ИНФОРМАЦИОННАЯ КУЛЬТУРА

БАЗЫ ДАННЫХ И ЭЛЕКТРОННЫЕ ТАБЛИЦЫ

Книга для учащихся 8 класса

ФЕДОРОВ

Самара
1997

ББК 32.97.я72
П 27

Эта книга из курса "Информационная культура" предназначена для учащихся 8 класса и рассказывает о том, как работать с базами данных и электронными таблицами.

Леонов А.Г., Подольская Н.А., Эпиктетов М.Г.
Базы данных и электронные таблицы.
Информационная культура: Книга для учащихся 8 класса.
— Самара: ОАО "Корпорация "Федоров", 1997. — 80 с.

ISBN 5-88833-012-4

Глава 1. Базы данных

§ 1. СПОСОБЫ ХРАНЕНИЯ ИНФОРМАЦИИ

1.1. Файловая система

Имеется много способов хранить данные в компьютере. Простой пример хранения информации — хранение символов в памяти компьютера.

Память современных компьютеров настолько велика, что в ней можно одновременно хранить столько символов, что их хватит на миллионы книжных страниц. Ориентироваться в таком количестве информации, а тем более быстро находить с ее помощью ответы на вопросы чрезвычайно трудно, даже если все ответы там содержатся. Представьте себе книгу даже в несколько сотен страниц, в которой нет ни заголовков, ни оглавления, и даже страницы не пронумерованы. Легко ли найти в такой книге нужный абзац? Для того, чтобы организовать и упорядочить большие объемы информации, пришлось придумать новые понятия и методы.

Основным понятием, используемым, когда речь идет о хранении информации в компьютере, является понятие ***файла***. Что такое файл? Английское слово *file* до появления компьютеров обозначало обыкновенную папку, в которую можно положить несколько листов бумаги, рисунки, фотографии, ноты и так далее. Для чего люди обыкновенно используют папки? Для того, чтобы сложить вместе все листы, объединенные чем-то общим. Например, в одной папке могут храниться все материалы о туристическом походе, в другой — все рисунки школьной художественной студии, в третьей — списки кассет с любимой музыкой и т.д. Очевидно, что ориентироваться и находить нужный материал несравнимо легче, если все листы разложены по папкам, а не свалены кучей на столе. Что можно делать с папками и листами? По существу, только две вещи — вкладывать листы в папку и вынимать их оттуда. Все остальные действия сводятся к этим двум. Например, чтобы переложить листок из одной папки в другую, нужно вынуть его из первой и вложить во вторую папку.

Говоря о файлах в применении к компьютерам, мы подразумеваем, по существу, то же самое. Только вместо листов, лежащих в папке, мы говорим о записях, а роль самой папки играет специально выделенный фрагмент памяти компьютера. Вместо

вынимания и вкладывания листов говорят о чтении и записи информации. ***Файлом*** называется именованный набор однотипных элементов данных, называемых ***записями***.

Примеры файлов:

1. Файл содержит текст. Запись — строка.
2. Файл содержит двоичный код. Запись — один байт кода.
3. Файл содержит справочник телефонных номеров. Запись — данные об одном человеке.

Упражнение:

1. Приведите другие примеры файлов и записей.

Отметим, что, глядя на один и тот же файл с разных точек зрения, можно выделять в нем записи различными способами. Например, при чтении файла с диска в оперативную память компьютера за одну операцию чтения всегда считывается один **сектор** информации (обычно это половина килобайта — 512 байт). С этой точки зрения любой файл разделен на записи по 512 байт.

Вся информация, с которой имеют дело компьютеры, так или иначе поделена на файлы. Однако деление информации на файлы подобно нумерации страниц в книге. И, подобно тому, как в книгах помещают оглавления, файлы организуют в более сложные структуры. Простейшим примером такой структуры является ***каталог*** — это файл, в котором содержится информация о других файлах (такая, как название, размер, время создания, место, где расположен файл и т.д.). С помощью каталогов удобно выбирать файлы, с которыми можно впоследствии работать.

Можно представлять себе каталог как большую папку, в которую вкладываются другие папки–файлы.

1.2. Информационные системы и базы данных

Для простейшей работы с компьютером деления информации на файлы вполне достаточно. Но для серьезных производственных применений компьютера нужны более мощные средства. В самом деле, если в телефонном справочнике содержится всего пара десятков записей, то для поиска нужного телефона их нетрудно просмотреть все. Однако если записей несколько тысяч, надо иметь специальный механизм поиска (иначе такой справочник станет совершенно бессмысленным).

На практике для хранения больших объемов информации используются ***информационные системы***. В отличие от файловой системы, которая позволяет хранить вместе информацию разных типов, информационные системы обычно специализированы и служат для хранения и обработки информации из одной, довольно узкой области.

Электронный депозитарий

Примером большой информационной системы является ***электронный депозитарий*** — место, где хранятся сведения о ценных бумагах. Очень часто предприятия выпускают свои акции в безналичной форме, и тогда информация, содержащаяся в электронной системе, является единственным основанием, по которому можно узнать о распределении акций.

Что же именно хранится в депозитарии? Во-первых, сведения о людях, владеющих акциями, — фамилия, адрес, телефон, паспортные данные. Во вторых, сколько у каждого из них акций в настоящий момент времени. В-третьих, история движения акций — когда они были приобретены, у кого куплены, кому проданы, по какой цене. В-четвертых, сведения об эмитентах — организациях, выпустивших акции в обращение. Относительно каждого эмитента надо хранить его полное название, адрес, телефон, размер уставного капитала, общее число выпущенных в обращение акций. И это далеко не полный перечень сведений, которые необходимо хранить.

Базы данных

В основе каждой информационной системы лежит ***база данных*** — совокупность предназначенных для машинной обработки данных, которая служит для удовлетворения нужд многих пользователей. Программа, которая обеспечивает обработку таких данных, называется *системой управления базами данных* (СУБД).

База данных электронного депозитария включает всю информацию, перечисленную в предыдущем разделе. Попробуем оценить ее размер, исходя из следующих данных: сведения о каждом акционере (включая данные об имеющихся акциях) занимают 2000 байт информации, сведения о каждой покупке/продаже — 100 байт, сведения об эмитенте — 1000 байт. Представим себе, что в электронном депозитарии содержатся сведения о 20 000 человек, владеющих акциями 20 эмитентов и что у каждого из них происходит в среднем одно изменение количества акций в месяц. Каков же будет объем базы данных через год? Он будет равен $20\,000 \times 2000$ (данные об акционерах и акциях) плюс 20×1000

(сведения об эмитентах) плюс 20 000 × 12 × 100/2 (данные о движении акций) байт. Всего получается 12 260 000 байт, или примерно 12 мегабайт информации (из которых две трети составляет информация о движении акций).

Упражнения:

2. На сколько вырастет объем базы данных за следующий год, если интенсивность торговли повысится и у акционеров будет в среднем по пять изменений количества акций в месяц?
3. Представим себе депозитарий биржи, на котором действуют сто участников, работающие с акциями десяти тысяч эмитентов и выполняющие в среднем по десять актов купли-продажи в день. Какой объем информации будет в базе данных через месяц после начала работы биржи?

1.3. Реляционные базы данных

Знакомство с информационными системами и базами данных мы проведем на примере ***реляционных баз данных***, которые наиболее широко используются на персональных компьютерах. Информация в этих базах хранится в виде упорядоченного набора ***записей***, каждая из которых состоит из ***полей*** фиксированного размера. Такой способ хранения информации очень похож на представление данных в виде нескольких таблиц на бумаге. Отдельным строкам таблицы соответствуют записи в базе данных, а графам — поля записей.

В простейших случаях (например, при реализации телефонного справочника) для представления данных достаточно одной таблицы, в более сложной ситуации (например, в электронном депозитарии) может потребоваться несколько таблиц, связанных между собой при помощи ссылок.

> Практически во всех существующих системах в дополнение к файлу с данными хранятся отдельно еще один или несколько служебных файлов — индексов, которые используются для ускорения поиска нужных данных.

С другой стороны, работа с базами данных на компьютере отличается от работы с таблицами на бумаге и больше напоминает работу с картотекой: записи, как и бумажные карточки, легко переставлять, добавлять и удалять.

Экранные формы

Как известно, компьютеры были созданы для людей, а не люди — для компьютеров. Знать, как представляются данные внутри компьютера, конечно, важно, но в процессе практической работы с правильно сконструированной информационной системой об этом задумываться не приходится. Разработчики таких систем обычно уделяют большое внимание тому, как хранимые данные будут показываться на экране при поиске, добавлении или изменении информации.

Формат показа данных на экране компьютера при работе с информационной системой называется ***экранной формой***. Правильно составленная экранная форма устроена так, что человеку удобно с ней работать. Заметим, что даже при работе с одной базой данных может потребоваться несколько экранных форм. Например, при поиске телефона в справочнике на экране удобно иметь краткую информацию о нескольких людях, а при внесении изменений — информацию только про одного человека, но зато в самой полной форме.

Отчеты

Возможность быстро найти нужную информацию в базе данных и посмотреть ее на экране — это, конечно, очень большое достоинство информационных систем. Однако обычно для работы нужно большее. Например, возможность напечатать ***отчет*** о состоянии базы данных. В самом деле, размеры экрана ограничены и, глядя на него, трудно сформировать общее представление о всей базе данных. Кроме того, в некоторых случаях такое «бумажное» представление данных требуется по закону.

> Не нужно думать, что закон несовершенен и не учитывает возможностей компьютеров. Просто напечатанное на бумаге гораздо труднее подделать, не оставив следов.

Поскольку с отчетами, как и с экранными формами, предстоит работать человеку, требования к ним те же самые. Однако отчеты имеют и свои особенности. Если в экранную форму обычно включается информация об одной или нескольких записях, то в отчет чаще всего входят данные о большой группе записей (или даже о всех) в базе данных. В отчет также обычно добавляются некоторые автоматически подсчитанные данные (это может быть общее число записей или, в примере электронного депозитария, сумма количества акций у всех лиц, включенных в отчет).

Упражнение:

4. Чем еще подготовка отчетов в реальных информационных системах должна отличаться от создания экранных форм?

1.4. Пример базы данных

Рассмотрим теперь пример базы данных — хранение информации о музыкальных записях на компакт–кассетах. Каждой песне, записанной на кассете, в нашем примере будет соответствовать одна запись в базе данных. Про каждую песню будут указаны:

- номер кассеты и номер песни на кассете;
- имя исполнителя или название группы;
- название песни;
- продолжительность песни.

Конечно, если вы серьезно увлекаетесь музыкой, то согласитесь, что описанной выше информации недостаточно. Но пока мы ограничимся ею, а впоследствии еще вернемся к этому примеру и дополним его.

Выберите в гипертексте раздел «Пример базы данных» и поработайте с имеющимися там примерами.

В информационной системе, которая работает с базой данных музыкальных записей, имеются две различные экранные формы для просмотра базы данных и всего два типа отчетов: полный список всех песен на всех кассетах и список песен по исполнителям. Во втором случае система сама считает общее количество песен и их общую продолжительность.

Упражнения:

5. Какие еще отчеты, по вашему мнению, должна уметь составлять информационная система музыкальных записей?
6. Почему в приведенном примере базы данных вся информация хранится в одной таблице, а не заводится несколько таблиц (как это было бы сделано, если бы картотека на кассеты заводилась без компьютера)?

§ 2. ПРИНЦИПЫ РАБОТЫ С БАЗАМИ ДАННЫХ

2.1. Представления базы данных

Детальное знакомство с реляционными базами данных мы начнем с различных способов представления данных на экране компьютера. Ранее уже говорилось, что формат такого представления называется *экранной формой*. В современных системах управления базами данных широко используется три типа экранных форм, основанных на трех типах представления данных.

N кас	N п/п	Исполнитель	Название	время
1	1	ABBA	The Winner Take It All	4:58
1	2	ABBA	The Day Before You Came	5:48
1	3	A.Faltskog	Take Good Care of Your Ch	3:46
...	...	...	...	...
5	9	Vaya Con Dios	I Don't Want to Know	3:34

Рис. 1. Табличное представление данных

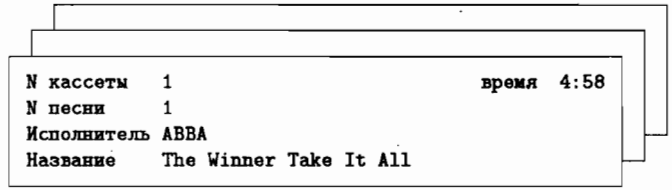


Рис. 2. Страничное представление данных

Табличное представление

Как уже говорилось, реляционная база данных очень похожа на обычную таблицу, нарисованную на бумаге. Более того, отдельный файл с данными в таких базах данных очень часто также называют *таблицей* (мы тоже будем так поступать, если это не будет вызывать путаницы). И первый тип экранных форм, который мы рассмотрим, основан на ***табличном представлении*** данных (рис. 1).

Такой способ представления удобен для просмотра данных, в нем удобно отмечать группы записей, копировать и удалять записи. Однако на рисунке виден и основной недостаток табличного представления — размер экрана (и листа бумаги) ограничен и даже в простейшем случае некоторые данные могут не поместиться в одну строку (например, название третьей песни на рис. 1 пришлось сократить).

Страничное представление

Для того, чтобы избежать отмеченного недостатка табличного представления, обратимся к другой аналогии баз данных — к картотеке. В ***страничном представлении*** данных (рис. 2) на каждой странице изображаются данные только из одной записи

N кассеты: 1		
N п/п	Исполнитель	Название
1	ABBA	The Winner Take It All
2	ABBA	The Day Before You Came
3	A.Faltskog	Take Good Care of Your Children
Авторы: Benny Andersson / Bjorn Ulvaeus		4:58
Диск: Super Trouper		Год выпуска: 1980
Фирма: Polar Music International AB		

Рис. 3. Смешанное представление данных

базы данных (как и на карточке).

Конечно, просматривать данные в таком виде не так удобно — приходится «листать» страницы (точно так же, как в картотеке приходится перекладывать карточки), но зато не возникает потребности в сокращениях.

Смешанное представление

В приведенном на рисунке 1 примере в строке все же уместилось почти все. Но как быть, если мы хотим добавить в нашу базу данных еще сведения о названии диска, на котором вышла песня, авторе музыки и слов и т.д.? Графы таблицы получатся такими узкими, что в них уже ничего не поместится. На помощь приходит ***смешанное представление*** данных (рис. 3). На экране изображена таблица с наиболее важной информацией, относящейся к нескольким записям. Кроме того, про одну запись приведены дополнительные сведения.

Несмотря на все преимущества смешанного представления, оно не очень широко используется на практике. Это связано с тем, что, с одной стороны, его труднее реализовать. Кроме того, вопрос о том, какие данные «более важны», не имеет однозначного решения. Все зависит от того, что именно просматривается и с какой целью. В приведенном примере в таблице показывается номер песни, исполнитель и название. Этот выбор удобен при просмотре списка кассет, однако при других просмотрах более удобным может оказаться другой выбор полей.

Упражнения:

7. Придумайте аналогию смешанному представлению базы данных в бескомпьютерном мире.

8. Приведите пример, в котором нерационально вставлять в табличную часть формы:
а) номер песни;
б) исполнителя песни;
в) название.

2.2. Ввод данных

Вы, конечно, уже знакомы с текстовым редактором и, наверное, удивитесь: «Что сложного может быть во вводе данных? Достаточно подвести курсор к нужному месту и просто ввести данные с клавиатуры». В общем-то, вы совершенно правы, и, хотя ввод информации в базу данных имеет свои особенности, его нетрудно освоить.

Первое, чем отличается база данных от текстового редактора, — это наличие в ней механизмов защиты информации. Дело в том, что текст обычно пишет один человек, а базу данных составляют (и пользуются ей) много человек. Если в своем текстовом файле вы случайно удалите слово или даже строчку, восстановить их не составит особого труда (даже если вы еще не научились пользоваться командой откатки). Но при работе с базой данных вы можете, случайно нажав не ту кнопку, испортить ценную информацию и даже не заметить этого. Поэтому во всех информационных системах имеются разные режимы работы:

- просмотр базы данных;
- внесение изменений (редактирование);
- добавление (ввод) информации в базу.

Как правило, для перехода из одного режима в другой используется специальная команда.

Чуть позже мы разберем все команды подробно, а пока рассмотрим одну важную характеристику баз данных, которую мы пропустили при первом знакомстве.

Типы полей базы данных

Раньше, в пункте 1.3, говорилось о том, что поля в реляционной базе данных имеют фиксированный размер. Однако кроме размера у них есть еще одна важная характеристика — ***тип поля***. Рассмотрим наиболее важные типы полей.

- **Текстовое поле.** В поле такого типа можно записывать произвольный текст (разумеется, не длиннее размера поля). Это

наиболее универсальный тип поля, в системе КуМир/СУБД он обозначается буквой A.

В некоторых СУБД существуют еще текстовые поля переменной длины (например, в dBase такие поля называются MEMO), однако на практике они используются довольно редко (так как обычно на использование таких полей налагаются ограничения — по ним, например, нельзя проводить поиск).

- **Числовые поля** предназначены для записи в них чисел. В некоторых СУБД у числового поля можно задавать количество цифр до и после запятой, в других (например, в Paradox и КуМир/СУБД) имеется всего два вида числовых полей (короткое целое S, в которое можно записать числа от −32767 до +32767, и длинное N, с практически неограниченным диапазоном допустимых значений).

- **Дата.** В поле этого типа (в системе КуМир/СУБД оно обозначается D) можно записывать даты.

Конечно, можно построить базу данных, у которой все поля текстовые (как у таблицы на бумаге), но в реальных системах обычно так никто не поступает, поскольку использование нетекстовых полей позволяет упростить обработку данных на компьютере.

Ввод информации в текстовые поля базы данных практически ничем не отличается от ввода в обычном текстовом редакторе. Нужно только учитывать, что в некоторых ситуациях не разрешается оставлять поле пустым. Например, в каталоге музыкальных кассет вполне допустима запись без указания исполнителя или названия песни, но в депозитарии не может быть счета с незаполненной фамилией владельца (это просто не разрешено по закону).

При заполнении числовых полей и полей–дат действуют те же клавиши, но есть и отличия. В самом деле, в такие поля можно вводить не все, что угодно, и все информационные системы контролируют правильность ввода.

Некоторые информационные системы просто не позволяют вводить неправильные символы (например, если вы при вводе числа нажмете букву *ю*, то раздастся сигнал и появится сообщение «Неправильный символ», но в поле ничего записано не будет). Другие системы (в том числе все, которые написаны на базе КуМир/СУБД) не вмешиваются в процесс ввода, а все свои замечания выдают после того, как поле заполнено.

Упражнение:

9. Приведите причины, по которым в базах данных используются числовые поля и поля–даты.

2.3. Создание и заполнение простой базы данных

После того, как мы обсудили теорию ввода информации в базу данных, настало время перейти к практике. Давайте создадим и заполним базу данных, в которой будут храниться данные о ваших знакомых. Про каждого человека будем указывать следующие сведения:

A20 – фамилия,
A40 – имя и отчество,
N – рост (в *см*),
D – дата рождения,
A10 – номер телефона.

Слева указан тип для каждого поля. Обратите внимание на то, что для хранения номера телефона выбрано не числовое поле, а текстовое. Это связано с тем, что номера телефонов, хотя и записываются цифрами, на самом деле — не вполне числа (например, их совершенно бессмысленно складывать или выяснять, что больше).

Выберите в гипертексте на странице «Телефонная книга» пункт «Ввод новой информации» и нажмите клавишу Enter.

Если вы все сделаете правильно, на экране у вас появится бланк (экранная форма, в которую можно вводить данные). В начале работы курсор устанавливается в первое поле. Как и в текстовом редакторе, при заполнении поля можно вводить текст и пользоваться клавишами перемещения курсора ← и →*), а также командами вставки и удаления символов Ins, Del и Bs.

После того, как вы решите, что поле заполнено, перейти в следующее можно, нажав Enter или Tab (кстати, в любой момент вы можете вернуться и к предыдущему полю — для этого служит команда Shift Tab).

Попробуйте ввести некорректные данные в поля «рост» и «дата рождения». Удастся ли вам добиться того, чтобы

*) Однако клавиши ↑ и ↓ не работают — поле имеет высоту всего в одну строчку.

оба эти поля были заполнены некорректно?

Вот, наконец, первая запись подготовлена. Как же перейти к вводу следующей? Давайте отвлечемся на минуту от клавиатуры и посмотрим внимательно на экран. В верхней части экрана вы увидите строку с названием информационной системы (сейчас там написано "Телефонная книга") и режимом работы ("Ввод данных"), а в нижней строке перечислены все специальные команды, которые можно выполнять в данный момент:

F1 – **Подсказка.** При нажатии на эту клавишу на экране появится окно со списком всех команд редактирования; возврат к вводу данных — по нажатию Esc.

F2 – **Запись.** Служит для записи информации в базу данных — именно эта команда нам и нужна.

Esc – **Выход.** Окончание ввода данных.

Нажмите на клавишу F2. Вся информация, которую вы ввели, будет записана в базу данных, и на экране появится сообщение "Введено записей: 1". Экранная форма очистится — все готово для ввода следующей записи.

! Команда выхода Esc, выполнением которой надо закончить сеанс ввода данных, ничего не записывает в базу данных. Поэтому после ввода последней записи надо сначала сохранить информацию, нажав F2, а уже потом давать команду выхода.

2.4. Импорт данных

Иногда бывает удобнее не вводить информацию в базу данных по полям с клавиатуры, а подготовить заранее в текстовом файле. Чаще всего это бывает, когда информация, заносимая в базу данных, вычисляется (или корректируется) при помощи какой-нибудь другой компьютерной программы. Было бы глупо распечатывать данные для того, чтобы потом вводить их вручную. И, конечно, так никто не поступает. Практически во всех информационных системах есть возможность ввода данных из заранее подготовленного текстового файла. Такая операция обычно называется ***импорт данных***.

2.5. Просмотр информации в базе данных

Любая информация в базе данных будет совершенно бесполезной, если ею нельзя воспользоваться. Как минимум, любая информационная система должна давать возможность просматривать введенные данные. Мы уже работали с просмотром базы данных музыкальных записей, но пока еще не обсуждали вопрос, в каком порядке показываются записи.

Система управления базами данных, которая умеет показывать записи только в том же порядке, в котором эти записи вводились, не очень полезна. При составлении каталога песен на кассетах эту систему еще можно было бы применить, но с телефонной книгой уже работать было бы нельзя. Поэтому все реальные СУБД позволяют задавать порядок для просмотра записей.

Лексикографический порядок

Всем хорошо известно, как сравнивать числа. Рассмотренная ранее музыкальная база данных показывала записи в порядке возрастания номеров песен (и в данном случае это было вполне естественно). Однако не имеет большого смысла показывать телефонную книгу в порядке возрастания номеров телефонов или роста записанных там людей.

К счастью, текстовые поля тоже можно сравнивать. И в телефонной книге было бы естественным расположить записи «по алфавиту». Порядок слов (или фраз), который в разговорной речи называется «по алфавиту», по-научному называется ***лексикографическим***. Именно такой порядок используется во всех словарях: из двух слов раньше идет слово, у которого первая буква «меньше» (т.е. стоит в алфавите раньше); если первые несколько букв двух слов одинаковы, то сравнение производится по первой различающейся букве.

При расстановке слов в словаре, как правило, не учитываются пробелы в словосочетаниях. Например, в Большом англо–русском словаре используется такой порядок слов: *go*, *goal*, *go back*, *good*... При работе на компьютере пробел рассматривается как полноправный символ (при сравнении считается, что он «меньше» любой буквы), поэтому в базе данных слова будут расположены в другом порядке: *go*, *go back*, *goal*, *good*... (Впрочем, многие считают такой порядок даже более удобным.)

Сравнивать даты совсем просто — одна дата считается «меньше» другой, если относится к более раннему времени. Заметим, что при лексикографическом сравнении текстовых записей дат получится совсем другой порядок: в нашей стране принято вначале писать день, а потом месяц, поэтому "31/01" (31 января) окажется после "01/02" (1 февраля).

Ключевые поля

Если мы можем сравнивать содержимое полей каждого типа, уже нетрудно задать порядок в базе данных. В реляционных базах данных для задания порядка следования записей используются ***ключевые поля*** (как правило, в роли ключевых используется первое поле или несколько первых полей записи). С точки зрения пользователя, записи располагаются в таком порядке, что содержимое ключевых полей возрастает от начала базы к концу.

Если ключевых полей несколько, то при определении порядка записей используется тот же метод, что и при лексикографическом сравнении текстов. Вначале записи упорядочиваются по содержимому первого ключевого поля; записи с одинаковым содержимым этого поля расставляются по второму ключевому полю и т.д.

Как правило, информационные системы не позволяют записать в базу данных несколько записей, у которых значения всех ключевых полей совпадают.

В базе данных музыкальных записей ключевыми были два первых поля (номер диска и номер песни на диске). Для телефонной книги естественно выбрать в качестве ключевых также два первых поля (содержащих фамилию, имя и отчество).

Выберите в гипертексте на странице «Телефонная книга» пункт «Просмотр данных» и найдите информацию, которую вы ввели.

Поиск по ключу

Если в базе данных всего несколько десятков записей, то найти нужную можно и вручную. При нескольких сотнях это уже утомительно. А если записей 20 000 (как в рассмотренном выше примере депозитария), то поиск записи может занять слишком много времени, даже если записи упорядочены. Поэтому во всех информационных системах есть команда поиска данных по ключу. В телефонной книге для этой цели служит команда:

F3 – **Переход.** Найти запись с указанным значением ключевых полей и перейти на нее.

При нажатии на клавишу F3 вначале запрашивается образец поиска. Поскольку ключ в нашем случае состоит из двух полей, надо ввести две строки (и в конце ввода каждой строки нажать на Enter). Если запись с указанным значением ключевых полей

содержится в базе данных, то она сразу после этого появится на экране.

Выясните, что произойдет, если записи с указанным значением ключевых полей нет в базе данных. Можно ли использовать такое поведение информационной системы для упрощения поиска по ключу?

2.6. Поиск информации в базе данных

Далеко не всегда при просмотре информации в базе данных вы можете сказать заранее, какие записи вам нужны. Например, вы хотите найти телефон знакомого, фамилию которого не помните (а помните только имя); или хотите посмотреть, какие записи группы *ABBA* есть у вас на кассетах. Название исполнителя вообще не входит в число ключевых записей музыкальной базы, а имя хотя и входит в ключ телефонной книги, но не первым полем. Так что поиск по ключу в данных случаях неприменим.

При работе без компьютера в подобных ситуациях приходится последовательно просматривать все записи. Если данные записаны в базу данных, то просматривать все записи все равно надо, но теперь эту работу можно поручить компьютеру.

Поиск по значению

Рассмотрим сначала простейший способ поиска информации в базе данных — поиск по значению определенного поля. В информационных системах, построенных на базе КуМир/СУБД, для поиска обычно используется команда

F4 – **Поиск.** Найти и показать все записи с указанным значением поля.

При нажатии на клавишу F4 вначале запрашивается, по какому полю требуется произвести поиск. На экране появляется список полей базы данных, из которого вам нужно выбрать одно поле (перемещая курсор клавишами ↑ и ↓) и нажать клавишу Enter. После этого будет запрошен образец поиска (его ввод также надо завершить нажатием на Enter). В случае успешного поиска на экране появится список записей, которые в указанном вами поле содержат указанное значение.

Выберите на странице «База данных музыкальных записей» пункт «Поиск информации» и найдите все песни группы "ABBA". Попробуйте найти в телефонной книге

человека по имени Корнелий (для этого надо запустить пункт «Поиск информации» со страницы «Телефонная книга»). Выясните, что произойдет, если записи с указанным значением полей нет в базе данных.

Нажатие на клавишу Esc в режиме просмотра найденных записей приведет к переходу в режим обычного просмотра базы данных.

Задание шаблона поиска

Поиск по значению, конечно, очень полезная операция и сильно облегчает получение нужной информации. Однако и ее возможностей часто бывает недостаточно. Например, вы помните, что в названии песни есть слово "California", но точное название и исполнителя забыли. Не расстраивайтесь — в команде поиска по текстовому полю не обязательно указывать точное значение, можно задать ***шаблон поиска***.

Для того, чтобы найти все песни, в названии которых встречается слово "California", надо при задании образца поиска по названию указать

```
~California~
```

Символ «~» означает «любая последовательность символов». При поиске по такому шаблону будут найдены все записи, у которых в названии вначале стоят любые символы, потом текст "California", а потом снова любые символы (а это как раз то, что нам нужно). Заметим, что:

- последовательность символов поля, которая соответствует символу «~» в шаблоне, может быть пустой: при поиске по указанному выше шаблону будет, в частности, найдена песня *Hotel California* группы *Eagles*, в названии которой искомое слово стоит в самом конце (и после него ничего нет);
- не обязательно в шаблоне задавать целое слово, можно указать его часть (или, наоборот, несколько слов): текст при поиске рассматривается просто как строка символов, в которой пробел ничем не отличается от буквы.

Попробуйте найти в базе данных музыкальных записей все песни, в название которых входит слово "light" (используя символ «~» в шаблоне).

Как быть, если полученный результат вас не удовлетворяет? Может быть, попробовать добавить пробелы до и после слова *light*? Увы, это не поможет — ведь искомое слово может стоять в

названии самым первым (и тогда перед ним нет ни одного пробела) или последним (тогда нет пробелов после него), или может заканчиваться не пробелом, а, например, запятой. На помощь приходит другой специальный символ — «'» (обратная кавычка), который означает начало или конец слова.

Например, при поиске по шаблону «~'light~» будут выбраны названия *Travelling Light* и *10,000 Lightyears*, но уже не будет выбрано *I'm Going Slightly Mad*. Если указать «'» еще и в конце (т.е. искать по шаблону «~'light'~»), то будет найдено только первое.

Выполните на компьютере приведенные ниже упражнения (запустив пункт «Поиск информации» со страницы «Телефонная книга»).

Упражнения:

10. Найдите в телефонной книге все фамилии, которые:
    а) содержат суффикс *-овск-*;
    б) оканчиваются на *-ева*;
    в) начинаются на букву *С* и кончаются на *-а*;
    г) содержат буквы *л* и *м* (в указанном порядке);
    д) содержат по крайней мере три буквы *е*.
11. Найдите в телефонной книге всех людей, у которых имя или отчество начинается на букву *И*.
12. Вы помните, что у вашей знакомой в середине номера стоит –55–, но забыли ее фамилию. Как проще всего найти сведения о ней в базе данных?
13. Можно ли при помощи поиска по образцу найти всех людей, рост которых превышает 2 метра?

2.7. Редактирование базы данных

Информационная система, которая позволяет только вносить информацию в базу данных и просматривать ее, не очень интересна. В самом деле, введенные сведения могут со временем устареть (например, ваш знакомый переехал на новое место, и у него поменялся телефон). Вы можете захотеть добавить в базу данных новые записи или удалить старые (например, ошибочно введенные или просто устаревшие).

Процесс внесения изменений в базу данных называется ***редактированием базы данных*** (аналогично тому, как внесение изменений в текст называется *редактированием текста*). Любое

изменение базы данных сводится к последовательному применению одной из трех операций: добавлению, удалению и редактированию записи.

Добавление информации

Собственно говоря, добавление записей почти ничем не отличается от ввода информации. Для того, чтобы вставить в базу данных сведения о новом человеке, надо в режиме просмотра данных нажать Shift Ins. На экране появится бланк, который надо заполнить (точно так же, как это делалось при первоначальном вводе информации), а после этого нажать клавишу F2. Обратите внимание на то, что после ввода одной записи система не предлагает вам сразу же ввести следующую (как это было раньше). Если вы хотите ввести несколько записей, то вам нужно нажимать Shift Ins перед вводом каждой (или воспользоваться режимом ввода информации, рассмотренным ранее).

Выберите в гипертексте на странице «Телефонная книга» пункт «Просмотр и редактирование данных» и добавьте в базу данных сведения о вашем любимом литературном персонаже.

Упражнение:

14. При добавлении записи система сама определяет ее положение. Как можно вставить запись в какое-нибудь другое место?

Удаление информации

Теперь рассмотрим, как удалять из базы данных ненужную информацию. Это очень просто — достаточно подвести курсор к той записи, которую вы хотите удалить, и нажать Shift Del. Удаление — опасная операция, ведь в информационных системах, как правило, нет команды откатки (которая есть в большинстве текстовых редакторов). Поэтому перед тем, как удалить запись, система задаст вопрос: "Удалить запись? [Да/Нет]". В ответ надо нажать либо клавишу Д (после чего запись удалится из базы данных), либо Н (если вы нажали Shift Del по ошибке и вовсе не хотите ничего удалять).

Удалите из базы данных «Телефонная книга» те записи, которые, по вашему мнению, там лишние.

Изменение информации

Что делать, если в базе данных потребовалось изменить только одно поле в некоторой записи (например, только номер телефона)? В принципе, можно воспользоваться двумя рассмотренными ранее командами — сначала удалить старую запись, а потом ввести новую. Однако в этом нет нужды: во всех информационных системах есть команды редактирования записей.

Для того, чтобы отредактировать запись, надо ее выбрать (т.е. поставить на нее курсор) и нажать клавишу Enter. На экране появится бланк, полностью аналогичный тому, с которым вы работали при добавлении информации. Но теперь все его поля уже заполнены. Выберите нужное вам поле, используя команды Tab («перейти к следующему полю») и Shift Tab («перейти к предыдущему полю»), и измените записанные в нем данные. Если вы хотите изменить несколько полей в одной записи — повторите эту операцию несколько раз. Как и при добавлении информации, закончить изменения надо нажатием на клавишу F2.

Исправьте в базе данных «Телефонная книга» ошибки, которые вы заметили при предыдущих просмотрах данных.

2.8. Составление отчетов

Составление отчетов — довольно сложная операция. Надо четко и ясно сообщить компьютеру, что вы хотите от него получить. В реальных СУБД для описания отчетов часто даже используется специальный язык (он так и называется — *язык составления отчетов*). Однако мы сейчас не будем изучать никакой язык, а рассмотрим механизм составления отчетов, принятый в системе КуМир/СУБД.

Выбор данных для отчета

Первым делом при составлении отчета надо решить, какие данные дожны быть включены в отчет и в каком порядке (имеется в виду как порядок записей в отчете, так и порядок записи отдельных полей каждой записи). Конечно, в отчет нельзя включить больше информации, чем есть в базе данных, но не обязательно включать все, что есть.

При выборе пункта *Составление отчета* на экране сначала появляется табличка, в которой перечислены названия всех полей. Вы можете перемещать курсор по этой табличке (как обычно, используя клавиши ↑ и ↓), а также выполнять следующие

операции:

Shift Del – Удалить из отчета данное поле
Caps ↑ – Переставить данное поле вверх
Caps ↓ – Переставить данное поле вниз

! – Задать поле, по которому будет упорядочиваться отчет
F2 – Закончить редактирование описания отчета

Например, при работе с базой данных музыкальных записей вы получите такую табличку:*)

```
* Nкассеты    5
* Nпесни      5
  Исполнитель 20
  Название    32
  Время       5
```

Предположим, вы хотите составить отчет, который включал бы имя (название) исполнителя, название песни, ее продолжительность и номер кассеты, на которой песня записана (именно в таком порядке, как указано), причем записи должны быть упорядочены по исполнителям. Этот отчет может быть задан следующей табличкой:

```
! Исполнитель 20
  Название    32
  Время       5
* Nкассеты    5
```

Выберите на странице «База данных музыкальных записей» пункт «Составление отчетов» и попробуйте составить описанный выше отчет. Какие клавиши и в какой последовательности надо нажать, чтобы добиться желаемого результата?

Оформление отчета

Если вы выполнили задание из предыдущего раздела, то после нажатия на клавишу F2 у вас на экране появилось сообщение "Слишком длинная строка" (не волнуйтесь, так оно и было задумано, вы все сделали правильно).

*) Звездочками в этой табличке помечены ключевые записи; они будут определять порядок следования записей в отчете, если не указать явно другого. Что означают числа справа от названия поля, написано в следующем разделе.

Дело в том, что ширина страницы отчета в системе КуМир/СУБД ограничена 64 символами (чтобы отчет можно было напечатать на обычном принтере). В нашем примере суммарная ширина полей равна 20 + 32 + 5 + 5 = 62 (кстати, числа в табличке справа от имени поля как раз и означают ширину поля); еще 3 позиции занимают пробелы между полями. Итого получается 62 + 3 = 65 позиций — на одну больше допустимого.

В данном примере мы легко можем уменьшить длину строки на один символ. Например, заведомо можно сузить поле с номером кассеты (вряд ли в нашей базе наберется больше 9999 кассет). Для выполнения этой операции нужно подвести курсор к цифре 5 в последней строке и исправить ее на 4.

Выполните описанные выше действия и нажмите клавишу F2. Просмотрев полученный отчет, попробуйте составить еще один, включив в него всю информацию о записях.

Шпаргалка. При просмотре отчета можно пользоваться клавишами ↑ и ↓, которые работают так же, как обычно. Для выхода надо нажать клавишу Esc.

Наверняка, вы справились с последним заданием и нашли, какие поля можно сократить. Тогда выполните задание потруднее: надо составить отчет по базе данных «Телефонная книга». При выборе пункта *Составление отчетов* на странице *Телефонная книга* вам будет предложена такая табличка:

```
* Фамилия     20
* Имя         40
  Рост        12
  Дата рождения 10
  Телефон     10
```

Конечно, для вывода роста вполне хватило бы трех позиций, но компьютер очень мало знает о людях и предлагает отвести для поля Рост 12 позиций, как и для любого поля типа N (длинное числовое поле).

Даже после уменьшения размеров полей Рост и Телефон (в котором одна позиция была добавлена «про запас») остается 20 + 40 + 3 + 10 + 9 + 4 = 86 позиций. Сокращать поля Фамилия и Имя не очень хорошо (конечно, у большинства записей в нашей базе данных имя значительно короче 40 символов, но ведь не зря под него отвели столько места).

К счастью, выход есть — КуМир/СУБД умеет разбивать длинный текст на несколько строк. Для включения режима разбиения строк нажмите серую клавишу * (справа, на дополнительной клавиатуре). Прямоугольник под словом "Режим" будет разрезан на две части, и теперь можно спокойно сокращать размеры полей,

не боясь, что часть информации пропадет:

```
* Фамилия        20
* Имя            18
  Рост           3
  Дата рождения  10
  Телефон        9
```

Режим

Полученный по такому описанию отчет будет выглядеть примерно так:

```
Булычев     Кир (Игорь Всеволодович)   175 15/04/1936 987-65-43
Грибкова    Илья                       154 25/06/1965 145-55-67
Селезнева   Алиса                      155 17/11/2065 888-01-02
```

Обратите внимание: при переносе длинных строк компьютер всегда старается разрезать строку между словами.

Полученный отчет выглядит не очень хорошо — последние три поля расположены слишком тесно. Система подготовки отчетов КуМир/СУБД позволяет это исправить — после любого поля можно добавить дополнительные пробелы. Например, для того, чтобы после колонки с содержимым поля Дата рождения всегда вставлялось по три пробела вместо одного, надо написать

```
Дата рождения 10+2
```

(т.е. два дополнительных пробела).

Закончите составление отчета: переставьте колонку с именем в самое начало, поменяйте местами поля Телефон *и* Рост, *а также добавьте пробелы для разделения трех последних полей.*

2.9. Простейшие логические выражения

Во многих случаях, когда в базе данных содержится много информации, найти нужные данные не так просто, и рассмотренных выше команд поиска может не хватить. Все информационные системы позволяют при поиске и при составлении отчетов использовать сложные условия. Мы разберем их в пункте 2.11, а пока остановимся на простейших условиях (или ***логических выражениях***), из которых можно будет впоследствии «собрать» сложные.

Первое, с чем надо разобраться, — как при составлении условий ссылаться на значение полей (например, если мы хотим сравнить значение поля с константой, нам надо указать, какое поле имеется в виду). Впрочем, это не сложно: каждое поле имеет имя, и для ссылки на любое поле надо просто указать это имя

(полностью сохраняя его написание, ничего не сокращая и не заменяя большие буквы на маленькие). Вот некоторые примеры условий (их смысл будет рассмотрен далее):

```
Имя = "Алиса"
Рост <= 150
день(Дата рождения)=17 и месяц(Дата рождения)=11
```

Сравнения

Важнейший элемент условий — это сравнения. В одном сравнении могут быть указаны два или более числовых или литерных выражения (или выражения типа «дата»), разделенные одним из следующих знаков сравнения:

```
<   меньше
=   равно
>   больше
<=  меньше или равно (≤)
>=  больше или равно (≥)
<>  не равно (≠)
```

Система КуМир/СУБД, в отличие от большинства других СУБД, позволяет использовать в одном сравнении две операции (как это принято в математике), например:

```
0 < t <= 1.5  ⇔  0 < t и t <= 1.5
```

Конечно же, при помощи условий можно задавать и поиск по шаблону. Операция сравнения строки (значения некоторого поля) с шаблоном записывается так:

`?(`*строка*`,`*шаблон*`)` – истинно, если указанная строка подходит под шаблон

Например, для решения упражнения 10(б) из пункта 2.6 можно использовать условие

```
?(Фамилия,"~ева").
```

Следующие три функции предназначены для работы с датами. Аргументом их должно быть имя поля типа «дата», а значением будет целое число (которое, в частности, можно использовать в арифметическом сравнении):

`день(`*дата*`)` – день месяца указанной даты
`месяц(`*дата*`)` – номер месяца (1 – январь, 2 – февраль, 3 – март...)
`год(`*дата*`)` – номер года

Обратное преобразование выполняется функцией

Дата(*день*,*месяц*,*год*) – получение даты по номерам дня, месяца и года,

которая имеет три целых аргумента и значение типа «дата».

Найдите в базе данных «Телефонная книга» сведения о всех людях, чей рост больше двух метров. Как найти всех людей с ростом около 172 см?

2.10. Логические операции

Для составления сложных условий из простых обычно используются ***логические операции***. Чаще всего используются операции и, или и не. Рассмотрим их по порядку.

Операция "и"

Очень часто запись, которую нужно найти, должна удовлетворять сразу нескольким простым условиям. Например, в телефонной книге требуется найти всех людей по имени Коля, у которых фамилия начинается с буквы *С*. В этом случае надо использовать логическую операцию и, которая действует следующим образом:

А и Б – соблюдается, если соблюдается как А, так и Б, в противном случае не соблюдается

(буквами А и Б обозначены другие условия, например сравнения). Значение логического выражения "А и Б" в зависимости от значений А и Б можно описать следующей таблицей:

А	Б	А и Б
да	да	да
нет	да	нет
да	нет	нет
нет	нет	нет

Например, для решения поставленной выше задачи можно использовать условие

```
Имя = "Коля" и ?(Фамилия,"С~")
```

Упражнения:

15. Составьте условие для поиска всех высоких людей (выше 180 см), номер телефона которых начинается с цифры 1.
16. Как найти в базе данных человека с инициалами К.И.У.?

Операция "или"

Эта операция может оказаться полезной, если искомым значением является одно из нескольких возможных:

А или Б – соблюдается, если соблюдается либо А, либо Б, либо они оба, в противном случае не соблюдается.

Таким образом, значение логического выражения "А или Б" описывается таблицей:

А	Б	А или Б
да	да	да
нет	да	да
да	нет	да
нет	нет	нет

Упражнение:

17. Составьте условие для нахождения всех людей с именами Коля или Фима.

Операция "не"

Часто при поиске легче сформулировать, какие записи **не** нужны. В таком случае полезной может оказаться операция:

не А – соблюдается, если А не соблюдается, и наоборот.

Значение логического выражения "не А" описывается таблицей:

А	не А
да	нет
нет	да

Упражнение:

18. Как найти в телефонной книге все фамилии, в которых нет буквы *а*?

2.11. Составные условия

Точно так же, как с использованием арифметических операций составляются арифметические выражения, можно, используя логические операции, составлять ***логические выражения*** (или *составные условия*). Например, для поиска всех людей по

имени Коля, у которых фамилия начинается с буквы *С*, а номер телефона — с цифры 1, можно использовать условие

```
Имя="Коля" и ?(Фамилия,"С~") и ?(Телефон,"1~")
```

Упражнение:

19. Как найти все номера телефонов, у которых две средние цифры одинаковы (например, XXX-22-XX или XXX-55-XX)?

Порядок действий при проверке условия

Операцию **и** часто называют *логическим умножением*, а операцию **или** — *логическим сложением*. Как и арифметическое, логическое умножение имеет большее старшинство, чем логическое сложение, и при отсутствии скобок выполняется раньше логического сложения. Например:

```
A и B или C и D     означает (A и B) или (C и D)
A или B и C или D   означает A или (B и C) или D
```

Операция **не** выполняется раньше, чем операции **и**, **или**, то есть

```
не A и B            означает (не A) и B
```

Если в выражении идут подряд несколько одноименных логических операций, то они выполняются подряд слева направо. Порядок действий в логическом выражении можно указать явно, расставив подходящим образом скобки.

Упражнения:

20. Составьте условие для нахождения всех людей с именами Коля или Фима, фамилии которых:
    а) кончаются на букву *в*;
    б) не содержат буквы *а*;
    в) начинаются на букву *Н* или *К*.
21. Как найти все номера телефонов, в которых нет цифр 0 и 1?

§ 3. ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

3.1. Зачем нужно много таблиц?

При проектировании баз данных часто возникают случаи, когда для представления информации одной таблицы не хватает. Даже в сравнительно простых базах данных может оказаться удобнее располагать данные не в одной таблице, а в двух.

Рассмотрим, например, рисунок 3 из пункта 2.1. Три последних поля на изображенной там форме относятся на самом деле

не к песне, а к диску, на котором эта песня записана. И, если в базе данных музыкальных песен содержится по несколько песен с каждого диска, имеет смысл завести отдельную таблицу для хранения общей информации.

3.2. Электронный депозитарий

В самом начале книги (пункт 1.2) уже упоминался электронный депозитарий — информационная система для хранения сведений об акционерах. Конечно же, для всей перечисленной информации не хватит одной таблицы.

Например, информацию, хранящуюся в электронном депозитарии, можно структурировать следующим образом (в реальных депозитариях указанные ниже таблицы содержат гораздо больше информации):

Таблица акционеров. Каждая запись содержит поля
- код акционера;
- фамилия;
- имя;
- отчество;
- адрес;
- телефон;
- серия и номер паспорта.

Таблица акций у акционеров содержит поля:
- код акционера;
- код акции;
- количество.

Таблица сделок с акциями содержит поля:
- номер сделки;
- дата сделки;
- код акции;
- код продавца;
- код покупателя;
- количество.

Таблица эмиссий содержит поля:
- название акции;
- код акции;
- название эмитента;
- адрес эмитента;
- уставной капитал;
- общее число акций.

Конечно, наша модель сильно упрощена по сравнению с тем, что используется на практике. На рис. 4 приведен фрагмент так называемой *объектной модели* реальной системы. Вы можете сами оценить, насколько сложна эта система (а ведь на рисунке показаны еще не все детали).

Связь между таблицами

Отметим, что некоторые поля в разных таблицах имеют одинаковые названия. Это не случайно — по этим полям осуществляется связь между таблицами. Например, чтобы узнать, кто про-

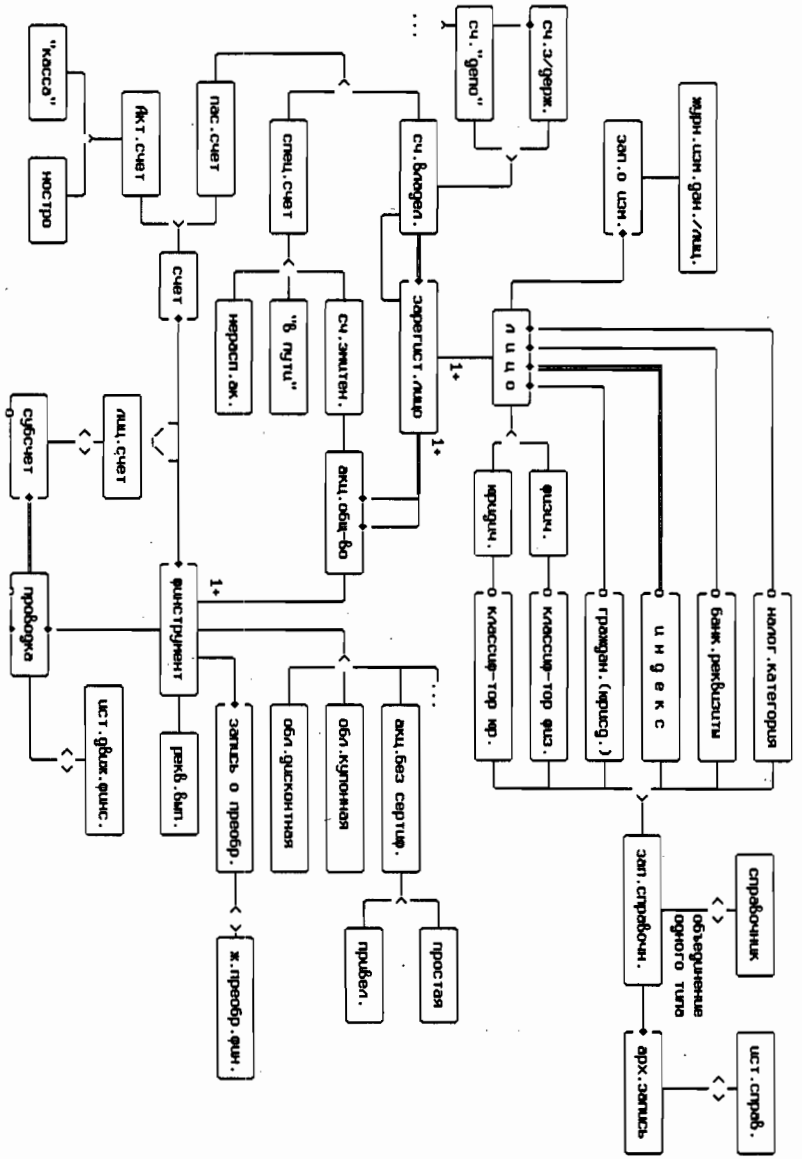


Рис. 4.

давал акции в сделке номер 125, надо найти в таблице сделок с акциями запись с номером сделки 125. Допустим, что в этой записи записан код продавца 2. Это означает, что продавцом в данной сделке выступал человек, записанный в таблице акционеров с кодом акционера, равным двум.

3.3. Этапы проектирования базы данных

Выбор ключевых полей

В процессе проектирования базы данных приходится учитывать следующие два ограничения на выбор ключевых полей:

- многие СУБД (в том числе Paradox и КуМир/СУБД) разрешают использовать в качестве ключевых только первые подряд идущие поля (нельзя, например, использовать в качестве ключа только 1-е и 3-е поля);
- в одной таблице базы данных не может быть двух записей, у которых значения всех ключевых полей совпадают.

Первое ограничение не очень существенно (поля в записи всегда можно переставить нужным образом, поскольку их порядок в экранной форме не обязан совпадать с порядком в базе данных на диске). Второе ограничение может доставить много хлопот (например, в нашу телефонную книгу нельзя записать данные о двух полных тезках). В некоторых случаях в базу данных приходится вводить дополнительные поля, не несущие никакой смысловой нагрузки.

Упражнения:

22. Спроектируйте базу данных для хранения информации из классного журнала.
23. Как можно расширить базу данных из предыдущего упражнения, чтобы в ней можно было хранить данные не по одному классу, а по всей школе?

Глава 2. Электронные таблицы

Что такое *таблица*? Можно сказать, что это прямоугольник, расчерченный на клетки. Для чего нужны таблицы? Для того, чтобы записывать в них информацию: числа, имена, даты.

Очень часто табличная форма задания информации оказывается удобнее текстовой. Сравните, например:

По математике: за первую четверть – 4, за вторую – 5, за третью – 4, за четвертую – 4, за год – 4. По русскому языку: за первую четверть – 3, за вторую – 4, за третью – 4, за четвертую – 4, за год – 4 ...

Предмет	Четверти				Год
	1	2	3	4	
Математика	4	5	4	4	4
Русский яз.	3	4	4	4	4
...					

Очевидно, что в таблице информация легче читается и выглядит красивее, чем в тексте.

А что такое *электронные таблицы*? Это таблицы, в которых можно запрограммировать содержимое каждой клетки. Что значит «запрограммировать клетку»? Когда создают термометр, то предписывают ему (программируют его) показывать определенное число — температуру (по Цельсию, по Фаренгейту или по Реомюру). Когда создают амперметр, то предписывают ему показывать определенное число — силу тока. А когда создают электронную таблицу, то предписывают ее клеткам показывать определенные числа — те, которые нужны пользователю. Процесс создания электронной таблицы обычно состоит в том, что сначала описывается таблица, а затем программируются ее клетки.

Систему, которая позволяет создавать и использовать электронные таблицы, часто также называют «электронной таблицей». В мире существует много таких систем (например, SuperCalc, Lotus 1-2-3 и др.). Они отличаются друг от друга способами описания или программирования таблиц.

Мы будем изучать электронные таблицы под названием «МикроКальк». МикроКальк разработан одним из авторов этой книги на механико-математическом факультете МГУ.

МикроКальк отличается от других электронных таблиц тем, что он встроен в редактор текстов, который называется «МикроМир» и тоже разработан на мехмате МГУ. Напомним, что редактор текстов — это компьютерная система, которая позволяет

набирать тексты наподобие печатной машинки. Но, в отличие от печатной машинки, он позволяет изменять набранный текст и может напечатать его сколько угодно раз.

Встроен МикроКальк в текстовый редактор так, как шкафы встраиваются в стену: стена используется как стена, но при желании можно открыть в ней дверцу, а там — шкаф. Так же и с МикроКальком: можно пользоваться текстовым редактором, набирать текст, а при желании — использовать электронные таблицы.

§ 4. РЕДАКТОР «МИКРОМИР»

4.1. Редактирование в МикроМире

Вы уже знакомы с редактором текстов «Микрон»? МикроМир на него очень похож, а отличается только тем, что в нем некоторые команды вызываются другими клавишами и есть еще дополнительные команды.

Посмотрите, какие команды Микрона и МикроМира вызываются одинаковыми клавишами:

Команда	Клавиши
Перемещение курсора	
На страницу вверх	PageUp
На страницу вниз	PageDown
Быстрое движение курсора вправо	Tab
В начало следующей строки	Enter
Вставки и удаления	
Удалить символ	Del
Удалить строку	Ctrl Del

Команда	Клавиши
Вставки и удаления	
Вставить пробел	Ins
Вставить пустую строку	Ctrl Ins
Стереть символ слева от курсора	Bs
Стереть слово слева от курсора	Shift Bs
Работа с алфавитом	
Постоянное включение рус/лат	Caps Alt
Пост. включение прописных/строчных	Caps Shift
Временное включение рус/лат	Alt *буква*
Врем. включение прописных/строчных	Shift *буква*
Дополнительные команды	
Откатка	Ctrl Bs
Накатка	Ctrl Enter

А вот эти команды вызываются разными клавишами:

Команда	Микрон	МикроМир
Перемещение курсора		
В конец строки	Shift →	Ctrl →
В начало строки	Shift ←	Ctrl ←
Вставки и удаления		
Стереть слово справа от курсора	Shift Enter	+
Стереть конец строки	Ctrl End	Ctrl D
Перемещения текста		
Разрезать строку	Ctrl Ins	5 Enter
Склеить две строки	Ctrl Del	5 Bs
Центрировать строку	Ctrl пробел	Ctrl T
Запомнить символ	F2	F1
Вспомнить символ	Shift F2	F2
Перемещения текста		
Запомнить строку	F3	F3
Вспомнить строку	Shift F3	F4

Команда	Микрон	МикроМир
Прочие команды		
Поиск	F4	*
Изменение шаблона поиска	Shift F4	5 *
Форматировать абзац	F5	F9
Изменение шаблона форматирования	Shift F5	5 F9
	Shift F5	5 F9
Выход из программы	Ctrl Break	Esc ↑

Дополнительные команды МикроМира можно узнать из шпаргалки. Команда вызова шпаргалки на экран (Ctrl ?) всегда изображается в редакторе после конца текста в строке

```
[* конец текста *]                                шпаргалка - Ctrl+?
```

4.2. Файловые оглавления

Рассмотрим вначале лишь одну дополнительную возможность МикроМира — оглавления.

При вызове МикроМира (командой MIM в MS DOS) вы сначала попадете в оглавление (файл MIM.DIR), которое выглядит вот так:

```
(1:1)      рус зам     1 таб код:49 -                    MIM.DIR
 1. Редактирование в "МикроМире"                    63 u01.txt
 2. Рисование таблиц                                63 u02.txt
 3. Изменение рисунка таблиц                        63 u03.txt
 4. Суммирование строк                              63 u04.txt
 5. Суммирование граф                               63 u05.txt
 6. Вычисление граф таблицы по стандартным формулам 63 u06.txt
 7. Математические функции                          63 u07.txt
 8. Вычисления по форме                             63 u08.txt
 9. Вычисления в зависимости от условий             63 u09.txt
10. Управление форматом чисел и режимами вычисления 63 u10.txt
11. Построение гистограмм                           63 u11.txt
12. Вычисление формул                               63 u12.txt
13. Калькулятор                                     63 u13.txt
Стандартные формулы                                 63 calc.mim
[* конец текста *]                          шпаргалка - Ctrl+?
```

Верхняя строка называется ***статусной строкой***. В ней содержится информация о состоянии МикроКалька. Выключить или включить статусную строку можно командой Ctrl S.

Для того, чтобы попасть в какой-либо текст, указанный в оглавлении, нужно поставить курсор на нужную строку и выпол-

нить команду «внутрь» (клавиши Esc ↓). Вернуться из текста в оглавление можно командой «наружу» (клавиши Esc ↑). Этой же командой можно выйти из оглавления и, тем самым, из МикроМира.

Войдите МикроМиром в текст 1. **Редактирование в МикроМире** *и исправьте ошибки.*

4.3. Рисование таблиц

Нажмите в МикроМире клавишу Caps и, не отпуская ее, двигайте курсор клавишами-стрелками (←, ↑, ↓, →). Вы увидите, что курсор оставляет за собой след — одинарную или двойную линию. Этой линией можно рисовать таблицы.

Замечание. С линиями можно обращаться как с обычными символами: удалять, запоминать и т.д. При сдвиге символов строки линии тоже сдвигаются.

Давайте нарисуем прямоугольник. Нажмите четыре раза Caps →, один раз Caps ↓, четыре раза Caps ← и один раз Caps ↑. Обратите внимание на то, что при изменении направления движения курсор оставляет за собой не прямую линию, а угол. В итоге получилось нечто, похожее на прямоугольник:

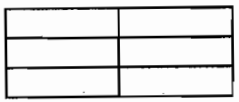
Для того, чтобы сформировался левый верхний угол, еще раз смените направление движения курсора, например нажмите Caps →. Прямоугольник готов:

Теперь сделаем из этого прямоугольника таблицу — проведем посередине вертикальную линию.

Поставьте курсор посередине верхней стороны прямоугольника, нажмите Caps ↓ и Caps →. Вот что получилось:

Точно так же можно проводить в таблице и горизонтальные линии.

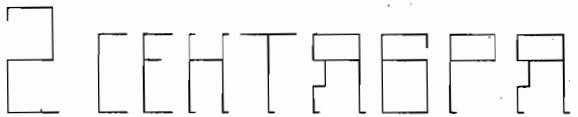
Нарисуйте следующую таблицу, у которой ширина колонок – 12 символов:



4.4. Двойные линии

Как поменять вид линии, которая тянется за курсором? Чтобы сделать линию двойной, выполните команду Ctrl Shift - (клавиша - в этом аккорде берется из основной клавиатуры). Чтобы сделать линию одинарной, выполните команду Ctrl -. При изменении линии будет меняться и значок линии в статусной строке: из '-' он превратится в '=' и наоборот.

Нарисуйте дату своего рождения, например:



4.5. Изменение рисунка таблиц

Нарисованную таблицу в МикроМире очень легко изменить: расширить, сузить, добавить разграфленные строки. Это делается в режиме редактирования таблиц, который включается/выключается командой Ctrl - (в этом случае используется клавиша - с дополнительной клавиатуры, справа от основной; обычно эта клавиша серого цвета). Текущий режим всегда показан в статусной строке: в режиме редактирования таблиц в ней присутствует слово "таб".

В режиме редактирования таблиц строки, вставляемые внутри таблицы, автоматически разграфляются (в нужных местах МикроМир записывает в них вертикальные линии). В этом режиме линии нельзя удалить или сдвинуть как обычные символы.

Это можно сделать только с помощью специального аппарата — блоков. Для расширения или сужения таблицы сначала надо отметить расширяемое или сужаемое место прямоугольным блоком.

Как это сделать? Надо поставить курсор на место левого верхнего угла будущего прямоугольника и нажать клавишу F5 (начать отметку). После этого при перемещениях курсора на экране будет выделен прямоугольник, один угол которого зафиксирован, а другой находится в позиции курсора. Перемещая курсор, добейтесь того, чтобы прямоугольник занимал все расширяемое или сужаемое место.

После отметки блока расширяйте таблицу на один символ нажатием клавиши Ins или сужайте нажатием клавиши Del.

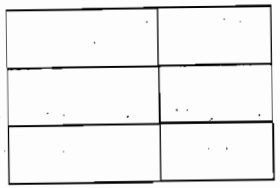
Закончить отметку блока можно командой Ctrl C или нажатием клавиши F5. В последнем случае отмеченный блок запом-

нится, а вспомнить его (записать в текст) можно будет нажатием клавиши F6).

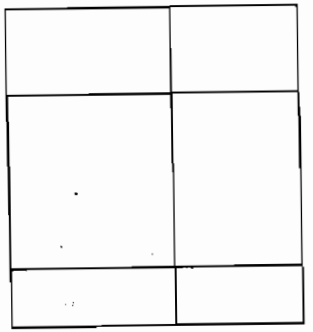
Пример. Сузим правую графу таблицы, нарисованной при выполнении задания из пункта 4.3, на 4 символа:

1. Включаем режим редактирования таблиц: Ctrl - (в статусной строке при этом должно появиться слово "таб" — сокращение от слова *таблица*).
2. Ставим курсор правее пересечения верхней горизонтальной и средней вертикальной линий.
3. Начинаем отметку блока: F5.
4. Доводим курсор до пересечения нижней горизонтальной и правой вертикальной линий.
5. Сужаем таблицу на 4 символа: четыре раза нажимаем Del.
6. Заканчиваем отметку блока: Ctrl C.
7. Выключаем режим редактирования таблиц: два раза нажимаем Ctrl - (после первого нажатия включится режим редактирования таблиц с отступом от линии, а в статусной строке вместо слова "таб" появятся знаки "| |").

Таблица сужена:



Расширьте левую графу таблицы на один символ, правую на два символа и вставьте в таблицу 5 строк так, чтобы получилась изображенная справа таблица.



§ 5. СИСТЕМА «МИКРОКАЛЬК»

МикроКальк, как и все электронные таблицы, позволяет «программировать» таблицы. Разберем простейший способ такого программирования — суммирование строк таблицы.

5.1. Суммирование строк

Заполним таблицу, полученную после выполнения задания предыдущего раздела (рис. 5, слева).

Пусть требуется просуммировать количество уроков по всем дням и результат записать в последнюю строчку таблицы. Можно все это сделать вручную. А можно запрограммировать таблицу так, что это за нас сделает МикроКальк.

Взгляните на рисунок 5. Таблица, изображенная справа, запрограммирована на суммирование чисел 2-й графы и запись суммы в пустую клетку. Как же она запрограммирована? Ведь она отличается от предыдущей таблицы всего одним словом — "Всего"?

В этом слове "Всего" и есть все дело. Если оно записано в первой графе какой-то строки, то во всех остальных графах этой строки МикроКальк запишет сумму чисел из предыдущих граф. Таким образом, слово "Всего", употребленное в первой графе, — это команда программирования таблиц.

Для того, чтобы МикроКальк вычислил таблицу, надо поставить курсор в любое место таблицы и вызвать команду «Вычислить таблицу» (нажать клавишу F12 или, если ее нет на вашей клавиатуре, нажать клавиши Caps Home).

Вот что получилось из таблицы после того, как ее обработал МикроКальк:

День недели	Количество уроков
Понедельник	6
Вторник	5
Среда	5
Четверг	6
Пятница	5
Всего	27

День недели	Количество уроков
Понедельник	6
Вторник	5
Среда	5
Четверг	6
Пятница	5

День недели	Количество уроков
Понедельник	6
Вторник	5
Среда	5
Четверг	6
Пятница	5
Всего	

Рис. 5. Суммирование количества уроков.

Количество уроков в неделю	
День недели	Количество уроков
Понедельник	6
Вторник	5
Среда	5
Четверг	6
Пятница	5
Итого	
Количество самостоятельных занятий в неделю	
Понедельник	2
Вторник	2
Среда	3
Четверг	2
Пятница	2
Итого	
Всего	

Рис. 6.

А теперь составим таблицу посложнее (рис. 6). В этой таблице используется еще одна возможность программирования таблиц: запись промежуточного результата суммирования строк. Если нужно узнать отдельно количество уроков в неделю и отдельно количество часов самостоятельных занятий, то используется команда "Итого". Она суммирует не все числа в графе, а только числа в строках после предыдущей команды "Итого". Заставьте МикроКальк вычислить эту таблицу, и вы увидите сами, как работает команда "Итого".

Иногда МикроКальк отказывается вычислять таблицу. Часто это бывает из-за ошибки в рисунке таблицы. В таких случаях МикроКальк говорит, что рисунок таблицы неверный, и выделяет ошибочное место другим цветом.

Объясните, почему МикроКальк отказывается вычислять приведенные ниже таблицы. Исправьте все ошибки в таблицах.

	1
	2
Всего	

	1
	2
Всево	

5.2. Суммирование граф

В МикроКальке, как и в любой другой электронной таблице, можно суммировать не только строки, но и графы. Посмотрите на следующую таблицу (рис. 7).

В нижней части таблицы расположен специальный отсек, в котором задана формула суммирования. Она предписывает МикроКальку в каждой строке таблицы сложить число из 2-й графы с числом из 3-ей графы и результат суммирования записать в 4-ую графу.

В формулах вместо номеров граф можно использовать и их заголовки. Строка с заголовками должна располагаться непосредственно над 1-й строкой с числами.

В приведенной таблице в строке заголовков некоторые графы пусты. Чтобы использовать в формулах заголовки, следует изменить таблицу так, чтобы в строке заголовков в каждой используемой в вычислениях графе был какой-нибудь текст. В этом случае

День недели	Число уроков	Часы для самостоятельных занятий	Всего часов
Понедельник	5	3	8
Вторник	4	3	7
Среда	5	4	9
Четверг	5	3	8
Пятница	4	2	6
Итого	23	15	38
Г4 = Г2 + Г3			

Рис. 7. Суммирование граф таблицы.

в нижнем отсеке можно написать следующую формулу:

Всего часов = уроков + ных занятий

Слова "ных занятий" выглядят очень загадочно, но МикроКальк все вычисляет правильно (поскольку он не знает русского языка и ему все равно, что написано). Однако нельзя сказать, что таблица выглядит эстетично. Поэтому изменим ее следующим образом:

День недели	В школе	Дома	Всего
Понедельник	5	3	8
Вторник	4	3	7
Среда	5	4	9
Четверг	5	3	8
Пятница	4	2	6
Итого	23	15	38
Всего = в школе + дома			

Рис. 8. Суммирование граф таблицы.

Заголовки стали покороче, а формулы понятнее.

МикроКальк очень не любит ошибки в формулах. Употребите в формуле нашей таблицы неверный заголовок (например, "дом" вместо "дома"), и МикроКальк скажет вам, что формула неверна, а саму формулу выделит другим цветом.

При вычислении следующей таблицы МикроКальк пищит и ничего не вычисляет. Почему? Исправьте таблицу.

День недели	В школе	Дома	Всего
Понедельник	5	3	
Вторник	4	3	
Среда	5	4	
Четверг	5	3	
Пятница	4	2	
Итого			
Всего = вшколе + дома			

5.3. Правила записи формул в МикроКальке

Формулы в МикроКальке записываются в так называемой **линейной** записи — все символы формулы должны быть записаны в одну строку. Для обозначения операции умножения используется символ '*', для операции деления — косая черта '/'. Нельзя опускать знаки операций, например писать "4a" вместо "4*a".

5.4. Вычисление граф таблицы по стандартным формулам

В МикроКальке можно задавать формулы, по которым вычисляются графы, не только в таблице, но и в специальном файле CALC.MIM. В этот файл обычно записывают наиболее часто используемые формулы.

Взгляните на таблицу, в которой вычисляется скорость удара об землю предмета, упавшего с различной высоты:

Номер этажа	Высота	Скорость в м/с	Скорость в км/ч
1	0.00	0.00	0.00
2	3.00	7.75	27.89
3	6.00	10.95	39.44
4	9.00	13.42	48.30
5	12.00	15.49	55.77

Для этой таблицы в файле CALC.MIM заданы формулы:

```
Высота_этажа = 3.00
g = 10
Высота = (номер этажа - 1) * высота_этажа
Скорость в м/с = SQRT(ABS(2.00*g*высота))
Скорость в км/ч = скорость в м/с * 3600 / 1000
```

Файл CALC.MIM представлен в оглавлении строкой "Стандартные формулы".

Пример. Перевычислим таблицу для значения высоты этажа, равного 4 м. Для этого изменим в формулах значение параметра высота_этажа на 4 следующим образом. Находясь в оглавлении, поставим курсор на строку "Стандартные формулы", выполним команду «внутрь» (клавиши Esc ↓). На экране появится содержимое файла CALC.MIM. Поставим курсор на цифру 3 и заменим ее на 4. Затем вернемся в оглавление командой «наружу» (клавиши Esc ↑). Войдем в файл с таблицей (поставив курсор в оглавлении на строку "6. Вычисление граф..." и выполнив команду «внутрь»). На экране появится таблица. Поставим на нее курсор и выполним команду «вычислить таблицу» (клавиши F12 или Caps Home).

Обратите внимание на то, что слова Высота_этажа и g, использованные в формулах, не являются заголовками граф таблицы. Это временные переменные, которые делают формулы более наглядными и легко меняемыми.

Измените в файле CALC.MIM *значение величины ускорения свободного падения на 9.8, а высоту этажа — на 3.5 и перевычислите таблицу.*

5.5. Математические функции

МикроКальк, кроме выполнения 4-х арифметических действий, умеет вычислять значения некоторых математических функций:

SQRT (квадратный корень из числа),
MAX (максимальное из двух чисел),
ABS (абсолютное значение числа — модуль числа),
SIN (синус),
COS (косинус),
TAN (тангенс),
EXP (экспонента, которую вы еще будете изучать в школе),
LOG (логарифм),

Число	Кв.кор.	Модуль	sin	cos	tg	Экспон.	Логар.
-5.00	####.##	5.00	0.96	0.28	3.38	0.01	####.##
-2.00	####.##	2.00	-0.91	-0.42	2.19	0.14	####.##
-1.00	####.##	1.00	-0.84	0.54	-1.56	0.37	####.##
0.00	0.00	0.00	0.00	1.00	0.00	1.00	####.##
1.00	1.00	1.00	0.84	0.54	1.56	2.72	0.00
1.57	1.25	1.57	1.00	0.00	***.**	4.81	0.45
2.00	1.41	2.00	0.91	-0.42	-2.19	7.39	0.69
3.00	1.73	3.00	0.14	-0.99	-0.14	20.09	1.10
5.00	2.24	5.00	-0.96	0.28	-3.38	148.41	1.61
10.00	3.16	10.00	-0.54	-0.84	0.65	****.**	2.30

Рис. 9. Таблица математических функций.

INT (целая часть числа).

Давайте составим таблицу значений этих функций для некоторых чисел (рис. 9). Таблица невелика, но и по ней уже можно судить о некоторых функциях.

Например, квадратный корень упорно не хочет извлекаться из отрицательных чисел. Модуль просто-напросто убирает знак '–' у отрицательных чисел. Синус и косинус наотрез отказываются превышать 1 и быть меньше –1. Тангенс числа 1.57 настолько большое число, что даже не поместилось в колонку. Не потому ли, что 1.57 близко к числу $\pi/2$?

Экспонента для положительных чисел растет все быстрее и быстрее, так что уже для 10 результат не помещается в колонку. А логарифм не берется не только для отрицательных чисел (как и квадратный корень), но и даже для нуля.

Попытайтесь выяснить, отличается ли значение тангенса, вычисленное через функцию TAN, *от значения, вычисленного путем деления* SIN *на* COS. *Для этого составьте табличку с некоторым количеством чисел и заведите в файле* CALC.MIM *дополнительную формулу для нового тангенса (например,* "Мой тангенс = SIN(число)/COS(число)").

5.6. Вычисления граф по форме

Займемся применением электронных таблиц в бухгалтерии.

Обычно бухгалтерия составляет ведомости нескольких видов, или, иначе говоря, ***форм***. У разных форм могут быть графы с одинаковыми названиями (например, "К выдаче") и различными фор-

мулами вычисления (например, "К выдаче = Начислено - Всего", "К выдаче = Начислено + Надбавка - Всего" и т.д.). Это мешает поместить все бухгалтерские формулы в общий файл CALC.MIM. Записывать формулы в отсек внизу таблицы тоже нельзя: бухгалтерская ведомость будет выглядеть очень странно, да и бухгалтеры вряд ли согласятся вписывать эти формулы в каждую ведомость.

Остается один выход — записывать формулы для каждой формы в отдельный файл. Рассмотрим таблицу (рис. 10). В верхнем отсеке таблицы указано имя формы ведомости. Для каждой формы (например, «Поступления по договорам», «Ведомость на стипендии») определен свой набор формул вычисления одних колонок через другие.

Форма=Ведомость на зарплату

	Начислено	Удержано			К выдаче
		Под.налог	Проф.нал	Всего	
1.Иванов И.И.	100-00	0.00	1.00	1.00	99.00
2.Петров П.П.	1'512-32	140.44	15.12	155.56	1'356.76
Итого	1'612.32	140.44	16.12	156.56	1'455.76
3.Сидоров А.	342-00	0.00	3.42	3.42	338.58
4.Сидоров Б.	343-00	0.12	3.43	3.55	339.45
Итого	685.00	0.12	6.85	6.97	678.03
Всего	2'297.32	140.56	22.97	163.53	2'133.79

Рис. 10. Таблица «Ведомость на зарплату»

Например, форме «Ведомость на зарплату» (использованной в приведенной таблице) соответствует следующий набор формул:

```
Под.налог = MAX(Начислено-342,0)*12%
Проф.нал  = начислено/100
Всего     = Под.налог + Проф.нал
К выдаче  = Начислено - Всего
```

Формулы можно посмотреть или отредактировать (поменять или дополнить), если установить курсор на строку "Форма" (в любое место строки) и скомандовать «внутрь» (нажать клавиши Esc ↓). Напомним, что обратный выход «наружу» — это команда Esc ↑.

При реальной работе достаточно указать в таблице имя формы и, если форма ранее не встречалась или изменилась, войти

«внутрь» (Esc ↓) и набрать формулы. Если же форма уже встречалась, то достаточно просто задать ее имя.

Замечание. Обратите внимание, что в некоторых числах в приведенной таблице цифры разделены одинарными кавычками. Это делается для облегчения чтения больших чисел (сравните 1000000000 и 1'000'000'000). При записи чисел в таблицу можно разделять цифры пробелами и одинарными кавычками. А как заставить МикроКальк разделять цифры в числах, написано в разделе 6.1.

Измените в формулах для формы «Ведомость на зарплату» 12% на 13%.

5.7. Вычисления в зависимости от условий

В предыдущем разделе была рассчитана ведомость на зарплату, которая удерживает лишнее со всех сотрудников. Дело в том, что по российскому законодательству налог 12% взимается только с той суммы, на которую зарплата превышает минимальную. А с зарплаты, превышающей некоторую сумму, берется повышенный процент. Поэтому формулы придется изменить:

```
Мин.зарпл.= 20'500
Макс.зарпл.=1'000'000
Налог на макс.зарпл.=(Макс.зарпл.-Мин.зарпл.)*12%
Под.налог =
 при Начислено < Мин.Зарпл.  : 0
 при Начислено < Макс.Зарпл. : (Начислено-Мин.Зарпл.)*12%
 иначе : (Начислено-Макс.зарпл.)*30%+Налог на макс.зарпл.
Проф.нал  = начислено/100
Всего     = Под.налог + Проф.нал
К выдаче  = Начислено - Всего
```

В одной из формул вычисление производится по-разному в зависимости от значения числа в графе "Начислено". Такие вычисления будем называть вычислениями в зависимости от условий.

Для того, чтобы МикроКальк вычислял графу по-разному, в зависимости от некоторых условий, в формуле для вычисления этой графы после знака равенства надо оставить пустое место, а в следующих строках разместить условия и формулы, по которым должна вычисляться графа при выполнении этих условий.

Условие и соответствующая ему формула должны располагаться в одной строке, начинающейся со слова "при". Условие записывается после слова "при" и может содержать операции сравнения

числовых выражений ('>' - больше, '<' - меньше, '=' - равно). Формула отделяется от условия знаком ':'. После строк с условиями может располагаться строка "иначе", содержащая формулу для вычисления графы в случае, когда не выполняется ни одно из условий. Формула может отделяться от слова "иначе" знаком ':' для единообразия.

Пример. Перевычислим таблицу по новым формулам. Поставим курсор на строку "Форма=..." и выполним команду «внутрь» (клавиши Esc ↓). На экране появится содержимое файла CWNZ.MIM — формулы для указанной в таблице формы.

Поставим курсор после знака равенства в 3-й формуле и удалим все символы справа от курсора многократным выполнением команды «удалить символ» (клавиша Del). Затем поставим курсор на 4-ю формулу и вставим над ней 3 пустые строки (команда Shift Ins). Введем в эти строки новые формулы (см. выше).

Вернемся в файл с таблицей командой «наружу» (клавиши Esc ↑) и перевычислим таблицу. По новым формулам в таблице и числа вычисляться будут новые (см. рис. 11).

Форма=Ведомость на зарплату N2

	Начислено	Удержано			К выдаче
		Под.налог	Проф.нал	Всего	
1.Иванов И.И.	52'100-00	3'792.00	521.00	4'313.00	47'787.00
2.Петров П.П.	41'512-32	2'521.48	415.12	2'936.60	38'575.72
Итого	93'612.32	6'313.48	936.12	7'249.60	86'362.72
3.Сидоров А.	20'500-00	0.00	205.00	205.00	20'295.00
4.Сидоров Б.	20'501-00	0.12	205.01	205.13	20'295.87
Итого	41'001.00	0.12	410.01	410.13	40'590.87
Всего	134'613.32	6'313.60	1'346.13	7'659.73	126'953.59

Рис. 11. Таблица, вычисленная по новым формулам.

Составьте таблицу значений функции SIGN, которая равна 1 для всех положительных чисел, −1 для отрицательных и 0 для 0. Для задания формулы, по которой будут вычисляться значения функции, используйте конструкцию "при".

§ 6. ДОПОЛНИТЕЛЬНЫЕ КОМАНДЫ МИКРОКАЛЬКА

6.1. Разделение цифр в числах

МикроКальку можно сообщить, каким способом он должен разделять цифры в числах:

- не разделять;
- разделять пробелами;
- разделять кавычками. Способ разделения изменяется командой Caps пробел.

Как узнать, какой способ разделения цифр действует в данный момент? В статусной строке МикроКалька способ разделения изображается следующим образом:

не разделять – пробел
разделять пробелами – знак подчеркивания: _
разделять кавычками – кавычка: '

Но как же разглядеть один символ в большой статусной строке? Нажмите клавиши Caps пробел и не отпускайте их. Символ разделения цифр «замигает», и вы его увидите.

Установите режим разделения цифр пробелами и вычислите таблицу, использованную в предыдущем разделе. Затем отмените разделение цифр и снова вычислите таблицу.

6.2. Режим автовычислений

После изменения некоторых данных в уже вычисленной таблице следует перевычислить таблицу, так как в результате изменений могут измениться записанные МикроКальком числа. Для того, чтобы вам не приходилось все время помнить о необходимости перевычислить таблицу, в МикроКальке есть режим автовычислений. В этом режиме МикроКальк автоматически перевычисляет таблицу после каждого ее изменения.

Режим автовычислений включается и выключается командой Caps A. Выполните эту команду несколько раз подряд, и вы увидите, что в статусной строке будет изменяться один символ. Символ 'a' соответствует включенному режиму автовычислений, пробел соответствует выключенному режиму автовычислений, а символ 'c' — режиму вычисления ссылок (который остался за пределами нашего внимания).

Установите режим автовычисления и измените в таблице, использованной в предыдущем разделе, зарплату Ива-

нову. Проследите, изменятся ли подоходный налог и другие числа при выходе курсора из измененной строки.

6.3. Построение гистограмм

Иногда даже таблицам не хватает наглядности. Тогда пользуются графическим представлением информации: графиками, диаграммами, гистограммами и т.д.

МикроКальк умеет строить горизонтальные гистограммы и графики. Для этого числовые данные должны быть расположены по вертикали. Данные надо отметить блоком и выполнить команду Esc F1 для построения гистограммы или Esc F2 для построения графика.

Давайте попробуем вместе построить график значений какой-нибудь функции, например квадратного корня. Вот наши действия:

1. Рисуем таблицу.
2. Заполняем 1-ю колонку таблицы числами, отстоящими друг от друга на равные промежутки.
3. Вычисляем таблицу нажатием клавиши F12.
4. Отмечаем 2-ю колонку прямоугольником (блоком):
   а) ставим курсор в верхнюю строку с числами, в самую левую позицию 2-й колонки (справа от вертикальной линии);
   б) начинаем отметку блока нажатием клавиши F5;
   в) ставим курсор в нижнюю строку с числами, на вертикальную линию справа от 2-й колонки (при этом левый верхний угол прямоугольника остается в левом верхнем углу 2-й колонки, а правый нижний угол следует за курсором).
5. Выполняем команду Esc F2.

число	кв.корень
0.00	0.00
1.00	1.00
2.00	1.41
3.00	1.73
4.00	2.00
5.00	2.24
6.00	2.45
7.00	2.65
8.00	2.83

На экране появляется график значений функции *квадратный корень*. Для возврата в текст следует нажать любую клавишу.

Постройте гистограмму количества ваших уроков в каждый день недели.

6.4. Вычисление формул

Кроме электронных таблиц МикроКальк содержит еще несколько полезных вычислительных возможностей. Одна из них — вычисление числовых формул.

Поставьте курсор на формулу, содержащую только числа, и нажмите клавишу F11 (или Caps Home) — рядом с формулой появится ее значение.

Для чего нужно вычисление формул? Для того же, для чего и вычисления на калькуляторе. А чем же вычисления в МикроКальке лучше вычислений на калькуляторе? Примерно тем же, чем набор текстов на компьютере лучше набора на печатной машинке:

- Вычисляемая формула — перед глазами, а не в уме, поэтому легко заметить ошибку.
- Допущенную ошибку легко исправить.
- Результат вычислений не надо списывать с табло калькулятора, а можно вывести на принтер.

Например, давайте узнаем, сколько квадратных метров приходится на человека в классной комнате. Измерим ее длину (допустим, получилось 7 м 35 см), ширину (допустим, 5 м 72 см). Запишем формулу вычисления площади прямоугольника:

```
7.35*5.72
```

Поставим курсор в любое место формулы и выполним команду «вычислить формулу» (клавиша F11). Справа от формулы появится результат:

```
7.35*5.72 =42.04
```

Полученный результат — это площадь классной комнаты. Теперь надо разделить этот результат на число человек. Припишем к результату знак деления и число человек:

```
7.35*5.72 =42.04/27
```

Поставим курсор на вторую формулу (42.04/27) и вызовем команду «вычислить формулу»:

```
7.35*5.72 =42.04/27 =1.56
```

МикроКальк может отказаться вычислять формулу, если он не может выделить ее из текста. Чтобы этого не случилось, надо окружить формулу пробелами.

Еще один способ вычислить формулу — выделить ее блоком и выполнить команду «вычислить блок» (F12 или Caps Home). Таким способом можно вычислить сразу несколько формул (расположенных в разных строках друг под другом). Надо отметить блоком их все и вызвать команду «вычислить блок».

Давайте проверим формулу $\sin 2x = 2\sin x \cos x$ для угла в 1 радиан, используя вычисление формул в МикроКальке. Запишем в разных строках формулы:

```
SIN(1)
2*SIN(0.5)*COS(0.5)
```

Отметим формулы блоком и нажмем F12. Получим:

```
SIN(1)              =1
2*SIN(0.5)*COS(0.5) =0.8
```

Ответы почему-то получились разные, хотя формула правильная. В чем же дело? Все объясняется просто — МикроКальк всегда выводит результат с точностью, равной максимальной точности используемых в формуле аргументов. Увеличим точность, добавив нулей после запятой в числах:

```
SIN(1.00)
2*SIN(0.50)*COS(0.5)
```

Вычислив снова, получим:

```
SIN(1.00)           =0.84
2*SIN(0.50)*COS(0.5) =0.84
```

Теперь результаты совпали.

Проверьте формулу $\cos(x + y) = \cos x \cos y - \sin x \sin y$ *для углов 1 и 2 радиан.*

6.5. Калькулятор в МикроКальке

В МикроКальке есть возможность вычислений, аналогичных вычислениям на калькуляторе.

Имеется ячейка памяти, в которой может храниться число. Содержимое ячейки можно складывать с другими числами, умножать и делить на них и вычитать из него другие числа. Результат операции помещается в эту же ячейку.

В начале сеанса редактирования ячейка памяти пуста. Для того, чтобы положить в нее какое-нибудь число, нужно выполнить операцию сложения содержимого ячейки памяти с этим числом. Содержимое ячейки памяти (если ячейка не пуста) изображается на табло, которое находится в статусной строке МикроМира, слева от имени файла.

Команды калькулятора	Клавиши
Выполнить операцию с содержимым памяти и с суммой чисел, содержащихся в отмеченном блоке	+, −, *, / (справа на доп.клавиатуре)
Выполнить операцию с содержимым памяти и с числом, на котором стоит курсор	Caps +, Caps −, Caps *, Caps /
Записать в текст содержимое памяти	Caps F6
Очистить содержимое памяти	Caps Shift F6
Увеличить у содержимого памяти число цифр после десятичной точки	Caps PageUp
Уменьшить у содержимого памяти число цифр после десятичной точки	Caps PageDown

Пусть в тексте встречается фраза «Произведение чисел 521.1 и 718.7 равно...», и мы хотим подставить вместо многоточия правильный результат. Для этого надо сначала вычислить значение 521.1×718.7.

Поставим курсор на первое число и выполним команду «сложить» (клавиши Caps +). В результате этих действий число 521.1 попадет в ячейку памяти и появится на табло.

Затем поставим курсор на второе число и выполним команду «умножить» (клавиши Caps *). Число 718.7 умножится на содержимое ячейки памяти, и теперь в ячейке будет лежать произведение этих двух чисел. Это произведение и появится на табло.

Итак, на табло изображается число 374'514.6. Оно имеет точность 1 знак после запятой, как и оба сомножителя. Однако точное значение произведения имеет 2 знака после запятой. Следовательно, на табло изображается неточный результат. Увеличим точность клавишами Caps PgUp. На табло появится число 374'514.57.

Теперь запишем в текст результат вычислений. Для этого поставим курсор на место многоточия и запишем в текст число, лежащее в ячейке памяти (команда Caps F6). Получим:

```
Произведение чисел 521.1 и 718.7 равно 374'514.57
```

Выясните, сколько знаков после запятой может иметь число, лежащее в ячейке памяти. Зачем нужно столько знаков, если размер табло не позволяет увидеть столько цифр?

Глава 3. Сортировка и поиск

§ 7. АЛГОРИТМЫ СОРТИРОВКИ

7.1. Что такое сортировка?

С процессом сортировки часто встречаются в реальной жизни. Например, с этим сталкиваются учителя, заполняя ежегодно классный журнал фамилиями новых учеников. При заполнении журнала учитель расставляет фамилии учеников по алфавиту, облегчая себе в дальнейшем поиск фамилии ученика в журнале (например, при проставлении оценок). При этом учитель знает алгоритм, по которому он первоначально размещает фамилии учеников в журнале.

На каждом этапе добавления фамилии нового ученика к уже готовому списку учитель однозначно находит место в журнале, где разместится фамилия. Однако журнал уже заполнен, и в нем нельзя проводить никаких исправлений. Поэтому в реальной жизни учитель, как правило, просматривает произвольно составленный список учеников и выбирает фамилию ученика, размещая ее в верхней строчке журнала. Затем снова просматривает список и выбирает фамилию для второй строчки. Процесс повторяется до тех пор, пока журнал не заполнен полностью. То, что рассмотрено выше, и есть процесс сортировки.

Формально процессом сортировки называют операцию **упорядочения** некоторого множества по ***ключу***. В предыдущей задаче **множество** учеников учитель упорядочивал по алфавиту, что вполне естественно. Можно сказать, что *ключом* в этой задаче явилась фамилия ученика. Конечно, бывают и более сложные процессы сортировки, при которых требуется упорядочение по нескольким ключам. Тогда скорректированное определение процесса сортировки выглядит так: «Процессом сортировки называют операцию упорядочения некоторого множества по **ключам**».

Интуитивно в процессе сортировки часто отождествляют элементы множества с ключами, по которым и проводится сортировка. Например, учитель может предположить, что в журнале на самом деле сортирует учеников, а не некоторые записи о них. Старшина, напротив, действительно сортирует солдат взвода по команде: «Стройся!». На алгоритмы это замечание никак не влияет, поэтому здесь не надо обращать на него никакого внимания.

Кроме того, что алгоритмы сортировки имеют большое прикладное значение, они интересны и сами по себе. Эта достаточно глубоко исследованная область информатики используется в информационно-поисковых системах, в военном и банковском деле. На сегодняшний день трудно представить себе область компьютерных приложений, не использующую процесс сортировки.

7.2. Сложность алгоритмов сортировки

В каждом алгоритме интересно оценить его так называемую, ***сложность***. Под сложностью предполагается максимальное число элементарных шагов алгоритма.

В задаче про учительский журнал за первый шаг учитель должен выбрать фамилию ученика на первое место в журнале. Для этого ему потребуется просмотреть все множество фамилий. Затем, изъяв эту фамилию из неупорядоченного списка-множества, учитель помещает ее в журнал. За второй шаг учитель снова просматривает множество фамилий, выбирая претендента на второе место в журнале. Этот шаг потребует уже на один просмотр меньше. И так далее, пока множество не исчерпается и все фамилии учеников не перекочуют в журнал упорядоченными по алфавиту.

Пусть в классе 40 учеников. Тогда первый шаг потребует 40 просмотров, второй $39 = 40 - 1$, третий $38 = 40 - 2$ и т.д. И по известной формуле (арифметической прогрессии) получается что всего потребуется $40 \times (40+1)/2 = 820$ просмотров. (В общем случае формула выглядит так: $N*(N+1)/2$, где N – число учеников в классе). Это и будет сложностью алгоритма.

Понятно, что чем меньше элементарных шагов требуется для сортировки одного и того же множества, тем лучше алгоритм. Рассмотренный алгоритм является одним из наихудших, т.е. самых сложных алгоритмов сортировки. Он относится к так называемым *сортировкам извлечением*. Основная идея сортировок извлечением состоит в «выдергивании» элемента с минимальным ключом и размещении его в отведенном месте.

7.3. Пузырьковая сортировка

Это тоже достаточно простой алгоритм, но работающий более эффективно, если карточки с фамилиями учеников уже частично упорядочены. Пузырьковая сортировка названа так по образной интерпретации алгоритма, когда более «легкие» элементы множества как бы «всплывают на поверхность».

Алгоритм работает следующим образом. Пусть у учителя есть неотсортированный список учеников, представленный на карточках (чтобы было удобно переставлять их местами). Учитель снова просматривает все карточки с фамилиями учеников, сравнивая ближайшие две. Если две ближайшие карточки расположены не в правильном порядке, то учитель меняет их местами. И так далее. Алгоритм заканчивает работу, когда учитель за шаг просмотра всех карточек ни разу не переставил ни одной.

Рассмотрим пример на множестве из 4-х фамилий:

{ ПЕТРОВ, СИДОРОВ, ИВАНОВ, ПЕРВИН }

Проследим работу по шагам простого алгоритма сортировки извлечением (см. п. 7.1) и алгоритма пузырьковой сортировки из настоящего пункта:

Пузырьковая сортировка	**Простое извлечение**
список фамилий до начала алгоритма	
ПЕТРОВ ИВАНОВ СИДОРОВ ПЕРВИН	ПЕТРОВ ИВАНОВ СИДОРОВ ПЕРВИН
первый шаг алгоритма	
За первый шаг осуществляется 3 сравнения: ПЕТРОВ и ИВАНОВ меняются местами; ПЕТРОВ и СИДОРОВ находятся в правильном порядке; СИДОРОВ и ПЕРВИН меняются местами; на этом этапе перестановки были — значит, алгоритм продолжает работу.	Просматривается весь список и выбирается элемент для первой позиции в списке фамилий; в данном случае потребуется 4 элементарных операции и выбранный элемент ИВАНОВ; элемент, находящийся на первой строчке, и ИВАНОВ меняются местами.
список фамилий после первого просмотра	
ИВАНОВ ПЕТРОВ ПЕРВИН СИДОРОВ	ИВАНОВ ПЕТРОВ СИДОРОВ ПЕРВИН

второй шаг алгоритма

Повторяется предыдущий шаг: ИВАНОВ сравнивается с ПЕТРОВым, и оба остаются на своих местах; ПЕТРОВ сравнивается с ПЕРВИНым, и осуществляется их перестановка; ПЕТРОВ сравнивается с СИДОРОВым, и оба остаются на своих местах. Алгоритм продолжает работу.	Вновь просматривается список фамилий, но уже начиная со второй строки; за 3 действия ищется «минимальный» элемент ПЕРВИН и помещается на второе место перестановкой ПЕТРОВа и ПЕРВИНа.

список фамилий после второго просмотра

ИВАНОВ	ИВАНОВ
ПЕРВИН	ПЕРВИН
ПЕТРОВ	СИДОРОВ
СИДОРОВ	ПЕТРОВ

третий шаг алгоритма

Хотя интуитивно понятно, что список уже составлен правильно, алгоритм продолжает работу, сравнивая ИВАНОВа и ПЕРВИНа, ПЕРВИНа и ПЕТРОВа, а ПЕТРОВа с СИДОРОВым; перестановок на этом этапе нет, поэтому алгоритм завершает свою работу.	Просматривается список фамилий, начиная с третьей; ПЕТРОВ является наименьшим элементом, и перестановкой ПЕТРОВа с третьей строчкой завершается работа алгоритма.

список фамилий после третьего просмотра

ИВАНОВ	ИВАНОВ
ПЕРВИН	ПЕРВИН
ПЕТРОВ	ПЕТРОВ
СИДОРОВ	СИДОРОВ

Оба алгоритма завершили свою работу. Внешне кажется, что пузырьковая сортировка работает быстрее, однако сложность этого метода больше, чем метода простых извлечений. Если претендент на первую строчку находится на последнем месте, то только за $N-1$ просмотров (где N – число учеников) этот элемент займет свое место, на каждом же шаге требуется $N-1$ сравнение. Таким образом, налицо сложность алгоритма пузырьковой сортировки $(N-1)^2$. Однако пузырьковая сортировка хорошо работает на частично упорядоченных множествах.

Упражнение:

24. Сравните работу алгоритмов пузырьковой сортировки и простых извлечений на задаче про заполнение журнала при исходном расположении фамилий:

а)	б)	в)
Иванов	Иванов	Сидоров
Петров	Первин	Петров
Первин	Петров	Первин
Сидоров	Сидоров	Иванов

7.4. Слияние двух отделений

Постановка задачи. Большой военный праздник. Принятие присяги. Два отделения солдат уже построены, но командир первого отделения оступается, ломает ногу, и сержант Левенчук (командир второго отделения) должен взять команду на себя. Первое отделение построено по росту. Второе тоже. Перед присягой сержант должен объединить два отделения в одно так, чтобы в новом отделении солдаты тоже располагались по росту. Сержант Левенчук может скомандовать «Разойдись!» и потом снова «Становись!», но солдаты привыкли к своему месту в строю своего отделения и могут создать суету при построении в одно отделение. Тогда сержант Левенчук сам решает эту задачу.

Решение задачи. Отвлекаясь от местности и внешности рассматриваемых людей (так как нас интересует только их рост), можно заменить солдат в задаче трехзначным числом, равным их росту в сантиметрах. Например, рядового Иванова с ростом 158 см можно заменить числом 158. Рост солдат и будет ключом, по которому станет проводиться сортировка. Наличие солдат с одинаковым ростом не мешает решению задачи, так как все равно, какой из этих солдат будет располагаться первым. Сержант Левенчук строит солдат по отделениям одно позади другого, командует «Направо!» и сам располагается со стороны начала строя:

```
                1-е отделение
сержант         200 187 180 180 176 164
Левенчук
                2-е отделение
                189 185 180 175 158
```

Прямо перед ним стоят два солдата с наибольшим ростом: 200 см из 1-го отделения и 189 см — из второго. Сержант Левенчук решает, который из этих солдат выше, и, подзывая его к

себе, направляет на первое место в новом отделении:

```
                1-е отделение
   сержант      187 180 180 176 164
  Левенчук
                2-е отделение
 Новое          189 185 180 175 158
 отделение
 200
```

Снова перед сержантом стоят два солдата — один с ростом 187 см из 1-го отделения и второй с ростом 189 см — из второго. Сейчас, конечно, рядовой из второго отделения выше, и он должен занять второе место в новом отделении:

```
                1-е отделение
   сержант      187 180 180 176 164
  Левенчук
                2-е отделение
 Новое          185 180 175 158
 отделение
 200 189
```

Операция повторяется снова и снова:

```
                1-е отделение
   сержант      180 180 176 164
  Левенчук
                2-е отделение
 Новое          180 175 158
 отделение
 200 189 187 185
```

Возникла проблема, когда оба солдата (которых надо перераспределить в новый строй) одинакового роста. Не беда! Можно отправить туда любого, например из своего родного, второго отделения:

```
                1-е отделение
   сержант      180 180 176 164
  Левенчук
                2-е отделение
 Новое          175 158
 отделение
 200 189 187 185 180
```


Задача, наконец, решена:

```
                1-е отделение
   сержант
  Левенчук
                2-е отделение
 Новое
 отделение
 200 189 187 185 180 180 180 176 175 164 158
```

В отделениях 1 и 2 не осталось никого, зато новое отделение построено по росту. Можно принимать присягу!

7.5. Сортировка слиянием, комбинация алгоритмов

Рассмотренный выше пример демонстрирует работу алгоритма *Сортировка слиянием* на двух уже отсортированных множествах. Сложность такого алгоритма очень мала и, как видно из примера, требует $N - 1$ сравнений (где N – суммарное число солдат в отделениях). Однако такой алгоритм применим в только для решения специфических задач (например, слияние отделений). Очень часто задачи сортировки решаются не одним методом, а комбинацией нескольких методов.

Задача про спортивные занятия взвода лейтенанта Никитина

Постановка задачи. Командир взвода лейтенант Никитин каждое утро, ровно в 6:00, производит пробежку со своим взводом через ближайший лес сразу после зарядки. На зарядке лейтенант Никитин требует, чтобы взвод становился в 3 шеренги, в каждой шеренге солдаты стоят по росту. А на пробежке все бегут по одному (также по росту) за лейтенантом. Таким образом, перед лейтенантом и взводом стоят две задачи: сортировка в шеренгах и сортировка слиянием из трех шеренг.

Решение задачи. Совершенно неважно, кто является непосредственным исполнителем этих задач. Проще предположить, что сортировку производит сам лейтенант Никитин. Задача сводится к троекратному выполнению сортировки пузырьком (в шеренгах) и сортировки слиянием из трех шеренг в одну. Сортировка пузырьком подробно рассмотрена в п. 7.3, сортировка слиянием рассмотрена в п. 7.4. Как и в п. 7.4, солдат сравнивают по ключу–

росту. Итак:

```
                178 158 176 193 168     - 1 шеренга
лейтенант
                163 166 190 188 187     - 2 шеренга
Никитин
                176 180 176 190         - 3 шеренга
```

Нет необходимости рассматривать сортировку пузырьком каждой шеренги в отдельности. Достаточно рассмотреть этапы сортировки на примере первой шеренги:

```
                178 158 176 193 168     - 1 шеренга
1 просмотр      178 176 193 168 158
2 просмотр      178 193 176 168 158
3 просмотр      193 178 176 168 158
                193 178 176 168 158     - 1 шеренга
```

Аналогично сортируются и остальные две шеренги. Итак, построенные на зарядку солдаты в шеренгах выглядят так:

```
                193 178 176 168 158     - 1 шеренга
лейтенант
                190 188 187 166 163     - 2 шеренга
Никитин
                190 180 176 176         - 3 шеренга
```

После проведения занятий лейтенант Никитин перестраивает шеренги в одну по росту, пользуясь описанным в п. 7.3 алгоритмом. 1-й шаг:

```
                178 176 168 158         - 1 шеренга
лейтенант
                190 188 187 166 163     - 2 шеренга
Никитин
                190 180 176 176         - 3 шеренга
Общая шеренга
193
```

2-й шаг:

```
                178 176 168 158         - 1 шеренга
лейтенант
                188 187 166 163         - 2 шеренга
Никитин
                190 180 176 176         - 3 шеренга
Общая шеренга
193 190
```

Результат:

```
Общая шеренга лейтенанта Никитина
193 190 190 188 187 180 178 176 176 176 168 166 163 158
```

Упражнение:

25. Сравните работу алгоритмов пузырьковой сортировки, простых извлечений на примере про спортивные занятия взвода лейтенанта Никитина и сравните эффективность с рассмотренной в п. 7.4 комбинацией алгоритмов для взвода {178, 158, 176, 193, 168, 163, 166, 190, 188, 187, 176, 180, 176, 190}.

7.6. Быстрая сортировка

Метод быстрой сортировки относится к методам сортировки обменами, как и метод пузырьковой сортировки. Однако идеи, заложенные в этом методе, позволяют гораздо эффективнее проводить сортировку, чем в случае пузырька.

Метод быстрой сортировки состоит в делении множества на две части с выбором так называемого **главного элемента**, при котором в одной части все элементы меньше главного, а во второй части все элементы больше главного. В дальнейшем тот же алгоритм применяется к каждой части множества, деля их соответственным образом, и т.д. Так как при каждом обращении к алгоритму обязательно появляется один элемент, стоящий на своем месте, то в конце концов алгоритм заканчивает работу.

Задача про дырявый мешок

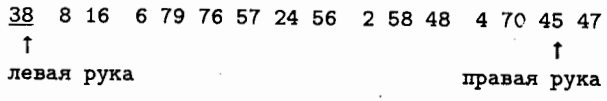
Постановка задачи. Однажды вечером семья играла в лото. Бабушка доставала бочонки из мешка и выкрикивала их номер. Но тут оказалось, что части бочонков не хватает (они выпали на пол, и их раскатал по комнате домашний кот Мурзик). Все бросились их искать и собрали целых 16 штук. Тут папа подал идею, что лучше их расставить по порядку возрастания, чтобы было ясно, каких еще нет (снимать бочонки с карт никто не хотел, ведь шла игра), и сам занялся исполнением алгоритма быстрой сортировки (папа преподавал информатику в школе).

Решение задачи. Бочонков было много — целых шестнадцать штук. Папа взял *первый попавшийся* под руку бочонок и сказал, что он *выбрал главный элемент* — 38. Теперь надо поделить эту кучу бочонков на две. Для этого он расставил все бочонки в ряд перед собой:

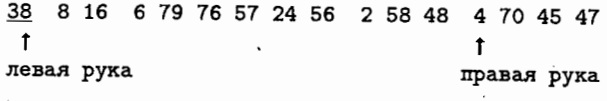
```
38  8 16  6 79 57 24 56  2 58 48  4 70 45 47
Главный элемент - 38.
```

Поставив указательный палец левой руки на левый край (38), а указательный палец правой руки на правый край (47), папа сравнил число на бочонке под правой рукой с числом на бочонке под левой рукой (38 < 47): папа просто сдвинул правую руку на

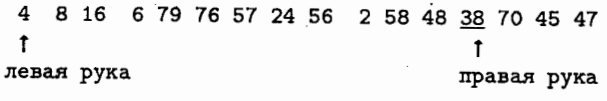
один бочонок к центру:



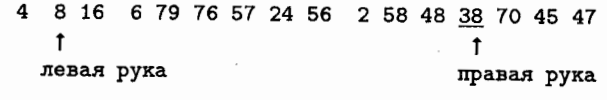
Папа произвел еще одно сравнение: 38 < 45-и сдвинул правую руку. Снова: 38 < 70, еще влево:



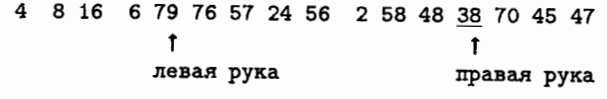
Вот тут обнаружилось, что 38 > 4. Папа поменял местами 38 и 4:



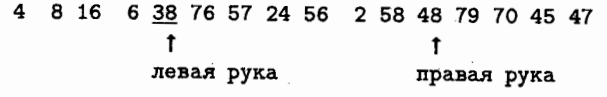
Теперь папа обратил взор на левую руку и сразу после обмена сдвинул указательный палец на один бочонок вправо:



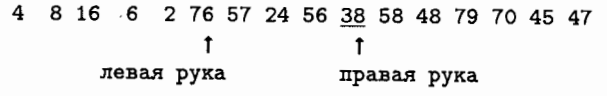
8 < 38 — сдвиг вправо. 16 < 38 — еще сдвиг. 6 < 38 — снова вправо:



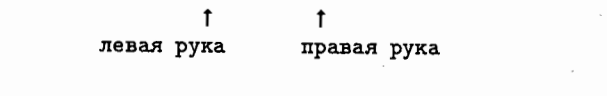
Вот! 79 > 38. Папа поменял их местами и сдвинул правую руку на бочонок влево:



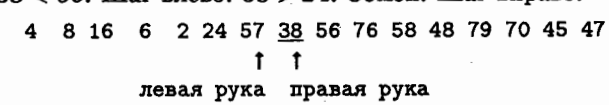
Опять к правой руке. 38 < 48. Шаг влево правой рукой. 38 < 58, еще влево, но 38 > 2 — обмен! И левую руку на шаг вправо:



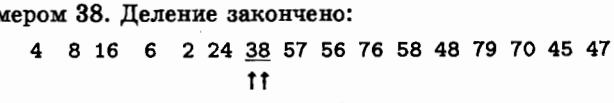
Снова обмен: 76 > 38. И шаг влево:



38 < 56. Шаг влево. 38 > 24. Обмен. Шаг вправо:



57 > 38. Обмен. И шаг... Но тут указательный палец правой руки и указательный палец левой руки оказались на бочонке с номером 38. Деление закончено:



Что получилось в результате: все бочонки слева от главного элемента с номерами меньшими, а справа от главного элемента — с номерами большими, чем у главного. Хитрый папа отдал сортировать правую часть маме, а левую — бабушке, наказав, чтобы они четко следовали предложенным алгоритмам. Мама не послушалась папу и, считая, что у нее слишком мало бочонков, отсортировала их простым пузырьком. А бабушка отсортировала свою часть. Поделила ее на две с НОВЫМ главным элементом. Позвала внуков и еще за один–два этапа закончила работу.

Упражнения:

26. Посмотрите, как работает алгоритм быстрой сортировки на следующем ряду бочонков лото:

2, 4, 24, 16, 38, 57, 56, 79, 76.

Сравните работу этого алгоритма с пузырьком на этом же ряду.

27. Отсортируйте методом быстрой сортировки следующий ряд бочонков лото:

4, 8, 16, 6, 79, 57, 24, 56, 2, 58, 48, 38, 70, 45, 47.

Что у вас получилось при первом делении ряда на две части? Какой получится главный элемент и его место в ряду?

7.7. Метод простого включения

Метод простого включения предполагает, что имеется отсортированное множество из N элементов, например упорядоченный список молодежного клуба «Альфа–бета–гамма». Предлагается алгоритм включения на свое место нового элемента множества, например нового члена клуба. Новый элемент добавляется в конец списка, и осуществляется просмотр с конца списка и обен пары соседних элементов, пока новый элемент не встретит в списке элемент меньше себя. Здесь опять элемент отождествляется со

своим ключом и список предполагается отсортированным по возрастанию.

Задача про прием в клуб «Альфа–бета»

Постановка задачи. В молодежном клубе «Альфа–бета» всех называют только по именам, и нового члена при приеме надо вписать в общий список. Естественно, что список членов клуба уже отсортирован. И задача сводится к вставлению нового имени в уже готовый список.

Решение задачи. Список членов клуба составлен из их имен:

```
Александр
Алексей
Иван
Петр
Сергей
Яков
```

При приеме нового члена клуба — Евгения — его имя заносится последним в список, а затем сравнивается с предыдущим именем. Яков, естественно, должен стоять в списке после Евгения, поэтому их имена переставляются в списке:

```
Александр
Алексей
Иван
Петр
Сергей
Евгений
Яков
```

Таким же образом Евгений сравнивается с Сергеем, и они снова меняются местами. Далее Евгений переставляется с Петром и Иваном:

```
Александр
Алексей
Евгений
Иван
Петр
Сергей
Яков
```

Однако Алексей должен стоять раньше Евгения. На этом работа алгоритма заканчивается, так как список членов клуба был уже отсортирован (без Евгения).

Упражнение:

28. В клубе потерялся список членов клуба. Для восстановления списка требуется отсортировать его целиком методом простого включения:

```
Евгений
Яков
Иван
Александр
Петр
Алексей
Сергей
```

§ 8. ПОИСК

8.1. Зачем нужен порядок? Последовательный поиск

Есть часто встречающиеся задачи поиска какой-либо вещи или информации. Например, бабушка ищет в лотошном мешочке бочонок с номером 11. Или бизнесмен перебирает визитные карточки знакомых, чтобы найти некоего господина и его телефон. Такие задачи встречаются сплошь и рядом, и самое обидное, что при всей простоте алгоритма поиска он требует подчас пересмотра всех бочонков в мешке или всех визиток в визитнице. Ведь, как назло, нужный телефон всегда оказывается в последней просмотренной визитке или бочонок номер 11 лежит на дне мешка. Эти задачи решаются так называемым ***последовательным поиском***.

Последовательным он называется потому, что требуется *последовательно* перебирать все элементы множества, в котором осуществляется поиск, пока не найдется искомый. Как и в случае алгоритмов сортировки, поиск осуществляется по ключу, но, не ограничивая общности, всегда можно рассматривать алгоритмы, отождествляя ключи с элементами множества.

Поиск в этих множествах осуществляется последовательно, так как в множестве нет никакого порядка. Бочонки в лотошном мешке лежат как попало, визитки в визитницу вставлялись по мере поступления (действительно, если бы надо было найти визитку, положенную в визитницу самой первой, то это было бы легко, так как визитки расположены по порядку их получения). Зато к положительным качествам последовательного поиска относится его универсальность.

Постановка задачи. Пусть у бабушки в мешке 10 бочонков

лото и она достала их в следующем порядке (здесь порядок вынимания бочонков из мешка относится к выполнению алгоритма, а не к порядку в множестве — замечательная игра слов!):

23, 27, 16, 67, 90, 11, 3, 80, 66, 19.

Решение задачи. Если требуется найти бочонок «барабанные палочки» — 11, то бабушка сделает это, *шестой* раз запустив руку в мешок. При поиске конкретного бочонка (11) в вышеописанном опыте получается, что алгоритм требует выполнения примерно половины шагов от общего числа элементов. Этот опыт дает практически верное представление о *сложности* последовательного поиска. Но если бабушка ищет бочонок 27, то удача ждет ее на втором шаге (потребовалось втрое меньше времени на поиск), а если искомый бочонок имеет номер 19, то придется перебрать все бочонки. Из проделанного опыта следует, что время поиска существенно зависит от расположения искомого элемента в множестве, но в множестве нет никакого порядка. Вот если бы порядок был, то можно было бы предложить более эффективный алгоритм.

8.2. Двоичный поиск

Постановка задачи. Папа — глава семьи — преподавал информатику в школе и как раз рассказывал своим ученикам про алгоритмы сортировки. Поэтому, если папа видел что-то, что можно было отсортировать, сразу же применял изучаемые алгоритмы на практике. Это случилось и тогда, когда бабушка искала бочонок в лотошном мешке. Папа *высыпал* бочонки из мешка, *отсортировал их по возрастанию*, предложив бабушке другой алгоритм поиска.

Решение задачи. На самом деле папа, отсортировав бочонки по возрастанию, установил *порядок* на множестве (по возрастанию):

3, 11, 16, 19, 23, 27, 66, 67, 80, 90.

Все бочонки сейчас выставлены на стол, и папа начал поиск бочонка (например) с номером 19 по своему алгоритму. Бабушка только наблюдала за ним. Папа встал перед столом и положил левую руку напротив бочонка с номером 3, а правую — напротив бочонка с номером 90:

```
 3 11 16 19 23 27 66 67 80 90
 ↑                          ↑
левая рука                  правая рука
```

Затем папа посмотрел на *середину*: левая рука указывает на бочонок, стоящий на первом месте, правая рука — на бочонок,

стоящий на 10 месте, середина $(1 + 10)/2 = 5.5$, или, можно считать, на 5 месте, т.е. на бочонке с номером 23:

```
 3 11 16 19 23 27 66 67 80 90
 ↑           ↑              ↑
левая рука   |              правая рука
             папа смотрит
             на бочонок с номером 23
```

Папа *сравнил* номер бочонка (23) с искомым (19). Так как 19 меньше, чем 23, это означает, что 19 находится *левее* бочонка с номером 23. И тогда папа *передвинул правую руку* на бочонок с номером 23:

```
 3 11 16 19 23 27 66 67 80 90
 ↑           ↑
левая рука   правая рука
```

На *втором* шаге алгоритма папа снова посмотрел на *середину*: левая рука указывает на бочонок, стоящий на первом месте, правая рука — на бочонок, стоящий на 5 месте, середина $(1 + 5)/2 = 3$ — на 3 место, т.е. на бочонок с номером 16:

```
 3 11 16 19 23 27 66 67 80 90
 ↑     ↑     ↑
левая  |     правая рука
рука   |
       папа смотрит на бочонок с номером 16
```

Папа снова *сравнил* номер бочонка (16) с искомым (19). Т.к. 19 больше, чем 16, это означает, что 19 находится *правее* бочонка с номером 16. И тогда папа *передвинул левую руку* на бочонок с номером 16:

```
 3 11 16 19 23 27 66 67 80 90
       ↑     ↑
левая рука   правая рука
```

На *третьем* шаге алгоритма папа посмотрел на *середину*: левая рука указывает на бочонок, стоящий на 3 месте, правая рука — на бочонок, стоящий на 5 месте, середина $(3 + 5)/2 = 4$ — на 4 место, т.е. на бочонок с номером 19 (вообще-то, это и есть искомый бочонок).

```
 3 11 16 19 23 27 66 67 80 90
       ↑  ↑  ↑
левая рука | правая рука
           |
       папа смотрит на бочонок с номером 19
```

Папа, следуя предложенному им алгоритму, должен *сравнить* номер бочонка с искомым. Но это и есть искомый бочонок.

Алгоритм, приведенный выше, называют *двоичным поиском*. Двоичным он называется потому, что каждый шаг алгоритма делит множество пополам и поиск осуществляется уже в оставшейся части множества. Легко подсчитать, что в приведенном примере максимальное число шагов равно 4. Это означает, что двоичный поиск намного эффективнее последовательного (максимальное число шагов последовательного поиска для приведенного выше примера равно 10).

Однако что было бы, если бы папа искал бочонок с номером, отсутствующим в данном множестве? Пусть требуется найти бочонок с номером 20. Первые два шага алгоритма будут точно такими же, как и в рассмотренном выше примере. Различия начинаются тогда, когда папа доходит до бочонка с номером 19:

```
 3 11 16 19 23 27 66 67 80 90
       ↑  ↑  ↑
левая рука | правая рука
           |
           папа смотрит на бочонок с номером 19
```

Папа сравнивает номер бочонка 19 и 20. 20 больше, чем 19, и папа передвигает левую руку на бочонок с номером 19:

```
 3 11 16 19 23 27 66 67 80 90
          ↑  ↑
левая рука  правая рука
```

Тут папа видит, что *левая и правая руки находятся рядом*, между ними нет больше бочонков. Это означает, что бочонка с номером 20 вообще нет в этом множестве.

Проведенный поиск (как последовательный, так и двоичный) позволяет или *найти элемент множества*, или *сообщить о том, что элемента в множестве нет*. Однако двоичный поиск в последнем примере имеет еще одну очень интересную особенность: ненайденный бочонок должен был *располагаться между папиными руками*. Т.е. если бы бочонок 20 был в множестве, то его место находилось бы как раз между 19 и 23. Этот полезный факт может использоваться при добавлении элемента в упорядоченное множество. Если добавить элемент 20 на место между 19 и 23, то порядок в множестве сохранится.

Упражнения:

29. Сравните сложность (число шагов) алгоритмов последовательного и двоичного поиска на примере с лотошными бочонками:

а) 3, 12, 16, 18, 19, 30, 33, 35, 56, 67, 69, 89 при поиске 69 и 16;

б) 5, 9, 17, 27 при поиске 5 и 17.

30. Какое максимальное число шагов алгоритма потребуется при двоичном поиске, если число элементов множества равно:

а) 10; б) 100; в) 1000.

8.3. Индексирование. Смешанный поиск

Если обратить внимание на обычную телефонную записную книжку, которая есть у каждого, то окажется, что поиск нужного телефона каждый раз реализуется достаточно сложным алгоритмом, основанным на принципе устройства телефонной книги. Опять совершенно не важно, что *объектом поиска* является телефонный номер, а *ключом* — фамилия. Здесь интересен сам алгоритм поиска, поэтому снова можно отождествить объект поиска с его ключом. Можно считать, что объектом поиска является фамилия.

Как же устроена обычная телефонная книга? Она состоит из 3-х десятков страниц. Каждая страница помечена буквой: А, Б, В, Г, Д, ..., Я (по алфавиту). Буква на странице, например «В», означает, что на этой странице находятся фамилии, начинающиеся на букву «В» (Волгин, Волобуев и т.п.). В процессе заполнения книги фамилии на страницы добавляются в произвольном порядке. Поиск в телефонной книге осуществляется в два этапа: *индексный*, когда по первой букве разыскиваемой фамилии однозначно определяется страница, на которой следует продолжать поиск, и второй этап — последовательный просмотр фамилий на странице; до тех пор, пока не будет найдена нужная.

Постановка задачи. У пятиклассника Пети телефоны школьных друзей записаны в телефонной книге. В ней уже почти 50 фамилий, т.к. в школе у Пети два пятых класса. Однажды вечером Пете позвонил друг Вася и попросил номер телефона их одноклассника Ахмета Шарафетдинова, и Петя, так как не помнил телефон Ахмета наизусть, взял в руки свою телефонную книжку.

Решение задачи. Первый этап поиска состоял в том, что Петя *за один шаг* нашел нужную страницу в книжке ("Ш"), используя поиск по индексу. Пусть страницы в книге пронумерованы от 1 до 28 (33 буквы в алфавите, в книжке нет букв «Й», «Ё», «Ь»,

«Ы»; «Ъ»):

1 – А	8 – З	15 – П	22 – Ц
2 – Б	9 – И	16 – Р	23 – Ч
3 – В	10 – К	17 – С	24 – Ш
4 – Г	11 – Л	18 – Т	25 – Щ
5 – Д	12 – М	19 – У	26 – Э
6 – Е	13 – Н	20 – Ф	27 – Ю
7 – Ж	14 – О	21 – Х	28 – Я

Тогда Петя *вычисляет*, что фамилии, начинающиеся на букву «Ш», находятся на 24 странице, и *сразу* открывает нужную страницу. Это и был поиск по индексу. Далее, на втором этапе, открыв страницу на букву «Ш», Петя видит несколько фамилий:

```
Шевцов Алексей - 888-13-11
Шилинговская Вера - 888-18-67
Шарафетдинов Ахмет - 888-78-87
```

В этом списке фамилий Петя ищет нужную последовательным поиском, начиная с первой фамилии: «Шевцов» — это не «Шарафетдинов», значит, надо смотреть ниже. «Шилинговская» — это тоже не "Шарафетдинов", Петя ищет дальше.

И вот, наконец, найдена нужная фамилия (и нужный телефон). Но друг Вася хочет еще узнать и телефон Ефимова Вити. И Петя продолжает поиск: фамилии, начинающиеся на букву «Е», находятся на 6 странице. Петя открывает эту страницу и ищет на ней фамилию «Ефимов»:

```
Еленев Николай - 888-13-12
Евстратов Петя - 888-66-69
Елкин Гена - 888-45-45
```

Это не «Еленев», Петя смотрит дальше. Это не «Евстратов», дальше. «Ефимов» не «Елкин», но список окончен, значит, Ефимова Вити в книжке нет. Попутно следует обратить внимание, что если бы Петя хотел записать телефон Ефимова Вити, то, не найдя его в книжке, Петя, тем не менее, нашел бы место, куда можно было бы записать телефон Вити.

Постановка задачи. Бабушка снова разбирала бочонки из лотошного мешочка. Папа — преподаватель информатики в школе — решил помочь ей, предлагая заполнить *таблицу* номерами имеющихся бочонков, используя номер бочонка в качестве *индекса*.

Решение задачи. Подготовительный этап состоял в том, что бабушка перебрала все бочонки в мешке, а папа заполнил таблицу из 90 строк (рис. 12). Графу «номер бочонка» папа заполнил номерами от 1 до 90. В графе «наличие» папа проставлял «есть»,

номер бочонка	наличие
1	
2	
3	
4	
5	
6	
7	

номер бочонка	наличие
8	
9	
10	
. . .	
88	
89	
90	

Рис. 12. Таблица наличия бочонков.

если бабушка называла номер бочонка. По окончании подготовительного этапа все пустые места в графе «наличие» папа заполнил словами «нет», показывая, что бочонков с такими номерами нет у бабушки в мешке. В результате получилась таблица, где номер бочонка использовался в качестве индекса при поиске номера бочонка в мешке-множестве.

Остался ли бочонок с номером 23 у бабушки в мешке? Для этого папа смотрит на 23 строку таблицы:

номер бочонка	наличие
1	нет
2	нет
. . .	
19	да
20	нет
21	нет
22	нет

номер бочонка	наличие
23	да
24	нет
25	нет
26	нет
27	да
. . .	
90	да

Напротив номера 23 в графе «наличие» написано «есть», это означает, что бочонок с номером 23 есть у бабушки в мешке. Если нас интересует номер 26, то из папиной таблицы видно, что такого бочонка нет у бабушки, т.к. в строке 26 в графе «наличие» стоит слово «нет».

При поиске в таблице папа, за один шаг попадая в нужную строку при помощи индекса-номера бочонка, узнавал ответ на вопрос: есть бочонок с заданным номером у бабушки в лотошном мешке или нет.

8.4. Сложный индекс

В примере с индексным поиском про бабушкин лотошный мешок в качестве индекса использовались номера бочонков, т.е. ин-

«Ы»; «Ъ»):

1 – А	8 – З	15 – П	22 – Ц
2 – Б	9 – И	16 – Р	23 – Ч
3 – В	10 – К	17 – С	24 – Ш
4 – Г	11 – Л	18 – Т	25 – Щ
5 – Д	12 – М	19 – У	26 – Э
6 – Е	13 – Н	20 – Ф	27 – Ю
7 – Ж	14 – О	21 – Х	28 – Я

Тогда Петя *вычисляет*, что фамилии, начинающиеся на букву «Ш», находятся на 24 странице, и *сразу* открывает нужную страницу. Это и был поиск по индексу. Далее, на втором этапе, открыв страницу на букву «Ш», Петя видит несколько фамилий:

```
Шевцов Алексей - 888-13-11
Шилинговская Вера - 888-18-67
Шарафетдинов Ахмет - 888-78-87
```

В этом списке фамилий Петя ищет нужную последовательным поиском, начиная с первой фамилии: «Шевцов» — это не «Шарафетдинов», значит, надо смотреть ниже. «Шилинговская» — это тоже не "Шарафетдинов", Петя ищет дальше.

И вот, наконец, найдена нужная фамилия (и нужный телефон). Но друг Вася хочет еще узнать и телефон Ефимова Вити. И Петя продолжает поиск: фамилии, начинающиеся на букву «Е», находятся на 6 странице. Петя открывает эту страницу и ищет на ней фамилию «Ефимов»:

```
Еленев Николай - 888-13-12
Евстратов Петя - 888-66-69
Елкин Гена - 888-45-45
```

Это не «Еленев», Петя смотрит дальше. Это не «Евстратов», дальше. «Ефимов» не «Елкин», но список окончен, значит, Ефимова Вити в книжке нет. Попутно следует обратить внимание, что если бы Петя хотел записать телефон Ефимова Вити, то, не найдя его в книжке, Петя, тем не менее, нашел бы место, куда можно было бы записать телефон Вити.

Постановка задачи. Бабушка снова разбирала бочонки из лотошного мешочка. Папа — преподаватель информатики в школе — решил помочь ей, предлагая заполнить *таблицу* номерами имеющихся бочонков, используя номер бочонка в качестве *индекса*.

Решение задачи. Подготовительный этап состоял в том, что бабушка перебрала все бочонки в мешке, а папа заполнил таблицу из 90 строк (рис. 12). Графу «номер бочонка» папа заполнил номерами от 1 до 90. В графе «наличие» папа проставлял «есть»,

номер бочонка	наличие
1	
2	
3	
4	
5	
6	
7	

номер бочонка	наличие
8	
9	
10	
. . .	
88	
89	
90	

Рис. 12. Таблица наличия бочонков.

если бабушка называла номер бочонка. По окончании подготовительного этапа все пустые места в графе «наличие» папа заполнил словами «нет», показывая, что бочонков с такими номерами нет у бабушки в мешке. В результате получилась таблица, где номер бочонка использовался в качестве индекса при поиске номера бочонка в мешке-множестве.

Остался ли бочонок с номером 23 у бабушки в мешке? Для этого папа смотрит на 23 строку таблицы:

номер бочонка	наличие
1	нет
2	нет
. . .	
19	да
20	нет
21	нет
22	нет

номер бочонка	наличие
23	да
24	нет
25	нет
26	нет
27	да
. . .	
90	да

Напротив номера 23 в графе «наличие» написано «есть», это означает, что бочонок с номером 23 есть у бабушки в мешке. Если нас интересует номер 26, то из папиной таблицы видно, что такого бочонка нет у бабушки, т.к. в строке 26 в графе «наличие» стоит слово «нет».

При поиске в таблице папа, за один шаг попадая в нужную строку при помощи индекса-номера бочонка, узнавал ответ на вопрос: есть бочонок с заданным номером у бабушки в лотошном мешке или нет.

8.4. Сложный индекс

В примере с индексным поиском про бабушкин лотошный мешок в качестве индекса использовались номера бочонков, т.е. ин-

декс *определялся* равным номеру на бочонке. В примере про записную книжку Пети индекс *вычислялся* по первой букве фамилии ученика. Однако и в том, и в другом случае процесс определения (вычисления) индекса был достаточно простым.

Далее, если в задаче про лотошный мешок не требовалось дополнительно использовать другие алгоритмы поиска и весь процесс поиска завершался за один шаг, напротив, в задаче про записную книжку за один шаг завершался индексный поиск и затем производился последовательный поиск по странице с фамилиями учеников, начинающимися на ту же букву. Как хорошо было бы, если бы фамилии друзей Пети начинались на разные буквы, т.е. чтобы на каждой странице книжки была только одна фамилия. Тогда поиск нужной фамилии осуществлялся бы за *один* шаг. Идеально было бы придумать некоторый процесс, называемый *сложным индексированием* и для каждого элемента множества ставящий в соответствие *уникальное значение индекса*. Если снова обратиться к задаче про записную книжку, то станет видно, что процесс вычисления индекса по первой букве фамилии здесь не подойдет, т.к. даже на приведенной в примере странице на букву "Е" уже имеется более одной фамилии:

Еленев Николай - 888-13-12
Евстратов Петя - 888-66-69
Елкин Гена - 888-45-45

Можно попробовать в качестве подходящего процесса вычислять индекс по первым 2-м, или лучше 3-м, буквам фамилии.

Пусть записная книжка — это таблица из $32 \times 32 \times 32 = 28\,672$ строк, буквам поставлены в соответствие числа от 1 до 32 (вторыми и третьими буквами фамилии могут быть любые буквы из алфавита, возможно лишь выбрасывание буквы «ё», однако, чтобы не усложнять вычислений, можно считать, что и первая буква фамилии тоже может быть любой):

1 – А	9 – И	17 – Р	25 – Ш
2 – Б	10 – Й	18 – С	26 – Щ
3 – В	11 – К	19 – Т	27 – Ь
4 – Г	12 – Л	20 – У	28 – Ы
5 – Д	13 – М	21 – Ф	29 – Ъ
6 – Е	14 – Н	22 – Х	30 – Э
7 – Ж	15 – О	23 – Ц	31 – Ю
8 – З	16 – П	24 – Ч	32 – Я

Страница "Ш" из задачи про записную книжку выглядела так:

Шевцов Алексей - 888-13-11
Шилинговская Вера - 888-18-67
Шарафетдинов Ахмет - 888-78-87

Надо вычислить индекс каждой фамилии. Например, Шевцов (значимы только первые три буквы — ШЕВцов), три первые буквы заменяются соответствующими числами: «Ш» — 25, «Е» — 6, «В» — 3, или $25 \times 32^2 + 6 \times 32 + 3 = 25\,795$. Таким образом, строка, на которой будет находиться телефон Шевцова, имеет номер 25 795. Аналогично вычисляются и индексы-номера строк для других фамилий:

ШИЛинговская — $25 \times 32^2 + 9 \times 32 + 12 = 25\,900$;
ШАРафетдинов — $25 \times 32^2 + 1 \times 32 + 17 = 25\,649$.

При таком сложном индексировании поиск фамилии будет происходить за *один шаг*, например, если записная книжка-таблица устроена следующим образом:

индекс фамилии	телефон
1	
...	
25649	888-78-87
...	

индекс фамилии	телефон
25815	888-13-11
...	
25900	888-18-67
...	

Такое «табличное» устройство не только позволяет за один шаг искать нужную фамилию и телефон, но и имеет ряд других положительных качеств:

а) все фамилии *упорядочены* в таблице по алфавиту;
б) фамилии «ШИЛинговская» и «ШИМинговская» расположены рядом в таблице, что является прямым следствием пункта (а).
в) если фамилия отсутствует в книжке, то ее индекс *однозначно* указывает место, на котором должна находиться эта фамилия.

Однако имеется и ряд трудностей использования таблицы-книжки:

а) фамилии "ШИЛинговская" и "ШИЛинговский" должны располагаться *в одной и той же* строке (эта «неприятность» называется *коллизией*);
б) таблица имеет *очень* большой размер. Заполнение ее даже на один процент является проблематичным.

Упражнения:

31. Подсчитайте индексы в таблице-книжке для следующих фамилий:

Еленев Николай - 888-13-12
Евстратов Петя - 888-66-69
Елкин Гена - 888-45-45

32. Попробуйте расположить в таблице-книжке, вычисляя индекс

по первым двум буквам, следующие фамилии:

а) Шевцов Алексей - 888-13-11
Шилинговская Вера - 888-18-67
Шарафетдинов Ахмет - 888-78-87

б) Еленев Николай - 888-13-12
Евстратов Петя - 888-66-69
Елкин Гена - 888-45-45

Оглавление

Учебное издание

ЛЕОНОВ Александр Григорьевич
ПОДОЛЬСКАЯ Нина Аркадьевна
ЭПИКТЕТОВ Михаил Геннадьевич

Базы данных и электронные таблицы

Курс "Информационная культура"
Книга для учащихся 8 класса

Технический редактор М. Н. Кращина

ОАО "Корпорация "Федоров"
443086, г. Самара, а/я 10374
Лицензия ЛР N 063811 от 5.01.95

Подписано в печать 3.11.97. Формат 84x108 $^1/_{32}$. Бумага типографская.
Гарнитура "Школьная". Печать офсетная. Усл. печ. л. 4,2.
Тираж 16500 экз. Заказ 6639.
Отпечатано с оригинал-макета в типографии издательства
"Самарский Дом печати". 443086, г. Самара, пр. К. Маркса, 201.