любого цвета, т. е. перед такой плиткой может быть любая дорожка длины 3. Значит, дорожек длины 4, оканчивающихся на зеленую плитку, ровно 5 (рис. 36, *а*).

Если же последняя плитка — красная, то перед ней лежит обязательно зеленая. А перед зеленой может располагаться любая дорожка длины 2. Значит, дорожек длины 4, оканчивающихся на красную плитку, ровно 3 (рис. 36, *б*).

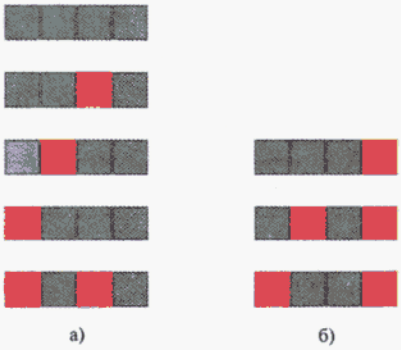


Рис. 36

Ясно теперь, что общее число дорожек длины 4 будет 8.

Такими же рассуждениями можно найти, что число дорожек длины 5 будет 13. И вообще если X — количество дорожек длины N, то имеет место такое равенство:

$$X_N = X_{N-1} + X_{N-2}.$$

Это тоже рекуррентное соотношение — ведь в нем N-е значение переменой X вычисляется по двум предыдущим значениям. Например, $X_6 = X_5 + X_4 = 13 + 8 = 21$, т. е. существует ровно 21 дорожка длины 6.

Как видим, и в этой задаче оказались полезными рекуррентные соотношения.

ВОПРОСЫ И ЗАДАНИЯ

1. Что такое рекуррентное соотношение?
2. а) Напишите программу вычисления суммы чисел, обратных квадратам первых ста натуральных чисел.

   б) Напишите программу, согласно которой будет запрошено несколько натуральных чисел и сообщено их среднее арифметическое (количество запрашиваемых чисел тоже запрашивается).
3. Напишите программу вычисления следующей алгебраической суммы:

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \frac{1}{7} - \frac{1}{8} + \ldots + (-1)^N \cdot \frac{1}{N}.$$

4. Как бы вы объяснили, почему в программе вычисления квадратного корня условием продолжения цикла Петя выбрал неравенство $|A - X*X| > 0.0001$?
5. а) Заказчик хочет иметь паркет шириной в 7 клеток и длиной в 35 клеток. На каждой клетке должна лежать либо красная, либо зеленая плитка, при этом в паркете не должно быть двух одинаковых вертикальных полос и в каждой вертикальной полосе никакие две красные плитки не могут быть соседними. Осуществимо ли желание заказчика?

   б) Напишите программу вычисления количества дорожек длины N, удовлетворяющих требованию, чтобы никакие две красные плитки не лежали рядом. Число N запрашивается в начале исполнения этой программы.
6. Представьте, что Паркетчик имеет в своем распоряжении еще и синие плитки.

   а) Напишите программу вычисления количества дорожек длины N (и ширины 1), удовлетворяющих требованию, чтобы никакие две красные плитки не лежали рядом. Число N запрашивается в процессе исполнения этой программы.

   б) Напишите программу вычисления количества дорожек длины N (и ширины 1), удовлетворяющих требованию, чтобы никакие две красные плитки и никакие две синие плитки не лежали рядом. Число N запрашивается при исполнении этой программы.

7. Последовательность значений переменной X строится так:

$$X_1 = 1;\ X_2 = 3;\ \ldots;\ X_n = X_{n-2} - 2X_{n-1}$$

для каждого $n > 2$.

а) Чему равны X_3, X_4, X_7?

б) Напишите и отладьте следующие программы:
программу нахождения первого значения, большего 1000;
программу нахождения суммы первых пятнадцати значений переменной X;
программу нахождения первых десяти положительных значений переменной X.

8*. Иногда рекуррентно задаются значения не одной, а нескольких переменных. Вот пример *системы двух рекуррентных соотношений*:

$$A_n = A_{n-1} - 2B_{n-1};$$
$$B_n = 2A_{n-1} + B_{n-1}.$$

а) Найдите значения A_3 и B_3, если $A_1 = 2$ и $B_1 = 1$.

б) Напишите программу вычисления A_n и B_n по запрашиваемым n, A_1 и B_1.

9*. Пусть в выражении $(1+\sqrt{2})^n$ раскрыты скобки (здесь n — натуральное число). Нетрудно понять, что для каждого конкретного n у нас будет получаться выражение вида $a_n + b_n\sqrt{2}$.

Например, при $n = 3$ имеем $7 + 5\sqrt{2}$, т.е. $a_3 = 7$, $b_3 = 5$ (проверьте).

а) Напишите программу, после выполнения которой по заданному n будет сообщено a_n и b_n. (Совет: составьте систему рекуррентных соотношений, связывающих a_n, b_n с a_{n-1}, b_{n-1}.)

б) Введем новую переменную $c_n = \dfrac{a_n}{b_n}$. Напишите программу для вычисления c_n, использующую не более трех переменных. (Совет: попытайтесь найти рекуррентное соотношение, связывающее c_n и c_{n-1}.)

Работаем с рекуррентными соотношениями

Недолго раздумывая, возьмем быка за рога:

❶ В качестве разминки отладьте программу, которую вы написали, выполняя задание 3.

❷ Для вычисления кубического корня из положительного числа A пользуются следующим рекуррентным соотношением:

$$X_N = \frac{1}{3}\left(2X_{N-1} + \frac{A}{X_{N-1}^2}\right).$$

Напишите и отладьте программу вычисления кубического корня. Поэкспериментируйте для различных A, на каком шаге получается приемлемое значение кубического корня.

❸ Попытайтесь угадать, как могло бы выглядеть рекуррентное соотношение для вычисления корня четвертой степени. Напишите соответствующую программу и проверьте свою гипотезу для различных исходных данных.

А теперь продолжим работу.

❹ Отладьте программы, которые вы написали, выполняя задание 7 б).

❺ Отладьте программу, которую вы написали, выполняя задание 9 б).

5 Компьютерные модели в задачах управления

До сих пор мы рассматривали динамические системы, считая, что факторы, влияющие на функционирование системы, неизменны. Они заданы самой природой системы и не подвергаются воздействиям человека. На самом деле, человек, познавая природу и общество, все активнее и шире вмешивается в действие этих факторов, иногда не осознавая своего влияния, но чаще сознательно, преследуя конкретную цель — заставить систему функционировать нужным человеку образом.

Целенаправленное воздействие на факторы динамической системы называется ***управлением*** *этой системой.*

В этой главе мы на примерах покажем, как информационные технологии и компьютерное моделирование могут применяться для решения задач управления.

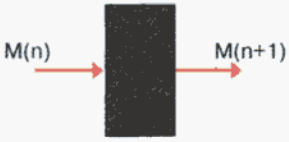
§39. Сколько можно взять у природы

Богата российская земля различными ресурсами. Ее недра хранят самые разнообразные полезные ископаемые. Ее леса служат источником для работы лесоперерабатывающей промышленности... Вот о добыче леса и пойдет у нас речь.

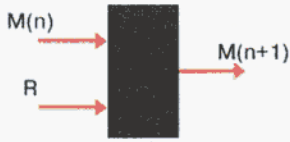
Лес относится к так называемым *возобновляемым ресурсам* — вы его рубите, а он снова вырастает на этом же месте, в отличие, скажем, от железной руды, которую если один раз добыл, то больше уже на этом же месте не найдешь. Поэтому руды получили название *невозобновляемых ресурсов*. Но о них мы здесь говорить не будем.

Лес, конечно, снова вырастет. Но растет он очень медленно, и если сразу вырубить леса слишком много, то можно и без леса остаться. Вот и возникает задача управления: сколько леса можно рубить ежегодно, чтобы обеспечить его нормальное воспроизводство?

Мы уже знаем модель прироста растительной массы без вмешательства человека — это модель ограниченного роста. Ее схематично можно изобразить черным ящиком с одним входом и одним выходом:



Теперь у нас появился еще один вход — воздействие человека:



Если мы считаем, что ежегодно вырубается одна и та же масса древесины, то наша модель ограниченного роста изменится по внешнему виду незначительно:

$$M(n+1) = M(n) + aM(n)(L - M(n)) - R.$$

Здесь через $M(n)$ по-прежнему обозначена масса (в данном случае древесины), которая имеется на данном участке через n лет.

Построенную модель назовем **моделью потребления возобновляемых ресурсов**.

ВОПРОСЫ И ЗАДАНИЯ

1. Что называется управлением динамической системой? Приведите примеры управления какими-либо динамическими системами.
2. Какие предположения положены в основу модели потребления возобновляемых ресурсов?
3. Подготовьте заполнение электронной таблицы для вычисления массы древесины через заданное количество лет, используя модель потребления возобновляемых ресурсов.

Управление добычей возобновляемых ресурсов

Для проведения компьютерного эксперимента вы можете сначала загрузить ту же таблицу, что и при выполнении лабораторной работы № 8 по модели ограниченного роста. Но теперь в ней надо

зарезервировать клетку, куда мы будем вводить значение R, и поменять формулу для вычисления $M(n)$. Полезно иметь еще столбец, где указывается ежегодный прирост: $aM(n)(L - M(n))$.

❶ Внесите в электронную таблицу необходимые изменения.

Промышленно добываемый лес не растет, конечно, в тундре, степи или пустыне. Так что мы будем работать со строкой «Тайга» и коэффициент прироста считать равным 1,8. Пусть у нас по-прежнему $L = 11\,000$ т. Введем начальную массу, близкую к этой границе, — считаем, что леспромхоз начал действовать в лесу, где еще никто ничего не успел натворить. Итак, пусть $M(0) = 10\,000$ т. Пусть принято решение с этой лесной делянки ежегодно брать 1000 т древесины.

❷ Выполните соответствующие вычисления в электронной таблице. Постройте графики изменения массы и прироста древесины по годам.

Интересно получается: величина прироста в точности совпадает с забираемой массой! Случайность? Проверим: попробуем забирать 3000 т.

❸ Выполните соответствующие вычисления в электронной таблице. Постройте графики изменения массы и прироста древесины по годам.

Видите: сначала прирост составил 3491 т, затем этот прирост уменьшился и через 15 лет стал равным 3000 т. Он снова сравнялся с забираемой массой. Поэкспериментируйте еще с несколькими значениями R. Но не жадничайте — не берите R больше чем 4000 т. Фактически вы сейчас «открыли» важное свойство живой природы — явление *саморегуляции*. Природа стремится восстановить то стабильное состояние, из которого ее вывели тем или иным способом.

А теперь попробуйте изъять 5000 т.

❹ Выполните соответствующие вычисления в электронной таблице. Постройте графики изменения массы и прироста древесины по годам.

Видите: через 6 лет масса древесины вдруг стала отрицательной. Реально такого, конечно, быть не может. Просто это означает, что от нашего хозяйствования лес погиб.

Значит, 5000 т — это слишком много. Ну а какую же наибольшую массу все-таки можно изымать?

❺ Организуйте и проведите соответствующий компьютерный эксперимент.

Вот мы и выяснили, каким должно быть здесь оптимальное управленческое решение.

§ 40. Задача об организации летнего отдыха

В окрестностях Екатеринбурга по берегам озера Шарташ раскинулся замечательный лесопарк «Каменные палатки». Когда-то он был практически за городом. Но теперь около него возник огромный жилой микрорайон, и с лесопарком стало происходить что-то неладное. Причина понятна: резко возрос поток посетителей и парк уже не справляется с их нашествием. Управлять потоком посетителей шарташского лесопарка невозможно. Но есть места (например, природные парки, заповедники и заказники), куда можно ограничить вход любителей отдохнуть на природе. Вот и возникла управленческая задача:

Как организовать посещение природного парка, чтобы не нанести ему непоправимого ущерба?

Вы уже привыкли, что решение задачи надо начинать с построения модели. Здесь нужно прежде всего решить, что значит «организовать посещение». Можно это понимать так: парк открыт для посещения несколько дней подряд (обозначим это число K) и в эти дни его ежедневно посещает N человек. Потом парк закрывается до практически полного его восстановления.

Теперь об ущербе, наносимом посетителями. Ученые-экологи научились измерять ущерб в процентах к исходному состоянию и установили: если ущерб составляет 80%, то такая ситуация становится непоправимой — парк погибает. Кроме того, как вы уже знаете, любая природная система способна к самовосстановлению. Исследования показали, что за ночь восстанавливается примерно 30% нанесенного ущерба. (Конечно, для другого парка и в других условиях обе цифры могут быть другими.)

Итак, пусть за один день посетители нанесли парку некоторый ущерб. Обозначим его величину через A и будем считать, что она зависит только от числа посетителей N и не меняется в зависимости от дня посещения.

Прошел день, и прошла ночь — парк встречает новых посетителей уже с ущербом; уровень его состояния оценивается на 100% – – (100% – 30%)A. Если мы не хотим все время писать знак процентов, то можем уровень состояния выражать просто в долях от 1. Формула уровня состояния парка к началу второго дня получится такая:

$$1-(1-0{,}3)A.$$

Нетрудно понять, что к началу третьего дня уровень состояния будет уже

$$(1-(1-0{,}3)A)^2,$$

к началу четвертого:

$$(1-(1-0{,}3)A)^3$$

и т. д. К началу последнего, K-го дня уровень состояния будет

$$(1-(1-0{,}3)A)^{K-1}.$$

Иными словами, если мы допустим посетителей еще один день, то ущерб станет непоправимым, т. е. уровень состояния, равный $(1-(1-0{,}3)A)^K$, будет ниже чем $1-0{,}8$.

Итак, если нам известно A, то значение K — это наибольший показатель степени в выражении $(1-(1-0{,}3)A)^K$, для которого данное выражение остается больше чем 0,2.

Теперь надо решить задачу, как величина A связана с числом посетителей, т. е. определить A как функцию от N. Для этого надо снова провести исследования. Что ж, в управлении без науки осмысленного шага не сделаешь. Прикидки на глазок да надежда на авось до добра не доводят.

Опять-таки экологи установили, что величина A хорошо описывается функцией $b + c\sqrt{N}$. Вот только b и c для каждой местности свои. Известны также для нашего парка такие показатели ущерба (см. табл. 17):

Таблица 17

Численность посещающих	30	50	100	120	180	250	400	600
Ущерб (в процентах)	7,0	10,8	18,3	20,7	27,0	33,0	43,5	55,0

Нам теперь надо подобрать числа b и c так, чтобы значение A, вычисленное по формуле, не сильно отличалось от значений, приведенных в таблице. Воспользуемся снова электронной таблицей (табл. 18).

Таблица 18

A	B	C	D	E
N	b	c	Ущерб	Отклонение от значения, вычисленного по формуле
30			0,07	| B2 + C2 * √A2 – D2 |
50	Максимум:	max(E2:E9)	0,108	| B2 + C2 * √A3 – D3 |
100			0,183	| B2 + C2 * √A4 – D4 |
120			0,207	| B2 + C2 * √A5 – D5 |
180			0,27	| B2 + C2 * √A6 – D6 |
250			0,33	| B2 + C2 * √A7 – D7 |
400			0,435	| B2 + C2 * √A8 – D8 |
600			0,55	| B2 + C2 * √A9 – D9 |

Во второй клетке второго и третьего столбцов мы будем подбирать значения b и c, стремясь сделать как можно меньше наибольшее из отклонений. Следить за этим максимумом мы будем в третьей клетке столбца C.

После того как будет найдена формула для A, мы сможем находить значение K по N и, наоборот, N по K. Но этим мы займемся на лабораторной работе.

ЗАДАНИЕ

Пусть уже найдены коэффициенты b и c в формуле для A. Придумайте, как по заданному числу N ежедневных посетителей организовать вычисление числа K, показывающего, сколько дней подряд можно посещать парк, и наоборот, как по заданному числу K находить ежедневное количество посетителей парка.

Организация посещений парка

Прежде всего надо определить коэффициенты b и c в формуле для A.

❶ Заполните электронную таблицу так, как указано в объяснительном тексте, и найдите значения b и c.

❷ Теперь электронную таблицу заполните так, чтобы можно было находить K по N и N по K (о том, как нужно заполнить таблицу, вы размышляли, выполняя задание 1 к § 40). Найдите максимально допустимое количество посещающих, если парк будет работать пять дней в неделю. В каком режиме должен работать парк, чтобы его могли посещать 150 человек ежедневно? (В обоих заданиях считается, что двух дней подряд хватает для полного восстановления парка.)

§ 41. Учимся у природы правильной организации управления

В предисловии к этой главе мы определили управление как целенаправленное воздействие на те или иные факторы динамической системы. Название параграфа на первый взгляд выглядит странно: природа в своем существовании не преследует какие-либо цели, поэтому и управлять она никем и ничем не может. Как же тогда учиться управлению у природы?

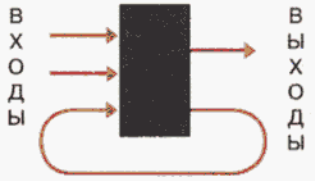
Все это, конечно, так. Но, наблюдая природные явления, мы можем «подсмотреть» у нее те механизмы, которые позволяют природе в течение тысячелетий существовать и развиваться. Ведь и человек обычно стремится к тому, чтобы его деятельность тоже могла обеспечить устойчивое существование и развитие. Это не всегда ему удается. Вот и попробуем разобраться почему.

Вспомните свойство живой природы, которое вы обнаружили, выполняя лабораторную работу № 26, — явление **саморегуляции**: природа стремится восстановить то стабильное состояние, из которого ее вывели тем или иным способом. Это возможно только в том случае, когда само *состояние, в котором находится динамическая система, является фактором, воздействующим на систему.*

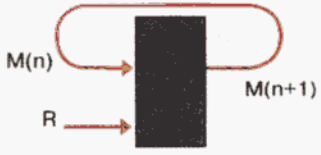
Если представить себе динамическую систему как черный ящик, то в этом случае некоторые его выходы будут параметрами самой системы. Именно так обстоит дело, например, в модели ограниченного или неограниченного роста: выходной параметр $M(n + 1)$ характеризует очередное состояние системы.

Но поскольку это состояние является фактором, воздействующим на систему, его параметры одновременно являются параметрами входов черного ящика. В этом мы снова можем убедиться, рассматривая две упомянутые модели: фактически один и тот же параметр — масса живых организмов — является одновременно и входным и выходным параметром. Их различают только моменты времени, в которых рассматриваются значения этого параметра.

Следовательно, изображая черный ящик, мы можем такие выходы замкнуть на соответствующие входы:



В модели управления добычей возобновляемых ресурсов схема такого черного ящика выглядит следующим образом:

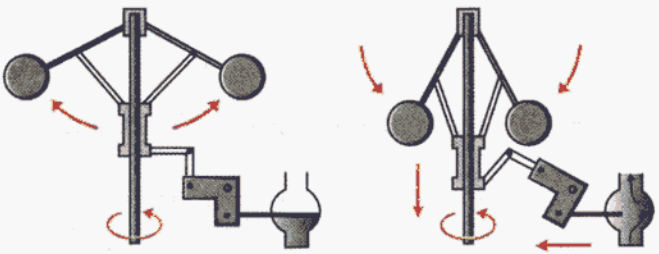


Воздействие выходных параметров динамической системы на ее же входные параметры называют ***обратной связью.***

В живой природе любая динамическая система обязательно обладает механизмом обратной связи. Простейший пример — увеличение теплоотдачи живым организмом при увеличении выделения энергии (например, при интенсивной работе). Этот механизм настолько эффективен, что при здоровом состоянии организма он способен поддерживать постоянную температуру тела при воздействии самых разнообразных внешних и внутренних факторов. Здесь мы имеем дело с так называемой **отрицательной обратной связью**. Эпитет «отрицательная» означает, что она обеспечивает выработку управляющего воздействия, направленного на уменьшение рассогласования между заданным и действительным значениями параметра системы. Возникающий при этом эффект устойчивости системы к внешним воздействиям называют **гомеостазом** системы.

Впрочем, нередко в динамических системах встречается и **положительная обратная связь**, усиливающая возникшее рассогласование. В природе примером положительной обратной связи является реакция пищеварительной системы на слабое раздражение, вызывающее выделение пищеварительных энзимов и тем самым усиление раздражения с еще большим выделением энзимов и т. д. Это явление описывается крылатой фразой «Аппетит приходит во время еды». В школьных курсах физики и химии изучается другой тип явлений с положительной обратной связью — цепные реакции. К ним относится атомный взрыв, взрыв гремучего газа и т. д.

Понятие обратной связи возникло в процессе построения различных автоматических регуляторов. Одним из первых таких регуляторов был регулятор Уатта (см. рисунок), управлявший подачей пара в паровой машине (о нем вам, возможно, рассказывали на уроках физики). Позже понятие обратной связи было выделено как самостоятельное, описывающее саморегулирующиеся системы.



Разумеется, понятие обратной связи вовсе не привязано намертво к понятию черного ящика. Модель объекта может быть достаточно явно структурирована, в частности, в ней могут быть выделены управляющий и управляемый подобъекты. Эти подобъекты связаны информационными каналами: по одним каналам передается управляющая информация от управляющего объекта к управляемо-

му, по другим — информация о результатах управления. Скажем, для паровой машины с регулятором Уатта управляющий объект — сам регулятор, управляемый — паровой котел.

Главным в понятии обратной связи является то, что *изменение состояния объекта непосредственно связано с этим состоянием*. В паровой машине, как только скорость маховика начинает превышать установленные пределы, объем подачи пара уменьшается пропорционально этой скорости, и, наоборот, когда скорость падает ниже установленных пределов, увеличивается объем подачи пара. Отметим, что в этой системе регулятор не «выясняет» причин увеличения или уменьшения скорости (разнообразие этих причин не поддается описанию), он просто поддерживает заданный режим работы паровой машины.

ВОПРОСЫ И ЗАДАНИЯ

1. При каких условиях возможно явление саморегулирования? Приведите примеры саморегулирующихся динамических систем.
2. Что такое обратная связь?
3. Какую обратную связь называют отрицательной; положительной? Приведите примеры динамических систем с отрицательной и положительной обратной связью.
4. Что такое гомеостаз системы?
5. Среди приведенных ниже примеров взаимодействия укажите те, которые относятся к понятию обратной связи. Для каждого указанного вами случая укажите тип обратной связи (положительна она или отрицательна):
   а) изменение диаметра кровеносных сосудов организма и теплоотдачи организма;
   б) раскрытие (закрытие) зонтика и начало (прекращение) дождя;
   в) изменение рентабельности производства и объема выпускаемой продукции;
   г) изменение покупательского спроса и цены на товар;
   д) изменение яркости горения лампочки и подаваемого напряжения.

§ 42. Изучаем системы с обратной связью

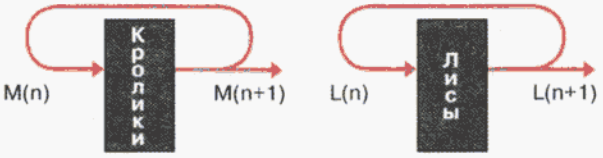
Читая предыдущий параграф, вы уже, наверно, поняли, что системами с обратной связью мы занимались еще в третьей главе, изучая модели ограниченного и неограниченного роста. Но вы об этом тогда не догадывались. Да это было и не нужно — в то время вы осваивали понятия динамической системы и черного ящика. Теперь же мы обсудим именно действие обратной связи. Начнем, как обычно, с разбора задачи.

На некотором острове живут лисы и кролики. Кролики питаются травой, которой вдоволь на острове, а лисы охотятся на кроликов. Экологи время от времени пересчитывают кроликов и лис. Вот что они установили:

- коэффициент прироста числа кроликов зависит от колебаний погоды (холодная или теплая зима, сухое или влажное лето, ранняя или поздняя весна и т. д.) и колеблется в пределах от 3,2 до 4,7;
- коэффициент прироста числа лис при избытке крольчатины колеблется от 5,2 до 5,7; при недостатке кроликов он пропорционален приросту кроликов; коэффициент пропорциональности примерно равен $\frac{1}{c}$, где c — масса крольчатины, в среднем съедаемой одной лисой за год; величину c можно принять равной 50.

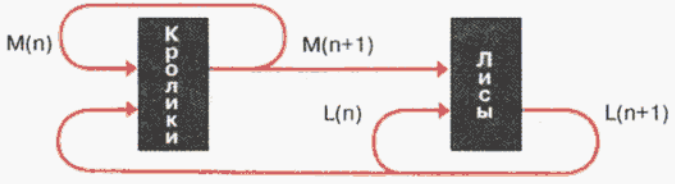
Требуется установить, как меняется численность кроликов и лис с течением времени.

Решение задачи, как обычно, начинаем с построения модели. Про собственную жизнь кроликов и лис мы ничего не знаем, поэтому можем считать, что нам даны два черных ящика. Один ящик мы назовем «Кролики», другой — «Лисы».



Здесь $M(n)$ — масса кроликов через n лет, а $L(n)$ — масса лис в тот же момент времени.

Но эти ящики существуют не раздельно, каждый сам по себе, а взаимодействуют друг с другом. Это взаимодействие можно изобразить такой схемой:



В этой динамической системе легко усматриваются три контура обратных связей. Опишем их расчетными формулами.

Сначала займемся кроликами. Поскольку корма у них вдоволь, то их прирост идет, по-видимому, в соответствии с моделью неограниченного роста. Вот только лисы их поедают. Значит, изменение массы кроликов за год можно записать так:

$$M(n+1) - M(n) = kM(n) - cL(n).$$

Здесь k — коэффициент прироста кроликов за год, а c — масса крольчатины, съедаемая одной лисой за тот же год. Конечно, аппе-

тит у разных лис может оказаться разным, поэтому за c мы примем некоторое среднее значение, характеризующее прожорливость лис в целом.

То же соотношение можно переписать так:

$$M(n+1) = (1+k)M(n) - cL(n).$$

А как обстоят дела у лисиц? Ясно, что если кроликов очень много, то численность лис, как и численность кроликов, растет по *модели неограниченного роста* с соответствующим коэффициентом; обозначим его буквой a. Если же крольчатины едва хватает, чтобы прокормиться уже живущим, тут не до прироста. В этом случае прирост определяется величиной $\frac{M(n+1) - M(n)}{c}$, показывающей, сколько «новых» лисиц может прокормиться за счет прироста кроликов. Значит, для лис можно написать такую формулу:

$$L(n+1) - L(n) = (\min(a, \frac{M(n+1) - M(n)}{c})) \cdot L(n).$$

Перепишем эту формулу в виде, удобном для вычисления $L(n+1)$:

$$L(n+1) = (1+\min(a, \frac{M(n+1) - M(n)}{c})) \cdot L(n).$$

Выбрав теперь некоторый год в качестве начального, т. е. считая $n = 0$, мы, зная массу кроликов $M(0)$ и массу лис $L(0)$, можем последовательно вычислить $M(1)$ и $L(1)$, $M(2)$ и $L(2)$ и т. д.

Модель построена, пора приступать к компьютерному эксперименту с этой моделью.

ВОПРОСЫ И ЗАДАНИЯ

1. Как сказано в объяснительном тексте, в рассмотренной модели «Лисы и кролики» имеется три контура обратных связей. Укажите их. Какие из этих связей являются положительными, а какие отрицательными?
2. а) Предположим, что выполняется пропорция

$$\frac{L(0)}{M(0)} = \frac{k}{c}.$$

Какие значения будут иметь $M(1)$ и $L(1)$, $M(2)$ и $L(2)$, ... , $M(n)$ и $L(n)$?
б) Пусть имеет место пропорция, записанная в пункте а). В этом случае говорят, что значения $M = M(n)$ и $L = L(n)$ характеризуют положение **экологического равновесия** в данной биологической системе. Как вы думаете, почему используется такое название?
3. Какую роль играют положительные обратные связи в эволюции биологических систем?

Лисы и кролики

Подготовим заполнение электронной таблицы для проведения запланированного нами компьютерного эксперимента (см. табл. 19).

Таблица 19

A	B	C	D	E	F
a	Год	Масса кроликов	Масса лис		
<*a*>	0	<*M*(0)>	<*L*(0)>		
k	B2+1	(1+A4)*C2–A6*D2	(1+min(A2, (C3–C2)/A6)*D2		
<*k*>	B3+1	(1+A4)*C3–A6*D3	(1+min(A2, (C4–C3)/A6)*D3		
c	...	...	...		
<*c*>	...	...	...		
	...	...	...		
	...	...	...		
	...	...	...		
	B9+1	(1+A4)*C9–A6*D9	(1+A2*(C10–C9))*D9		

Как обычно, в угловых скобках записано обозначение величины, которая должна быть занесена в данную клетку в качестве начальных данных. Строки, выделенные цветом, получаются копированием блока B3:D3.

❶ Заполните электронную таблицу указанным образом. Введите следующие исходные данные (табл. 20).

Таблица 20

a	*k*	*c*	*M*(0)	*L*(0)
0,1	4,0	50	10 000	100

Поручите компьютеру построить графики *M*(*n*) и *L*(*n*) как функций от *n*.

§ 43. Управление по принципу обратной связи

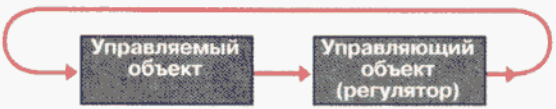
Сравним функционирование двух систем — экологической системы «Лисы и кролики» и паровой машины с регулятором Уатта (табл. 21).

Таблица 21

Система	Изменение в системе		Реакция системы на изменение
Лисы и кролики	Увеличение (или уменьшение) численности кроликов	→	Увеличение (соответственно уменьшение) численности лис
Паровая машина с регулятором Уатта	Увеличение (или уменьшение) скорости	→	Увеличение (соответственно уменьшение) сброса пара в котле

Сходство налицо, не правда ли? Можно сказать, что популяция лис выступает регулятором численности кроликов.

Конечно, в природной системе и речи идти не может ни о какой целенаправленности воздействия. Но природные системы подсказывают нам, что и при создании управляемой системы ее устойчивость обеспечивается наличием обратной связи между управляемым и управляющим (регулирующим) объектами системы. Управление, использующее обратную связь между управляемым и управляющим объектами, называется **управлением по принципу обратной связи**. Если управляемый и управляющий объект-регулятор представить в виде черных ящиков, то мы можем изобразить такую систему следующей схемой:



Управление по принципу обратной связи

Сравним, к примеру, управление по принципу обратной связи с работой светофора, который регулирует движение на перекрестке. Здесь отсутствует всякая обратная связь, ибо каждый сигнал горит заранее заданное время вне зависимости от того, имеются ли машины, движущиеся в данном направлении. Вот и получается нередко, что на зеленый свет ехать некому, хотя в то же время на красном стоит вереница машин. К сожалению, в нашей социальной жизни (в том числе экономической), как правило, имеет место уп-

равление именно такого типа. Не случайно большинству людей управление представляется как грубое насилие, выражающееся в указах, приказах, законах, постановлениях. Подобное управление стремится предусмотреть и регламентировать все возможные случаи поведения управляемого объекта, и если вдруг обнаруживается лазейка для обхода того или иного постановления, то проводится расследование причин и принимаются меры по дальнейшему недопущению. Но лазейки-то всегда есть и будут... Напомним, что главное преимущество управления по принципу обратной связи состоит в том, что регулятору не требуется выяснять причины и обстоятельства, приведшие к изменению состояния управляемого объекта, он просто реагирует на это изменение.

В экономической сфере принцип обратной связи проявляется в действии рыночных законов. Скажем, повышение спроса приводит к повышению цены, повышение цены делает выгодным увеличение производства данного товара, увеличение производства приводит к снижению спроса, снижение спроса влечет падение цены и т. д. Иными словами, мы имеем следующую схему экономического взаимодействия:

Конечно, эта модель весьма упрощена. Здесь, к примеру, не отражено действие такого фактора, как покупательная способность населения, и многое другое. Но мы обращаем ваше внимание именно на наличие обратной связи уже в данной, совсем простой модели экономического управления.

Проведенный вами компьютерный эксперимент с моделью «Лисы и кролики» демонстрирует еще один важный момент в функционировании динамической системы. Вспомните: если численность лисиц намного отличалась от той, которая необходима для экологического равновесия, система не могла прийти к этому равновесию и разрушалась. Значит,

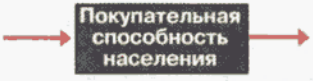
Для каждой динамической системы существуют определенные границы устойчивости этой системы, в которых она сохраняет свою целостность; выход за эти границы приводит к разрушению системы.

Именно выходом за границы устойчивости объясняются экологические катастрофы и экономические кризисы. И чтобы система была восстановлена, необходимо вернуть ее в область саморегулирования. А для этого важно знать, каковы же границы этой области. Находить их как раз и помогает компьютерное моделирование задач управления.

ВОПРОСЫ И ЗАДАНИЯ

1. Что такое управление по принципу обратной связи?
2. В чем причины разрушения систем, способных к саморегулированию?
3. Приведите примеры управления по принципу обратной связи.
4. В объяснительном тексте параграфа приведена следующая схема экономического взаимодействия:

Введем еще один черный ящик:



Как бы вы встроили его в указанную схему? Перечислите возникающие при этом контуры обратных связей и для каждой из обратных связей определите, положительна она или отрицательна.

5. В § 40 рассматривалась задача об организации посещения парка. Предложенное для этой задачи решение не предусматривает обратной связи в управлении посещениями парка. Предложите управление, регулирующее посещение парка и основанное на принципе обратной связи. Совет: при построении управления используйте экономические факторы.
6. Практически все государства мира законодательно борются с мздоимством (дачей и получением взяток) своих чиновников. Эта борьба иногда бывает более успешной, зачастую менее, но не может искоренить зла, поскольку опирается только на страх перед разоблачением и последующим наказанием. Предложите механизм борьбы со взяточничеством, основанный на принципе обратной связи.

§ 44. Глобальные модели

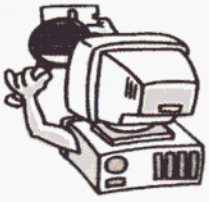
Конечно, в предыдущих параграфах мы рассмотрели весьма простые модели управления. Относятся они к малым территориям или весьма малым совокупностям объектов. Это так называемые локальные модели. В реальном управлении используются намного более сложные модели, учитывающие многочисленные и разнородные факторы. Но принципы моделирования и управления остаются общими что для больших моделей, что для маленьких. А главное, что роль маленьких моделей совсем не маленькая. Ведь мы можем соединять между собой черные ящики, словно элементы конструктора, и получать сложные модели, описывающие большие пространства и многофакторное воздействие. Такие модели, описывающие многофакторные процессы, протекающие на территориях больших

регионов, стран или даже всего земного шара, получили название **глобальных моделей**. В 70-е годы в Вычислительном центре Академии наук СССР под руководством академика Н. Н. Моисеева была начата разработка первой глобальной модели биосферы Земли, способной показать, что будет происходить с Землей в результате тех или иных воздействий на нее человечества. Именно на этой модели были предсказаны последствия ядерной войны, в которой не может быть победителя, ибо последствия ее будут необратимы.

Другие глобальные модели, построенные в 80-е годы XX в., показали, что безудержный рост потребления человечеством даже возобновляемых ресурсов тоже приведет к краху. Нужна продуманная, экологически выверенная стратегия управления на всех уровнях — от района до государства и сообщества государств, чтобы человечество осталось на Земле. Такие стратегии есть, и находить их помогают информационные компьютерные технологии.

ВОПРОСЫ И ЗАДАНИЯ

1. Что такое глобальные модели?
2. В чем вы видите роль информатики и информационных технологий в жизни современного общества?



6 Принципы работы вычислительной техники

В предыдущих главах вы познакомились с тем, как применять компьютер для решения различных задач. Знакомы вы также с его основными компонентами и внешними устройствами, обеспечивающими взаимодействие человека и компьютера. Но осталось «за кадром» то, *как* работает компьютер. Вот об этом и пойдет речь в данной главе.

Прежде всего надо вспомнить, что вся информация, циркулирующая внутри компьютера, закодирована двумя символами (см. § 3); можно считать, что эти символы 0 и 1. Ясно, что могут существовать самые разные способы кодирования информации. О некоторых из них (например, ASCII- и RGB-кодировании) мы рассказывали в главе 1. И это далеко не все варианты кодирования информации, которые используются в компьютерной технике. Еще об одном способе кодирования *числовой* информации, правда, известном человечеству задолго до изобретения компьютеров, мы начнем рассказывать в ближайшем параграфе.

§ 45. Системы счисления

Система счисления — это способ записи чисел с помощью заданного набора специальных знаков, называемых цифрами. Еще в пятом классе вы узнали, что привычная нам система счисления — **позиционная** и **десятичная**. Это значит, что для записи любых чисел используется *десять* цифр (обычно 0, 1, 2, 3, 4, 5, 6, 7, 8, 9), а значение каждой цифры (ее «вклад» в обозначаемое число) определяется той *позицией*, которую цифра занимает в записи числа. Так, запись 23 означает, что это число составлено из 3 единиц и 2 десятков. Если мы поменяем позиции цифр, то получим совсем другое число — 32. Это число содержит 3 десятка и 2 единицы. «Вклад» двойки уменьшился в 10 раз, а «вклад» тройки в 10 раз возрос. Легко понять, что цифры десятичной записи числа — это просто коэффициенты его представления в виде суммы степеней

числа 10, которое называют **основанием** данной системы счисления. Например:

$$38\,054 = 3 \cdot 10^4 + 8 \cdot 10^3 + 0 \cdot 10^2 + 5 \cdot 10^1 + 4 \cdot 10^0.$$

На самом деле числа можно записывать как сумму степеней не только числа 10, но и любого другого натурального числа b, большего 1. Такую *запись называют представлением данного числа в системе счисления с основанием b*. Скажем, в Древнем Вавилоне использовалась система счисления с основанием 60. Делением часа на 60 минут, а минуты на 60 секунд мы обязаны именно этой системе счисления. Тот факт, что основанием используемой нами системы счисления является число 10, объясняется, вероятно, тем, что природа наделила нас десятью пальцами на руках.

Главное же удобство позиционной нумерации состоит в том, что действия над числами в такой системе счисления выполняются *поразрядно* (вспомните, как вы складываете и умножаете, вычитаете и делите многозначные числа в десятичной системе). Все, что нужно знать, — это таблицы сложения и умножения для однозначных чисел. И конечно, правила выполнения действий столбиком. Теперь, когда вы знаете слово «алгоритм», каждый сообразит, что правила выполнения действий столбиком — это просто алгоритмы для любого исполнителя, который умеет проделывать соответствующие действия над однозначными числами.

Самая простая из всех позиционных систем счисления, конечно, **двоичная** — ведь в ней всего две цифры: 0 и 1. И значит, имеется только два однозначных числа. Поэтому в этой системе счисления очень просто выглядят таблицы сложения и умножения:

$$\begin{array}{ll} 0 + 0 = 0 & 0 \cdot 0 = 0 \\ 1 + 0 = 1 & 1 \cdot 0 = 0 \\ 0 + 1 = 1 & 0 \cdot 1 = 0 \\ 1 + 1 = 10 & 1 \cdot 1 = 1 \end{array}$$

В этих равенствах лишь результат 10 выглядит удивительно. Но стоит только заметить, что в двоичной системе 10 — это $1 \cdot 2^1 + 0 \cdot 2^0$, и все становится на свои места.

Вспомните-ка, сколько времени ушло у вас в начальной школе, чтобы выучить таблицы сложения и умножения для однозначных чисел в столь привычной десятичной системе. А в двоичной системе, кроме отмеченного выше «удивительного» равенства, и учить-то нечего!

За простоту действий над числами в двоичной системе приходится расплачиваться тем, что запись даже совсем небольших чисел в ней оказывается весьма длинной. Приведем таблицу, содержащую первые 16 натуральных чисел и их представление в двоичной системе счисления (табл. 22).

Таблица 22

Десятичная система	Двоичная система
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000

ВОПРОСЫ И ЗАДАНИЯ

1. Что такое система счисления? Что называют основанием позиционной системы счисления?
2. Что называют представлением числа в позиционной системе счисления с данным основанием?
3. Сколько цифр используется в позиционной системе счисления с основанием, равным: а) 2; б) 8; в) 10; г) 16; д) 256; е) b?
4. Существует ли позиционная система счисления, в которой для записи чисел используется ровно одна цифра?
5. Исполнитель умеет сравнивать в некоторой позиционной системе счисления однозначные числа. Составьте для этого исполнителя алгоритм сравнения:
   а) двух двузначных чисел;
   б) двух n-значных чисел (n — заданное число);
   в) двух чисел, имеющих в своей записи одинаковое количество цифр.
6. Сравните между собой числа, записанные в двоичной системе счисления:
   а) 10101 и 101010;
   б) 1101 и 1001;
   в) 111001 и 110111.
   Для пары чисел из пункта в) определите, на сколько одно число больше другого.
7. Пользуясь таблицами сложения и умножения в двоичной системе, сложите и перемножьте следующие числа: 101 и 1101. С помощью таблицы 22 определите, какими будут результаты этих действий в десятичной системе.

8. а) К чему приводит умножение на 2 числа, записанного в двоичной системе?
б) Пользуясь правилом, выведенным в пункте а), и таблицей 22, запишите в двоичной системе следующие числа: 32, 64, 128, 20, 30.
в) Пользуясь равенствами $31=16+15$, $52=32+20$, $75=64+11$ и результатами из пункта б), запишите в двоичной системе числа 31, 52, 75.

§ 46. Как в компьютере реализуются вычисления

В самом названии «компьютер», происходящем от английского computer, т. е. вычислитель, заложено первоначальное предназначение этого агрегата — он изобретался для того, чтобы облегчить людям вычисления. Много позже, почти 30 лет спустя, когда мощность компьютеров возросла в тысячи раз, было осознано, что они представляют собой универсальное средство обработки информации. Мы говорили об этом в двух первых главах, а сейчас вернемся к истокам и расскажем, как компьютер выполняет действия над числами. Собственно говоря, действия над числами — это тоже обработка информации: по двум данным числам и указанному действию выдается новое число, являющееся результатом этого действия над заданными числами. Разобравшись, как производятся компьютером вычисления, нетрудно будет понять, как им обрабатываются данные любого вида.

Прежде всего отметим, что в электронных вычислительных машинах — будь то компьютер или его младший родственник калькулятор — используется двоичная система счисления. Выбор двоичной системы объясняется тем, что имеющиеся в этой системе всего две цифры легко «зашифровать» при помощи каких-нибудь технических средств, например электрического тока или светового луча. Цифра 0 двоичной системы счисления может означать, что ток (луч) не проходит, а цифра 1 — что ток (луч) проходит. При таком представлении цифр действия над числами производятся подходящими комбинациями включений и выключений тока или света. Поэтому любую электронную вычислительную машину можно представить себе как совокупность соединенных между собой выключателей тока (или света). Отличие электронного выключателя от выключателя настольной лампы состоит в том, что в электронном выключателе нет механических движущихся частей и переключается он не рукой человека, а электрическим сигналом от другого выключателя. Время переключения поэтому оказывается очень малым, порядка 10^{-9} с.

Самый простой прибор, осуществляющий такую операцию, вы видите на рисунке 37. Цепь разомкнута до тех пор, пока на обмотку железного сердечника не подано напряжение. В этот момент в

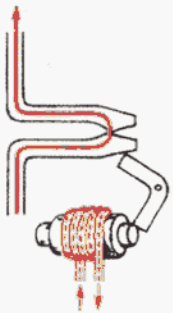


Рис. 37. Электромагнитное реле

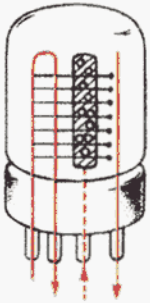


Рис. 38. Электронная лампа

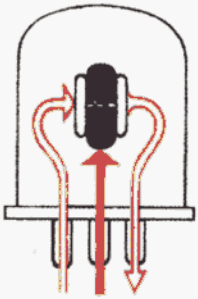


Рис. 39. Полупроводниковый триод

сердечнике создается магнитное поле, притягивающее один конец вращающегося на шарнире рычажка. Другой его конец в этот момент сжимает контакты — цепь замыкается. На принципах такого переключения работал один из первых компьютеров — МАРК-1.

В дальнейшем в качестве переключателей стали использовать электронные лампы-триоды (рис. 38). Их действие вы, вероятно, изучали на уроках физики.

Затем настала пора плоскостных транзисторов (рис. 39). Мы не будем вдаваться в теорию полупроводников, оставляя и это урокам физики.

И наконец, планарные транзисторы (рис. 40). Они идентичны по своему действию плоскостным, но не превышают в длину сотой доли сантиметра. Современная технология изготовления этих приборов позволяет размещать на поверхности одной микросхемы несколько миллионов транзисторов.

Как видите, при неизменном принципе — получить переключатель — шла борьба за размеры и скорость срабатывания.

Важно, однако, не только иметь миниатюрные электронные переключатели, но и знать, как их соединить между собой, чтобы с их помощью выполнять арифметические действия.

Если в вашем распоряжении оказалось два переключателя, то есть ровно два способа соединить их между собой. Первый вариант соединения называется, как вы знаете из физики, *последовательным* (рис. 41, *а*), а второй — *параллельным* (рис. 41, *б*). Ясно, что в первом случае ток в цепи идет (лампочка горит) только тогда, когда включены оба переключателя. Во втором случае для прохождения тока в цепи (лампоч-

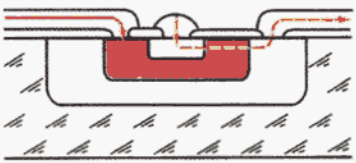


Рис. 40. Планарный транзистор

Рис. 41. Последовательное (*а*) и параллельное (*б*) соединение переключателей

ка горит) достаточно, чтобы включен был хотя бы один переключатель. Кроме того, рассматривают еще один вариант: лампочка горит тогда и только тогда, когда переключатель выключен (рис. 42).

Рис. 42.

Такие конструкции управления называются **вентилями**. Можно, конечно, условиться называть их вентилем первого типа, вентилем второго типа... Но столь обезличенные названия не отражают специфику их работы.

Поскольку первый вентиль зажигает лампочку только тогда, когда замкнут *и* первый переключатель, *и* второй, его назвали вентилем И. Второй вентиль зажигает лампочку, когда замкнут *или* первый переключатель, *или* второй — его назвали вентилем ИЛИ. Третий вентиль зажигает лампочку тогда и только тогда, когда переключатель *не* замкнут, — он называется вентилем НЕ. Будем условно изображать вентили следующим образом (рис. 43, *а* — *в*):

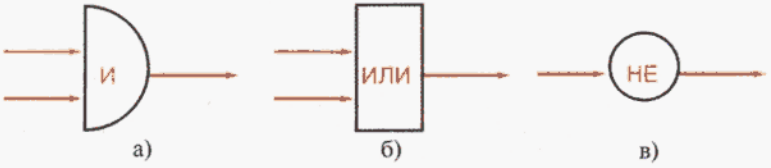


Рис. 43. Условное изображение вентилей И, ИЛИ, НЕ

В дальнейшем мы не будем писать их названия — геометрическая форма однозначно указывает, какой это вентиль.

А теперь из вентилей соберем схему, указанную на рисунке 44. Нетрудно проверить (это вы сделаете, выполняя задание 4), что, подавая на входы *x* и *y* сигналы 0 или 1, мы на выходах получаем два сигнала, которые поразрядно кодируют сумму двух однозначных чисел. А поскольку действия над числами, записанными в позиционной системе, выполняются поразрядно, то ясно, что аналогичным образом можно построить электронные схемы для сложения многозначных чисел, представленных в двоичной системе счисления.

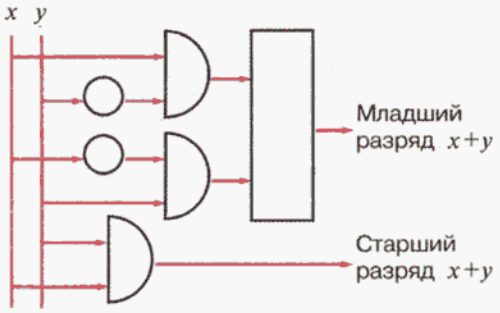


Рис. 44. Сумматор для двух одноразрядных двоичных чисел

Более того, математики доказали, что для любой функции от любого числа двоичных переменных можно создать подходящую электронную схему из вентилей И, ИЛИ, НЕ, вычисляющую значение этой функции.

ВОПРОСЫ И ЗАДАНИЯ

1. Почему в электронной вычислительной технике обычно используется двухсимвольное кодирование?
2. а) Обозначим в вентиле И (см. рис. 43, *а*) один вход буквой *x*, другой — буквой *y*, а выход — буквой *z*. Заполните таблицу, показывающую, как значение *z* зависит от значений *x* и *y*.

x	*y*	*z*
0	0	
1	0	
0	1	
1	1	

б) Выполните такое же задание для вентилей ИЛИ и НЕ.

в) Как, по-вашему, выглядит «двоичный мультипликатор» — устройство для перемножения двух однозначных двоичных чисел?

3. Для схемы, изображенной на рисунке 45, составьте таблицу, показывающую зависимость значения *z* от значений *x* и *y*.

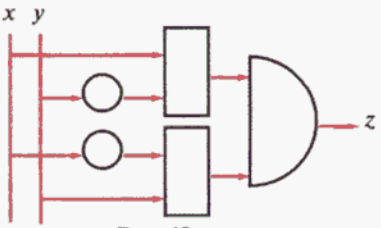


Рис. 45

4. Для двоичного сумматора (см. рис. 44) заполните таблицу по следующему образцу:

x	y	Старший разряд	Младший разряд
0	0		
1	0		
0	1		
1	1		

Убедитесь, что двоичный сумматор в качестве результата выдает сумму однозначных чисел x и y.

5* а) Постройте схему сумматора, вычисляющего сумму трех однозначных чисел в двоичной системе.

б) При сложении многозначных чисел в старших разрядах приходится учитывать появление так называемой «единицы переноса». Это означает, что в этих разрядах складываются не два однозначных числа, а три. Используя результат пункта а), постройте схему сумматора для сложения двух двузначных чисел в двоичной системе счисления.

в) Постройте схему «мультипликатора» для умножения двух двузначных двоичных чисел.

§ 47. Двоичная и шестнадцатеричная системы счисления

Внимательное чтение § 45 могло навести вас на грустные мысли: мы привыкли обращаться с числами, записанными в десятичной нумерации, а компьютер имеет дело лишь с двоичными числами. Что же теперь, нужно обзаводиться словарем, в котором содержатся переводы чисел из одной системы счисления в другую? Прямо как при общении с иностранцем на незнакомом языке!

Правда, в заданиях к этому параграфу вы могли подметить некоторые приемы превращения десятичных чисел в двоичные, но вряд ли это удовлетворит того, кто хотел бы иметь простой и надежный способ перевода чисел из одной системы счисления в другую. Вот о таких способах и пойдет речь.

Мы начнем с изложения алгоритма перевода чисел из десятичной системы счисления в систему счисления с основанием b. Что значит записать число в системе счисления с основанием b? Если вы в свое время ответили на вопрос 2 к § 45, то вам ясно, что запись $a_1a_2...a_{n-1}a_n$ является представлением числа c в системе счисления с основанием b, если

$$c = a_1 \cdot b^{n-1} + a_2 \cdot b^{n-2} + \ldots + a_{n-1} \cdot b^1 + a_n \cdot b^0,$$

причем каждый коэффициент a_i неотрицателен и меньше b.

Из этого равенства видно, что последняя цифра a_n представляет собой остаток при делении числа c на b. А частное от такого деления используется для нахождения предпоследней цифры a_{n-1} — она получается как остаток при делении этого частного снова на b и т. д.

Приведем пример перевода десятичного числа 793 в шестеричную систему счисления:

```
793 | 6
6   132 | 6
19  12  22 | 6
18  12  18 | 3
 13 12  4
 12  0
  1
```

Результат: 3401.

Как видите, чтобы перевести число в b-ичную систему счисления, нужно выполнить несколько раз деление столбиком на b и затем прочитать получившиеся остатки справа налево.

Конечно, указанный алгоритм годится и для перевода чисел из b-ичной системы счисления в десятичную. Однако деление в этом случае надо было бы производить в b-ичной системе. Но ведь деление уголком — это фактически последовательное выполнение операций умножения и вычитания, и, значит, пришлось бы выучить таблицы сложения и умножения в системе счисления с любым основанием. Кто же на такое решится!

Поэтому мы рассмотрим алгоритм перевода числа из b-ичной системы, использующий действия в той системе счисления, куда мы собираемся это число переводить. Если, например, мы переводим число в десятичную систему, то и действия будут выполняться в обычной десятичной системе. Этот алгоритм заключается в следующем. Записываем в одной строке число, которое необходимо перевести, а строкой ниже будем получать число в нужной нам системе счисления. Для этого первую цифру перепишем без изменения, а под каждой следующей цифрой будем писать число, полученное сложением этой цифры с произведением слева стоящего числа на основание системы счисления.

Ниже показано исполнение этого алгоритма для перевода двоичного числа 100111011 в десятичную запись:

1	0	0	1	1	1	0	1	1
1	2	4	9	19	39	78	157	315

Результат: 315.

Как видите, эти вычисления легко проделать и устно. Называется этот алгоритм **схемой Горнера**.

Двоичное представление чисел (и информации вообще), столь удобное для электронной техники, вряд ли можно признать удоб-

ным для человека. У любого программиста — специалиста, который разрабатывает программное обеспечение для компьютеров, — зарябит в глазах от нулей и единиц, если он попытается прочитать программу прямо в машинном коде. А такая потребность иногда возникает. Поэтому программисты решили слегка (в четыре раза) сократить количество знаков в записи машинных чисел, использовав для этого **шестнадцатеричную** систему счисления.

Прежде всего заметим, что для записи чисел в шестнадцатеричной системе счисления требуется 16 цифр. Первые десять цифр в ней используются те же, что и в десятичной системе счисления, а дальше обычно пользуются латинскими буквами. Как именно, показано в таблице 23.

Таблица 23

Десятичная система	Двоичная система	Шестнадцатеричная система
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Фактически это таблица 22, дополненная еще одним столбцом.

Если взять уже встречавшееся нам число 315, то в шестнадцатеричной системе оно запишется как 13B. Убедиться в этом можно с помощью алгоритма, использующего последовательное деление на 16 с остатком. Но ту же запись 13B можно получить и по-другому: двоичное число 100111011 разбивается на четверки справа налево и каждая такая четверка заменяется шестнадцатеричной цифрой согласно выше написанной таблице:

$$\underset{1}{\underbrace{1}}\ \underset{3}{\underbrace{0011}}\ \underset{B}{\underbrace{1011}}$$

Правда, в самой левой четверке оказалась только одна цифра, значит, к ней надо мысленно слева приписать недостающие нули. Такие четверки называются *тетрадами*.

Разумеется, это правило перевода чисел из двоичной системы в шестнадцатеричную справедливо для любого числа, а не только для рассмотренного нами. Легко выполнять и обратный перевод — из шестнадцатеричной системы в двоичную: для этого каждую цифру надо расписать в тетраду в соответствии все с той же таблицей.

Значит, шестнадцатеричная система по сути своей — это та же двоичная, но в ней каждая группа из четырех цифр заменяется всего одним символом. Такое превращение двоичного кода в шестнадцатеричный легко выполнить в уме, чего никак не скажешь о переводе в десятичную систему и обратно. Понятна теперь любовь программистов к шестнадцатеричной системе (хотя у них, как и у всех других людей, на руках десять пальцев).

ВОПРОСЫ И ЗАДАНИЯ

1. а) Как переходить от записи числа в шестнадцатеричной системе к записи в двоичной и обратно?
   б)* Придумайте легкий (т. е. устно выполнимый) алгоритм перевода числа из восьмеричной системы в двоичную и обратно. Какой частью таблицы 23 удобно для этого пользоваться?
2. Переведите из десятичной в двоичную и шестнадцатеричную системы следующие числа: 19, 44, 129, 561, 1322.
3. Переведите из двоичной в десятичную систему следующие числа: 1001, 10101, 111001, 10111101.
4. Переведите из шестнадцатеричной в десятичную систему следующие числа: 25, 4F, 1A7, ABC, D1AE, FFFF.
5. Предположим, что компьютер, работающий в двоичной системе счисления, оперирует с семизначными числами. Какое максимальное число он воспринимает?
   Ответьте на тот же вопрос в случае 11-разрядного и 15-разрядного компьютеров.

6*. Попробуйте объяснить:
   а) почему схема Горнера дает перевод из одной системы счисления в другую;
   б) почему алгоритм последовательного деления с остатком дает перевод из одной системы счисления в другую;
   в) алгоритм перевода из шестнадцатеричной системы в двоичную и обратно.

§ 48. Представление информации в компьютере. Принцип программного управления

Вы уже хорошо усвоили, что вся информация, вводимая в компьютер и возникающая в ходе его работы, хранится в его памяти. Как устроена память компьютера? Ее можно представлять себе как длинную страницу, состоящую из отдельных строк. Каждая такая строка называется **ячейкой памяти** и, в свою очередь, разделяется на разряды. Содержимым любого разряда может быть либо 0, либо 1.

Так что в любую ячейку памяти записан некоторый набор нулей и единиц — **машинное слово**.

Все ячейки памяти пронумерованы. Номер ячейки называют ее **адресом**.

Наличие у каждой ячейки адреса позволяет отличать ячейки друг от друга, обращаться к любой ячейке, чтобы записать в нее новую информацию или извлечь ту информацию, которая в ней хранится. Нужно помнить, что при считывании хранящегося в ячейке машинного слова содержимое ячейки не изменяется. А вот при записи в ячейку информации прежнее содержимое ячейки безвозвратно исчезает, как говорят, стирается.

Это обстоятельство, вероятно, напомнило вам выполнение присваивания нового значения переменной. Сходство здесь далеко не внешнее. На каждую переменную в памяти компьютера отводится определенное количество ячеек (в зависимости от объявленного типа переменной); присваивание нового значения переменной — это запись новой информации в зарезервированные ячейки.

Все компьютеры работают в принципе одинаково. Для наглядности мы будем рассматривать совсем маленький виртуальный компьютер, который назовем «Малютка». В его памяти 256 ячеек, в каждой из которых 12 разрядов. Когда бы вы ни заглянули в память компьютера, в ней хранятся наборы нулей и единиц. Однако анализировать двенадцатизначные двоичные числа не так уж легко, поэтому на экране вашего компьютера они отображаются в шестнадцатеричном коде. Сокращение длины записи данных в ячейках позволяет высветить всю память «Малютки» на вашем экране. Адреса ячеек тоже записываются в шестнадцатеричном коде — от 00 до FF.

На рисунке 46 показано, как выглядит память «Малютки» в один из моментов ее работы (слева от ячейки красным цветом напечатан ее адрес, черным — содержимое).

Память ЭВМ. Страница 0

00	004	10	000	20	000	30	000	40	000	50	000	60	000	70	000
01	003	11	000	21	000	31	000	41	000	51	000	61	000	71	000
02	007	12	000	22	000	32	000	42	000	52	000	62	000	72	000
03	00C	13	000	23	000	33	000	43	000	53	000	63	000	73	000
04	001	14	000	24	000	34	000	44	000	54	000	64	000	74	000
05	B02	15	000	25	000	35	000	45	000	55	000	65	000	75	000
06	B03	16	000	26	000	36	000	46	000	56	000	66	000	76	000
07	C00	17	000	27	000	37	000	47	000	57	000	67	000	77	000
08	B01	18	000	28	000	38	000	48	000	58	000	68	000	78	000
09	C00	19	000	29	000	39	000	49	000	59	000	69	000	79	000
0A	F00	1A	000	2A	000	3A	000	4A	000	5A	000	6A	000	7A	000
0B	000	1B	000	2B	000	3B	000	4B	000	5B	000	6B	000	7B	000
0C	000	1C	000	2C	000	3C	000	4C	000	5C	000	6C	000	7C	000
0D	000	1D	000	2D	000	3D	000	4D	000	5D	000	6D	000	7D	000
0E	000	1E	000	2E	000	3E	000	4E	000	5E	000	6E	000	7E	000
0F	000	1F	000	2F	000	3F	000	4F	000	5F	000	6F	000	7F	000

Рис. 46. Пример заполнения оперативной памяти компьютера «Малютка»

Посмотрев на последний адрес — 7F, легко понять, что на экране отображена только первая половина всех ячеек памяти. Вторая половина памяти — это страница 1.

Каждая ячейка способна хранить одиннадцатиразрядное целое двоичное число. Двенадцатый (самый левый) разряд ячейки отведен для знака числа: знак «+» кодируется символом 0, а знак «–» кодируется символом 1. Приведем примеры представления целых чисел в «Малютке»:

Число	Машинное слово	Свернутый вид на экране
1	000000000001	001
45	000000101101	02D
–2	100000000010	802
–19	100000010011	813

Создатели любого компьютера наделяют его умением выполнять ряд элементарных операций — сложение, умножение и т. п. Но и обыкновенные калькуляторы тоже способны выполнять эти операции. Чем же тогда отличается компьютер от калькулятора? Вы, конечно, сразу ответите на этот вопрос: главное отличие заключается в том, что компьютер может выполнить без участия человека не только одну команду, но и длинную заранее заданную последовательность команд — программу. Более того, включенный компьютер и не работает иначе, нежели выполняя какую-либо программу. В этом состоит один из основных принципов работы компьютера — **принцип программного управления**.

Каждая команда тоже кодируется двоичной последовательностью и размещается, как и число, в памяти компьютера. Этим реализуется так называемый **принцип хранимой программы**. Оперативная память тем самым выполняет роль объединенного буфера как для числовой информации, так и для команд. Более того, «Малютке» (как, впрочем, и другим компьютерам) совершенно безразлично, что у нее в оперативной памяти. В зависимости от ваших потребностей «Малютка» может, например, складывать команды, как числа, и выполнять числа, как команды.

Принципы программного управления и хранимой программы вместе с **принципом двоичного представления информации** были сформулированы группой американских ученых во главе с Дж. фон Нейманом при разработке одного из первых компьютеров. С тех пор эти принципы называют **принципами фон Неймана**.

Каждая команда состоит из операционной и адресной частей. Операционная часть содержит указание компьютеру, какую операцию надо выполнить, а адресная часть информирует компьютер, где располагаются данные, над которыми нужно выполнить данную операцию.

В обычной жизни мы привыкли, что большинство арифметических операций выполняется над двумя числами. Поэтому естествен-

ной кажется мысль, что в адресной части команды нужно указывать три адреса: адрес первого компонента действия, адрес второго компонента и адрес, по которому компьютер отправит результат. И действительно, существуют компьютеры, в которых команды содержат три адреса. Такие компьютеры называют *трехадресными*.

Оказалось, однако, что *одноадресные* компьютеры имеют большее быстродействие, чем трехадресные. Это и понятно — ведь на сами действия тратится столько же времени, а разыскивать данные по трем адресам гораздо дольше, чем по одному. Поэтому большинство современных компьютеров является одноадресными. И «Малютка» тоже одноадресный компьютер.

Зная, что адрес каждой ячейки в памяти «Малютки» записывается двумя шестнадцатеричными цифрами, легко догадаться, что в каждой команде адресная часть занимает 8 двоичных разрядов. Остальные 4 разряда — это самые левые разряды машинного слова — отведены под код операции. Иными словами, в «Малютке» используется следующий формат команд:

NNNN	NNNNNNNN
код операции	адрес ячейки данных

Разумеется, в других компьютерах может использоваться другой формат команды (в частности, команда может занимать несколько ячеек).

Нетрудно подсчитать, что для «Малютки» имеется 16 кодов операций. Все они использованы, т. е. этот виртуальный компьютер умеет выполнять 16 элементарных операций. Коды этих операций в шестнадцатеричном виде представлены цифрами от 0 до F. Список всех операций приведен в приложении 2, а знакомиться с ними вы будете постепенно в последующих параграфах этой главы.

Для выполнения команд компьютер имеет специальное арифметико-логическое устройство. В любом компьютере оно содержит в своем составе три особые ячейки — **регистр команд**, **регистр адреса** и **сумматор**. При выполнении компьютером программы в регистр команд последовательно заносятся номера ячеек, где находятся исполняемые команды. Сами команды тоже помещаются в специальную ячейку арифметико-логического устройства и там анализируются компьютером; при этом содержимое ячейки с адресом, указанным в команде, заносится в регистр адреса. Действия, предписанные командой, выполняются в сумматоре. Ячейки арифметико-логического устройства имеют, как правило, иное количество разрядов, чем ячейки памяти. У «Малютки» регистр команд содержит 8 разрядов, регистр адреса и сумматор — по 12 разрядов (хотя в реальном компьютере сумматор, как правило, имеет больше разрядов, чем ячейка оперативной памяти). В момент включения «Малютки» во все ячейки оперативной памяти и арифметико-логического устройства заносятся нули. Это, в частности, означает, что в начальный момент в регистре команд стоит 00. Поэтому содержимое ячей-

ки с адресом 00 воспринимается «Малюткой» всегда как команда. Более того, «Малютка» сконструирована так, что команда в ячейке 00 всегда воспринимается как указание начать исполнение программы с того адреса, который указан в адресной части этой команды.

Приведем несколько команд, понимаемых «Малюткой»:

- 0NN — пересылка содержимого ячейки с адресом NN в сумматор;
- BNN — умножение содержимого сумматора на содержимое ячейки с адресом NN;
- C00 — выдача на табло содержимого сумматора в формате целых чисел;
- F00 — остановка.

Для примера разберем, как выполняется следующая программа, написанная во втором столбце таблицы 24.

Таблица 24

Адрес ячейки	Содержимое ячейки	Комментарий
00	002	Первая команда программы находится во второй ячейке. В регистр команд заносится код 02
01	002	Число 2
02	001	Содержимое ячейки с адресом 01 вызывается в сумматор. В регистр команд заносится код 03
03	B01	Содержимое сумматора умножается на содержимое ячейки с адресом 01. В регистр команд заносится код 04
04	C00	Содержимое сумматора высвечивается на табло. В регистр команд заносится код 05
05	F00	Остановка

Каждому ясно, что «Малютка» по этой программе вычисляет дважды два. Но обращаем ваше внимание, что для создания даже такой совсем простой программы нужно было точно знать, в какой ячейке будет находиться константа 2 и с какой ячейки начинается программа. Поэтому при программировании в машинных кодах программисту в первую очередь приходится заботиться о распределении памяти компьютера.

ВОПРОСЫ И ЗАДАНИЯ

1. Как устроена память компьютера?
2. Что такое адрес ячейки памяти?
3. Что представляет собой содержимое ячейки памяти?
4. В чем состоит принцип программного управления компьютером? В чем состоит принцип хранимой программы? Какой еще принцип был предложен фон Нейманом для конструирования компьютеров?

5. Какова структура машинной команды? В чем отличие одноадресного компьютера от трехадресного?
6. Для чего предназначены счетчик команд и регистр команд?
7. Для чего предназначен сумматор?
8. Как будут выглядеть в оперативной памяти «Малютки» следующие числа: −3, 86, −33, −111, 24?
9. Каков диапазон целых чисел, которые могут быть записаны в оперативную память «Малютки»?
10. Рассмотрите заполнение оперативной памяти «Малютки», приведенное на рисунке 46.
    а) С какой ячейки начинается программа?
    б) Какие результаты напечатает «Малютка» на табло при исполнении этой программы?

Первые программы для «Малютки»

Картинка, появившаяся на экране вашего компьютера, изображает «внутренний мир» виртуального компьютера «Малютка». Возможно, правда, вместо изображения памяти вы видите программ-

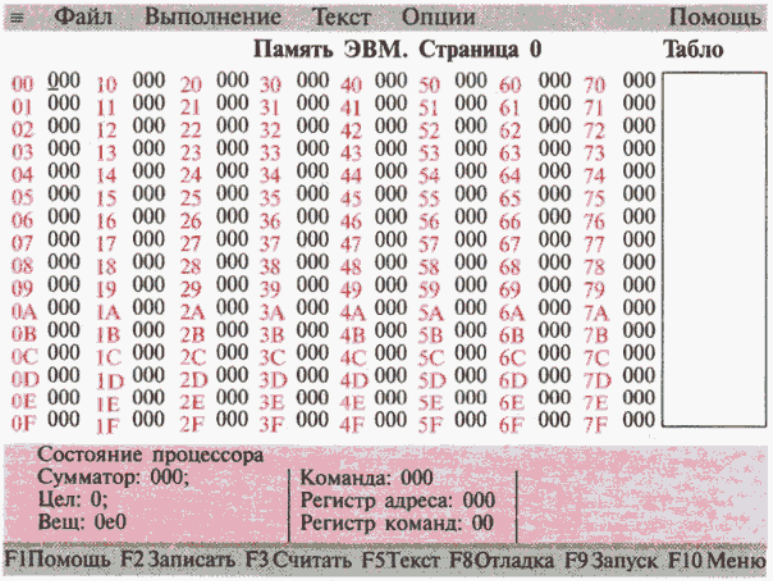


Рис. 47. Общий вид экрана после загрузки «Малютки»

ный листок, похожий на тот, который был перед вами, когда вы работали с Паркетчиком. О его назначении мы поговорим позже, а сейчас нажмите клавишу <F5>. На вашем экране появилось изображение нулевой страницы оперативной памяти. Короче говоря, экран выглядит так, как показано на рисунке 47.

Мигающая черточка — курсор, показывающий разряд, в который будет вводиться очередной символ.

❶ Заполните ячейки памяти так, как показано на рисунке 46. Чтобы запустить программу, нажмите клавишу <F9>. Если вы все сделали правильно, на табло появились два числа, а на экране — диагностика: «Исполнение завершено успешно». Сравните получившиеся два числа с вашими ответами в задании 10 б).

Если исполнение программы произошло не так, как должно быть— выдана диагностика об ошибке во время исполнения или результаты не совпали с вашими, — проверьте, правильно ли набрана программа. Кроме того, вы можете применить режим «Отладка»: при нажатии клавиши <F8> «Малютка» выполняет за один раз ровно одну команду программы, при этом на панели «Состояние процессора» вы можете увидеть, как именно исполняется эта команда.

Как вы думаете, какие результаты напечатала бы «Малютка», если в ячейке 08 вместо B01 разместить команду B03?

❷ Замените указанным образом команду в ячейке с адресом 08. Запустите программу.

На экране появилась диагностика: «Ошибка во время исполнения! Переполнение». Это означает, что для записи результата «Малютке» не хватает одиннадцати разрядов ячейки, т. е. результат вышел за границы промежутка [–2047; 2047]. Конечно, в реальных компьютерах диапазон допустимых чисел значительно шире, но разрядная сетка все равно ограничена, поэтому эффект переполнения разрядной сетки, хотя и редко, иногда дает о себе знать.

Разобравшись с готовыми программами, составьте свою собственную программу, которая бы вычисляла и выдавала на табло произведение числа, расположенного в ячейке 01, на 11, 12, 13, 14. Не забудьте, что начать надо с распределения памяти!

❸ Заполните память «Малютки» в соответствии с написанной вами программой. Отладьте программу. Проверьте ее работу для трех различных чисел, записанных в ячейку 01.

Конечно, одной операцией умножения не обойдешься. Команда ANN — это команда сложения содержимого сумматора с содержимым ячейки с адресом NN. Используя эту команду и команды, описанные в объяснительном тексте параграфа, составьте программу для вычисления значений выражения

$$5x^3 - 6x^2 - 18x + 14$$

для разных значений переменной x.

❹ Заполните память «Малютки» в соответствии с написанной вами программой. Отладьте программу. Поручите «Малютке» вычислить значения этого выражения для всех целых значений x, заключенных в диапазоне от -2 до $+2$.

Надеемся, что ваше первое знакомство с «Малюткой» прошло успешно.

§ 49. Команды передачи управления

В предыдущем параграфе вы познакомились с некоторыми арифметическими командами «Малютки». Но все рассматривавшиеся там программы реализовывали лишь линейные алгоритмы. Вы же, конечно, помните, что в алгоритмизации важную роль играют конструкции ветвления и цикла. Их общее свойство заключается в том, что *эти конструкции меняют порядок выполнения действий в алгоритме*. Значит, в компьютере, чтобы можно было реализовывать такие конструкции, нужно иметь команды, изменяющие по нашему указанию регистр команд. Такие команды называются **командами передачи управления**. В «Малютке» таких команд три:

4NN — безусловная передача управления на ячейку с адресом NN;
DNN — передача управления на ячейку с адресом NN, если 12-й разряд равен 1, т. е. содержимое сумматора меньше или равно 0;
ENN — передача управления на ячейку с адресом NN, если содержимое сумматора тождественно равно 0 (все разряды нулевые).

Команда безусловной передачи управления работает очень просто: «Малютка», встретив в программе такую команду, просто засылает в регистр команд адрес NN, и следующая команда выполняется уже из этой ячейки. Встретив команду DNN, «Малютка» проверяет содержимое 12-го разряда сумматора и, если оно 1, то в регистр команд посылается адрес NN, в противном случае регистр команд, как обычно, увеличивается на 1. Аналогично работает и команда ENN.

Чтобы лучше освоить эти команды, разберем следующую задачу.

Условие: выдать таблицу умножения заданного числа.

Уточнение: если задано число A, то требуется вывести на табло $1 \cdot A$, $2 \cdot A$, ..., $9 \cdot A$.

Алгоритм решения этой задачи написать совсем несложно:

```
Ввести A;
Делать от I=1 до 9;
{       Вычислить R=A·I;
        Напечатать R
} (* конец цикла *)
Остановить машину;
```

Теперь надо преобразовать этот алгоритм в программу для «Малютки». Как вы, вероятно, помните, это значит записать алгоритм допустимыми действиями данного исполнителя.

Написание программы для «Малютки» удобнее начинать с распределения памяти (табл. 25). Из записи алгоритма видно, что нам потребуются ячейки под переменные *A*, *I*, *R*, под константы 1 и 9. Быть может, и еще для чего-нибудь.

Таблица 25

Распределение памяти

Адрес ячейки	Содержимое ячейки	Комментарий
00		Здесь будет указано, где находится первая команда программы
01		Здесь будет значение переменной *A*
02	1	Константа; вводится один раз
03	9	Константа; вводится один раз
04		Здесь будет значение переменной *I*
05		Здесь будет значение переменной *R*
06—09		Оставим на всякий случай место

В ячейку 00 занесем константу 00A, указывающую, что программа начинается с ячейки 0A. Теперь можно приниматься за программу.

Ввод числа *A* производится вручную. Поэтому в программе никаких операторов для этого не требуется.

Далее стоит оператор цикла. Есть много способов его реализации. Воспользуемся таким способом:

- Сначала запишем в переменную цикла начальное значение.
- Затем проверим, не пора ли закончить цикл.
- И если пора, выйдем из цикла (на оператор остановки).
- Если не пора, выполним тело цикла.
- Последними операторами цикла сделаем увеличение I на 1.
- Поставим безусловный переход на проверку в начале.

Постараемся записать это на языке «Малютки»:

```
0A  002     Начальное значение засылаем в сумматор...
```

Теперь надо это значение из сумматора переслать в ячейку 04, где располагается переменная *I*. Но такой команды нам пока в описании команд не попадалось. Но, конечно, такая команда есть: раз имеется вызов содержимого ячейки в сумматор, то должна быть и команда, выполняющая обратное действие. Команда пересылки содержимого сумматора в ячейку с адресом NN выглядит так: 1NN. Воспользуемся этой командой:

```
0B  104    Сохраняем значение в переменной цикла.
```

Теперь встает вопрос о проверке. Имеет смысл вычесть из *I* конечное значение цикла 9 и, если в результате получится отрицательное число, выходить из цикла.

Но у «Малютки» нет проверки только на отрицательность (см. выше). Зато есть проверка на нуль. Поэтому вычитаем из *I* число 10 и в случае равенства 0 выходим из цикла.

Нет у «Малютки» и операции вычитания. Поэтому имеет смысл вместо 9 запасти в третьей ячейке константу –10 (отображается как 80A). Так что немного меняем заполнение памяти.

Продолжаем написание программы:

```
0C  A03    Вычитаем из I десятку (переменная цикла к этому
           моменту находится в сумматоре).
0D  E...   Если получили 0, выходим из цикла. Пока неизвест-
           но куда.
0E  001    Вызываем A в сумматор.
0F  B04    Умножаем A на I.
```

Собственно говоря, *R* нам и не надо. Можно:

```
10  C00    Сразу выдать сумматор на табло.
```

Пора заканчивать цикл:

```
11  004    Вызываем I в сумматор.
12  A02    Добавляем к сумматору 1 и...
13  104    Сохраняем в I.
```

Осталось передать управление на начало цикла. Начало цикла, между прочим, предполагает, что в сумматоре должна быть переменная *I*. Это в нашей программе выполняется. Итак,

```
14  40C    Безусловная передача на начало цикла.
15  F00    Остановка машины.
```

Не забудем записать в ячейку 0D адрес передачи управления:

```
0D  E15
```

... И программа готова.

Если теперь посмотреть на получившуюся программу, то станет ясно, что мы перестраховались при распределении памяти. Более того, даже переменная *R* нам не понадобилась. Однако в жизни всякое бывает, так что лучше в меру перестраховаться.

ВОПРОСЫ И ЗАДАНИЯ

1. Какие команды называют командами передачи управления? Какова роль таких команд в реализации алгоритмов?
2. Созданную нами программу по составлению таблицы умножения можно укоротить на одну команду. Догадайтесь какую. Запишите модернизированную программу.
3. Напишите программу, вычисляющую функцию y, заданную следующим образом:

а) $y = \begin{cases} x - 5, \text{ если } x \le 0 \\ x^2 - 5, \text{ если } x > 0; \end{cases}$

б) $y = \begin{cases} x + 5, \text{ если } x < 0 \\ x^2 - 5, \text{ если } x \ge 0; \end{cases}$

в) $y = \begin{cases} x - 5, \text{ если } x \le 0 \\ x^2 + 5, \text{ если } x > 0. \end{cases}$

4. Напишите программу, вычисляющую значения многочлена

$$5x^3 - 6x^2 - 18x + 14$$

для всех целочисленных x из промежутка $[a; b]$, где a и b — заданные целые числа.

5. Напоминаем, что рядом Фибоначчи называется ряд из чисел

1, 1, 2, 3, 5, 8, 13, 21, 34, 55,... ,

в котором каждый член, начиная с третьего, получается суммированием двух предыдущих.

Напишите программу, выдающую по номеру члена ряда его значение. (Например, введя 5, должны получить 5, 7 → 13, 3 → 2, 10 → 55 и т. д.)

6. а) Напишите программу, определяющую, делится ли целое число y на целое число x ($x < y$). Для этого используйте вычитание x из y в цикле.

б)* Напишите программу, определяющую, является ли число, находящееся в ячейке 01, степенью тройки. Если да — выдать на табло 1, если нет — выдать 0.

Ветвления и циклы на «Малютке»

Сначала займемся таблицей умножения.

❶ Заполните память «Малютки» в соответствии с программой, разобранной в объяснительном тексте или модифицированной вами при выполнении задания 2. Отладьте программу. Проверьте ее работу для трех различных чисел A (тех, для которых вы еще помните таблицу умножения).

❷ Заполните память «Малютки» в соответствии с программой, созданной вами при выполнении какого-либо пункта задания 3 (по вашему выбору). Отладьте программу. Проверьте ее работу для трех различных чисел A: -2, 0, 5.

А теперь используем отлаженную программу для получения таблицы значений избранной вами функции. Предлагаем вам последовательно печатать на табло «Малютки» значение аргумента и значение функции последовательно для всех целочисленных значений аргумента из промежутка $[-10; 10]$. Для этого мы советуем вам организовать цикл, в котором значение x за одно исполнение тела цикла будет увеличиваться на 1 (начальное значение x равно -10). Заметьте, что этот цикл вы можете организовать в памяти «Малютки» в любом удобном вам месте — для этого достаточно правильно организовать нужные передачи управления на те или иные фрагменты программы. Кроме того, с помощью клавиш <Вставка> (<Insert>) и <Удаление> (<Delete>) вы можете перемещать фрагменты программы по памяти «Малютки». Обдумайте наши советы и составьте соответствующую программу.

❸ Заполните память «Малютки» нужным образом. Отладьте программу и получите результаты. Используя полученные данные, постройте график избранной вами функции.

В следующем задании вам предстоит провести небольшое исследование. Но сначала вспомните программу, созданную вами при решении задачи 4 из § 49.

❹ Запишите в память «Малютки» программу, составленную вами к задаче 4 из § 49. Отладьте программу и проверьте правильность ее работы для нескольких значений a, b и x. Найдите наименьшее значение a и наибольшее значение b, для которых «Малютка» по вашей программе может вычислять значение указанного многочлена для всех целых значений переменной x из промежутка $[a; b]$.

В последнем задании результат зависит от того, какую программу вы создали. Можно составить программу, для которой $a = -7$ и $b = 8$. Удалось ли вам получить такой результат?

❺ Запишите в память «Малютки» программу, составленную вами к задаче 5 из § 49. Отладьте программу и проверьте правильность ее работы для нескольких первых членов ряда Фибоначчи. Найдите наибольший номер, для которого «Малютка» по вашей программе может вычислить член ряда с этим номером.

Осталось проверить на «Малютке», как вы справились с заданием 6 к этому параграфу.

❻ Запишите в память «Малютки» программу, составленную вами к задаче 6 а) из § 49. Отладьте программу и проверьте правильность ее работы для нескольких чисел x и y. Используйте эту программу при составлении программы для решения задачи 6 б).

§ 50. Поиск максимума

Нередко приходится обрабатывать большое количество однородной информации, например найти самого быстрого ученика в школе или составить список двадцати крупнейших городов мира, упорядочив их по численности населения. Фактически здесь речь идет об одной и той же задаче: *найти наибольший элемент в некотором заданном множестве чисел.*

Чтобы лучше понять, как решать такую задачу, представьте себе, что ваш класс в алфавитном порядке вызывают на медосмотр, а любознательный врач задался целью выявить самого высокого ученика. Скорее всего, врач поступит так. Вызвав первого ученика, он запишет его фамилию и рост. Пригласив следующего, он сравнит его рост с заданным числом. Если этот ученик оказался выше, врач предыдущую запись зачеркнет и запишет сведения о новом ученике, в противном случае сохранится прежняя запись. Затем он вызовет следующего ученика и т. д. Когда пройдет весь класс, на листке у врача останется фамилия и рост самого высокого ученика.

Давайте посмотрим, как с такими задачами справляется «Малютка». Пусть, к примеру, в ячейках 41, ..., 4A находятся различные целые числа. Требуется найти максимальное число среди этих чисел.

Описанный выше подход к поиску максимального числа можно представить следующим алгоритмом, в котором M — переменная, значением которой будет искомое максимальное число, N — переменная, значением которой является номер ячейки, где хранится это число, а R — вспомогательная переменная:

```
{Заслать в M содержимое 41-й ячейки;
 Заслать в N число 41;
 Делать от I=2 до 10
        { Заслать в ячейку R содержимое 40+I-й ячейки;
              Если (R>MIN) то {M=R; N=40+I}
        } (* конец цикла *)
 Выдать на табло M и N}
```

Преобразование этого алгоритма в программу для «Малютки» начинаем, как обычно, с распределения памяти. Нам требуется выделить ячейки под переменные N, I, R, M, под константы 2, –11 (для проверки условия окончания цикла), 40, 1. Быть может, и еще для чего-нибудь. В таблице 26 представлено возможное распределение памяти.

Таблица 26

Адрес ячейки	Содержимое ячейки	Комментарий
00		Здесь будет указано, где находится первая команда программы
01	1	Константа; вводится один раз
02	2	Константа; вводится один раз
03	−11	Константа; вводится один раз
04	40	Константа; вводится один раз
05		Здесь будет значение переменной *I*
06		Здесь будет значение переменной *R*
07		Здесь будет значение переменной *N*
08		Здесь будет значение переменной *M*
09		Оставим на всякий случай место

В ячейку 00 занесем константу 00A, указывающую, что программа начинается в ячейке 00A, и начнем исполнение алгоритма...

```
0A  004  Не запасли константу 41, а ее надо заслать в N,
0B  A01  поэтому вызываем константу 40 в сумматор,
0C  107  добавляем 1 и сохраняем в N.
0D  041  Содержимое ячейки 41 вызываем в сумматор
0E  108  и сохраняем в ячейке M.
0F  002  Вызываем начальное значение I, равное 2,
10  105  и засылаем его в I.
11  A03  Проверяем, не пора ли кончать цикл.
12  E...  Если да — выход из цикла, только пока неизвестно
         куда.
```

Немного приостановимся. Теперь от нас требуется в ячейку *R* записать содержимое ячейки с номером *I* + 40. Конечно, мы можем к *I* прибавить 40. Но как обратиться к ячейке с переменным номером? Вот здесь-то и воспользуемся способностью «Малютки» не различать числа и команды. Число 40 + *I*, воспринятое как команда, означает посылку в сумматор содержимого этой ячейки. Стало быть, надо вычислить 40 + *I* и сохранить в ячейке, входящей в программу:

```
13  004  Вызываем константу 40 (шестнадцатеричной).
14  A05  Добавляем к ней I и
15  116  сохраняем в 16-й ячейке.
16       Здесь можно ничего не писать: ведь к моменту перехода
         «Малютки» на исполнение команды из данной ячейки
         в нее будет записано число, которое, воспринятое
         как команда, вызовет в сумматор содержимое ячейки с
         адресом 40 + I.
17  106  Записываем в R содержимое ячейки с адресом 40 + I.
```

Пришла пора проверить, что больше: значение *M* или значение *R*. Для этого нужно найти разность *M* – *R*. Однако в «Малютке», как вы помните, нет вычитания, зато есть операция смены знака сумматора: 300. Адресная часть этой команды не задействована, поскольку операция происходит над содержимым сумматора и не затрагивает содержимого ячеек памяти «Малютки». Воспользуемся этой командой:

18 300 Меняем знак у содержимого сумматора и
19 A08 вычисляем *M* – *R*.

При отрицательном значении этой разности мы должны значение M поменять на значение *R* и в *N* записать 40 + *I*, а при положительном значении ничего не менять, а снова пойти на начало цикла — увеличение счетчика цикла на 1 и проверку условия окончания цикла. Но передача управления в «Малютке» происходит, если сумматор отрицателен! Что ж, поменяем знак у сумматора еще раз и пойдем дальше:

1A 300 Меняем знак у содержимого сумматора.
1B D... Если *R* < *M*, то переходим на изменение счетчика цикла (только еще не знаем, где это будет происходить).
1C 016 Теперь «Малютка» рассматривает содержимое ячейки с адресом 16 как число и
1D 107 отправляет его в *N*.
1E 006 Заодно занимаемся пересылкой *R*
1F 108 в *M*.
20 005 Вызываем *I*.
21 A01 Увеличиваем счетчик *I* на 1.
22 105 Сохраняем новое значение счетчика цикла.
23 412 Передаем управление на проверку условия окончания цикла.
24 008 Вызываем *M*.
25 C00 Печатаем на табло максимальное из заданных чисел.
26 007 Вызываем *N*.
27 C00 Печатаем адрес ячейки, в которой располагается максимальное число.
28 F00 Остановка.

Осталось в командах 12 и 1B записать соответствующие адреса ячеек, куда надо передать управление. Надеемся, для вас это не составит труда.

Если задуматься над ключевым моментом в составленной программе, то станет ясно, что здесь мы в полной мере воспользовались принципом хранимой программы, согласно которому содержимое любой ячейки оперативной памяти может интерпретироваться компьютером и как число, и как команда. Сам прием программного изменения адреса в команде называют *переадресацией команды*.

ВОПРОСЫ И ЗАДАНИЯ

1. В условии задачи поиска наибольшего числа предполагается, что все числа, из которых выбирается максимальное, различны. Какие результаты напечатает «Малютка», если ту же программу применить к набору чисел, среди которых имеется несколько максимальных чисел?
2. а) Измените составленную в объяснительном тексте программу так, чтобы она разыскивала наименьшее число.

   б) Составьте программу, посредством которой в заданном множестве чисел разыскивались бы одновременно максимальное число и минимальное число. (Разумеется, самый простой путь — соединить программу из объяснительного текста с программой, составленной вами при решении задачи 2 а). Мы, однако, надеемся, что вам удастся составить значительно более короткую и быстрее работающую программу.)
3. Достаточно часто приходится не только находить наибольшее число в заданном наборе чисел, но и располагать их по порядку, например в порядке убывания. Такую задачу называют **задачей сортировки**.

   а) Решать задачу сортировки можно так. Найдем сначала в заданном множестве чисел максимальное число. Запишем его на отдельном листке, а из заданного набора вычеркнем. Затем в получившемся наборе снова найдем максимальное число, запишем его на листке следующим по порядку и вычеркнем из заданного набора и т. д. Ясно, что в конце концов у нас на листке будут записаны все числа из данного набора в порядке убывания.

   Составьте программу для «Малютки», реализующую данный метод сортировки (исходные числа по-прежнему располагаются в ячейках 41—4A; пусть отсортированный набор располагается в ячейках 51—5A). Возможно, кто-то уже заметил, что фактически здесь придется 10 раз применить алгоритм поиска наибольшего числа. Подумайте, как это можно использовать.

   б) Задачу сортировки можно решать и по-другому: просмотреть весь набор чисел, сравнивая каждый раз два соседних числа, и если они расположены в неправильном порядке, переставить их. При этом число, которое должно располагаться самым первым, за один просмотр поднимается ровно на одну ступеньку выше (всплывает). Именно поэтому данный метод называют «сортировкой методом пузырька». Просмотр, естественно, идет снизу.

   Составьте программу для «Малютки», реализующую сортировку методом пузырька (исходные числа так же располагаются в ячейках 41—4A).
4. Составьте программу для «Малютки», реализующую выбор из заданного набора чисел наибольшего четного числа.
5*. По кругу стоят N детей. Они выбирают того, кто будет водить. Для этого используется считалка, выводящая из круга k-го ребенка. После того как кто-либо «вышел вон», счет начинается со следующего по кругу. Договоримся, что в самом начале все дети пронумерованы по порядку (чтобы не беспокоиться о том, как их называть). Кто же будет водить?

   Составьте программу для «Малютки», с помощью которой по заданным N и k можно определить, какой ребенок будет водить в игре.

Использование переадресации при программировании для «Малютки»

В данной лабораторной работе вы будете исполнять на «Малютке» те программы, которые подготовлены вами при решении задач к § 50. Но начнем мы с программы поиска максимума, разобранной в объяснительном тексте.

❶ Заполните память «Малютки» в соответствии с программой, разобранной в объяснительном тексте. Введите в ячейки 41 — 4A десять различных целых чисел (как положительных, так и отрицательных). Отладьте программу. Проверьте с помощью этой программы свой ответ на задание 1.

❷ Заполните память «Малютки» в соответствии с программой, созданной вами при выполнении задания 2 б). Отладьте программу.

А теперь посмотрим, как «Малютка» справляется с задачей сортировки.

❸ Заполните память «Малютки» в соответствии с программой, созданной вами при выполнении задания 3 а). Отладьте программу. Заметьте, за сколько секунд ваша программа производит сортировку десяти целых чисел.

❹ Теперь заполните память «Малютки» в соответствии с программой, реализующей метод пузырька (задание 3 б). Отладьте программу. За сколько секунд эта программа производит сортировку десяти целых чисел? Как бы вы объяснили получившуюся разницу во времени?

❺ Запишите в память «Малютки» программу, составленную вами к задаче 4. Отладьте программу и проверьте правильность ее работы для нескольких различных наборов целых чисел (в частности, для набора, вообще не содержащего четных чисел).

Наконец, если кто-то взялся программировать на «Малютке» задачу про считалку (задание 5), пусть попробует свои силы в ее реализации.

❻ Запишите в память «Малютки» программу, составленную вами к задаче 5. Отладьте программу и проверьте правильность ее работы для различных значений N и k (обязательно рассмотрите случаи, если $N > k$ и если $N < k$).

§51. Вещественные числа в «Малютке»

Вспомните то, что рассказывалось о типе данных в § 37. Там говорилось, что в языке программирования, как правило, для числовых данных есть два типа — целый и вещественный. В «Малютке»

же вы до сих пор имели дело только с данными целого типа. Но это не значит, что вещественный тип данных «Малютке» абсолютно чужд. Конечно, в памяти «Малютки» любые данные представлены в виде машинного слова (вспомните о принципе двоичного кодирования информации в компьютере), поэтому «Малютка» может распознать тип данных только в том случае, если сообщить ей об этом в команде. Приведем эти команды:

2NN — сложение содержимого сумматора с содержимым ячейки с адресом NN;
5NN — умножение содержимого сумматора на содержимое ячейки с адресом NN;
600 — содержимое сумматора заменяется на обратное к нему число;
C01 — выдача на табло содержимого сумматора в формате вещественных чисел.

Когда «Малютка» встречает в программе любую из этих команд, то она и содержимое сумматора, и содержимое ячейки с данными, и результат воспринимает в формате вещественных чисел. Для «Малютки» этот формат означает следующее.

Любое ненулевое вещественное число можно представить в виде произведения числа, модуль которого заключен между 0,1 и 1, на подходящую степень числа 10. Например:

$245 = 0{,}245 \cdot 10^{3}$ $0{,}0245 = 0{,}245 \cdot 10^{-1}$ $0{,}245 = 0{,}245 \cdot 10^{0}$
$-24{,}5 = -0{,}245 \cdot 10^{2}$ $-24\ 500 = -0{,}245 \cdot 10^{5}$ $-0{,}00245 = -0{,}245 \cdot 10^{-2}$.

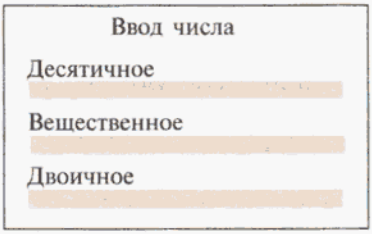
Показатель степени называют **порядком** числа, а дробный множитель в указанном представлении — **мантиссой** того же числа. Поскольку число разрядов в ячейке памяти всегда ограничено, то фактически в любом компьютере большинство дробных чисел может быть записано лишь приближенно. Это, в частности, означает, что вся арифметика с вещественными числами в компьютере, как правило, является приближенными вычислениями с выполнением соответствующих правил округления до нужного разряда.

У «Малютки» число разрядов одной ячейки, как вы помните, совсем невелико — всего 12. Один разряд отводится под знак числа, один разряд нужен для хранения знака порядка, несколько разрядов надо отдать под модуль порядка... Ясно, что округлять числа «Малютке» приходится постоянно. Итак, сообщаем: в «Малютке» все вещественные числа округлены до двух первых значащих цифр. Кроме того, каждое число приводится к произведению двузначного целого числа на степень числа 10. Например:

$120\ 000 \to 12 \cdot 10^{4}$ $0{,}000013 \to 13 \cdot 10^{-6}$ $-0{,}02674 \to -27 \cdot 10^{-3}$.

Диапазон значений показателя степени от –7 до +7.

Разумеется, в памяти «Малютки» и порядок, и значащая часть числа записываются в шестнадцатеричном коде. Но вам не нужно беспокоиться о переводе дробных чисел в коды «Малютки» — она это делает автоматически. Для этого достаточно нажать клавишу +, и перед вами появится меню:



Первая строка предназначена для ввода целых чисел, именно в ней мигает черточка-курсор, когда вы вызвали это меню. Нажав клавишу <Tab>, вы перейдете в режим ввода вещественных чисел. Здесь вы набираете двузначное число — первые две значащие цифры (и знак «–» перед ними, если число отрицательное), затем латинскую букву e, затем порядок (вместе со знаком «–», если порядок отрицателен). Например, –16e–5. После нажатия клавиши <Ввод> вы обнаружите в ячейке, где стоял курсор, машинный код этого числа: 90E. Наконец, перейдя в режим ввода двоичных чисел, вы можете набрать здесь набор нулей и единиц, который при записи в ячейку будет преобразован в шестнадцатеричный код.

Кроме того, содержимое сумматора постоянно отображается на экране и в двоичном, и в десятичном целочисленном, и в вещественном виде. Все это сделано для удобства проведения отладки программы.

Познакомившись с тем, как «Малютка» воспринимает вещественные числа, разберемся с одной полезной программой.

У «Малютки», как и у многих компьютеров, нет специальной команды процессора, вычисляющей квадратные корни. Это вовсе не означает, что нельзя заставить «Малютку» решать квадратные уравнения. Чтобы научить «Малютку» извлекать квадратный корень, давайте вспомним § 38 — историю о том, как научить калькулятор извлекать квадратный корень. Там даже алгоритм приведен. Воспроизведем его еще раз, чтобы легче было писать программу для «Малютки»:

```
{Ввести A;
Если (A<0) то {Сообщить «Корень не извлекается»;}
    Иначе
        {X=1; (* начальное приближенное значение корня *)
        N=0; (* N — счетчик числа выполнений тела цикла *)
        Делать пока (|A – X*X| > 0,0001)
```

```
            {X=0,5*(X+A/X);
            N=N+1;
            Сообщить N;
            Сообщить X (* N-е приближение к корню *)
          }(* конец цикла *)
      }   (* конец ветвления *)
}   (* конец алгоритма *)
```

Начнем, как обычно, с распределения памяти. С переменными и константами дело уже привычное. Но тут еще есть сообщение: «Корень не извлекается». Конечно, «Малютка» такое сообщить не может. Договоримся, что она в этом случае напечатает на табло букву «E» — первую букву английского слова «Error» (в переводе «Ошибка»). Но и для этого надо знать, что «Малютка» умеет печатать числа в шестнадцатеричном формате; она это делает по команде C02. Так договорившись, приступим к распределению памяти (табл. 27).

Таблица 27

Адрес ячейки	Содержимое ячейки	Комментарий
00		Здесь будет указано, где находится первая команда программы
01	1e—4	Константа 0,0001; вводится один раз
02	1	Константа; вводится один раз
03	0	Константа; вводится один раз
04	5e—1	Константа; вводится один раз
05	00E	Обозначение ошибки; вводится один раз
06		Здесь будет значение переменной *A*
07		Здесь будет значение переменной *N*
08		Здесь будет значение переменной *X*
09		Оставим на всякий случай место

В ячейку 00 занесем константу 00A, указывающую, что программа начинается в ячейке 00A, и начнем потихоньку:

```
0A  006  Вызываем A для начальной проверки на неотрицательность.
0B  E... Если да, отправляемся сообщать об ошибке; только вот куда?
0C  002  Вызываем 1 и
0D  108  засылаем в X.
0E  003  Вызываем 0 и
0F  107  засылаем в N.
10  008  Вызываем X.
11  508  Возводим X в квадрат.
12  300  Меняем знак сумматора.
```

```
13  206  Вычисляем разность A − X².
14  301  Здесь вычисляется абсолютная величина содержимого
         сумматора.
```

С этой командой вы еще не встречались, теперь и познакомились.

```
15  300  Готовим проверку окончания цикла.
16  201  Заканчиваем подготовку проверки окончания цикла.
17  E19  Если содержимое сумматора положительно, заканчива
         ем работу, иначе еще раз исполняем тело цикла.
18  F00  Остановка.
19  008  Вызываем X.
1A  600  ...
1B  506  ...
1C  208  ...
1D  504  ...
1E  108  ...
1F  C01  Печатаем очередное приближение X.
20  007  ...
21  A02  ...
22  C00  Печатаем номер напечатанного приближения.
23  410  Переходим на проверку условия продолжения цикла.
```

Теперь можно написать фрагмент программы, сообщающий об отрицательности A:

```
24  A05  Вызываем в сумматор сообщение об ошибке и
25  C02  выводим его на табло.
26  418  Переходим на остановку.
```

Чтобы закончить программу, осталось в ячейке 0B написать адрес, по которому нужно передать управление. Сделайте это самостоятельно.

ВОПРОСЫ И ЗАДАНИЯ

1. Почему в компьютере арифметические действия над вещественными числами выполняются, как правило, приближенно?
2. а) Какое наибольшее вещественное число можно сохранить в одной ячейке «Малютки»? А наименьшее?
   б) Какое наименьшее положительное число можно сохранить в одной ячейке «Малютки»?
3. В приведенной в объяснительном тексте программе прописаны не все комментарии. Заполните эти пропуски.
4. Составьте программу для «Малютки», реализующую решение квадратного уравнения $ax^2 + bx + c$ по заданным коэффициентам a, b и c.
5. Составьте программу для «Малютки», реализующую решение уравнения $x^3 - 3x + 3 = 0$ методом деления пополам с точностью до 0,001. Советуем вам сначала освежить в памяти решение задачи 5 а) из § 29.

6. Вспомните рассказанную в начале § 38 историю о том, как Петя и Коля с помощью компьютера собрались проверить, что сумма

$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\frac{1}{6}+\frac{1}{7}+\ \ldots\ +\frac{1}{N}$$

может стать больше любого числа, если только выбрать N достаточно большим. Для этого они составили такие программы (один на языке QBasic, другой на языке Pascal):

QBasic
```
N=1
S=1
WHILE (S<1000)
N=N+1
S=S+1/N
WEND
PRINT N
```

Pascal
```
Program Summa
Var N: integer;
Var S: real;
begin
    N:=1;
    S:=1;
    WHILE (S<1000) DO
        begin
        N:=N+1;
        S:=S+1/N;
        end;
    WRITE (N);
end
```

а) Когда они запустили свои программы на исполнение, то компьютер сообщил им о переполнении разрядной сетки. Какая переменная и почему могла вызвать данный эффект?

б)* Петя и Коля высказали разные гипотезы, почему произошло переполнение. Чтобы проверить свою гипотезу, каждый изменил свою программу так, чтобы на печать выдавались значения S и N после каждого выполнения тела цикла:

QBasic
```
N=1
S=1
WHILE (S<1000)
N=N+1
S=S+1/N
PRINT N; S
WEND
```

Pascal
```
Program Summa
Var N: integer;
Var S: real;
begin
    N:=1;
    S:=1;
    WHILE (S<1000) DO
        begin
        N:=N+1;
        S:=S+1/N;
        WRITE (N);
        WRITE (S);
        end;
end
```

Запустив программу, они через некоторое время обнаружили, что переменная S, достигнув некоторого значения (немного меньшего 15), пере-

стала меняться. А переменная N на каждом шаге, разумеется, продолжала увеличиваться (она-то и вызвала в конце концов переполнение). Как вы думаете, почему S перестала меняться?

Работа с вещественными числами в «Малютке»

Сначала научим «Малютку» извлекать квадратный корень.

1. Заполните память «Малютки» в соответствии с программой, разобранной в объяснительном тексте. Отладьте программу и проверьте ее работу для чисел 25, 0 и –25.
2. Заполните память «Малютки» в соответствии с программой, созданной вами при выполнении задания 4 из § 51. Отладьте программу. Решите с помощью вашей программы какое-нибудь квадратное уравнение.
3. Заполните память «Малютки» в соответствии с программой, созданной вами при выполнении задания 5 из § 51. Отладьте программу. Найдите корень указанного там уравнения с точностью до 0,01.

§ 52. В поисках общего языка

Испытав на себе все трудности программирования в машинных кодах, вы, без сомнения, способны лучше оценить достоинства языка программирования при общении с компьютером. И вам более понятным стало наше заявление, сделанное в § 37, что подавляющее большинство компьютеров имеет совершенно нечеловеческую систему команд. Об истории создания языков программирования мы будем рассказывать в следующем параграфе. А пока мы обсудим, как были сделаны самые первые шаги в поисках общего языка с компьютером.

Вы уже хорошо усвоили, что работа по созданию программы в машинных кодах начинается с распределения памяти. Ее уж никак нельзя назвать творческой. Поэтому пусть компьютер сам выполняет данную работу. Надо только сообщить ему набор переменных, используемых в алгоритме, и тип этих переменных. Именно с автоматизации распределения памяти началось «очеловечивание» общения с компьютером.

Одновременно с этим было решено научить компьютер «понимать» некоторые человеческие слова, эквивалентные командам компьютера. Ведь человеку удобнее сказать «Сложи», чем постоянно помнить код операции сложения (для «Малютки» это А). У «Ма-

лютки» еще не так много команд, а у настоящего компьютера машинных команд обычно больше полусотни. Выучить их и запомнить хоть и несложно, но малоприятно. Более того, список слов, используемых для общения с компьютером, можно стандартизировать (т. е. договориться, что все программисты будут употреблять одно и то же слово для данной машинной команды), в то время как коды машинных команд у каждого компьютера свои — они определяются конструктивными особенностями (и фантазиями конструктора) ЭВМ.

Обучать компьютер понимать человеческие слова первыми стали американцы. Поэтому эти слова являются сокращениями соответствующих английских слов. К примеру, вместо «Сложи» практически всегда используется английское «Add». Конечно, нужно иметь специальную программу в машинных кодах, исполняя которую компьютер произведет и распределение памяти, и перевод человеческих слов в последовательности машинных команд. Такая программа называется **Ассемблер**.

Чтобы лучше понять, как программируют с использованием Ассемблера, рассмотрим Ассемблер, созданный для «Малютки». Начнем, как обычно, с программы, умножающей 2 на 2. На языке машинных команд эта программа выглядит так:

```
00  002   Первая команда во второй ячейке.
01  002   Константа — число 2.
02  001   Вызов содержимого 1-й ячейки в сумматор.
03  B01   Умножение сумматора на содержимое 1-й ячейки.
04  C00   Выдача сумматора на табло.
05  F00   Стоп-машина!
```

А теперь та же программа на Ассемблере для «Малютки»:

```
DVA: dw 02;   Заказали ячейку, назвали ее DVA и записали туда 2.
ENT;          Начало программы.
LDA (DVA);    Загрузили DVA в сумматор (команда 0...).
MULT(DVA);    Умножили DVA на сумматор (команда B...).
IPRT;         Напечатали (команда C00).
HLT;          Остановили машину (команда F00).
```

Фактически то же самое, только не надо помнить, где располагается константа и с какой ячейки начинается программа, а вместо кода операции пишем соответствующее сокращение. Мы сейчас приведем список сокращений для тех команд «Малютки», которыми вы пользовались в предыдущих параграфах; полный их список вы можете найти в приложении 2.

```
LDA (...)  — 0...  — LoaD Adder (загрузка в сумматор);
STA (...)  — 1...  — Store Adder (сохранение содержимого сумматора);
RADD(...)  — 2...  — Real ADDition (вещественное сложение);
NEG        — 300   — NEGative (смена знака);
```

ABS — 301 — ABSolute value (абсолютная величина);
JMP ... — 4... — JuMP (прыжок, т. е. передача управления);
RMUL (...) — 5... — Real MULtiplication (вещественное умножение);
1DIV — 600 — 1 DIVeded to adder (1, деленная на сумматор, т. е. обратное число);
ADD (...) — A... — ADDition (целочисленное сложение);
MULT (...) — B... — MULTiplication (целочисленное умножение);
IPRT — C00 — Integer PRinT (печать на табло в формате целых чисел);
RPRT — C01 — Real PRinT (печать на табло в формате вещественных чисел);
HPRT — C02 — Hexadecimal PRinT (печать на табло информации в шестнадцатеричном формате);
CLT — C03 — CLear Table (очистка табло);
IN — C05 — INput (ввод с клавиатуры);
BPRT — C10 — Binary PRinT (печать на табло информации в двоичном формате);
JPNP ... — D... — JumP if Not Positive (прыжок, если в сумматоре неположительное число);
JZ ... — E... — Jump if Zero (прыжок, если в сумматоре нуль);
HLT — F00 — HaLT (остановка);
ENT — указание на начало программы.

Многоточие в скобках показывает, что вместо него в этой команде должно быть указано придуманное вами имя переменной. А вместо многоточия без скобок — такое стоит, как вы заметили, в командах передачи управления — надо поставить метку. **Метка** — это тоже имя, только не переменной, а команды, с которой мы хотим что-либо сделать (например, передать на нее управление). Для «Малютки» метка, как и имя переменной, представляет собой комбинацию от одной до восьми букв или цифр, причем первый символ обязательно буква. Разумеется, в качестве имен запрещено использовать слова-команды языка Ассемблера (если вы по забывчивости воспользуетесь таким словом, «Малютка» выдаст вам сообщение об ошибке).

Осталось прояснить, как объявляются переменные и метки. Для переменной вы указываете имя, затем ставите двоеточие и пишете один из операторов: ds или dw. После оператора ds ставится пробел и затем записывается натуральное число. Встретив такой оператор, компьютер зарезервирует в памяти столько ячеек, сколько указывает данное натуральное число. Этот оператор полезен, когда нужны переменные для промежуточных вычислений, значения которых заранее неизвестны, или для хранения результатов работы программы. Вспомните, к примеру, задачу сортировки, в которой десять исходных чисел, размещенных в одном фрагменте памяти, нужно было расположить в порядке убывания в другом фрагменте (см. задание 3 к § 50).

После оператора dw вы записываете число, которое намерены иметь в качестве начального значения переменной, имя которой указано перед этим оператором. Если же после оператора dw записать несколько чисел через запятую, то «Малютка» заполнит ими в своей памяти подряд столько ячеек, сколько чисел записано. А первое из них будет «Малюткой» отождествляться с именем переменной, указанным перед оператором dw. Дадим один совет: числа в операторе dw лучше начинать с цифры 0, чтобы не возникло путаницы с именем метки.

Именно о метках сейчас пойдет речь. Если после dw записать имя метки, то «Малютка» в качестве значения переменной будет рассматривать адрес ячейки, отмеченной данной меткой. Например, в результате работы с фрагментом программы

```
BEGIN:    dw MASSIV;
MASSIV:   dw 012, 801, 00A;
```

в ячейку, отведенную для значения переменной BEGIN, будет помещен адрес ячейки, в которой будет размещаться первое из трех подряд записанных чисел.

Точка с запятой после каждой команды показывает, что описание команды, по мнению составителя программы, закончено.

Вот и все, что мы хотели рассказать об Ассемблере.

ВОПРОСЫ И ЗАДАНИЯ

1. Что такое Ассемблер?
2. Зачем нужны метки?
3. а) Определите, для решения какой задачи была составлена следующая программа:

```
VVOD1: ds1;
VVOD2: ds1;
ENT;
IN;
STA (VVOD1);
IN;
STA (VVOD2);
NEG;
ADD (VVOD1);
JPNP YES;
LDA (VVOD1);
IPRT;
LDA (VVOD2);
IPRT;
JMP END;
YES: LDA (VVOD2);
IPRT;
LDA (VVOD1);
IPRT;
END: HLT;
```

б) Укажите в этой программе имена переменных и меток.

в) Определите, числовые данные какого типа обрабатывает данная программа.

г) Измените программу так, чтобы «Малютка» запрашивала тип вводимых данных и обрабатывала их в соответствующем формате. (Совет: обдумайте, как вы будете сообщать «Малютке», в каком формате вы собираетесь вводить данные.)

4. Составьте программу на Ассемблере «Малютки», реализующую сортировку заданного набора чисел по убыванию.

Программирование на Ассемблере «Малютки»

Перед вами задания последней лабораторной работы. Но в ней вам предстоит познакомиться с несколькими новыми режимами, предусмотренными в «Малютке». Прежде всего мы сообщаем, что при загрузке в компьютер «Малютки» вы как раз видите окно, в котором пишутся программы на Ассемблере.

❶ Запишите в текстовом окне программу, приведенную в задании 3 а) из § 52. О том, как редактировать текст программы, вы можете узнать, войдя в пункт «Редактор», расположенный в верхнем меню. Чтобы программу перевести в машинные коды, выберите в верхнем меню пункт «Выполнение», а в нем пункт «Компиляция» (что означает это слово, мы расскажем в следующем параграфе, хотя вы, вероятно, способны догадаться о его значении и сейчас). Если в тексте программы вы допустили ошибки, то при компиляции компьютер вам на это укажет. После компиляции посмотрите выполнение программы в режиме отладки. Проверьте, совпали ли результаты работы программы с вашими ожиданиями.

❷ Запишите в текстовом окне «Малютки» программу, составленную вами при выполнении задания 4. Откомпилируйте программу. Проверьте правильность ее работы. Сравните заполнение памяти «Малютки», полученное в результате компиляции, с тем, что было у вас в результате выполнения задания 3 к § 50.

§ 53. История создания языков программирования

Изобретение Ассемблера очень облегчило жизнь программистов. И это не замедлило сказаться — производительность их труда сразу же возросла в 2 – 3 раза. Но тем не менее работа на машине все

равно требовала достаточно высокой профессиональной подготовки. А хотелось «очеловечить» язык общения с компьютерами настолько, чтобы он стал доступен даже широкому кругу пользователей.

Не случайно инициатором этой работы стала знаменитая фирма IBM (International Business Mashin Company). С опозданием выйдя на рынок компьютеров, фирма испытывала трудности с их продажей. Для нее было бы крайне заманчиво предложить компьютер, доступный не только профессиональным программистам, но и научным работникам и инженерам, привыкшим к алгебраическим формулам.

В 1953 г. Джон Бэкус предложил разработать язык, позволяющий резко упростить программирование модели IBM-704. Естественно, предстояло разработать и **транслятор** — программу, переводящую предложения (операторы) нового языка в машинные коды. С самого начала группе Бэкуса не очень-то доверяли, поскольку более ранние попытки совершенствования машинного кода, в том числе язык Ассемблера, нередко рекламировались как создание «почти человеческих» средств общения с компьютером.

Тем не менее система, названная Фортран (FORTRAN — FORmula TRANslator — переводчик формул), в апреле 1957 г. была готова и позволяла не только переводить формулы в машинный язык, но и, самое главное, автоматизировала организацию циклов. Так, небезызвестная вам конструкция

```
Делать от i = 1 до 10
{ ..... }
```

записывается на Фортране почти «по-человечески»:

```
DO 1 I = 1, 10
...
1 CONTINUE
```

Здесь единицей отмечен последний оператор цикла, поскольку в заголовке имеется ссылка на эту метку.

Успех системы превзошел все ожидания — уже к 1958 г. более половины всех машинных команд на компьютерах IBM-704 было получено не вручную, а с помощью транслятора с языка Фортран.

Подчеркнем еще раз следующее: Фортран создавался не как принципиально новое средство программирования, а как некий «довесок» к конкретной машине, делающий ее более конкурентоспособной на рынке ЭВМ.

Несмотря на отдельные недостатки, язык быстро стал нормой и его адаптировали для машин многих других марок. Адаптация, как вы понимаете, выразилась в том, что были написаны программы-переводчики с Фортрана на язык этих самых других машин, что, конечно, весьма непросто.

Тем не менее впервые в истории программирования выяснилось, что написать программу по хорошо поставленному заданию (имеющемуся описанию языка Фортран) несравнимо легче и дешевле, чем

писать «то, не знаю что». Иными словами, хорошая постановка задачи сама по себе, без всякого программирования, тоже стала цениться чрезвычайно высоко. Таким образом, Фортран выполнил и другую не менее важную роль — он стал тем средством, с помощью которого самые различные компьютеры нашли общий язык между собой.

Фортран несколько раз улучшался и дополнялся, дожив до конца восьмидесятых годов, когда его почти вытеснили более современные языки. До самого последнего времени он был языком создания программ для ЭВМ Национального управления по аэронавтике и космическим исследованиям США (NASA). Мало того, его варианты до сих пор используются для создания программного обеспечения вычислительного характера для самых мощных компьютеров.

Неожиданно обнаружившееся свойство Фортрана быть универсальным языком для различных БИ (можно было бы при желании научить и «Малютку» понимать его) рождало мечту разработать «значительно более лучший, универсальный, изящный и замечательный» язык, пригодный на все случаи жизни, тем более что писать трансляторы для него предстояло совсем другим людям.

Несмотря на девиз разработчиков «Лучшее — враг хорошего», разработка нового языка, названного Алгол (ALGOL — ALGOrithmic Language), заняла более двух лет, и он использовался в основном на Европейском континенте, скорее, для того, чтобы подтвердить существование в Европе специалистов по языкам программирования. Получился изящный язык, который Грейс Хоппер (о ней мы еще поговорим) определила так: «Похож на большую поэму: простой и ясный с точки зрения математики, но отнюдь не практичный».

Если европейские аналитики от программирования поставили во главу угла изящность универсального языка, то фирму IBM, потихоньку становившуюся мировым лидером в производстве компьютеров, обуяла гордыня суперуниверсальности. В 1964 г. ею был предложен язык PL/1 (Programming Language One — Язык Программирования Номер Один — и никак не меньше!). Этот язык очень многие сравнивали со складным ножом со 100 лезвиями и недоумевали, почему в него не встроена кухонная раковина. Фирма надеялась, что язык станет кульминацией всего того, что напридумывали разработчики языков программирования. Так и оказалось. В языке на самом деле было множество «изюминок». Тем не менее идея суперуниверсальных языков программирования оказалась настолько же неплодотворной, как и идея телевизора со встроенной стиральной машиной, а большое количество «изюма» набивало оскомину.

Пока шла интеллектуальная борьба за наилучший и самый универсальный из языков программирования, теоретики программирования напоминали футбольные команды, стремящиеся отбить друг у друга единственный «мяч» — признание их языка самым замечательным и универсальным.

Выход между тем был подсказан давным-давно. В детской сказочной повести «Старик Хоттабыч» каждому из игроков дали собственный мячик, с которым он мог забавляться без всяких помех. Первый такой «мячик» связан с именем бабушки современного программирования Грейс Мюррей Хоппер. В разгар второй мировой войны доктор математики Хоппер вступила в резерв ВМФ США и в июне 1944 г. получила офицерское звание. Заметим, что она дослужилась до адмирала, и перейдем к описанию ее достижений в области программирования.

Грейс Хоппер занималась разработкой программ для машины «МАРК-1», о которой мы уже рассказывали. По совершенно понятным для вас причинам она пользовалась восьмеричной системой счисления. И вот как-то раз, столкнувшись с тем, что не может разобраться со своим банковским счетом, Хоппер обратилась к своему брату-банкиру, который обнаружил, что все вычисления Грейс успешно проделывала в восьмеричной системе. Банк же по старинке считал деньги, пользуясь десятичной системой.

С этого момента Хоппер заинтересовалась системами, позволяющими общаться с машиной на более человеческом языке. Плодом ее интересов стали трансляторы А-1, А-2, А-3, В-0 и т. д. Но во всех этих трансляторах входной язык был все же очень далек от человеческого. Целью Грейс Хоппер стала возможность программировать на английском языке. При этом история с банковским счетом заставила ее ориентироваться не на физиков или на создание универсального языка программирования, а на такой язык, который бы облегчил в первую очередь экономические расчеты.

Не описывая подробно всех перипетий, связанных с этим проектом, отметим лишь, что военно-морские связи Хоппер позволили ей заинтересовать языком для деловых приложений военное ведомство США, имеющее свыше тысячи компьютеров всевозможных марок и размеров, которые использовались в основной массе именно для экономических расчетов. В итоге в 1959 г. был создан комитет по разработке нового языка. В апреле 1960 г. он опубликовал его описание, а в конце того же года несколько фирм уже предлагали трансляторы. Этот язык был назван Кобол (COBOL — COmmon Business Oriented Language — универсальный язык, предназначенный для бизнеса), и его сразу же высоко оценил деловой мир Америки. Конечно, аристократы от программирования считают его чересчур многословным и громоздким, однако, поскольку в нем используются обычные английские слова, даже администраторы и менеджеры, не знакомые с программированием, без труда понимают программный код.

Даже сегодня, в возрасте более четверти века, язык еще широко используется. Стоимость программ, написанных на нем, оценивается в 50 миллиардов долларов. Он и до сих пор вполне эффективен, если речь идет об обработке деловой информации. На основе Кобола создан вполне современный язык работы с базами данных

Кларион (Clarion). Именно на Кларионе написана база данных «Ученик», с которой вы познакомились во второй главе.

Хотелось бы обратить ваше внимание на факт, который, быть может, не бросился в глаза. Если первые разработчики не различали такие вещи, как разработка языка и написание для него транслятора, то в дальнейшем эти два процесса силами теоретиков, для которых наконец-то появилась работа, были совершенно отделены друг от друга. Языки разрабатывали одни люди, а трансляторы писались совершенно другими.

Тем не менее пользователям компьютеров очень бы хотелось, чтобы сохранялось такое ценное качество, как универсальность программ, чтобы программы, написанные, скажем, на Коболе для одного типа компьютеров, так же хорошо работали и на компьютерах другого типа.

Это привело к необходимости стандартизации описаний языков, по которым в дальнейшем различными фирмами и создавались трансляторы.

Вместо одного выкристаллизовались три направления работы:

1. Разработка языка.
2. Определение стандарта языка.
3. Написание транслятора с языка программирования.

К тому же стало ясно, что для каждого вида человеческой деятельности, связанного с обработкой информации, желательно иметь свой собственный язык программирования (вот они, долгожданные мячики для каждого игрока!):

- язык для деловых применений (например, Кобол);
- язык научно-технических расчетов (Фортран);
- язык обработки таблиц (APL — A Programming Language, или язык программирования, — еще одно скромненькое название);
- язык программирования металлообрабатывающих станков (APT — Automatically Programmed Tools — автоматически программируемые инструменты);
- язык, моделирующий, по мнению специалистов, работу мозга и позволяющий быстро создавать системы искусственного интеллекта (IPL — Information Processing Language — язык обработки информации);
- язык для управления объектами в режиме реального времени и с некоторой претензией на универсальность (АДА, русский вариант аббревиатуры ADA);
- язык «среднего» уровня для системных программистов, позволяющий получать максимально быстро работающие программы, занимающие минимум памяти (Си, от английского C);
- язык для «критичных» задач, работающий в режиме реального времени, и для бортовых компьютеров (FORTH);
- язык для обучения программированию (Паскаль, от Pascal);
- язык программирования для детей (LOGO);

- язык для тех, кто не способен изучать программирование, но очень хочет программировать (Бейсик, или Basic);
- и так далее и тому подобное...

Достаточно сказать, что только в военном ведомстве США в начале семидесятых годов использовалось до 450 языков высокого уровня и их диалектов. (Это, кстати, послужило причиной появления языка АДА с его архаичной претензией на универсальность.)

Говоря о языках программирования, нельзя обойти вниманием идеологию структурного программирования, связанную как раз с именем Эдсгера Дийкстры — выдающегося теоретика программирования. Собственно говоря, именно структурное программирование лежит в основе четвертой главы нашего учебника, а конструкции языка Паркетчика взяты нами из ядра всех современных процедурно-ориентированных структурных языков.

Поработав с Паркетчиком, вы, вероятно, даже не заметили, что он в отличие от «Малютки» не допускает передачу управления из произвольного места программы в любое другое. Отсутствие так называемого оператора GOTO — оператора безусловной передачи управления — характерная черта хорошо структурированных программ.

Казалось бы, утверждение Дийкстры, что неструктурированным программам не хватает четкой математической и логической структуры, типичная придирка интеллектуального сноба. Однако реализация его идей фирмой IBM при создании банка данных газеты «Нью-Йорк Таймс» привела к резкому удешевлению работы и ускорению сроков сдачи комплекса. Можно смело сказать, что чисто умозрительная на первый взгляд идея Дийкстры привела к экономии миллиардов долларов во многих странах. Теперь ни один из языков программирования не имеет права на существование, если на нем нельзя реализовать структурно-ориентированное программирование. Структурно-ориентированным стал даже Бейсик в варианте Quick Basic фирмы Microsoft, с которым вы имели дело, когда писали программы на QBasic. Одним словом, как заявил специалист по системному программированию Харлан Миллс: «Не думаю, что полеты космического «челнока» стали бы возможны без применения структурного программирования». Средством обучения структурному программированию и должен был стать язык Паскаль. Но по многим причинам он вырос из этих рамок, став одним из основных языков.

Вернемся к описанию процесса «очеловечивания» языков программирования. Он шел по пути все большего использования слов из человеческого языка и его грамматики, но не покушался на святая святых — процедурный стиль программирования. Эти слова — **процедурный стиль** — означают, что компьютер выступает в качестве БИ, которому нужно подробно и точно описать всю последовательность действий (процедур), необходимую для получения результата. Да и сама фон-неймановская архитектура ЭВМ словно

бы предполагает рассматривать компьютер как мощный, но глуповатый БИ.

Разумеется, теоретики программирования не могли смириться с тем, что компьютеру, словно малолетке, нельзя сказать: «Подумай сам и получи результат, связанный с исходными данными известными тебе соотношениями». Например, если тренер предложил вам построиться по росту, то он вряд ли расскажет вам алгоритм выполнения задачи. Разве что для младших ребятишек в самый первый раз он пояснит, что тот, кто выше, должен стоять правее (т. е. укажет в явном виде связь между исходными данными — вашим ростом — и результатом).

Представлялось соблазнительным предложить такой язык программирования, на котором достаточно было бы описать исходные данные, указать требуемый результат и связи между ними, а построение решения свести к некоей стандартной процедуре, которая была бы встроена в транслятор с этого языка.

В 1972 г. шотландский ученый Р. Ковальский предложил использовать в качестве такого языка модификацию языка математической логики, оперирующего с высказываниями и операциями над ними (именно о нем мы фактически упоминали, когда в конце § 43 обсуждали, что из вентилей И, ИЛИ и НЕ можно построить любую логическую функцию). Эта идея была практически реализована и обобщена учеными Франции, Шотландии, Португалии, России и других стран. Разработанный язык программирования получил название ПРОЛОГ (от французского PROgramming et LOGicque). И он действительно явился прологом в новом направлении, получившем название **логического программирования**.

Логическое программирование основано на том, что компьютер должен решать задачу в свойственной человеку манере. Оно предполагает, что сведения о задаче и предположения, достаточные для ее решения, формулируются в виде логических аксиом. Эта совокупность представляет собой **базу знаний** задачи.

База знаний может быть использована для решения задачи, если постановка задачи формализована в виде логического утверждения, подлежащего выводу из аксиом и называемого **целевым утверждением**. База знаний и соответствующее целевое утверждение называется **логической программой**.

Решение задачи состоит в выводе целевого утверждения с использованием совокупности знаний о задаче. Процесс построения решения, которое в этом случае называется исполнением программы, при этом сводится к определенной стандартной процедуре, разработанной специалистами-математиками.

Другой подход, развивавшийся практически одновременно с логическим программированием, опирался на математическое понятие функции и получил название **функционального программирования**.

В математике **функция** — это отображение объектов из одного множества, называемого **областью определения функции**, в объекты

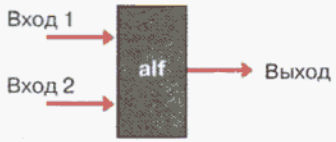


Рис. 48. Изображение функции в виде черного ящика

другого множества, именуемого **областью значений функции**. Одним из простых примеров может служить функция alf, выбирающая из двух произвольных букв букву, ранее следующую в алфавите. Например, если даны две буквы «с» и «а», то результатом отображения будет «а». Областью определения этой функции будет множество всевозможных пар букв русского алфавита, а областью значений является множество букв. Отметим, что функция alf отображает каждый элемент из области определения в один-единственный элемент из области значения. В математике такие функции называют однозначными, а в программировании **правильно определенными** или **детерминированными**.

Функцию можно представлять себе как некий агрегат или станок по переработке элементов одного множества в элементы другого множества. Эти агрегаты как на настоящем заводе можно собирать в автоматизированные линии: продукт, полученный на одном станке, поступает для обработки на следующий. Правда, если пользоваться терминологией информатики, то подобные агрегаты следует называть не станками, а черными ящиками. Скажем, рассмотренную выше функцию alf можно изобразить, как показано на рисунке 48.

Вспомните, именно так мы в третьей главе изображали черный ящик, подчеркивая, что нам совершенно неважно его внутреннее устройство. Важно лишь то, что, подав на его входы по букве, мы обязательно получим интересующий нас результат.

Впрочем, здесь как раз возникает отличие понятия функции в математике и программировании: ведь черный ящик вовсе не обязан иметь один выход. Если, скажем, мы по двум натуральным числам находим частное и остаток от деления первого числа на второе, то соответствующий черный ящик выглядит так, как показано на рисунке 49.

Принцип, состоящий в том, что функция является механически зафиксированным правилом преобразования входов в выходы, и есть одно из фундаментальных положений функционального про-

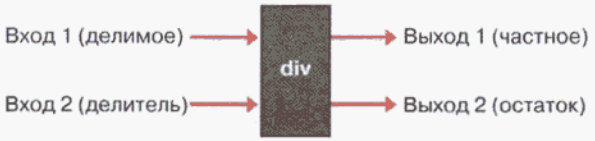


Рис. 49. Черный ящик, реализующий деление с остатком

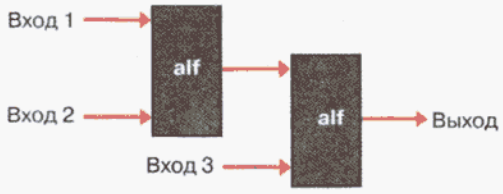


Рис. 50. Функциональная композиция черных ящиков

граммирования. Черный ящик является конструктивным блоком для функциональной программы, и с помощью объединения таких ящиков можно порождать более сложные операции. Процесс объединения ящиков называется **функциональной композицией**. Для иллюстрации процесса функциональной композиции рассмотрим функцию alf3, которая из трех букв выбирает первую по алфавиту. Ее легко представить с помощью уже определенной функции alf так, как это, например, сделано на рисунке 50.

Поскольку alf3 обеспечивает детерминированное отображение тройки букв в одну, то ее можно рассматривать как черный ящик, работающий по своему собственному правилу (рис. 51). Эта конструкция эквивалентна изображенной на предыдущем рисунке. Теперь можно забыть о том, какая работа совершается внутри нового ящика, и использовать его как единицу для вычисления или как конструктивный блок для других более сложных функций.

Таким образом, задав множество предварительно определенных черных ящиков-функций, называемых **встроенными функциями** или **примитивами**, для выполнения простых операций, можно строить новые функции, т. е. новые черные ящики, для выполнения более сложных операций с помощью этих примитивов. Затем эти новые блоки можно и далее наращивать.

Идеи функционального программирования реализованы на практике. Первым языком функционального программирования является язык LISP, созданный в начале 1960-х гг. Слово LISP является аббревиатурой английской фразы LISt Processing. Это достаточно распространенный язык. Он широко используется для решения задач искусственного интеллекта, особенно в США. В настоящее время существуют версии языка LISP для большинства современных компьютеров. Но этот язык не является единственным средством

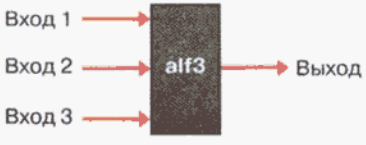


Рис. 51. Черный ящик, соответствующий функции alf3

функционального программирования. К подобным языкам относятся и такие, как Miranda, Hope, FP.

В начале 80-х гг. наметился еще один подход к программированию. На плечи компьютера теперь перекладывается и установление связей между исходными данными и результатами. Мы предлагаем компьютеру описание объектов, с которыми ему предстоит работать, а он пусть сам установит, как эти объекты связаны между собой. Такой акцент на описание свойств объектов получил название объектно-ориентированного. К **объектно-ориентированным** языкам программирования относятся языки C++ и Smalltalk.

Но это взгляд с высот сегодняшнего дня. А исторически **объектно-ориентированное программирование** выросло из расширения возможностей ЭВМ и того круга задач, которые с помощью ЭВМ решаются. Первые языки программирования умели обрабатывать только одно число. Обработка массивов чисел оформлялась специальным итерационным процессом.

В дальнейшем стало возможным с помощью одного оператора заставить машину складывать сотни и тысячи чисел, находить в массивах чисел максимумы, производить сортировку и т. п. Массив чисел стал в мышлении программистов восприниматься как абстрактный объект, над которым можно производить те или иные действия.

Однако в реальной жизни мы имеем дело не только с числовой информацией. Скажем, ученика характеризует не только его возраст, но и имя, фамилия, класс, домашний адрес... Разнородные типы данных стало возможным (и необходимым) объединять в **структуры данных**. Каждую такую структуру тоже можно обрабатывать как *единое целое*. Формализованный язык описания таких структур в единстве с действиями, которые над этими структурами можно выполнять, состоит из имен объектов и имен свойств этих объектов. Эти свойства сигнализируют компьютеру, как можно преобразовывать данный объект. Программирование на таком языке сводится к описанию объектов и структурированию исходных данных и результатов в такие объекты.

Возможность практически без ограничений увеличивать сложность структур и объектов (можно, скажем, создать объект, являющийся структурой объектов) позволяет оперировать со все большим количеством информации, и, значит, использовать компьютер для решения все более сложных задач.

ВОПРОСЫ И ЗАДАНИЯ

1. Что такое транслятор?
2. Что такое процедурный стиль программирования?
3. Чем характеризуется структурный подход к программированию?
4. В чем состоит идея функционального программирования? А объектно-ориентированного?
5. Подсчитайте, сколько языков программирования упомянуто в этом параграфе.

Эпилог

Вы прощаетесь с этой книгой, но не прощаетесь с информатикой

Перед вами последние страницы учебника. Оглядываясь на пройденный путь, легко различить наиболее значительные вершины, которые вы, надеемся, смогли покорить:

- работа с прикладным программным обеспечением;
- знакомство с миром компьютерных сетей и Интернетом;
- построение компьютерных моделей для решения жизненных задач;
- изучение алгоритмов и знакомство с языками программирования;
- изучение принципов работы самого компьютера.

Все это надежная база для дальнейшего плодотворного сотрудничества с компьютером. В современном информационном обществе все вами изученное составляет основу информационной культуры, которой должен владеть каждый.

Покидая школу, вы выбираете сферу своих будущих профессиональных интересов. Кто-то ощущает себя гуманитарием, кого-то влекут науки о природе, кто-то готов окунуться в мир абстрактных математических построений. Некоторых из вас всерьез заинтересовал открывшийся в этой книге мир возможностей компьютеров. Но любому из вас понадобится информатика, поскольку она изучает важнейшие стороны человеческого существования — процессы получения, накопления, передачи и обработки информации. А без информации невозможна никакая продуктивная деятельность!

Конечно, для каждой профессиональной сферы характерен свой взгляд на информатику. Но разноликая и в то же время единая в своих основах информатика останется с вами навсегда.

Приложения

1. Краткий справочник по HTML

Элемент	Значение
<HEAD>...</HEAD>	Заголовок
<BODY>...</BODY>	Тело документа
<TITLE>...</TITLE>	Название документа
<BASE>	Базовый адрес
<ISINDEX>	Поисковый документ
<LINK>	Общая гипертекстовая ссылка
<H?>...</H?>	Размер символов (? = 1, ..., 6)
 	Перевод строки
<P>	Начало абзаца
<PRE>...</PRE>	Отображение без изменения с учетом пробелов
<HR>	Разделитель — горизонтальная линия
<IMG SRC = "file_name">	Вставка изображения из файла
<A HREF = URL>...</A>	Ссылка на URL
<ADDRESS>...</ADDRESS>	Задание подписи или адреса
<MARQUEE>...</MARQUEE>	Вывод бегущей строки
<UL>...</UL>	Тег неупорядоченного списка
<LI>	Элемент списка
<OL>...</OL>	Тег упорядоченного списка
<CENTER>...</CENTER>	Центрирование текста
<I>...</I>	Курсив
<B>...</B>	Жирный шрифт
<TT>...</TT>	Телетайп
<U>...</U>	Подчеркивание
<S>...</S>	Перечеркнутый текст
<BIG>...</BIG>	Увеличенный текст
<SMALL>...</SMALL>	Уменьшенный текст
_{...}	Подстрочные символы
^{...}	Надстрочные символы
<FONT COLOR="color_name">... </FONT>	Цвет текста, заключенного между данными тегами

<CITE>...</CITE>	Цитирование
<STRONG>...</STRONG>	Усиление
<CODE>...</CODE>	Отображение примеров кода
<SAMP>...</SAMP>	Последовательность литералов
<KBD>...</KBD>	Ввод символов с клавиатуры
<VAR>...</VAR>	Переменная
<DFN>...</DFN>	Определение
<Q>...</Q>	Текст, заключенный в скобки
<LANG>...</LANG>	Задание языка отображения
<AU>...</AU>	Автор
<ACRONIM>...</ACRONIM>	Акроним
<ABBREV>...</ABBREV>	Аббревиатура
<INS>...</INS>	Вставка текста
<DEL>...</DEL>	Удаление текста
<TEXTAREA>...</TEXTAREA>	Поле для ввода текста

2. Виртуальный компьютер «Малютка»

2.1 Система команд для «Малютки»

Группа команд обмена содержимым ячейки и сумматора:

0NN — пересылка содержимого ячейки с адресом NN в сумматор;
1NN — пересылка содержимого сумматора в ячейку с адресом NN.

Группа команд целочисленной арифметики:

ANN — сложение содержимого сумматора с содержимым ячейки с адресом NN;
BNN — умножение содержимого сумматора на содержимое ячейки с адресом NN.

Группа команд преобразования сумматора:

300 — смена знака сумматора;
301 — вычисление абсолютной величины содержимого сумматора;
310 — сдвиг разрядов сумматора вправо, на место крайнего левого разряда заносится 0;
311 — сдвиг разрядов сумматора влево, на место крайнего правого разряда заносится 0.

Группа команд дробной арифметики:

2NN — сложение содержимого сумматора с содержимым ячейки с адресом NN;
5NN — умножение содержимого сумматора на содержимое ячейки с адресом NN;
600 — замена содержимого сумматора на обратное к нему число.

Группа логических команд:

7NN — логическое сложение сумматора с содержимым ячейки с адресом NN (ИЛИ);

8NN — логическое умножение сумматора с содержимым ячейки с адресом NN (И);
900 — логическое отрицание содержимого сумматора (НЕ).

Группа команд передачи управления:

4NN — безусловная передача управления на ячейку с адресом NN;
DNN — передача управления на ячейку с адресом NN, если 12-й разряд равен 1, т. е. содержимое сумматора меньше или равно 0;
ENN — передача управления на ячейку с адресом NN, если содержимое сумматора тождественно равно 0 (все разряды нулевые);
F00 — остановка.

Группа команд ввода/вывода информации:

C00 — выдача на табло содержимого сумматора в формате целых чисел;
C01 — выдача на табло содержимого сумматора в формате вещественных чисел;
C02 — выдача на табло содержимого сумматора в шестнадцатеричном формате;
C03 — очистка табло;
C04 — звуковой сигнал, высота которого определяется содержимым сумматора;
C05 — ввод данных в сумматор с клавиатуры;
C10 — выдача на табло содержимого сумматора в двоичном формате;
CFF — скроллинг табло, т. е. продвижение всех данных на одну строку вверх, самое верхнее значение теряется.

2.2 Ассемблер для «Малютки»

LDA (...) — 0... — LoaD Adder (загрузка в сумматор);
STA (...) — 1... — Store Adder (сохранение содержимого сумматора);
RADD (...) — 2... — Real ADDition (вещественное сложение);
NEG — 300 — NEGative (смена знака);
ABS — 301 — ABSolute value (абсолютная величина);
SAR — 310 — Shear Adder to the Right (сдвиг сумматора вправо);
SAL — 311 — Shear Adder to the Left (сдвиг сумматора влево);
JMP ... — 4... — JuMP (прыжок, т. е. передача управления);
RMUL (...) — 5... — Real MULtiplication (вещественное умножение);
1DIV — 600 — 1 DIVided to adder (1, деленная на сумматор, т.е. обратное число);
OR (...) — 7... — OR (ИЛИ);
AND (...) — 8... — AND (И);
NOT — 900 — NOT (НЕ);
ADD (...) — A... — ADDition (целочисленное сложение);
MULT (...) — B... — MULTiplication (целочисленное умножение);
IPRT — C00 — Integer PRinT (печать на табло в формате целых чисел);
RPRT — C01 — Real PRinT (печать на табло в формате вещественных чисел);

HPRT — C02 — Hexadecimal PRinT (печать на табло информации в шестнадцатеричном формате);
CLT — C03 — CLear Table (очистка табло);
BELL — C04 — BELL (звуковой сигнал);
IN — C05 — INput (ввод с клавиатуры);
BPRT — C10 — Binary PRinT (печать на табло информации в двоичном формате);
ROLL — CFF — skROLLing (скроллинг табло);
JPNP ... — D... — JumP if Not Positive (прыжок, если в сумматоре неположительное число);
JZ ... — E... — Jump if Zero (прыжок, если в сумматоре нуль);
HLT — F00 — HaLT (остановка);
ENT — указание на начало программы.

Предметный указатель

Оглавление

Учебное издание

Гейн Александр Георгиевич
Сенокосов Александр Иванович
Юнерман Нина Ароновна

Учебник для 10—11 классов общеобразовательных учреждений

Зав. редакцией *Т. А. Бурмистрова*
Редактор *Т. В. Ак[illegible]*
Младший редактор *Н. В. [illegible]*
Художник *Н. С. [illegible]*
Художественный редактор *О. В. Попович*
Компьютерная верстка *Н. Ю. Соколова*
Технический редактор *Н. Ю. Соколова*
Корректоры *Н. А. Смирнова, О. Н. Леонова*

Налоговая льгота — Общероссийский классификатор продукции ОК 005-93—953000. Изд. лиц. Серия ИД № 05824 от 12.09.01. Подписано в печать с диапозитивов 29.10.04. Формат [illegible]. Бумага офсетная. Гарнитура Таймс. Печать офсетная. Усл. печ. л. 16+0,31 форз. Усл. кр.-отт. [illegible]. Уч.-изд. л. 15,81+0,47 форз. Тираж 30 [illegible] экз. Заказ № [illegible].

Федеральное государственное унитарное предприятие ордена Трудового Красного Знамени «Издательство «Просвещение» Федерального агентства по печати и массовым коммуникациям. 127521, Москва, 3-й проезд Марьиной рощи, 41.

Федеральное государственное унитарное предприятие Смоленский полиграфический комбинат Федерального агентства по печати и массовым коммуникациям. 214020, г. Смоленск, ул. Смольянинова, 1.