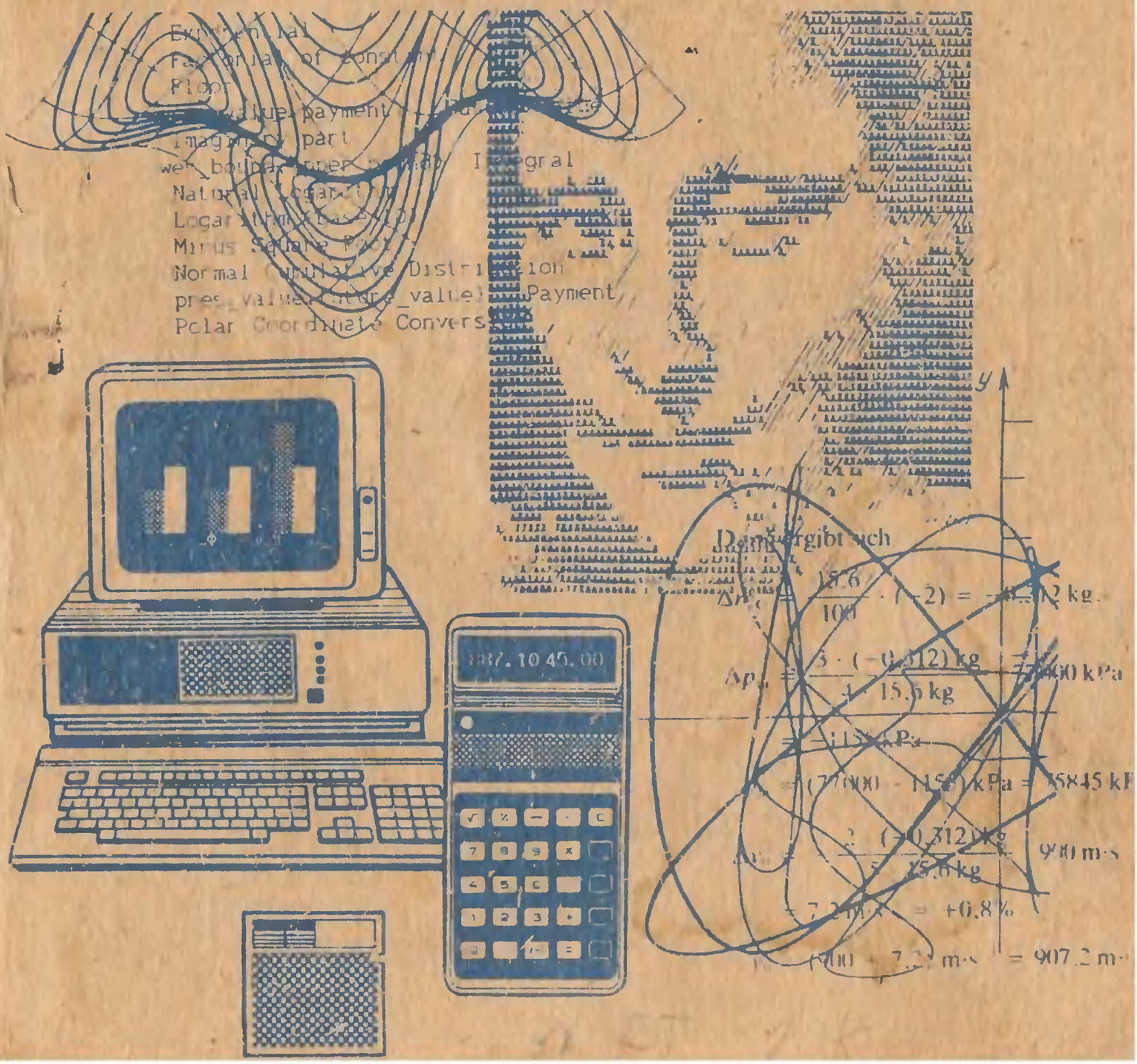
В.Ф.Очков В.А.Хмелюк

От микрокалькулятора к персональному компьютеру



В.Ф.Очков В.А.Хмелюк

От микрокалькулятора к персональному компьютеру

Москва
Издательство МЭИ
1990 г.

ББК 32.973
О-957
УДК 681.3

Рецензент И. Д. Коган
Редактор Л. Г. Никольская

Очков В. Ф., Хмелюк В. А.
О-957 От микрокалькулятора к персональному компьютеру /Под ред. А.Б. Бойко. – М.: Изд-во МЭИ, 1990. – 224 с.: ил.

ISBN 5-7046-0002-6

Книга поможет освоить программирование на микрокалькуляторах серии "Электроника", познакомит со многими "секретами" их работы, не раскрытыми в "Руководстве по эксплуатации". Излагаются принципы составления программ на Бейсике и Паскале для персональных компьютеров, методики решения игровых и деловых программ на программируемых микрокалькуляторах и персональных компьютерах не только языковыми средствами, но и в современных программных средах.

Для широкого круга читателей (от школьников до научных сотрудников), желающих освоить программируемый микрокалькулятор и использовать полученные знания при работе с персональным компьютером.

$О \frac{2404060000-006}{097(02)-90}$ КБ – 7 – 22 – 89 ББК 32.973

Художник *А. М. Драговой*
Художественный редактор *А. Ю. Землеруб*
Технический редактор *Н. Н. Хотулева*
Корректор *В. В. Сомова*
Операторы *Л. В. Сидорова, М. С. Новикова, Е. Б. Ухловская*
Набор и верстка выполнены на компьютерах Издательства МЭИ

Подписано в печать 16.05.90 Л – 35278 Формат 70×100/32 Гарнитура Таймс.
Печать офсетная. Усл. печ. л. 9,03. Усл. кр.-отт. 9,1. Уч. изд. л. 13,24
Тираж 150 000 экз. Заказ 431. Цена 2 р. 65 коп.

Издательство МЭИ 105835, ГСП. Е-250. Красноказарменная, 14.

Типография № 4 Госкомпечати СССР
Москва, Б. Переяславская ул., 46

© Очков В. Ф., Хмелюк В. А., 1990

ISBN 5-7046-0002-6

ПРЕДИСЛОВИЕ

Книга адресована читателям, которые, с одной стороны, желали бы совершенствоваться в работе с программируемыми микрокалькуляторами, а с другой, – хотели бы освоить персональный компьютер, используя опыт работы с микрокалькуляторами.

В книге рассматриваются такие темы, как

начальные и углубленные сведения о программируемых микрокалькуляторах семейства "Электроника",

"секреты" аппаратной и программной частей микрокалькуляторов,

программирование для персонального компьютера на языках Бейсик и Паскаль,

работа на персональном компьютере с использованием готовых программных сред.

В книге раскрываются "маленькие хитрости" программируемых микрокалькуляторов, не отмеченные в инструкциях по эксплуатации.

Основы программирования в кодах микрокалькулятора и на языках персонального компьютера даются на примере решения задач, с которыми сталкиваются инженеры, студенты и школьники, т. е. та категория пользователей, на какую, по мнению авторов, и рассчитана книга. Читатель найдет в ней и игровые программы, которые помогут ему не только скрасить досуг, но и лучше познакомиться с тонкостями организации диалога "человек–ЭВМ".

Чтобы переход от программирования на микрокалькуляторах к программированию на персональных компьютерах был по возможности плавным, в книгу включены трилингвы – программы решения одной и той же задачи, написанные в кодах микрокалькулятора, на языках Бейсик и Паскаль. Для выделения алгоритма решения задач и иллюстрации принципов структурного программирования многие программы вписаны в рамки структурных диаграмм.

Обращается внимание читателей на те особенности программирования для персонального компьютера, какие не встречаются при работе с программируемым микрокалькулятором: проек-

тирование структуры данных, обработка текстовых переменных, работа с файлами на магнитных дисках и др.

Завершает книгу рассказ о популярных программных продуктах (среда программированния Eureka, электронные таблицы и деловая графика пакета Framework III, надстройки над дисковой операционной системой Norton Commander и PCTOOLS), позволяющих работать на персональном компьютере, не прибегая к кодированию на каком-либо языке программирования. Этим подчеркивается, что искусство общения с компьютером – в первую очередь не умение составлять новые программы, а способность эффективно использовать богатое программное обеспечение ЭВМ.

Через всю книгу проходит идея "мирного сосуществования" вычислительных средств разной "весовой категории" (микрокалькуляторы, персональные компьютеры, большие ЭВМ) при решении разноплановых задач. Первая глава книги написана В. А. Хмелюком, вторая – В. Ф. Очковым.

Авторы благодарят редакторов книги Л. Г. Никольскую и А. Б. Бойко и рецензента И. Д. Когана, затративших много сил при доработке рукописи, а также выражают признательность школьникам Алексею Рубцову и Игорю Хмелюку (из подмосковного города Мытищи) и Михаилу Рахаеву (из Москвы) за большую помощь в проверке программ.

Мы будем благодарны за отзывы и замечания по книге, которые следует направлять по адресу: 105835, Москва, Красноказарменная, 14, Издательство МЭИ.

ГЛАВА 1

ПРОГРАММИРУЕМЫЕ МИКРОКАЛЬКУЛЯТОРЫ

ПЕРВОЕ ЗНАКОМСТВО

КОРОТКО О БОЛЬШОМ СЕМЕЙСТВЕ

Сегодня микрокалькулятор практически заменил логарифмическую линейку, стал надежным помощником инженера, конструктора, студента и школьника, занял позиции, которые прежде удерживал арифмометр в плановых отделах и бухгалтериях, потеснил такой универсальный инструмент, как счеты, прослуживший человечеству верой и правдой много веков, позволил расширить круг задач, решаемых непосредственно на рабочем месте.

Главная причина популярности микрокалькуляторов – простота обращения с ними. Работать с этой электронной машинкой столь просто, что правила вычислений без труда запоминают люди, не имеющие специальной подготовки. Освоить навыки вычислений на простейшем микрокалькуляторе, заменяющем счеты, можно значительно быстрее, чем на них – в считанные минуты.

Портативность микрокалькуляторов (их лучшие образцы не больше записной книжки и могут несколько тысяч часов работать от внутреннего источника энергии) при массовом выпуске и сравнительно невысокой цене, несомненно, способствовала их широкому распространению.

Немаловажным является и то, что выпускается множество специализированных моделей, например: микрокалькуляторы – записные книжки, – секретари, – астрологи, микрокалькуляторы для любителей спортивных соревнований (помогающие подсчитывать очки в ходе состязаний) и для аранжировки мелодий (не только позволяющие аранжировать введенную в них мелодию, но и способные выдавать ее в ритме рока, вальса, марша, самбы, румбы и т. д.)

И все же широкое распространение микрокалькуляторов связано прежде всего с тем, что они стали удобными помощниками человека. Например, электронные секретари способны запоминать указания своего владельца и время, когда следует напомнить об этих делах. В нужный момент раздается музыкальный сигнал, а на дисплее появляется надпись-напоминание. В таких микрокалькуляторах непрерывно работают часы и календарь.

Микрокалькуляторы, предназначенные для бизнесменов, экономистов, статистиков, способны хранить в своей электронной памяти обширный набор различных программ для расчета рентабельности, затрат, экономических и статистических показателей.

Наибольшими возможностями обладают программируемые микрокалькуляторы, работающие по программе пользователя. Владелец такого микрокалькулятора не обязательно должен уметь программировать. Фирмы, производящие микрокалькуляторы, обычно предлагают обширный набор стандартных программ.

Выпускается немало различных по своим возможностям моделей программируемых микрокалькуляторов. Например, фирмой "Panasonic" был создан программируемый микрокалькулятор с обширной внутренней памятью, возможностью подключения блоков внешней памяти, с модулем для работы по телефонным линиям, принтером для вывода результатов на бумагу, адаптером, позволяющим подключить к микрокалькулятору домашний телевизор, чтобы использовать его как экран дисплея. Этот микрокалькулятор, однако, стоил дороже персонального компьютера IBM PC.

Предметом нашего разговора будут микрокалькуляторы "Электроника МК-61 и -52", получившие в нашей стране широкое распространение. Говоря о них, нельзя забывать и о модели "Электроника Б3-34", выпускавшейся ранее в большом количестве и совместимой с новыми моделями по программному обеспечению "снизу–вверх"*.

По своей архитектуре, т. е. в данном случае по внутреннему устройству, отечественные программируемые микрокалькуляторы являются последовательной вычислительной машиной. Такую архитектуру легко организовать относительно простыми средствами. Платой за это стало низкое быстродействие – две арифметические операции в секунду. Впрочем этого вполне достаточно для решения разнообразных задач.

---

* Совместимость "снизу–вверх" означает, что любую программу, написанную для Б3-34, можно ввести в микрокалькуляторы МК-52 и -61. Работать будут программы, в которых используются только команды, указанные в "Руководстве по эксплуатации Б3-34".

Первый отечественный программируемый микрокалькулятор "Электроника Б3-21", выпущенный в конце 1970-х годов, уже давно не производится. Все последующие модели программируемых микрокалькуляторов (Б3-34, МК-54, -61 и -52) унаследовали от него функциональную схему и лишь незначительно отличаются друг от друга.

По признаку программной совместимости принято различать два типа программируемых микрокалькуляторов "Электроника". К первому относятся микрокалькуляторы Б3-21, МК-46, -47 и -64, ко второму – Б3-34, МК-54, -56, -61 и -52*.

Функциональная схема программируемого микрокалькулятора показана на рис. 1.1. Нетрудно заметить на ней элементы, присущие универсальной ЭВМ, хотя, конечно, аналогичные по назначению, они значительно различаются своими возможностями. Подключение внешних устройств, расширяющих функции микрокалькулятора, предусмотрено только у модели МК-52.

Основные технические данные микрокалькуляторов Б3-34, МК-52 и -61, а также популярных моделей HP-41C, TI-58C и -59, созданных около десяти лет назад известными американскими фирмами "Hewllet-Packard" и "Texas Instruments", приведены в табл. 1.1.

Программируемый микрокалькулятор ориентирован прежде всего на инженера и научного работника и позволяет вести расчеты, не требующие большого объема памяти, высокого быстродействия и сложного программирования. Но беспорно, есть у микрокалькулятора и свои преимущества перед ЭВМ. Прежде всего они проявляются, когда требуется решить сиюминутную задачу. Творческая работа нередко идет и дома. В уютной домашней обстановке можно заниматься решением многих задач с помощью "карманного помощника".

Отечественные микрокалькуляторы (за исключением МК-52) не имеют энергонезависимой памяти, способной хранить информацию после того, как выключено питание. Это означает, что программа, которая была введена в микрокалькулятор и отлажена, будет потеряна при хотя бы кратковременном отключении питания. И если программа понадобится вновь, ее придется еще раз вводить, набирая с помощью клавиатуры всю последовательность команд и исправляя допущенные по ходу ввода ошибки (а чем длиннее программа, тем больше вероятность их появления).

Составление программы имеет смысл лишь тогда, когда расчеты многократно повторяются (например, для получения

---

* В отечественной литературе обычно говорят о семействе Б3-21 и семействе Б3-34. Некоторые авторы выделяют модели МК-61 и -52 в отдельное, третье, семейство.

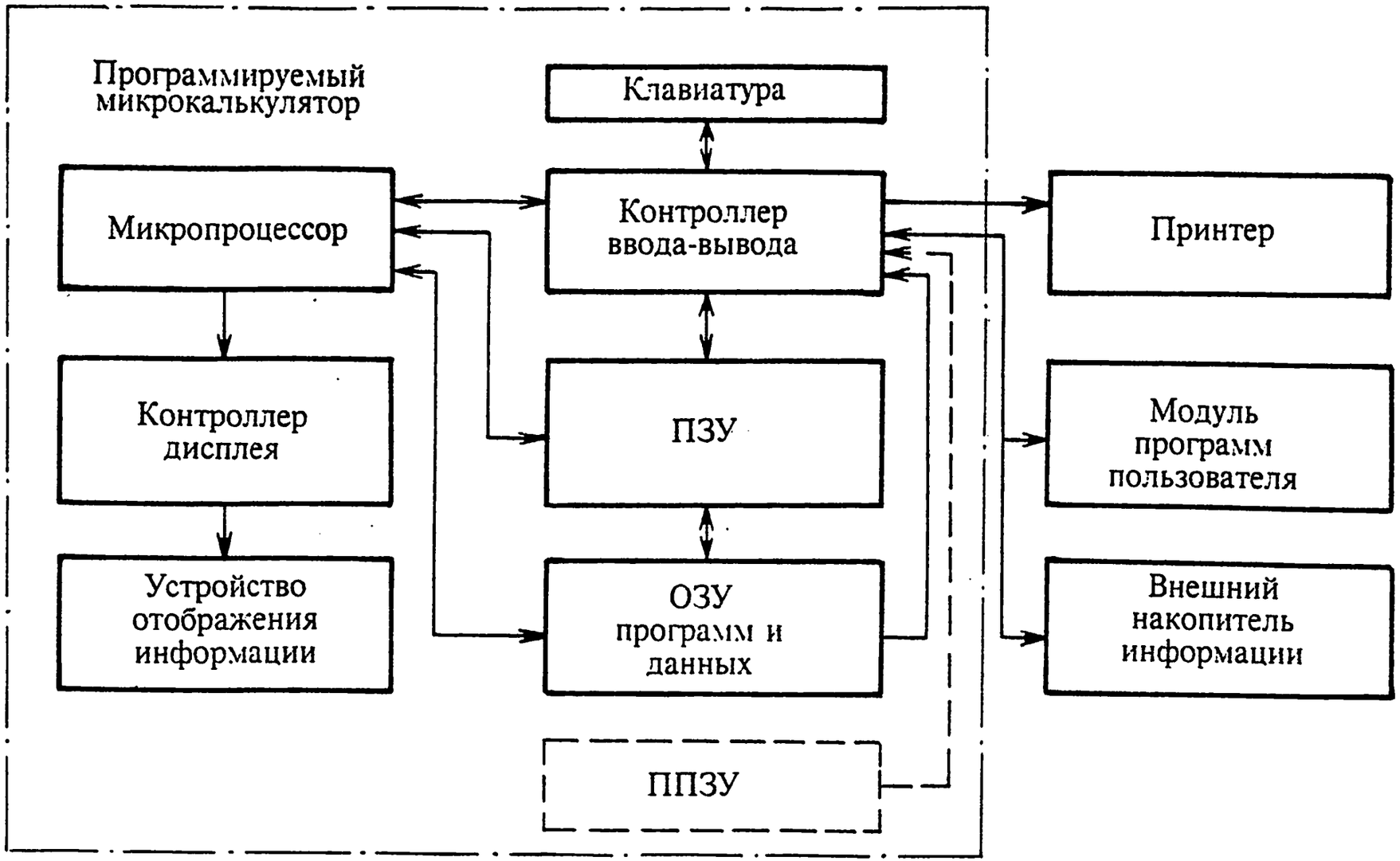


Рис. 1.1. Функциональная схема программируемого микрокалькулятора

ОЗУ, ПЗУ, ППЗУ – оперативное, постоянное и перепрограммируемое ("полупостояннное") запоминающее устройство соответственно

Таблица 1.1

Основные характеристики распространенных отечественных и зарубежных программируемых микрокалькуляторов

Параметры	Б3-34	МК-61	МК-52	HP-41C	TI-58C	TI-59
Число шагов программы	98	105	105	448	480	960
Количество регистров памяти	14	15	15	63	60*	100*
Количество операций, производимых автоматически	51	65	65	130	108	108
Логика вычислений	Обратная				Алгебраическая	
Число разъемов для подключения периферийных устройств	0	0	1	2	1	6
Габариты, мм	185 × 100 × 48	167 × 78 × 36	212 × 78 × 42	144 × 79 × 33	168 × 80 × 36	162 × 80 × 36
Масса, кг	0,39	0,25	0,25	Нет свед.	0,35	0,35

* Возможно перераспределение емкости ОЗУ между данными и программой по "цене" 10 регистров данных за 80 шагов программы.

таблиц, решения сложных уравнений или поиска альтернативных решений)*.

Другой способ хранения отлаженных программ – запись их на магнитную ленту, подобно тому, как это делалось в универсальных ЭВМ 1970-х годов. Ленты для этого требуется совсем немного – несколько сантиметров. Отрезок ленты наклеивают на пластиковую полоску, получая так называемую магнитную карту. Магнитофон для записи и считывания информации с магнитных карт в этом случае встраивают непосредственно в корпус микрокалькулятора. Иногда микрокалькулятор снабжают только разъемом для подключения магнитофона, а его выполняют в виде отдельного блока. Возможность работать с магнитными картами заложена в микрокалькуляторах HP-41C и TI-59.

Немаловажный параметр программируемых микрокалькуляторов – тип индикатора.

В отечественных моделях используют катодолюминисцентные индикаторы. Они обладают приятным зеленовато-голубым свечением. К числу их недостатков принято относить хрупкость (чувствительность к ударам и даже к сильным сотрясениям), а также достаточно большое энергопотребление, связанное с постоянным расходом энергии на разогрев нити накаливания.

Модели семейства HP традиционно снабжаются жидкокристаллическим индикатором. Его достоинство – исключительно малое потребление энергии, что позволяет долгое время работать от одного комплекта батарей, а недостаток – зависимость контрастности изображения от внешнего освещения.

В моделях .семейства TI обычно используют светодиодные индикаторы, потребляющие довольно много энергии, зато более простые в управлении и ударопрочные.

Сравнивая микрокалькуляторы, перечисленные в табл. 1.1., следует заметить:

зарубежные модели работают почти в десять раз быстрее микрокалькуляторов семейства "Электроника" ("платой" за проигрыш в быстродействии является высокая экономичность конструкции наших машинок – отметим хотя бы то, что в них вместо дорогих микросхем памяти используются регистры сдвига, которые значительно дешевле);

подключение печатающего устройства в наших моделях не предусмотрено, а зарубежные модели снабжены для этого специальным разъемом.

---

* Созданное программное обеспечение для отечественных микрокалькуляторов – это обширный набор готовых программ для многих математических, технологических, инженерных и научных задач, а также для игр. В списке рекомендуемой литературы – издания, в которых приведены разнообразные программы для микрокалькуляторов.

Разумеется, зарубежные разработчики не остановились на перечисленных моделях. Новые микрокалькуляторы обладают такими возможностями, что нередко на верхней панели "суперкалькуляторов" можно встретить надпись "карманный компьютер".

Готовятся к выпуску и новые мощные микрокалькуляторы отечественного производства.

В любом деле есть свои крайности – поклонники микрокалькуляторов, например, способны составить головоломную программу, которая даст результат на вторые сутки непрерывного счета. Некоторые счастливые обладатели ЭВМ, напротив, только раздражаются при упоминании о микрокалькуляторах, считая их игрушкой, непригодной для серьезного дела, и забывая об их экономичности и низкой цене.

Постараемся не увлекаться и предоставим компьютеру компьютерово, а все остальное – микрокалькулятору. Впрочем, этого "остального" достаточно много.

Кстати, микрокалькулятор – самое "демократичное" из электронных вычислительных устройств, поэтому он интересен еще одной большой группе пользователей – тем людям, которые умеют обращаться с ЭВМ, но пока что имеют к ней ограниченный доступ или лишены его вовсе. Именно эти пользователи, знающие языки высокого уровня, первыми стали "перебрасывать мостик" между программируемыми микрокалькуляторами и ЭВМ.

УЧИМСЯ ПРОГРАММИРОВАТЬ

Создание первых электронных вычислительных устройств было связано с необходимостью решить проблему ввода в них управляющих воздействий. Дело в том, что логические устройства – своего рода "кирпичики", из которых состоит ЭВМ, – имеют два устойчивых состояния: либо устройство открыто (тогда оно представляет собой малое сопротивление току), либо оно закрыто (его сопротивление велико, и ток по нему практически не протекает). Этим двум состояниям электронного устройства условились поставить в соответствие символы "0" и "1". Их же приписывают и командам "открыть" или "закрыть" устройство.

Таким образом, чтобы общаться с электронным вычислительным устройством, оператор должен был овладеть языком, состоящим из мозаики нулей и единиц – так называемым машинным языком.

В этом случае для общения с машиной оператору необходимо досконально знать не только тонкости языка, но и

архитектуру используемого вычислительного устройства. Машинные языки сдерживали развитие и использование ЭВМ, так как являлись барьером между пользователем-непрограммистом и машиной.

Вскоре машину научили исполнять функции переводчика. Специальная программа-транслятор превращает команды пользователя, записанные на более сложном условном языке, в последовательность машинных кодов. Часть памяти ЭВМ пришлось выделить для хранения программ-трансляторов. Теперь условный язык стал ближе к человеческому, точнее к языку математики. Профессия программиста и оператора ЭВМ из уникальной постепенно стала массовой.

Язык общения с ЭВМ, как и любой язык, имеет свой словарный фонд, или лексикон. Каждое слово лексикона – это символы одной или нескольких команд, которые приводят к выполнению какой-либо операции. Часто эти слова называют операторами.

Существует много языков программирования, в которых ЭВМ управляется командами-словами, как правило, заимствованными из естественного языка, чаще всего английского. Такие языки программирования называют языками высокого уровня. Трансляция программ с них требует от вычислительного устройства больших объемов памяти.

У микрокалькуляторов ресурс памяти невелик – для транслятора в них нет места. Наиболее приемлемым для микрокалькуляторов оказался кодово-символьный язык (промежуточный между машинным языком и языком высокого уровня). Указания в нем выдаются не командами-словами, а комбинациями нажатий клавиш.

Клавиатура микрокалькулятора "Электроника МК-61" схематически показана на рис. 1.2.

Для выполнения написанной на клавише команды достаточно нажать эту клавишу. Команду, указанную под или над клавишей, отдают нажатием префиксной клавиши F или K, а затем функциональной клавиши. Какую клавишу нажать (F или K, окрашенные соответственно в желтый или синий цвет), зависит от того, каким цветом обозначена используемая команда на панели микрокалькулятора. Различные комбинации тридцати клавиш пульта позволяют вводить 65 команд. Список этих команд приведен в Приложении 1, а их коды – в Приложении 2.

Каждая команда кодируется двумя цифрами шестнадцатиричной системы счисления, в которой написание первых десяти цифр (0 – 9) не отличается от принятого в десятичной, а натуральным числам от 10 до 15 соответствуют латинские буквы A, B, C, D, E, F. Каждая из этих букв означает одну шестнадцате-

ричную цифру. В программируемых микрокалькуляторах семейства "Электроника" цифры от A до E на индикаторе изображаются символами: —, L, Г, Г, E. Цифре F соответствует пробел (отсутствие символа), что в некоторых случаях оказывается неудобным.

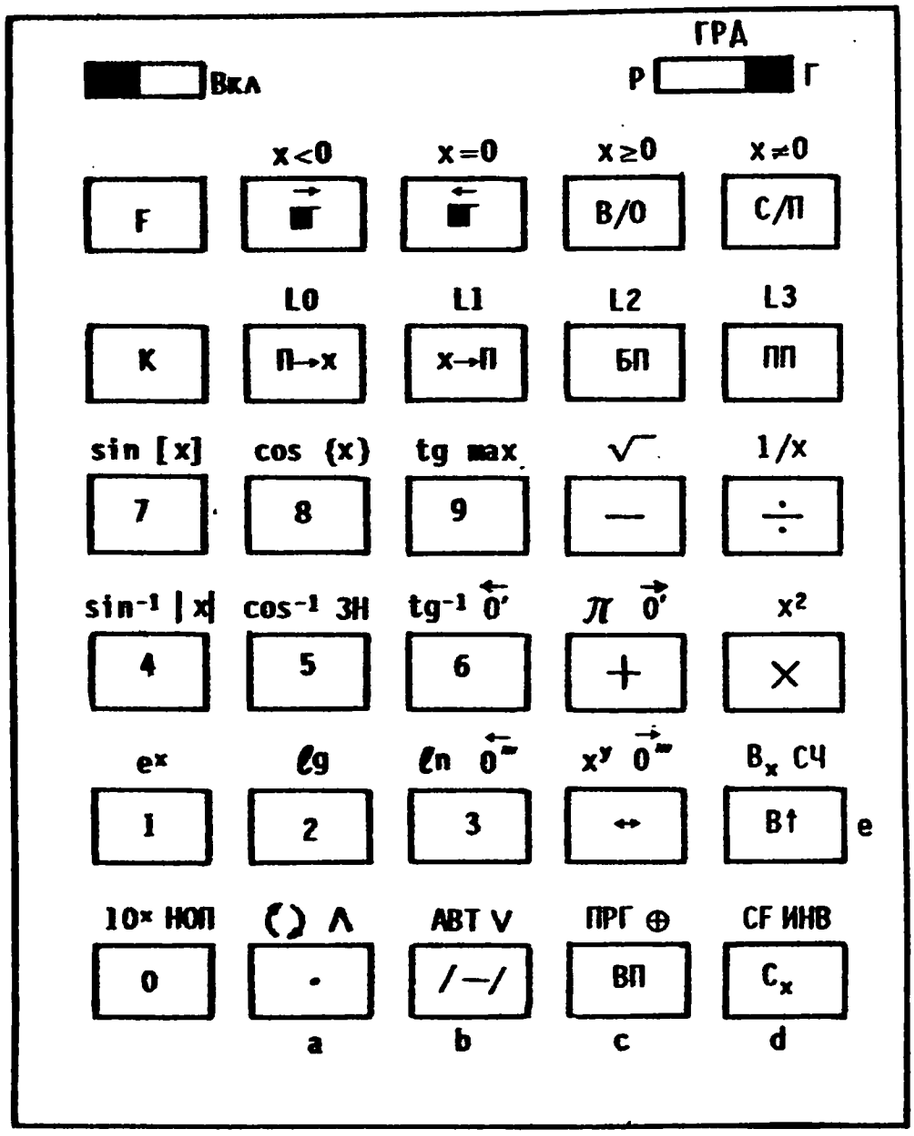


Рис. 1.2. Клавиатура микрокалькулятора "Электроника МК-61"

Перевод кода команды микрокалькулятора в последовательность машинных команд гораздо проще, чем превращение в машинные команды слов любого языка высокого уровня, и требует значительно меньших ресурсов памяти вычислительного устройства.

Коды операций в программируемых микрокалькуляторах "Электроника Б3-34, МК-52, -54, -56, -61" одинаковые. Однако обозначения ряда команд – символы, нанесенные на клавиатуре, – для некоторых операций не совпадают. В целях унифи-

кации в программах для микрокалькуляторов будем использовать некоторые символы языков высокого уровня (табл. 1.2).

Таблица 1.2

Обозначение некоторых операций кодово-символьного языка

Операция	На клавиатуре микрокалькулятора		В книге
	Б3-34	МК-52, -54, -56, -61	
Запись копии числа из регистра X в регистр M	ПМ	$x \to$ ПМ	ПМ
Вызов копии числа из регистра M в регистр X	ИПМ	П $\to$ xM	ИПМ
Обмен содержимым между регистрами X и Y	XY	$\leftrightarrow$	XY
Продвижение информации в стеке "вверх"	↑	B↑	↑
Кольцевое передвижение информации в стеке	F↻	F↻	F →
Умножение	×	×	*
Деление	÷	÷	/
Вызов копии числа π в регистр X	F π	F π	FPI
Вызов результата предыдущей операции в регистр X	F BX	F BX	FBX
Очистка регистра X	C_x	C_x	CX
Вычисление функций			
x^2	F x^2	F x^2	FX^2
$\sqrt{x}$	F $\sqrt{\ }$	F $\sqrt{\ }$	F SQR
x^y	F x^y	F x^y	FX^Y
e^x	F e^x	F e^x	F EXP
10^x	F 10^x	F 10^x	F EXT
lg$x$	F lg	F lg	F LG
ln$x$	F ln	F ln	F LN
arcsin$x$	F arcsin	F $\sin^{-1}$	F ASN
arccos$x$	F arccos	F $\cos^{-1}$	F ACS
arctg$x$	F arctg	F tg^{-1}	F ATN
$x \geq 0$	F $x \geq 0$	F $x \geq 0$	F X > 0
$x \neq 0$	F $x \neq 0$	F $x \neq 0$	F X <> 0

Продолжение таблицы 1.2

Операция	На клавиатуре микрокалькулятора		В книге
	Б3-34	МК-52, -54, -56, -61	
Перевод*			
минут и их десятичных долей в десятичные доли часа (градуса)	—	→ КО′	КМ →
десятичных долей часа (градуса) в минуты и десятичные доли минуты	—	← КО′	КМ ←
минут, секунд и десятичных долей секунды в десятичные доли часа (градуса)	—	→ КО′″	КМС →
десятичных долей часа (градуса) в минуты, секунды и десятичные доли секунды	—	← КО′″	КМС ←

* Эти команды есть только в МК-52 и -61.

В программируемых микрокалькуляторах семейства "Электроника" вычисления организованы с так называемой обратной бесскобочной логикой, связанной с порядком ввода команд и данных: вводится первое число, оно перемещается во второй операционный регистр командой ↑, затем вводится второе число, и лишь тогда дается команда исполнить ту или иную операцию. Например, при умножении 1,2 на 3,4 последовательность нажатий на клавиши будет такой:

1 · 2 ↑ 3 · 4 ×

На индикаторе появляется результат*

" 4.08 "

Обратная бесскобочная логика – дань экономному устройству микрокалькуляторов. В ряде случаев она позволяет обойтись меньшим количеством нажатий клавиши, чем при использовании алгебраической системы. Это особенно заметно при вычислении сложных выражений с большим количеством вложенных скобок. К тому же команда ↑ необходима лишь тогда, когда два или несколько чисел вводятся с клавиатуры друг за другом. Если же число является результатом операции или вызывает-

* В дальнейшем результат, заключенный в кавычки, будет означать – "на индикаторе".

ся из регистра памяти, движение информации происходит автоматически.

Бесскобочная логика аппаратно реализуется с помощью стека – устройства памяти, благодаря которому удается обходиться без скобок. Освоить вычисления с обратной бесскобочной логикой несложно, но для этого необходимо хорошо представлять работу стека. Перемещение информации в стеке при исполнении различных команд показано на рис. 1.3.

Стек состоит из операционных регистров X, Y, Z, T и регистра X1. Содержимое регистра X – это то, что в большинстве случаев отображается на индикаторе. Нажимая клавишу ↑ (рис. 1.3, *а*), мы выталкиваем информацию из верхнего регистра стека T, а информация из регистра X при этом копируется в регистр Y. Если теперь набрать на клавиатуре новое число (рис. 1.3, *б*), то оно займет место того числа, которое до этого было на индикаторе и в регистре X.

При вызове информации из любого регистра памяти M в регистр X происходит перемещение, показанное на рис. 1.3, *в*. Казалось бы вызов константы π (рис. 1.3, *г*) не должен отличаться от операции вызова из памяти. Но при этой операции информация из регистра X перемещается не только в регистр Y, что закономерно и понятно, но и в регистр X1, уничтожая его прежнее содержимое.

Кстати, в "Руководстве по эксплуатации" об этом ничего не сказано. Впрочем, это лишь одна из многих загадок, которые подготовили для поклонников программируемых микрокалькуляторов те, кто их разрабатывал.

Обмен информацией между регистрами X и Y (рис. 1.3, *д*) также обновляет запись в регистре X1, перемещая туда копию числа, ранее находившегося в регистре X.

Вычислительные операции, выполняемые микрокалькулятором, по использованию стека можно разделить на два типа: одно- и двухместные.

Операции, в которых участвуют два операнда (сложение, вычитание, деление, умножение), называются двухместными. Перемещение информации в стеке при такой операции показано на рис. 1.3, *е*. Характерное смещение информации "вниз" по стеку создает большие вычислительные удобства, что несомненно оценит на практике каждый читатель.

Операции вычисления функций (за исключением x^y) называются одноместными, потому, что в них участвует только аргумент функции, находящийся в регистре X (рис. 1.3, *ж*).

Информацию в стеке можно продвинуть по кругу (рис. 1.3, *з*) – в регистре X1 информация при этом замещается информацией из регистра X.

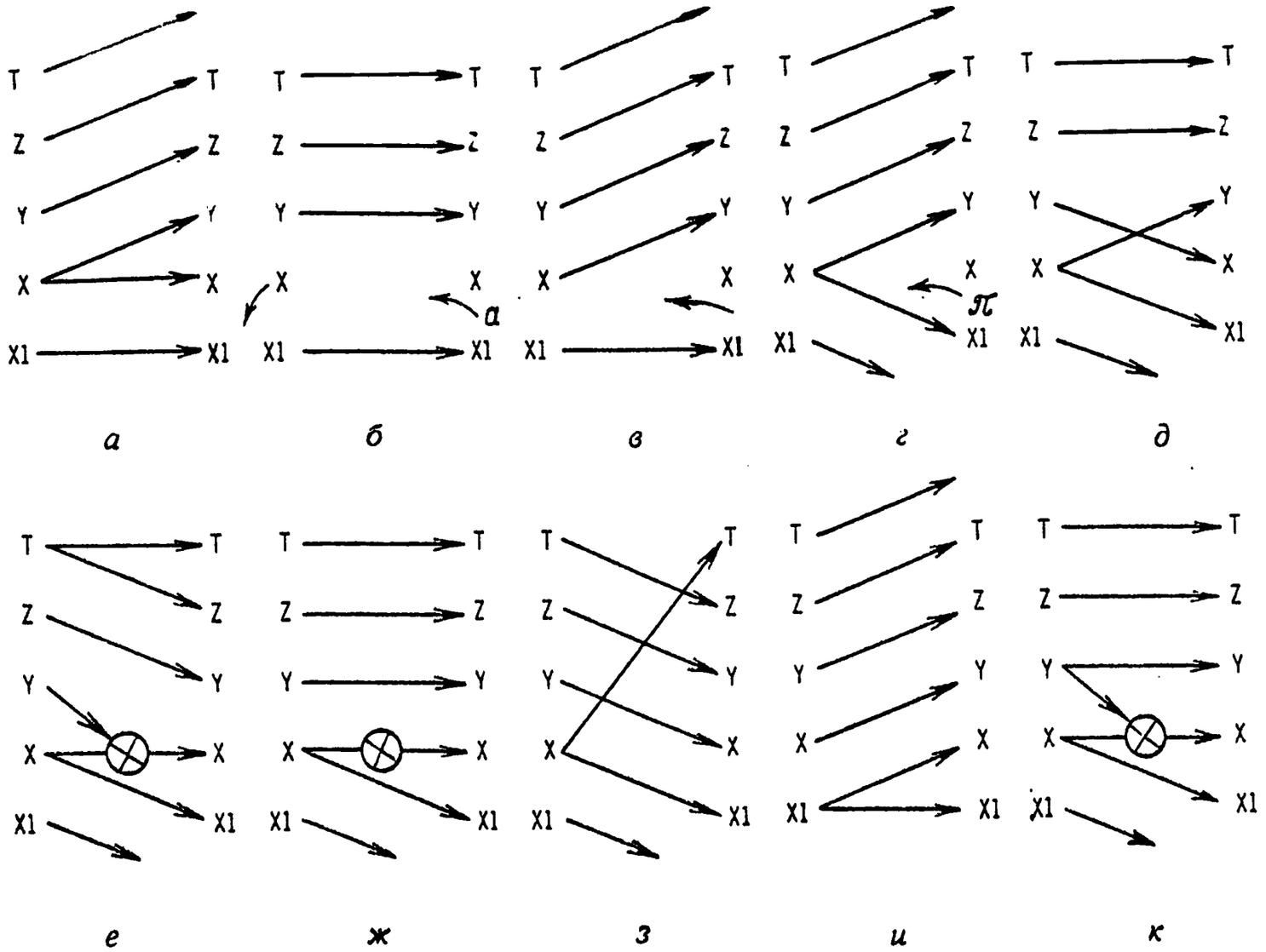


Рис. 1.3. Передвижение информации в стеке микрокалькулятора при выполнении различных команд

Внимательный читатель обратит внимание, что при любых вычислениях информация из регистра X перемещается в регистр X1. Именно поэтому регистр X1 называют регистром предыдущего результата: информацию из него можно вернуть в регистр X с помощью команды FBX (рис. 1.3, *и*).

Читатель, по-видимому, знает, что программа – это перечень команд машине, записанных в той последовательности, в которой их следует выполнять. Хранится программа в ячейках памяти оперативного запоминающего устройства. Пара шестнадцатиричных цифр – код операции и одновременно шаг программы.

Если для микрокалькулятора двух цифр достаточно, то, записывая программы на бумаге, каждый оператор удобно дополнять порядковым номером и его кодом. Это позволяет вводить программу практически без ошибок – каждый шаг программы можно проверить, не заглядывая в Приложение 2.

Такая форма записи используется в литературе по программированию. Распространен и другой способ записи, когда в каждой строке записывают по десять операторов, не указывая номера шагов и коды. Иногда в написании операторов опускают и обозначения префиксных клавиш.

Чтобы подготовить микрокалькулятор к вводу программы, нажимают клавиши В/О, F и ПРГ. Команда В/О устанавливает счетчик шагов программы на нуль, а команда F ПРГ переводит микрокалькулятор в режим ввода программы. Показания счетчика шагов программы – "00" высвечиваются на индикаторе в двух правых знакоместах.

Если ввести любой оператор, то его код появится в двух левых знакоместах, а номер шага увеличивается на единицу. Код каждого нового оператора продвигает коды предыдущих слева направо, а сам занимает два освободившихся слева знакоместа.

В режиме программирования шесть полей индикатора представляют собой бегущую строку, вмещающую три кода, последние символы которой выталкиваются вновь введенными.

Если программа набрана, ее текст можно проверить и отредактировать. Для этого служат клавиши ШГ (шаг вправо) или ШГ (шаг влево). Стрелка указывает направление движения строки. При каждом нажатии на эти клавиши на единицу изменяются номер шага и содержимое информационной строки. Счетчик шагов все время показывает номер того оператора, код которого высвечивается на индикаторе слева.

Может случиться, что в процессе отладки программы какая-нибудь команда окажется ненужной. Исключить команду нетрудно, если ввести вместо нее команду К НОП – "НЕТ ОПЕРАЦИИ".

Останавливается исполнение программы, например для ввода или вывода результатов, командой С/П.

Казалось бы, пора составлять первую программу. Но не будем торопиться. Прежде чем писать программу, предназначенную для решения какой-либо задачи, необходимо продумать последовательность этапов ее решения.

Попробуем сделать это на примере вычисления факториала. Из курса математики мы знаем, что $n! = 1 \times 2 \times 3 \times \ldots \times n$.

Надеемся, что читателю знакомо понятие "алгоритм". Напомним, что в схеме алгоритма овалами обозначают начало и конец, в параллелограммах – операции ввода и вывода данных, в прямоугольниках – все вычислительные операции, а в ромбах – различные условия, влияющие на последовательность исполнения программы. В алгоритме "Факториал-1" (рис. 1.4, *а*) присутствуют условия: 0! = 1, 1! = 1 и умножение предыдущего результата на величину k, увеличивающуюся после каждого очередного умножения на единицу. Это происходит до тех пор, пока $k < n$. При $k = n$ вычисления прекращаются, поскольку последнее произведение и есть $n!$. Этот алгоритм реализует программа 1.1.

Программа 1.1

"Факториал-1"

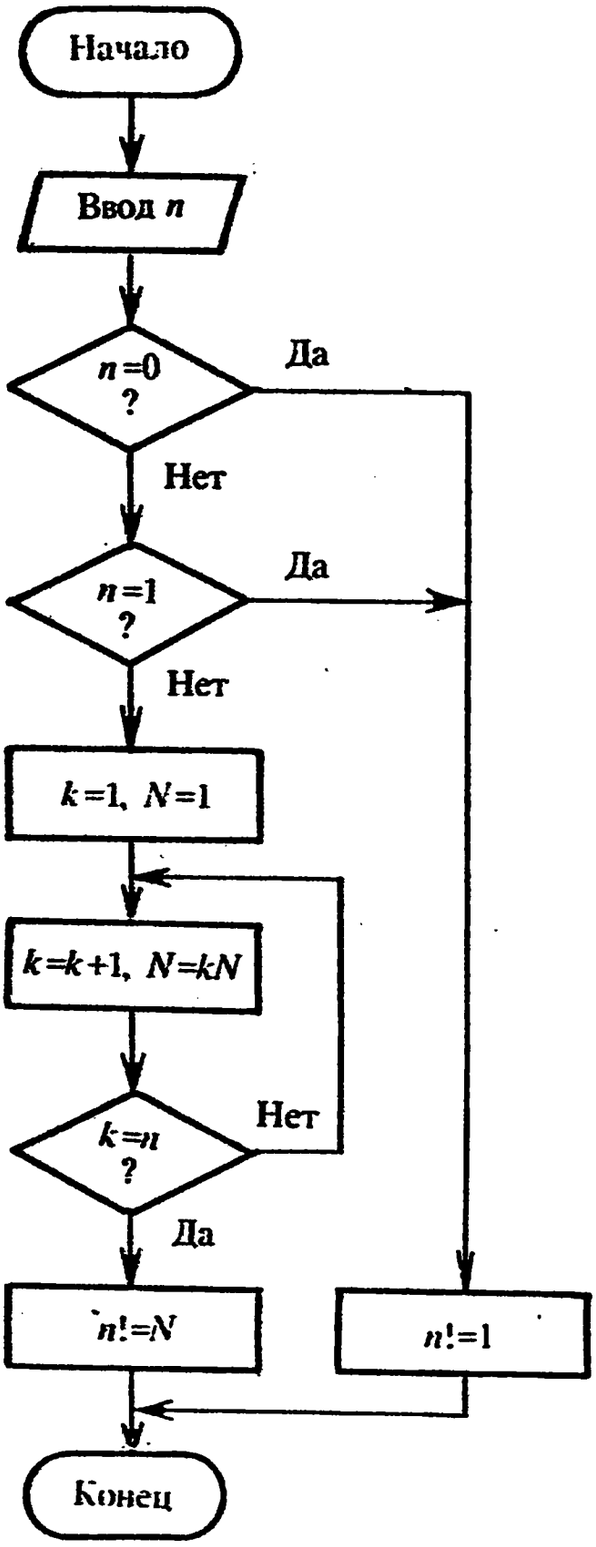
	00	01	02	03	04	05	06	07	08	09
00	FX<>0	24	П0	1	-	FX<>0	24	1	П1	П2
10	ИП2	1	+	П2	ИП1	*	П1	ИП2	ИП0	-
20	FX=0	10	ИП1	С/П	1	С/П				

Мы столкнулись с повторением какого-то фрагмента программы до тех пор, пока не будет выполнено заданное условие (в данном случае им является $k = n$). Такую последовательность действий принято называть циклом.

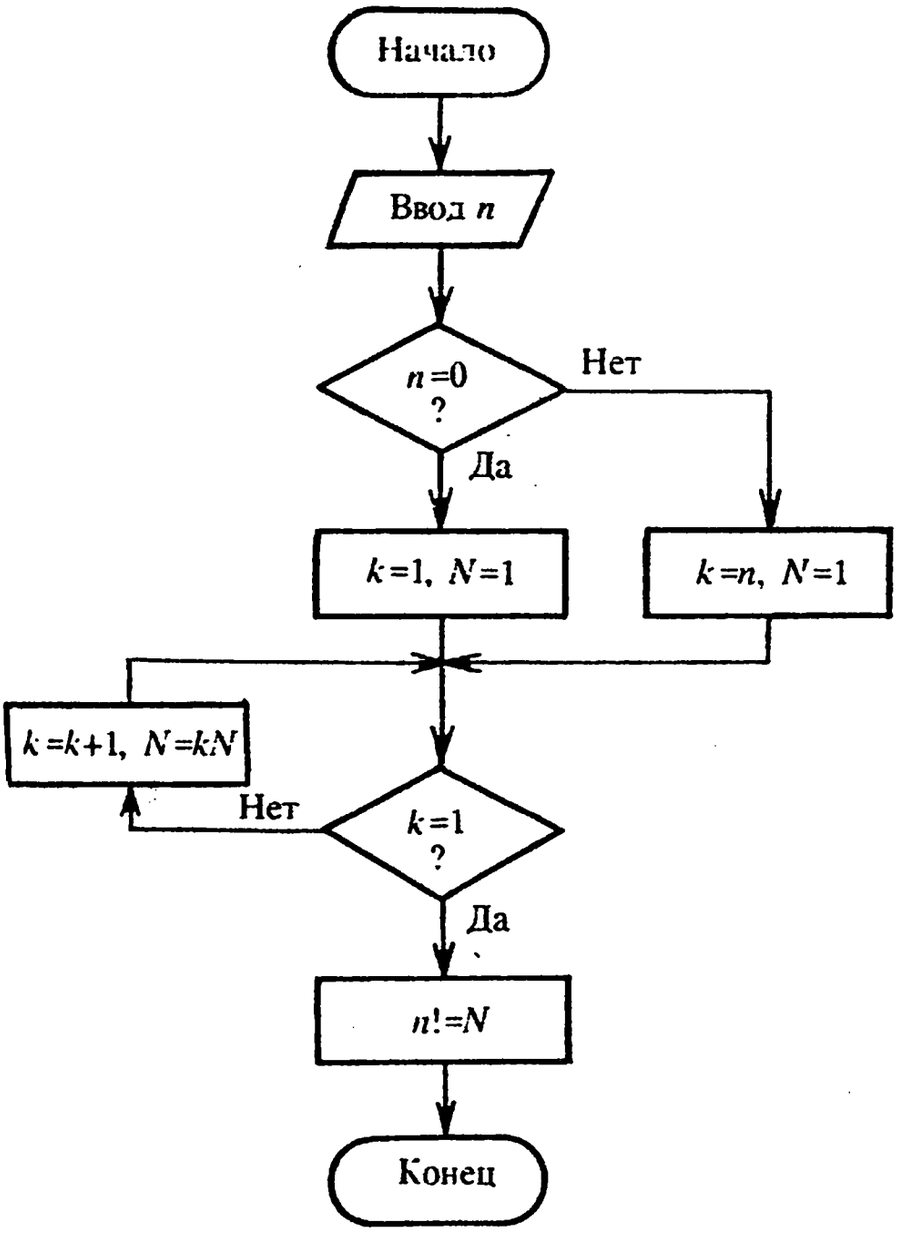
Циклы, выход из которых осуществляется в соответствии с тем или иным условием, можно организовать с помощью команд

FX<0 AD, FX>0 AD, FX=0 AD, FX<>0 AD .

Нужно помнить, что во всех отечественных микрокалькуляторах переход по адресу AD происходит тогда, когда условие не выполнится. В большинстве зарубежных микрокалькуляторов и во всех языках высокого уровня для ЭВМ переход осуществляется в том случае, если условие выполняется. В противном случае адрес перехода игнорируется и продолжается исполнение программы.


Начало
Ввод n
n=0 ?
Да
Нет
n=1 ?
Да
Нет
k=1, N=1
k=k+1, N=kN
k=n ?
Нет
Да
n!=N
n!=1
Конец

а



б

Рис. 1.4. Алгоритмы "Факториал-1" (а) и "Факториал-2" (б)

Возвратимся к нашей задаче. Говоря о цикле при вычислении $n!$, мы знаем, что умножение необходимо совершить $n-1$ раз, если $n \geq 2$. Предыдущее произведение каждый раз умножают на сомножитель, увеличенный на единицу. Чтобы организовать такой цикл (с фиксированным числом повторений) используют команду FLN AD. Здесь N = 0, 1, 2 или 3 – номер регистра памяти, где записано число повторений цикла; AD – номер шага программы, с которого начинается повторяемый участок. Завершает его команда, образующая цикл. Алгоритм "Факториал-2" вычисления $n!$ при помощи команды FLO приведен на рис. 1.4, *б* и реализован в программе 1.2.

Программа 1.2

"Факториал-2"

	00	01	02	03	04	05	06	07
00	ВП	ПО	1	ИПО	*	FLO	03	С/П

Программы для микрокалькуляторов можно документировать по-разному. В черновом варианте автор программы иной раз записывает только команды. В технической литературе их нередко дополняют номерами шагов (см. программы 1.1 и 1.2) – так программу удобнее вводить. Иногда, чтобы помочь тем, кто хочет разобраться в чужой программе, ее текст дополняют кодами команд и комментариями.

Запишем программу 1.2 двумя способами.

Полная запись:

Шаг	Оператор	Код	Комментарий
00	ВП	0С	Команда ВП превращает 0 в 1 и не изменяет других чисел
01	ПО	40	Запись числа циклов в регистр памяти 0
02	1	01	Запись 1 в регистр X
03	ИПО	60	Вызов в регистр X чисел n, n-1, n-2, ..., 1
04	*	12	Операция умножения
05	FLO	5Г	Команда организации цикла
06	03	03	Адрес начального шага цикла
07	С/П	50	Команда останова

Краткая запись:

ВП ПО 1 ИПО * FLO 03 С/П

Особо стоит пояснить роль команды ВП. Если с клавиатуры введено любое положительное число, команда не производит никакого действия. Но в случае, если таким числом будет

N = 0, то команда ВП превратит 0 в 1. К сожалению, в "Руководстве по эксплуатации", прилагаемому к микрокалькулятору, об этом не пишут.

Для отладки любой программы удобно использовать тесты – контрольные задачи, ответ на которые известен. Нашу программу проверим вычислениями 0! и 3!*:

```
0 В/О С/П "1"  ,
3 В/О С/П "6"  .
```

Тесты помогли нам за несколько секунд убедиться в правильном функционировании программы.

Работу с программой можно упростить. Нажатие клавиши В/О, предшествующее нажатию клавиши С/П, необходимо для того, чтобы исполнение программы начиналось каждый раз с адреса 00. Если предполагается вычислять по программе несколько раз, то ее завершают командой 08.БП 09.00. Эта команда каждый раз будет возвращать управление к началу программы. И только при первом расчете необходимо нажать клавиши В/О и С/П. Для всех последующих вычислений требуется после ввода нового значения N нажимать лишь одну клавишу С/П.

Команда безусловного перехода БП AD, обеспечивает передачу управления на шаг AD. В качестве AD могут выступать любые две цифры в интервале 00, ..., 99, –0, ..., –4**. Команду БП AD используют всякий раз, когда надо прервать естественное исполнение программы (в порядке линейного возрастания номеров шагов) и перейти на шаг с номером AD.

В программах для вычисления n! использованы команды записи в регистр памяти и вызова из регистра памяти. В микрокалькуляторе Б3-34 таких регистров 14, в МК-61 и -52 их 15. Регистры нумеруют цифрами от 0 до 9 и буквами от A до D (E)***. В том случае, когда речь идет о произвольном регистре, договоримся обозначать его буквой M. Команда ПМ – запись в регистр M – стирает текущее содержимое регистра M и помещает туда новое значение из регистра X. Команда ИПМ – вызов из регистра M – копирует на индикатор и в регистр X содержимое регистра M. При этом число в регистре M остается неизменным.

Существуют и другие операции работы с регистрами памяти: КПМ и КИПМ, так называемые команды косвенной адресации. Сложное название скрывает за собой простую и очень полезную операцию. В регистре M хранится число – номер того регистра,

---

* Напоминаем, что число в кавычках – это результат, считываемый с индикатора.

** В модели Б3-34 используют числа от 00 до 98.

*** Регистр E имеется лишь в моделях МК-61 и -52.

к которому мы обратимся, чтобы извлечь или поместить в него информацию.

Рассмотрим, например, команду КП7. Она поместит содержимое регистра X в тот, номер которого укажет число в регистре 7. Номер регистра – целое положительное число от 0 до 14. Если в регистре будет не целое число, то при косвенном обращении через этот регистр дробная часть будет отброшена, и в нем останется лишь целая часть исходного числа. Это свойство команд косвенной адресации нередко используется для получения целой или дробной частей чисел в микрокалькуляторе Б3-34, не имеющем для этого специальных команд.

Иногда косвенную адресацию используют для организации переходов. Так же, как и в случае с прямой адресацией, различают безусловную косвенную адресацию (команда КБПМ) и условную косвенную адресацию (команда КХ<?>ОМ). Знаком "<?>" обозначено условие (= , < , ≥ , ≠).

Нередко какой-то фрагмент программы необходимо повторять, но не в цикле. Этот фрагмент выделяют в так называемую подпрограмму – последовательность команд, завершающуюся командой В/О. Для передачи управления подпрограмме служат команды прямого обращения к ней (ПП AD) или косвенного (К ПП М). В первом случае управление переходит к команде, записанной по адресу AD, во втором – по адресу, который хранится в регистре М.

Если число было записано в одном из регистров 0 – 3, то при каждом косвенном обращении через этот регистр, целая часть числа будет уменьшаться на единицу. В регистрах 4 – 6 целая часть, напротив, будет увеличиваться, а в регистрах с 7 по D(E) останется неизменной, сколько бы раз через них ни обращались.

Основой механизма обращения к подпрограммам является адресный стек. Каждый раз, когда управление передается на подпрограмму, после ее исполнения управление необходимо возвратить на команду, следующую за командой обращения к подпрограмме. Для этого и служит адресный стек. По команде ПП в него записывается адрес той команды, к которой необходимо будет вернуться по команде В/О, завершающей подпрограмму.

Адресный стек по своему устройству напоминает уже известный нам стек из регистров X, Y, Z и T. Есть, однако, и отличия:

во-первых, кроме ПП и В/О, нет других команд, управляющих его поведением;

во-вторых, глубина стека составляет не четыре, а пять регистров.

Глубина стека определяет, сколько подпрограмм можно вложить одну в другую, подобно матрешкам. Всего таких вложений может быть четыре.

Освоив основы программирования, так и хочется составлять программы – одну за другой. Если требуется решить какую-то задачу и возможности микрокалькулятора позволяют это, то не стоит тратить часы на то, чтобы усовершенствовать программу, сделать ее эффективнее (к примеру, чтобы получать результат не за 20, а за 10 мин). И лучше не торопитесь, а подумайте над задачей. Этот навык пригодится при решении сложных задач, когда придется составлять программу, используя все ресурсы компьютера.

Итак, подумаем над тем, как решать задачу из рассказа А.П. Чехова "Репетитор".

"Купец купил 138 арш. черного и синего сукна за 540 руб. Спрашивается, сколько аршин купил он того и другого, если синее стоило 5 руб. за аршин, а черное 3 руб.?"

«"Это задача на неопределенные уравнения..." – думает репетитор – гимназист Зиберев. "Эта задача, собственно говоря, алгебраическая," – говорит он... "И без алгебры решить можно," – отвечает наблюдающий за занятиями сына отставной губернский секретарь Удодов, "протягивая руку к счетам и вздыхая. – "Вот, извольте видеть..." Он щелкает на счетах и у него получается 75 и 63, что и нужно было».

Конечно, задача легко решается алгебраически. Обозначив число аршин черного сукна через X, синего через Y, достаточно решить систему уравнений с двумя неизвестными:

$$X+Y=138,$$

$$3X+5Y=540.$$

Программу для решения этих уравнений составить несложно.

Однако попробуем представить, как решал задачу губернский секретарь. Скорее всего он воспользовался одним из приемов решения таких задач на счетах. Вот его алгоритм:

1. Пусть `Ч` – количество аршин черного сукна.
2. Определим количество аршин синего сукна как `C=138-Ч`.
3. Пусть `Ч=Ч+1`, а `C=C-1`.
4. Проверим условие `3*4+5*C<>540` (отличны ли стоимость двух отрезков и заплаченная сумма?). Если условие не выполняется, т.е. `3*4+5*C=540`, то следует, что задача решена. Если условие выполняется, то вновь возвращаемся к шагу 3.

Отметим, что в этом алгоритме предполагается, что `Ч > C`. Это сделано, чтобы упростить нашу программу, которую назовем "Купец". Однако желающим можно предложить составить программу, свободную от этого ограничения. Мы вернемся к задаче, "о сукне", чтобы решить ее в среде Eureka, а сейчас при-

ведем запись программы "Купец" на языке микрокалькулятора:

Шаг	Оператор	Код	Комментарий
00	КИП3	Г3	Операция С*=С-1
01	КИП4	Г4	Операция Ч=Ч+1
02	ИП4	64	Вызов числа Ч в РХ**
03	ИП8	68	Вызов в РХ числа Ц=3 – цены аршина черного сукна
04	*	12	Сколько стоит Ч аршин черного сукна – 3*Ч
05	ИП3	63	Вызов в РХ числа С (3*Ч смещается в РУ)
06	ИП9	69	Вызов в РХ числа Ц=5 – цены аршина синего сукна
07	*	12	Сколько стоит С аршин синего сукна – 3*С
08	+	10	Сколько стоит такая покупка
09	ИП7	67	Вызов в РХ суммы, заплаченной за все сукно
10	-	11	Вычисление разности 540 – (3×Ч+5×С)
11	FX=0	5E	Если условие выполнено, мы определили С и Ч
12	00	00	Если нет, то управление переходит на адрес 00
13	ИП3	63	Вызов С
14	ИП4	64	Вызов Ч
15	С/П	50	Окончание расчета: в РХ – Ч, в РУ – С

Инструкция к программе "Купец".

1. Обозначив число аршин черного сукна Ч, а число аршин синего – С, предположим, что Ч=С=138/2.
2. Введем постоянные: Ч в регистр 4, С в регистр 3. Сумму, уплаченную за все сукно, (540) запишем в регистр 7, цену аршина черного сукна – в регистр 8, цену аршина синего сукна – в регистр 9.
3. Запустим программу на счет командами В/О С/П. Время решения задачи около 30 с.
4. После останова в регистре Х и регистре 4 – количество аршин черного сукна (при заданных начальных условиях – 75), в регистре Y и регистре 3 – количество аршин синего сукна (63). Для вызова на индикатор числа аршин синего сукна достаточно нажать клавишу XY или ИП3.

В мире большого программирования бушуют страсти. Программы становятся все сложнее. Торопливое устранение одной ошибки влечет за собой появление других. Программисты порой не способны разобраться в своих собственных программах. Еще не так давно, составляя программу для большой ЭВМ с небольшими по современным понятиям памятью и быстродействием, приходилось исхитряться, пытаясь сделать программу покороче, а время решения поменьше. Сегодня ресурсы новых ЭВМ значительно увеличились, но аналогичные заботы доста-

---

* В этой записи знак равенства показывает, что числу С присвоено старое значение, уменьшенное на единицу.

** Используем сокращение РМ, где Р – регистр, М – его номер.

ются на долю владельцев программируемых микрокалькуляторов. Ограниченный объем памяти и малое быстродействие заставляют искать оптимальные и даже сверхоптимальные методы решения. Иной раз получаются такие головоломные программы, в которых непросто разобраться.

Что же говорить о программах для ЭВМ. Там вопросы понятности, поиск возможных ошибок и их устранение вырастают до проблемы, решить которую не проще, чем составить программу. Один из возможных выходов, по мнению программистов, – структурное программирование. Его суть заключается в том, что программа, как из кубиков, собирается из отдельных блоков, внутренняя структура которых абсолютно не зависит от содержания предыдущих и последующих частей программы.

В языках высокого уровня команда безусловного перехода GO TO (в микрокалькуляторе – команда БП) была объявлена "вне закона". Впрочем, она появилась в некоторых новых версиях тех языков, которые первоначально считались чуть ли не эталонами структурного подхода, например в Паскале.

Запрет использовать при структурном программировании все структуры, кроме следования, повторения (цикл), выбора (альтернатива), привел к тому, что появилась возможность составлять каждый блок и отлаживать его автономно промышленными поточными методами. Заплатить за это пришлось ресурсами ЭВМ – задачи стали отнимать больше машинного времени и для их решения необходимо увеличение объема памяти.

Для мощных универсальных ЭВМ такая плата оказалась приемлемой, а для малышей-микрокалькуляторов – чрезмерной. И хотя мы покажем, как записать программу для микрокалькуляторов в той форме, в какой принято записывать структурные программы, это не должно вводить читателя в заблуждение. Структурные программы для микрокалькуляторов – лишь иллюстрация, призванная облегчить переход от создания программ для микрокалькуляторов к созданию программ для ЭВМ.

В спорах о программировании подчас забывают о том, что оно лишь один из этапов решения задачи на ЭВМ. Если, прочитав условие задачи, тотчас же начать вводить программу в микрокалькулятор, составляя ее по ходу дела и не записывая, то это успеха не принесет. Хорошо, если в ходе исполнения программы, в которой имелась ошибка, произойдет зацикливание или аварийный останов, что привлечет к ней внимание. Хуже, если будет выдан неверный результат, который пользователь посчитает истинным.

Итак, первое, что следует сделать прежде, чем обращаться к микрокалькулятору или компьютеру, – это уяснить задачу. Обратимся еще раз к рассказу А. П. Чехова "Репетитор". Вспом-

ним, как мальчик Петя и его репетитор решали задачу о покупке сукна.

«Петя повторяет задачу и тотчас же, ни слова не говоря, начинает делить 540 на 138. "Для чего же вы это делите? Постойте! Впрочем, так ... продолжайте. Остаток получается? Здесь не может быть остатка. Дайте-как я разделю! ... Это задача на неопределенные уравнения, а вовсе не арифметическая ... Сложить 5 и 3, а потом делить 540 на 8? Так что ли? Нет не то".»

Все это напоминает действия неумелого программиста.

Чтобы избежать серьзных ошибок и глубокого разочарования – спутников неправильно работающей программы, попытаемся сформулировать принципы подхода к решению задачи на ЭВМ, которые пригодны и при решении задач на программируемом микрокалькуляторе.

1. *Постановка задачи.* Программист уясняет задачу, определяет пределы, в которых могут находиться исходные данные и результаты. Если задача сформулирована нечетко, то самое время уточнить условие, добиться однозначной формулировки.

Поясним это на примере решения треугольников. Уточняя условие, определяем, что требуется решить только один вариант, когда заданы три стороны треугольника. Их длины – это исходные данные. Программа должна быть составлена так, чтобы микрокалькулятор проверил, являются ли эти величины сторонамии треугольника, а также вычислил периметр, площадь, углы, радиусы вписанной и описанной окружностей.

2. *Организация работы с данными.* Присвоим исходным величинам обозначения: a, b, c. Аналогично поступим с выходными величинами, или результатами: S – площадь; P – периметр; r – радиус вписанной окружности; R – радиус описанной окружности; A – угол против стороны a; B – угол против стороны b; C – угол против стороны c. Будем считать, что все исходные и выходные величины положительны и вещественны.

3. *Выбор метода решения.* Это ответственный момент работы, который оказывает влияние на все последующие этапы. Для нашего примера метод решения будет таким: вычисления по известным формулам после того, как выполнено очевидное условие – самая большая сторона треугольника должна быть меньше суммы двух других.

Перечислим необходимые формулы:

$P = a + b + c, \quad r = \sqrt{(p-a)(p-b)(p-c)p},$

$A = 2\mathrm{arctg}(r/(p-a)), \; B = 2\mathrm{arctg}(r/(p-b)), \; C = 2\mathrm{arctg}(r/(p-c)),$

$R = a/2\sin A, \; S = rp,$

где полупериметр $p = P/2$.

4. *Алгоритмизация, или структурирование.* На этом этапе выбирают логику работы программы. Внимательно отнестись к нему – значит избежать многих ошибок. Для задачи о треугольнике используем словесно-формульное описание алгоритма:

1. Начало.
2. Найти бо́льшую из трех величин и присвоить ей имя a: двум другим – b и c.
3. Определить, выполняется ли условие $a > b + c$. Если нет, то перейти к шагу 4, а если да – к шагу 11.
4. Вычислить P и сохранить это значение.
5. Вычислить p и сохранить это значение.
6. Вычислить r и сохранить это значение.
7. Вычислить A, B, C и сохранить эти значения.
8. Вычислить R и сохранить это значение.
9. Вычислить S и сохранить это значение.
10. Выдать результаты на индикатор и перейти к шагу 12.
11. Выдать условное сообщение, соответствующее ситуации, когда треугольник с такими сторонами невозможен.
12. Конец.

Разумеется, составление схемы – это лишь начало работы над алгоритмом. Опуская тонкости, отметим, что в дальнейшем для расчета A, B и C целесообразно организовать подпрограмму, так как они вычисляются по одной и той же формуле, в которой изменяется только значение длины противолежащей стороны.

5. *Программирование.* Первый вопрос, на который необходимо ответить самому себе, – на каком языке будет написана программа. Пока речь идет о программируемом микрокалькуляторе, ответ единственный. Если же программа составляется для большой ЭВМ, то вопрос о языке оказывается принципиальным и непростым.

После выбора языка программирования следует вновь просмотреть алгоритм, чтобы проверить, есть ли в нем такие конструкции, которые нельзя записать на этом языке. Необходимо их исключить либо сменить язык. Но можно пересмотреть алгоритм – учтены ли в нем возможности избранного языка. В языке микрокалькулятора, например, нет такого понятия, как рекурсия, поэтому программа игры "Ханойская башня" (будет рассмотрена ниже) содержит множество операций, которые лишь в совокупности реализуют частный случай рекурсии. Запись этой же программы на языке Паскаль, в котором предусмотрены рекурсивные операции, занимает всего несколько строк.

6. *Тестирование и отладка.* Создавая тесты, старайтесь опровергнуть распространенное среди программистов утверждение: "Тест может обнаружить ошибку, но не может доказать, что в программе их нет!" К тому же желательно, чтобы тест проверял и правильность решения задачи.

Тест должен быть таким, чтобы при его исполнении были пройдены все без исключения ветви программы и отработаны все циклы. Если в программе есть переходы, прямая и косвенная адресация, то тест должен дать возможность проследить и их. Необходимо проверить работу программы при вводе недопустимых данных. Например, для треугольника ввести в качестве длины стороны отрицательное число.

Поиск причины, которая привела к ошибке, – дело непростое. В микрокалькуляторах можно воспользоваться командой пошагового исполнения программы – ПП. Зная верное решение, стараются определить ошибки, продвигаясь шаг за шагом. Этот метод гораздо реже используют на больших ЭВМ. Там поиск ошибки облегчают различные средства, например большинство ошибок замечает сама ЭВМ (разумеется, речь идет об особой программе). Впрочем успех во многом зависит от искусства, опыта и интуиции программиста.

Каждый раз, устранив очередную ошибку, необходимо вновь повторить тестирование.

7. *Документирование.* Кому не приходилось попадать в такую, например, ситуацию: надо найти законченную вчера программу, а она, как и многие другие, была записана без названия и комментариев. Глядя на несколько программ, не сразу найдешь нужную.

А сколько времени теряется порой на восстановление или небольшую доработку программы, написанной лишь неделю или несколько дней тому назад.

После двух-трех подобных случаев владельцы программируемых микрокалькуляторов дают себе слово описывать все программы.

Какой же все-таки должна быть домашняя документация?

Начинают с инструкции. Ее составляют так, чтобы ясно было, как вводить исходные данные, запускать программу, получать результаты. Достаточно подробный комментарий необходим для каждого шага программы. Желательно привести тест и указать время его исполнения.

И последний совет: не выбрасывайте описание алгоритма, пусть оно тоже войдет в вашу документацию.

Задача о треугольниках была простой, хотя и там встретилось немало типичных проблем. Рассмотрим более сложный пример, также убеждающий в том, что не стоит составлять программу раньше, чем алгоритм будет обдуман в деталях. Время, потраченное на обсуждение задачи, не пропадет зря.

Познакомимся с правилами игры "Бридж-ит", которая была популярна в Америке в первой половине нашего века. Это игра для двух партнеров. "Красные", соединяя за один ход две любые ближайшиие красные точки (крестики на рис. 1.5, *а*), стремятся продолжить непрерывную ломаную линию сверху вниз, а "зеленые" стараются помешать этому и, в свою очередь, провести непрерывную ломаную слева направо. Нельзя проводить диагональные линии и пересекать ломаную противника своей.

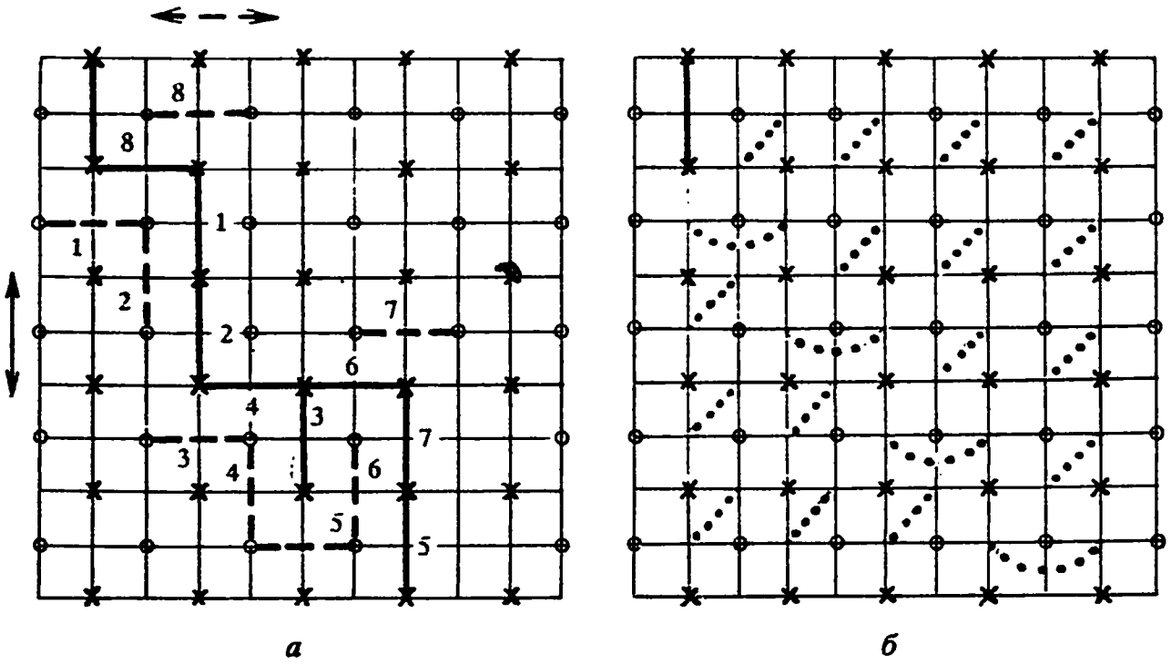


Рис. 1.5. Игра "Бридж-ит"

а – игровое поле; *б* – графическая форма победного алгоритма; сплошные линии – "красные"; штриховые – "зеленые"

Популярность игры была настолько велика, что выпускались наборы – игровая доска и цветные перемычки. В России игру называли "Перекинь мостик" или "Проложи тропку".

В 1951 г. американский кибернетик К. Шеннон изготовил автомат "Птичья клетка", играющий в "Бридж-ит". Первый ход был за автоматом, и он уверенно доводил любую партию до выигрыша.

Вскоре известный американский специалист по теории игр О. Гросс опубликовал победный алгоритм для начинающего игру. Алгоритм был представлен графически с указанием первого хода (рис. 1.5, *б*). Краткий комментарий пояснял, что каждый раз, когда "мостик", проложенный соперником, коснется одного конца линии, "мостик", прокладываемый вами, должен коснуться второго конца этой же пунктирной линии. Человеку такого объяснения достаточно, сложнее научить игре компьютер, по-

скольку он не понимает графических инструкций. Кстати, публикация О. Гросса положила конец выпуску этой игры – кто согласится с ролью статиста, который вводит в автомат ходы и заранее уверен в собственном проигрыше.

Каждый, кто пробовал разрабатывать алгоритм игры "Бридж-ит", согласится, что первое, что приходит в голову, – построить координатную сетку и закодировать каждую точку координатами *x* и *y* (рис. 1.6).

Именно так поступил П. Шмуклер из Хабаровска в своей программе для этой игры ("Наука и жизнь", 1988, № 3). В его программе для микрокалькулятора предполагается игровое поле 11×11. Считается, что первый ход микрокалькулятора уже сделан и игрок должен нанести его на игровое поле. В варианте П. Шмуклера это ход 2.1 – 2.3 (см. рис. 1.6). Такая программа неоптимальна. В самом деле, требуется каждый раз вводить четыре числа, разделяя их ввод нажатием клавиши ↑, причем меньшая координата должна вводиться раньше, чем бóльшая. Программа длинная и медленная: в ней 98 шагов, а ответа приходится дожидаться полминуты.

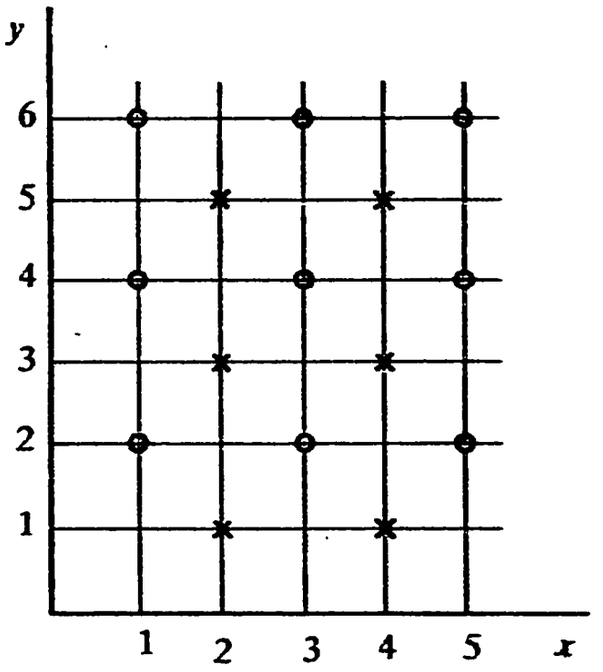


Рис. 1.6. Фрагмент игрового поля игры "Бридж-ит"

Киевляне В. Распопов и С. Кузнецов составили программу для этой игры всего из восьми шагов, не раскрыв, как пришли к такому алгоритму*.

Попробуем разобраться, как они рассуждали. Глядя на алгоритм Гросса (см. рис. 1.5, *а*), задумываешься прежде всего над тем, как выделить диагональные точки и упростить ввод – вывод. Если кодировать не точки, а перемычки, то легко сделать и еще один шаг – одинаково кодировать ходы игрока и автомата. Порядок ходов однозначно определяет, кем из игроков сделан очередной ход (рис. 1.7). Ходу 01 соответствует верти-

* См.: Формування комп'ютерної грамотності учнів. – Київ: Рад. шк., 1987. С. 124 – 137.

кальная линия при ходе автомата и горизонтальная – при ходе человека.

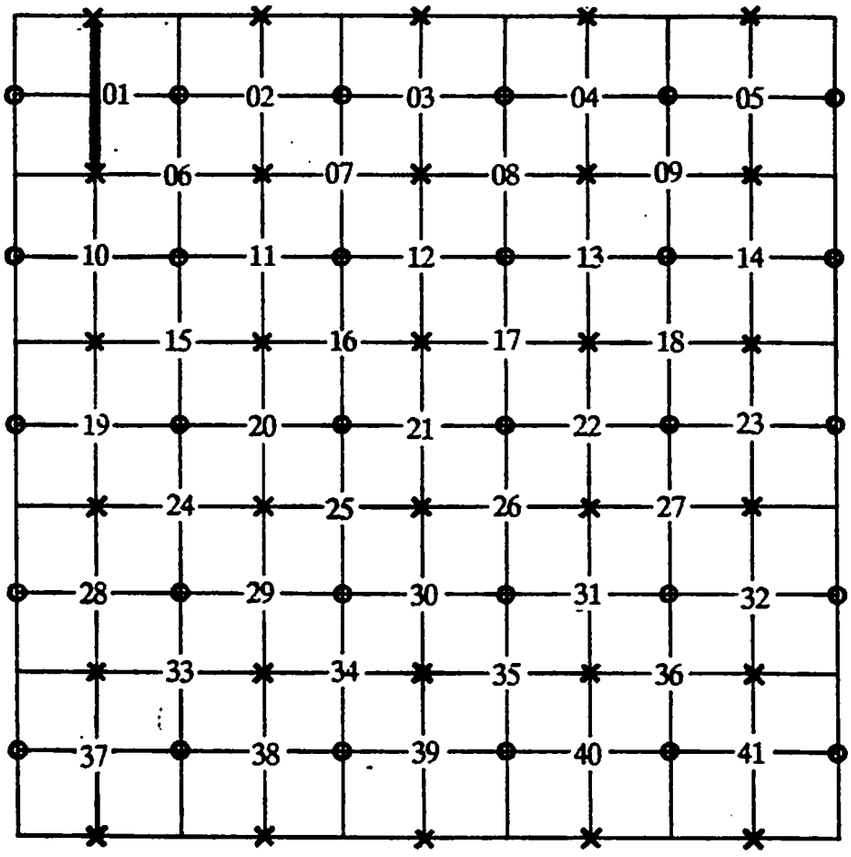


Рис. 1.7. Начало партии в игре "Бридж-ит"

Если бы у микрокалькулятора хватало памяти для хранения всех сделанных ходов, нетрудно было бы предотвратить повторение одним из партнеров хода другого. Этим исключалось бы пересечение линий, запрещенное правилами. Еще одна находка – особое кодирование игрового поля, которое позволило выразить алгоритм в понятной для машины форме (рис. 1.8).

Все ходы вводятся в микрокалькулятор в виде чисел РИ, где И – ход игрока, двузначное число; Р – любое целое число, кроме нуля. Оно необходимо лишь потому, что для микрокалькулятора все нули, встретившиеся на индикаторе слева, до первой отличной от нуля цифры, считаются незначащими. Микрокалькулятор гасит их, поэтому все ходы типа 05 на индикаторе выглядели бы как 5. Ответный ход автомата РА, где Р – число, которое вводил игрок; А – ответ автомата.

Вынужденное неудобство диалога с программой обыграно следующим образом: автомат дает сеанс одновременной игры, а Р – номер доски, с которой автомат играет в данный момент. Автомат, разумеется, не запоминает ходы, он отвечает каждый раз на конкретный ход. Словом, если исключить возможность

того, что человек будет нарушать правила (пересекать ломаную противника), то автомат способен дать сеанс одновременной игры на 999999 досках и всюду выиграть.

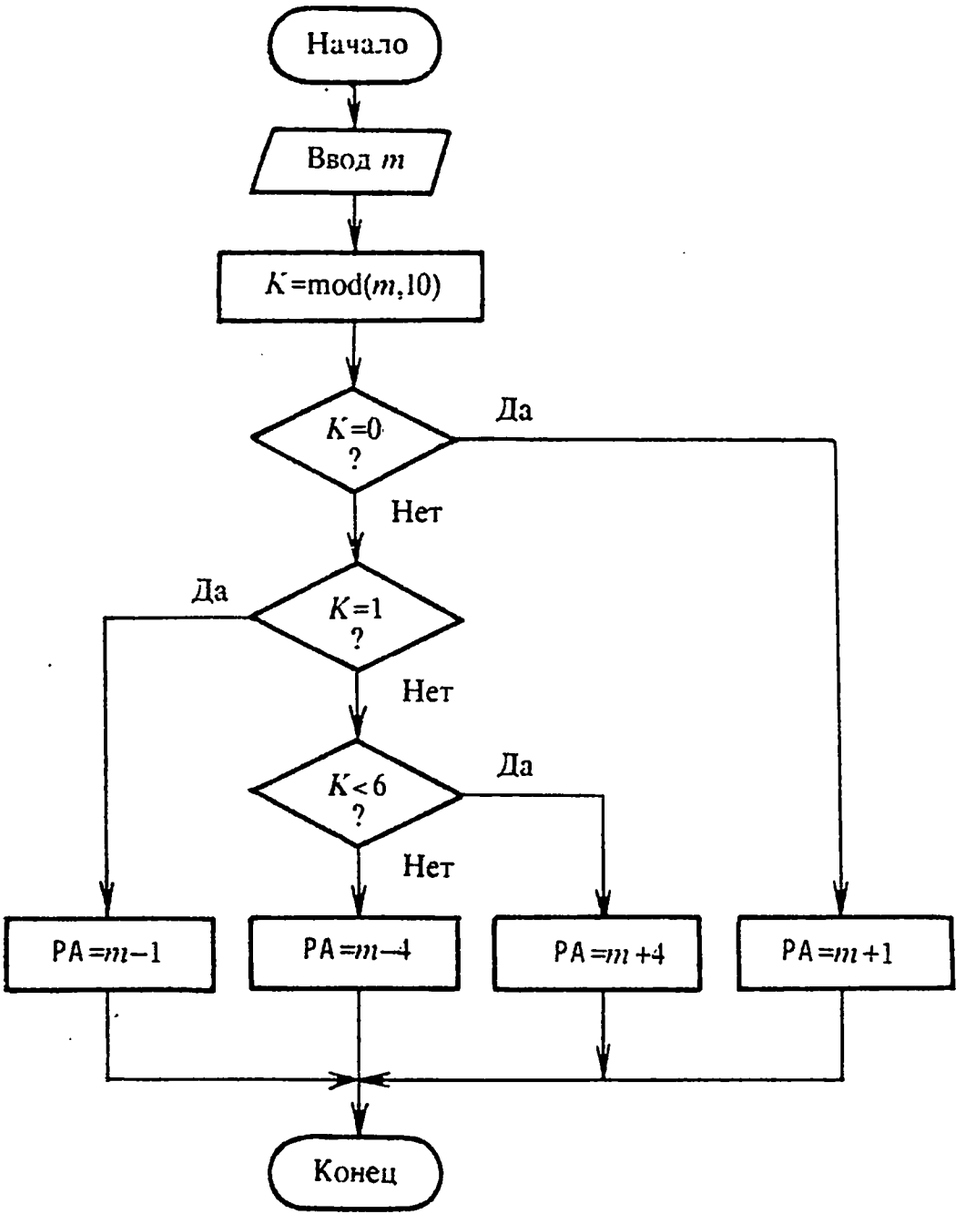


Рис. 1.8. Победный алгоритм Кузнецова-Распопова

PA - ответный ход автомата

Победный алгоритм (см. рис. 1.8) без труда можно запрограммировать для микрокалькулятора – программа укладывается в 40 шагов. Но и это не предел – использование косвенной адресации позволяет сократить программу в пять раз.

Программа 1.3

"Бридж-ит"

	00	01	02	03	04	05	06	07
00	ИП8	+	П9	КИП9	-	С/П	БП	00

Инструкция

1. Исходный ввод констант: 12 – в регистр 8; 8 – в регистры 0 и 1 (для МК-61 и -52 и в регистр Е); 16 – в регистры 2, 3, А, В; 11 – в регистр С.; 13 – в регистр D.

2. Первый ход автомата 01 нанести на игровое поле.

3. Ввести ход в форме Pab, где P – номер доски, на которой играет игрок ($1 \leq P \leq 999999$); ab – код хода игрока. Нажать клавишу С/П (при первом вводе В/О и С/П). Ответ автомата – Pcd.

4. Если игра не закончена, перейти к п. 3.

НАЧИНАЕМ РАБОТАТЬ НА МК-61

Итак, Вы стали обладателем одной из самых совершенных моделей отечественных программируемых микрокалькуляторов "Электроника": МК-61 или -52. Вам приятно сознавать, что по сравнению с Б3-34 увеличилось число шагов программы, добавился один регистр памяти, появились новые операции. Есть, конечно, и минусы. Вместо аккумуляторов в МК-61 элементы А-316, один комплект которых позволяет работать не более трех часов.

Освоить новый микрокалькулятор помогут статьи и книги (см. список рекомендуемой литературы), впрочем ответ на многие вопросы вы сможете найти в этой главе.

Прежде всего возникают трудности с переводом программ. То, что одни и те же операции в микрокалькуляторах Б3-34 и МК-61 обозначаются по-разному – это только полбеды. С проблемой разночтений поможет справиться табл. 1.2. Есть, однако, и другие отличия.

Если попытаться перевести команды КП↑ и КИП↑ (хотя и не указанные в "Руководстве по эксплуатации Б3-34", но иногда встречающиеся в программах для этого микрокалькулятора) как КПЕ или КИПЕ, ничего не получится. В Б3-34 команды "со стрелкой", например КИП↑ , эквивалентны командам косвенного обращения к нулевому регистру (в нашем случае КИП0) с тем отличием, что команды со стрелкой не уменьшают содержимое регистра 0 на единицу. Это как бы доступ в нулевой регистр с "черного хода". В микрокалькуляторе МК-61 такое свойство утрачено. Однако появился дополнительный регистр памяти Е.

Впрочем у владельца МК-61 остается возможность заменить исчезнувшую команду КИП↑ последовательностью других:

ИПО ПЕ F→ КИПЕ .

Аналогично поступают с командой КП↑:

ИПО ПЕ F→ КПЕ .

К сожалению, перевод получается неточным – безвозвратно теряется содержимое регистров T и X1, поэтому необходимо проверить, использовалось ли в программе содержимое одного из этих регистров. Если да, то перевод оказывается сложным.

Когда команды со стрелкой встречаются не один раз, фрагмент, отвечающий за перевод, выносят в подпрограмму. Он занимает лишние шаги программы, но поскольку в МК-61 на семь программных шагов больше, чем в Б3-34, обычно это удается сделать.

Чтобы не менять адреса переходов, вместо каждой команды КИП↑ в программу записывают команду КППЕ. В регистр E помещают 98, а программу дополняют фрагментом:

98. ИПО 99.1 -0.+ -1.ПО -2.F→ -3.КИПО -4.В/О .

Надеемся, что читатели помнят обозначения шестнадцатиричных цифр, а именно, что число 10, т.е. цифра A, в микрокалькуляторе обозначается знаком "–".

Можно предложить и другие варианты перевода, особенно, если в программе не используется содержимое регистров X1, Y и Z.

Составляя сложную программу, старайтесь не забывать о косвенной адресации. Команда КБПМ на шаг короче оператора БПАD. На первый взгляд, кажется, что платой за сохранение одного шага программы, является потеря регистра M, в котором записан адрес. Однако на практике иногда удается обойтись без использования специального регистра..

Дело в том, что адрес перехода определяется только двумя последними цифрами того числа, что хранится в регистре. Таким образом, один и тот же регистр способен хранить константу, все цифры которой используются для вычислений, а две последние – для адресации. Впрочем, когда предпоследняя цифра равна нулю, то переход осуществляется так, как бы он выполнялся, если бы в регистре была одна, лишь последняя, цифра.

Разобраться с переходами поможет табл. 1.3. То, что в ней есть запись по два знака через косую черту, объясняется разли-

чием в конструкции микрокалькуляторов Б3-34 и МК-61. Слева от черты – номер регистра, к которому происходит обращение в Б3-34, справа – в МК-61.

Таблица 1.3

Сложные случаи косвенной адресации

Регистр, в котором записан адрес	Последняя цифра многозначного числа, записанного в адресный регистр*									
	0	1	2	3	4	5	6	7	8	9
0-3	9	A	B	C	D	0/E	1/0	0	1	2
4-6	B	C	D	0/E	1/0	0	1	2	3	A
7-D(E)	A	B	C	D	0/E	1/0	0	1	2	3

* Предпоследняя цифра этого числа не равна нулю.

До сих пор нет единого алгоритма перевода программы для Б3-34 на язык МК-61. С другой стороны, нет смысла отказываться от применения новых команд, которых не было в Б3-34. В табл. 1.4 приведены фрагменты программ, осуществляемые микрокалькулятором МК-61 с помощью встроенных микропрограмм, выполняемых по одной команде, и реализуемых программно на Б3-34.

Таблица 1.4

Перевод некоторых операций микрокалькулятора МК-61 (-52)

Операция	МК-61	Б3-34
Выделение части числа $x > 0$		
целой	K[X]	1 + ПМ⁻ КИПМ⁻ F→ ИПМ⁻
дробной	K{X}	1 + ПМ КИПМ F→ ИПМ –
Определение		
модуля числа x	K|X|	FX<0 _ /–/
функции sign x	K ЗН	FX<>0 _ FX<0 _ 1 /–/ БП _ 1
большего из чисел, находящихся в регистрах X и Y	K MAX	FX<0 _ FBX + FBX БП _ FBX + FBX XY
Перевод		
десятичных долей x в минуты и десятичные доли минуты	←K0′	1 + ПМ КИПМ F→ ИПМ – 0,6 * ИПМ 1 – +

Операция	МК-61	Б3-34
десятичных долей x в минуты, секунды и десятичные доли секунды	$\overleftarrow{\text{КО}'''}$	1 + ПП _ X ПП _ / 1 — С/П ПМ КИПМ F→ ИПМ — 0,6 * ИПМ + 2 FEXT В/О
минут и десятчных долей минуты в десятичные доли часа (градуса)	$\overrightarrow{\text{КО}'}$	1 + ППМ КИПМ F→ ИПМ — 0,6 / ИПМ 1 — +
минут, секунд и десятичных долей секунды в десятичные доли часа (градуса)	$\overrightarrow{\text{КО}'''}$	1 + 2 FEXT * ПП 6 0 / ИПМ 2 FEXT / ПП — ИПМ 1 — + + С/П ПМ КИПМ F→ ИПМ — 6 0 / 2 FEXT * В/О
Логические операции**		
умножение	К ∧	В/2
сложене	К ∨	А — В /2
сложение по модулю 2	К ⊕	А — В

Примечание. M^- – любой из регистров памяти 0 – 3; M – любой из регистров памяти 7 – D. *В регистре X всегда большее число, а в регистре Y – меньшее. **Операции сложения и деления с двоичными числами X и Y проводятся по правилам действий над десятичными числами (например: $X=100$, $Y=101$, $A=X+Y=201$, B – это число A, в котором единицы заменены нулями; в данном случае $B=200$, при этом $X \wedge Y=100$, $X \vee Y=101$, $X \oplus Y=001$).

Использовать новые команды МК-61 при переводе необязательно, но с помощью такой замены нередко удается освободить несколько шагов программы или немного увеличить быстродействие. Если после перевода новая программа стала короче (длиннее), не забудьте скорректировать адреса всех переходов.

Иногда, если речь идет о времени (или угловых величинах), необходимо переводить минуты и секунды в десятичные доли часа (градуса). Дело в том, что микрокалькулятор "не умеет" работать с величинами, выраженными в минутах и секундах.

Соответствующих операций четыре. Каждая из них оставляет неизменной целую часть числа. Если оно состоит из целой части и десятичной дроби, команда $\overleftarrow{\text{КО}'}$ преобразует его в число, состоящее из целой части, минут и десятичных долей минуты. Команда $\overleftarrow{\text{КО}'''}$ представит исходное число в виде целой части, минут, секунд и десятичных долей секунды.

Каждой из команд $\overleftarrow{\text{КО}'}$ и $\overleftarrow{\text{КО}'''}$ соответствует обратная: $\overrightarrow{\text{КО}'}$ и $\overrightarrow{\text{КО}'''}$. Все эти операции являются одноместными, т. е. содержимое регистров Y, Z, T остается неизменным. Аргумент операции – число, которое хранилось в регистре X, отправляется в регистр X1, а в регистре X остается результат операции.

Лучший способ освоить новые команды – проверить их действие на конкретных примерах. Попробуем решить несколько несложных задач. Алгоритм будет таким: вводим данные, команды и получаем на индикаторе числовой результат, который необходимо интерпретировать в соответствии с условием задачи. Как и раньше, будем заключать в кавычки результат, прочитанный с индикатора.

1. Перевести 14,50847 ч в часы, минуты и десятичные доли минуты:

14,50847 $\overleftarrow{\text{КО}}'$ "14,305082" ,

т.е. 14 ч 30,5082 мин.

2. Перевести 14,50847 ч в часы, минуты и секунды и их десятичные доли:

14,50847 $\overleftarrow{\text{КО}}'''$ "14,303049" ,

т.е. 14 ч 30 мин 30,49 сек.

3. Перевести 1,3888888 10^{-3} ч в часы, минуты и секунды:

1,3888888 ВП 3 /-/ $\overrightarrow{\text{КО}}'''$ "0,5 -03" .

Попробуйте догадаться, что получилось.

Задачу можно упростить. Достаточно прибавить к результату 1: на индикаторе "1,0005", т.е. результат был равен 5 сек. Кстати, эта операция, похоже, единственная, которая позволяет получить на индикаторе результат, начинающийся цифрой 0.

Обратный перевод проще. Не приходится долго разбираться с результатами, достаточно помнить лишь о том, что нуль заменяет отсутствующие десятки или единицы: например, 23 ч 1 мин и 7 сек записываются как 23,0107, а 23 ч 10 мин 20 сек – как 23,102.

Выполняя операции выделения целой части – К[X] и дробной – К{X}, микрокалькулятор МК-61, к сожалению, дает верный ответ лишь для положительных чисел. Это легко доказать.

По определению целая часть числа меньше такого же числа с дробной частью. Согласно этому правилу целая часть от –4,25 должна быть равна "–5", поскольку "–4" больше, чем "–4,25". Проверим, знакомо ли это правило нашему микрокалькулятору:

4,25 /–/ К [X] "–4" .

Ошибка.

Нетрудно предположить, что и с дробной частью не все ладно. Как гласит правило, дробная часть числа – разность между числом и его целой частью. Проверяем:

4,25 /–/ К{X}"–2,5 –01" ,

а должно было получиться

"7.5 –01" .

Итак, если есть вероятность, что операции придется выполнять с отрицательными числами, лучше воспользоваться отработанным на Б3-34 надежным приемом выделения целой части числа:

```
5 ↑ 9 / — 7 FEXT + FBX — .
```

Часто программисту не хватает всего одного шага. Эту операцию на МК-61 можно реализовать программой, которая на шаг короче:

```
K{X} ↑ FX<0 08 FBX 1 - XY FBX K{X} .
```

Операции выделения целой и дробной части числа позволяют расширить ресурсы памяти микрокалькулятора. В одном регистре при этом размещаются две константы, такие, что одна из них – целое число, а вторая – меньше единицы. Сумма констант – это число с целой и дробной частями, причем дополнительные ограничения на значения констант накладывает разрядность микрокалькулятора – восемь разрядов*. Сумма констант хранится в одном регистре памяти. Необходимую константу извлекают из памяти двумя командами: ИПМ F{X} или ИПМ F[X] .

Определение знака числа – еще одна новая операция. Как и предыдущие, она одноместная. Ей соответствует команда КЗН. Эта операция, одна из немногих, полностью эквивалентна математической операции

$$\operatorname{sign} p = \begin{cases} 1, & \text{если} \quad p > 0, \\ 0, & \text{если} \quad p = 0, \\ -1, & \text{если} \quad p < 0. \end{cases}$$

После команды K MAX большее из двух чисел, находившихся в регистрах X и Y, окажется в регистре X. При этой операции происходит следующее перемещение информации в стеке:

X>Y	X<Y	X=0, Y≠0	X≠0, Y=0
Y→Y X→X ↘X1	Y→Y X↘X ↘X1	Y→Y X→X ↘X1	X→Y X↘X ↘X1

Плохо то, что самым большим числом микрокалькулятор считает нуль. Кроме того, неясно в каком из регистров (X или X1) находится меньшее число, выделить которое с помощью команды K MAX не удается.

---

* Соответственно и сумма констант должна выражаться числом, в котором не более восьми разрядов.

Определение модуля числа – это одноместная операция, выполняемая по команде К|Х|. Отрабатывая ее, микрокалькулятор не изменяет положительные числа и убирает знак "минус" у отрицательных.

По команде КСЧ происходит обращение к встроенному генератору псевдослучайных чисел. Эта команда перемещает информацию в стеке так, как будто бы исполняется одноместная операция, т. е. содержимое регистров Т и Y не изменяется, в регистр Х1 копируется число из регистра Х, а в нем появляется псевдослучайное число.

Это не самое неудачное свойство генератора. Хуже, что микропрограмма, по которой выполняются вычисления, имеет ярко выраженную склонность к зацикливанию – числа начинают повторяться. Подобный генератор не годится даже для игровых программ.

Исследование свойств таинственного генератора доступно каждому владельцу МК-61. Энтузиасты, которые хотят во что бы то ни стало использовать этот генератор, выявили немало особенностей в его работе. Но пока авторам неизвестны ни точный алгоритм, по которому работает генератор, ни достаточно надежный способ, позволяющий избавиться от зацикливания.

Единственный совет, который может пригодиться в таких исследованиях, относится к "шаманским". Возвратиться к началу псевдослучайной последовательности можно, не выключая питание. Достаточно подать команды СХ ↑ К МАХ , и генерация начнется со "стартового" числа 0,404067.

Итак, если в программе необходимы псевдослучайные числа, пригодятся способы, разработанные еще для микрокалькулятора Б3-34:

$$\xi_{i+1} = \{37\xi_i + 1\}, \tag{1}$$

$$\xi_{i+1} = \{11\xi_i + \pi\}, \tag{2}$$

$$\xi_{i+1} = \{e^{\xi_i + \pi}\}, \tag{3}$$

$$\xi_{i+1} = \{10^{\xi_i + \pi}\}, \tag{4}$$

$$\xi_{i+1} = \frac{1}{\pi} \arccos\,[\cos(\xi_i \cdot 10^9)]. \tag{5}$$

Зависимости (1) – (5) позволяют получить псевдослучайные числа в интервале от 0 до 1, исключая границы интервала. Период последовательности (т.е. количество различных псевдослучайных чисел до первого повторения) для всех этих последовательностей достаточно велик, например: 8000 чисел для алгоритма (2), если начальное значение ξ_0=0,5; 4500 чисел для алгоритма (5) при ξ_0=1.

Начальное значение, которое вводит пользователь, определяет параметры последовательности (прежде всего, ее период). Так, в алгоритме (4) величина ξ_0 должна иметь дробную часть, алгоритм (3) требует $\xi_0 \neq 0$. Весьма чувствителен к выбору ξ_0 алгоритм (1): при $\xi_0 = 0{,}1$ период генерируемых чисел равен четырем (0,1 0,7 0,9 0,3 и вновь 0,1 0,7 ...); если выбрать $\xi_0 = 0{,}125$, период окажется равен двум (0,125 0,625 0,125 0,625 ...). И все же алгоритм (1) нельзя считать неудачным. Правильный выбор ξ_0 позволяет получать достаточно длинный период неповторяющихся чисел.

Алгоритмы (1) – (5) обладают одним общим свойством: генерируемая последовательность чисел полностью определяется значением ξ_0. Это очень удобно для отладки программ. При практических вычислениях, изменяя значение ξ_0, легко переходить от одной последовательности к другой.

У микрокалькулятора МК-61 программная память 105 шагов. Перейти на шаги 00-99 нетрудно. При прямой адресации в ячейку программной памяти вслед за командой перехода записывают его адрес. При косвенной – адрес перехода помещают в тот регистр, к которому затем будут обращаться командами косвенной адресации. Чтобы передать управление на адреса 100-104 командой БП, нажимают на клавишу десятичной точки " " и ту из клавиш 0-4, которая соответствует последней цифре адреса перехода.

Своим происхождением такой способ обязан особенностям устройства микрокалькулятора: шестнадцатиричную цифру 10 представляет буква A (в обозначениях микрокалькулятора – средний сегмент одного из разрядов индикатора). Зная это, легко понять, почему клавиша " " в сочетании с соответствующей из клавиш 0-4 дает нужный код.

Возможны и косвенные переходы. Их адреса формируют с помощью логических команд:

```
12n ↑ 88 KV "8, -n" K{X} "-n -01" ВП 2 ПN "-n"  .
```

Здесь n – последняя цифра адреса перехода, N – номер регистра, к которому происходит обращение при косвенной адресации такими, например, командами, как КБПN или KX=0N.

Встроенные логические операции – это еще одно отличие МК-61. Читатель, по-видимому, знаком с основами математической логики. Прежде чем рассказывать о логических операциях МК-61, вспомним о ключевых понятиях математической логики.

В прошлом веке английский математик Дж. Буль предложил алгебру, в которой функции и аргументы могли принимать все-

го два различных значения. Булева алгебра нашла широкое использование в математической логике, теории исследования операций, математическом моделировании и некоторых других разделах математики.

Придуманная задолго до первой ЭВМ булева алгебра оказалась как бы специально созданной для компьютеров. Ведь и вся информация, с которой работают ЭВМ, – это различные комбинации 0 и 1 (так чаще всего обозначают два основных состояния, в которых могут находиться элементы ЭВМ).

Как и во всякой дисциплине, в булевой алгебре есть определенные правила. Истинные суждения обозначаются единицей, ложные – нулем. Правила легче всего пояснить с помощью табл. 1.5.

Таблица 1.5

Результаты логических операций над двоичными числами

A	B	$\overline{A}$	$\overline{B}$	$A \wedge B$	$A \vee B$	$A \oplus B$	$A \to B$	$B \to A$	$A \leftrightarrow B$
0	0	1	1	0	0	0	1	1	1
1	0	0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	1	0	1
1	1	0	0	1	1	0	1	1	1

Перечислим основные логические операции:

$\overline{A}(\overline{B})$ – инверсия, или логическое отрицание (эквивалентна суждению "*не A*";

$A \wedge B$ – конъюнкция, или логическое умножение (ее эквивалент – "*A и B*");

$A \vee B$ – дизъюнкция, или логическое сложение (ей соответствует высказывание "*A или B*");

$A \oplus B$ – сложение по модулю 2 ("*или только A, или только B*"), другое название этой же операции – "исключающее ИЛИ";

$A \to B$ – импликация, или логическое следствие ("*если A ..., то B ...*");

$A \leftrightarrow B$ – эквивалентность, или логическое тождество ("*если A ..., то B ...*" и "*если B..., то A ...*").

Четырем основным логическим операциям в микрокалькуляторе МК-61 соответствуют команды К ИНВ (инверсия), К $\wedge$ (конъюнкция), К $\vee$ (дизъюнкция) и К $\oplus$ (сложение по модулю 2). Импликацию на микрокалькуляторе можно вычислить как $\overline{A} \vee B$, а эквивалентность – как $\overline{A \oplus B}$.

Результат любой логической операции на индикаторе начинается с цифры "8", стоящей в левом разряде. Если требуется

ввести операнд логической операции с клавиатуры, необходимо также начать ввод с цифры (отличной от нуля) и лишь после этого записывать операнд. Таким образом, в операнде логических операций не может быть более семи цифр.

Микрокалькулятор МК-61 способен выполнять логические операции с числами, записанными в шестнадцатиричной, восьмиричной, четверичной и двоичной системах счисления, исключая операцию "инверсия", которая дает верный результат только для шестнадцатиричных чисел. В других системах счисления непосредственно выполняются логическое сложение, умножение и сложение по модулю 2, а перед инверсией операнд необходимо сложить по модулю 2 с некоторым дополнительным числом. Для двоичной системы им является 8.EEEEEEE, которое, в свою очередь, получают инверсией числа 11111111. В четверичной – выручает дополнение 8.LLLLLLL (инверсия 33333333). В восьмиричной – используют константу 88888888.

Кто работал с микрокалькулятором "Электроника Б3-21" – "прадедушкой" нынешних, наверное, не забыл, что его стек состоял всего из двух регистров – X и Y, и при любых вычислениях содержимое изменялось. При выполнении команды X^Y в микрокалькуляторах Б3-34 и МК-61 эта особенность уже отмечалась. Кроме того, в МК-61 она присуща всем вычислительным операциям, вводимым клавишей K. И если участвуют два операнда, то оставшееся неизменным содержимое регистра Y становится трудно преодолимым барьером на пути к информации, находящейся в регистрах Z и T.

Все, что было сказано о МК-61, относится и к МК-52, который отличается прежде всего тем, что в нем появилось так называемое перепрограммируемое запоминающее устройство (ППЗУ). Еще одна существенная особенность – разъем, который позволяет подключать к микрокалькулятору блок расширения памяти (БРП). Пока выпускаются только два блока: БРП-2 и -3. Есть надежда, что их семейство будет расти.

Микрокалькулятор МК-52 несколько больше, чем МК-61. Для управления ППЗУ служат два переключателя С-З-СЧ (стирание – запись – считывание) и Д-П (данные – программа). Появились две новые клавиши: А↑ (ввод адреса обращения к ППЗУ) и ↑↓ (начало обмена информацией с ППЗУ: запись, считывание или стирание информации из ППЗУ. На задней стенке корпуса четыре крышки (две правые защищают разъемы, две другие – бутафорские).

Встроенное ППЗУ позволяет сохранять программы и данные при выключенном питании и для микрокалькулятора МК-52 играет роль накопителя информации на магнитной ленте. Это запоминающее устройство организовано в виде матрицы, состоя-

щей из 64 строк. Каждая строка, в свою очередь, разбита на 16 ячеек по 4 элемента. Ячейки пронумерованы числами от 0000 до 1023. Команды программы записываются шестнадцатиричными цифрами. В памяти микрокалькулятора каждая такая цифра занимает 4 разряда (бита), а для записи кода команды, состоящего из двух цифр, необходимы две ячейки ППЗУ, или 8 бит.

В постоянное запоминающее устройство (ПЗУ) информацию записывают лишь один раз. Именно в таких ПЗУ хранятся сведения, известные микрокалькулятору "от рождения", например значение числа π, последовательность сегментов, на которые необходимо подавать напряжение, чтобы вывести на индикатор ту или иную цифру. Следует отметить, что специальной микросхемы-ПЗУ в микрокалькуляторе нет, его роль исполняют небольшие фрагменты двух основных микросхем.

Встроенное ППЗУ позволяет не только записывать информацию для длительного хранения, но и стирать ее, а на освободившееся место записывать новую. Количество циклов "стирание – запись" ограничено, но за сохранность ППЗУ можно не беспокоиться – оно рассчитано на 10 000 циклов перезаписи, т.е. информацию в нем можно ежедневно обновлять в течение 30 лет.

Информация в ППЗУ хранится в виде электрических зарядов. Со временем они рассасываются, и если их вовремя не восстановить, запись теряется. В выключенном микрокалькуляторе программа способна храниться не менее 5000 ч, т.е. около семи месяцев. Если непрерывно обращаться к ППЗУ, то заряды могут исчезнуть уже через 250 ч. Чтобы этого не случилось, содержимое ППЗУ восстанавливают или, как говорят специалисты, регенерируют не реже, чем один раз в полгода.

Сравнивая ППЗУ со встроенным магнитофоном, нетрудно заметить ряд преимуществ и недостатков. Говоря о недостатках, следует отметить, что на магнитной карте программа хранится годами. Кроме того, библиотека программ на магнитных картах может быть весьма обширной, в то время как объем ППЗУ строго ограничен. Есть и преимущества: вместо сложного и дорогого механизма магнитофона работает надежная и дешевая электронная схема, которая практически не знает износа.

Как и в случае с магнитными картами, информация, хранящаяся в ППЗУ, переписывается в оперативное запоминающее устройство. К сожалению, аналогия с картами есть и там, где этого не требовалось. Так, к ППЗУ невозможно обратиться из программы. В подобных случаях принято говорить о программной недоступности устройства.

Представим, что было бы, если бы к ППЗУ можно было обращаться из программы. Первый фрагмент программы, отрабо-

тав, вызывал бы из ППЗУ второй, тот, в свою очередь, третий и т. д. В этом случае маленький объем оперативной памяти уже не столь значительно ограничивал бы сложность программ. Однако из программы ни считать, ни записать информацию в ППЗУ нельзя. Это делается только вручную, причем программу можно направить в ППЗУ только начиная с шага 00, а данные – лишь с нулевого регистра (поэтому, чтобы сэкономить ячейки ППЗУ, регистры с большими номерами лучше использовать как служебные, а в регистрах с меньшими номерами размещать данные, предназначенные для хранения в ППЗУ).

Правила обращения к ППЗУ чрезвычайно сложны и неэффективны. Кажется, что в ППЗУ, объем которого равен 512 шагам программы, легко разместить пять программ по 98 шагов каждая. Однако правила записи таковы, что если четыре программы имеют длину 98 шагов, пятая не может быть длиннее 91 шага.

Прежде чем записывать программы, определяют адрес обращения к ППЗУ. Он состоит из семи цифр. Первая может быть любой, кроме нуля*. Затем идут четыре цифры адреса, с которого начинается обмен с ППЗУ. Две следующие цифры – шестая и седьмая, если считать слева, – это длина блока информации, который записывается в ППЗУ или считывается из него.

Длина блока, кратная семи, задается в шагах программы и не может превышать 98 шагов. Дело в том, что каждые семь команд в ППЗУ автоматически располагаются в несколько странном порядке: вторая, третья, четвертая, пятая, шестая, седьмая, первая. При считывании порядок команд становится прежним, т. е. за первой командой идет вторая и т. д. Такая перестановка – дань особенностям устройства микрокалькулятора, и с этим приходится считаться.

На адрес не влияют последняя цифра мантиссы, показатель степени, знак числа и положение десятичной точки.

Перед тем, как записывать информацию в ППЗУ, необходимо его подготовить – стереть прежнее содержимое, т. е. освободить память от электрических зарядов, хранящих информацию. Если этого не сделать, то к уже имеющимся зарядам добавятся новые (произойдет логическое сложение поступающей информации с той, что хранилась). Например, если по адресу, где находилась команда с кодом 01, записать команду с кодом 5E, то после считывания в оперативную память на этом месте окажется команда 5F (буквой F обозначен символ "пробел"). Если попытаться ее исполнить, то индикатор очистится, а микрокалькулятор окажется полностью парализованным, и чтобы

* Чтобы не ошибиться, ставьте всегда одну и ту же цифру, например единицу.

вывести его из такого состояния, придется на несколько секунд отключить питание.

Перед считыванием из ППЗУ программную память очищать необязательно, старая программа замещается новой без каких-либо эксцессов.

Информацию в ППЗУ можно стирать только построчно, по 8 шагов программы (16 адресов) одновременно. Если в какой-то строке ППЗУ окажется хотя бы одна ячейка из указанных в адресе обращения, то информация в этой строке будет стерта во всех ячейках от первой до последней.

Например, адрес обращения к ППЗУ 1002007 означает, что мы хотим стереть содержимое в 14 ячейках памяти ($7 \times 2 = 14$) начиная с ячейки под номером 20 по ячейку под номером 33 ($20 + 7 \times 2 - 1 = 33$). Вместо этого произойдет стирание информации начиная с ячейки 16 (поскольку она стоит первой в строке, где находится ячейка 20) по ячейку 47 (это последняя ячейка в строке с ячейкой 33). Так вместо 14 адресов мы очистим 31.

Чтобы не испортить предыдущую программу (или первые шаги следующей), начальный адрес каждой новой программы должен быть кратен 16. В итоге начальный адрес каждой программы приходится определять весьма замысловатым способом: к начальному адресу предыдущей программы прибавлять удвоенную длину новой. Если получившееся число не кратно семи, то его необходимо увеличить до ближайшего, кратного семи. Полученную сумму дополняют до ближайшего числа, кратного 16. Как видите, вся процедура напоминает математическую головоломку.

В отличие от стирания запись и считывание выполняются в точном соответствии с заданным адресом, но при этом, как уже говорилось, происходит перестановка команд. Всякий раз, когда длина записываемого или считываемого фрагмента не кратна семи, первая команда из последней неполной семерки не будет записана или не будет считана.

Лучшая иллюстрация – практический пример! Попробуем записать в ППЗУ три программы (из 34, 42 и 15 команд), размещая их в ППЗУ одну за другой. Начнем с адреса 0000.

Набираем первую программу. Убедившись, что она введена правильно, рассчитываем адрес обращения к ППЗУ. Длина программы 34 шага (не кратна семи), а ближайшее большее число, кратное семи, – это 35. Итак, в качестве длины первой программы укажем 35. С начальным адресом первой программы все просто, он равен нулю. Набираем адрес 1000035. Нажав клавишу A↑, вводим набранный адрес в специальный регистр памяти, который назовем AD. В этом регистре адрес будет

храниться до тех пор, пока мы не введем какой-либо новый адрес командой А↑ или не выключим питание.

Теперь необходимо очистить в ППЗУ место для нашей программы. Переключатель С-З-СЧ ставим в положение С (стирание). Переключатель Д-П обязательно должен быть в положении Д (данные). Дело в том, что при стирании в ППЗУ одновременно стирается такое же количество информации и в памяти данных (в регистрах) или в программной памяти – то или иное определяется переключателем Д-П. Чтобы подготовить ППЗУ, придется пожертвовать содержимым первых пяти регистров (0-4).

Читатель несомненно помнит, что один регистр памяти занимает в ППЗУ столько же ячеек, сколько отводится для хранения семи шагов программы. Если в стираемых регистрах записаны какие-то нужные данные, их обычно переписывают в другие регистры или выписывают вручную*.

Приготовления закончены, нажимаем клавишу ↑↓. Появление знака минус во всех разрядах индикатора говорит о том, что идет обмен с ППЗУ. Если нажимать другие клавиши или изменять положение переключателей, пока этот знак не погас, то информация будет потеряна. Минусы погасли, стирание завершено. Теперь можно записывать программу. Переключатель С-З-СЧ ставим в положение З, а переключатель Д-П – в положение П. Адрес обращения к ППЗУ уже хранится в регистре AD, поэтому остается только нажать клавишу ↑↓. На индикаторе на несколько секунд вновь появляются минусы. Итак, 35 шагов программы (начиная с адреса 00), точнее, соотвествующие им коды, переписаны в ППЗУ и стерты из оперативной памяти микрокалькулятора.

Программу можно восстановить. Для этого достаточно поставить переключатель С-З-СЧ в положение СЧ и нажать клавишу ↑↓.

Когда обращение к ППЗУ завершится, можно будет проверить правильность записи программы в ППЗУ, запустив ее на исполнение.

На очереди – вторая программа. Ее длина 42 шага, т. е. она кратна семи. Определим начальный адрес. Чтобы вычислить его, в регистр X вводят начальный адрес предыдущей программы (кратный 16, т.е. 0), ее длину в шагах (кратную семи, т. е. 35) и выполняют операции**:

---

* Существует другой вариант – заранее очистить нужную область ППЗУ. Тогда программу записывают сразу, а все данные останутся в целости и сохранности.

** Дополнив последовательность командой С/П, удобно записать эту мини-программу по адресу 1099214, т. е. в две последние строки ППЗУ, чтобы вызывать ее всякий раз, когда требуется изменить содержимое ППЗУ.

↑ 8 / K[X] FBX K{X} K3H + 1 6 * +

На индикаторе число 80. Это и есть начальный адрес, которым мы воспользуемся при обращении к ППЗУ. Число 1008042 записываем в регистр AD, нажимая клавишу A↑. Затем следуют: стирание (С А ↑↓), запись (З П ↑↓) и считывание (СЧ П ↑↓).

На этот раз при стирании были очищены регистры 0-5.

Третья программа имеет 15 команд (последняя команда по адресу 14), поэтому за ее длину принимаем число 21. Определяем начальный адрес: вводим команды 80 ↑ 42, – и повторяем уже знакомую цепочку расчетов. На индикаторе "176". Осталось записать в регистр AD число 1017621, очистить ППЗУ, записать программу и выполнить контрольное считывание.

Данные в ППЗУ записывают примерно так же, как и программу. Предположим, что во второй программе регистры используют следующим образом: регистр 0 – счетчик циклов; регистры 1 – 5 – данные; регистр 6 – счетчик данных; регистры 7 – 9 отведены для хранения промежуточных результатов.

Запишем данные в ППЗУ. Регистры 0 и 6 – 9 – служебные, информация в них появляется лишь по мере работы программы, поэтому их содержимое записывать в ППЗУ не требуется. Но поскольку пересылка данных в ППЗУ всегда начинается с регистра 0, то, чтобы сохранить данные из регистров 1 – 5, придется записать и содержимое регистра 0. Один регистр требует семь ячеек ППЗУ, поэтому в качестве длины записи выбираем число 42 ($6 \times 7 = 42$).

Определяем начальный адрес: вводим команды 176 ↑ 21, – и повторяем цепочку вычислений, уже знакомую нам по двум предыдущим расчетам.

На индикаторе "224", следовательно, адрес обращения будет 1022442. Устанавливаем переключатели в положения С и П (на этот раз нам нужны данные, и поэтому приходится жертвовать программой, если она записана только в программной памяти). Набрав адрес обращения, нажимаем клавиши A↑ и ↑↓ (при этом стираются первые 42 шага программы). Осталось записать данные (З А ↑↓) и провести контрольное считывание (СЧ А ↑↓).

Итак, в памяти нашего микрокалькулятора хранятся три программы и один набор данных.

Полезно иметь "карту" памяти, на которой все адресное пространство ППЗУ разбито на 64 строки – по 16 адресов в каждой. В нашем случае карта памяти будет выглядеть так:

№ строки	Адрес		Комментарий	№ строки	Адрес		Комментарий
1	0000	\	1000035	13	0192	|	Программа 3
2	0016	|	Программа 1	14	0208	/	Вычисление ...
3	0032	|	Обработка	15	0224	\	1022442
4	0048	|	результатов	16	0240	|	Данные
5	0064	/		17	0256	|	для
6	0080	\	1008042	18	0272	|	программы
7	0096	|	Программа 2	19	0288	|	(Набор 1)
8	0112	|	Расчет ...	20	0304	/	
9	0128	|		21	0320		
10	0144	|					
11	0160	/					
12	0176	\	1017621	64	1008		

Без расчетов видно, что новую информацию в ППЗУ можно записывать с адреса 320.

Кстати, при занесении информации в последние строки ППЗУ будьте внимательны: для ППЗУ адрес 1024 эквивалентен адресу 0000, адрес 1025 – адресу 0001 и т.д., поэтому необходимо следить, чтобы "хвост" последней программы не испортил начало первой.

И еще один нюанс: при обмене информацией с ППЗУ резко увеличивается потребляемая мощность и, если батарейки "подсели", возможен сбой. Поэтому к ППЗУ лучше обращаться, когда микрокалькулятор работает от блока питания.

После того, как информация записана в ППЗУ и составлена карта памяти, легко вызвать любую программу или набор данных: достаточно установить переключатель С-З-СЧ в положение СЧ, а переключатель Д-П в положение "данные" или "программа", выбрать указанный в карте памяти адрес обращения и нажать клавиши А↑ и ↑↓. Так, конечно, быстрее, чем набирать всю программу заново, и к тому же в программе не будет ошибок, связанных с повторным набором (нажата не та клавиша, пропущена команда и т. д.).

Существует немало программ, вычисляющих адрес обращения – от простейших, какой мы пользовались, до таких, которые хранят в памяти всю карту адресов и способны к тому же автоматически отыскивать свободное место и формировать полный адрес обращения. Но оптимум, по-видимому, находится где-то посередине.

Хорошим решением, например, можно считать программу 1.4, составленную ленинградцем В. Смирновым на основе опубликованной в журнале "Наука и жизнь" (1987, № 4).

Программа 1.4

Определение адреса для записи программы в ППЗУ

	00	01	02	03	04	05	06	07	08	09
00	↑	7	/	8	FLG	+	K[X]	7	*	1
10	6	ИП0	ИП1	+	FBX	+	1	6	/	9
20	FLG	+	K[X]	*	П2	2	FEXT	*	+	6
30	FEXT	+	XY	2	*	ИП2	+	1	-	С/П

Работая с программой 1.4, начальный адрес предыдущей заносят в регистр 0, ее длину – в регистр 1 (если речь идет о первой программе, то в этих регистрах должен быть нуль), число шагов новой программы – в регистр X и нажимают клавиши В/О С/П. После останова на индикаторе высвечивается готовый адрес обращения. Нажав клавишу XY, на индикатор выводят номер конечной ячейки и записывают его в карту адресов ППЗУ. Строка, в которую попадает этот номер, будет последней строкой ППЗУ, занятой новой программой. Чтобы возвратить адрес обращения к ППЗУ в регистр, нажимаем клавишу XY. Остается заменить вспомогательную программу той, которую необходимо записать, и отдать команды А↑ и ↑↓.

Контрольный пример:

192→P0, 35→P1, 54→PX, В/О С/П "1027256" XY "383" XY "1027256"
А↑ ↑↓

После останова можно нажимать клавишу А↑ , тогда клавишей XY придется пользоваться один раз.

Освоив эту программу и карту памяти, можно успешно работать с ППЗУ. Чтобы сохранить программу, записанную в энергонезависимой памяти, от случайного стирания (если будет нажата клавиша ↑↓), переключатель С-З-СЧ, когда им не пользуются, ставят в положение СЧ.

Если несколько программ имеют общий фрагмент, то в ППЗУ их записывают, разбив на блоки: общий для всех программ и специальные (для каждой программы свой).

Допустим, что несколько программ имеют общее начало. В этом случае, если длина общего блока окажется не кратной семи, необходимо либо включить в него начало одного из специальных блоков, либо записать последние команды общего блока в начале каждого из специальных, либо дополнить общий блок командами КНОП так, чтобы его длина была кратной семи. В первом случае начальные команды специальных блоков придется вписывать вручную после того, как программа будет скомпонована. В остальных случаях команды вписывать не требуется, но несколько возрастет объем, занимаемый в

ППЗУ пакетом программ. Какой вариант выбрать – зависит от длины общего блока, числа программ и их длины.

При чтении прежде всего считывается специальный блок (в адресе обращения длина программы увеличена на длину общего блока, а начальный адрес уменьшен на удвоенную длину общего блока). В результате в программной памяти на месте, где должен быть общий блок, оказывается "хвост" предыдущей программы. Программу компонуют полностью, считывая общий блок. На этот раз в качестве длины программы при обращении к ППЗУ указывают число команд общего блока.

Аналогичным образом можно поступать и в тех случаях, когда несколько программ имеют общее завершение.

Конечно, досадно, что из 105 шагов программной памяти в ППЗУ можно записать только 98. Впрочем по адресу 1AAAA-2 (здесь AAAA – начальный адрес, а -2 – длина программы в шестнадцатиричной системе) можно записать программу длиной 103 шага, однако с дефектом – пропадает команда по адресу 98. Если считать такую программу из ППЗУ, на шаге 98 в программе окажется "дырка" – там останется команда, которая была раньше (если никакой программы до этого не было, например если питание только что включено, то по адресу 98 записан код 00). Этот эффект обусловлен известной особенностью ППЗУ, т. е. тем, что команды в каждой "семерке" переставляются. В результате недостающую команду приходится вписывать вручную или, если позволяет программа, на шаге 99 ставить команду F→ , которая "выбросит" нуль, появившийся на предыдущем шаге.

Если набрать адрес обращения 1AAAA-5, то все равно не удастся записать все 105 шагов программы. В этом случае после записи и чтения на шаг 00 переместится команда с шага -3, на шаг 08 – команда с шага -4, а на шаг 09 – с шага 98. Разумеется, все команды, бывшие на шагах 98, -3 и -4, остаются на месте.

Воспользовавшись адресом 1AAAA-2, нетрудно записать данные из всех 15 адресуемых регистров. Однако число из регистра E потеряет показатель степени и знак, а десятичная точка окажется после первой цифры мантиссы. Данные из остальных регистров будут записаны правильно.

Если в ППЗУ необходимо сохранять одни и те же программы или данные, процедуру регенерации можно упростить. После того, как работа с программой завершена, переключатель C-3-CЧ устанавливают в положение 3 и нажимают клавишу ↑↓. Дело в том, что адрес в ячейке AD, попавший туда после нажатия A↑, хранится до тех пор, пока не выключим микрокалькулятор или введем новое значение.

Операцию стирания можно исключить, потому что в данном случае логическое сложение программы, хранившейся в ППЗУ, и ее копии из памяти не изменяет содержимое ППЗУ, так как A ∨ A = A.

В случае, когда необходимо регенерировать не только программу, но и данные, запись в ППЗУ начинают с той информации, что извлекалась из него последней, действуя "коротким" способом: з ↑↓, и лишь затем набирают адрес той части, которая извлекалась из ППЗУ первой. Особое внимание при этом следует уделять верному положению переключателя Д-П. И, как всегда, закончив операции, установите переключатель С-З-СЧ в позицию СЧ.

Работая с микрокалькуляторами Б3-34, МК-54 и -61, каждый раз приходится вручную вводить и проверять программу, константы и данные. Способности ППЗУ микрокалькулятора МК-52 также весьма ограничены – слишком мало программ оно может хранить. В то же время владельцы этого микрокалькулятора обладают потенциальной возможностью расширить свою библиотеку готовых к применению программ. Ее реализует блок расширения памяти, рассчитанный на подключение к разъему на задней стенке корпуса микрокалькулятора.

Блок расширения памяти "Электроника-астро" (БРП-2) представляет собой ПЗУ объемом 1024 четырехразрядных слов, собранное на основе микросхем серии КР556. Программируют ПЗУ на предприятии, пережигая плавкие перемычки. Восстановить их невозможно, поэтому стереть однажды записанную информацию уже нельзя. В результате любая ошибка при программировании оборачивается необходимостью выбросить исправное, но неправильно запрограммированное ПЗУ. Это конечно сказывается на себестоимости готовых блоков. Может быть поэтому предприятие-изготовитель не торопится наладить производство серии аналогичных блоков с различной "начинкой". Блок БРП-2 долгое время оставался единственным их представителем. Описание этого блока и его содержимого приведено в журнале "Морской сборник" (1988, № 12).

В 1988 г. началось производство БРП-3 с набором программ для решения шестидесяти задач вычислительной математики. За год выпустили более 15 тыс. таких блоков, а в 1989 г. к ним прибавилось еще 80 тыс.

Разрабатывают БРП-4 с программами решения бытовых и спортивно-профилактических задач, а также с игровыми программами.

Для сопряжения микрокалькулятора с блоком расширения памяти (и ППЗУ) потребовались дополнительные технические ухищрения. По сравнению с микрокалькулятором МК-61 в МК-52 появились еще три новых микросхемы: ППЗУ, микро-

схема, формирующая сигналы стирания – записи – считывания для ППЗУ, и микросхема интерфейса для связи с блоком расширения памяти.

Для чего нужна микросхема интерфейса? Именно она в режиме записи выдает для ППЗУ данные, а в режиме чтения принимает их. Эта же микросхема указывает, по какому адресу ППЗУ записать или считать данные. Она же обслуживает и разъем блока расширения памяти.

Дело в том, что внутри любого отечественного программируемого микрокалькулятора от Б3-21 до МК-52 информация между всеми микросхемами вычислительного блока передается по последовательной магистрали – всего по одному проводу бит за битом. Кроме информации, по четырем параллельным проводникам идут синхросигналы.

Большинство цифровых устройств обмениваются информацией одновременно по нескольким проводникам, например в ППЗУ данные должны поступать по четырем линиям, а адрес – по десяти.

Микросхема интерфейса обеспечивает преобразование информации, поступающей по последовательной магистрали от вычислительного блока микрокалькулятора. Цепочка сигналов преобразуется в четырехразрядные слова данных и двенадцатиразрядный адрес, необходимый для обращения к ППЗУ или БРП.

Интерфейс выполняет и обратное преобразование. Алгоритм работы прост: при нажатии клавиши А↑ символы, стоящие в числе из регистра X на позициях со второй по пятую, поступают в интерфейс в качестве начального адреса. Шестой и седьмой символы запоминаются как длина записи. После нажатия клавиши ↑↓ интерфейс выдает на адресные входы ППЗУ начальный адрес, а на информационные входы – соответствующие четыре бита из памяти данных или программ (это определяется положением переключателя Д-П). Затем адрес увеличивается на единицу, а на шине данных появляются следующие четыре бита. Эти действия повторяются до тех пор, пока адрес не станет равен сумме начального адреса и удвоенной длины записи, причем каждые семь байт (1 байт = 8 бит) переставляются соответствующим образом.

Линии (шины), по которым передаются адреса и данные, выведены на разъем БРП. К этому разъему можно подключить блоки объемом до 4096 слов (2048 шагов программы). Как уже говорилось, объем встроенного ППЗУ в четыре раза меньше, но поскольку ППЗУ и БРП обслуживает одна и та же микросхема, то при считывании информации, например из любого БРП, длина записи также должна быть кратна семи.

Кроме линий данных и адреса, на разъем блока расширения памяти выведены три из четырех тактовых сигналов последовательной магистрали, а также напряжения питания 0 и −15 В. Наконец, еще один контакт этого разъема служит для отключения ППЗУ от микросхемы интерфейса при подключении блока расширения памяти.

К сожалению, есть существенный недостаток в организации интерфейса и, особенно, подключении разъема. Если попытаться соединить с ним какое-либо внешнее устройство и записать в него информацию, ничего не выйдет – шина данных подключена таким образом, что возможно только чтение информации с внешних устройств. Чтобы вывести информацию из микрокалькулятора, пришлось бы пожертвовать гарантией или дожидаться ее окончания, поскольку необходимо вскрыть корпус микрокалькулятора. В то же время шину данных можно было бы вывести на разъем таким образом, чтобы получить возможность не только считывать программы и данные в МК-52, но и выводить их на внешние устройства без каких-либо переделок микрокалькулятора.

Есть и еще один недостаток у интерфейса – он программно недоступен. Это означает, что в системе команд отсутствуют команды обмена информацией с внешними устройствами (как и с ППЗУ). Дело в том, что и ППЗУ, и разъем блока расширения памяти обслуживает одна и та же микросхема. К сожалению, программную недоступность интерфейса и ППЗУ нельзя устранить небольшой "доработкой".

А жаль, ведь всего одна команда чтения информации с интерфейса позволила бы резко расширить сферу применения микрокалькулятора МК-52. Например, стало бы возможным использовать его для сбора и оперативной обработки информации с небольшой частотой (примерно один-два отсчета в минуту и реже), а подобных задач достаточно много. Команда вывода информации на внешний разъем обеспечила бы выдачу собранных данных на принтер или на центральную ЭВМ. В этом случае программируемый микрокалькулятор смог бы самостоятельно общаться с внешним миром и кое-где, пожалуй, потеснил бы более дорогие и сложные машины. Да и логические операции при обработке поступающей информации нашли бы более достойное применение, чем создание видеосообщений.

На меньший разъем (второй, считая справа) выведены все сигналы последовательной магистрали напряжения питания и напряжение, необходимое для программирования ППЗУ, а также сигналы опроса (считывания) и стирания ППЗУ.

Для тех, кто собирается подключать к разъему какое-либо самодельное внешнее устройство, заметим, что логической единице соответствует напряжение −15 В, а логическому нулю −0 В.

Итак, мы завершаем первое знакомство с микрокалькулятором "Электроника МК-52" – наиболее совершенной моделью из семейства "Электроника Б3-34".

Выбранная в конце 1970-х годов система команд позволяет адресоваться не более, чем к 16 регистрам и осуществлять переходы лишь на 160 адресов (от 00 до F9). То, что при косвенном обращении к регистрам 0 – 3, 4 – 6 и 7 – E их содержимое модифицируется по-разному, также создает трудности при программировании. Работу могли бы облегчить различные команды косвенной адресации, например команды косвенной адресации с инкрементированием (содержимое любого регистра независимо от его номера при обращении увеличивалось бы на единицу), косвенной адресации с декрементированием (содержимое нужного регистра уменьшалось бы на единицу) и косвенной адресации без изменения содержимого регистра (оно лишь модифицировалось бы). Именно так, кстати, осуществляется косвенная адресация в микропроцессорах, которые являются основой современных персональных ЭВМ.

Архитектура программируемого микрокалькулятора во многом определяет его быстродействие и удобство модернизации. Особенность архитектуры семейства Б3-34 не только в том, что все команды и данные передаются бит за битом по одному проводнику, но и то, что все блоки вычислительного устройства соединены в кольцо, где информация движется лишь в одном направлении. В результате постоянно возникает ситуация, когда информация, посланная одной из микросхем своей "соседке" по последовательной магистрали, прежде чем попадет к адресату, должна пройти через все остальные микросхемы. Ясно, что такой способ обмена информацией не позволяет рассчитывать на высокое быстродействие. Стоит добавить в кольцо какой-либо новый элемент, например дополнительную память, как быстродействие еще ухудшится.

Конечно, всем, кто хотя бы раз работал с программируемым микрокалькулятором, хочется, чтобы он имел энергонезависимую память (тогда не надо будет каждый раз вводить и проверять программу), перепрограммируемый блок программ пользователя, высокое быстродействие и программную совместимость с МК-52. Это позволило бы использовать программное обеспечение, созданное для микрокалькуляторов с кодово-символьным языком и обратной бесскобочной логикой. Вряд ли стоит надеяться на такое развитие событий.

Однако в любом случае программируемые микрокалькуляторы останутся для своих владельцев настоящими машинами, которые приобщили их к миру ЭВМ.

ДЕТАЛИ И ТОНКОСТИ

АЛФАВИТ КАРМАННОЙ ЭВМ

Современные ЭВМ способны работать с латинским, русским и многими другими алфавитами, а микрокалькулятор воспроизводит на индикаторе лишь несколько символов.

Если договориться о том, чтобы некоторые цифры интерпретировать как буквы, то получится небольшой алфавит. В табл. 1.6 предложен один из вариантов сопоставления символов микрокалькулятора и букв русского и латинского алфавитов.

Конечно алфавит получается бедным. Его можно несколько расширить, используя буквы обоих алфавитов или неочевидные интерпретации: например, считая цифру 7 буквой Т, число 177 – буквой *т*, а Г или 17 – буквой П. А если еще разрешить микрокалькулятору "говорить с акцентом" и читать слова "вверх ногами", словарный запас расширится.

Остался пустяк – научиться выводить буквы и слова на индикатор. В качестве сообщения нередко используют сигнал аварийного останова "ЕГГОГ", появляющийся на индикаторе всякий раз, когда микрокалькулятор выполняет некорректную операцию или происходит переполнение.

Хотя в различных ситуациях на индикаторе формируется почти одно и то же изображение, в каждом конкретном случае содержимое регистра X, скрытое сообщением аварийного останова, проявляет себя по разному. Например, при переполнении, получающемся в результате операций 1 ВП 50 FX^2 "ЕГГОГ" , результат можно записать в любой регистр памяти, а затем вызвать на индикацию. В этом случае, однако, на индикаторе вновь появится сообщение "ЕГГОГ". Если разделить число из регистра X на десять, то получится "1 90", т. е. 10^{90}, а "ЕГГОГ" исчезнет.

Для упрощения, говоря о сообщениях аварийного останова, будем считать, что содержимое регистра X и индикатора тождественно. Если команду FX^2 подать дважды, возникнет сообщение "Э.ГГОГ", которое также нетрудно записать в память.

Используя некоторые команды из числа запрещенных, нельзя определить, в результате чего появился "ЕГГОГ" – из-за некорректности или переполнения. К сочетаниям, вызывающим аварийный останов относится, например, комбинация ВП . (десятичная точка). Эту комбинацию часто применяют в играх, поскольку она дает мгновенный "ЕГГОГ" в отличие, например, от операций K√, K1/X, KX^2 , которые вызывают останов через 5 – 7 сек.

Некорректные операции, вводимые префиксной клавишей K, удобно использовать для получения буквы Е. В регистр X вво-

дят любую цифру, кроме нуля, и выполняют операцию К+ (или любую из только что перечисленных):

```
"ЕГГОГ" ВП "ЕГГОГ" ОО" ↑ "Е"  .
```

Это сообщение можно записать в любой регистр памяти. Из буквы "Е" удобно создавать другие символы, для чего ее заносят в один из четырех первых регистров (0 – 3), например в регистр 0. Далее действуют так: КИПО ИПО "0000000Г" ↑ "Г" .

Таблица 1.6

Интерпретация символов микрокалькулятора

Символ	Буква алфавита	
	русского	латинского
0	О	*O*
1		*i*
3	З	
4	Ч	
5		*s*
6	Б	
8	В	*в*
9		*q*
L		*l*
Г	С	*c*
Г	Г	*r*
Е	Е	*e*

Сообщение "Г" удается построить, не синтезируя заранее "Е", например: СХ "0" К1/Х "ЕГГОГ" ВП ВП ↑ "Г" .

Из "Е" получается "С": КИПО КИПО ИПО "0000000С" ↑ "С" .

Несложно сформировать "L" и тире:

```
КИПО КИПО КИПО ИПО "0000000L" ↑ "L"  ,
КИПО КИПО КИПО КИПО ИПО "0000000-" ↑ "-"  .
```

Если удлинять цепочку команд КИПО, получатся цифры 9, 8 и т. д.

Букву Е можно сформировать и по-другому:

```
1 ВП 50 FX^2 "ЕГГОГ" ВП "ЕГГОГ ОО" ↑ "Е"  .
```

Иначе некорректные операции работают с дробными числами.

Например, в дробном числе, содержащем более двух знаков, операции К... и ВП ↑ заменяют первую цифру буквой Е. Остальные цифры при этом не изменяются, в том числе знаки числа и порядка.

Вот еще один конкретный пример:

```
52,31 - ВП 09 К 1/Х "ЕГГОГ" ВП "-ЕГГОГ 00" ↑ "-Е2,31 - 09"  .
```

Конечно, буквы и числа, начинающиеся с буквы Е облегчают "общение" с микрокалькулятором, но хотелось бы получать целые слова. Эту задачу успешно решил москвич В. Архипов. Он предложил для получения слов-видеограмм, состоящих из символов, которые способен выдавать программируемый микрокалькулятор, короткую программу:

```
00.КИП0 01.ВП 02.7 03.П9 04.КИП9  05.КИП↑ 06.ИП9 07.ХУ 08.ХУ
09.ВП 10.ВП 11.1. 12./-/ 13.FLO 14.03 15.С/П. 16.БП 17.00  .
```

Эта программа предназначена для микрокалькулятора "Электроника Б3-34". Чтобы воспользоваться ею, владельцам микрокалькуляторов МК-61 и -52 необходимо изменить команду по адресу 05 на КПП Е, в регистр Е записать число 18 и дополнить программу фрагментом

```
18.ИП0 19.П0 20.КИП0 21.В/0  .
```

В регистр 0 вводят число $N + 1$, где N – число символов видеограммы, которые записывают в порядке их следования в регистры 1 – 8. Как и числа, видеограммы не могут начинаться символом "0", а те, что состоят из восьми символов, не должны им кончаться.

В качестве примера сформулируем сообщение "БЕГ-БЕГ". После того, как программа записана в память, в регистр 0 вносим число $8 = N + 1$, в регистры 1 и 5 – цифру 6 (буква Б, см. табл. 1.6), в регистры 2 и 6 – букву Е, в регистры 3 и 7 – букву Г, а в регистр 4 – тире. Тире, буквы Е и Г получают любым из уже известных нам способов. Запускают программу командой С/П (в первый раз В/0 С/П). Примерно через 30 сек на индикаторе появляется видеограмма "БЕГ-БЕГ".

Программа В. Архипова относится к "маленьким хитростям" программируемого микрокалькулятора. В режиме пошаговой проверки видеограммы не получаются. Их можно создавать только в режиме вычислений по программе.

Микрокалькулятор МК-61 позволяет создавать видеограммы иным, более простым путем. Для этого используют логические операции. Результат операции может содержать все символы,

которые позволяет формировать микрокалькулятор, в том числе -, L, Г, Г, E, и натуральное число 15, которое в шестнадцатиричной системе изображается буквой F, а в наших микрокалькуляторах – отсутствием всякого знака, что можно использовать как пробел между буквами или словами.

Говоря в дальнейшем об этом "пустом месте", будем именовать его F. Необходимо тщательно следить, чтобы буква F не оказалась в первой позиции видеограммы. Появившись здесь, "пробел" способен испортить программу: ввести те команды, которых там не было, и уничтожить записанные ранее. Старайтесь поэтому не допускать F на первое знакоместо. Впрочем, в

а

x	0	1	2	3	4	5	6	7	8	9
К ИНВ	F	E	Г	Г	L	—	9	8	7	6

б

К v	9	8	7	6	5	4	3	2	1	0
0	9	8	7	6	5	4	3	2	1	0
1	9	9	7	7	5	5	3	3	1	
2	L	—	7	6	7	6	3	2		
3	L	L	7	7	7	7	3			
4	Г	Г	7	6	5	4				
5	Г	Г	7	7	5					
6	F	E	7	6						
7	F	F	7							
8	9	8								
9	9									

иных ситуациях цифра F отличается от своих собратьев лишь тем, что она невидимка, пустое место.

Логические операции микрокалькулятор выполняет поразрядно. Поэтому прежде, чем создавать видеограмму, по рис. 1.9 следует определить, в результате каких действий получаются необходимые символы. Мы уже отмечали, что результат любой логической операции на индикаторе начинается с цифры 8 (буквы B). Видеограмма, первый знак которой – эта буква, может состоять из восьми знаков, во всех иных случаях видеограммы, формируемые с помощью логических операций, содержат не более семи знаков.

Рис. 1.9. Результаты логических операций

а – инверсия; *б* – логическое сложение; *в* – сложение по модулю два (исключающее ИЛИ)

в

К ⊕	9	8	7	6	5	4	3	2	1	0
0	9	8	7	6	5	4	3	2	1	0
1	8	9	6	7	4	5	2	3	0	
2	L	—	5	4	7	6	1	0		
3	—	L	4	5	6	7	0			
4	Г	C	3	2	1	0				
5	C	Г	2	3	0					
6	F	E	1	0						
7	E	F	0							
8	1	0								
9	0									

Пример 1. Как создать видеограмму "BEC"? Буква "E" получается при инверсии цифры 1 (см. рис. 1.9, *а*), а буква "C" – при инверсии цифры 3. Первая цифра числа заменяется цифрой 8. Таким образом операция "инверсия" превратит число N13, где N – любая цифра, кроме нуля, в видеограмму "BEC". Если десятичная точка после буквы B мешает эстетическому восприятию, то командами ВП 2 ↑ ее нетрудно перенести в конец слова.

Пример 2. Сформируем видеограмму "БOLБEC". Похоже, любители игр испытывают симпатию к этому слову при оценке неудачного хода. Эту видеограмму нельзя создать с помощью операции инверсии, так как мы не можем ввести с клавиатуры букву F, а 0 получается лишь при инверсии этой цифры. Такую видеограмму получают в результате логического сложения чисел N608688 и N003064. Буквой N вновь обозначена любая цифра, кроме нуля.

Пусть N = 1. Тогда

```
1608688 ↑ 1003064 K∨ "8,БOLБEC"  .
```

Цифру 8 убираем командой выделения дробной части:

```
K{X} "БOLБEC -01" ВП 6 ↑ "БOLБEC"  .
```

Пример 3. Столица Норвегии – город Осло. Создадим видеограмму "OSLO". Видеограмма начинается с нуля, но он оказывается в первой позиции десятичной дроби (результат любой логической операции – это десятичная дробь), а поэтому не будет индицироваться. Если отделить нуль от цифры 8 пробелом, а затем убрать ее, можно столкнуться с неприятными эффектами, связанными с тем, что в первой позиции окажется "невидимка" F. Проще поставить перед словом "OSLO" тире (шестнадцатиричное A в кодировке микрокалькулятора).

Десятичная дробь на индикаторе всегда заканчивается ненулевой цифрой, поскольку все нули, "незначащие" с точки зрения микрокалькулятора, автоматически "гасятся", т. е. не выводятся на индикатор. Конечно, изменив порядок числа-видеограммы, нуль легко перевести в разряд целых – в этом случае он займет на индикаторе свое законное место.

Для получения этой видеограммы воспользуемся командой сложения по модулю 2. С помощью рис. 1.9 *в*, определяем операнды нашей видеограммы:

```
888588 ↑ 828038 K⊕ "8.-OSL" K{X} "-.OSL -01" ВП 5 ↑ "-OSLO"  .
```

Чтобы надпись выглядела симметричной, поставим тире и в конце слова:

8885888 ↑ 8280382 K⊕ "8.-OSLO-" K{X} "-.OSLO- -01" .

После разбора этих трех примеров нетрудно построить любую видеограмму, используя рис. 1.9.

Так удается получать видеограммы не только "вручную", но и по программе. В этом случае регистров памяти видеограммы не занимают, для их получения не требуются особые программы. В отличие от программ для Б3-34, программы с видеограммами, полученными с помощью логических команд МК-61, дают одинаковый результат при пошаговой проверке и в автоматическом режиме.

На индикаторе микрокалькулятора МК-61 создают видеосообщения длиной до семи символов. Владельцы МК-52 при желании могут получить видеосообщение длиной до 12 символов, однако, с небольшими ограничениями: на месте знака мантиссы нельзя вывести 1, а на месте знака порядка будет либо "-", либо пробел.

Для этого микрокалькулятор переводят в режим программирования и набирают "программу", состоящую из кодов с символами, которые будут составлять видеосообщение. Ее, как обычно, записывают в ППЗУ, а считывают не в память программ, а в память данных (шаги 00-06 "программы" станут числом в регистре 0, шаги 07-13 – в регистре 1 и т. д.).

Записывая данные в ППЗУ, а затем считывая их в программную память, легко установить соответствие между числом в регистре и каждой семеркой команд (рис. 1.10):

Шаг	Код	Команда	Код
00	0 0	0	00
01	M7 M8	7	07
02	M5 M6	KX<0 A	C-
03	M3 M4	FX=0	5E
04	M1 M2	FBX	0
05	П2 ЗМ		EF
06	ЗП П1	KX≥0E	-L

Символы мантиссы отображаются на индикаторе на своих местах без каких-либо изменений. Если знак порядка равен нулю, то первый и второй разряды порядка также выводятся на индикатор без изменений. Знак мантиссы отображается в крайнем левом разряде индикатора (табл. 1.7). Если на месте знака мантиссы в видеосообщении стоит нуль, то он сохраняется и при записи в регистр, и после команды ↑.

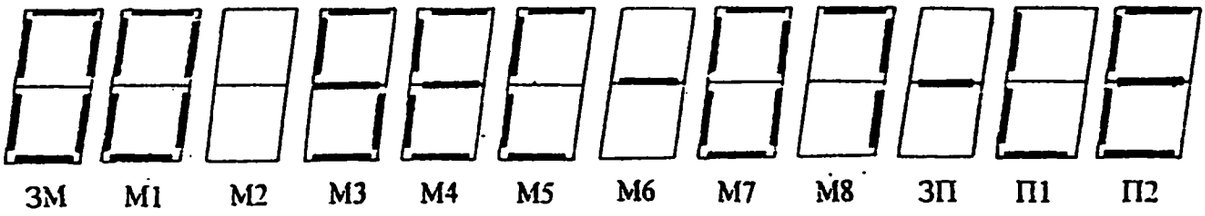


Рис. 1.10. Индикация фрагмента программы, считанного в память данных

ЗМ – знак мантиссы; М1-М8 – мантисса; ЗП – знак порядка; П1, П2 – порядок

Положение десятичной точки определяется цифрами порядка. В зависимости от них точка будет стоять соответственно после первого или второго, ..., восьмого символа мантиссы. Точку можно переместить с помощью команды ВП подобно тому, как это делается в видеосообщениях, полученных с помощью логических операций.

Таблица 1.7

Отображение символов на индикаторе при ЗП=9

Элемент кода	ЗМ	П1		П2
		П2 = 0	П2 ≠ 0	
0	F	0	9	0
1	2	9	8	9
2	3	8	7	8
3	4	7	6	7
4	5	6	5	6
5	6	5	4	5
6	7	4	3	4
7	8	3	2	3
8	9	2	1	2
9	-	1	0	1
-	L	0	F	0
L	C	F	E	F
C	Г	E	Г	E
Г	E	Г	C	Г
E	F	C	L	C
F	0	L	-	L

Если цифра в позиции знака порядка отлична от нуля или девяти, дело обстоит сложнее. Иные цифры например, 1, ..., 8, -, L, C, Г, F) весьма отличаются своим воздействием на микро-

калькулятор. Так, некоторые из них способны блокировать вывод индикации и парализовать вычислительный блок микрокалькулятора, другие отображаются на индикаторе в виде сообщений "ЕГГОГ" или "ЭГГОГ".

Если знак порядка равен нулю, то в десятом разряде индикатора окажется пробел.

Пример. Получим сообщение "00.SEC-07-LC" .

Определяем коды, из которых должна состоять "программа", и вводим ее:

Шаг	Код	Команда
00	00	0
01	07	7
02	C-	KX < 0A
03	5E	FX = 0
04	0F	FBX
05	EE	KX = 0E
06	9E	KX ≥ 0E

На шаге 05 требуется команда с кодом EF. Получить ее несложно. Введем программу в ППЗУ (например, по адресу 1000007), а затем вместо 5.KX=0E запишем 5.1 и еще раз нажмем клавишу ↑↓. Осталось считать эту запись из ППЗУ в память данных, нужное сообщение окажется в регистре 0.

Если попытаться записать это сообщение в другой регистр, оно примет вид "0.SEC-07-L1" .

Поскольку пробел отделен от крайнего разряда мантиссы нулем, попытка работать с видеосообщением может привести к "параличу" микрокалькулятора. Видеограмму следует уничтожить в стеке и регистре 0 командами CX ПО.

ЛОГИЧЕСКИЕ ОПЕРАЦИИ В ИГРОВЫХ ПРОГРАММАХ

Нестандартное использование логических операций рассмотрим на примере двух игр. Игра "Угадай число" (программа 1.5) является модернизацией известной игры, когда микрокалькулятор загадывает число, а игрок пытается его определить.

Учитель Е. Карпунькин из Нарьян-Мара предложил оценивать степень приближения к загаданному числу словами. Если число угадано, то на экране появляется слово BOSS, отсюда и название игры "Босс".

Программа этой игры состоит из следующих функциональных частей:

00-05: генератор псевдослучайных чисел;

06-11: преобразователь случайного числа в целое из интервала 1 ... 1 000 и запись результата в регистр;

12, 13: индикация готовности к работе (88888888) и останов для того, чтобы игрок ввел "пробное число";

14-83: определение разности между "задуманным" и "пробным" числами и в зависимости от нее выбор оценки;

84-85: переход к новой партии.

П р о г р а м м а 1.5

"Угадай число"

	00	01	02	03	04	05	06	07	08	09
00	ИПЕ	FPI	+	FEXP	К{X}	ПЕ	ИП9	*	1	+
10	К[X]	ПD	ИП0	С/П	ИПD	-	FX<>0	71	К|X|	6
20	-	FX>0	67	ИПА	-	FX>0	59	ИПВ	-	FX>0
30	55	ИПС	-	FX>0	48	ИП0	3	ВП	4	-
40	ИП1	К +	К{X}	↑	ВП	7	БП	13	ИП0	3
50	0	-	ИП2	БП	41	ИП5	ИП6	БП	41	ИП0
60	3	ВП	6	-	ИП3	БП	41	ИП7	ИП8	БП
70	41	ИП0	3	3	ВП	3	-	ИП4	К +	К{X}
80	↑	ВП	7	С/П	БП	00				

Инструкция

1. Исходный ввод констант: 88888888 – в регистр 0; 82606377 – в регистр 1; 808830607 – в регистр 2; 80638777 – в регистр 3; 8080777 – в регистр 4; 85188588 в регистр 5; 8003877 – в регистр 6; 8815885 – в регистр 7; 80002000 – в регистр 8; 999 – в регистр 9; 40 – в регистр А; 100 – в регистр В; 300 – в регистр С; стартовое число ξ_0 для генератора псевдослучайных чисел – в регистр Е (Это число $0 < \xi_0 < 1$ выбирает игрок).

2. Нажать клавиши В/О С/П. Появление на индикаторе числа 88888888 говорит о том, что микрокалькулятор число задумал.

3. Ввести пробное число и нажать клавишу С/П.

4. При появлении любой словестной оценки пробного числа, кроме слова BOSS, повторить п. 3. Оценка BOSS, означает, что число угадано и партия закончена.

5. Для начала новой партии нажать клавишу С/П и после появления на индикаторе числа 88888888 перейти к п. 3.

Особенность программы в том, что в подпрограмме, создающей видеограммы, работают команды БП 41 и БП 13, а не ПП и В/О. Такой прием иногда используют, особенно в тех случаях, когда вход в подпрограмму осуществляется с разных адресов, а выход должен быть по одному и тому же адресу.

В игре "Сокровище" по программе киевлянина Е. Бредни (программа 1.6) логические операции применяются для созда-

ния игрового поля, синтеза изображений ходов игрока и ответных ходов микрокалькулятора. Цель игры – добраться до "сокровища", находящегося слева на индикаторе и обозначенного цифрой 8.

Программа 1.6

"Сокровище"

	00	01	02	03	04	05	06	07	08	09
00	CX	П5	1	П4	6	П0	ПП	87	2	:
10	K{X}	П2	6	*	ИП4	1	+	FEXT	*	ИП1
20	+	П1	ИП5	ИП4	-	F1/X	ИП1	ИП4	FEXT	5
30	*	+	ИП5	FEXT	+	КИНВ	ИП6	XY	С/П	ИП2
40	FX<>0	72	ПП	87	П8	ПП	87	П9	ИП3	↑
50	↑	ИП8	↑	ИП3	↑	ИП9	↑	ИП3	С/П	ИП8
60	-	FX=0	80	С/П	ИП9	-	FX=0	80	ИП6	3
70	+	П6	КИП4	FL0	06	ИП6	8	+	П6	С/П
80	ИП6	3	-	П6	КИП5	БП	22	ИП7	FPI	+
90	FEXP	K{X}	П7	ВП	5	K[X]	В/О			

Сменный блок для второй части игры

	00	01	02	03	04	05	06	07	08	09
00	7	П0	1	ВП	6	ПА	ИП1	ИП0	FEXT	:
10	ИПА	+	K{X}	П2	ИП0	ИП5	-	БП	24	КНОП
20	КНОП	КНОП	БП	14	FX<>0	75	ИП1	ИП0		

С АДРЕСА 28 ПРОДОЛЖАЕТСЯ ОСНОВНАЯ ПРОГРАММА

Инструкция

1. Перед началом первой партии проделать операции 9 F1/X П3, выбрать стартовое число $0 < \xi_0 < 1$ для генератора псевдослучайных чисел и ввести его в регистр 7.
2. Перед началом каждой партии провести операции 7 F10^X П1 СХ П6.
3. Запустить программу командами В/О С/П.

Игрок движется справа налево. Его преследует чудище Е. После первого запуска на индикаторе отображается игровое поле. Положение искателя сокровищ указано знаком "-". Слева от этого знака может появиться дверь. На индикаторе ей соответствует буква С. Символ открытой двери – цифра 7. Если нажать клавишу С/П, то, когда двери перед игроком нет, он передвинется на одну позицию влево. Если же дверь была, то, как во всех сказках, ее нужно будет открыть с помощью заклинания.

На индикаторе на несколько секунд покажется пароль – два мерцающих целых числа из интервала 0 ... 9999. Мерцающие

числа постарайтесь запомнить. Чтобы привлечь внимание игрока, перед каждым из них на индикаторе мелькнет сообщение

"1,1111111-01"

Оно же останется на индикаторе после останова как сигнал к дальнейшим действиям.

Целые числа микрокалькулятор формирует из случайных чисел, создаваемых соответствующей программой – генератором случайных чисел. Этот же генератор служит для установки дверей. Появление двери зависит от целого случайного числа, формируемого перед каждым показом игрового поля. Дверь ставится лишь в том случае, если число нечетное. Это делает каждую партию отличной от предыдущей.

Когда на индикаторе появится приглашение "1,1111111-01" , введите первое из чисел пароля и нажмите клавишу С/П. Если память вас не подвела, то на индикаторе возникнет нуль. Введите второе число и нажмите клавишу С/П. Если и теперь все правильно, то на вновь появившемся игровом поле знак дверей "С" заменит цифра 7 – дверь открыта. Игрок стоит в дверном проеме, поэтому знака игрока "-" на индикаторе нет.

За каждую открытую дверь игроку начисляется три очка.

Если хотя бы одно число введено неверно, то на индикаторе вновь формируется игровое поле: игрок и дверь остаются на месте, а чудище "Е" продвигается на одну позицию левее, догоняя игрока. Кроме того, игрок получает три штрафных очка и должен повторить попытку – запомнить мерцающие цифры (они будут другими, нежели в первый раз) и с их помощью открыть дверь.

Игра завершается сигналом "ЕГГОГ", когда чудищу удается догнать игрока. Напротив, стоит игроку добраться до сокровища "8", к набранной сумме очков прибавится 8. Результат индицируется, причем, если неудачных попыток было много, он может оказаться отрицательным. Это говорит о том, что поиск сокровищ бывает и убыточным предприятием.

Для владельцев микрокалькуляторов МК-52 программу можно модернизировать. Добравшись до клада, игрок считывает из ППЗУ иной фрагмент начала программы. При этом сохраняется обстановка, предшествующая находке сокровища: на месте остаются двери и чудище "Е", покорно ожидающее, когда до него доберется разбогатевший или обедневший игрок. Как и раньше, необходимо преодолевать двери, открывая их уже известным способом. Но после потери клада чудищу и жизнь немила, поэтому, если игрок ошибается, то оно делает шаг к нему навстречу, а значит и навстречу собственной гибели. Таким образом, возможен единственный финал – победа игрока,

хотя и она нередко оборачивается потерями (об этом свидетельствует знак минус перед цифрами итога).

СЕКРЕТЫ МИКРОКАЛЬКУЛЯТОРА

Предлагаем читателю провести небольшой эксперимент: вызвать число π, прибавить к нему десять, а затем вычесть десять. Получится 3,141593. Отметим две интересные детали. Во-первых, на единицу уменьшилось число значащих цифр мантиссы, а во-вторых, последняя значащая цифра была округлена по правилам арифметического округления.

Зная эту особенность программируемого микрокалькулятора, можно округлять числа, сохраняя столько значащих цифр после запятой, сколько необходимо. Нетрудно запомнить простое правило: к исходному числу достаточно прибавить, а затем вычесть $10^{(7-n)}$, где n – число значащих цифр после запятой. Например, чтобы в числе π осталось два знака после запятой, проводят следующие операции:

```
Fπ 5 FEXT + FBX - "3,14"
```

Аналогичным приемом пользуются и для того, чтобы получать необходимое количество значащих цифр в целой части числа. Для этого к исходному числу следует прибавить, а затем вычесть число $10^{(7+m)}$, где m – число незначащих цифр. Незначащие цифры заменяются нулями, а последняя значащая цифра будет округлена по правилам арифметического округления.

Программируемые микрокалькуляторы семейства "Электроника" верно обрабатывают числа, не превышающие $9{,}9999999 \times 10^{299}$. Числа, равные $9{,}9999999 \times 10^{99}$ или меньшие, выводятся на индикатор в обычном виде, а бо́льшие, но не превышающие $9{,}9999999 \times 10^{299}$, скрыты сообщением "ЕГГОГ" или "ЗГГОГ". Тем не менее содержимое регистра X можно суммировать, умножать, делить, вычитать или возводить в квадрат. При вычислениях по программе ни запись в память при прямой и косвенной адресациях, ни обмен информацией между регистрами X и Y, ни продвижение или кольцевое перемещение информации в стеке не вызывают аварийного останова, если результат равен $9{,}9999999 \times 10^{299}$ или меньше.

В ходе непосредственных вычислений аварийный останов произойдет в любом случае, когда число имеет порядок более 99. Но при вычислениях по программе ее исполнение прервется лишь при вызове такого числа в регистр X из регистра па-

мяти или регистра предыдущего результата X1. На индикаторе будет все тот же сигнал аварийного останова "ЕГГОГ".

В отличие от случая, когда аналогичное сообщение появляется на индикаторе в результате некорректной операции (например, деления на нуль), микрокалькулятор ведет себя по-другому. Так, после

```
1 ВП 50 FX^2
```

"ЕГГОГ" – знак того, что микрокалькулятор не способен показать на индикаторе число 10^{100}, которое хранится в регистре X. Но, поскольку число там есть, его можно записать в память и проделать с ним некоторые операции:

```
П9 "ЕГГОГ" 2 * "ЕГГОГ" ИП9 "ЕГГОГ" : "2"
```

Таким образом, становится очевидным, что число 10^{100} все время скрывалось в регистре X. Иногда эту особенность программируемого микрокалькулятора удается использовать.

Хорошей иллюстрацией служит программа вычисления факториала, которая позволяет находить $n!$ для чисел, бóльших 69! (см. программы 1.1 и 1.2). Разумеется, при этом результат замаскирован сообщением "ЕГГОГ". Чтобы расшифровать его, используют несложный прием. После ввода программы, формируют число 10^{100} командами

```
1 ВП 50 ↑ * "ЕГГОГ"
```

и записывают его в один из свободных регистров памяти. Остается набрать число n и запустить программу.

Вычислим 166! Вводим

```
166 В/О С/П "ЭГГОГ" ,
```

вызываем записанное ранее в память число 10^{100}, которое скрыто под "ЕГГОГ". При этом результат вычислений 166! перемещается в регистр Y. Разделим содержимое регистра Y на содержимое регистра X, нажав клавишу деления. На индикаторе вновь "ЕГГОГ".

Известно, что при делении степеней с одинаковыми основаниями показатель степени делителя вычитается из показателя степени делимого.

Таким образом, зная, что показатель степени понижен на 100, можно утверждать, что поскольку число не индицируется, показатель степени больше, чем 99. Вызовем из памяти число 10^{100} и вновь проведем операцию деления. На этот раз на индикаторе "9.0036828 97" .

Поскольку показатель степени результата дважды уменьшался на 100, то 166! = $9{,}0036828 \times 10^{297}$.

Это, пожалуй, самое большое число, доступное нашей программе. Если попробовать вычислить 167!, произойдет сбой программы, управляющей всеми действиями микрокалькулятора. Он самопроизвольно переходит в режим программирования, а попытка вернуть его в автоматический режим командами F АВТ приводит к тому, что микрокалькулятор перестает выполнять любые команды, в том числе и команду "стоп". Чтобы восстановить его работоспособность, необходимо выключить питание (разумеется, будут потеряны программа и содержимое регистров памяти).

Прежде, чем продолжить разговор об операциях, не оговоренных "Руководством по эксплуатации", удобно разделить их на несколько групп.

К первой отнесем, например, ввод отрицательных чисел в регистры памяти, если они используются при косвенной адресации. В "Руководстве по эксплуатации" не указано, что адрес должен быть целым положительным числом.

По умолчанию* адрес, если он используется в командах переходов, должен быть только целым положительным числом, не превышающим числа шагов программной памяти. А если речь идет об использовании косвенной адресации для записи данных, то адрес не должен быть больше самого большого номера регистра памяти, а при записи адресов переходов – больше номера последнего шага программной памяти.

Ко второй группе относятся разрешенные операции, позволяющие получить эффект, не описанный в "Руководстве по эксплуатации". Примером является использование команд косвенной адресации КИПN (КПN) для выделения целой и дробной частей положительного числа (см. табл. 1.3). Подобные операции безусловно являются допустимыми. Приобретая опыт программирования, каждый пользователь находит новые операции этой группы. Приведем некоторые из них.

Для записи постоянной 0,125 требуется либо пять шагов программы, либо один регистр памяти и шаг программы для обращения к этому регистру. В то же время постоянную 0,125 можно сформировать всего двумя командами: 8 F1/X "0.125"

Аналогично создаются характерные числа, которые нередко используют в качестве сообщений:

```
9 F1/X "1,1111111-01" ,   2 ↑ 9 / "2,2222222 -01" ,
                3 ↑ 9 / "3,3333333 -01"
```

и т. д.

---

* Условие, которое существует, но специально не оговаривается, называется "условием по умолчанию".

Для определения четных и нечетных чисел переключатель Р-ГРД-Г ставят в положение Р (радианы). Умножив исследуемое число N на π, вычисляют косинус произведения. Если результат положительный, то число – четное, если отрицательный, – нечетное.

Впрочем, этот способ годится лишь тогда, когда число состоит из восьми или меньшего количества цифр. Большее число цифр не уместится в регистре X. В этом случае исходное число необходимо уменьшить для того, чтобы последнюю цифру можно было ввести в регистр X. Для чисел, в которых три знака и менее, четные числа индицируются цифрой 1, а нечетные – цифрой 1 со знаком минус. В микрокалькуляторе МК-61 в результате операций N FPI * FCOS КЗН получается "1", если число четное, и "-1", если нечетное.

Еще один пример. В ходе вычислений функции $\mathrm{arctg}(y/x)$ может встретиться случай, когда $x = 0$. Тогда в соответствии с алгоритмом работы микрокалькулятора произойдет аварийный останов. Избежать этого помогает небольшой фрагмент:

```
00.FX=0 01.05 02.ВП 03.9 04./-/ 05./ 06.FATN
```

Будем считать, что значение x введено в операционный регистр X, а значение y – в регистр Y. Поскольку в нашем микрокалькуляторе адрес перехода игнорируется тогда, когда условие выполняется, т. е. если $x \neq 0$, то после шага 01 управление переходит на шаг 05. Число, находящееся в регистре Y, делится на число из регистра X, а на следующем шаге вычисляется арктангенс. Но если $x = 0$, то выполняя второй, третий и четвертый шаги программы, микрокалькулятор заменяет содержимое регистра X на $1 \cdot 10^{-9}$ (читатель помнит, что результат операции 0 ВП – единица; это использовалось при вычислении $n!$). После двух последних шагов программы на индикатор микрокалькулятора выводится число, близкое к 90, и, самое главное, исключается аварийный останов.

В "Руководстве по эксплуатации" приведена таблица, где собраны данные о том, в каких пределах может изменяться аргумент вычисляемых функций. Если аргумент выйдет за эти ограничения, то нельзя гарантировать, что погрешность вычислений будет в указанных для микрокалькулятора пределах. Он не во всех ситуациях сигнализирует о том, что допустимые границы нарушены. Например, для синуса задан диапазон $1 \cdot 10^{-49} < |x| < 10^{10}$. Не будем придираться к тому, что не указано, в чем выражен аргумент – в градусах или радианах. Вычислим функцию для тех и других значений. При аргументе $1 \cdot 10^{-60}$, заданном в радианах, получаем $1 \cdot 10^{-60}$. Это соответствует правилу, что синус малого угла равен его аргументу. Повто-

ряя расчет для аргумента, выраженного в градусах, также получаем какое-то число, а при значениях, превышающих допустимые, – "ЕГГОГ". Вот и обнаружено новое значение этого сообщения – выход аргумента за заданные границы.

Функция x^y вычисляется по микропрограмме, реализующей известный прием:

$$x^y = \exp(y \ln x).$$

В этом случае x может быть только положительным числом. В то же время известно, что функция x^y определена для $x < 0$, если y – натуральное число (например, $(-2)^2 = 4$, $(-2)^3 = -8$). Определена также функция x^y, где $y=1/n$, если $x < 0$, а n – нечетное натуральное число (например, $(-8)^{1/3} = -2$).

Научить микрокалькулятор работать с такими числами можно с помощью дополнительных программ.

Например, программа 1.7 позволяет реализовать функцию x^y для любых значений x, кроме $x = 0$, если $|y|$ – целое число.

Программа 1.7

Возведение отрицательного числа x
в целую степень y при $y \neq 0$

	00	01	02	03	04	05	06	07	08	09
00	FX<0	11	FX^2	FSQR	FX^Y	XY	FPI	*	FCOS	*
10	С/П	FX^Y	С/П							

Перед пуском переключатель Р-ГРД-Г ставят в положение Р, в регистр X вводят число x, а в регистр Y – y и нажимают клавиши В/О С/П. Например: $(-2)^{-3} = -0{,}125$. Проверим:

3 /-/ ↑ 2 /-/ В/О С/П "-1.2500001-01"

Правильно.

В нашем микрокалькуляторе все миниатюрно. Мало регистров памяти, мало шагов программы. Код команды состоит из двух цифр. И порой удивляешься тому, каких результатов можно достичь при таких скромных возможностях.

Мы уже отмечали, что в программируемом микрокалькуляторе не предусмотрено автоматическое предупреждение пользователя о том, что неправильно введен адрес в тех случаях, когда число, являющееся адресом, отрицательное, нецелое, больше возможного числа шагов программы или количества регистров. Если число, являющееся адресом, положительно и имеет целую и дробную части, то при косвенном обращении микрокалькулятор отбрасывает дробную часть числа. А если при косвенной

адресации к программе адрес представлен числом, имеющим более двух разрядов, то номер шага, куда будет передано управление, определяется числом, составленным из цифр в двух младших разрядах.

Почему же об ошибках пользователя микрокалькулятор не сигнализирует? У него не хватило памяти. Этим же объясняется и то, что сигнал об ошибке ("ЕГГОГ") один в совершенно различных ситуациях: о том, что результат вычислений не может быть выведен на индикатор; о том, что операция некорректна (например, деление на нуль вычисление функций arcsin$x$ или arccos$x$ для $x > 1$).

Увидев сигнал аварийного останова, пользователь не знает, почему остановилась программа. Однако вопрос решается просто. Достаточно нажать клавишу продвижения информации в стеке вверх. Если "ЕГГОГ" по-прежнему на индикаторе, то произошло переполнение, а если появится какое-либо число, то была некорректная операция. Кстати, это появившееся число и есть виновник некорректной операции. Такого же эффекта можно достичь, если после аварийного останова попытаться направить число из регистра X в любой регистр памяти.

Иногда о причинах ошибки и аварийного останова можно узнать, нажав клавишу ВП. Изменение сигнала на индикаторе говорит о причинах останова. Появление перед "ЕГГОГ" знака минус, означает, что вычислялись: логарифм, квадратный корень, x^y от отрицательного аргумента; e^x при $x \leq -231$ или 10^x при $x \leq -100$.

Если в результате расчета получается число, порядок которого 200 или более, то выполняется переход на шаг программы, номер которого составляют две первых цифры числа, определяющего порядок. Так, при вычислении 166! счетчик адресов оказывается на шаге 29 (порядок результата равен 297). Возврат к началу программы проводят вручную командой В/О.

У аварийного останова есть еще одна особенность. Если он происходит при работе программы, то некорректная операция "съедает" один шаг программы (содержимое счетчика адресов увеличивается на единицу). Аналогичным образом микрокалькулятор функционирует при нажатии клавиши С/П во время исполнения программы.

И в том и в другом случае, прежде чем продолжать работу, необходимо восстановить содержимое счетчика адресов, нажав клавишу ШГ ("шаг влево"). Впрочем, здесь необходимо учитывать и другую особенность: если останов произошел после адреса перехода, то клавишу ШГ нажимают дважды.

И, наконец, следует помнить еще об одном исключении – если результатом останова было переполнение, то содержимое счетчика адресов не нуждается в коррекции.

Операции, которые использовались для получения букв, начинались аварийным остановом, обусловленным некорректной операцией. Но почему этот сигнал можно превратить в число – одна из многих тайн микрокалькулятора. К сожалению, у наших микрокалькуляторов подобных тайн настолько много, что иногда начинаешь опасаться за достоверность полученного результата.

Вычисляя функцию 10^x при $x = -99{,}1$ получаем 0. Это так называемый машинный нуль*. Как это не удивительно, но при $x = -100$, получается не нуль, а "ЕГГОГ". Казалось бы, как ни уменьшай аргумент, результатом должен оставаться машинный нуль. Однако микрокалькулятор классифицирует эту операцию как некорректную.

Если при $-230{,}26 < x < -227{,}96$ микрокалькулятор, вычисляя e^x, полагает эту функцию равной нулю и дальнейшее уменьшение аргумента расценивает как некорректность, то на индикаторе высвечивается "ЕГГОГ".

Эти и другие подобные недостатки нередко выявляются при отладке программ и могут быть учтены. Значительно хуже дело обстоит с теми операциями, которые правильно выполняются в режиме ручного счета и пошаговой проверки, но дают неверные результаты при работе по программе. Такие операции превращают микрокалькулятор из помощника в запутанную головоломку.

Этому способствуют иногда и некоторые популярные журналы, публикуя не только сообщения об аномалиях, но и программы, построенные на остроумном использовании таких аномалий. Знание о них позволяет избежать досадных ошибок, применение же опасно тем, что если в процессе производства или в новой модели аномалия будет устранена (а так уже бывало – например, отменен запрет на использование перед командой В/О команд одноместных операций), то потеряют ценность все программы, основанные на ее употреблении. А поскольку опубликовано немало таких программ, то будущему поколению программистов придется ломать голову над вопросом: почему та или иная программа работала на старых моделях микрокалькуляторов и не работает у нас?

В поисках несанкционированных операций совершалось немало "открытий". Например, автор одного из писем в редакцию журнала "Наука и жизнь" сообщает о том, что ему удалось обнаружить 20 регистров памяти в микрокалькуляторе МК-52, и делает многозначительный вывод, что "находка" значительно расширяет возможности микрокалькулятора.

На самом же деле никаких дополнительных регистров в

* Результат менее $1 \cdot 10^{-99}$ микрокалькулятор приравнивает к нулю.

микрокалькуляторе нет. Однако, если нажать клавиши П или ИП а затем любую из клавиш арифметических действий или XY, то число будет записано в один из регистров памяти или считано оттуда. Обращение происходит к регистру, номер которого легко определить по коду команды, клавиша которой нажата. Номер определяет младшая цифра кода: например, код команды "+" – это "10", т. е. команда П+ эквивалентна команде П0. Разумеется, то же правило распространяется и на команды косвенной адресации: например, команда КП XY (код XY – 14) эквивалентна КП4.

Еще одна особенность наших микрокалькуляторов, не оговоренная в "Руководстве по эксплуатации", заключается в том, что при программировании после команды перехода можно нажать не две цифровые клавиши, записывающие в программу адрес, а какие-нибудь другие. Подобно тому, как команда П+ эквивалентна команде П0, так и БП ++ эквивалентна БП 00, а FX<0 +- заменяет FX<0 01 (код команды "+" – 10, а команды "-" – 11).

Есть и исключения из этого правила. К ним относятся прежде всего команды С/П и В/О. При нажатии, например, вслед за БП клавиши С/П записывается после кода 51 код 50, который в таком сочетании будет выполнять роль адреса перехода.

Если есть дополнительные возможности, независимо от того, описаны они в инструкции, или нет, возникает желание их использовать. Программы для микрокалькуляторов наиболее насыщены непонятными для непосвященного структурами и к тому же редко снабжаются комментариями, поэтому разобраться в них трудно не только начинающим.

Вот один из таких примеров:

```
00.FX=0 01. FX^Y 02.ПП 03.FL2 04.КП+ 05.БП 06.В/О  ,
```

что означает

```
00.FX = 0 01.24 02.ПП 03.58 04.КП0 05.БП 06.52
```

Хорошая программа не должна напоминать кроссворд – возможно, с ней придется работать не только ее автору.

Большинство операций, из-за которых программа выполняется по-разному в режимах вычислений и пошаговой проверки, связаны с командой ввода порядка ВП. Ее удобно использовать для умножения или деления исходного числа на 10^n, когда число вводится с клавиатуры, вызывается из регистра памяти или записано непосредственно в программе.

Если число получено в результате любой другой операции, то команда ВП будет проигнорирована. Так, программа

```
00.CX 01.FPI 02.ВП 03.2 04.С/П
```

при пошаговой проверке дает 314,15926, а при автоматическом выполнении – 100*.

Чтобы обойти этот недостаток, вместо команд ВП n, где $c = |N|$, а N – порядок числа, для $N > 0$ можно использовать команды n FEXT * , а для $N < 0$ – команды n FEXT / .

Этот прием удлиняет программу на один шаг, если выполняется умножение, и не удлиняет ее, если производится деление, поскольку для $N < 0$ после ВП n добавляется команда смены знака /-/.

Впрочем, есть еще один прием: команда ↑ перед ВП все ставит на свои места, и программа работает без ошибки. При этом, однако, теряется информация, находящаяся в регистре T, как и при использовании функции 10^x.

Если же в программе встречается комбинация ПN ВП , т. е. копия записывается в регистр N, а лишь затем содержимому регистра X приписывается порядок, то при выполнении программы отбрасывается первая значащая цифра. Такой прием нередко используется в программах для микрокалькуляторов Б3-34 при выделении дробной части числа n при $1 < n < 10$.

В этом же интервале чисел фрагмент ↑ ПN ВП - выделяет целую часть числа. Если число являлось результатом вычислений, то фрагмент не будет работать так, как нам этого хотелось бы. Эффект проявляется в полной мере, если число вызывалось из памяти или создавалось программно.

Еще одна особенность работы микрокалькулятора связана со случаем, когда после ввода порядка и одноместной операции необходимо изменить знак результата.

Результаты пошаговой проверки и программного исполнения фрагмента

```
00.ИП9 01.ВП 02.2 03.ОМО 04./-/ 05.С/П
```

некоторых одноместных операций (x^2, lg, sign$x$ и $[x]$ – выделение целой части числа), обозначенных ОМО (в регистре 9 записано 2,5):

ОМО	Пошаговая проверка	Работа по программе
FX^2	"-6.25"	"2.5 -02"
Flg	"-3.9793999"	"2.5 -02"
K3H	"-1"	"2.5 -02"
K[X]	"-2"	"2.5 -02"

* Команда 01.FPI проигнорирована, а 0 превращен командой ВП в цифру 1. Это уже встречалось при вычислении $n!$.

Если раньше мы сталкивались лишь с тем, что микрокалькулятор игнорирует команду, стоящую перед командой ВП, то теперь ясно, что при смене знака микрокалькулятор способен игнорировать команду, стоящую после ВП. Способ, позволяющий обойти подобные явления, был найден еще для микрокалькуляторов Б3-21, неправильно вычислявших функцию x^y. Уже тогда ею обычно не пользовались, вычисляя при необходимости $10^{y\lg x}$ или $e^{y\ln x}$.

Экспериментируя, энтузиасты установили, что если перед операцией FX^Y дважды провести операцию F1/X, значение функции определяется правильно. Прием вызвал появление лавины подобных.

Всякий раз, когда расходятся результаты, полученные при пошаговой проверке и автоматических вычислениях, а текст программы введен правильно, ищите виновницу – команду ВП. Перед нею или после нее поставьте команду продвижения информации в стеке. Это помогает в подавляющем числе случаев. Ну, а если попытка не удалась и команду ВП нельзя заменить, то придется воспользоваться одним из "шаманских" наборов:

XY XY F→ FBX ; ИПN XY ВП ВП

Каждый раз, встречая в программах подобные сочетания, не вытекающие из логики работы, следует проверить, а не являются ли они следствием борьбы с недостатками микрокалькулятора или "находками" создателей микрокалькуляторных игр и фокусов. Кому придет в голову, что фрагмент CX XY XY ВП извлекает из регистра Y копию хранящегося там числа и показывает ее в регистре X, изменив так, что узнать исходное число невозможно, не заглянув в регистр Y.

Например, если в регистре Y число $|N| \geq 1$, то микрокалькулятор возьмет модуль этого числа, отбросит все значащие цифры, кроме первой, и увеличит ее на единицу. В таком виде число окажется в регистре X. Содержимое регистра Y не изменится. А если, например, записать в регистр X число, а в Y – другое, то команды XY XY ВП заменят у числа, находящегося в регистре X, первую цифру. На ее месте окажется первая цифра числа из регистра Y.

Размеры книги не позволяют рассмотреть все подобные ухищрения, а разгадать их достаточно трудно, так как адресация не очевидна и пошаговая проверка не помогает понять, что делает данный фрагмент при работе программы. Кроме "хитрых" приемов, заставляющих новичков часами разгадывать программы-головоломки, из программы в программу переходят неэффективные приемы.

В литературе нередко можно встретить фрагменты, заимствованные из других программ. Так, в программах для микрокалькуляторов МК-61 появляются фрагменты, которые легко заменить одной-двумя командами. По многим программам кочует комбинация команд 1 ВП n , занимающая три шага, хотя эффективнее фрагмент n FEXT всего из двух шагов.

Число примеров нетрудно увеличить, но мы остановимся, предоставив пытливым читателям самостоятельно углубляться в тайны микрокалькулятора.

До чего же бывает обидно, когда отлаженная и работающая программа вдруг искажается, в ней нарушаются связи, появляются новые команды и микрокалькулятор начинает своевольничать. Виноват обычно пользователь. Иногда это случается из-за пренебрежения правилами, а иногда из-за желания получить эффект, не предусмотренный разработчиками, т. е. при выполнении несанкционированных операций.

Попробуем объяснить, к чему могут привести некоторые нестандартные действия. По сути мы уже начали это рассмотрение, но пока что речь шла лишь о различиях в работе при пошаговой проверке и в режиме вычислений по программе.

Обычно оператор В/О завершает текст подпрограммы, а команда В/О служит для установки на нуль счетчика шагов программы, поэтому многие программы начинаются командами В/О С/П. Однако иногда оператор В/О используют для перехода на адрес 01, чтобы сэкономить один шаг программы (если переход осуществлять так, как указано в "Руководстве по эксплуатации", потребуется два шага для команд БП и 01).

Посмотрим, всегда ли можно воспользоваться таким приемом? Очевидно, что в этом случае оператор В/О должен быть вне подпрограммы, поскольку в противном случае В/О, как и должно быть, передает управление к шагу, на котором было прервано выполнение основной программы.

Еще одно условие, которое должно быть выполнено, если только мы рассчитываем передать управление на шаг 01, – это "чистый" адресный стек подпрограмм. Как и в основном стеке, здесь действует правило: вошедший туда последним, выходит первым.

Адресный стек подпрограмм рассчитан на пять адресов. Однако можно заслать в него и шестой, и *n*-й адреса. Разумеется, при этом из стека безвозвратно исчезнет и первый, и второй и (*n* – 6)-й адреса. Сигнал о том, что стек переполнен и возможна потеря информации, микрокалькулятор не дает.

Напомним, что по команде ПП в стек адресов записывается адрес, на котором было прервано естественное выполнение программы. По команде В/О к адресу, записанному в стек по-

следним, прибавляется единица, и управление передается по такому модифицированному адресу. Он при этом выталкивается из стека и заменяется записанным ранее или нулем, если ничего не записывалось. Если стек заполнен, то первая команда В/О вызывает продвижение всех адресов на одну позицию ближе к выходу. В пятом регистре стека появляется адрес-диверсант, обе цифры которого совпадают с последней цифрой адреса, находившегося там прежде. Каждое исполнение команды В/О, встретившееся в программе, будет заполнять еще одну ячейку стека копией адреса-диверсанта, пока копии не займут весь адресный стек. Очередная команда В/О вызовет передачу управления на адрес-диверсант, увеличенный на 1.

Проверить, чист или нет стек адресов, можно, записав на любой свободный шаг команду В/О. В режиме автоматических вычислений командами БП AD устанавливают счетчик адресов на шаг AD, где записана команда В/О. Остается нажать клавишу ПП и возвратиться в режим программирования. Число в правом углу индикатора укажет, на какой шаг было передано управление командой В/О. Если до эксперимента микрокалькулятор был выключен, то на счетчике адресов будет 01 – признак того, что стек адресов чист. В противном случае управление окажется на каком-то ином шаге. Не мудрствуя лукаво, лучше всего выключить и включить микрокалькулятор, очистив стек адресов.

Есть еще один "шаманский" способ, позволяющий очистить стек адресов подпрограмм, не выключая питание:

```
CX ↑ ВП 50 FX^2 "ЕГГОГ" FX^2 "Э.ГГОГ" CX XY CX
```

Объяснить связь между операцией очистки адресного стека и этими командами авторы не берутся.

Работая с логическими операциями и конструируя видеограммы, мы иногда сталкиваемся с пустым местом, а именно так, как известно, выглядит на индикаторе шестнадцатиричная цифра F. Эта "пустышка" далеко не безобидна. Оказавшись на первом месте, она уродует программу, а иногда и приводит микрокалькулятор к параличу, излечивает который только выключение питания. Владельцы МК-52, предусмотрительно записавшие программу в ППЗУ, теряют на ее вызов секунды, а всем остальным приходится вновь вводить программу и константы, да еще и заниматься поиском ошибок, которые могут появиться при вводе.

Трудно перечислить все нерегламентированные операции, приводящие к появлению буквы F в первой позиции индикатора.

В качестве примера рассмотрим небольшой фрагмент программы, часто используемый для создания видеограмм:

00.ИП8 01.ИП9 02.К⊕ 03.К{X} 04.↑ 5.ВП 06.5 07.С/П

Шаги 00 и 01 вызывают операнды в операционные регистры. Происходит сложение операндов по модулю 2, затем из видеограммы исключается первая цифра 8 (шаг 03), а запятая переносится в конец слова (шаги 04-06). Если в регистры 8 и 9 записать числа 88588 и 88063, то в результате исполнения этой программы получится оценка, которую нередко используют любители игр: 8,0SEL .

Результаты логических операций выглядят на индикаторе как десятичная часть числа, при этом нуль в разряде целых микрокалькулятор не индицирует. Если из полученного числа выделить дробную часть, то нуль пропадет и получится "S.EL -02" .

Чтобы этого не случилось, сконструируем видеограмму, в которой десятичная часть числа начнется пробелом (буквой F). Теперь в регистры 8 и 9 записываются операнды 888588 и 878063.

Теоретически должно получиться "0SEL" .

Вводим программу, записываем в регистры памяти операнды и нажимаем клавиши В/О и С/П. Хотя на индикаторе через несколько секунд появляется желанный "0SEL", кричать надо не "Ура!", а "Караул!", поскольку, если попытаться продолжать работу по программе, выясняется, что ее нет, что она уничтожена. В регистрах памяти вместо констант появился 0SEL, а в микрокалькулятор вселился "бес неповиновения". Команды либо не выполняются совсем, либо выполняются не так как надо.

Изгнание "осла" из регистра X нажатием клавиши СХ мнимое. Чтобы в этом убедиться, попробуйте воспользоваться командой ↑. Вместо того, чтобы исчезнуть, "осел" вновь появляется в регистре X. И вновь, чтобы "привести в чувство" микрокалькулятор, его нужно лишить питания.

Появление пробела часто связано с неосторожной попыткой избавиться от цифры 8, которой начинаются все видеограммы. При этом нередко забывают, что выполняется тождество $\overline{0} \equiv F$. Работая с микрокалькуляторами Б3-34, МК-54 и -56, для которых буквы и видеограммы получались в результате преобразований буквы E, можно, нечаянно получить и F, однако в этом случае делу еще можно помочь.

Если сразу после получения F (характерным признаком ее появления является одна-единственная точка, светящаяся на темном поле индикатора) не менее четырех раз нажать на клавишу ↑, на индикаторе появится нуль, и можно считать, что

крупных неприятностей удалось избежать. Но если не избавиться от пробела, программа и данные будут "убиты".

Полезно помнить некоторые конструкции, приводящие к появлению "пустого места" (НКО – это некорректная операция: К-, К+, К* и операция деления):

```
СХ НКО ВП КНОП ,
СХ НКО ВП ↑ ,
СХ НКО ВП ПN .
```

Операция очистки регистра СХ показывает, что все неприятности происходят тогда, когда в регистре Х нуль.

Ошибки нередко случаются из-за того, что при работе микрокалькулятора Б3-34 фрагментом

```
СХ НКО ВП ВП
```

создают букву Г.

Достаточно забыть, что клавишу ВП необходимо нажать два раза, а не один, и возникает критическая ситуация, из которой мы теперь умеем выходить.

ДЛЯ ЧЕГО НУЖЕН ТЕСТ

Для чего нужен тест? Прежде всего для того, чтобы отыскать ошибки, которые могли возникнуть во время ввода программы. Хороший тест, кроме того, способен выявить некоторые ошибки алгоритма, перенесенные в программу, или ее дефекты, связанные с ограниченными возможностями микрокалькулятора. Впрочем даже самый хороший тест не способен гарантировать полное отсутствие ошибок в программе. Рассмотрим конкретный пример.

Программист решил научить микрокалькулятор игре в "чет-нечет". Суть ее в том, что один из игроков задумывает число – нуль (чет) или единицу (нечет), а второй пытается его угадать. Если это удается, второй игрок выигрывает, в противном случае победитель – первый игрок. Программист загадывал числа, оставляя микрокалькулятору роль угадывающего.

Из теории игр известно, что наилучшая стратегия в подобных играх – это случайный выбор чисел нуль или единица. Однако не знающий оптимальной стратегии может

избрать какую-то другую. Наш Программист решил "усилить" игру микрокалькулятора, заложив в программу две стратегии.

Стратегия 1:

если четыре последних хода человека представляют собой одну из комбинаций 0000, 1000, 0101, 0110, 0011, то микрокалькулятор отвечает ходом "0";

на комбинации 1100, 1001, 0111, 1111 – ходом "1";

во всех остальных ситуациях микрокалькулятор выбирает нуль или единицу по случайному закону.

Эта стратегия является попыткой учесть психологию игрока и по его четырем последним ходам спрогнозировать пятый. Микрокалькулятор пользуется стратегией 1 только тогда, когда его очередной ход был выигрышным. В противном случае следует переход к стратегии 2.

Стратегия 2. В алгоритм этой игры (рис. 1.11) Программист ввел запись шести последних ходов человека, используя семь регистров памяти: 0, 1-3, 7-9 (программа 1.8). В регистр 0 записывается текущий ход. Он же "проталкивает" очередь в регистрах 1-3, 7-9, а прежнее содержимое регистра 9 теряется. Цепочка суммируется и делится на шесть. Обозначим эту операцию буквой P. Результат операции сравнивается с псевдослучайным числом, получающимся при программной реализации формулы

$$\xi_{i+1} = \{\arccos[\cos(\xi_i 10^9 + \Delta)]\}/\pi,$$

где ξ_{i+1} и ξ_i – псевдослучайные числа; Δ – ошибка программируемых микрокалькуляторов семейства "Электроника" при вычислении косинуса большого аргумента, заданного, в радианах.

Выбор микрокалькулятором хода α происходит по условию

$$\alpha = \begin{cases} 0, \text{ если } & \xi_{i+1} < P, \\ 1, \text{ если } & \xi_{i+1} > P, \end{cases}$$

где P – апостериорная вероятность того, что игрок задумал единицу, определенная по шести его предыдущим ходам. Этой стратегией микрокалькулятор пользуется при первом ходе и после любого ошибочного.

Во время описания стратегии начался разговор об их программной реализации. Продолжим его.

Содержимое четырех регистров памяти, в которых хранятся четыре последних хода игрока, рассматривается как число, записанное в двоичной системе. Оно пересчитывается в десятичное кодовое число (операция КЧ), в зависимости от которого происходит выбор нуля или единицы.

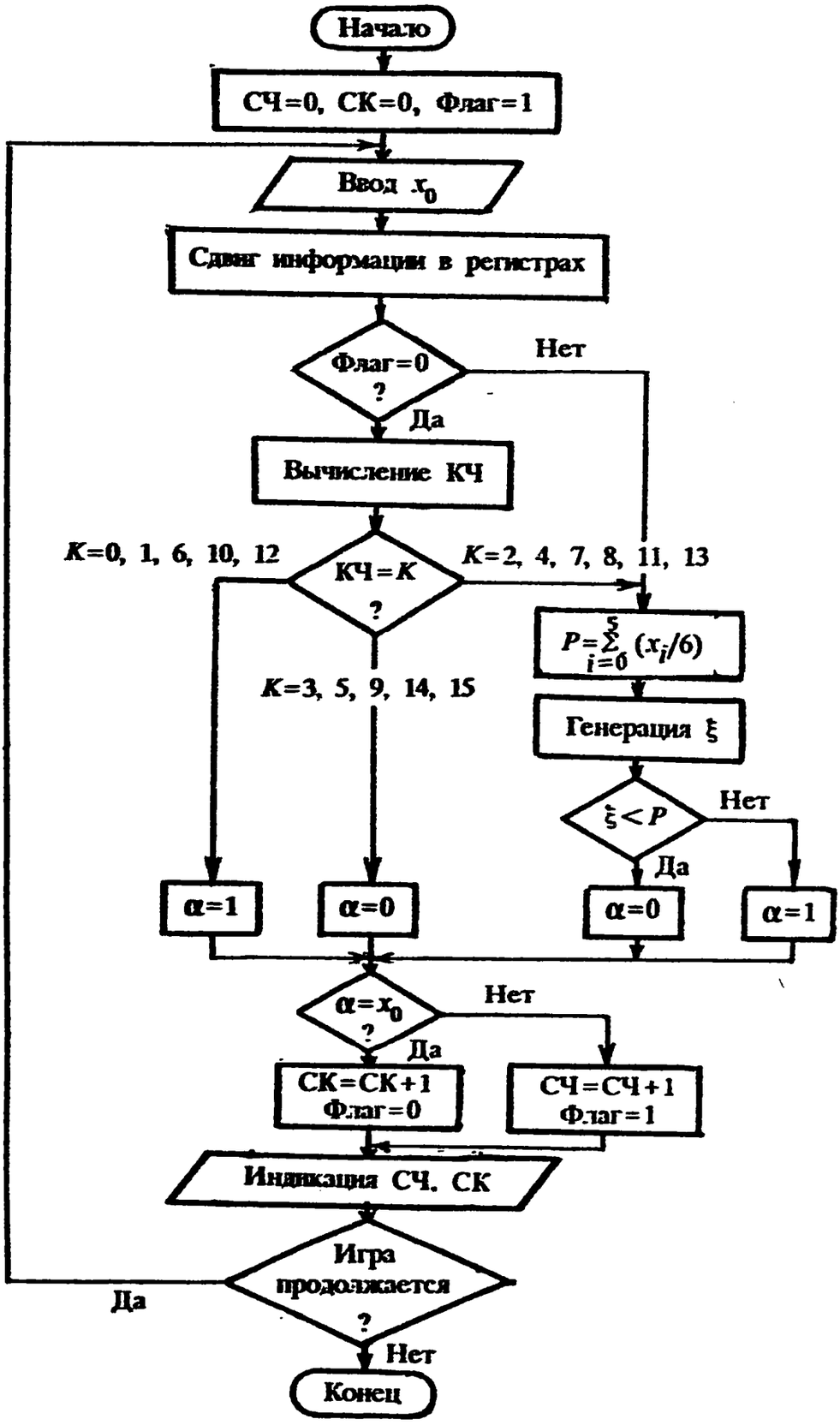


Рис. 1.11. Алгоритм-1 игры "Чет-нечет"

СЧ, СК – счетчик побед человека и микрокалькулятора; x_0 – ход игрока; КЧ – кодовое число; P – апостериорная вероятность того, что игрок задумал "1", определенная по шести его предыдущим ходам; ξ – псевдослучайное число.

Между кодовым и двоичным числами существует однозначное соответствие:

0 - 0000	4 - 0100	8 - 1000	12 - 1100
1 - 0001	5 - 0101	9 - 1001	13 - 1101
2 - 0010	6 - 0110	10 - 1010	14 - 1110
3 - 0011	7 - 0111	11 - 1011	15 - 1111

В алгоритме-1 игры "Чет-нечет" (см. рис. 1.11) есть и новая операция – "флаг". В данном случае "флагом" служит ответ на вопрос: угадано ли было предыдущее число? Если "да", то микрокалькулятор пользуется стратегией 1. Если же "нет", происходит переход к стратегии 2.

П р о г р а м м а 1.8

"Чет-нечет-1"

	00	01	02	03	04	05	06	07	08	09
00	ИП8	П9	ИП7	П8	+	ИП3	П7	+	ИП2	П3
10	+	ИП1	П2	+	ИП0	П1	+	6	/	ПD
20	ИП6	ИП5	-	ИП5	С/П	П0	КИП6	ИП4	FX<>0	71
30	ИП1	2	*	ИП2	+	2	*	ИП3	+	2
40	*	ИП7	+	КХ<>0А	1	-	КХ<>0А	2	-	КХ<>0В
50	2	-	КХ<>0В	1	-	КХ<>0А	3	-	КХ<>0В	1
60	-	КХ<>0А	2	-	КХ<>0А	2	-	КХ<>0В	1	-
70	КХ<>0В	КИП4	ИПС	ВП	9	FCOS	FACS	FPI	/	ПС
80	ИПD	-	КХ<>0В	ИП0	FX=0	00	БП	91	ИП0	FX<>0
90	00	КИП5	СХ	П4	БП	00				

Инструкция

1. Перед началом игры переключатель Р-Г установить в положение Р и ввести постоянные: 0 – в регистры 2, 5, 6, 7, 9, С; 1 – в регистры 0, 1, 3, 8; 81 – в регистр А; 88 – в регистр В. Нажать клавиши В/О С/П. Появление на индикаторе нуля означает, что микрокалькулятор готов к игре.

2. Ввести в регистр Х задуманное число 0 или 1 и нажать клавишу С/П. Через несколько секунд на индикаторе появится число побед микрокалькулятора. Чтобы узнать число побед игрока, следует нажать клавишу ХУ.

3. Для продолжения игры необходимо повторить п. 2.

Чем больше в шести предыдущих ходах одинаковых цифр, тем более вероятно, что микрокалькулятор будет прогнозировать изменение игроком его стратегии. Если все шесть цифр одинаковы, то вероятность этого достигает единицы.

Так происходит потому, что генерируемые числа лежат в диапазоне 0 – 1, но крайних значений никогда не принимают (не могут быть нулем или единицей), а P может быть нулем, если игрок шесть раз подряд задумывал нуль, и единицей, если шесть раз подряд была единица.

Если шесть последних ходов человека были одинаковыми, то при пользовании стратегией 2 микрокалькулятор будет прогнозировать изменение хода с вероятностью единица. В соответствии со стратегией 1 четыре одинаковых хода, сделанные человеком подряд, с вероятностью единица приведут к прогнозу, что и следующий ход будет повторением предыдущих.

Зная эти два обстоятельства, несложно выработать выигрышную для игрока стратегию.

Необходимо шесть раз подряд задумывать одну и ту же цифру, не обращая внимания на ответные ходы микрокалькулятора. При этом он перейдет к стратегии 1. На седьмой раз следует ввести другую цифру. Микрокалькулятор обязательно ошибется и перейдет к стратегии 2. Теперь и игрок должен вернуться к исходной цифре и вводить только ее.

Это еще не окончательная победа, так как в шести последних ходах только пять одинаковых цифр и $P = 5/6$ (для единиц) или $P = 1/6$ (для нулей). В первом случае игрок одержит окончательную победу, если в следующей шестерке случайных чисел не будет числа, которое равно 5/6 или больше, а во второй – меньше 1/6. В противном случае игроку придется начинать все сначала.

Поскольку вероятность появления случайных чисел, бóльших 5/6, и чисел, меньших 1/6, невелика, используя эту стратегию, игрок достигнет окончательной победы над автоматом, если не в первой, то в последующих попытках.

Победная стратегия является своеобразным тестом, выявившим дефект алгоритма, возможно, и не последний.

Игра достаточно привлекательна и заслуживает того, чтобы избавить алгоритм от недостатков. Новый вариант его приведен на рис. 1.12 и реализован в программе 1.9.

В роли отгадывающего могут выступать как игрок, так и микрокалькулятор. Задуманное число не вводится в микрокалькулятор заранее. Это уступка человеческой психологии.

Конечно, микрокалькулятор – честнейший партнер (не подсматривает и не подслушивает). Однако некоторым игрокам, особенно детям, не нравится требование заранее сообщать задуманное число. "Я ему скажу, что я задумал, а он после этого угадает?! – заявляют они. – В такую игру я не играю."

Принципиальные изменения заключаются в том, что при ошибке в угадывании вырабатываются числа, вероятность появления среди которых нуля или единицы равна 0,5.

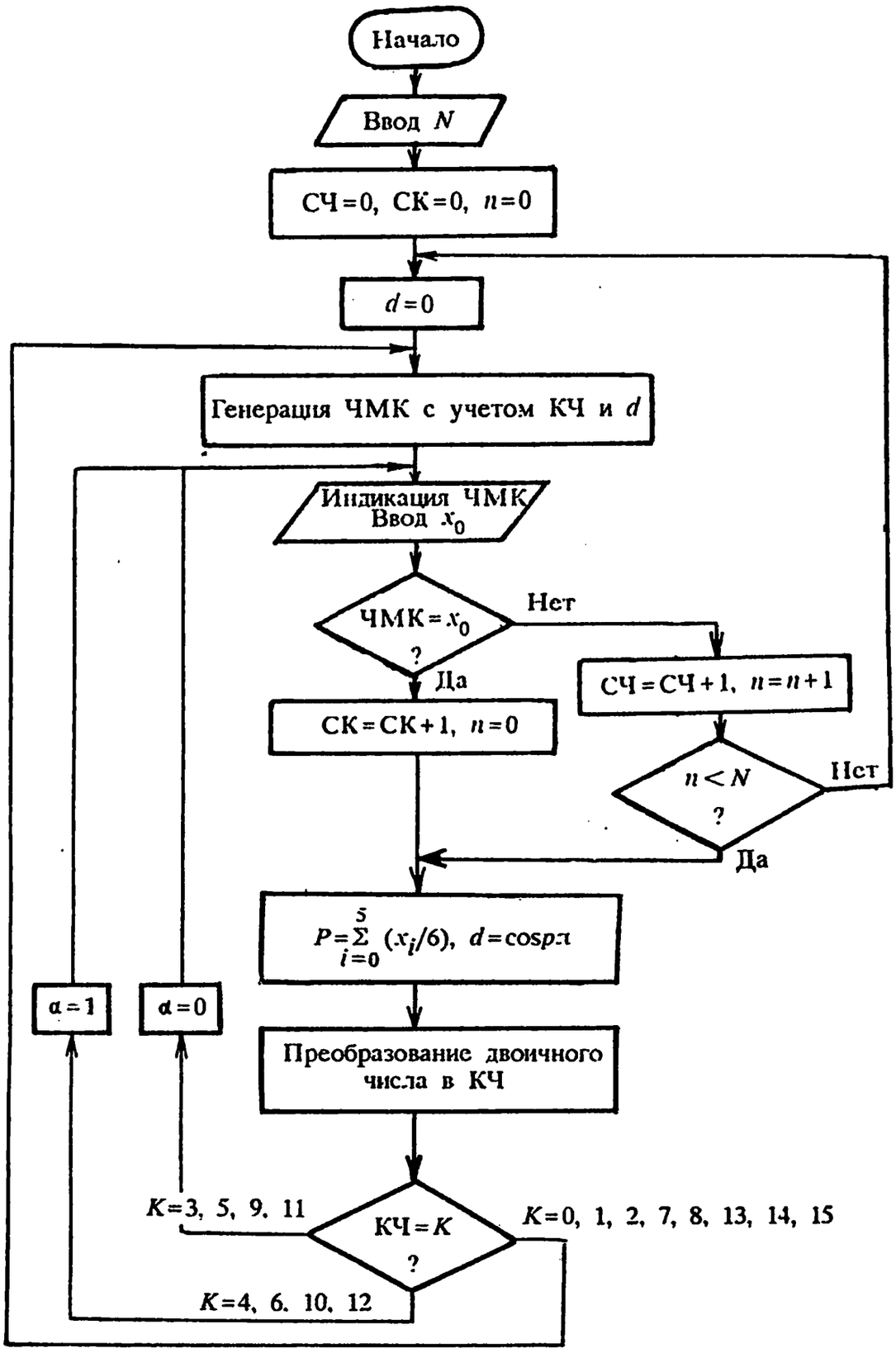


Рис. 1.12. Алгоритм-2 игры "Чет-нечет"

N – целое число от 1 до 9; ЧМК – число микрокалькулятора (0 или 1)

П р о г р а м м а 1.9

"Чет-нечет-2"

	00	01	02	03	04	05	06	07	08	09
00	СХ	ПD	ИП0	ВП	9	FCOS	FACS	FPI	/	П0
10	2	*	1	-	ИПD	-	КХ>0А	1	БП	21
20	СХ		С/П	-	FBX		ИПВ	+	ИП7	П8
30	+	ИП3	П7	+	ИП2	П3	+	ИП1	П2	+
40	6	/	FPI	*	FCOS	ПD	F→	П1	XY	FX=0
50	56	КИП5	СХ	П4	БП	58	КИП4	КИП6	ИП6	ИП5
60	С/П	ИП9	ИП4	-	FX>0	00	ИП7	2	*	ИП3
70	+	2	*	ИП2	+	2	*	ИП1	+	КППС
80	КППС	БП	02	3	-	КХ<>0В	1	-	КХ<>0В	1
90	-	КХ<>0А	1	-	КХ<>0В	В/0				

Инструкция

1. Переключатель Р-Г поставить в положение Р. Ввести константы: стартовое число генератора случайных чисел $0 < \xi_0 < 1$ – в регистр 0; 3 – в регистр 9; 20 – в регистр А; 17 – в регистр В; 83 – в регистр С.

2. Задумать число 0 или 1 и нажать клавишу С/П (в первый раз – В/0 С/П). После останова на индикаторе – прогноз микрокалькулятора о задуманном числе.

3. Сравнить с показанным задуманное число. Если микрокалькулятор угадал, то нажать клавишу С/П, в противном случае ввести в регистр Х задуманное число и нажать С/П. На индикаторе число побед микрокалькулятора. Число побед игрока в регистре Y. Для продолжения игры перейти к п. 2.

Апостериорная вероятность появления единицы учитывается лишь тогда, когда правильно угадано предыдущее число.

Если в регистр 9 ввести целое число $1 \leq N \leq 9$, то микрокалькулятор изменит стратегию не сразу, а лишь после того, как число ошибок, сделанных подряд, станет равно N.

БИБЛИОТЕКА ПРОГРАММ

УНИВЕРСАЛЬНАЯ ПРОГРАММА

Универсальной будем называть программу, позволяющую решать однотипные задачи или одну и ту же задачу при вариациях входящих в нее известных величин.

Рассмотрим в качестве примера определение объема жидкости в резервуарах различной конфигурации (рис. 1.13).

В резервуар опускают мерную линейку и определяют уровень жидкости h.

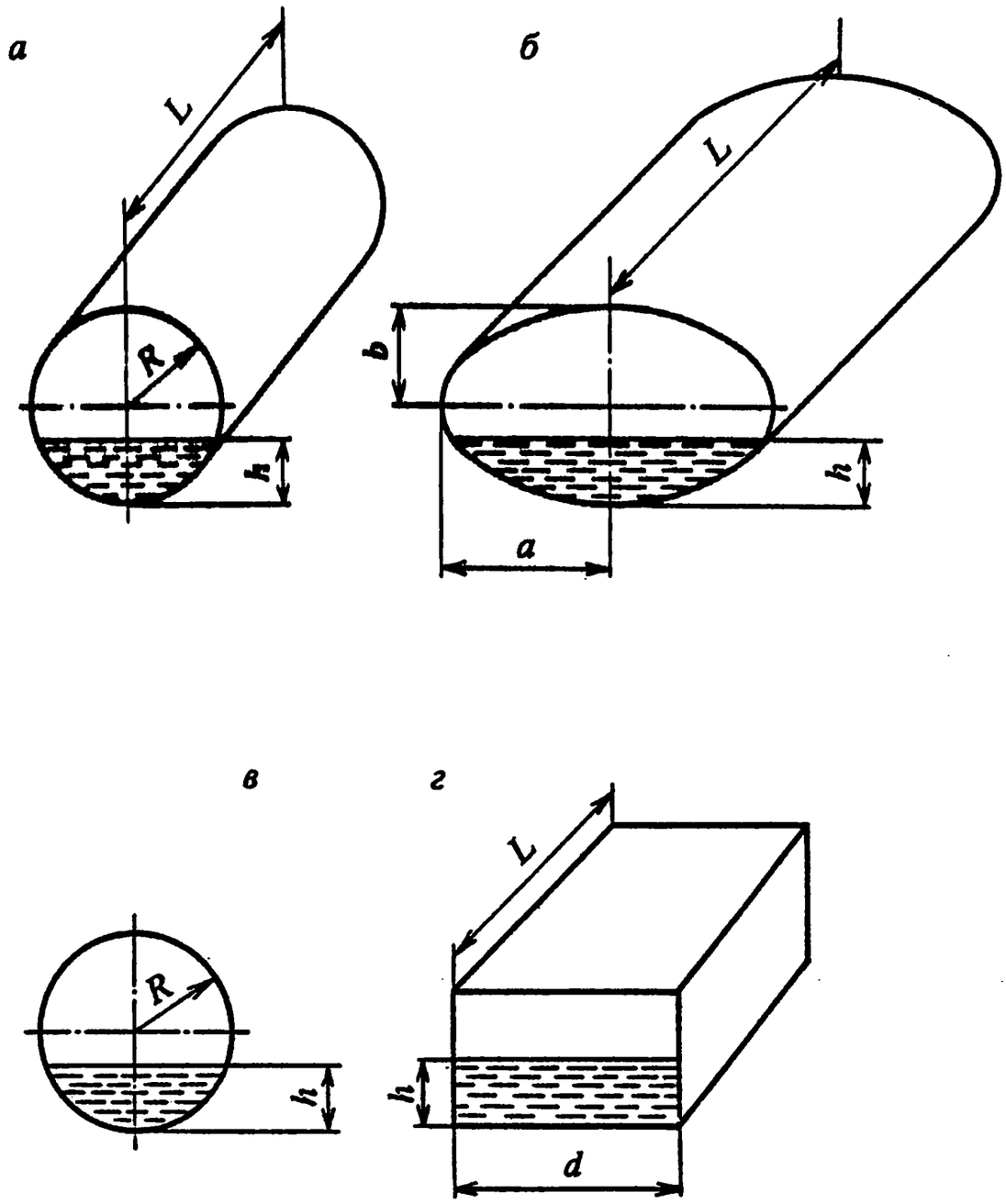


Рис. 1.13. Конфигурация резервуаров для хранения жидкости

а, б – цилиндр круглый и эллиптический; *в* – сфера; *г* – прямоугольный параллелепипед.

Переключатель Р-Г устанавливают в положение Р (радианы). Ввод исходных параметров зависит от конфигурации резервуара (программа 1.10):

цилиндр	
круглый	В/О L ПП R С/П ,
эллиптический	В/О L ПП a ПП b С/П ;
сфера	В/О СХ ПП R С/П ;
параллелепипед	В/О L ПП d ПП 0 С/П

Программа 1.10

Определение остатка жидкости в резервуаре

	00	01	02	03	04	05	06	07	08	09
00	ПС	ПА	ПВ	0	П1	П2	XY	С/П	ИП2	FX<>0
10	12	ИП1	+	П1	ИПС	FX<>0	45	ИПВ	FX<>0	60
20	ИП1	-	П3	ИПВ	/	FACS	1	FBX	FX^2	-
30	ИПА	FX^2	*	FSQR	ИП3	*	XY	ИПА	*	ИПВ
40	*	XY	-	БП	56	FPI	ИП1	FX^2	*	3
50	/	ИПА	3	*	ИП1	-	*	ИП1	БП	06
60	F→	ИПА	*	БП	56					

После того, как завершится предварительная обработка исходных данных, вводим значение уровня жидкости h и нажимаем клавишу С/П. Через некоторое время на индикаторе – объем жидкости в резервуаре (в кубических единицах используемых линейных параметров резервуара).

Эта программа позволяет также калибровать мерную линейку как с нулевого, так и с произвольного уровня h_0, определяемого уровнем осадка.

В режиме калибровки после ввода параметров и первого останова в регистр 2 заносят цену деления мерной линейки Δh, а в регистр 1 – значение h_0. Если оно не внесено, то программа по умолчанию считает, что $h_0 = 0$. Каждое нажатие клавиши С/П выводит на индикатор значение объема $V(h_i)$, где $i = 1, 2, 3, \ldots$. Полученные значения $V(h_i)$ наносят на мерную линейку или записывают в таблицу. Тогда объем жидкости в этом резервуаре можно определять непосредственно по показаниям мерной линейки.

Особенность данной программы в том, что выбор расчетного алгоритма микрокалькулятор проводит по исходным данным. Использование команды ПП, по которой программа выполняется шаг за шагом, – удобный прием для занесения данных в различные регистры. В этом случае пользователь только вводит число с клавиатуры, а в какой регистр его поместить, определяет программа.

Универсальной является также программа Е. Журавлева (программа 1.11). Она предназначена для определения любого из шести членов уравнения Менделеева – Клайперона

$$pV = \frac{m}{\mu}RT,$$

где p – давление, V – объем, m – масса газа, R – универсальная газовая постоянная, T – абсолютная температура, μ – молекулярная масса газа.

Программа 1.11

Решение уравнения Менделеева – Клайперона
(вариант Е. Журавлева)

	00	01	02	03	04	05	06	07	08	09
00	П1	П2	П3	П4	П5	П6	*	*	П7	ИП1
10	ВП	ИП2	ВП	*	ИП3	ВП	*	ИП4	ВП	ИП5
20	ВП	*	ИП6	ВП	*	ИП7	FX<>0	31	+	/
30	С/П	F→	XY	БП	29					

Перед началом вычислений в регистры памяти записывают константы: 11 – в регистр А, 39 – в регистр В и 45 – в регистр С. Исходные данные вводятся в строго определенном порядке:

В/О p ПП V ПП μ ПП m ПП T ПП R С/П ,

причем вместо неизвестной величины набирают нуль. Через несколько секунд после пуска программы вычисленное значение этой величины появляется на индикаторе. Каждый раз, решая задачу, полностью повторяют ввод данных.

Отметим, что в программе 1.11 использован уже знакомый нам прием превращения нуля в единицу командой ВП. Это позволило сделать программу достаточно короткой.

Однако справедливо утверждение, что нет программы, которую нельзя уменьшить хотя бы на один шаг. После публикации программы Е. Журавлева в журнале "Наука и жизнь" киевлянин Е. Рего прислал в редакцию свой вариант (программа 1.12), в котором использованы циклы с параметром и косвенная адресация записи в память. В результате программа Е. Рего на 12 шагов короче исходной.

Расчет по программе 1.12 ведут так: нажимают клавиши В/О С/П и после появления на индикаторе цифры 1 вводят исходные данные в следующем порядке:

В/О p С/П V С/П μ С/П m С/П T С/П R С/П

П р о г р а м м а 1.12

Решение уравнения Менделеева – Клайперона
(вариант К. Рего)

	00	01	02	03	04	05	06	07	08	09
00	2	П3	3	П0	1	П4	С/П	FX<>0	19	*
10	FL0	06	КП4	FL3	02	ИП2	ИП1	/	С/П	П4
20	F→	БП	10							

Как и в программе Е. Журавлева, вместо вычисляемой величины, вводят нуль. Через 5 сек после начала расчета на индикаторе появляется результат.

Во многих универсальных программах применяют кодирование цифрами, когда цифра является либо адресом регистра, где хранится адрес перехода, либо адресом перехода. Этот прием использован, например, в программе вычисления функций комплексного переменного $z = A + jB$ (программа 1.13).

Программа состоит из основного блока (адреса 00-64) и двух дополнительных (адреса 65-97). С помощью основного блока определяются sin$z$, cos$z$, sh$z$ и ch$z$ (здесь и далее комплексный аргумент $z = A + jB$), а дополнения служат для вычисления функций e^z, ln$z$, z^2, $1/z$, а также tg$z$ и th$z$.

После того, как программа набрана и микрокалькулятор переведен в режим вычислений, переключатель Р-Г устанавливают в положение Р. В регистры памяти записывают постоянные: 4, 9, 65, 14, 19 – соответственно в регистры 2-6; 6 – в регистр 9; 24, 39, 54 – соответственно в регистры А-С.

Функции, которые вычисляются в основном блоке, обозначим цифрами: sin$z$ – 2, cos$z$ – 3, sh$z$ – 5, ch$z$ – 6.

С помощью первого дополнительного блока вычисляются функции tg$z$ и th$z$. В этом случае в регистр 7 записывают число 67, в регистр 8 – 72, остальные константы не меняются, tg$z$ присваивают обозначение 4, а th$z$ – 7.

Второй дополнительный блок позволяет вычислять функции e^z, ln$z$, z^2. При этом изменяют содержимое регистров 7 и 8: 75 – записывают в регистр 7, а 87 – в регистр 8. Функциям присваивают следующие обозначения: e^z – 4, ln$z$ – 7, z^2 – 8.

Чтобы воспользоваться этой программой, необходимо: A записать в регистр 0, B – в регистр 1, кодовое число – в регистр X, а затем нажать клавишу С/П (в первый раз – В/О С/П). Результат получается в виде $f(z) = R(z) + I(z)$, при этом действительная часть $R(z)$ находится в регистре X, а мнимая $I(z)$ – в регистре Y. Время вычисления каждой функции менее 20 сек.

С помощью второго дополнительного блока комплексное число можно перевести в показательную форму: $f(z) = Me^{j\varphi}$. Для

этого, вычислив lnz, необходимо нажать клавиши F и e^x. В регистре X окажется M, а φ будет в регистре Y. Операцию нетрудно автоматизировать, если воспользоваться микрокалькулятором МК-61 и соответствующим вариантом программы. Тогда операции перевода присваивают кодовое число 4, в регистр E записывают 79, а в регистр 8 – 93. Содержимое остальных регистров изменять не требуется.

Программа 1.13

Вычисление функций комплексного числа $Z = A + jB$

Основной блок

	00	01	02	03	04	05	06	07	08	09
00	ПD	КИПD	ПD	КБПD	ИП0	КППА	С/П	БП	00	ИП0
10	КППВ	/-/	XY	КБП9	ПП	59	КППА	XY	КБП9	ПП
20	59	КППВ	XY	КБП9	FCOS	ИП1	FEXP	F1/X	/-/	КППС
30	*	ИП0	FSIN	ИП1	FEXP	F1/X	КППС	*	В/О	FCOS
40	ИП1	FEXP	F1/X	КППС	*	ИП0	FSIN	ИП1	FEXP	F1/X
50	/ /	КППС	*	В/О	FBX	+	2	/	В/О	ИП1
60	ИП0	П1	XY	П0	В/О					

Первый дополнительный блок

60						КПП8	КБП9	ПП	59	КПП8
70	XY	КБП9	ИП1	2	*	ПD	FEXP	F1/X	/-/	КППС
80	ИПD	FEXP	F1/X	КППС	ИП0	2	*	FCOS	+	ПD
90	/	ИП0	2	*	FSIN	ИПD	/	В/О		

Второй дополнительный блок

60						ИП1	FSIN	ИП0	FEXP	*
70	FBX	ИП1	FCOS	*	КБП9	ИП1	ИП0	/	FATN	ИП0
80	FX^2	ИП1	FX^2	+	FSQR	FLN	КБП9	ИП1	ИП0	2
90	*	*	ИП0	FX^2	ИП1	FX^2	-	КБП9		

Изменение во втором дополнительном блоке для МК-61
(вводятся с адреса 75)

70						ПП	82	FLN	КБП9	ПП
80	82	КБП9	ИП1	ИП0	/	FATN	ИП0	FX^2	ИП1	FX^2
90	+	FSQR	В/О	ИП1	ИП0	2	*	*	ИП0	FX^2
-0	ИП1	FX^2	-	КБП9						

Для вычисления функции 1/Z на МК-61
программа изменяется, начиная с 95 адреса

90						FX^2	ИП1	FX^2	+	/
-0	/-/	ИП0	FBX	/	КБП9					

Для определения функции z^2 кодовым числом служит 8, содержимое регистров памяти оставляют прежним.

После ввода программы рекомендуем проверить правильность ее работы, воспользовавшись контрольными примерами для числа $z = 3 + j2$:

Функция	Код	$R(z)$	$I(z)$
sin$z$	2	5.3092084 −02	−3.5905644
cos$z$	3	−3.7245453	−5.1182233 −01
tg$z$	4	−9.8843723 −03	9.6538587 −01
$1/z$	8	2.3076923 −01	−1.5384615 −01
e^z	4	−8.3585327	18.263726
ln$z$	7	1.2824747	5.8800261 −01
z^2	8	5	12
sh$z$*	5	−4.168907	9.1544984
ch$z$	6	−4.1896257	9.1092272
th$z$	7	1.0032386	−3.7640263 −03

При вычислении гиперболических функций происходит обмен содержимым между регистрами 0 и 1.

Воспользовавшись программой 1.12 для перевода числа $z = 3 + j2$ в показательную форму $f(z) = Me^{j\varphi}$, получаем M = 3,6055512 и φ = 0,58800261.

АЗИМУТ И ДАЛЬНОСТЬ

При решении астрономических и геодезических задач, расчетах радиотрасс и в некоторых других случаях необходимо определять азимут (A) и расстояние (D) между двумя точками, расположенными на сфере. Это делается по известным формулам:

$$A = \operatorname{arctg} \frac{\sin(\lambda_2 - \lambda_1)}{\cos\varphi_1 \operatorname{tg}\varphi_2 - \sin\varphi_1 \cos(\lambda_2 - \lambda_1)},$$

$$D = K \arccos\,[\sin\varphi_1 \sin\varphi_2 + \cos\varphi_1 \cos\varphi_2 \cos(\lambda_2 - \lambda_1)],$$

где K – коэффициент пропорциональности, φ и λ – координаты точек. Если они расположены на поверхности Земли, то φ_1, λ_1 – широта и долгота первой точки, а φ_2 и λ_2 – второй. Астрономы выражают D в часах, при этом K = 15. Геодезисты измеряют D в километрах, тогда K = 111,13 км. В математике при решении задач сферической тригонометрии расстояние D определяется в угловых мерах и K = 1.

Для определения A и D с помощью микрокалькулятора "Электроника Б3-34" используется программа 1.14. Величины φ и λ заданы в градусах, минутах и секундах, и в таких же единицах получается значение дуги или угла A.

Программа 1.14

"Азимут и дальность-1" (для Б3-34)

	00	01	02	03	04	05	06	07	08	09
00	КПП8	С/П	ИП6	F1/X	П6	ИП4	ИП2	-	FSIN	FBX
10	FCOS	ПВ	ИП1	FCOS	ИП3	FCOS	*	*	ИП1	FSIN
20	ИП3	FSIN	*	+	FACS	ИП5	*	ПD	F→	ИП1
30	FCOS	ИП3	FTAN	*	ИП1	FSIN	ИПВ	*	-	ПВ
40	XY	КПП7	XY	ИПВ	FX=0	49	ВП	9	/-/	:
50	FATN	КПП7	+	+	3	FEXT	+	КППА	0	.
60	6	П6	F→	С/П	FX<0	68	ИП0	В/О	0	В/О
70	2	FEXT	*	КПП9	:	КПП9	:	FBX	*	В/О
80	ПВ	КИПВ	F→	ИПВ	-	ИП6	:	ИПВ	+	2
90	FEXT	В/О	КПП9	*	КПП9	:	В/О			

Перед расчетами по программе переключатель Р-Г необходимо установить в положение Г (результаты в градусах) и записать постоянные 180, 0,6, 64, 67, 75, $1 \cdot 10^{-7}$, K соответственно в регистры 0, 6-9, А, С. Переменные размещаются в памяти следующим образом: в регистре 1 – φ_1, в регистре 2 – λ_1, в регистре 3 – φ_2, в регистре 4 – λ_2.

Чтобы минуты и секунды преобразовать в десятичные доли градуса, необходимо ввести абсолютное значение координаты в регистр Х и нажать клавиши В/О С/П. После окончания расчета на индикаторе будет координата, в которой минуты и секунды преобразованы в десятичные доли градуса. Так поступают со всеми координатами, прежде чем поместить их в соответствующие регистры памяти.

Для работы с координатой, абсолютное значение которой менее одного градуса, необходимо перед пуском программы прибавить к ней единицу, а после останова вычесть единицу и лишь тогда занести результат в память. Если координата имеет отрицательное значение, то преобразуют ее модуль, а перед тем, как занести ее в память, нажимают клавишу /-/. Когда все координаты введены, остается нажать клавиши БП 02 С/П .

После завершения расчета в регистре В записано значение дальности, а в регистре Х – азимут.

Формат числа, находящегося в регистре Х, предложен Э. Балуницем. Ему же принадлежит подпрограмма преобразования

значений величин в угловых единицах. Чтобы использовать ее для прямого и обратного преобразования значений величин в угловых единицах, необходимо изменить содержимое регистра 6 в процессе выполнения основной программы.

Поясним структуру программы 1.14:

00-01 и 64-74: преобразование минут и секунд в десятичные доли градуса;

02-04 и 58-61: преобразование и восстановление содержимого регистра 6;

05-11: подготовка данных для вычислений по формулам (1) и (2) (следует отметить, что значение $\sin(\lambda_2 - \lambda_1)$ удерживается в стеке начиная с адреса 08 и используется для вычислений по адресу 42; такой прием позволяет сэкономить один регистр памяти и две команды);

09-27: определение D;

28-50: вычисление азимута (с использованием подпрограммы 87-92, которая в зависимости от значения арктангенса и знака знаменателя в формуле (1) прибавляет 0, 180 или 360°;

44, 45 и 93-96: замена нуля на $1 \cdot 10^{-7}$ (позволяет избежать останова при расчете A);

57-63: преобразование вычисленного значения азимута в исходный формат (с использованием подпрограммы 75-86).

Для расчетов на микрокалькуляторах "Электроника МК-52 и -61" программу 1.14 легко упростить (программа 1.15). Этому способствуют встроенные микропрограммы для преобразования градусных величин и соответствующие им команды $\overleftarrow{\text{КО}'''}$ и $\overrightarrow{\text{КО}'''}$.

П р о г р а м м а 1.15

"Азимут и дальность-2" (для МК-61)

	00	01	02	03	04	05	06	07	08	09
00	ИП4	КМ→	ИП2	КМ→	-	FSIN	FBX	FCOS	П5	ИП3
10	КМ→	П6	FCOS	ИП1	КМ→	П7	FCOS	*	*	ИП6
20	FSIN	ИП7	FSIN	*	+	FACS	ИП0	*	ПD	XY
30	ИП7	FCOS	ИП6	FTAN	*	ИП7	FSIN	ИП5	*	-
40	П5	XY	ПП	60	XY	ИП5	FX=0	51	ВП	9
50	/-/	:	FATN	ПП	60	+	+	КМ←	ПА	С/П
60	FX<0	66	1	8	0	В/О	0	В/О		

Порядок вычислений в новой программе полностью сохраняется. Легко заметить, что программа стала короче, освободились регистры памяти.

Генератор событий с равномерным распределением легко построить в соответствии с алгоритмом, знакомым нам по игре "чет-нечет".

Существуют и другие алгоритмы генерирования псевдослучайных чисел. Например, в качестве рекуррентной воспользуемся формулой

$$\xi_{i+1} = \{11\xi_i + \pi\},$$

где число ξ_{i+1} – дробная часть выражения в фигурных скобках. При начальном $\xi_0 = 0{,}5$, программа, построенная по этому алгоритму, выдает около восьми тысяч неповторяющихся чисел, равномерно распределенных вокруг 0,5.

Такой генератор легко превратить в генератор событий с равномерным распределением: из ξ_{i+1} вычесть 0,5 и, если число отрицательное, приписать ему значение 0, а если положительное, то – 1. На практике результат получают, умножая ξ_i на 2 и выделяя из него целую часть.

Генератор событий 0 или 1 с заданной вероятностью можно создать, сравнивая псевдослучайное число, генерируемое в интервале 0 – 1, с требуемой вероятностью. Если обозначить вероятность появления единицы через $P(1)$, то при выполнении условия $P(1) - \xi_{i+1} \geq 0$, генератор должен выдавать 1, а при невыполнении – 0.

Иногда на такие генераторы накладывают более жесткие условия. Например, есть генератор, алгоритм работы которого определяется формулой

$$\xi_{i+1} = 9 \sin\xi_i - 9 \cos [\pi P(1)/2] - \xi_i + 9,$$

где $P(1)$ – по-прежнему вероятность появления 1. Если ξ_i – положительная величина, то индицируется 1, а если отрицательная, то – 0.

При использовании генераторов псевдослучайных чисел возникает вопрос: в какой мере получаемые числа близки к последовательности случайных чисел?

Существует немало способов проверки. Достаточно простым является метод, когда иглу бросают в квадрат. Вершина одного из углов квадрата при этом является центром сегмента вписанного круга, радиус которого равен стороне квадрата. Из теории вероятности известно, что при большом числе попыток число метких бросков n (попаданий в сегмент) связано с числом попыток N соотношением

$$4n/N \simeq \pi.$$

При большом числе попыток эту зависимость можно использовать, чтобы охарактеризовать генератор псевдослучайных чисел. Разность между вычисленным по программе значением $4n/N$ и числом π назовем ошибкой генератора ε.

П р о г р а м м а 1.16

Проверка генераторов псевдослучайных чисел по критерию π

	00	01	02	03	04	05	06	07	08	09
00	ПО	П1	СХ	П4	ПП	27	FX^2	П2	ПП	27
10	FX^2	ИП2	+	FSQR	1	-	FX<0	19	КИП4	FLO
20	04	4	ИП4	*	ИП1	/	С/П			

Подпрограмма для оценки встроенного генератора МК-61

20								КСЧ	В/О	

Подпрограмма для оценки генератора № 2

20								FPI	1	1
30	ИПD	*	+	K{X}	ПD	В/О				

Подпрограмма для оценки генератора № 3

20								FPI	ИПD	+
30	FEXP	K{X}	ПD	В/О						

Подпрограмма для оценки генератора № 4

20								FPI	2	-
30	FEXT	K{X}	ПD	В/О						

Подпрограмма для оценки генератора № 5

20								FPI	2	-
30	ИПD	+	FX^2	K{X}	ПD	В/О				

Подпрограмма для оценки генератора № 6

20								ИПD	ВП	9
30	FCOS	FACS	FPI	/	ПD	В/О				

Подпрограмма для оценки генератора № 7

20								ИПD	3	7
30	*	1	+	K{X}	ПD	В/О				

Будем считать, что два случайных числа, созданные генератором одно за другим, есть координаты нашего броска x и y. Радиус круга примем равным 1. Договоримся также, что попытка удачна, если $x^2 + y^2 \leq 1$.

Программа 1.16 позволяет провести численный эксперимент (табл. 1.8).

Таблица 1.8

Результаты исследования генераторов псевдослучайных чисел

№ п/п	Алгоритмы генерации	ξ_0	N	$4n/N$	$(4n/N-\pi)/\pi$
1	Встроенный в МК-61, -52 (расчетная формула неизвестна)		250 1000	3,024 3,088	3,74 −02 1,71 −02
2	$\xi_{i+1}=\{11\xi_i+\pi\}$	0,5	250 1000	3,232 3,116	2,88 −02 8,15 −02
3	$\xi_{i+1}=\{\exp(\xi_i+\pi)\}$	0	250 1000	3,26 3,368	3,77 −02 7,21 −02
4	$\xi_{i+1}=\{\mathrm{ext}(\xi_i+\pi)\}$	1/3	250 1000	2,928 3,096	6,80 −02 1,45 −02
5	$\xi_{i+1}=\{(\pi-2+\xi_i)^2\}$	0	250 1000	3,28 3,156	4,41 −02 4,60 −03
6	$\xi_{i+1}=\frac{1}{\pi}\arccos[\cos(\xi_i\cdot 10^9)]$	1	250 1000	3,168 3,148	8,41 −03 2,04 −03
7	$\xi_{i+1}=\{37\xi_i+1\}$	0,12345	250 1000	3,232 3,228	2,88 −02 2,75 −02

После ввода основной программы (шаги 00-26) начиная с адреса 27 записывают подпрограмму-генератор псевдослучайных чисел. Она может быть достаточно длинной (лишь бы поместилась в программную память).

Работают с программой 1.16 так: после записи основной программы и подпрограммы испытуемого генератора определяют стартовое число псевдослучайной последовательности ξ_0 и вводят его в регистр D. Общее число попыток N записывают в регистр X и нажимают клавиши В/О С/П.

Следует помнить, что время счета определяется значением N, поэтому для больших N результат будет получен нескоро. Каждый вариант из тех, что приведены в табл. 1.8, для $N = 1000$ микрокалькулятор рассчитывал за 30 – 40 мин.

Стартовое число определяет всю последовательность. Например, у генератора 2 при ξ_0 = 0,5 генерируемая последовательность выглядит так: 0,6415926, 0,199111, 0,3318136 и остальные числа (и порядок их следования) однозначно определены значением ξ_0. Это удобно, когда идет отладка программы, и неудобно, если требуется отыскать неизвестный заранее результат. В таких случаях заменяют стартовое число. Тот же генератор, но с другим стартовым числом будет генерировать уже иную последовательность псевдослучайных чисел. При ξ_0 = 0 получим: 0,1415926, 0,6991112, 0,831816.

Из табл. 1.8 видно, что лучшие результаты у генератора 6.

Генераторы псевдослучайных чисел, используемые в микрокалькуляторах, в большинстве случаев работают удовлетворительно, но за их простоту приходится расплачиваться ограничениями при выборе стартовых чисел. Особенно осторожным следует быть, когда в погоне за сокращением программы применяют простые мультипликативные способы. Генератор 7 (см. табл. 1.8) при вводе некоторых стартовых чисел (например, 0,25 или 0,5), зацикливается. При стартовом числе 0,013625 генерируется последовательность, период которой составляет 18 чисел, после чего она также начинает повторяться.

При желании читатель сам найдет другие числа, ведущие к короткому периоду или зацикливанию.

Говоря о генераторах псевдослучайных чисел, всегда указывают при каком стартовом числе ξ_0 проходили испытания. Например, генератор 2 при ξ_0 = 0,5 имеет период более 8000 чисел, генератор 6 при ξ_0 = 0 дает около 4500 неповторяющихся чисел.

ВЕЧНЫЙ КАЛЕНДАРЬ

Наверное, нет программиста, который в той или иной степени не отдал бы дань решению таких задач: определить день недели любой произвольной даты или число дней между двумя произвольными датами.

В самом деле, разве не интересно узнать, например, какой день недели был 5 апреля 1242 г., когда произошло Ледовое побоище, в котором князь Александр Невский со своей дружиной разгромил тевтонских псов-рыцарей. А в какой день недели была Полтавская битва? Как рассчитать, сколько дней прожил человек?...

Вопросы, вопросы и вопросы. Ответить на все невозможно.

Рассмотрим один из вариантов календаря для микрокалькуляторов "Электроника МК-52 и -61". Программа 1.17 "Вечный календарь-1" позволяет определить день недели для любой даты.

П р о г р а м м а 1.17

"Вечный календарь-1"

	00	01	02	03	04	05	06	07	08	09
00	ИП3	П8	ИП2	ПП	38	ИП1	+	П9	ИП6	П8
10	ИП5	ПП	38	ИП4	+	ИП9	-	С/П	БП	00
20	ИП3	П8	ИП2	ПП	38	ИП1	+	↑	↑	7
30	/	К[X]	7	*	-	С/П	БП	20	1	+
40	П7	4	-	FX<0	53	1	6	+	П7	XY
50	1	-	П8	ИП8	ИПЕ	*	К[X]	ИП7	ИПD	*
60	К[X]	+	4	+	ПА	ИП0	FX<>0	92	XY	ИП8
70	2	FEXT	/	К[X]	3	/-/	*	4	/	9
80	+	5	FBX	/	-	7	FEXT	+	FBX	-
90	+	ПА	ИПА	В/О						

Алгоритм (рис. 1.14) не содержит никаких ограничений в положении даты на оси времени. Некоторые ограничения связаны с вычислительными возможностями микрокалькулятора. Пользователь должен указать, по какому стилю определяется дата – старому или новому.

При подсчете числа дней между двумя датами находят значение *P* для большей даты, затем для меньшей и вычисляют их разность.

Перед началом вычислений в регистры памяти вводят постоянные: 365,25 – в регистр E; 30,6 – в регистр D. Эти величины не изменяются при любых вариантах исходных данных.

Если определяется дата по старому стилю, то в регистр 0 вводится 0, а если по новому стилю, то – 1. Чтобы найти день недели, порядковые номера дня, месяца и года вводят соответственно в регистры 1, 2 и 3 и нажимают клавиши БП 20 С/П .

Для определения затем дня недели другой даты после ввода исходных данных достаточно нажать клавишу С/П.

При подсчете числа дней между двумя датами, они должны относиться к одному стилю. Меньшую дату вводят так же, как и при расчете числа дней недели, а бóльшую записывают в регистры 4 (день), 5 (месяц) и 6 (год), после чего нажимают клавишу В/О С/П.

Если затем вычисляется число дней между двумя другими датами, то после ввода исходных данных достаточно нажать клавишу С/П.

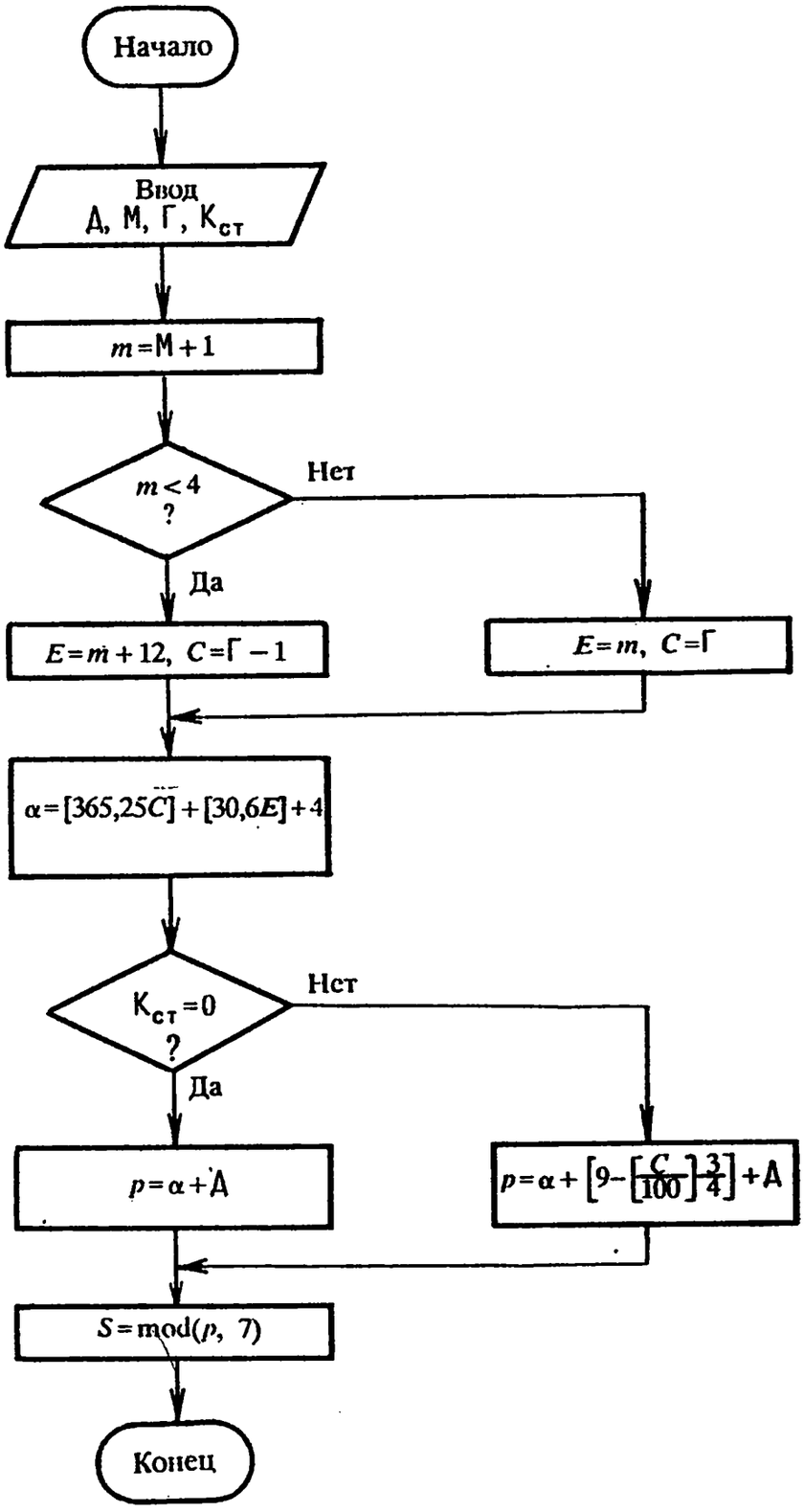


Рис. 1.14. Алгоритм "Вечный календарь-1"

Д – день; М – месяц; Г – год; $К_{ст}$ – коэффициент стиля летоисчисления; p – переменная

И, конечно, каждый раз при переходе от одного стиля к другому необходимо записывать в нулевой регистр число 0 или 1, определяющее, по какому стилю ведется расчет.

В программе 1.17 "Вечный календарь-1" впервые использована операция выделения целой части числа, имеющаяся в микрокалькуляторах МК-52 и -61.

В качестве теста можно воспользоваться одной из контрольных задач.

Контрольные задачи

1. *Известно, что битва на Калке произошла 31 мая 1223 г. В какой день недели это было?*

В регистр 1 вводим 31, в регистр 2 – 5, в регистр 3 – 1223, в регистр 0 – 0 (стиль старый):

```
БП 20 С/П "1"
```

Индикация "1" – понедельник. Время решения 13 с.

2. *Определить разность между датами 28 февраля 1900 г. и 1 марта 2100 г. по новому стилю.*

Записываем:

```
28 П1 2 П2 1900 П3 1 П4 3 П5 2100 П6 1 П0 В/О С/П "73052"
```

Время решения 35 с.

3. *Решить задачу 2, если даты даны по старому стилю.*

Поскольку цифры не менялись, вводим лишь коэффициент старого стиля 0 в регистр 0. Расчет ведется повторно, поэтому нажимаем только С/П.

Индикация "73052". Время решения 21 с.

К сожалению, если в программе для микрокалькуляторов МК-61 использованы "синие" команды (команды, вводимые клавишей К), то ее нельзя использовать в Б3-34. Если же "синих" команд нет, то программа годится и для Б3-34.

В нашем "календаре" есть подобные команды. Однако можно создать программу и для Б3-34. Во-первых, пожертвуем старым стилем, а во-вторых, ограничим диапазон нашего календаря.

Условимся, что все даты, с которыми собираемся работать, ограничены интервалом 1 марта 1900 г. – 28 февраля 2100 г. Может быть, этот диапазон не удовлетворит изыскательных пользователей, зато они смогут решать известную задачу расчета биоритмов: физического (период 23 дня), эмоционального (период 28 дней) и интеллектуального (период 33 дня).

Вычисление биоритмов основано на построении синусоиды, начало которой совпадает с днем рождения человека. Для каждой даты определяется мгновенное значение синусоиды, амплитуда которой равна единице.

Итак, чтобы воспользоваться программой 1.18 "Вечный календарь-2" для микрокалькулятора Б3-34, необходимо до начала работы ввести в регистры памяти постоянные: 30,6 – в регистр C и 365,25 – в регистр D.

При определении дня недели и числа дней между двумя датами данные вводят в том же порядке, что и в программе для микрокалькулятора МК-61 (см. программу 1.17).

П р о г р а м м а 1.18

"Вечный календарь-2"

	00	01	02	03	04	05	06	07	08	09
00	ИП3	П8	ИП2	П7	ПП	55	ИП1	+	П0	ИП6
10	П8	ИП5	П7	ПП	55	ИП4	+	П7	ИП0	-
20	П0	ИП7	5	+	П7	7	/	П8	КИП8	ИП7
30	ИП8	7	*	-	П7	3	3	ПП	84	П8
40	2	8	ПП	84	П9	2	3	ПП	84	ПА
50	ИП8	ИП9	+	+	С/П	3	-	FX<0	68	XY
60	1	-	П8	ИП7	1	2	+	П7	ИП7	1
70	+	ИПС	*	П7	ИП8	ИПD	*	П8	КИП7	КИП8
80	ИП8	ИП7	+	В/О	ИП0	XY	/	ПА	КИПА	XY
90	ИПА	-	3	6	0	*	FSIN	В/О		

Рассчитывая биоритмы, день, месяц и год рождения вводят в регистры 1, 2, 3, а день, месяц и год, для которых определяют биоритмы, – в регистры 4, 5, 6. Во всех случаях вычисления запускают командами В/О С/П. После останова на индикаторе – сумма значений трех биоритмов на день расчета. В регистре 8 значение интеллектуального биоритма, в регистре 9 – эмоционального, в регистре A – физического, в регистре 7 день недели (0 – воскресенье, 1 – понедельник, 2 – вторник, 3 – среда, 4 – четверг, 5 – пятница, 6 – суббота), в регистре 0 – число дней между двумя датами.

ПОМОЖЕТ СТРУКТУРНАЯ ДИАГРАММА

Мы уже отмечали, что структурная диаграмма – это хорошее подспорье программиста, позволяющее ему без труда быть понятым другими или легко разбираться в чужих программах. В таком виде несложно записать и программу для микрокалькулятора, чтобы была видна ее структура. Но если для больших ЭВМ требование структурного программирования может быть доминирующим, то для микрокалькулятора, обладающего

скромными резервами памяти вряд ли необходимо вписывать все без исключения программы в структурные диаграммы. В то же время структурные диаграммы целесообразны там, где это сделать легко.

Наверное, каждому хотя бы раз приходилось расписывать программы по регистрам стека, чтобы понять чужой замысел или вспомнить собственный, скрытый за вязью команд переходов. Диаграммы позволят на практике ознакомиться с принципами структурного программирования, а также облегчат работу с составленными вами программами не только другим, но и вам, если вы решите к ним вернуться через некоторое время.

В структурном программировании возможны такие конструкции:

следование (программа выполняется последовательно шаг за шагом);

альтернатива (выбор) (в микрокалькуляторах реализованы лишь частные случаи альтернативы – условные переходы $x < 0$, $x \geq 0$, $x = 0$ и $x \neq 0$);

циклы "пока", "до" (создаются в микрокалькуляторах с помощью операторов условных переходов);

цикл с параметром (известно, сколько раз его следует выполнить с помощью команды FLN, где N = 0, 1, 2, 3 – номер регистра, в котором записано число повторений цикла).

Попытаемся вписать в структурную диаграмму какую-нибудь программу.

Для примера возьмем еще один вариант игры "Чет-нечет". Пусть микрокалькулятор отгадывает задуманные человеком числа 0 или 1. Если задуман 0, а на индикаторе 1, то электронный игрок штрафуется на три очка. Если была задумана 1, а показан 0, то штраф в одно очко. Если же число угадано, то микрокалькулятор получает два премиальных очка. Ясно, что штрафы микрокалькулятора – это премия игрока, а премия микрокалькулятора – штраф игрока. На индикаторе отражается цена последней партии для игрока, а в регистре 0 – общий результат всех сыгранных партий.

Эти условия легко реализуются (программа 1.19). Составим схему алгоритма (рис. 1.15). Заметим, что игра является циклом, в котором есть две альтернативы.

На индикатор микрокалькулятора нельзя вывести текстовые комментарии, как это делается в ЭВМ, поэтому даже вписанная в структурную схему программа нуждается в инструкции и дополнительном описании. В этом плане иногда более наглядными выглядят схемы алгоритмов, в особенности если на них, как это сделано на рис. 1.15, указано, какие шаги программы выполняют ту или иную операцию.

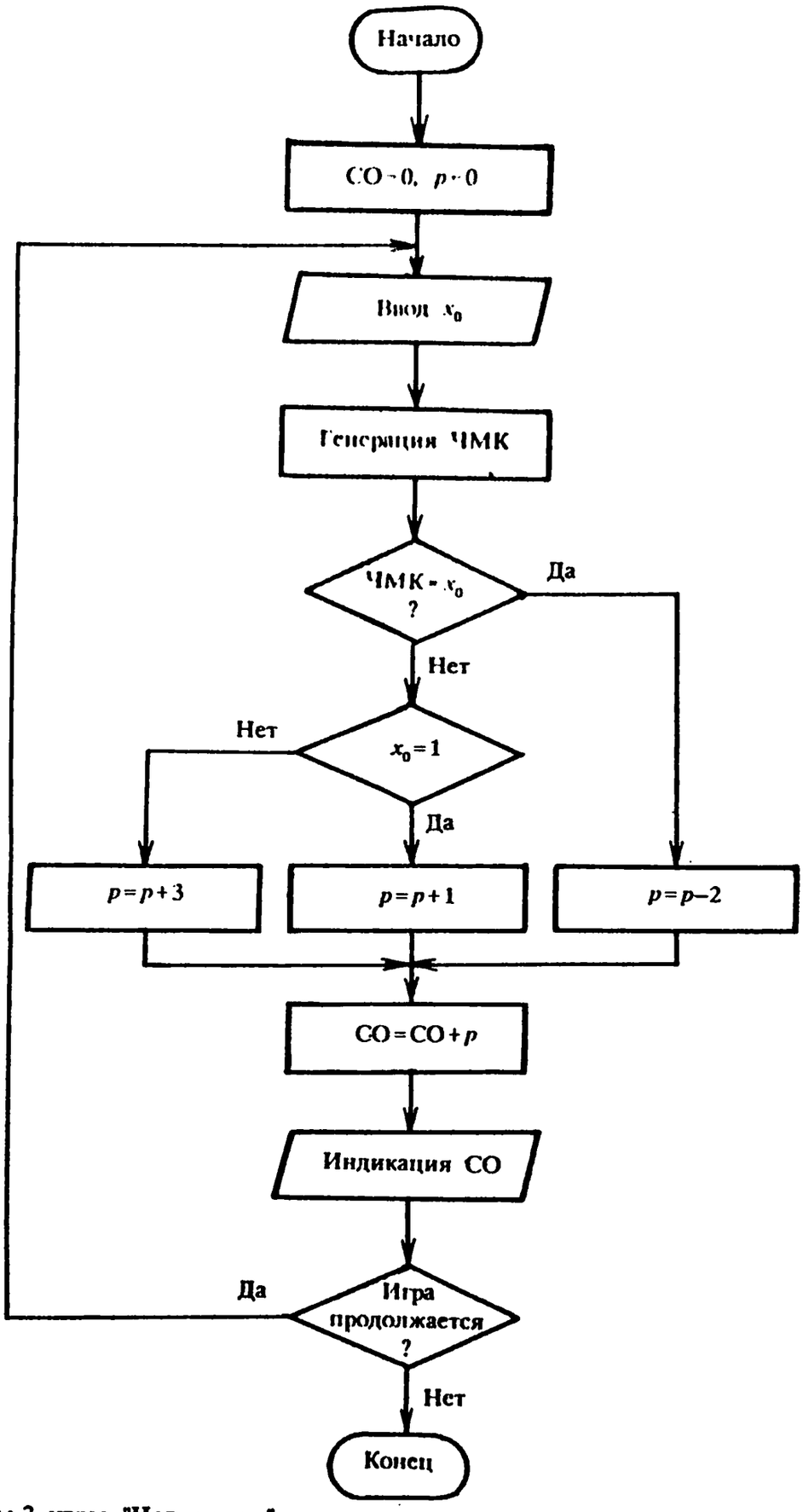


Рис. 1.15. Алгоритм-3 игры "Чет–нечет"

СО – счетчик очков; p – переменная

Программа 1.19

"Чет–нечет-3"

```
00.СХ   01.П0
02.ИПЕ  03.FПИ  04.+    05.FEXP 06.K{X} 07.ПЕ 08.2
09.*    10.K[X]
11.С/П  **ВВОД ИГРОКОМ ЗАДУМАННОГО ЧИСЛА**
12.-    13.FX=0 14.20
15.2    16./-/  17.П1
18.БП   19.28           20.FX<0 21.26
                        22.1    23.П1
                        24.БП   25.28   26.3 27.П1
28.ИП0  29.ИП1  30.+    31.П0   32.FBX  33.С/П
34.БП   35.02
```

Инструкция

1. Введите стартовое число генератора псевдослучайных чисел ξ_0 в регистр Е ($0 < \xi_0 < 1$).

2. Задумайте число. Чтобы запустить программу, нажмите клавишу С/П (в первый раз – клавиши В/О С/П).

3. После останова на индикаторе высвечивается предположение микрокалькулятора о загаданном числе. Введите число, задуманное игроком, в регистр Х и нажмите клавишу С/П (если загадан нуль, то следует нажать клавишу 0, так как клавишей СХ на этот раз воспользоваться нельзя).

4. После останова на индикаторе высвечивается цена данной партии для игрока (со знаком "плюс" – выигрыш, со знаком "минус" – проигрыш). При желании нетрудно узнать суммарный результат турнира: достаточно нажать клавишу XY.

5. Повторяйте пп. 2 – 4 до завершения турнира. Чтобы начать новую игру, нажмите клавишу В/О и перейдите к п. 2.

МИКРОКАЛЬКУЛЯТОР ИГРАЕТ В СПОРТЛОТО

Использование программируемого микрокалькулятора для заполнения карточек спортлото освобождает от долгих раздумий или расчетов. Рассмотрим два варианта программы "Лототрон" (первый – для Б3-34 (программа 1.20) и второй – для МК-61 (программа 1.21)), с помощью которых можно заполнять карточки спортлото "5 из 36" или "6 из 45".

Эти программы подойдут, если появится новое лото "n из N", при условии, что $n \leq 6$. Для МК-61 игру легко модернизировать для $n \leq 7$. На N никаких условий не накладывается, и в принципе его значение может быть любым.

П р о г р а м м а 1.20

"Лототрон-1"

	00	01	02	03	04	05	06	07	08	09
00	П1	XY	П2	ИП1	П3	СХ	П6	П7	6	П4
10	КИП6	ИП6	П0	6	П5	ИПD	FPI	+	FEXP	ПD
20	КИПD	XY	ИПD	-	↑	ИП2	*	1	+	ПD
30	КИПD	F→	F→	ИПD	XY	ПD	КИП5	FBX	-	FX<>0
40	11	FL0	36	FBX	КП4	FL3	10	С/П	БП	03

П р о г р а м м а 1.21

"Лототрон-2"

	00	01	02	03	04	05	06	07	08	09
00	П1	XY	П2	ИП1	П3	СХ	П6	П7	6	П4
10	КИП6	ИП6	П0	6	П5	ИПD	FPI	+	FEXP	K{X}
20	ПD	ИП2	*	K[X]	1	+	XY	КИП5	FBX	-
30	FX<>0	11	FL0	27	FBX	КП4	FL3	10	С/П	БП
40	03									

Перед началом расчетов в регистр D вводят стартовое число ξ_0 для генератора случайных чисел, с помощью которого формируются числа для заполнения карточек. Стартовое число должно быть положительным и меньше единицы. В регистры X и Y вводят параметры лотереи, для которой делается расчет: 5 ↑ 36 или 6 ↑ 45 .

Затем нажимают клавиши В/О С/П. После завершения расчета, продолжающегося одну-две минуты, результаты извлекают из регистров 7, 8, 9, A, B, C. В регистре C хранится результат только для лотереи "6 из 45". При заполнении карточек "5 из 36" им не пользуются. Чтобы получить другой вариант заполнения карточки этой же лотереи достаточно нажать на клавишу С/П.

В качестве самостоятельного упражнения читатели могут попытаться вписать программу "Лототрон" в структурную диаграмму.

СЛОЖНЫЕ ПРОГРАММЫ

Цикл, возможность ветвления, косвенные переходы позволяют создавать достаточно сложные программы, которые микрокалькулятор выполняет часами, а то и сутками. Стремясь доказать, что микрокалькуляторы могут многое, патриоты "карманной ЭВМ" создают подчас очень сложные программы.

Многим, наверное, знакома головоломка "Ханойская башня". Речь в ней идет о трех вертикальных стержнях, на левом из которых собрано 64 кольца в порядке уменьшения их диаметра, т.е. самое большое внизу. Задача в том, чтобы перенести все кольца на правый стержень и сложить их в том же порядке, в каком они были на левом. Нельзя переносить за один раз более одного кольца́ или класть бо́льшее кольцо на меньшее. Средний стержень вспомогательный.

Читатели, знакомые с рекурсивными операциями, догадались, что перекладывание колец в данном случае – рекурсивная операция. Это легко сделать, зная, что при рекурсии часть программы многократно повторяется, причем каждое повторение включает в себя всю предыдущую часть, дополненную новым фрагментом программы.

Такие операции есть в языках высокого уровня, но отсутствуют в кодово-символьном языке программируемого микрокалькулятора.

И все же и на микрокалькуляторе можно создать программу, реализующую рекурсивные операции.

Рекурсивные операции нередко удается выразить через другие. Например, харьковчанин А. Карабанов получает порядок перекладывания, превращая номер хода в двоичное число. Анализируя положения нулей и единиц, нетрудно определить, как переносить кольцо в очередном ходе.

Программа 1.22 "Ханойская башня-1" – пример вдумчивого подхода к задаче, когда исходное условие преобразовано к виду, удобному для реализации на микрокалькуляторе.

Программа 1.22

"Ханойская башня-1"

	00	01	02	03	04	05	06	07	08	09
00	СХ	ПЕ	1	3	ПО	П1	1	КПО	FL1	07
10	1	ПD	ИПЕ	1	+	ПЕ	2	/	↑	К{X}
20	FX=0	30	ИПD	1	+	ПD	F→	F→	БП	16
30	ИПD	2	+	К{X}	FX<>0	46	КИПD	ПО	1	-
40	FX=0	43	3	КПD	БП	57	КИПD	ПО	1	+
50	КПD	4	-	FX=0	57	1	КПD	ИПО	КИПD	1
60	0	/	+	С/П	БП	10				

Программа указывает порядок перекладывания кружков числом из двух цифр, разделенных запятой. Первая цифра говорит о том, с какого стержня нужно снять кольцо, а вторая на какой его переместить. Число перекладываемых колец равно 13.

Пользоваться программой несложно. После ввода нажать клавиши В/О С/П. На индикаторе появятся указания по переносу

первого кольца. После каждого нажатия клавиши С/П инструкции для игрока будут обновляться до тех пор, пока все 13 колец не будут перенесены со стержня 1 на стержень 3.

На микрокалькуляторе можно осуществить рекурсию в чистом виде, как это сделано, например, в программе 1.23 "Ханойская башня-2".

П р о г р а м м а 1.23

"Ханойская башня-2"

	00	01	02	03	04	05	06	07	08	09
00	КНОП	2	FX^Y	ПО	XY	2	/	К{X}	FX<>0	16
10	3	П3	2	П2	БП	20	3	П2	2	П3
20	1	П1	ПП	25	КППВ	ПП	28	КППА	ПП	31
30	КППВ	ПП	34	КППА	ИП1	ИП3	КППС	ИП1	ИП2	КППС
40	ИП3	ИП2	КППС	ИП1	ИП3	КППС	ИП2	ИП1	КППС	ИП2
50	ИП3	КППС	ИП1	ИП3	КППС	В/О	ИП1	ИП2	БП	62
60	ИП2	ИП1	КППС	ИП1	ИП2	ИП3	П1	F→	П3	F→
70	П2	В/О	1	0	/	+	С/П	КИПО	ИПО	FX=0
80	89	3	3	1	КИНВ	↑	ВП	2	С/П	В/О

В ней сначала заполняется адресный стек подпрограмм (шаги 22-32). Команда В/О, записанная по адресу 55, вызывает повторение шагов 33-55, затем 30-55, 27-55 и, наконец, 24-55. Программа позволяет перенести семь колец.

В регистры памяти вводят постоянные: 56 – в регистр А, 60 – в регистр В и 72 – в регистр С. В регистр X заносят число колец N ($0 < N < 7$) и нажимают клавиши В/О С/П. Примерно через 10 сек на индикаторе появится число, где первая цифра – номер стержня, с которого снимают кольцо, а вторая – номер стержня, куда переносят. Как и в программе "Ханойская башня 1" считается, что первоначально кольца собраны на стержне 1, а перенести их требуется на стержень 3. Все последующие ходы происходят по команде С/П. После того как последнее кольцо перенесено, на индикаторе появляется надпись "ВСЁ".

Программа пригодна и для микрокалькулятора Б3-34, если изменить шаги

```
00 ПП1  01 0  02.FCOS 03.FX<0  ,
04 04  05 1  06 П1/X  83.С/П  84.В/О
```

Через некоторое время "ВСЁ" на индикаторе появятся восемь троек. Легко установить, что нужно сделать $2^{64}-1$ перекладываний.

В обычной записи это число выглядит так:

18 446 744 073 709 531 615.

Если считать, что на одно перекладывание требуется не менее секунды, то для переноса всех колец потребуется пять с лишним миллиардов веков.

МИКРОКАЛЬКУЛЯТОР-КОМПЬЮТЕР

Наш рассказ о программируемом микрокалькуляторе подходит к концу. И настало время поделиться одним важным наблюдением. Освоившие программируемый микрокалькулятор и начинающие изучать языки высокого уровня с удивлением узнают, что кодово-символьный язык микрокалькулятора занимает промежуточное место между языком ассемблера и языками высокого уровня, в ряде случаев приближаясь к ним.

Многие команды языка микрокалькуляторов существуют в таком же или в более развитом виде в языках для ЭВМ. Запись в память – это операция присвоения, а команды FL0 – FL3 напоминают циклы с параметром.

С этими и другими аналогиями читатель может познакомиться, заглянув в Приложение 1, где приведены полный лексикон микрокалькуляторов и аналоги обозначения операций на языках Бейсик и Паскаль. И хотя словарь нашего электронного помощника не так уж богат, но и бедным его не назовешь.

Микрокалькулятор позволил нам освоить многие конструкции языков высокого уровня и облегчил переход к ним. С помощью нашего микровычислителя познакомимся с применяемыми в ЭВМ принципами трансляции. Идея интерпретации является фундаментальной для всех вычислительных машин.

Рассмотрим комбинацию цифр 1589. Ее можно интерпретировать как число тысяча пятьсот восемьдесят девять или как дату 1.5.89 – первое мая 89 г. Но она может и означать: взять содержимое регистров 1 и 5, выполнить операцию 8 и записать результат в регистр 9. Операция 8 – это или простая операция сложения, или сложная вычислительная программа.

Основная задача интерпретатора – уметь распознавать коды-команды или слова-команды. Для этого во всех вычислительных устройствах (в том числе и в нашем микрокалькуляторе) в постоянном запоминающем устройстве хранится программа-интерпретатор. Но мало распознать команду, необходимо сделать ее понятной для машины. Распознанную команду переводит на язык ЭВМ программа-транслятор. Иногда программы, выполня-

ющие эти две функции, называют одним словом – транслятор. Программа-транслятор создается в виде машинного кода.

Хотя вычислительные возможности микрокалькулятора невелики, но они позволяют создать простейшую модель транслятора.

Введем ограничение: микрокалькулятор работает с однозначными цифрами от 1 до 4, т. е. в условном языке цифрам 1, ..., 4 будут соответствовать числа, а цифрам от 5 до 8 операции. Пусть 5 – сложение, 6 – вычитание, 7 – умножение и 8 – деление. На этом условном языке 272 означает: определить, сколько будет дважды два, и показать результат.

Придуманный нами язык реализован в программе 1.24 "Транслятор".

Прежде, чем ознакомиться с алгоритмом, необходимо выполнить подготовительные действия.

Анализируемое число обозначим символом *N*. В этом числе не может быть более семи цифр. Числа обозначим символом Ч, а команды – К. Количество чисел на единицу больше количества команд.

П р о г р а м м а 1.24

"Транслятор"

	00	01	02	03	04	05	06	07	08	09
00	КНОП	7	FEXT	/	ПЕ	К[X]	FX<>0	10	ВП	.
10	П1	П2	П3	П5	П7	П8	П9	ПА	6	П6
20	2	3	ПС	ИПЕ	1	0	*	↑	К{X}	ПЕ
30	-	FX=0	37	ИПЕ	FX<>0	48	КБПС	4	XY	-
40	FX>0	45	FBX	КП6	КБПС	FBX	КП5	КБПС	ИП6	ИП5
50	7	+	-	FX=0	·0	П5	ПВ	6	П6	6
60	3	ПD	КИП6	КИП6	FX<>0	98	КИП5	8	-	FX<>0
70	86	1	+	FX<>0	89	1	+	FX<>0	92	1
80	+	FX<>0	95	ИПD	КИНВ	С/П	F→	/	КБПD	F→
90	*	КБПD	F→	-	КБПD	F→	+	КБПD	F→	С/П
-0	9	2	КИНВ	С/П						

Ошибочными условиями являются следующие ситуации: *N* содержит более семи цифр; применена команда, не внесенная в список команд; не выполняется соотношение команд и чисел.

Распределим регистровую память: регистры 1-3 для записи К; регистры 7-9 и А для записи Ч; регистры 5 и 6 для организации записи Ч и К в соответствующие регистры; регистр Е для записи *N*; регистры С и D для записи адресов, с которых начинаются циклы.

Переменная Ч может принимать значения от 1 до 4, а К – от 5 до 8.

Введем в регистр X число N и познакомимся с алгоритмом.

Алгоритм

1. Разделить N на 10 000 000 и записать результат в регистр E.

2. Записать исходные данные: 0 - в регистры 1-3, 5, 7-9, A; 6 - в регистр B; 23 (адрес, с которого начинается цикл анализа) – в регистр C.

3. Выделить целую часть N. Если она равна нулю, перейти к шагу 4, в противном случае – к шагу 18.

4. Вызвать N в регистр X, умножить на 10, найти целую и дробную части полученного числа, записать дробную часть в регистр E.

5. Если целая часть, определенная на шаге 4, больше нуля, перейти к шагу 7, в противном случае – к шагу 6.

6. Если дробная часть N, определенная на шаге 4, не равна нулю, перейти к шагу 4, в противном случае – к шагу 10 (сортировку К и Ч интерпретатор закончил).

7. Если $N < 5$, то это – Ч, переходим к шагу 9; в противном случае это – К, переходим к шагу 8.

8. Записать N в регистр П = П + 1 (начальное значение П = 0) и перейти к шагу 4.

9. Записать N в регистр памяти П = П + 1 (начальное значение П = 6) и перейти к шагу 4.

10. Проверка условия Ч = К + 1:

если условие выполняется, работа интерпретатора окончена, перейти к программе-транслятору на шаг 11;

в противном случае вернуться на шаг 9.

11. Подготовка исходных данных для работы транслятора: 0 записать в регистры 5 и B, 6 – в регистр 6, адрес начала цикла 63 – в регистр D.

12. Считать в стек число Ч из регистра П = П + 1 (начальное значение П = 0).

13. Выполнить один раз шаг 12.

14. Проверка условия Ч = 0:

если условие выполнено, перейти к шагу 15;

в противном случае – конец работы программы, перейти к шагу 17.

15. Считать в стек код команды К из регистра П = П + 1 (начальное значение П = 0).

16. Определить команду и соответствующую ей вычислительную операцию.

Если в регистре X число:

5 – сложить содержимое регитров Y и Z, перейти к шагу 13;

6 – вычесть из содержимого регистра Z содержимое регистра Y, перейти к шагу 13;

7 – перемножить содержимое регистров Y и Z, перейти к шагу 13;

8 – разделить содержимое регистра Z на содержимое регистра Y, перейти к шагу 13.

Если в регистре X любое другое число, т. е. определить команду не удалось, перейти к шагу 20.

17. Вызвать в регистр X результат последней операции. Конец работы программы-транслятора.

18. Число цифр в первоначальном числе более семи. Вывать на индикацию сигнал ЕГГОГ.

19. Останов. Вызвать на индикацию сигнал 8,Г нарушено соотношение между командами и числами.

20. Останов. Вызвать на индикацию сигнал 8,С - использована команда, которой нет в списке.

Правила работы с программой нам поможет уяснить небольшой пример: $3 \times 4:2+1$.

Несложно сосчитать, что результат равен 7. Число Ч в этом случае на придуманном нами языке будет выглядеть так: 3748251. Введем Ч в регистр X и нажмем клавиши В/О С/П. Задача для нашего микрокалькулятора оказалась не из простых. Он решает ее около минуты. Если ввести Ч = 4321785, результат будет тот же.

П р о г р а м м а 1.25

"Бенчмарк"

```
                          !
 10 REM БЕНЧМАРК          !
 20 PRINT "НАЧАЛО"        !
 30 K=0                   ! 00.СХ  01.П4
 40 DIM M(5)              !
 50 K=K+1                 ! 02.КИП4  03.ИП4
 60 A=K/2*3+4-5           ! 04.2  05./  06.3   07.*
                          ! 08.4  09.+  10.5   11.-
 70 GOSUB 140             ! 12.ПП 13.25
 80 FOR L=1 TO 5          ! 14.ИП4  15.ПО
 90 M(L)=A                ! 16.ИП4
100 NEXT L                ! 17.FLO  18.16
110 IF K<1000 THEN 50     ! 19.3    20.FEXT   21.-
                          ! 22.FX>0  23.02
120 PRINT "КОНЕЦ"         ! 24.С/П
130 END                   !
140 RETURN                ! 25.В/О
```

Прежде, чем прощаться с микрокалькулятором, посмотрите на программу 1.25 "Бенчмарк". Слева ее запись на Бейсике, справа – на языке микрокалькулятора. С помощью подобных программ оценивают быстродействие вычислительных устройств.

Компьютер СМ-1800 отрабатывает эту программу на Бейсике за 104 сек. Желающие могут попробовать ввести ее в свой микрокалькулятор, нажать клавиши В/О С/П и поставить натурный эксперимент. Решившихся на него просим набраться терпения, так как микрокалькулятор "Электроника МК-52", выполняя программу "Бенчмарк", проработал от пуска до останова 139 мин, а Б3-34 – 165 мин. Так что микрокалькуляторы считают в сто раз медленнее больших ЭВМ.

Микрокалькулятор помогает познать принципы управления работой вычислительного устройства. С этими знаниями легче перейти к языку ассемблера и машинным кодам, нежели пытаться освоить машинный язык, минуя кодово-символьный.

Переходя от программируемого микрокалькулятора к персональному компьютеру, мы не расстаемся с нашим маленьким верным другом, который еще нам послужит.

ПРИЛОЖЕНИЯ

Приложение 1

Лексикон программируемых микрокалькуляторов "Электроника Б3-34, МК-61, -52" и их модификаций

Операция	Кодово-символьный язык	Бейсик*	Паскаль*
Ввод			
цифр от 0 до 9	0 – 9	0 – 9	0 – 9
десятичной точки	•	•	•
порядка набранного числа	ВП	E(D)	E(D)
Изменение знака числа	/-/	-	-
Стирание числа в регистре	СХ	X=0	X:=0
Сложение	+	+	+
Вычитание	-	-	-
Умножение	×	*	*
Деление	÷	/	/
Вычисление функций			
10^x	F10^x	10^X(EXT(X))	Нет
e^x	Fe^x	X=EXP(X)	X:=EXP(X)
lg$x$	FLG	X=LG(X)	Нет
ln$x$	FLN	X=LN(X)	X:=LOG(X)
$\sqrt{x}$	F$\sqrt{\ }$	X=SQR(X)	X:=SQRT(X)
x^2	FX^2	X=X^2	X:=SQR(X)
x^y	FX↑Y	X=X^Y	Нет
sin$x$	FSIN	X=SIN(X)	X:=SIN(X)
cos$x$	FCOS	X=COS(X)	X:=COS(X)
tg$x$	FTG	X=TAN(X)	X:=TAN(X)
arcsin$x$	FSIN^{-1}	X=ASN(X)	X:=ARCSIN(X)
arccos$x$	FCOS^{-1}	X=ACS(X)	Нет
arctg$x$	FTG^{-1}	X=ATN(X)	X:=ARCTAN(X)
$1/x$	F1/X	X=1/X	X:=1/X
Запись числа A в регистр памяти М	ПМ	M=A	M:=A
Вызов из регистра памяти М числа A	ИПМ	X=A	X:=A

Операция	Кодово-символьный язык	Бейсик*	Паскаль*
в регистр результата предыдущей операции	FBX	Нет	Нет
в регистр числа π	Fπ	X=PI	X:=PI
Преобразование			
минут, секунд и десятичных долей секунды в десятичные доли часа (градуса)	$\longrightarrow$ K 0'''	Нет	Нет
десятичных долей часа (градуса) в минуты, секунды и десятичные доли секунды	$\longleftarrow$ K 0'''	Нет	Нет
минут и десятичных долей минуты в десятичные доли часа (градуса)	$\longrightarrow$ K 0'	Нет	Нет
десятичных долей часа (градуса) в минуты и десятичные доли минуты	$\longleftarrow$ K 0'	Нет	Нет
Определение			
модуля числа	K|X|	X=ABS(X)	X:=ABS(X)
знака числа	KЗН	X=SGN(X)	Нет
большего из чисел, находящихся в регистрах X и Y	KMAX	Нет	Нет
Выделение части числа			
целой	K[X]	X=INT(X)	Нет
дробной	K{X}	X=FRAG(X)	Нет
Логические операции			
умножение двух чисел	K $\wedge$	and	and
сложение двух чисел	K $\vee$	or	or
сложение двух чисел по модулю 2	K$\oplus$	Нет	Нет
инверсия	KИНВ	Нет	Нет
Генерация псевдослучайного числа** в диапазоне]0 – 1[	KСЧ	X=RND(1)	Нет
Смещение содержимого стека "вверх"	$\uparrow$ или B$\uparrow$	Нет	Нет
Поворот стека по часовой стрелке	F$\rightarrow$	Нет	Нет
Оператор			
пуск программы	C/П	RUN	G (в интерпретаторах)

Операция	Кодово-символьный язык	Бейсик*	Паскаль*
останов программы	С/П	STOP, END	END
безусловный переход на шаг ab	БП ab	GOTO ab	GOTO ab
переход к подпрограмме, начинающейся с шага ab	ПП ab	GOSUB ab	Вызов по имени
возврат из подпрограммы	В/О	RETURN	Конец процедуры
условные переходы на шаг ab в случае			
$x>0$	FX>0ab	IF X<0 THEN	IF X<0
$x<0$	FX<0ab	IF X>0 THEN	IF X>0
$x\neq 0$	FX≠0ab	IF X=0 THEN	IF X=0
$x=0$	FX=0ab	IF X<>0 THEN	IF X<>0
безусловный переход к шагу программы, номер которого записан в регистре М	КБПМ	Нет	Нет
переход к подпрограмме, номер первого шага которой записан в регистре М	КППМ	ON M GOSUB...	OF CASE M
переход к шагу программы, номер которого записан в регистре М, при выполнении условия $x>0$, $x<0$, $x\neq 0$, $x=0$	К(условие)М	ON M GOTO...	Нет
занесение числа A в регистр, номер которого находится в регистре М	КПМ	Массивы	Массивы
вызов в регистр числа из регистра памяти, номер которого записан в регистре М	КИПМ	Массивы	Массивы
повтор части программы, записанной между шагом ab и оператором Р столько раз, сколько записано в регистре Р (цикл с параметром)	FLP ab	FOR<переменная>=n TO m*** STEP I:...:NEXT	FOR<переменная>=n TO m*** DO (DOUNDO) begin ... end

Операция	Кодово-символьный язык	Бейсик*	Паскаль*
переход в режим записи и редактирования программы	F ПРГ	Нет	Нет
продвижение программы на один шаг вперед	$\overrightarrow{\text{ШГ}}$	→	Нет
продвижение программы на один шаг назад	$\overleftarrow{\text{ШГ}}$	←	Нет
исключение выполнения шага ab	ab.КНОП	Нет	Нет
переход в режим вычислений	F АВТ	RUN	G (в интерпретаторах)
сброс неправильно набранной функции	CF	Нет	Нет
Выражение аргумента тригонометрических функций в			
радианах	Р	RAD	Нет
градах (гонах)	ГРД	GRA	Нет
градусах	Г	DEG	Нет

* В разных версиях Бейсика и Паскаля возможны различные варианты написания операторов.

** В БЗ-34 и его аналогах таких операций нет.

*** Символами n и m обозначены начальная и конечная точки цикла.

Приложение 2

Коды программируемых микрокалькуляторов

Первая цифра кода	Вторая						
	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	+	-	×	:	↔	F 10^x	F e^x
2	F π	F √	F x^2	F 1/X	F x^y	F ↻.	K^{01} →
3	K^{011} →	K|X|	K3H	K^{01} ←	K[X]	K{X}	KMAX
4	X-П0	X-П1	X-П2	X-П3	X-П4	X-П5	X-П6
5	С/П	БП	В/О	ПП	К НОП		
6	П-X0	П-X1	П-X2	П-X3	П-X4	П-X5	П-X6
7	KX≠00	KX≠01	KX≠02	KX≠03	KX≠04	KX≠05	KX≠06
8	КБП0	КБП1	КБП2	КБП3	КБП4	КБП5	КБП6
9	KX≥00	KX≥01	KX≥02	KX≥03	KX≥04	KX≥05	KX≥06
-	КПП0	КПП1	КПП2	КПП3	КПП4	КПП5	КПП6
L	KX-П0	KX-П1	KX-П2	KX-П3	KX-П4	KX-П5	KX-П6
C	KX<00	KX<01	KX<02	KX<03	KX<04	KX<05	KX<06
Г	КП-X0	КП-X1	КП-X2	КП-X3	КП-X4	КП-X5	КП-X6
E	KX=00	KX=01	KX=02	KX=03	KX=04	KX=05	KX=06

"Электроника МК-52, -61"

цифра кода								
7	8	9	-	L	C	Г	E	F
7	8	9	•	/-/	ВП	C_x	В↑	FBX
Flg	Fln	$Fsin^{-1}$	$Fcos^{-1}$	Ftg^{-1}	Fsin	Fcos	Ftg	
			$K^{0/11}$					
K∧	K∨	K⊕	K ИНВ	КСЧ				
X-П7	X-П8	X-П9	X-ПA	X-ПB	X-ПC	X-ПD	X-ПE	
FX≠0	FL2	FX≥0	FL3	FL1	FX<0	FL0	FX=0	
П-X7	П-X8	П-X9	П-XA	П-XB	П-XC	П-XD	П-XE	
KX≠07	KX≠08	KX≠09	KX≠0A	KX≠0B	KX≠0C	KX≠0D	KX≠0E	
КБП7	КБП8	КБП9	КБПA	КБПB	КБПC	КБПD	КБПE	
KX≥07	KX≥08	KX≥09	KX≥0A	KX≥0B	KX≥0C	KX≥0D	KX≥0E	
КПП7	КПП8	КПП9	КППA	КППB	КППC	КППD	КППE	
KX-П7	KX-П8	KX-П9	KX-ПA	KX-ПB	KX-ПC	KX-ПD	KX-ПE	
KX<07	KX<08	KX<09	KX<0A	KX<0B	KX<0C	KX<0D	KX<0E	
КП-X7	КП-X8	КП-X9	КП-XA	КП-XB	КП-XC	КП-XD	КП-XE	
KX=07	KX=08	KX=09	KX=0A	KX=0B	KX=0C	KX=0D	KX=0E	

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Антипов И.Н. Программирование на микрокалькуляторе МК-64. – М.: Просвещение, 1988. – 64 с.

Архипов В. Сделай сам себе ракету // Техника – молодежи. 1986. № 6. – с. 46 – 48.

Астанин Л.Ю., Дарский Ю.Д., Костылев А.А. Применение программируемых микрокалькуляторов для научных и инженерных расчетов. – Л.: Энергоатомиздат, 1986. – 157 с.

Белый Ю.А. Считающая микроэлектроника. – М.: Наука, 1984. – 120 с.

Белый Ю.А. Электронные микрокалькуляторы и техника вычислений. – М.: Знание, 1981. – 64 с.

Блох А.Ш., Павловский А.И., Пенкрат В.Б. Программирование на микрокалькуляторах. – Минск: Вышэйш. шк., 1981. – 192 с.

Бойко А.Б. Игры с микрокалькулятором. – М.: Знание, 1987. – 48 с.

Варламов Р.Г. Эксплуатация и ремонт современных микрокалькуляторов. – М.: Лег. и пищ. пром-сть, 1983. – 120 с.

Виленкин Н.Я., Оксман В.М., Шварцбурд С.И. Микрокалькулятор – школьнику. – М.: Просвещение, 1986. – 95 с.

Гильде В., Альтрихтер З. С микрокалькулятором в руках. – М.: Мир, 1987. – 215 с.

Гильде В., Альтрихтер З. С микрокалькулятором повсюду. – М.: Мир, 1988. – 200 с.

Горстко А.Б., Кочковская С.В. Азбука программирования. – М.: Знание, 1988. – 143 с.

Данилов И.Д. Секреты программируемого микрокалькулятора. – М.: Наука, 1986. – 160 с.

Данилов И.Д., Славин Г.В. 5 вечеров с микрокалькулятором. – М.: Финансы и статистика, 1988. – 109 с.

Дафферт-Смит П. Практическая астрономия с калькулятором. – М.: Мир, 1982. – 176 с.

Дьяконов В.П. Расчет нелинейных и импульсных устройств на программируемых микрокалькуляторах. – М.: Радио и связь, 1984. – 176 с.

Дьяконов В.П. Справочник по расчетам на микрокалькуляторах. – М.: Наука, 1985. – 224 с.

Епанечников В.А., Цветков А.Н. Справочник по прикладным программам для микрокалькуляторов. – М.: Финансы и статистика, 1988. – 320 с.

Кройль Г. Что умеет мой микрокалькулятор? – М.: Мир, 1981. – 132 с.

Микрокалькуляторы в играх и задачах. – М.: Наука, 1986. – 160 с. – (Сер. Кибернетика – неограниченные возможности и возможные ограничения).

Микрокалькуляторы: Технические и конструктивные характеристики / Е.Ю. Кузнецов, Б.В. Острецов, Л.К. Минкин, Ю.Н. Егорова – М.: Радио и связь, 1984. – 128 с. – (Сер. Массовая библиотека инженера. Вып. 40.)

Оксман В.М. Применение микрокалькуляторов при обучении математике в средних ПТУ: Метод. пособие. – М.: Высш. шк., 1984. – 96 с.

Поснова М.Ф., Поснов Н.Н. Микрокалькуляторы с программным управлением в учебной лаборатории. – Минск: Изд-во Белорус. ун-та, 1985. – 207 с.

Пухначев Ю.В., Данилов И.Д. Микрокалькуляторы для всех. – М.: Знание, 1986. – 192 с.

Романовский Т.Б. Микрокалькуляторы в рассказах и играх. – Рига: Зинатне, 1984. – 124 с.

Романовский Т.Б. Микрокалькуляторы в рассказах и играх. – Рига: Изд-во Риж. ун-та, 1987. – 192 с.

Свидерский Э.А. Решение технологических задач в машиностроении с применением микрокалькуляторов. – М.: Машиностроение, 1987. – 157 с.

Славин Г.В. Программирование на программируемых микрокалькуляторах типа "Электроника Б3-34". – Таллинн.: Валгус, 1984. – 126 с.

Технико-экономические расчеты на программируемых микрокалькуляторах "Электроника". – М.: Финансы и статистика, 1987. – 150 с.

Трохименко Я.К. Игры с микроЭВМ. – Киев: Техника, 1986. – 120 с.

Трохименко Я.К. Программирование микрокалькуляторов МК-61 и МК-52. – М.: Радио и связь, 1987. – 209 с.

Трохименко Я.К., Любич Ф.Д. Инженерные расчеты на микрокалькуляторах. – Киев.: Техника, 1980. – 394 с.

Трохименко Я.К., Любич Ф.Д. Инженерные расчеты на программируемых микрокалькуляторах. – Киев.: Техника, 1985. – 328 с.

Трохименко Я.К., Любич Ф.Д. Микрокалькулятор, Ваш ход! – М.: Радио и связь, 1985. – 224 с.

Трохименко Я.К., Любич Ф.Д. Радиотехнические расчеты на микрокалькуляторах. – М.: Радио и связь, 1983. – 256 с.

Трохименко Я.К., Любич Ф.Д. Справочник: радиотехнические расчеты на программируемых микрокалькуляторах. – М.: Радио и связь, 1988. – 304 с.

Трохименко Я.К., Любич Ф.Д. Программирование микрокалькуляторов "Электроника МК-52" и "Электроника МК-61". – Киев: Техника, 1987. – 208 с.

Францевич Л.И. Обработка результатов биологических экспериментов на микроЭВМ "Электроника Б3-21". – Киев: Наук. думка, 1979. – 91 с.

Цветков А.Е., Епанечников В.А. Прикладные пограммы для микроЭВМ "Электроника Б3-34", "Электроника МК-56" и "Электроника МК-54". – М.: Финансы и статистика, 1984. – 175 с.

Цветков А.Н. Прикладные программы для микроЭВМ "Электроника Б3-21". – М.: Финансы и статистика, 1982. – 127 с.

Цимринг Ш.Е. Специальные функции. Программы для микрокалькулятора "Электроника Б3-21". – М.: Радио и связь, 1983. – 120 с.

Чакань А. Что умеет карманная ЭВМ? /Пер. с венгер. – М.: Радио и связь, 1982. – 144 с.

Чапка А.М. Расчетно-проектировочные работы на программируемых микрокалькуляторах. – М.: Машиностроение, 1988. – 143 с.

Человек и компьютер //Наука и жизнь, 1988. № 2; 1989. № 2, 4, 12.

Шандрина Л.П. Торговые вычисления. – М.: Экономика, 1986. – 142 с.

Шапиро С.И., Бойко А.Б. Программируемые микрокалькуляторы в обучении. – М.: Радио и связь, 1989. – 256 с.

Шелест А.Е. Микрокалькуляторы в физике. – М.: Наука, 1988. – 272 с.

Школа начинающего программиста //Наука и жизнь. 1985. № 6 – 8, 11, 1986. № 1, 3, 5, 7; 1989. № 1.

ГЛАВА 2

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР

ОТ МИКРОКАЛЬКУЛЯТОРА К ПЕРСОНАЛЬНОМУ КОМПЬЮТЕРУ

Персональные компьютеры трудно даже сравнивать с программируемыми микрокалькуляторами, настолько различны их возможности. Однако ни один сегодняшний компьютер не может соперничать с микрокалькулятором по стоимости.

Есть и еще аргумент в пользу того, чтобы не сразу расставаться с микрокалькуляторами. Программы для них и даже для простейшего персонального компьютера различаются не менее, чем сами эти устройства. И все же отличия не так уж велики. Подтверждение тому – микрокалькулятор "Электроника МК-85", а также множество аналогичных зарубежных микрокалькуляторов, программируемых не на символьно-кодовом языке, а на Бейсике. Переход к использованию компьютера будет не столь трудным, если программы на языках Бейсик и Паскаль на первых порах пояснять программами для микрокалькуляторов.

Триады программ, посвященных одной проблеме, назовем трилингвами*. Впрочем, когда возможности микрокалькулятора уже не будут соответствовать решаемым задачам, трилингвы уступят место билингвам, которые послужат своего рода ступенями в освоении компьютера, а также самых популярных языков программирования – Бейсика и Паскаля.

---

* Трилингва – трехъязычная надпись. Один из примеров – текст на Розеттском камне. Жан Франсуа Шампольон (1790 – 1832), расшифровав эту трилингву, раскрыл тайну древнеегипетских иероглифов. Надеемся, что при знакомстве с трилингвами во второй главе книги знание одного языка программирования (кодов программируемых микрокалькуляторов) поможет нашим читателям освоить другие (Бейсик, Паскаль).

Язык Бейсик был разработан сотрудниками Дартмутского колледжа (США) в 1964 г. Считается, что своим названием Бейсик обязан упрощенному английскому языку "basic English" (в дословном переводе – основной английский), насчитывающему около 500 слов. Этот вариант английского предназначался для эмигрантов из неанглоязычных стран. Авторы Бейсика поясняют, что BASIC – это аббревиатура английской фразы "Beginner's All-purpose Symbolic Instruction Code" (в переводе – многоцелевой язык символических команд для начинающих).

Бурный рост его популярности был вызван массовым приобщением к программированию неспециалистов в этой области. В результате сегодня не встретишь персональный компьютер, который не понимал бы этого языка.

Транслятор Бейсика в большинстве случаев "зашит" в постоянное запоминающее устройство* и копируется в оперативное запоминающее устройство** при включении так, что компьютер сразу превращается в Бейсик-машину. Это можно встретить, например, в ДВК-1, "Commodore 64", IBM PC и других ЭВМ.

Язык Паскаль, названный в честь французского математика и философа Блеза Паскаля (1623 – 1662), разработал в 1971 г. швейцарский математик Николаус Вирт. Он намеревался на основе фундаментальных понятий создать язык, пригодный для обучения программированию.

В адрес этих языков, несмотря на их популярность, высказывают множество замечаний. Бейсик "воспринимает критику" и довольно быстро эволюционирует, перенимая от Паскаля и других языков такие свойства, как, например, структурность и модульность. Наиболее уязвимым для критики Бейсик считался из-за того, что основные его версии были реализованы на распространенных ЭВМ лишь в виде интерпретаторов. Не так давно этот недостаток преодолен – появилось несколько мощных компиляторов Бейсика.

Результат превзошел ожидания – распространение получила технология, когда программу составляют и отлаживают с помощью интерпретатора, а затем готовую программу компилируют в машинные коды. Особенно удобен этот прием всякий раз, когда недостаточно быстродействие или исчерпан лимит памяти.

Язык Паскаль в свою очередь приобретает полезные качества, вошедшие в практику программирования благодаря Бейсику: возможность перехода к метке, использование цветной графики и музыкальных возможностей компьютера.

---

* Постоянное запоминающее устройство (ПЗУ) – аналог английского Read Only Memory (ROM), в переводе – память с доступом только для считывания.

** Оперативное запоминающее устройство (ОЗУ) – аналог английского Read Access Memory (RAM), в переводе – память с произвольным доступом.

Можно утверждать, что языки Бейсик и Паскаль идут навстречу друг другу. Иной раз бывает нелегко отличить фрагменты программ, написанные на так называемых Турбоверсиях Бейсика и Паскаля.

Программы, входящие в трилингву "Метод секущих" (программа 2.1), вписаны в рамки структурной диаграммы. С ее помощью удается графически выделить управляющие конструкции алгоритмов: цикл "пока", альтернативу, цикл "до", цикл с параметром, множественное ветвление, вызов процедуры.

П р о г р а м м а 2.1

Метод секущих
(цикл "пока")

```
00 С/П  П1   СХ   С/П  П2   СХ   С/П  ПЕ
08 ИП1  П0   ПП   48   П4
13 ИП1  ИП2  —    F|X| ИПЕ  —    /—/  FX<0 46
22 ИП2  П0   ПП   48   П5
27 ИП2  ИП1  —    X    ИП5  ИП4  —    :
35 ИП2  F→   —    ИП2  П1   F→   П2   ИП5  П4
44                                                БП 13
46 ИП2  С/П

48 ...  С/П
```

```
 0 INPUT X1 : INPUT X2 : INPUT E
 8 X=X1 : GOSUB 48 : Y1=Y
13 IF ABS(X1-X2)<=E GOTO 46
22 X=X2 : GOSUB 48 : Y2=Y
27 X3=X2-Y2*(X2-X1)/(Y2-Y1)
35 X1=X2 : X2=X3 : Y1=Y2
44                                  GOTO 13
46 PRINT X2 : STOP

48 Y=... : RETURN
```

```
VAR E,X,X1,X2,X3,Y,Y1,Y2:REAL;
PROCEDURE FNY BEGIN Y:=... END;
BEGIN READLN(X1); READLN(X2); READLN(E)
X:=X1; FNY; Y1;=Y;
WHILE ABS(X1-X2)>E DO BEGIN
X:=X2; FNY; Y2;=Y;
X3:=X2-Y2*(X2-X1)/(Y2-Y1);
X1:=X2; X2:=X3; Y1:=Y2                END;
WRITELN(X2) END.
```

Сетка структурной диаграммы помогает выделить алгоритм, заложенный в программу, а также освоить правила структурного программирования.

Хотя читатель уже мог видеть программы, вписанные в структурные диаграммы, вряд ли ему приходилось использовать диаграммы на практике. Владельцам микрокалькуляторов хорошо известно, что связанные с мизерным объемом их памяти жесткие ограничения заставляют создавать своего рода программные "спагетти", в которых иной раз не может разобраться и автор. Что же говорить о тех, кто захочет воспользоваться программой или модернизировать ее?

ЦИКЛ ТИПА "ПОКА"

Трилингва "Метод секущих" предназначена для поиска корня алгебраического уравнения, когда на оси X выбирают две точки $X1$ и $X2$, а вычисления ведут итерационным методом по формуле

$$X3 = X2 - F(X2)\,\frac{X2 - X1}{F(X2) - F(X1)}\,. \qquad (2.1)$$

Очередные приближения к точке пересечения графика $F(x)$ с осью X, т. е. к корню уравнения рассчитываются до тех пор, пока расстояние между двумя соседними опорными точками $X1$ и $X2$ не станет меньше заданного значения E. В этом случае E называют заданной точностью.

Слово "пока" в описании алгоритма определило программное средство, которым удобно воспользоваться для его реализации, – это цикл "пока". Он состоит из тела цикла и заголовка.

Телом цикла называют функциональный блок, операторы которого выполняются в естественной последовательности: слева-направо и сверху-вниз. В заголовок входит булево выражение, принимающее два значения: "истинно" (англ. true) или "ложно" (англ. false).

Как и в программах для микрокалькуляторов, в основных версиях Бейсика цикл "пока" реализуется лишь парой операторов: условного перехода перед телом цикла и замыкающего его безусловного перехода. Оба перехода детерминированы. Так, безусловный возвращает управление условному, который, если логическое условие выполняется, передает управление на оператор, стоящий за последним в цикле оператором безусловного перехода. Словесное описание логики выполнения цикла "пока", опирающегося на метку, выглядит сложно. Разобраться с ним поможет структурная диаграмма "Метод секущих".

В новых версиях Бейсика появились специальные операторы, позволяющие реализовать цикл "пока" без опоры на метку:

```
WHILE БУЛЕВО ВЫРАЖЕНИЕ: ФУНКЦИОНАЛЬНЫЙ БЛОК: WEND
```

Функциональный блок, составляющий тело цикла, выполняется до тех пор, пока булево выражение возвращает значение "истинно". Специалисты знают, что такой вариант цикла "пока" позаимствован у языка Паскаль. Впрочем, в Паскаль эта конструкция перешла из Алгола – дедушки многих современных языков программирования.

Из итерационной формулы (2.1) следует, что для расчета нового приближения к корню $X3$ (точка пересечения секущей, проходящей через $F(X1)$ и $F(X2)$, с осью X) необходимо вычислить два значения анализируемой функции в точках $X1$ и $X2$. В программах "Метод секущих" достаточно выполнить лишь одно вычисление. Это особенно важно для тех случаев, когда значение находят по сложному алгоритму или определяют опытным путем. Два значения функции необходимы только в первом приближении: один раз до входа в цикл и один раз в теле цикла.

За все, однако, приходится платить. Ускорение работы программы достигается, как правило, за счет некоторого увеличения объема памяти, отводимого под хранение программы и переменных. Быстродействие и необходимый объем памяти – это взаимосвязанные ресурсы, и если удается оптимизировать одновременно то и другое, значит речь идет о высоком классе программиста или низком качестве исходной программы.

Первое, что бросается в глаза при сравнении программ 2.1, – громоздкое описание всех переменных (на этот раз они вещественные – real) в заголовке Паскаль-программы. Без такого описания они как бы не существуют, и попытка запустить Паскаль-программу будет напрасной, а транслятор выдаст на экран сообщение "Использование необъявленных переменных".

Этого нет в программах на языке микрокалькулятора или на Бейсике. Число ячеек памяти в микрокалькуляторе определяется его конструкцией и заранее известно. На Бейсике по умолчанию считается, что все переменные (речь идет об основном типе переменных – неиндексных) уже существуют до запуска программы и хранят нулевые значения. Но это нередко оказывается "медвежьей услугой", так как транслятор будет опираться на нулевое значение всякой переменной, исходное значение которой не было определено по забывчивости. Из-за этого может быть найден неверный ответ.

В микрокалькуляторе оперативная память жестко разделена на область хранения кодов программы и область регистров па-

мяти. В персональном компьютере такого деления нет – граница зон памяти подвижна и определяется характером решаемой задачи.

АЛЬТЕРНАТИВА

Какие же еще управляющие конструкции используются при программировании для ЭВМ? Трилингва "Построение треугольника" (программа 2.2) дает ответ на вопрос, можно ли построить треугольник из отрезков длиною *A*, *B* и *C*. На индикатор микрокалькулятора (дисплей компьютера) выводится нуль, если сумма любых двух отрезков меньше третьего и их набор не годится для построения треугольника, или единица, если треугольник возможен. В программе использована еще одна управляющая конструкция – альтернатива.

Альтернатива – это булево выражение и два функциональных блока (два плеча), из которых при каждом исполнении альтернативы работает лишь один. Какой, зависит от текущего значения булева выражения.

В программах часто встречаются неполные альтернативы, когда одно из плеч пусто. На неструктурированных языках неполную альтернативу, как правило, реализуют условным переходом вниз по программе. Полную альтернативу в подобных случаях записывают, как и цикл "пока", парой операторов условного и безусловного переходов, причем все переходы ведут к нижерасположенным меткам.

Кстати, о метках. Продолжая сравнивать программы трилингвы, читатель заметит, что строки Паскаль-программы не нумеруются. Более того, строка такой программы не имеет для машины самостоятельного смысла. Программу разбивают на строки лишь для того, чтобы ее легче было воспринимать человеку. В то же время, строка на Бейсике (или шаг в программе для микрокалькулятора) – это четкая языковая единица. Номера строк (шагов) выполняют функции меток, опираясь на которые можно реализовать разветвленный алгоритм.

Понятие "метка" есть и в Паскале. Здесь, однако, все используемые метки необходимо объявить в заголовке программы, подобно тому, как это делалось с переменными.

Впрочем, в языке Паскаль можно обойтись без меток. Квалификацию программиста иногда ставят в обратную зависимость от числа меток в его Паскаль-программах.

Последние версии Бейсика также дополнены служебным словом ELSE и позволяют записывать полные альтернативы без опоры на метку. Справедливости ради отметим, что некоторые версии Бейсика к тому же не используют нумерацию строк.

Программа 2.2

Построение треугольника
(альтернатива)

```
00 СХ   С/П ПА  С/П   ПВ  С/П  ПС
07 +    —   ИПА ИПВ   +   ИПС  —   *  ИПА  ИПС  +
18 ИПВ  —   *   FX<0  26
23 1    БП  27
26                    0
27 С/П
```

```
 0 CLS:INPUT A,B,C
 7 IF (A+B-C)*(A+C-B)*(B+C-A)<0 THEN 26
23 PRINT 1:GOTO 27
26                    PRINT 0
27 STOP
```

```
VAR A,B,C:REAL;
BEGIN
READLN (A,B,C);
IF (A+B-C)*(A+C-B)*(B+C-A)<0
THEN WRITELN(1)  ELSE WRITELN(0)
END.
```

Иногда альтернатива имеет более двух плеч, и тогда ее называют множественным ветвлением. В заголовке стоит уже не булево, а целочисленное выражение. Результат его вычисления – натуральное число, определяющее номер функционального блока (плеча), который должен выполняться.

ВЛОЖЕННЫЕ СТРУКТУРЫ

Ввести еще одно важное понятие структурного программирования поможет трилингва "Метод половинного деления"*

* Математикам известны и другие названия этого метода – "бисекция" или "дихотомия". Рассказ о нем часто иллюстрируют шуточной задачей о поимке льва: пустыню, где он обитает, огораживают забором, а затем разгораживают пополам; ту половину, в какой оказался лев, вновь делят пополам; и так до тех пор, пока лев не очутится в клетке.

(программа 2.3), выполняющая поиск корней алгебраического уравнения, но не вблизи двух заданных точек $X1$ и $X2$, как в методе секущих (см. программу 2.1), а в интервале (a, b).

Тело цикла "пока" трилингвы 2.3 состоит из двух функциональных блоков, второй из которых – альтернатива. В подобных случаях говорят, что одна управляющая конструкция (в данном случае альтернатива) вложена в другую (в цикл "пока").

В примере

"на структурных диаграммах, в которые вписаны программы, реализующие метод половинного деления, входящие в книгу "От микрокалькулятора к персональному компьютеру", вышедшую в издательстве МЭИ, ..."

такое вложение просматривается ясно.

Текст этот выглядит корявым и надуманным, но иллюстрирует то, что вложенность свойственна не только языкам программирования, но и естественным. Удачный пример такого рода – известное стихотворение из английской народной поэзии (русский перевод С. Маршака) "Дом, который построил Джек".

Цикл также не является изобретением программистов. Вспомним повторяющийся припев любой песни.

В традиционной записи вложенные структуры отмечают отступами от левого края, своего рода абзацами.

Итак, все операторы компьютер выполняет в их естественной последовательности, нарушаемой только управляющими конструкциями. Некоторые операторы допустимо поменять местами, а иные нельзя, поскольку они находятся в причинно-следственной связи. Так, операторы строки 21 Бейсик-программы (см. программу 2.3) можно переставлять и выполнять последовательно в любом порядке. Строки 15 и 21 поменять местами нельзя – возникнет ошибка.

Время последовательного выполнения программы определяется как сумма временны́х интервалов, необходимых для вычисления каждого оператора. Но если операторы допустимо менять местами, лучше выполнять их параллельно. В этом случае временны́е затраты на выполнение всех таких операторов определялись бы самым "медленным" из них*.

Для персональных компьютеров проблема распараллеливания решается по-другому: периферийные устройства дополняют вспомогательной буферной памятью и иногда даже собственным микропроцессором. Что это дает? Например, если значение X (строка 21 в программе 2.3) выводить на принтер, то распечатка промежуточных результатов будет ограничивать скорость решения задачи. Работая с принтером, снабженным буферной

---

* Так действуют системы и языки параллельного программирования, например Ада.

Программа 2.3

"Метод половинного деления"
(цикл "пока" со встроенной альтернативой)

```
00 П0     F→     П3     F→     П2     ПП     38     П1
08 ИП3    ИП2    —      ИП0    —      FX>0   36
15 ИП2    ИП3    +      2      :      П4
21 С/П    ПП     38
24 ИП1    X      FX>0   32
28 ИП4    П2     БП     34
32                                    ИП4    П3
34                                                  БП     08
36 ИП4    С/П

38 FX^2   3      —      В/О
```

```
 0 INPUT A,B,E : X=A : GOSUB 38 : Y1=Y
 8 IF B-A<E GOTO 36
15 X=(A+B)/2
21 PRINT X : GOSUB 38
24       IF Y*Y1>=0 GOTO 32
28 B=X : GOTO 34
32                          A=X
34                                          GOTO 8
36 PRINT X : STOP

38 Y=X^2-3 : RETURN
```

```
VAR A,B,E,X,Y,Y1:REAL;
PROCEDURE FNY BEGIN  Y:=SQR(X)-3  END;
BEGIN READLN(A,B,E); X:=A; FNY; Y1:=Y;
WHILE B-A>=E DO BEGIN
X:=(A+B)/2; WRITELN(X); FNY;
            IF Y*Y1>=0
THEN A:=X          ELSE B:=X          END;
WRITELN(X)    END.
```

памятью, ЭВМ сбрасывает в нее очередной результат и продолжает вычисления, не ожидая, пока "неповоротливый" принтер его распечатает. Тем самым операторы строки 21 Бейсик-программы будут выполняться параллельно.

ЦИКЛ ТИПА "ДО"

Трилингва 2.4 отыскивает корни алгебраического уравнения методом Ньютона (частный случай метода секущих). Метод секущих сводится к методу Ньютона всякий раз, когда исходные точки на оси X совпадают. В таком случае секущая превращается в касательную, поэтому для расчета методом Ньютона необходимо знать не только вид функции, но и ее производную. Итерационная формула при этом приобретает вид

$$X = X - \frac{F(X)}{F'(X)}. \qquad (2.2)$$

Программа 2.4

"Метод Ньютона"
(цикл "до")

```
00 С/П ПО   ПЕ
03 ИПО П1   ПП   20 ИПО  —    /—/   ПО
11 ИПЕ ИП1  ИПО  — —   FX>0  03
18 ИПО С/П
20 ИПО FX2  3    — ИПО 2    *     :    В/О
```

```
 0 INPUT X,E
 3 X1=X:GOSUB 20:X=X-D
11 IF ABS(X1-X)>E THEN 3
18 PRINT X:STOP
20 D=(X*X-3)/2/X:RETURN
```

```
VAR D,X,X1,E:REAL;
PROCEDURE DELTA BEGIN D:=(SQR(X)-3)/2/X END;
BEGIN
READLN(X,E)
REPEAT X1:=X;DELTA;X:=X-D
UNTIL ABS(X1-X)<=E;
WRITELN(X)
END.
```

В трилингве 2.4 "Метод Ньютона" использован цикл "до". Он отличается от цикла "пока" тем, что тело цикла "до" выполняется как минимум один раз, в то время, как тело цикла "пока" может не быть выполнено ни разу.

Цикл "до" считается вспомогательной структурной управляющей конструкцией. Впрочем, есть ситуации (например, в программах 2.4), когда управляющее циклом булево выражение приобретает смысл только после выполнения тела цикла. В подобных случаях цикл "до" удобнее*. В неструктурных языках цикл реализуют условным переходом вверх по программе.

Возвращаясь к программам "Метод секущих", отметим, что и в этом случае цикл "до" был бы удобнее: он исключает ситуацию, когда ввод двух начальных точек $X1$ и $X2$, отстоящих друг от друга менее, чем на значение точности, дает неверный результат.

В поисках корня алгебраического уравнения рассмотренными методами помог бы цикл с выходом из середины, когда сначала вычисляется очередное приближение к корню, проверяется точность расчета, а затем, если заданная точность не достигнута, осуществляется подготовка к новому приближению (пересылка текущих значений X). Такой цикл редко встречается при структурном программировании. На языке Паскаль цикл с выходом из середины можно реализовать только опираясь на метку, а это, как мы уже говорили, считается признаком низкой культуры программирования.

ЗОЛОТОЕ СЕЧЕНИЕ

Прежде чем расставаться с определением циклов, рассмотрим еще несколько примеров.

Если у вас есть знакомый художник, попросите его разделить произвольный отрезок на две неравные части. Возможно, он сделает это в золотом отношении. Золотое сечение можно узреть на картине, когда отношение половины периметра к длине большой стороны равно отношению длин большой и малой сторон.

Такое чувство меры полезно внушить и компьютеру, занимающемуся поиском единственной точки минимума некоторой унимодальной** функции на интервале (a, b) (трилингва 2.5 "Золотое сечение").

---

* Здесь цикл "до" можно заменить циклом "пока", стоит придать булеву выражению соответствующее значение перед входом в цикл.

** Унимодальной называют функцию, имеющую один минимум или максимум.

Программа 2.5

"Золотое сечение"
(цикл "до" со встроенной альтернативой)

```
00 ↑   0    .    6    1    8   ПС   0        3    8    2    ПD
13 С/П ПЕ   ПВ   ПА   ПП   58  ПП   70
21 ИП2 ИП1  –    FX>0 36
26 ИП4 ПВ   ИП3  П4                          ИП3  ПА   ИП4  П3
   ИП1 П2   ПП   58   БП   43                ИП2  П1   ПП   70
36
43 ИПВ ИПА  –    K|X|  ИПЕ –    /–/ FX>0 21
52 ИПА ИПВ  +    2     /    С/П
58 ИПС ИПD  *    ИПD   ИПВ *    +    П3   ПП   82   П1   В/0
70 ИПD ИПА  *    ИПС   ИПВ *    +    П4   ПП   82   П2   В/0
82 ....................В/0
```

```
 0 C=.618:D=.382
13 INPUT 'A,B?';A,B:IF A>=B GOTO 13
15 INPUT 'ТОЧНОСТЬ?';E:GOSUB 58:GOSUB 70
21 IF Y1>Y2 THEN 36
26 B=X2:X2=X1:Y2=Y1
   GOSUB 58:GOTO 43
36                          A=X1:X1=X2:Y1=Y2
                            GOSUB 70
43 IF ABS(B-A)>E THEN 21
52 X=(A+B)/2:PRINT X:STOP
58 X1=C*A+D*B:X=X1:GOSUB 82:Y1=Y:RETURN
70 X2=D*A+C*B:X=X2:GOSUB 82:Y2=Y:RETURN
82 Y=..........................:RETURN-
```

```
CONST C=.618;D=.382;
VAR A,B,E,Y1,Y2,Y,X1,X2:REAL;
PROCEDURE FNY
BEGIN Y:=....................... END;
PROCEDURE XY1
BEGIN X1:=C*A+D*B;X:=X1;FNY;Y1:=Y END;
PROCEDURE XY2
BEGIN X2:=D*A+C*B;X:=X2;FNY;Y2:=Y END;
BEGIN
READLN(A,B,E);XY1;XY2;
REPEAT IF Y1>Y2
THEN BEGIN
A:=X1;X1:=X2;
Y1:=Y2;XY2 END
                            ELSE BEGIN
                            B:=X2;X2:=X1;
                            Y2:=Y1;XY1 END
UNTIL ABS(B-A)<=E;
X:=(A+B)/2;    WRITELN(X)
END.
```

Основа алгоритма – последовательное приближение к минимуму в цикле "до", пока не будет достигнута заданная точность E. В исходном интервале (a, b) проставляют две точки с координатами $X1$ и $X2$, разделяющие его в золотом соотношении. Если $F(X1) > F(X2)$, то для дальнейшего деления выбирают интервал ($X1$, b). Соответственно при $F(X1) \leq F(X2)$ будет выбран интервал (a, $X2$). В первом случае одной из двух краевых точек выбранного отрезка останется $X1$, а во втором – $X2$. Это следует из свойств золотого сечения.

Таким образом, из двух значений функции, необходимых при каждом новом приближении, вычисляется только одно: другое было получено в предыдущем приближении. Альтернатива, вложенная в цикл "до", сужает интервал неопределенности (a, b), как бы подтягивая его левый или правый край.

Сравнивая в трилингве "Золотое сечение" Бейсик-программу с программой для микрокалькулятора, можно заметить, что фрагменты ввода исходного интервала (a, b) различны.

Во-первых, Бейсик-оператор ввода в строке 13 содержит комментарий, подсказывающий пользователю, что именно следует ввести. Это облегчает диалог и не требует каких-либо дополнительных инструкций в отличие от работы с микрокалькулятором.

Во-вторых, строка 13 содержит элемент, который программисты называют "защитой от дурака"*, не позволяющий вводить в ЭВМ заведомо неверные данные. Действия человека, способные привести к необратимым последствиям (например, стиранию данных в архивной памяти), подвергаются двойному и тройному контролю или обставляются комментариями типа "Вы действительно собираетесь это сделать?". Обидеться на такую фразу компьютера способен лишь человек, никогда не стиравший по рассеянности результаты своей работы. В программу необходимо вставлять фрагменты "защиты от дурака", ибо есть и шутники, получающие особое удовольствие от попыток розыгрыша компьютера и сообщающие ЭВМ заведомо ложную информацию. Машина должна быть защищена от ошибок человека, будь они вольные или невольные.

ЦИКЛ С ПАРАМЕТРОМ

Возвращаясь к разговору о циклах, следует отметить, что в программах чаще всего можно встретить не цикл "до" и не цикл "пока", а цикл с параметром. Его применяют, когда заранее известно, сколько раз должно быть выполнено тело цикла.

---

* Есть и соответствующее понятие "дуракоустойчивость" (от англ. foolproof), характеризующее защиту программы в целом.

Пример – трилингва 2.6 аппроксимации методом наименьших квадратов табличной зависимости функции одной переменной линейным уравнением

$$Y = A + BX, \tag{2.3}$$

коэффициенты которого находят, решая систему

$$\begin{aligned} AN + B\Sigma X &= \Sigma Y, \\ A\Sigma X + B\Sigma X^2 &= \Sigma XY. \end{aligned} \tag{2.4}$$

Программы, входящие в трилингву 2.6, решают задачу, возникающую, например, при статистической обработке результатов однофакторного эксперимента, когда необходимо линейным уравнением описать полученные точки. В микрокалькулятор или компьютер достаточно ввести число точек (N) и их координаты.

Коэффициенты системы линейных уравнений (2.3) формируют в цикле, где в качестве параметра используется номер точки. Число повторений цикла при этом задает N. После ввода данных система решается известным численным методом Крамера.

Организуя циклы с параметром, следует помнить о двух "нельзя":

нельзя изменять параметр цикла в его теле;

нельзя использовать параметр цикла после выхода из него.

Этими ограничениями цикл с параметром отличается от циклов "до" и "пока".

Тему "нельзя" уместно продолжить. Обратимся, например, к первой строке Бейсик-программы "Метод наименьших квадратов" (программа 2.6). Записанные в ней операторы обнуления могут показаться лишними, если вспомнить, что по принципу умолчания после запуска Бейсик-программы командой RUN все переменные обнуляются.

Но не стоит злоупотреблять использованием принципа умолчания, если хотите следовать классическому стилю программирования.

Нередко принцип умолчания по-разному толкуется в различных реализациях одного языка. Так, Бейсик-ДВК не обнуляет переменные, и те, кто привык работать с более совершенными компьютерами, часами пытаются отыскать элементарную ошибку в, казалось бы, отлаженной программе. То же самое приходится иногда наблюдать при программировании на Паскале.

Цикл с параметром несложно заменить на цикл "пока" или цикл "до", организовав счетчик. Это, впрочем, предусмотрено в Бейсик-машинах, где цикл с параметром реализуется циклом "пока" ("Электроника Д3-28") или циклом "до" ("Искра 226").

Программа 2.6

"Метод наименьших квадратов" (цикл с параметром)

```
00 П7    СХ    П1    П2    П3    П4
06 ИП7   П0
08 С/П   П5    F→    П6    ИП1   +     П1
15 ИП2   ИП5   +     П2    ИП3   ИП6   ИП5   X     +     П3
25 ИП4   ИП6   FX^2  +     П4
30 FL0   08
32 ИП7   ИП4   *     ИП1   FX2   —     ПD
39 ИП2   ИП4   *     ИП3   ИП1   *     —     ИПD   :     ПА
49 ИП7   ИП3   *     ИП1   ИП2   *     —     ИПD   :     ПВ
59 ИПА   С/П   ИПВ   С/П
```

```
 0 INPUT N : SX=0 : SY=0 : SXY=0 : SX2=0
 6 FOR I=1 TO N
 8 INPUT X,Y : SX=SX+X
15 SY=SY+Y : SXY=SXY+X*Y
25 SX2=SX2+X*X
30 NEXT I
32 D=N*SX2-SX*SX
39 A=(SY*SX2-SXY*SX)/D
49 B=(N*SXY-SX*SY)/D
59 PRINT A : PRINT B
```

```
VAR A,B,D,X,Y,SX,SY,SXY,SX2:REAL;
VAR I,N:INTEGER;
BEGIN READLN(N);
SX=:0; SY:=0; SXY:=0; SX2:=0;
FOR I:=1 TO N DO BEGIN
READLN(X,Y); SX:=SX+X;
SY:=SY+Y; SXY:=SXY+X*Y;
SX2:=SX2+X*X;
END;
D:=N*SX2-SX*SX;
A:=(SY*SX2-SXY*SX)/D;
B:=(N*SXY-SX*SY)/D;
WRITELN(A); WRITELN(B);END.
```

На Бейсике параметр цикла может принимать любые вещественные значения с шагом (step)*, который указывают в

* Здесь и далее будем указывать английский эквивалент соответствующего термина.

заголовке. По умолчанию (строка 6 в программе 2.6) шаг принимают равным единице. На Паскале параметр цикла – всегда целочисленная переменная, изменяющаяся с шагом, равным единице (цикл DO) или минус единице (цикл DOWNDO).

Если параметры заголовка несовместимы, например

```
FOR X=2 TO 6 STEP -2 ,
```

тело цикла выполняется единожды (цикл типа "до"), либо ни разу (цикл типа "пока").

ПОДПРОГРАММЫ И ПРОЦЕДУРЫ

Трилингва 2.7 "Календарь" предназначена для обработки дат. Она позволяет определить число дней между двумя датами и дни недели (0 – воскресенье, 1 – понедельник, ... 6 – суббота), приходящиеся на эти даты. Алгоритм решения задачи прост: по введенной дате (ДД, ММ, ГГГГ) вычисляется номер дня на хронологической оси. Зная разность между двумя номерами, нетрудно решить поставленную задачу.

Программы с подпрограммами, входящие в трилингву "Календарь", напоминают тексты песен, где припев записан всего раз, хотя его исполняют после каждого куплета. Участки программ, выполняемые два и более раз, выделяют в подпрограммы (язык микрокалькулятора, Бейсик) или процедуры (Паскаль), вызываемые специальными операторами (сравните: ПРИПЕВ – GOSUB). Так удается экономить место в памяти ЭВМ, занимаемое программой, и время программиста, затрачиваемое на ее составление и ввод в машину.

В подпрограммы нередко выделяют отдельные участки программ, имеющие самостоятельный смысл. Это позволяет разбить программу на фрагменты, которые создают и отлаживают порознь. Такие фрагменты компонуют лишь на заключительном этапе.

В программе, разбитой на подпрограммы, легче разобраться. На вызов подпрограммы требуется время. И все же ради наглядности иной раз жертвуют быстродействием.

Аналогия программы с песней не совсем полная. Вряд ли есть песня, припев которой содержал бы новый припев, так сказать, припев второго уровня. В программах подобное встречается часто. Конкретная версия языка программирования определяет допустимое число уровней подпрограмм.

Структурные диаграммы, помещенные в этой книге, можно раскрасить. Образец раскраски приведен на цветных вкладках журнала "Наука и жизнь" (1987, № 4):

Программа 2.7

"Календарь"
(использование подпрограмм)

```
00 ПП    11
02 ИП1   П2   F→   ПП   11
07 ИП1   ИП2  —    XY
11 С/П   П3   П4   П0
15 ИП4   3    —    FX<0 26
20 1     5    +    П4   КИП0
26 КИП4  ИП4  3    0    .  6  *  К[X]  П7
35 ИП0   3    6    5    .  2  5  *     К[X]  П8
45 ИП7   ИП8  ИП3  +    +  1  1  9     —     П1
55 7     /    К[X] П8   ИП1   ИП8  7  *  —  В/О
```

```
 0 GOSUB 11
 2 D1=D:PRINT N:GOSUB 11
 7 R=D-D1:PRINT N,R:STOP
11 INPUT K,M,G
15 IF M>=3 THEN 26
20 M=M+12:G=G-1
26 M=M+1:C=INT(M*30.6)
35 U=INT(G*365.25)
45 D=C+U+K-119
55 C=INT(D/7):N=D-C*7:RETURN
```

```
VAR D,D1,R,N:REAL;
PROCEDURE DNI
VAR K,M,G,C,U:REAL;
BEGIN
READLN(K,M,G);
IF M<3 THEN
  BEGIN M:=M+12;G:=G-1 END;
M:=M+1;C:=TRUNC(M*30.6);
U:=TRUNC(G*365.25);
D:=C+U+K-119;
C:=TRUNC(D/7);N:=D-C*7
END;
BEGIN
DNI;
D1:=D;WRITELN(N);DNI;
R:=D-D1;WRITELN(N,R)
END.
```

желтый цвет выбран для зеркальной буквы Г, в которую вписаны заголовок и конец цикла "пока" (см. программу 2.1);

зеленым окрашивается буква Т, в которую вписан заголовок альтернативы (см. программу 2.2);

синий цвет удобно использовать для опрокинутой буквы Г, в которую вписаны заголовок и конец цикла "до" (см. программу 2.4);

красный цвет соответствует положенной на бок букве П, в которую вписаны заголовок и конец цикла с параметром (см. программу 2.6).

Раскраска программы позволяет выявить заложенный в нее алгоритм, вложенность структур. Функциональные блоки программы можно уподобить бесцветным кирпичам, скрепленным друг с другом цветным раствором структурных управляющих конструкций. Многих поражает великолепие готических построек. Но мало кто знает, что во времена господства этого стиля форма и размер используемых камней были стандартными.

Управляющих конструкций немного, но их сочетания и возможность вложений позволяют строить самые сложные алгоритмические конструкции.

НОВАЯ МАШИНА – НОВЫЕ ВОЗМОЖНОСТИ

Пришло время расстаться с трилингвами – исчезнут программы для микрокалькуляторов. Во-первых, новые программы нельзя было бы уместить в прокрустово ложе памяти микрокалькулятора. Во-вторых, в них реализованы принципы и особенности, присущие языкам программирования высокого уровня.

ВЕЛИКОЛЕПНАЯ СЕМЕРКА ПРОГРАММИРОВАНИЯ

Мы немало говорили об алгоритмах решения, но почти не касались структур данных. "Виноват" в этом микрокалькулятор, поскольку на нем можно реализовать только два простейших типа данных: вещественные и целые величины, хранящиеся в регистрах памяти.

Швейцарский математик Николаус Вирт, написавший книгу "Алгоритмы + структуры = программы"*, коротко и емко обобщил в ее названии (в этой своеобразной формуле) сущность программирования.

---

* См.: *Вирт Н.* Алгоритмы + структуры = программы / Пер. с англ. – М.: Мир, 1985.

Языки программирования высокого уровня позволяют реализовывать и использовать более сложные структуры данных. Их удачный выбор – это, если вспомнить "формулу" Н. Вирта, наполовину решенная задача. Поясним утверждение конкретным примером.

Бейсик-программа 2.8 "Частотный словарь-1" укажет корректору разночтения в тексте и таким образом поможет выявить ошибки. Программа составляет частотный словарь введенного через клавиатуру ЭВМ текста и выводит содержимое словаря на дисплей в алфавитном или весовом порядке*. Так как одну и ту же опечатку практически невозможно сделать дважды, то просматривая слова, встречающиеся в тексте по одному разу (это, по существу, "хвост" частотного словаря, выводимого в весовом порядке), нетрудно обнаружить случайные опечатки.

Большую пользу эта программа способна оказать редактору. Если, например, в тексте из двухсот строк слово "компьютер" встречается двадцать раз, то имеет смысл воспользоваться синонимами: "ЭВМ", "машина" и т. д.

Запустив программу "Частотный словарь-1", требуется сообщить машине число строк в анализируемом тексте (первый оператор). Второй оператор резервирует в памяти ЭВМ место для хранения двух массивов литерных** переменных и одного массива целочисленных переменных.

Типы переменных в данном случае определяются стоящими за их именами специальными символами:

¤ – знак денежной единицы*** (свидетельствует о том, что данная переменная будет хранить цепочку литер);

% – знак процента (означает, что соответствующая переменная будет хранить целочисленное значение).

В Бейсик-программах можно встретить и два других знака:

– диез (отмечает вещественную переменную двойной точности);

! – восклицательный знак (соответствует вещественной переменной одинарной точности).

Руководствуясь принципом умолчания, восклицательным знаком обычно не пользуются – переменные, чьи идентификаторы не оканчиваются знаками, указывающими на тип, относятся к вещественным одинарной точности, т. е. к самому распространенному типу в программировании инженерных задач. Однако

---

* Весовым порядком будем считать такой, когда слова выстроены в зависимости от числа упоминаний в тексте.

** Нередко эти переменные встречаются в литературе под названиями – "символьные", "строковые", "стринговые".

*** На жаргоне программистов распространены также следующие названия этого знака: "клоп", "солнышко", "рубль". На машинах зарубежного производства используют знак $.

и в этом случае есть исключения: так, некоторые версии Бейсика по умолчанию работают с переменными двойной точности.

П р о г р а м м а 2.8

"Частотный словарь-1"
(семь структурных управляющих конструкций)

```
10 INPUT "ЧИСЛО СТРОК?";N:DIM S¤(N),W¤(5*N),W%(5*N)
11 FOR I=1 TO N:LINE INPUT S¤(I):NEXT:L=0
12 FOR I=1 TO N:REM ПЕРЕБОР СТРОК ДЛЯ АНАЛИЗА:
   S¤=S¤(I):H=0:K=0:REM ПЕРЕХОД К НАЧАЛУ СТРОКИ
20 K=K+1:B¤=MID¤(S¤,K,1):REM ПЕРЕБОР СИМВОЛОВ:
            IF B¤>="A" OR B¤="-" OR B¤="'" GOTO 20
30 IF K=H+1 THEN 80
40 W¤=MID¤(S¤,H+1,K-H-1):REM СЛОВО ВЫДЕЛЕНО:J=0
50 J=J+1:REM ПЕРЕБОР СЛОВ В СЛОВАРЕ:
          IF W¤<>W¤(J) AND J<=L GOTO 50
60 IF J=L+1 THEN
   REM СТАРОЕ СЛОВО:                    L=J:W¤(L)=W¤:
                                        W%(L)=1:GOTO 80
70 W%(J)=W%(J)+1:                       REM НОВОЕ СЛОВО:
   WHILE J>1 AND W%(J-1)<W%(J):
   GOSUB 150:J=J-1                 :WEND
80 H=K:REM ПЕРЕХОД К НОВОМУ СЛОВУ НА СТРОКЕ:
       IF K<LEN(S¤) GOTO 20:REM СТРОКА НЕ КОНЧИЛАСЬ
90 NEXT:REM КОНЕЦ ЦИКЛА ПЕРЕБОРА СТРОК:
   PRINT "1-ВЫВОД ПО АЛФАВИТУ,2-ПО ЧАСТОТЕ?";
100 B¤=INKEY¤:REM ПРИЕМ БАЙТА БЕЗ ПРЕРЫВАНИЯ:
                     IF B¤<"1" OR B¤>"2" GOTO 100
110 ON VAL(B¤) GOTO 120,140
120 FOR J=2 TO L:REM ЦИКЛ СОРТИРОВКИ ПО АЛФАВИТУ:
        IF W¤(J-1)>W¤(J) THEN GOSUB 150:GOTO 120
130 NEXT
140 FOR J=1 TO L:PRINT J". "W¤(J),W%(J):NEXT:END
150 SWAP W%(J),W%(J-1):SWAP W¤(J),W¤(J-1):RETURN
```

Итак, первой строкой Бейсик-программы "Частотный словарь-1" мы определили форму, в которой в памяти машины хранится исходный анализируемый текст (литерный массив S) и составляемый словарь (литерный массив W¤ – слова частотного словаря; целочисленный массив W% – частота встречи отдельных слов в тексте). В программе используются и другие переменные (целочисленные по сути, но вещественные по форме), так как за их именами не стоят знаки процента:

N – число строк в анализируемом тексте;
I – номер анализируемой строки текста;
L – число слов в частотном словаре;
J – номер слова в частотном словаре;
H – номер позиции, занимаемой пробелом перед словом, вычленяемым из строки текста;
K – номер позиции, занимаемой пробелом после слова, вычленяемого из строки текста.

Смысл переменных H и K поясним примером. Выделим из строки

Микрокалькулятор – младший брат ЭВМ

слово "*брат*". Отсчитывая буквы от начала фразы, получим, что H (начало слова) равно 27, а K (конец слова) – 32*. В программе 2.8 "Частотный словарь-1" слово вычленяет оператор, стоящий в 40-й программной строке, где записана литерная функция MID¤ (от англ. middle – середина).

Целочисленным по сути переменным N, I, L, J, H и K в программе "Частотный словарь-1" был присвоен статус вещественных. Здесь скрыта некоторая тонкость программирования. Целочисленная переменная по сравнению с вещественной занимает в памяти компьютера вдвое меньше места. В то же время, если простые переменные заменить на целочисленные, то за их именами придется проставить знак процента и увеличится объем памяти, отводимой на хранение текста программы. В результате замена простых вещественных переменных целочисленными оказывается неэффективной (как говорится, "скупой платит дважды"). Кроме того, листинг** программы "засоряется" знаками процента, что сказывается на наглядности.

Как правило, эффект экономии памяти от использования целочисленных переменных наблюдается лишь в том случае, если переменные объединены в достаточно большие массивы, как в программе "Частотный словарь-1". Вооружившись этим правилом, иногда полезно проделывать и обратную операцию – придавать вещественным по сути переменным статус целочисленных. Так, заменяя метры на миллиметры, рубли на копейки, тонны на граммы и т. д., удается избавиться от десятичных точек и хранить соответствующие переменные в целочисленных, а не в вещественных массивах, высвобождая память машины.

Не менее важным ресурсом компьютера является время исполнения программы. И здесь многое зависит от выбора пра-

---

* За начало и конец принимаем пробелы до и после слова.

** Листинг (от англ. listing – распечатка) – распечатка текста программы на принтере, в широком смысле – текст программы.

различных способа ввода информации в Бейсик-машину через клавиатуру:

ввод числа или цепочки символов, завершающийся нажатием клавиши "Ввод" (начало строки 10);

ввод цепочки символов, содержащей запятые, которые принимаются за знаки-разделители констант (второй оператор строки 11);

ввод байта без прерывания (начало строки 100) (незаменим в программировании динамических игр и удобен при ответе на запрос по меню: во-первых, блокируются все клавиши, кроме необходимых в данный момент (1 или 2); во-вторых, нет необходимости завершать ввод клавишей "Ввод").

В программе 2.8 "Частотный словарь-1" применяются два принципиально разных способа сортировки массивов: непосредственно в процессе заполнения массива (строка 70) и ранее заполненного (строки 120-130). В обоих случаях выбраны не самые быстрые, но самые короткие алгоритмы.

Структурные конструкции, состоящие всего из одного оператора (строки 11 и 140), не выделены рамками структурной диаграммы.

Частотный словарь сортируется (строки 120-130) в соответствии с алфавитом, который "зашит" в постоянное запоминающее устройство компьютера. Большинство ЭВМ оперируют весьма своеобразным русским алфавитом, начинающимся с буквы Ю и заканчивающимся буквой Ч, что связано с особенностями клавиатуры: каждая клавиша компьютера имеет различные назначения в зависимости от выбранного регистра (например, C и Ц, F и Ф, R и Р). Алфавит машины определяется тем, какие коды присвоены латинским и русским буквам. Русские буквы, по капризу разработчиков ЭВМ, получили коды в несколько отличном от алфавитного порядке.

Второй оператор строки 10 задает максимальные размеры массивов. Размер выбирается в соответствии с гипотезой, что число слов в словаре не превышает пятикратного числа строк анализируемого текста. Особое внимание необходимо уделить минимальным индексам массивов, которые судя по логике работы программы "Частотный словарь-1" и по заголовкам циклов с параметром (строки 11 и 12) должны быть равны единице. Но это не так. Необходимо зарезервировать в памяти машины область для нулевых элементов, иначе выполнение программы будет прервано на строке 70. Дело в том, что записанная в ней операция логического умножения излишне педантична: второе логическое условие проверяется и тогда, когда первое неверно и происходит обращение к элементу массива с нулевым номером.

Все номера строк программы, за исключением 10 и 11, выполняют функцию меток. Номер 10 – дань особенностям

Бейсика, поскольку первая строка программы обязательно начинается с номера-метки. Число 11 открывает вторую строку для того, чтобы при необходимости можно было запустить программу не с начала (очень удобное качество языков-интерпретаторов). Например, если после просмотра словаря требуется исправить ошибки и опечатки в ранее введенном тексте, достаточно повторно запустить программу, но не с самого начала (командой RUN), а со строки 11 (командами RUN 11 или GOTO 11). Оператор ввода, записанный в теле цикла (строка 11) не только запрашивает элементы литерного массива S, но и выводит на дисплей их прежние значения, позволяя внести исправления и дополнения.

Программа 2.8 "Частотный словарь-1" имеет ряд недочетов*. Назовем лишь два основных:

слово текста, введенное в ЭВМ с переносом на следующую строку, машина принимает за два самостоятельных;

одно и то же слово, использованное в разных грамматических формах или содержащее одновременно прописные и строчные буквы, машина также принимает за отдельные слова.

Оформляя программу, четко оговаривают принятые допущения, чтобы пользователь имел ясное представление о ее возможностях.

МОЖНО ЛИ ИГРАТЬ НА СКРИПКЕ С ОДНОЙ СТРУНОЙ?

Мы сравнивали программирование с игрой на семиструнной гитаре. Попробуем выяснить, какие из семи структурных управляющих конструкций следует считать основными, а какие – вспомогательными, без которых нетрудно обойтись. А может быть, напротив, и семи недостаточно? Так, вместо цикла "пока" в программе 2.1 "Метод секущих" и вместо цикла "до" в программе 2.4 "Метод Ньютона" подошел бы цикл с выходом из середины. В этом цикле высчитывается очередное приближение к корню алгебраического уравнения (строка 27 в программе 2.1), а затем проверяется, достигнута ли заданная точность расчета E. Вслед за тем идет подготовка к новому приближению (строка 35 в программе 2.1).

И все же на практике цикл с выходом из середины встречается редко. На языке Паскаль, например, такой цикл можно реализовать только с опорой на метку, а это, как уже отмеча-

* Они в какой-то мере свойственны всем программам, приведенным во второй главе. Невозможно предусмотреть все особенности русского языка в программе, предназначенной лишь для того, чтобы иллюстрировать нюансы использования тех или иных операций. В то же время программа, которая была бы свободна от недостатков, не уместилась бы на страницах книги.

вильной структуры данных. Так, в программе 2.8 "Частотный словарь-1" за заголовком цикла с параметром (строка 12) следует оператор, пересылающий элемент литерного массива S в одноименную простую переменную, которая затем обрабатывается в теле цикла. Этот прием, называемый "чистка цикла", существенно ускоряет расчет, так как обращение к простой переменной выполняется быстрее, чем к индексной.

Переходя от структуры данных к алгоритму, которые "в сумме", если вспомнить формулу Н. Вирта, составляют программу, можно заметить, что в программе "Частотный словарь-1" использованы все семь управляющих конструкций: цикл с параметром, циклы "пока" и "до", альтернатива, множественное ветвление, вызов процедуры и переход к метке.

Гвоздь программы "Частотный словарь-1" – строка 20, отыскивающая знаки препинания и пробелы, которые разделяют слова текста. Ее действие основано на том, что коды этих символов меньше, чем код заглавной латинской буквы A. Знаки дефис "-" и апостроф "'", которыми иногда заменяют твердый знак, отсутствующий на клавиатуре некоторых ЭВМ, также приравниваются к буквам. В программе за это отвечает операция логического сложения в строке 20.

Сравнивая программирование с игрой на семиструнной гитаре, структурные управляющие конструкции можно уподобить ее струнам. Что же рождает мелодию "Частотный словарь-1"?

1. Цикл "пока". Сортировка словаря в порядке убывания числа одинаковых слов в тексте (строка 70 без первого оператора). Проводить такую сортировку полезно даже в том случае, если составленный словарь не придется выводить на печать в алфавитном порядке. Периодическая сортировка в порядке убывания ускорит работу программы: если какое-то слово часто встречалось в первой половине текста, то вероятно та же тенденция сохранится дальше. Очередное слово будет сравниваться с набором слов частотного словаря начиная с тех слов, что чаще других встречаются в тексте.

2. Альтернатива. Неполная альтернатива игнорирует промежутки между словами, когда встречаются два и более знака препинания или пробела (строки 30-70). Полная альтернатива предназначена для уточнения – встречалось ли ранее в словаре только что выделенное слово (строки 60-70). Если "да" (строка 70), то счетчик этого слова увеличивается на единицу, а соответствующие элементы массива по-возможности передвигаются в начало словаря (переход к п. *1*). Если слово не встречалось (конец строки 60), словарь расширяется на одну позицию для хранения нового слова из текста.

3. Множественное ветвление после ответа на запрос меню (строки 110-130). Казалось бы, можно обойтись простой неполной

альтернативой с пустым правым плечом, но режимов вывода словаря может быть больше двух (например, вывод на принтер, "сброс" на магнитный диск, на перфоленту и т. д.). Из нескольких равноценных приемов выбирают тот, который при прочих равных параметрах позволяет модернизировать и расширять программу.

4. Цикл "до". В программе он встречается несколько раз, выполняя различные функции: поиск в очередной строке анализируемого текста знаков-разделителей (строка 20); перебор слов в частотном словаре и сравнение их с только что выделенным (строка 50); перебор слов очередной строки (программные строки 20-80); опрос клавиатуры в ожидании ответа человека на запрос по меню (строка 100).

5. Цикл с параметром. Хотя строки 120 и 130 несут формальные признаки цикла с параметром, но тем не менее они им не являются. Цикл с параметром осуществляет: ввод исходного анализируемого текста (строка 11 без последнего оператора); перебор строк при анализе (строки 12-90); распечатка частотного словаря (строка 140 без последнего оператора).

6. Вызов процедуры (подпрограммы) перестановки соседних слов частотного словаря. Перестановку, как правило, используют в том случае, когда какой-то функциональный блок встречается в программе более одного раза. В программе 2.8 перестановка необходима для сортировки словаря по частоте (строка 70) и по алфавиту (строка 120). Для этого служит подпрограмма (строка 150).

7. Переход к метке. В этой программе он используется при досрочном прерывании цикла с параметром и возвращении к его началу при алфавитной сортировке (строка 120). В остальных случаях переходы к меткам реализуют одну из стандартных управляющих конструкций (пп. *2 – 4* и *6*). В строках 120 и 130 переход к метке "ломает" цикл с параметром (п. *5*).

Метка не случайно оказалась последней в списке. Структурные диаграммы допускают переход к метке, но лишь в пределах функционального блока, ограниченного прямоугольными рамками без "внутренних перегородок" (строки 120, 130).

Иногда приходится слышать, что структурированная программа – программа без меток. Это не совсем верно: в таких программах переходы к меткам реализованы с помощью стандартных управляющих конструкций. При желании структурированную программу можно написать на Бейсике (см. программу 2.8), а неструктурированную – на Паскале. Так что исключение меток из программы – вовсе не самоцель.

Программа 2.8 "Частотный словарь-1" обладает несколькими существенными особенностями. Так, в ней реализованы три

лось, не красит программу. Цикл с выходом из середины легко заменить на цикл "пока" или цикл "до". Более того, нетрудно доказать, что из семи управляющих конструкций, проиллюстрированных программой 2.8 "Частотный словарь-1", в арсенале программиста достаточно оставить только одну – цикл "пока".

Программа 2.9 "Частотный словарь-2" использует только циклы "пока".

П р о г р а м м а 2.9

"Частотный словарь-2"
(использование только цикла "пока")

```
10 INPUT"ЧИСЛО СТРОК?";N:DIM S¤(N),W¤(5*N),W%(5*N):L=0
11 I=0:WHILE I<N:I=I+1:LINE INPUT S¤(I):WEND:I=0
12 WHILE I<N:REM ПЕРЕБОР СТРОК ДЛЯ АНАЛИЗА
13 I=I+1:S¤=S¤(I):H=0:K=0:S=LEN¤(S¤)
20 WHILE K<=S
21 B¤="A":REM ИНАЧЕ НЕ ВОЙДЕШЬ В ЦИКЛ
22 WHILE B¤>="A" OR B¤="-" OR B¤="'"
23 K=K+1:B¤=MID¤(S¤,K,1)          :WEND
24 P1=1:REM ИМИТАЦИЯ АЛЬТЕРНАТИВЫ
30 WHILE K>H+1 AND P1
40 P1=0:W¤=MID¤(S¤,H+1,K-H-1):J=1:P2=1
50 WHILE W¤<>W¤(J) AND J<=L
51 J=J+1:                 WEND
60 WHILE J>L AND P2
61 P2=0:L=J:W¤(L)=W¤:W%(L)=1:  WEND
70 WHILE P2
71 P2=0:W%(J)=W%(J)+1
72 WHILE J>1 AND W%(J-1)<W%(J)
73 SWAP W¤(J-1),W¤(J)
74 SWAP W%(J-1),W%(J):J=J-1:  WEND  :WEND  :WEND
80 H=K:REM ПЕРЕХОД К НОВОМУ СЛОВУ НА СТРОКЕ:  WEND  :WEND
90 PRINT "1-ВЫВОД ПО АЛФАВИТУ,2-ПО ЧАСТОТЕ?";:B¤="3":P3=1
100 WHILE B¤<"1" OR B¤>"2"
101 B¤=INKEY¤          :WEND
110 WHILE B¤="1" AND P3
111 P3=0:P4=1
112 WHILE P4
120 P4=0:J=1:REM ЦИКЛ СОРТИРОВКИ
121 WHILE J<L
122 J=J+1:P5=1
123 WHILE W¤(J-1)>W¤(J) AND P5
124 P5=0:SWAP W¤(J-1),W¤(J)
125 SWAP W%(J-1),W%(J):P4=1  :WEND  :WEND  :WEND  :WEND
140 J=0:WHILE J<L:J=J+1:PRINT J"."W¤(J) W%(J):WEND
```

Данная конструкция является универсальной, ею можно заменить остальные:

цикл с параметром – это цикл "пока" со встроенным счетчиком (сравните строки 11, 12-90 и 140 программ 2.8 и 2.9);

цикл "до" отличается от цикла "пока" лишь тем, что тело цикла "до" выполняется как минимум один раз, в то время, как тело цикла "пока" может не выполниться ни разу (встречаются, однако, ситуации, когда логическое выражение, управляющее циклом, приобретает смысл только после выполнения тела цикла (строки 20 и 100 программы 2.8); такие "до"-циклы заменить сложнее: приходится, например, искусственно создавать ситуацию, когда выполняются логические условия, управляющие циклом "пока" (строки 1-23 и 90-101 программы 2.9));

полную альтернативу при желании нетрудно заменить двумя неполными, логические заголовки которых имеют противоположный знак (так, кстати, поступают, исключая метки в программах на языке Фортран);

неполная альтернатива есть ничто иное, как цикл "пока", тело которого выполняется один раз или ни разу (такое своеобразное поведение цикла "пока" легко смоделировать: достаточно ввести в программу дополнительную булеву переменную; она должна изменять свое значение на противоположное после первого выполнения тела цикла (строки 60-80 программы 2.9));

множественное ветвление – это серия вложенных альтернатив: если A=1, то управление переходит на первый функциональный блок, иначе, если A=2, то на второй и т. д. (альтернативу, кстати, несложно свести к двум циклам "пока");

переход к метке (строка 20 программы 2.8) является либо признаком незавершенности программы, либо вариантом ускорения ее работы (когда становится ясно, что выполнять цикл далее не следует, его обрывают переходом к метке; известны два основных способа исключения меток*: дублирование операции и использование признака; второй способ применен в программе 2.9 "Частотный словарь-2": цикл проверки расстановки слов по алфавиту (строки 123-125) не прерывается, но любая перестановка соседних строк в процессе сортировки изменяет значение булевой переменной P4, что служит сигналом к повторению цикла сортировки (строка 112));

от подпрограммы (строка 150) программы 2.8 "Частотный словарь-1" избавиться весьма просто (см. программу 2.9 "Частотный словарь-2"): достаточно переставить две соседние записи в словаре и продублировать в двух местах (строки 73-74 и 124-125) (впрочем, подпрограмма (процедура) – это не только

---

* См.: Ван Тассел Д. Стиль, разработка, эффективность и испытание программ / Пер. с англ. – М.: Мир, 1985.

Программа 2.10

"Покоординатный спуск" (использование локальных переменных в процедуре)

```
VAR H,K,X:ARRAY[10] OF REAL;
    J,N:INTEGER;A,B,E,S,X0,X9:REAL;
PROCEDURE ЗОЛОТОЕ_СЕЧЕНИЕ;VAR X1,X2,Y,Y1,Y2:REAL;
  BEGIN ЛЕВЫЙ;ПРАВЫЙ;
    REPEAT
      IF Y1>Y2
       THEN BEGIN A:=X1;X1:=X2;Y1:=Y2;ПРАВЫЙ END
       ELSE BEGIN B:=X2;X2:=X1;Y2:=Y1;ЛЕВЫЙ END
    UNTIL ABS(A-B)<=E/N;
    X0:=(A+B)/2    (*УТОЧНЕНИЕ МИНИМУМА
  END;(*ПРОЦЕДУРЫ ЗОЛОТОЕ_СЕЧЕНИЕ
PROCEDURE ЛЕВЫЙ;
   BEGIN X1:=.618*A+.382*B;X[J]:=X1;FNY;Y1:=Y END;
PROCEDURE ПРАВЫЙ;
   BEGIN X2:=.618*B+.382*A;X[J]:=X2;FNY;Y2:=Y END;
PROCEDURE FNY;BEGIN Y:=FNY(X[1],...,X[N]) END;
BEGIN (*НАЧАЛО ПРОГРАММЫ
   WRITE('ЧИСЛО АРГУМЕНТОВ,ТОЧНОСТЬ');READLN(N,E);
   FOR J:=1 TO N DO BEGIN
    WRITE('X H',J);READLN(H[J]);
    WRITE('X K',J);READLN(K[J]);X[J]:=(H[J]+K[J])/2
   END;(*ЦИКЛА ВВОДА ДИАПАЗОНОВ
   REPEAT S:=0;
      FOR J:=1 TO N DO BEGIN
         X9:=X[J];A:=H[J];B:=K[J];ЗОЛОТОЕ_СЕЧЕНИЕ;
         X[J]:=X0;S:=S+SQR(X9-X0)
      END
   UNTIL SQRT(S)<=E;
   FOR J:=1 TO N DO WRITELN('X ОПТ.',J,'=',X[J])
END.
```

```
(* 10 DIM H(10),K(10),X(10):GOTO 170                          *)
(* 20 REM ПОДПРОГРАММА 'ЗОЛОТОЕ СЕЧЕНИЕ'                      *)
(* 30 GOSUB 120:GOSUB 140                                     *)
(* 40 REM НАЧАЛО ИТЕРАЦИИ                                     *)
(* 50 IF Y1<=Y2 THEN 70                                       *)
(* 60 A=X1:X1=X2:Y1=Y2:GOSUB 140:GOTO 80                      *)
(* 70 B=X2:X2=X1:Y2=Y1:GOSUB 120                              *)
(* 80 IF ABS(A-B)>E/N GOTO 50                                 *)
   90 X0=(A+B)/2                                               *)
   100 RETURN                                                  *)
(* 110 REM ПОДПРОГРАММА 'ЛЕВЫЙ'                               *)
(* 120 X1=.618*A+.382*B:X(J)=X1:GOSUB 150:Y1=Y:RETURN*)
(* 130 REM ПОДПРОГРАММА 'ПРАВЫЙ'                              *)
(* 140 X2=.618*B+.382*A:X(J)=X2:GOSUB 150:Y2=Y:RETURN*)
(* 150 Y=FNY(X(1),...,X(N)):RETURN                            *)
   160 REM НАЧАЛО ПРОГРАММЫ                                    *)
(* 170 INPUT'ЧИСЛО АРГУМЕНТОВ,ТОЧНОСТЬ?';N,E                  *)
(* 180 FOR J=1 TO N                                           *)
(* 190 PRINT'X H'J;:INPUT H(J)                                *)
(* 200 PRINT'X K'J;:INPUT K(J):X(J)=(H(J)+K(J))/2             *)
   210 NEXT J                                                  *)
(* 220 S=0                                                    *)
(* 230 FOR J=1 TO N                                           *)
(* 240 X9=X(J):A=H(J):B=K(J):GOSUB 30                         *)
(* 250 X(J)=X0:S=S+(X9-X0)*(X9-X0)                            *)
(* 260 NEXT J                                                 *)
(* 270 IF SQR(S)>E GOTO 220                                   *)
(* 280 FOR J=1 TO N:PRINT'X ОПТ.'J'='X(J):NEXT J              *)
```

средство избавить программиста от многократного ввода одного и того же участка программы, а машину от неэкономного расходования памяти на их хранение; процедура имеет и другие полезные свойства, поэтому ее вместе с циклом "пока" иногда причисляют к основным структурным управляющим конструкциям).

Итак, следует говорить о двух основных структурных управляющих конструкциях. Программа 2.10 "Покоординатный спуск" и программа 2.11 "Числа Фибоначчи-П*" подтверждают такое толкование основной структурной теоремы.

П р о г р а м м а 2.11

"Числа Фибоначчи-П"
(единство формы и содержание)

```
F
U
NC
TIO
N FIB
O(N:INTE
GER):INTEGER;
BEGINIFN<3THENFIBO:=1
ELSEFIBO:=FIBO(N-2)+FIBO(N-1);END;
(***ФУНКЦИЯ ОПРЕДЕЛЕНИЯ ЧИСЕЛ ФИБОНАЧЧИ С РЕКУРСИЕЙ***)
```

ЕЩЕ РАЗ О ПОДПРОГРАММАХ И ПРОЦЕДУРАХ

На заре эры программирования почти все программы разрабатывали по технологии, которую сейчас называют технологией "снизу–вверх": один программист писал программу от начала до конца, включая разработку подпрограмм низшего уровня. По мере усложнения задач, решаемых на ЭВМ, технологию "снизу–вверх" стали использовать все реже, поскольку она требовала слишком больших затрат времени, сил и средств. Процесс программирования попытались распараллелить, разбивая программу на независимые подпрограммы (модули, блоки, процедуры), которые можно создавать и отлаживать отдельно. Новая технология получила название "сверху–вниз".

* В заголовке программы П означает, что программа написана на Паскале.

В распространенных версиях Бейсика нет подпрограмм как таковых. Оператор GOSUB фактически служит не для вызова подпрограммы, а для перехода к метке, когда транслятор запоминает, откуда был сделан переход и куда следует вернуться, встретив слово RETURN. Это одна из причин, почему многие программисты считают Бейсик "несерьезным" языком программирования. Тем не менее подавляющее большинство программ (по числу, а не по объему) до сих пор пишутся на Бейсике. Основную лепту вносят специалисты-непрограммисты, для которых программирование – хобби или средство решения отдельных прикладных задач. В результате системные программисты совершенствуют Бейсик и создают его новые версии (например, Турбо-Бейсик), допускающие модульное программирование.

Процедура языка Паскаль зависит от программы в меньшей степени, нежели подпрограмма на Бейсике. Процедура может иметь свои собственные локальные переменные, функции и метки, которые, если и совпадут по имени со своими аналогами в основной программе или в других процедурах (независимых частях программы), то с ними не смешаются. Это позволяет программисту, разрабатывающему какую-либо процедуру, не вникать в тонкости других. Ему достаточно знать то небольшое число так называемых глобальных переменных, часть из которых получит исходные значения перед вызовом процедуры, а остальные будут хранить результат выполнения процедуры.

Программа 2.10 "Покоординатный спуск-П" для поиска минимума многомерной функции содержит четыре процедуры: ЗОЛОТОЕ СЕЧЕНИЕ, ЛЕВЫЙ, ПРАВЫЙ и FNY.

Процедура FNY – ядро программы. Результат выполнения этой процедуры – значение анализируемой функции, аргументами которой являются элементы вещественного массива X.

При покоординатном спуске минимизация сводится к серии одномерных оптимизаций. Для этого используют процедуру ЗОЛОТОЕ СЕЧЕНИЕ (см. программу 2.5) до тех пор, пока два последовательных приближения не будут отстоять друг от друга менее чем на заданную точность E. Переменная X1 в программе "Покоординатный спуск-П" имеет двойной характер: объявлена глобальной в заголовке программы и в то же время локальной в заголовке процедуры ЗОЛОТОЕ СЕЧЕНИЕ. В программе 2.10 "Покоординатный спуск-Б"*, записанной в комментариях Паскаль-программы, глобальной переменной X1 соответствует переменная X. Это связано с тем, что на Паскале нельзя давать переменным разного типа одни и те же имена: первым оператором в программе уже объявлен вещественный массив с именем X. В Бейсик-программах переменные-тезки различного типа допустимы, хотя и не во всех версиях.

---

* В заголовке программы Б означает, что программа написана на Бейсике.

В ПОГОНЕ ЗА СОБСТВЕННЫМ ХВОСТОМ

Сравнивая программы "Частотный словарь-1" и "Частотный словарь-2", легко убедиться, что алгоритм любой сложности можно реализовать всего двумя конструкциями: следование (последовательное выполнение функциональных блоков) и повторение (цикл "пока", тело которого – функциональный блок). Но, оказывается, есть алгоритмы, в которых вызов процедуры – это незаменимая структурная конструкция. Такой алгоритм не удается изобразить графически: не годятся ни блок-схема, ни структурная диаграмма. Подобные алгоритмы называют рекурсивными – процедура вызывает сама себя.

Классический пример рекурсивной процедуры – Паскаль-функция* генерации чисел Фибоначчи**. Программа 2.11 "Числа Фибоначчи-П" выглядит необычно. Расположение текста подчеркивает единство формы и содержания: число знаков в строках – это первые 11 чисел Фибоначчи (включая первое, равное нулю). Кроме того, программа наглядно демонстрирует тот факт, что понятие строки на языке Паскаль не играет никакой смысловой роли. Программисты, долгое время работавшие на Бейсике, где строка – это законченная конструкция, переходя к Паскалю, иной раз допускают ошибки, например:

```
IF A=B THEN C:=2; E:=3;
```

Если A действительно равно B, то C становится равно 2, а E = 3. Различие значений A и B приводит к тому, что второй из операторов будет выполнен, если речь идет о Паскаль-программе, хотя в Бейсик-программе этого бы не произошло.

КРОЛИКИ И МИНИМУМЫ

Программу рекурсивного вычисления чисел Фибоначчи можно встретить почти в каждом учебнике по языку Паскаль. Однако рекурсия здесь напоминает пушку, из которой бьют по воробьям. Для расчета было бы эффективнее (в смысле экономии главных ресурсов ЭВМ – времени работы и занимаемой памяти) воспользоваться не рекурсивным (см. программу 2.11), а итеративным алгоритмом, который использован в программе 2.12 "Числа Фибоначчи-Б".

---

* Функция – это особый вид процедуры.

** В соответствии с определением каждое новое число Фибоначчи за исключением двух первых, равно сумме двух предыдущих.

Программа 2.12

"Числа Фибоначчи-Б"
(переход от рекурсивного к итеративному алгоритму)

```
0 DEF FNY(X)=...'ВИД АНАЛИЗИРУЕМОЙ ФУНКЦИИ
1 CLS:?"ПОИСК МИНИМУМА ФУНКЦИИ В ИНТЕРВАЛЕ [A,B]
1 -Й ШАГ - РЕДАКТИРОВАНИЕ СТРОКИ 0, 2-Й ШАГ - RUN"
2 LIST 0:INPUT"A,B,ЧИСЛО ПРИБЛИЖЕНИЙ?";A,B,N
3 F1=0:F2=1'НУЛЕВОЕ И 1-Е ЧИСЛО ФИБОНАЧЧИ
5 FOR I=2 TO N:F3=F1+F2:F1=F2:F2=F3:NEXT
8 X1=A+(B-A)*F1/F2:Y1=FNY(X1)
13 FOR I=1 TO N'*********************************
21 X2=A+B-X1:Y2=FNY(X2)'                          *
34 IF X1<X2 AND Y1<Y2 THEN B=X2:GOTO 233'         *
55 IF X1<X2 THEN A=X1:X1=X2:Y1=Y2:GOTO 233'       *
89 IF Y1<Y2 THEN A=X2:GOTO 233'                   *
144 B=X1:X1=X2:Y1=Y2'                             *
233 NEXT'****************************************
377 X=(A+B)/2:?"Y МИН="FNY(X)" ПРИ X="X
610 END' 15-Е ЧИСЛО ФИБОНАЧЧИ
```

Леонардо Пизанский (1180 – 1240), более известный под именем Фибоначчи, в 1202 г. описал одну из первых моделей развития замкнутой биологической системы, населенной "условными кроликами"*. Если соответствующим образом определить их плодовитость и продолжительность жизни, то численность популяции кроликов из поколения в поколение будет меняться по строгому закону, характеризуемому числами Фибоначчи:

Поколение	0	1	2	3	4	5	6	7	...
Число кроликов	0	1	1	2	3	5	8	13	...

Со временем про кроликов Фибоначчи забыли, но его числа в наше время нашли неожиданное применение в прикладной математике. Они, подобно "золотому соотношению", позволяют так делить интервал неопределенности при поиске оптимума одномерной унимодальной функции (строки 8 и 21 в программе 2.5), что при каждом новом приближении достаточно рассчитать только одно ее значение, второе – уже известно из предыдущего приближения.

* Кроликов первоначально не было, затем в клетку посадили одного, затем другого, у них появилось потомство, первый кролик умер и т. д.

Заметим, что в программе "Числа Фибоначчи-Б" номера строк – это первые 16 чисел Фибоначчи (единство формы и содержания).

Сравнивая Паскаль-функцию "Числа Фибоначчи-П" и строки 3 и 5 программы "Числа Фибоначчи-Б", можно отметить, что итеративный метод здесь оказался эффективнее в смысле затрат памяти машины и времени на расчет. Реализация рекурсивных алгоритмов требует создавать, а затем ликвидировать "невидимые" пользователю локальные переменные, чтобы сохранить параметры процедуры при нисходящем расчете.

Рекурсия – это способ внешне упростить программу, сделать ее более наглядной. Трудности, связанные с увеличением сложности расчетов, выпадают на долю машины. Фактически, чтобы решить вопрос о том, является ли вызов процедуры основной структурной управляющей конструкцией, необходимо доказать, можно ли любой рекурсивный алгоритм преобразовать в итеративный, как это было с алгоритмом поиска чисел Фибоначчи.

В любом случае рекурсия останется одним из мощных средств программирования. В качестве доказательства этого предлагаем составить итеративную программу поиска значений целочисленной двумерной функции Аккермана, рекурсивная запись которой весьма компактна:

```
A(0, N) = N + 1,
A(M, 0) = (M-1, 1),
A(M, N) = A(M-1, A(M, N-1)).
```

Впрочем, рекурсия предназначена лишь для удобства программиста: любой транслятор, переводящий программу с естественного языка на язык, понятный машине, преобразует рекурсивные алгоритмы в итеративные.

В итоге вопрос о том, является ли вызов процедуры основной или вспомогательной конструкцией, заданный в начале раздела, остается в этой книге без ответа.

КОГДА ЗАДАЧУ РЕШАЮТ С КОНЦА

Мы уже рассказывали в первой главе об игре "Ханойская башня". Говоря о рекурсии, нельзя не вспомнить об этой интересной задаче. Напомним ее условия.

В нашем распоряжении 64 кольца. Вначале все кольца собраны на одном стержне (назовем его *A*). Кольца различны по размерам и расположены как в детской пирамидке, причем

кольцо большего диаметра находится внизу. Требуется по одному переложить кольца на третий стержень *C*, используя промежуточный *B*. Не допускается, чтобы большее кольцо оказывалось на меньшем.

Если колец всего два, задача решается просто:

$$A \to B,\ A \to C,\ B \to A.$$

Три кольца можно переместить следующим образом:

$$A \to C,\ A \to B,\ C \to B,\ A \to C,\ B \to A,\ B \to C \text{ и, наконец, } A \to C.$$

Чем больше колец на стержне, тем легче запутаться: часто оказывается, что какая-то очередная позиция уже встречалась. Ясно, что без определенной стратегии с задачей не справиться. Вот тут-то и приходит на выручку рекурсия, позволяющая решать задачу не с начала, а с конца.

Решая задачу, мы прийдем к ситуации, когда самое большое кольцо можно переместить со стержня *A* на *C*. При этом вся остальная пирамида должна находиться на стержне *B*. Получилась более простая задача – переложить меньшую пирамиду со стержня *B* на *C*. При этом уже стержень *A* используется как промежуточный. Если неполную пирамиду считать за одно кольцо, то задача превращается в задачу о трех кольцах.

Если *N* – число колец в пирамиде, то решение разбивается на три этапа, необходимо переложить:

N-1 колец с *A* на *B*;

самое большое кольцо с *A* на *C*;

пирамиду из *N*-1 колец с *B* на *C*.

Легко заметить, что третий этап – это уменьшенная копия всей задачи. Изменилось лишь *N* и маркировка стержней.

Для решения задачи на ЭВМ необходим язык программирования, допускающий рекурсию, когда процедура вызывает сама себя еще до того, как ее полностью определят. В таких популярных языках программирования, как Бейсик (простые версии) и Фортран, рекурсии запрещены. Точнее, в описаниях не сказано, что подпрограммы можно вызывать рекурсивно. Но, как говорится, если нельзя, но очень хочется, то можно: рекурсия допустима и на Бейсике.

В своей программе Фам Ван Чиен из Вьетнама решает задачу о ханойской башне на версии Бейсика, используемой на машинах WANG-2200 и "Искра 226" (программа 2.13). В этой версии есть подпрограмма с параметрами, вызываемая по ее номеру (первый оператор строки 30). В подпрограмме (строки 40-130) дважды встречается ее рекурсивный вызов: строки 30 и 110. При рекурсивном решении задачи о ханойской башне про-

блема N колец сводится к проблеме N-1 колец до тех пор, пока задача не станет тривиальной при одном кольце. В этом случае его следует переложить со стержня A на C – правое плечо альтернативы подпрограммы (строка 70).

Программа 2.13

"Ханойская башня-1"
(использование рекурсивной подпрограммы)

```
10 DIM A¤1,B¤1,C¤1
20 INPUT     'ЧИСЛО КОЛЕЦ'    ,N
30 GOSUB ' 1(N,"A","B","C"):PRINT :END

40 DEFFN ' 1(N,X¤,Y¤,Z¤)
50 A¤=X¤:B¤=Y¤:C¤=Z¤
60 IF N>1 THEN 80
70                                  PRINT A¤;"->";C¤;" ";
                                    :GOTO 120
80 N1=N:GOSUB ' 1(N1-1,A¤,C¤,B¤)
90 PRINT A¤;"->";B¤;" ";
100 GOSUB ' 1(N1-1,C¤,A¤,B¤)
110 N1=N1+1:A1¤=A¤:A¤=B¤:B¤=A1¤
120 RETURN
```

И все же, решая задачу, мы нарушили запрет, касающийся использования рекурсии в Бейсике. Важно четко знать, к каким последствиям это может привести. Представьте себе, что в микрокалькуляторе есть встроенная функция, которая, вычисляя синус x, изменяет значения, хранящиеся в других ячейках памяти. Таким микрокалькулятором никто, конечно, пользоваться не будет и без всяких запретов. Рекурсивный вызов подпрограммы на Бейсике для "Искры 226" также приводит к неприятному побочному эффекту – изменяется входной параметр подпрограммы. Если не восстановить его в прежнем значении (первый оператор строки 80), то результат расчета окажется неверным. Подобный эффект наблюдается и в некоторых некачественных трансляторах языка Паскаль.

Вот еще одно соображение, поясняющее причину запрета рекурсии на Бейсике и возможные ограничения ее использования на Паскале.

Если запустить коротенькую, из одного оператора, Бейсик-программу

```
1 GOSUB 1 ,
```

которая по смыслу напоминает древний символ алхимиков "змея, поедающая свой хвост", то компьютер вскоре выдаст на экран сообщение об аварийном останове: Память переполнена. Выполняя оператор GOSUB, машина каждый раз фиксирует точку, откуда сделан переход к подпрограмме, а отведенная для этого память не так велика. Кроме того, некоторые версии Бейсика и Паскаля ограничивают количество уровней подпрограмм не объемом свободной памяти, а каким-либо числом: 16, 32 и т. д.

На рис. 2.1 показана распечатка результатов, полученная при работе программы "Ханойская башня-Б" при $N=7$. Решение требует довольно много времени: в общем случае для перестановки пирамиды из N колец требуется 2^N-1 перестановок, т. е. затраты времени увеличиваются по экспоненте.

```
RUN
   ЧИСЛО КОЛЕЦ? 7
A->C A->B C->B A->C B->A B->C A->C A->B C->B C->A B->A C->B
A->C A->B C->B A->C B->A B->C A->C B->A C->B C->A B->A B->C
A->C A->B C->B A->C B->A B->C A->C A->B C->B C->A B->A C->B
A->C A->B C->B C->A B->A B->C A->C B->A C->B C->A B->A C->B
A->C A->B C->B A->C B->A B->C A->C A->B C->B C->A B->A C->B
A->C A->B C->B A->C B->A B->C A->C B->A C->B C->A B->A B->C
A->C A->B C->B A->C B->A B->C A->C B->A C->B C->A B->A C->B
A->C A->B C->B C->A B->A B->C A->C B->A C->B C->A B->A B->C
A->C A->B C->B A->C B->A B->C A->C A->B C->B C->A B->A C->B
A->C A->B C->B A->C B->A B->C A->C B->A C->B C->A B->A B->C
A->C A->B C->B A->C B->A B->C A->C
END PROGRAM
FREE SPACE- 65176
```

Рис. 2.1. Протокол игры "Ханойская башня"

Кстати, если составить алгоритм решения задачи о ханойской башне не так уж сложно, то решить ее даже для небольших N трудно или даже невозможно.

Самые современные машины способны вычислить одно перемещение за одну миллионную долю секунды. Даже при такой скорости вычислений для расчета переноса башни из 64 колец потребуется около миллиона лет*.

Программу 2.13 можно рассматривать как своего рода рекордную по времени, затрачиваемому на решение поставленной задачи. Экономия памяти машины и выигрыш в быстродействии достигаются переходом к итеративному алгоритму, реализованному билингвой 2.14 Бейсик-Фортран, автор которой Н. Саква.

---

* В самом деле. $2^{64} - 1 = 1.845 \cdot 10^{19}$. хотя вычитание единицы можно и не учитывать.

Программа 2.14

"Ханойская башня-2"
(отказ от рекурсии)

```
1  'БЕЙСИК
2  DIM M(64)'
3  FOR I=1 TO 64:M(I)=1:NEXT'
4  PRINT     "ЧИСЛО КОЛЕЦ"'
5  INPUT N'
6  FOR I=1 TO N'
7  IF I>1 AND M(I)=M(1) GOTO 13'
8  K=M(I)+I+N-2*INT((I+N)/2)+1'
9  IF K>3 THEN K=K-3'
10 PRINT M(I),"->",K'
11 M(I)=K'
12 IF I>1 GOTO 6'
13 NEXT'
14 STOP'
15 '
16 '
17 '
18 END'
```

```
C ФОРТРАН
  DIMENSION M(64)
  DATA M/64*1/,N/1/
  WRITE(5,15)
  READ(5,16) N
6 DO 13 I=1,N
  IF(I.GT.1.AND.M(I).EQ.M(1))GOTO13
  K=M(I)+I+N-2*(I+N)/2+1
  IF(K.GT.3) K=K-3
  WRITE(5,17) M(I),K
  M(I)=K
  IF(I.GT.1)GO TO 6
13 CONTINUE
  STOP
15 FORMAT    ('ЧИСЛО КОЛЕЦ')
16 FORMAT(I2)
17 FORMAT(I10,'->',I7)
  END
```

ПРОГРАММА БЕЗ ЧИСЕЛ И ВЫЧИСЛЕНИЙ

Работа с микрокалькулятором приучает к мысли, что программирование – это искусство (и, конечно, наука) оперирования числами и алгебраическими выражениями. Убедиться в обратном помогает программа 2.15 "Энциклопедический словарь", в которой нет ни одной вычислительной операции, ни одной числовой переменной.

Программа 2.15

"Энциклопедический словарь"
(отказ от числовых переменных и вычислительных операций)

```
1 DIM P¤250,P1¤250,E¤4,N¤4: ON ERROR E¤,N¤ GOTO 8
2 C¤=" ": LINPUT"КЛЮЧЕВОЕ СЛОВО ",C¤:IF C¤=" "THEN 13
3 DATA LOAD DC OPEN R C¤: DATA LOAD DC P¤
4 P1¤=P¤:LINPUT "ИНФОРМАЦИЯ-",P¤: IF P1¤=P¤ THEN 2
5 SCRATCH R C¤: DATA SAVE DC OPEN R(C¤) C¤:
  DATA SAVE DC P¤: DATA SAVE DC END: GOTO 2
6 P¤=" ":LINPUT"ИНФРМАЦИИ НА ДИСКЕ НЕТ. ДАЙТЕ ЕЕ",P¤:
  IF P¤=" " THEN 2
7 DATA SAVE DC OPEN R(2) C¤:DATA SAVE DC P¤:
  DATA SAVE DC END:GOTO 2
8 IF N¤="0003" AND E¤="73" THEN 6
9 IF N¤="0003" THEN 11
10 IF E¤="8005" THEN 12
11 INPUT "ДИСКОВОД НЕ В ПОРЯДКЕ.ИСПРАВИЛИ",A¤: GOTO 3
12 INPUT"СНИМИТЕ ЗАЩИТУ С ДИСКА.ГОТОВО",A¤:
   IF N¤="0005" THEN 5: GOTO 7
13 ON ERROR :REM ВЫКЛЮЧЕНИЕ АНАЛИЗАТОРА ОШИБОК
```

Разобравшись в назначении литерных переменных, легче будет понять алгоритм программы-словаря:

P – содержание справки (250 символов), запрашиваемой человеком у программы-словаря;

P1 – дублирование справки, чтобы зафиксировать сделанные в ней изменения;

C – ключевое слово поиска;

E – код ошибки, возникающей при работе по программе;

N – номер строки, где произошла ошибка.

Последний оператор строки 1 включает анализ ошибок. Если случилась ошибка, то аварийного останова не будет. Управление в этом случае передается строке 8, начиная с которой в прог-

рамме записаны операторы обработки ошибок. Даже если не включить дисковод, не вставить в него диск, не закрыть карман дисковода, не снять защиту от записи – все это не остановит работу программы. Машина попросит человека (строки 11 и 12) выполнить необходимые действия и повторит оператор, где произошел сбой.

Алгоритм работы программы прост: ключевое слово – это имя файла на диске, хранящего необходимую информацию. Если такого файла пока нет, то попытка открыть его из строки 3, вызовет программную ошибку с номером 73. Эту ситуацию обработает строка 8 и передаст управление на строку 6, согласно которой человек должен отыскать требуемую информацию и поделиться ею с машиной.

Если соответствующей информации нет, то переменная P остается пустой и запрашивается новое ключевое слово (строка 2).

Найденная информация записывается на диск R (строка 7) в виде файла, имя которого совпадает с ключевым словом C, запоминающего два (указано в скобках) сектора. Для записи файл сначала открывается (первый оператор строки 7), а затем закрывается (последний оператор строки 7).

В полученную от машины информацию можно внести исправления: для этого предназначен оператор LINPUT – своеобразный гибрид от операторов PRINT и INPUT (строка 4). При этом переменные P и P1 становятся неравными, что служит сигналом к перезаписи файла на диске (конец строки 4): старому файлу присваивается статус ликвидированного (первый оператор строки 5), а на его месте записывается новая информация (второй оператор строки 5).

В программе 2.15 "Энциклопедический словарь" следует отметить свойственные языкам высокого уровня особенности:

возможен обмен информацией между оперативной и внешней (архивной) памятью ЭВМ (это позволяет обойти их недостатки и эффективно использовать достоинства той и другой: быстрота поиска информации в оперативной памяти и сохранность после отключения питания внешней памяти);

реализован метод проб и ошибок – программная ошибка является не следствием дефекта программирования или исходных данных, а заранее планируется в соответствии с задуманным алгоритмом (построены специальные операторы для обработки особых ситуаций, например второй оператор строки 1 "деструктурирует" программу: если забыть о том, что он декларирует будущий переход при возникновении аварийной ситуации, то становится неясным, каким образом управление программой передается на строку 8);

реализован известный принцип "клиент всегда прав" (составляя программу, необходимо уметь моделировать всевозможные промахи и ошибки будущего пользователя и вносить в нее дополнения, изменения и инструкции, чтобы исключить аварийные остановы при любом направлении диалога человека и ЭВМ; при этом машину следует обучить правильным и вежливым реакциям на ошибки человека, а также внести в программу операторы, поясняющие суть ошибки и позволяющие человеку исправить ее; в удобных для пользователя или, как их еще называют, дружественных программах почти 90% операторов могут служить только для поддержания диалога с человеком; обширная память современных компьютеров, быстрота и удобство обмена информацией с архивной памятью позволяют такую роскошь).

КИТ ИЛИ КОТ?

Говоря о качественных отличиях персональных компьютеров и микрокалькуляторов, нельзя не коснуться такой важной темы, как самообучение компьютеров. В процессе общения с человеком ЭВМ способна накапливать информацию, необходимую для реализации алгоритма.

Элементы самообучения можно было увидеть в программе 2.15 "Энциклопедический словарь". Компьютер, действующий по этой программе, учится на своих ошибках. Этот же принцип заложен и в программу 2.16 "Животные-1". Правила и ход этой игры поясняет фрагмент диалога компьютера и игрока:

```
ЭТО КОТ ? 0
СДАЮСЬ, КТО ЭТО ? КИТ
ЧЕМ КИТ ОТЛИЧАЕТСЯ ОТ КОТ? ОН ЖИВЕТ В ВОДЕ

БУДЕТЕ ЕЩЕ ИГРАТЬ? 1
ОН ЖИВЕТ В ВОДЕ? 0
ЭТО КОТ ? 0
СДАЮСЬ, КТО ЭТО ? СТУДЕНТ
ЧЕМ СТУДЕНТ ОТЛИЧАЕТСЯ ОТ КОТ? ОН ХОДИТ НА ЛЕКЦИИ

БУДЕТЕ ЕЩЕ ИГРАТЬ? 1
ОН ЖИВЕТ В ВОДЕ? 0
ОН ХОДИТ НА ЛЕКЦИИ? 1
ЭТО СТУДЕНТ? 0
СДАЮСЬ, КТО ЭТО ? ПРОФЕССОР
```

и т. д. до бесконечности.

В этой игре человек задумывает животное (предмет, явление), которое компьютер должен отгадать, задавая альтернативные вопросы, разбивающие множество выдумок человека на две части. Машина как бы взбирается на двоичное дерево, ветви которого (альтернативные вопросы) при ответе человека (да–нет) раздваиваются. На ветвях "висят" ответы на вопросы.

П р о г р а м м а 2.16

"Животные-1"
(хранение бинарного дерева вопросов и ответов в массивах)

```
10 DIM B¤(511)50,O¤(511),P%(511,4),P¤1:N,M,P=1:O¤(1)="КОТ"
20 IF P=0 THEN 120:REM НАЧАЛО ЧИКЛА "ПОКА"
30 I,J=1:REM ПЕРЕХОД К КОРНЮ БИНАРНОГО ДЕРЕВА
40 IF B¤(I)=" " THEN 60:REM КОНЕЧ ВЕТКИ
50 PRINT B¤(I);:GOSUB'1:J=P%(I,P+3):I=P%(I,P+1) :GOTO 40
60 PRINT "ЭТО ";O¤(J);:GOSUB'1
70 IF P=1 THEN 110:REM ТУР ИГРЫ ЗАКОНЧЕН
80 M=M+1:INPUT "СДАЮСЬ,КТО ЭТО",O¤(M)
90 PRINT "ЧЕМ ";O¤(M);" ОТЛИЧАЕТСЯ ОТ ";O¤(J);:INPUT B¤(I)
100 P%(I,1)=N+1:N=N+2:P%(I,2)=N:P%(I,3)=J:P%(I,4)=M
110 PRINT "БУДЕТЕ ЕЩЕ ИГРАТЬ ";:GOSUB'1   :   GOTO 20
120 END
130 DEFFN ' 1:PRINT "1-ДА,0-НЕТ ?";
140 KEYIN P¤,150,140:GOTO 140
150 IF P¤<"0" OR P¤>"1" THEN 140
160 CONVERT P¤TOP:PRINT P:RETURN
```

Программа самообучающаяся, т. е., машина сама "выращивает" дерево вопросов и ответов, которое хранится в двух литерных массивах B (вопросы) и O (ответы) и в одном числовом P (логические переходы, ведущие к ответу).

Если нового уточняющего вопроса у компьютера не окажется, а это возможно в тех случаях, когда пуст элемент массива B (строка 40 в программе 2.16), то ЭВМ выдаст свою догадку (строка 60). Она чаще всего оказывается неудачной. Тогда машина запросит (левое плечо альтернативы на строках 70-100) задуманное человеком (строка 80) и попросит сформулировать новый альтернативный вопрос, ответ на который отличал бы задуманное от догадки ЭВМ. Так программа учится на своих ошибках.

Человек в игре "Животные-1" должен честно отвечать "да" или "нет" на альтернативные вопросы машины. Диалог поддерживает подпрограмма с именем "1", хранящаяся в строках 130-160. Она подсказывает форму ответа (строка 130) и требует нажатия клавиш "1" или "0", блокируя остальные.

Оператор ввода информации без прерывания (строка 140) отличается от своего аналога с прерыванием (строка 80) еще и тем, что не выводит на экран дисплея вводимый символ. Об этом небходимо позаботиться в программе (второй оператор строки 160). Протокол фиксирует все ответы человека. В конце протокола мы подписали "и т. д. до бесконечности". Такого, конечно, быть не может. Более того, число вопросов, которое может задать машина, отгадывая задуманное человеком, не будет превышать девяти* (строка 10 программы 2.16).

Ячейки памяти, составляющие по сути ветви бинарного дерева, а по форме – элементы массивов, используются весьма неэффективно. Литерные массивы B и O и числовой двумерный массив P при выращивании дерева вопросов и ответов остаются практически пустыми, или "дырявыми" на жаргоне программистов. Это следствие того, что массивы относятся к статическим типам данных, – их размер необходимо задавать в самом начале программы, когда еще не ясно, сколько элементов массива потребуется для выполнения поставленной задачи.

Наиболее естественное представление бинарного дерева получается на языке Паскаль. Здесь выручают гибкие (динамические) средства описания структур, в частности ссылочные структуры данных. Каждый узел дерева в Паскаль-программе 2.17 "Животные-2", написанной А. Рыжовым, содержит две ветви и две ссылки на другие узлы. На одной ветви записан уточняющий вопрос, а на другой – отгадка, предлагаемая машиной, если на ее вопрос был дан положительный ответ.

Всякий раз, когда отгадка не подходит, идет поиск нового уточняющего вопроса, расположенного в узле, на который указывает первая ссылка (указатель "да"). Если же ответом на уточняющий вопрос было "нет", то отгадка не предлагается, а идет поиск уточняющего вопроса по второй ссылке (указатель "нет"). Электронный партнер сдается, если необходимая ссылка пока что пуста (NIL), и просит нарастить дерево.

В программе 2.17 не описана процедура NEXTPLAY – запрос о желании продолжить игру и булева функция YES, необходимая для ввода ответа игрока ("да" или "нет").

Динамические типы данных позволяют более эффективно использовать столь важный ресурс, как память: резервировать новые ячейки заранее не требуется, поскольку они будут заняты по мере надобности.

---

* Расчет прост: $2^9 = 511 + 1$.

П р о г р а м м а 2.17

"Животные-2"
(использование ссылочного типа данных
для хранения бинарного дерева вопросов и ответов)

```
PROGRAM ANIMALS (INPUT,OUTPUT);
LABEL 1,2;
TYPE TRELEMENT=RECORD
        QUESTION,ANSWER:STRING;
        YESPOINTER,NOPOINTER:^TRELEMENT
     END;
VAR INITIAL,CURRENT:TRELEMENT;NEXT:^TRELEMENT;
...
BEGIN
   INITIAL.QUESTION:='ОНО ЛОВИТ МЫШЕЙ';
   INITIAL.ANSWER:='КОТ';
   INITIAL.YESPOINTER:=NIL;
   INITIAL.NOPOINTER:=NIL;
   1:  CURRENT:=INITIAL;
       WRITELN('ЗАДУМАЙТЕ ЖИВОТНОЕ');
   2:  WRITE(CURRENT.QUESTION);
       IF YES THEN BEGIN
              WRITE('ЭТО -',CURRENT.ANSWER,',УГАДАЛА');
              IF YES
                 THEN BEGIN NEXTPLAY;GOTO 1 END
                 ELSE NEXT:=CURRENT.YESPOINTER
                   END
              ELSE NEXT:=CURRENT.NOPOINTER;
       IF NEXT=NIL THEN BEGIN
            NEW(NEXT);WRITE('СДАЮСЬ.КТО ЭТО');
            READLN(NEXT^.ANSWER);
            WRITE('А ЧЕМ ОТЛИЧАЕТСЯ');
            READLN(NEXT^.QUESTION);NEXTPLAY;GOTO 1
                         END;
       CURRENT:=NEXT^;GOTO 2
END.
```

ЭЛЕКТРОННЫЙ СОВЕТЧИК

Программы 2.16 и 2.17 страдают существенным недостатком, исключающим их практическое применение. Стоит выключить компьютер или ввести другую программу, и бинарное дерево будет потеряно. В новой игре его придется выращивать снова.

Проблема хранения бинарного дерева и его плотной упаковки легко решается с помощью накопителя на магнитных дисках. Ветви бинарного дерева при этом можно хранить в виде файлов. В этом случае имя каждого файла в литерной форме повторяет номер ветви.

Такой алгоритм игры заложен в программу 2.18 "Животные-3". Он почти не отличается от алгоритма, реализованного

в программе 2.16, но здесь электронный партнер сдается не тогда, когда соответствующий элемент массива B является пустым, а в том случае, если файл отсутствует на диске. Второй оператор строки 1 программы 2.18 отслеживает возможную аварийную ситуацию – попытку открыть несуществующий файл (второй оператор строки 5), и передает управление на строку 10.

Программа 2.18

"Животные-3"
(хранение бинарного дерева вопросов и ответов
в файлах магнитного диска)

```
1 DIM O¤(500),B¤45,P¤1,A¤1:ON ERROR E¤,N¤ GOTO 10
2 I,J=0:O¤="КОТ":GOTO 7
3 J=J+1:IF I>0 THEN 4:I=1:GOTO 5
4 I=2*I:IF P¤=A¤ THEN 5:I=I+1
5 CONVERT I TO I¤,(########):
  DATA LOAD DC OPEN R I¤:DATA LOAD DC B¤,O¤,P¤
6 O¤(J)=O¤:PRINT B¤;:GOSUB 14:A¤=V¤:IF A¤<>P¤ THEN 3
7 PRINT "ЭТО -";O¤;
8 PRINT " УГАДАЛА ";:GOSUB 14:IF V=0 THEN 3
9 PRINT "ЕЩЕ БУДЕТЕ ИГРАТЬ ";:GOSUB 14:
  IF V=1 THEN 2:STOP
10 INPUT "СДАЮСЬ. КТО ЭТО ",O1¤
11 FOR K=1 TO J:IF O1¤<>O¤(K) THEN 12:PRINT
   "НЕЧЕСТНО ИГРАЕТЕ.":GOTO 9
12 NEXT K
13 PRINT "ЗАДАЙТЕ АЛЬТЕРНАТИВНЫЙ ВОПРОС,ОТВЕТИВ НА
   КОТОРЫЙ МОЖНО ОТЛИЧИТЬ ";O1¤;" ОТ ";O¤;:INPUT B¤:
   PRINT "ПРАВИЛЬНЫЙ ОТВЕТ ";:GOSUB 14:P¤=V¤:
   DATA SAVE DC OPEN R(2)I¤:DATA SAVE DC B¤,O1¤,P¤:
   DATA SAVE DC END:GOTO 9

14 PRINT " ДА-НЕТ ? ";
15 KEYIN V¤,16,15:GOTO 15
16 IF V¤=HEX(64)ORV¤=HEX(C4)ORV¤="1"ORV¤=HEX(79) THEN21
17 REM        D            Д                    Y СТРОЧНЫЕ
18 IF V¤=HEX(6E)ORV¤=HEX(CE)ORV¤="0"ORV¤=HEX(68) THEN22
19 REM       N             Н РУССК              Н ЛАТ СТР.
20 GOTO 15
21 V=1:V¤="1":D¤="ДА":GOTO 23
22 V=0:V¤="0":D¤="НЕТ"
23 PRINT D¤:RETURN
```

Программа формирует запрос, обращенный к человеку. В цикле с параметром (строки 11 и 12) проверяется соблюдение правил игры (предшествующие предположения электронного партнера, хранящиеся в литерном массиве O, не должны совпа-

дать с ответом), запрашиваются новый альтернативный вопрос и правильный ответ на него. Они и записываются на диск в виде нового файла (строка 13).

Игра "Животные" несет в себе зачатки так называемой экспертной системы (электронного советчика), помогающей специалисту принимать верное решение. Вот примеры протоколов диалога с одной из таких систем:

```
У БОЛЬНОГО ВЫСОКАЯ ТЕМПЕРАТУРА? ДА
ЕСТЬ ХРИПЫ В ЛЕГКИХ? ДА

:

ЭТО ПНЕВМОНИЯ.

ДАВЛЕНИЕ НА ВХОДЕ В РЕАКТОР НИЖЕ 40 БАР? ДА
ТЕМПЕРАТУРА В РЕАКТОРЕ НИЖЕ 190°С? НЕТ

:

ВЫШЕЛ ИЗ СТРОЯ НАСОС № 1.
```

Ответы, даваемые компьютером на основе диалога с человеком, нуждаются в проверке. Причиной нарушений в работе реактора на деле может оказаться не выход из строя насоса № 1, а что-то иное. "Разговор" должен продолжить квалифицированный специалист. Он вводит новый альтернативный вопрос, подчеркивающий разницу между последним предположением системы и действительным явлением или предметом. В следующий раз в подобной ситуации компьютерный эксперт уже не оплошает, поскольку в его памяти будет храниться новая информация, позволяющая более точно детерминировать явления и предметы окружающего мира. Так ЭВМ позволяет создавать модель, которая помнит массу различных ситуаций, связанных с конкретной системой или объектом, и причины возникновения этих ситуаций.

ЧТО В ИМЕНИ ТЕБЕ МОЕМ?

Имя, память... Подобные понятия относятся скорее к категориям поэтическим. Однако очень часто ими оперируют программисты, создавая компьютерные базы данных. Если при этом опираться на систему файлов и давать им продуманные

имена, то даже с помощью простой Бейсик-программы удается оперативно искать необходимую информацию, дополнять ее или вносить изменения.

Одна из первых проблем, которую решают родители новорожденного, – это выбор для него имени. Накладывает ли имя отпечаток на характер человека, на его судьбу – вопрос сложный и спорный. Тот же факт, что имя, присваемое новому файлу (переменной, функции, процедуре, программе), может сыграть огромную роль в его судьбе, не будет оспаривать ни один более-менее опытный программист.

На микрокалькуляторе имена переменных (их еще называют идентификаторами) скрываются за безликими номерами ячеек памяти: П1, ИПА, КИП2 и т.д. Программы, собранные в этой книге, хотя и написаны на языках персональных ЭВМ, однако содержат довольно безликие имена переменных в один-два символа. Но это, скорее, дань традиции, нежели необходимость. Дело в том, что первые версии языков определяли имена переменных лишь из одного или двух символов A, A1, XZ и т. д. В современных версиях языков это ограничение снято, так как память машины перестала быть лимитирующим фактором.

Начнем со строгого определения.

Файл (от англ. file – папка, подшитые бумаги, дело, картотека) – это набор семантически связанных между собой данных, воспринимаемых как одно целое, хранящихся в форме записи на внешнем носителе памяти ЭВМ. Для непосвященных в премудрости информатики, подобное определение может показаться туманным. А ведь каждый из нас в повседневной жизни нередко сталкивается с файлами. Обратимся к операторам, работающим с файлами. Воспользуемся конкретным примером.

Вы захотели перечитать стихотворение А. С. Пушкина "Пророк". Берете с полки томик его стихов, по названию стихотворения находите в оглавлении номер страницы, открываете на ней книгу и... "перегружаете" файл "ПРОРОК" с "внешнего носителя памяти" (бумажного листа) в "оперативную память" (человеческий мозг).

Теперь представьте, что это стихотворение вместе с другими стихами на магнитном диске, а вы захотели перенести его на экран дисплея вашего компьютера, например "Искры 226". Какие же операторы Бейсика понадобятся для этого?

```
DIM A¤(30)35
```

– в оперативном запоминающем устройстве ЭВМ стихотворение "Пророк" будет храниться в виде литерного одномерного массива с тридцатью элементами-строками, самая длинная из которых "Восстань, пророк, и виждь и внемли," состоит из 35 знаков:

DATA LOAD DC OPEN R "ПРОРОК"

- на магнитном диске, вставленом в карман R дисковода, открывается (OPEN*) файл с именем "ПРОРОК" для загрузки (LOAD**) записанных в нем данных (DATA***) в оперативную память машины: в данном случае открыть файл означает: по его имени найти в каталоге диска информацию о том, в каком месте магнитного слоя находится начало нужной записи и передвинуть соответственно магнитную головку дисковода;

DATA LOAD DC A¤()

- руководствуясь этой инструкцией, ЭВМ считает (LOAD) данные (DATA) из открытого файла и заполнит ими элементы литерного массива A;

MAT PRINT A¤

- стихотворение "Пророк" переносится на экран дисплея; если же вкралась ошибка, например в первую строку стихотворения, то исправить ее и в оперативной памяти, и на магнитном диске можно следующим образом:

A¤(1) = "Духовной жаждою томим,"

- первая строка стихотворения исправлена в оперативной памяти,

SCRATCH R "ПРОРОК"

- файл на магнитном диске, хранящий стихотворение "Пророк" с ошибкой в первой строке, стирается (SCRATCH****), точнее, получает статус ликвидированного - место на диске этот файл занимает по-прежнему, но его уже нельзя считать;

DATA SAVE DC OPEN R ("ПРОРОК") "ПРОРОК"

- на диске на месте ликвидированного файла с именем "ПРОРОК" (указано в скобках) открывается (OPEN) для записи (SAVE) новый файл данных (DATA) с тем же именем (указано после скобок);

DATA SAVE DC A¤()

- элементы литерного массива A переносятся на магнитный диск;

---

* Open (англ.) - открывать файл.

** Load (англ.) - загружать (программу).

*** Data (англ.) - данные, информация.

**** Scratch (англ.) - затирать информацию на магнитном носителе.

```
DATA SAVE DC END
```

– файл после записи закрывается (END)*;
как же попадет стихотворение "Пророк" на магнитный диск:

```
DIM A¤(30)35
```

– резервируем место в оперативной памяти ЭВМ для записи стихотворения;

```
MAT INPUT A¤
```

– организуем ввод строки, например:
?"Духовной хахаю томим," (здесь ошибка, вернее, опечатка)
?"В пустыне мрачной я влачился,"
и т. д. все 30 строк (строки заключены в кавычки для того, чтобы оператор INPUT** смог "проглотить" и запятые, которые вне кавычек служат для разделения значений переменных);

```
DATA SAVE DC OPEN R(6) "ПРОРОК"
```

– на диске открывается (OPEN) для записи (SAVE) новый файл данных (DATA) с именем "ПРОРОК", под который отводится шесть секторов (каждый сектор магнитного диска ЭВМ "Искра 226" может вместить до 256 байт, т.е. принять не более семи строк стихотворения (7·35 = 245); под все стихотворение "Пророк" достаточно отвести пять секторов, шестой – служебный).

Подобные расчеты приходится вести, открывая на диске новый файл для записи данных (персональный компьютер "Искра 226"):

```
DATA SAVE DC A¤()
DATA SAVE DC END
```

В описанных операторах работы с файлами нерасшифровано только значение символов DC. Они означают, что действия с файлами ведутся на диске (D), предварительно размеченном для работы в режиме каталога (C).

Кроме того, распространена бесфайловая система записи информации на диск с прямой адресацией к секторам и дорожкам. На этом же принципе основаны системы управления базой данных, входящие в стандартное программное обеспечение компьютера.

Создание программных комплексов под силу только системным программистам-профессионалам. Специалистам, для кото

---

* End (англ.) – служебное слово, указывающее на конец программной конструкции.

** Input (англ.) – ввод (данных).

рых программирование – только вспомогательное средство для решения прикладных задач, при организации базы данных без файлов не обойтись. Вот здесь может помочь правильный выбор имени файла.

В программе 2.15 "Энциклопедический словарь" в качестве имени файла, хранящего справку, выступает само ключевое слово. Бейсик "Искры 226", на котором написана эта программа, позволяет давать файлу имя с русскими и латинскими буквами длиною до восьми знаков. При этом слова ВЫЧИСЛИТель и ВЫЧИСЛИТельная машина компьютер воспринимает как одно слово, что накладывает на использование программы определенные ограничения.

В программе 2.18 "Животные-3" имя файла в литерной форме повторяет номер* ветви бинарного дерева вопросов и ответов, необходимых для игры.

Программа 2.19 "Экзаменатор" позволяет использовать "Искру 226" для контроля знаний: на дисплей выдаются заранее подготовленные вопросы и несколько вариантов ответов, один из которых верный. Вопросы хранятся на магнитном диске в виде файлов, имена которых совпадают с номерами этих файлов (использована идея, уже знакомая читателю по программе "Животные"). Отдельный файл с именем "ВОПРОСЫ" хранит общее число вопросов на диске.

Неудачный вопрос можно корректировать, сообщив машине его номер (строка 15).

Перед очередным экзаменом обуславливаются количество задаваемых экзаменуемому вопросов и время на обдумывание каждого (строка 16). Это исключает возможные попытки экзаменуемого "пересидеть" экзаменатора (строка 21).

Имитируя реальный экзамен, компьютер выбирает с диска вопросы случайным образом, не допуская повторов (строка 18).

Экзаменуемому, знакомому с вычислительной техникой, не удастся вмешаться в работу программы, поскольку "лишние" клавиши клавиатуры при ответе заблокированы (строка 22).

Оператор опроса клавиатуры (строка 20) не прерывает исполнения программы и не требует завершающего нажатия клавиши "Ввод" после выбора ответа.

Пустой цикл с параметром (строка 25) напоминает путы, надеваемые на ноги пасущейся лошади: он позволяет регулировать темп выдачи вопросов.

Операторы строки 31 работают только тогда, когда формируется банк вопросов на диске. На эту строку управление передается в том случае, если на диске еще нет файла "ВОПРОСЫ"

---

* Имя файла "00000015" получается как результат несложного расчета: $2^4 - 1 = 15$. Пятнадцатая ветвь бинарного дерева, например, находится с правого края четвертого яруса.

Программа 2.19

"Экзаменатор"
(линейное расположение файлов на магнитном диске)

```
1 DIM V¤200,O¤(5)100,M¤,B(50):ON ERROR N¤,E¤ GOTO 31
2 M¤="ВОПРОСЫ":GOSUB 26:DATA LOAD DC M
3 PRINT "ВОПРОСОВ НА ДИСКЕ -";M
4 INPUT"1-НОВЫЙ ВОПРОС,2-ИСПРАВЛЕНИЕ,3-ЭКЗАМЕН,4-КОНЕЦ"
  ,A:ON A GOTO 5,15,16:STOP
5 M=M+1:K=M:GOSUB 27:PRINT "ВОПРОС";M:V¤=" "
6 LINPUT V¤:IF A=2 THEN 9:REM ИСПРАВЛЕНИЕ
7 INPUT "ЧИСЛО ВАРИАНТОВ ОТВЕТА (2-5)",N:IF N<2 OR N>5
  THEN 7:FOR I=1 TO N:O¤(I)=" ":NEXT I
8 INPUT "НОМЕР ПРАВИЛЬНОГО ОТВЕТА",P:IF P>N THEN 8
9 FOR I=1 TO N:PRINT I;"-й ВАРИАНТ ОТВЕТА (";:
  IF I=P THEN 10:PRINT "НЕ";
10 PRINT "ВЕРНЫЙ)":LINPUT O¤(I):NEXT I
11 IF A=2 THEN 12:DATA SAVE DC OPEN R(7) M¤:GOTO 13
12 GOSUB 28:REM ПЕРЕОТКРЫТИЕ ФАЙЛА С ИМЕНЕМ M¤
13 DATA SAVE DC V¤,N,P,O¤():GOSUB 29:IF A=2 THEN 3
14 M¤="ВОПРОСЫ":GOSUB 28:DATA SAVE DC M:GOSUB 29:GOTO 3
15 INPUT "НОМЕР ВОПРОСА",K:IF K>M THEN 15:GOSUB 27:
   GOSUB 26:GOSUB 30:GOTO 6
16 INPUT"ЧИСЛО ВОПРОСОВ",V:IF V>M OR V>50 THEN 16:
   MAT B=ZER:R=0:INPUT"ВРЕМЯ НА ОТВЕТ (СЕК)",S
17 FOR J=1 TO V
18 K=1+INT(V*RND(1)):IF B(K)=1 THEN 18:B(K)=1:GOSUB 27:
   GOSUB 26:GOSUB 30:CONVERT P TO P¤,(#):CONVERT N
   TO N¤,(#)
19 PRINT HEX(03);"ВОПРОС -";J:PRINT STR(V¤,1,LEN(V¤)):
   PRINT:FOR I=1TON:PRINT I;".";STR(O¤(I),1,LEN(O¤(I)))
   :NEXT I:INPUT ¤T0
20 A¤=" ":KEYIN A¤,22,20
21 INPUT ¤T:IF (T0-T)/2000<S THEN 20:
   PRINT "ВРЕМЯ НА ОБДУМЫВАНИЕ ИСТЕКЛО.":GOTO 25
22 IF A¤<"1" OR A¤>N¤ THEN 20
23 IF A¤=P¤ THEN 24:PRINT "НЕВЕРНО.":PRINT "ПРАВИЛЬНЫЙ
   ОТВЕТ -";O¤(P):GOTO 25
24 PRINT "ПРАВИЛЬНО.":R=R+1
25 FOR L=1 TO 1000:NEXT L:NEXT J:PRINT " ВЫ ОТВЕТИЛИ
   ПРАВИЛЬНО В ";R;" СЛУЧАЯХ ИЗ ";V:GOTO 3
26 DATA LOAD DC OPEN R M¤:RETURN
27 CONVERT K TO M¤,(########):RETURN
28 SCRATCH R M¤:DATA SAVE DC OPEN R (M¤)M¤:RETURN
29 DATA SAVE DC END:RETURN
30 DATA LOAD DC V¤,N,P,O¤():RETURN
31 DATA SAVE DC OPEN R(2)M¤:
   DATA SAVE DC O:GOSUB 29:GOTO 3
```

(попытка открыть его приводит к ошибке). Такая программная инициализация диска имеет свои плюсы и минусы: файл "ВОПРОСЫ" не требуется создавать до первого запуска программы; но "Экзаменатор" будет создавать файл "ВОПРОСЫ" на любом диске, в том числе и вставленном в дисковод по ошибке.

В то же время своеобразным пропуском диска в дисковод может служить сам файл "ВОПРОСЫ", если он создается вне программы. В этом случае из программы "Экзаменатор" следует убрать второй оператор и строку 31.

Разрабатывая программное обеспечение автоматизированных систем для научных исследований, в качестве имени файла можно использовать имя "8811141545" и ему подобные. Такое имя весьма наглядно и легко расшифровывается, например, как имя файла, хранящего показания приборов экспериментального стенда, снятые 14 ноября 1988 г. в 15 ч. 45 мин.

Аналогичным образом удобно вести электронный дневник, записывая в файлы дела, мысли, впечатления прошедшего дня. В подобных случаях для файлов подходят имена типа "23.10.88". В таком дневнике легко и быстро отыскиваются необходимые данные с помощью несложной программы для ЭВМ.

Есть и другие принципы именования файлов, образующих базы данных. Главное, чтобы легко было найти тот или иной файл.

КОМПЬЮТЕР-СЕКРЕТАРЬ

Компьютеры, составляющие ядро автоматизированных систем научных исследований, должны работать в режиме реального времени и согласовывать свои действия с показаниями встроенных часов. Большинство программ, собранных в книге, можно уподобить хорошему бегуну – поставленные задачи требовалось решать за минимальное время. Программы, работающие в режиме реального времени, подобны участникам автомобильного ралли, которые должны пересекать промежуточные финиши в строго оговоренное время. Досрочный финиш наказывается не менее чем опоздание.

Примером Бейсик-программы реального времени является программа 2.20 "Электронный секретарь". Она заменяет секретаря, старательно напоминающего о текущих делах. По сути эта

Программа 2.20

"Электронный секретарь" (режим реального времени)

```
1 DIM A¤(50)200,A¤200,S(2,50):I=0:PRINT"ТОЧНОЕ ВРЕМЯ";
  :GOSUB 17:S0=S:GOSUB 18:T0=T
2 IF I=50 THEN 10
3 I=I+1:PRINT I;"-ОЕ ЗАДАНИЕ (RETURN-ВСЕ)";:
  INPUT A¤(I):IF A¤(I)=" " THEN 10
4 PRINT "НА КАКОЕ ВРЕМЯ НАЗНАЧЕНО";:GOSUB 17:S(1,I)=S:
  GOSUB 18:IF S(1,I)>S THEN 5:PRINT "ПОЗДНО.":GOTO 4
5 INPUT "ПРОДОЛЖИТЕЛЬНОСТЬ(МИН)",S:S(2,I)=S(1,I)+60*S:
  IF I=1 THEN 2
6 FOR J=1 TO I-1
7 IF S(1,I)>=S(2,J) OR S(2,I)<=S(1,J) THEN 9
8 PRINT "НА ЭТО ВРЕМЯ УЖЕ НАЗНАЧЕНО - ";A¤(J):GOTO 4
9 NEXT J:GOTO 2
10 P=0:
   FOR J=2 TO I:IF S(1,J-1)<=S(1,J) THEN 11:
   FOR K=1 TO 2:S=S(K,J-1):S(K,J-1)=S(K,J):S(K,J)=S:
   NEXT K:A¤=A¤(J-1):A¤(J-1)=A¤(J):A¤(J)=A¤:P=1
11 NEXT J:
   IF P=1 THEN 10
12 FOR J=1 TO I:PRINT HEX(03):PRINT:FOR K=1 TO 2:PRINT
   INT(S(K,J)/3600);" ЧАС.";S(K,J)/60-60*INT(S(K,J)/
   3600);" МИН.";:IF K=2 THEN 13:PRINT "- ";
   STR(A¤(J),1,LEN(A¤(J)));" -";
13 NEXT K
14 GOSUB 18:PRINT HEX(01);"ВРЕМЯ ";:PRINTUSING "##.",
   H;M;S-3600*H-60*M:IF ABS(S(1,J)-S)>=5 AND
   ABS(S(2,J)-S)>=5 THEN 15:PRINT HEX(07)
15 IF S<S(2,J) THEN 14
16 NEXT J:STOP "НА СЕГОДНЯ ВСЕ.ДО ЗАВТРА."
17 INPUT "(ЧАСЫ,МИНУТЫ)",H,M:IF H>23 OR M>59 THEN 17:
   S=3600*H+60*M:RETURN
18 INPUT ¤T:S=S0+(T0-T)/2000:H=INT(S/3600):
   M=INT(S/60-60*H):RETURN
```

программа превращает компьютер стоимостью в несколько десятков тысяч рублей в многофункциональный будильник*.

Программу "Электронный секретарь" запускают в начале рабочего дня. Машина запрашивает план дня (строки 2-5) и держит на дисплее информацию о ближайшем деле, подавая звуковой сигнал, например, в моменты, когда следует начать и когда отведенное время истекло (строка 14).

Коротко о программе 2.20. Для удобства изложения разобъем ее на блоки.

---

* Это все же лучше, чем если бы машина простаивала, в то время как ее владелец "разрывался" бы между несколькими делами, не сумев спланировать свою работу.

Блок 1 (строка 1): резервирование места для хранения в памяти компьютера списка мероприятий (массив A), времени их начала (первая строка массива S) и окончания (вторая строка массива S); выставление часов компьютера (все временные данные ЭВМ хранит, переведенными в секунды, прошедшие от полуночи).

Блок 2 (строки 2-9): начальный диалог человека с машиной: ввод планов на день (компьютер следит, чтобы не было накладок: новое мероприятие должно начинаться не ранее, чем завершится предыдущее, и не позднее, чем начнется очередное (условный переход на строке 7).

Блок 3 (строки 10 и 11): основной диалог (информацию о делах дня в память компьютера вводят так, как будто бы пишут на чистой странице дневника: в произвольной последовательности записывают важнейшие дела, а затем заполняют паузы более мелкими; так и с нашей программой: работая с блоком 2, можно вводить дела дня в любом удобном порядке; компьютер рассортирует их* по хронологии, руководствуясь операторами строк 10 и 11.

Блок 4 (строки 12-16): монолог компьютера, "поглядывающего на часы" и сообщающего человеку о начале и конце очередного дела дня (компьютер "Искра 226", для которого написана программа, говорить не может (хотя машины с речевым вводом и выводом уже существуют), он только подает звуковой сигнал, код которого 07 (конец строки 14), призывающий взглянуть на дисплей).

Блок 5 (строки 17 и 18): две подпрограммы, позволяющие обращаться к часам ЭВМ и переводить машинное время, убывающее на 2000 условных единиц за одну секунду, в привычные часы (H), минуты (M) и секунды (S).

ПРОГРАММА, ПЕЧАТАЮЩАЯ САМА СЕБЯ

Эта проблема формулировалась еще Н. Винером. Можно ли написать программу, которая после запуска напечатала бы сама себя? Задача была широко известна, и многие считали ее неразрешимой.

---

* Специалист по численным методам легко распознает известный "метод пузырька".

Колумбу приписывают два великих открытия. Первым, по-видимому, можно считать открытие Америки, вторым – решение задачи о "колумбовом яйце". Суть ее была в том, что яйцо требовалось поставить на острый конец на гладкой поверхности. Многие пытались для этого раскручивать яйцо, делать незаметные подпорки. Колумб, как утверждает легенда, поставил яйцо, надломив скорлупу.

Бейсику также можно приписать два значительных достижения. Прежде всего, именно с помощью Бейсика многие открывают для себя прекрасный мир программирования. Не менее интересно для нас и то, что Бейсик, так сказать, по-колумбовски решил задачу о самораспечатывании программы.

Например, чтобы решить эту задачу на языке PL/1, требуется чуть ли не полсотни операторов. Практически на любой современной версии Бейсика любая программа может быть распечатана единственным оператором LIST, как в непосредственном, так и в программном режимах. При этом, если колумбово решение задачи о яйце, похоже, не нашло практического приложения, то решение задачи о самораспечатывании программы можно использовать. Пример тому – программа 2.21 "16 формул". Она позволяет статистически обработать результаты однофакторного эксперимента, выбрать аппроксимирующую функцию из шестнадцати возможных.

Строки 1-16 программы не только служат для расчета значения функций (строка 29), но и распечатываются самой же программой (строка 19) перед тем, как на экран будет выведен запрос номера формулы.

Колумбово решение задачи с яйцом имеет дефект. В самом деле, яйцо пытались поставить тупым концом вверх, чтобы его было удобно есть, а установленное по-колумбовски, оно будет слишком неустойчиво.

Решение о распечатке формул также не без изъяна: у формул будет ненужный "хвостик" – RETURN. На практике регрессионные уравнения выводят на печать оператором PRINT.

Тем не менее метод распечатки фрагмента программы одновременно с ее прогоном весьма привлекателен. Кстати, в нашей книге он уже был использован: взгляните на строку 2 в программе 2.12 "Числа Фибоначчи-Б". Первый оператор этой строки распечатывает нулевую строку программы 2.12, где записана анализируемая функция.

П р о г р а м м а 2.21

"16 формул"
(распечатка части программы)

```
0 CLS:? "РЕГРЕССИЯ Y=F(X)":GOTO 17
1 Y=A+B*X:RETURN                2 Y=1/(A+B*X):RETURN
3 Y=A+B/X:RETURN                4 Y=X/(A+B*X):RETURN
5 Y=A*B^X:RETURN                6 Y=A*EXP(B*X):RETURN
7 Y=A*EXT(B*X):RETURN           8 Y=1/(A+B*EXP(-X)):RETURN
9 Y=A*X^B:RETURN                10 Y=A+B*LGT(X):RETURN
11 Y=A+B*LOG(X):RETURN          12 Y=A/(B+X):RETURN
13 Y=A*X/(B+X):RETURN           14 Y=A*EXP(B/X):RETURN
15 Y=A*EXT(B/X):RETURN          16 Y=A+B*X^N:RETURN
17 INPUT "ЧИСЛО ТОЧЕК?";T:DIM X(T),Y(T)
18 FOR I=1 TO T:?"X,Y"I;:INPUT X(I),Y(I):NEXT
19 CLS:LIST 1-16:INPUT"НОМЕР ВИДА РЕГ.?";K:IF K=16 INPUT "N?";N
20 X1=0:X2=0:Y1=0:P=0:FOR I=1 TO T
21 : X=X(I):Y=Y(I)
22 : ON K GOSUB 48,35,36,37,38,39,38,40,41,42,43,35,44,45,46,47
23 : X1=X1+X:Y1=Y1+Y:X2=X2+X*X:P=P+X*Y
24 NEXT ' КОНЕЦ ЦИКЛА ФОРМИРОВАНИЯ СИСТЕМЫ 2-Х ЛИНЕЙНЫХ УРАВНЕНИЙ
25 B=(X1*Y1-T*P)/(X1*X1-T*X2):A=(Y1-B*X1)/T
26 ON K GOSUB 48,48,48,48,49,50,51,48,51,48,48,52,53,50,51,48
27 ?" I"," X"," Y"," F(X)","ОТКЛОНЕНИЕ (%)" ' ШАПКА ТАБЛИЦЫ
28 D=0:FOR I=1 TO T ' ВЫВОД ТАБЛИЦЫ И ПОДСЧЕТ ДИСПЕРСИИ
29 : X=X(I):ON K GOSUB 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
30 : ? I,X,Y(I),Y,100*(Y(I)-Y)/Y:D=D+(Y-Y(I))^2
31 NEXT
32 ?"A="A" B="B" СРЕДНЕЕ КВАДРАТИЧНОЕ ОТКЛОНЕНИЕ="SQR(D/(T-1))
33 INPUT "1-ДРУГОЕ УРАВН.,2-ИСПРАВ.,3-STOP?";Z:ON Z GOTO 19,34,54
34 INPUT"N ТОЧКИ?";I:? "БЫЛО"X(I);Y(I):INPUT X(I),Y(I):GOTO 33
35 Y=1/Y:RETURN                 36 X=1/X:RETURN
37 Y=X/Y:RETURN                 38 Y=LGT(Y):RETURN
39 Y=LOG(Y):RETURN              40 X=EXP(-X):Y=1/Y:RETURN
41 X=LGT(X):Y=LGT(Y):RETURN     42 X=LGT(X):RETURN
43 X=LOG(X):RETURN              44 X=1/X:Y=1/Y:RETURN
45 X=1/X:Y=LOG(Y):RETURN        46 X=1/X:Y=LGT(Y):RETURN
47 X=X^N:RETURN                 48 RETURN
49 A=EXT(A):B=EXT(B):RETURN     50 A=EXP(A):RETURN
51 A=EXT(A):RETURN              52 A1=A:A=1/B:B=A1/B:RETURN
53 A=1/A:B=B*A:RETURN           54 END
```

РАБОТАЕМ С ГОТОВЫМИ ПРОГРАММАМИ

ПРОГРАММИРОВАНИЕ БЕЗ ПРОГРАММИРОВАНИЯ

На первый взгляд такое название парадоксально. И все же некоторые современные программы, точнее их комплексы, позволяют успешно использовать компьютер специалистам, которые не знают ни одного языка программирования.

Рассмотрим в качестве примера решение пяти задач в программной среде Eureka, разработанной известной фирмой Borland*.

Здесь и далее в книге некоторые термины будут встречаться без пояснений (например, "активизировать", "войти в окно дисплея"). Дело в том, что наша задача – рассмотреть особенности некоторых программных продуктов, а не технику общения с ЭВМ, кстати, относительно не зависимую от конкретной программы.

Компьютер редко используют для сложных вычислений по специальной программе, реализующей разработанный пользователем алгоритм. Значительно чаще это повседневные расчеты, выполняемые различными специалистами. В таких случаях в ход идут бухгалтерские счеты, лист бумаги или программируемый микрокалькулятор.

Созданная специально для общения с компьютером среда Eureka, как и ее аналоги, позволяет специалистам-непрограммистам отказаться от использования вспомогательных средств, особенно если речь идет о прикладных расчетах.

Среда Eureka превращает машину не только в быстродействующие, но и весьма интеллектуальные счеты, обладающие знаниями о том, как вести вычисления. Компьютеру достаточно сообщить лишь условия задачи, а как там он "прощелкает ее на счетах", будет он делить или вычитать – это его дело. Главное, чтобы ответ получался верным. Впрочем, для проверки Eureka располагает достаточно мощными средствами.

И современная, и прошлого века методики преподавания школьных (и не только школьных) дисциплин ориентированы, как правило, не на постановку задачи, а на решение уже сформулированной. Это мешает обучаемому смотреть на решаемую проблему иначе, чем через призму выбора способа решения. Компьютеризация школы, базирующаяся на преподавании языков программирования (Бейсик, Паскаль, Рапира и др.), фактически ничего не изменяет в существующем положении вещей.

Другое дело – изучение среды программирования, аналогичной Eureka, которая "общается" на естественном (почти) языке. Здесь над пользователем не довлеет выбор метода решения и, тем более, не требуется умение использовать этот метод. Все внимание специалист может уделить постановке задачи, математическому моделированию реальных ситуаций.

Воспользуемся распечаткой изображения с экрана (hardcopy), иллюстрирующей решение в среде Eureka задачи "о сукне", из

---

* Фирма славится своими Turbo-версиями популярных языков для персональных компьютеров: Turbo-C, Turbo-Pascal, Turbo-BASIC, Turbo-prolog.

рассказа А. Чехова "Репетитор", с которой мы уже познакомились в первой главе.

Сделаем небольшое отступление. В дальнейшем в тексте все чаще будут встречаться английские термины. К сожалению, это неизбежно, – всякий, кто хочет "быть на ты" с компьютером, должен стремиться знать английский язык или хотя бы запомнить несколько десятков слов, которые уже стали международными в области вычислительной техники.

Экран в среде Eureka разбивается на пять самостоятельных областей или окон (рис. 2.2).

Первое из них – это окно основного меню (Eureka: The Solver) из восьми "блюд": работа с файлами (File*), ввод с клавиатуры и редактирование условий задачи (Edit), решение задачи (Solve), специальные команды (Commands), создание отчетов о решении задачи (Report), построение графиков (Graph), дополнительные возможности (Options) и работа с окнами дисплея (Window).

Второе – предназначено для ввода с клавиатуры и редактирования условий задачи (Edit). В этом окне записана задача "о сукне". Точка с запятой (;) означает начало комментария (вспомним слово REM или апостроф на Бейсике), который может содержать и буквы кириллицы. Хуже обстоит дело с текстом задачи – здесь допустимы лишь латинские и греческие буквы. Впрочем, в данном случае особых знаний английского не требуется – задача простая, поэтому достаточно записать систему линейных алгебраических уравнений с двумя неизвестными: blue – количество синего сукна, black – черного.

В третьем окне формируется решение задачи (Solution). Сюда ЭВМ записывает переменные (Variables) и их вычисленные значения (Values), удовлетворяющие условиям задачи. По умолчанию Eureka присваивает переменным статус вещественных с восьмиразрядной мантиссой. При необходимости оговаривают, например, что переменные целочисленные или комплексные.

Четвертое – предназначено для проверки (верификации – Verify) полученного результата. Если будет отдана соответствующая команда, ЭВМ подставит значения переменных из окна Solution в формулы окна Edit и рассчитает разницу (difference) между левыми и правыми частями выражений. Заодно будут распечатаны исходные данные для их дополнительной проверки.

Пятое окно – это своеобразный компьютерный "бюрократ", составляющий отчет (Report) о решении задачи. По команде машина запишет в это окно дату и время решения задачи, название и место хранения файла с условием, само условие,

---

* File (англ.) – последовательность или совокупность однотипных данных.

```
Eureka: The Solver
File   Edit   Solve   Commands   Report   Graph   Options   Window

Edit
C:NONAME.        Line 2    Col 18
;          Задача "о сукне"
  blue  +  black = 138;  всего аршин сукна
5*blue + 3*black = 540;  стоимость покупки

Solution
C:SOLUTION.      Line 1
Solution:

Variables     Values

black       =     75.000000

blue        =     63.000000

Report
D:REPORT.        Line 3    Col 7
Wednesday February 28, 1990, 7:01 pm.
Name of input file: NONAME
*************************************
;          Задача "о сукне"

Verify
C:VERIFY.        Line 5
blue+black       =     138.0000
138              =     138.0000
  difference =       .00000000
5*blue+3*bla...  =     540.0000

F1-Help F2-Save F3-Load F5-Zoom F6-Next F7-Beg Blk F8-End Blk SCROLL-Size/move
```

Рис. 2.2. Задача "о сукне" в среде Eureka

решение и его проверку. Отчет можно распечатать на принтере или записать на винчестер или дискету. В нижней части экрана высвечивается подсказка – назначение функциональных клавиш, позволяющих одним нажатием отдать ту или иную команду. Например, клавишей F1 пользователь открывает на экране новое окно, в котором появляется справочная информация (Help*) о работе в среде Eureka. К исходной картинке возвращаются, нажав клавишу Esc**.

Описывать все возможные картинки среды Eureka – неблагодарное занятие. Каждый владелец компьютера без особых усилий познакомится с ними самостоятельно. Попробуем рассмотреть более общие вопросы, например, почему перед названием Eureka постоянно употребляется слово "среда".

Условие задачи "о сукне" находится в окружении (environment) возможных команд пакета программ Eureka, которые в совокупности образуют среду. Но это условие нетрудно окружить иными командами, "погрузив" его в другую программную среду, например в текстовый процессор (Word, WordPerfect, Лексикон или какой-либо другой), в инструментальную систему (например, PCTOOLS), в среду новой версии Eureka и т.д.

Просматривая файлы на диске, легко заметить, что они, как правило, имеют не только имя, но и "фамилию" (расширение) (например, .exe, .txt, .doc, .hlp, .sys, .bas и т.п.). Фамилия файла подсказывает, в какой среде он был создан и где его можно использовать. Есть своя фамилия и у Eureka-файлов – .eka. В фирменном пакете Eureka обычно находится множество файлов с расширением .eka.

Для чего нужны эти файлы? Во-первых, просматривая файлы с формулировкой той или иной задачи, можно разобраться в методике ее решения в среде Eureka. Во-вторых, занимаясь конкретной практической задачей, достаточно вызвать с диска ее аналог (поиск корней системы уравнений, одномерная или многомерная оптимизация и т. д.), отредактировать его, т. е. подставить собственные данные, условия или ограничения, "прогнать" через режим Solve и получить ответ значительно быстрее, нежели при решении "от нуля".

Окна Report и Verify ограничены размером экрана и позволяют видеть лишь часть информации о ходе решения задачи и точности результатов (см. рис. 2.2). Чтобы заглянуть в любое из этих окон, его следует активизировать (через основное меню соответствующего окна), а затем раскрыть клавишей-командой F5 (Zoom***). О том, что окно активизировано, т. е. готово к тому,

---

* Help (англ.) – помощь, подсказка.

** Esc – от англ. to escape – убегать, исчезать.

*** Zoom (англ.) – увеличивать масштаб изображения, размер окна до размеров всего экрана.

чтобы его открыли, сигнализирует двойная рамка, (такая рамка на рис. 2.2 охватывает окно Report). Раскрыв окно, можно, например, дополнить текст отчета, а уж затем распечатать его.

Так, кстати, были подготовлены отчеты о решениях пяти демонстрационных задач, иллюстрирующих особенности работы в среде Eureka. Каждый из отчетов разбит на три зоны: заголовок, условие и решение задачи.

З а д а ч а 1. *На скотном дворе бегают куры и кролики. Всего у них 12 голов и 28 ног. Попробуйте определить, сколько кур и сколько кроликов.*

Записанное в среде Eureka условие этой простейшей задачи (рис. 2.3) почти не отличается от ее словесного описания. Добавились лишь два выражения, связывающие число ног и голов у живности. Машине предстоит решить систему двух линейных алгебраических уравнений.

```
*****************************************************
        Eureka:  The Solver, Version 1.0
      Name of input file: A:\ANIMALS.EKA
*****************************************************
;              Задача о курах и кроликах
golovy = 12;          На скотном дворе всего 12 голов
nogi   = 28;                                  и 28 ног.
;Спрашивается, сколько кур и кроликов это все имеют,
;          если у курицы и у кролика по одной голове,
1 * kury + 1 * kroliki = golovy
;           но у курицы две ноги, а у кролика четыре.
2 * kury + 4 * kroliki = nogi;  Вот и вся программа.
*****************************************************
Solution (Решение):
Variables     Values (Переменные    Значения)
 golovy      =     12.000000 (Головы)
 kroliki     =     2.0000000 (Кролики)
 kury        =     10.000000 (Куры)
 nogi        =     28.000000 (Ноги)
*****************************************************
```

Рис. 2.3. Решение задачи "о курах и кроликах" в среде Eureka

Технология решения этой задачи носит универсальный характер, поэтому ограничимся лишь общим описанием. Включите компьютер и запустите программу "eureka.exe". На экране появятся окна (рис 2.4): четыре нижних окна пустые, а первое "блюдо" меню (File – работа с файлами) выделено светящимся

фоном. Клавиши управления курсором позволяют выбрать из меню нужное "блюдо", а светящийся фон всякий раз отмечает этот режим. Чтобы его активизировать, достаточно нажать клавишу Enter (Ввод).

Все режимы обладают собственным меню услуг, как, например, режим работы с файлами (рис. 2.4):

Load – загрузка с диска в окно Edit старой задачи, стандартной или оригинальной, например поставленной ранее (если, конечно, на диске хранятся файлы с задачами);

New – очистка окон перед решением новой задачи;

Save – запись задачи на диск (с помощью этой команды специалист-прикладник может расширять библиотеку задач, решаемых в среде Eureka, постепенно превращая свой персональный компьютер в автоматизированное рабочее место).

Write to... – запись на диске задачи с новым именем (этот режим используют для сохранения новой модификации задачи или для записи опять-таки новой, но пока еще безымянной задачи (NONAME*) (см. рис. 2.2);

Directory** – просмотр содержимого дискет или винчестера, где файлы хранятся в виде "дерева" с ветвями-директориями;

Change dir*** – смена текущей директории;

Rename – переименование файла на диске;

OS shell**** – временный выход из среды Eureka в операционную систему, необходимый, например, для смены шрифта принтера (в свою очередь, выход из операционной системы по команде Exit означает возвращение в среду Eureka);

Quit – завершение работы со средой Eureka и возврат в операционную систему.

В то же время у некоторых подменю, в свою очередь, есть собственные подменю и т. д. Сколько различных вопросов может задать Eureka человеку, никто, по-видимому, точно не знает: как говорится, разработчик уже забыл, а пользователь еще не запомнил. Впрочем, не стоит даже стараться запоминать цепочки команд. Работая на клавиатуре компьютера или, еще лучше, с помощью "мыши", можно найти нужный режим, даже не имея большого опыта работы со средой Eureka.

Есть и исключения. Так, режимы Edit и Solve не имеют собственных меню. Если включить режим редактирования Edit (переместив светящийся прямоугольничек фона клавишами курсора, а затем нажав клавишу Enter), то можно будет запи-

---

* Noname – от англ. no name – без имени.

** Directory (англ.) – структура данных в файловой системе, элементы которой описывают файлы и обеспечивают доступ к ним по текстовым именам.

*** Change (англ.) – изменять.

**** Shell (англ.) – командный процессор.

```
                                  Eureka: The Solver
   File    Edit    Solve    Commands    Report    Graph    Options    Window

   Load                 Edit                                  Solution
   New                   Line 1    Col 1
   Save
   Write to..
   Directory
   Change dir                        C:\EKA\*.EKA
   Rename        CARDS.EKA     CARLOAN.EKA    CHEMBAL.EKA    CHEMEQ1.EKA
   OS shell      CHEMEQ2.EKA   CHEMEQ3.EKA    CLASSICS.EKA   COMPLEX.EKA
   Quit          CONFIG.EKA    DECAY.EKA      DERINT.EKA     EQUITY.EKA
                 FOSFAT.EKA    HELP.EKA       HOUSE.EKA      INTEG.EKA
                 LADDERS.EKA   LINECHG.EKA    LINPROG.EKA    LSQRFIT.EKA
                 MASSCHG.EKA   MAXMIN.EKA     POLYNOM.EKA    PRED.EKA
                 PROJCTL.EKA   ..\

   F1-Help  -Choose  <┘-Select  ESC-Abort  SCROLL-Size/move
```

Рис. 2.4. Просмотр файлов в среде Eureka

сать условие задачи (см. рис. 2.2). После того, как задача сформулирована, легко возвратиться в главное меню, нажав клавишу Esc. Чтобы заполнить решением окно Solution, достаточно отдать команду Solve. Точно так же заполняют окна Verify и Report.

Команды можно выбирать и по-другому: например, "утопив" клавишу Ctrl, нажимают клавишу, обозначенную той же буквой, что и заглавная буква названия режима. Еще проще работать с "мышью": передвигая ее по поверхности стола, достаточно установить стрелочку-указатель на название нужного режима и нажать левую кнопку "мыши".

З а д а ч а 2. *Найти координаты центра и радиус окружности на плоскости, проходящей через три точки с координатами (26,4), (9,21), (17,17).*

На рис. 2.5 представлен отчет о решении этой задачи в среде Eureka. Задача сводится к поиску корня системы трех нелинейных алгебраических уравнений. Эту систему нетрудно составить по теореме Пифагора (читатель помнит, что сумма квадратов катетов прямоугольного треугольника равна квадрату гипотенузы). Система имеет два решения: радиусы равны по модулю, но отличны по знаку. В подобной ситуации может случиться, что машина будет метаться между возможными решениями, не склоняясь ни к одному из них*. Избежать такой ситуации можно, если достаточно верно выбрать первое приближение, опираясь на которое машина быстрее и точнее найдет ответ.

Правые и левые части уравнений в условии разделяются знаками "=" или ":=". Это, своего рода, расстановка приорететов. Решая поставленную задачу, машина в первую очередь старается выполнить условия со знаком "=", а лишь по-возможности, и те, какие отмечены знаком ":=".

Поскольку задача может иметь несколько решений, из которых пригодно только одно, вычисления (Solve) всегда следует контролировать (Verify). Отчет о решении на рис. 2.5 завершается проверкой отклонений левых и правых частей уравнений.

Задачу можно усложнить, например в нашем случае заставить машину работать не с тремя, а с четырьмя или более точками на плоскости. Компьютер при этом найдет (или, по крайней мере, попытается найти) координаты окружности, которая наилучшим образом вписывалась бы в заданные точки. Такая особенность среды Eureka – ее несомненное достоинство, но в то же время это может привести к ошибке.

---

* Здесь может проявляться парадокс, известный под названием "Буриданов осел" (осел, помещенный на равном расстоянии от двух одинаковых охапок сена, умирает от голода, ибо не может выбрать ту или иную связку).

```
*****************************************************
        Eureka: The Solver, Version 1.0
       Name of input file : C:\OKRUG.EKA
*****************************************************
;              Задача о поиске координат окружности
;(cx,cy и R),проходящей через три точки на плоскости.
x1=26 : y1=4;                  Координаты первой точки.
(x1-cx)^2+(y1-cy)^2=R^2;Вытекает из теоремы Пифагора.
x2=9  : y2=21;                 Координаты второй точки.
(x2-cx)^2+(y2-cy)^2=R^2;     Второе уравнение системы.
x3=17 : y3=17;                Координаты третьей точки.
(x3-cx)^2+(y3-cy)^2=R^2;     Третье уравнение системы.
;        Ход решения системы нелинейных алгебраических
;       уравнений сильно зависит от начальных заданных
;           значений неизвестных.  Пусть поэтому будет
cx:=(x1+x2+x3)/3;           ордината центра окружности,
cy:=(y1+y2+y3)/3;                          ее абсцисса и
R:=abs(x1+x2)/2 ;   радиус в первом приближении. Все!
*****************************************************
Solution (Решение):
       Variables     Values
       cx          =     2.0000000
       cy          =    -3.0000000
       R           =     25.000000
       x1          =     26.000000
... (Далее повтор исходных данных)
      Maximum error is 2.2737368e-13
Warning: root or log of negative number.(Предупрежде-
ние: брался корень или логарифм от отрицательн.числа)
Evaluation of formulas (Проверка формул):
       Formulas            Values (Формулы    Значения)
... (x1, y1)
       (x1-cx)^2+(y... =      625.00000
       R^2             =      625.00000
         difference (error) = 2.2737368e-13
... (x2, y2)
       (x2-cx)^2+(y... =      625.00000
       R^2             =      625.00000
         difference =      .00000000
... (x3, y3)
       (x3-cx)^2+(y... =      625.00000
       R^2             =      625.00000
         difference =      .00000000
       cx              =      2.0000000
       cy              =     -3.0000000
       R               =      25.000000
      Maximum error is 2.2737368e-13
*****************************************************
```

Рис. 2.5. Решение задачи "о координатах окружности" в среде Eureka

С одной стороны, Eureka позволяет решать даже некорректно поставленные задачи. При этом машина способна находить какие-либо удовлетворительные решения (например, сглаживания табличной зависимости аналитической функцией). С другой – может получиться результат, далекий от оптимального.

И еще одна особенность. Систему трех нелинейных уравнений, описывающих задачу об окружности, математик легко сведет к системе линейных уравнений (например, вычитая первое нелинейное уравнение из второго). Среда Eureka тем и хороша, что позволяет решать задачи в их, так сказать, первозданном математическом описании, не прибегая к упрощениям, которые нередко делают неясной физическую суть задачи.

З а д а ч а 3. *Из круглой жестянки сделать пожарное ведро максимальной вместимости.*

Технология изготовления пожарного ведра очень проста: из круглой заготовки вырезают сектор, полученную выкройку сворачивают в конус, а шов проваривают (клеят или паяют). Такое ведро с водой на пол не поставишь, что сделано для того, чтобы использовать его можно было лишь по прямому назначению – для тушения пожара.

Обозначив буквой *a* угол сектора (вырезки), а буквой *V* объем ведра, задачу можно сформулировать и решить без компьютера. Для этого необходимо вывести зависимость, которая связывала бы объем ведра-конуса и угол вырезки-сектора. Если взять от этого выражения первую производную, а затем найти корень, то получим решение задачи. Проверить, отвечает ли оно цели задачи, поможет вычисление второй производной. Жестянщик, которому поручат изготовить пожарное ведро, скорее всего не знаком с теорией математического анализа. В то же время, система Eureka могла бы найти интересующий мастера ответ (рис. 2.6).

Пусть диаметр заготовки равен одному метру. Впрочем, это условие выдвигать необязательно, поскольку оптимальный угол вырезки не зависит от диаметра заготовки. Кроме того, машина автоматически присвоила бы переменной D единичное значение (здесь работает принцип умолчания), подобно тому, как в Бейсике обнуляется любая переменная, значение которой не определено явно.

На следующем этапе, исходя из здравого смысла, устанавливается допустимый диапазон, в котором машине разрешается изменять угол *a*, чтобы не попасть в отрицательную область или в область больших углов (то и другое явно не имеет физического смысла).

Еще два выражения вытекают из несложной геометрии круга и конуса: дуга выкройки становится окружностью основания конуса, а высота конуса, радиус его основания и радиус заготовки

связаны теоремой Пифагора. Число π в первом выражении можно сократить, но тогда потерялась бы наглядность, если рассуждать с позиций геометрии. Хорошо это или нет, но общение со средой Eureka постепенно вырабатывает у человека так называемый KISS-принцип*.

```
*********************************************************
          Eureka: The Solver, Version 1.0
     Name of input file: C:\EKA\TEMPO\VEDRO.EKA
*********************************************************
;                    Задача о проектировании конусообразного
;                    пожарного ведра с максимальным объемом
D = 1;Диаметр круглой заготовки,из которой вырез.сектор.
0 < a < 360;                Диапазон изменения угла вырезки.
3.14*d = 3.14*D*(360-a)/360;  Длина окружности основания
;ведра-конуса (d) равна длине дуги заготовки с вырезкой.
(D/2)^2 = h^2 + (d/2)^2;        Радиус заготовки, высота и
;              радиус основания ведра-конуса - это стороны
;                              прямоугольного треугольника.
V(a) = 1/3*3.14*d^2/4*h;                Объем ведра-конуса.
V = V(a);                            Анализируемая функция.
$max(V);  Установка на поиск максимального объема ведра.
tan(3.14*A/180/2) = d/2/h;Подсчет угла раскрытия конуса.
*********************************************************
Solution (Решение):
Variables     Values (Переменные   Значения)
 a     =     66.061930 (Угол вырезки в заготовке, градусы)
 A     =     109.52675 (Угол раскрытия конуса, градусы)
 D     =     1.0000000 (Диаметр заготовки)
 d     =     .81649464 (Диаметр основания ведра-конуса)
 h     =     .28867651 (Высота ведра-конуса)
 V     =    .050357773 (Объем ведра-конуса)
Confidence level =    99.4% (Доверительный уровень)
All constraints satisfied.(Все ограничения выполнены)
*********************************************************
```

Рис. 2.6. Решение задачи "об оптимальном пожарном ведре" в среде Eureka

Если определить из первого выражения диаметр основания конуса, а из второго – его высоту, то нетрудно рассчитать объем конуса $V = V(a)$. Затем поручаем машине максимизировать эту величину, варьируя значение угла сектора.

З а д а ч а 4. *В городе два продовольственных склада и две пекарни. С первого склада необходимо ежедневно вывозить 50 т муки, а со второго – 70 т. При этом первая пекарня*

* От англ. "Keep it simple, stupid!" – делай это проще, дурачок.

должна ежедневно получать 40 т муки, а вторая – 80 т. Известны затраты на перевозку 1 т муки со складов на пекарни. Как оптимально спланировать работу транспорта?

Минимум затрат на перевозку муки за сутки составляет 134 руб. (рис. 2.7). Как ни странно, но в первую пекарню со второго склада муку не завозят (M3=0), хотя такие перевозки самые дешевые – 80 копеек за тонну. В условиях валовой экономики с ее девизом "План любой ценой" этим парадоксом может воспользоваться плановик автокомбината. Стоит слово min в условии задачи заменить на max, и затраты в сутки увеличиваются до 142 руб. Внешне все выглядит достаточно прилично – M1=0, M2=50, M3=40, M4=30, т. е. в первую пекарню муку завозят только со второго склада.

Возможны и другие "маленькие хитрости". Например, дополнительный "выигрыш" автокомбинат может получить, используя встречные перевозки. Для этого достаточно в условии задачи снять ограничения M1>0, M2>0 и т. д.

З а д а ч а 5. *Мебельной фабрике поручено выпускать стулья двух типов ценой по 8 и 12 руб. Под этот госзаказ выделены необходимые ресурсы: 440 погонных метров досок, 65 м^2 обивочной ткани, 320 человеко-часов. Известно, сколько досок, ткани и времени идет на изготовление каждого стула. Требуется так спланировать производство стульев, чтобы получить наибольшую выручку* (рис. 2.8).

Если фабрика сделает 60 стульев по 8 руб. и 80 – по 12 руб., будут израсходованы все доски, исчерпан ресурс трудового времени, но останется 15 м^2 обивочной ткани.

Как и в предыдущей задаче, Eureka позволяет "поиграть" с условиями, например отказаться от максимума прибыли, но зато сделать больше стульев. Те же самые ресурсы позволяют собрать 150 стульев вместо 140. Из них 110 будут дешевыми, а 40 – по 12 руб.

Среди функций среды Eureka, извлеченных из файла help.eka с помощью редактора инструментальной системы The Norton Commander (еще один очень популярный программный продукт), кроме обычных функций (синусы, косинусы, логарифмы и т. п.), можно увидеть и более сложные: производная (deriv), интеграл (integ), нормальное распределение (ncum), сумма ряда (sum) и расчет сложных процентов (fval, paymt, pval) (рис. 2.9).

Эти функции иной раз позволяют легко решать довольно сложные задачи, но не стоит обольщаться: серьезная проблема, связанная с решением десяти или более алгебраических, дифференциальных или интегральных уравнений часто оказывается "не по зубам" среде Eureka. В подобных ситуациях Eureka расходует лимит времени, отведенный на решение и,

```
*****************************************************
        Eureka:  The Solver, Version 1.0
      Name of input file: C:\EKA\TRANS.EKA
*****************************************************
;                    Транспортная задача.
;     В городе есть два склада муки и две пекарни.
;Ежедневно с 1-го склада нужно вывезти 50 т муки,
M1 + M2 = 50
;                                а со второго - 70 т.
M3 + M4 = 70
;                 При этом 1-я пекарня получает 40 т,
M1 + M3 = 40
;                                     а вторая - 80 т.
M2 + M4 = 80
;                  С пекарен на склады муку не везут:
M1 > 0  :  M2 > 0  :  M3 > 0  :  M4 > 0
;   Стоимость перевозки муки (руб/т)
;        .      Пекарня 1   Пекарня 2
;   Склад 1       1.2          1.6
;   Склад 2       0.8          1.0
;    Целевую функцию (затраты на перевозку муки)
zatraty = 1.2* M1 + 1.6* M2 + 0.8* M3 + 1* M4
$min(zatraty);                нужно минимизировать.
*****************************************************
 Solution (Решение):
 Variables     Values (Переменная  Значение)
 M1 = 40.000000 (С 1-го склада на 1-ю пекарню)
 M2 = 10.000000 (С 1-го склада на 2-ю пекарню)
 M3 = .00000000 (Со 2-го склада на 1-ю пекарню)
 M4 = 70.000000 (Со 2-го склада на 2-ю пекарню)
 zatraty = 134.00000 (рубля в сутки)
 Confidence level =   99.4%
 All constraints satisfied.
 *****************************************************
```

Рис. 2.7. Решение транспортной задачи в среде Eureka

```
*************************************************
      Eureka: The Solver, Version 1.0
     Name of input file: B:\LINPROG.EKA
*************************************************
;        Задача планирования производства
;       На фабрике делают стулья двух типов
    stul1 > 0        :        stul2 > 0.
;     Далее таблица с линейными неравенствами:
;                        Идет на стул 1 Стул 2   Ресурс
;       Доски (м^2)              2       4          440
2 * stul1  +   4 * stul2 <= 440
;       Обивочная ткань (м^2)    .5     .25          65
.5 * stul1 + .25 * stul2 <=  65
;      Трудозатраты (чел.-часы) 2       2.5         320
2 * stul1  + 2.5 * stul2 <= 320
;       Стоимость (руб)          8       12          ---
8 * stul1  + 12 * stul2 = val;      Целевая функция
$ max (val)   ;   Ищется  максимум целевой функции
*************************************************
Solution (Решение):
 Variables    Values (Переменные  Значения)
val          =      1440.000 (Стоимость стульев,руб.)
stul1        =      59.99999 (Выпуск стульев  1 типа)
stul2        =      80.00000 (Выпуск стульев  2 типа)
Confidence level =    95.1%
All constraints satisfied.
Evaluation of formulas            (Проверка формул):
Formulas            Values       (Формулы  Значения)
2*stul1+4*st...  =      440.00000 (Реальный расход)
440              =      440.00000           (Фонды)
difference = -2.5466801e-07     (Опилки и стружка)
.5*stul1+.25...  =      50.000000 (Реальный расход)
65               =      65.000000           (Фонды)
difference = -15.000000 (Остаток - ресурс фабрики)
2*stul1+2.5*...  =      320.00000
320              =      320.00000
difference = -7.321270e-07(Людей заняли полностью)
  ...          Далее все тривиально и неинтересно.
*************************************************
```

Рис. 2.8. Задача линейного программирования в среде Eureka

```
Edit: C:\eka\help.eka                    *  Line 1     Col 57      1,577 Free    EOL
-------------------------------------------------------------
              Функции EUREKA (HELP FILE.)
-------------------------------------------------------------
 abs(x)  Absolute value              atan2(y,x) Arc Tangent
 cos(x)  Cosine                  cosh(x)  Hyperbolic Cosine
 deriv(f(x),x)  Derivative              exp(x)  Exponential
 fact(n)  Factorial of constant             floor(x)  Floor
 fval(rate,time,pres_value,payment)  Future Value
 im(x)  Imaginary part
 integ(f(x),x,lower_bound,upper_bound)  Integral
 ln(x)  Natural Logarithm  log10(x)  Logarithm (base 10)
 msqrt(x)  Minus Square Root
 ncum(x)  Normal Cumulative Distribution
 paymt(rate,time,pres_value,future_value) Payment
 polar(x,y) Polar Coordinate Conversion
 poly(x,...)  Polynomial              pos(x)  Positive Part
 pval(rate,time,payment,future_value)   Present Value
 re(x)  Real Part          sgn(x)  Sign          sin(x) Sine
 sinh(x)  Hyperbolic Sine              sqrt(x)  Square Root
 sum(f(x),x,first,last)  Sum                tan(x)  Tangent
 tanh(x)  Hyperbolic Tangent
-------------------------------------------------------------

1Help   2Save   3      4      5      6      7Search 8      9      10Quit
```

Рис. 2.9. Встроенные функции среды Eureka

как бы извиняясь, выдает какой-либо приближенный результат. Человек должен сам решить, что делать с этим вариантом дальше. Для этого следует прежде всего проверить результат в режиме Verify.

Успех решения во многом определяется начальными условиями (выбором так называемой опорной точки). Умение выбирать эти условия зависит от квалификации специалиста по решаемой проблеме. Точный выбор как бы подталкивает машину к правильному решению.

Завершая разговор о среде Eureka, нельзя не подчеркнуть, что Eureka – это, в первую очередь, инструмент специалиста-непрограммиста. Другое применение – сфера обучения. Здесь на первый план выходят графические возможности среды. На рис. 2.10 показано окно – Plot*, в котором легко строить различные графические объекты, используя встроенные функции или выражения из окна Edit. Так можно значительно лучше понять сущность решаемой задачи, подобрать приемлемые начальные условия (решение прикладных задач) и, что немаловажно, красиво оформить отчет о решении.

Вернемся в заключение к задаче 3 об оптимальном пожарном ведре (см. рис. 2.6). Прежде чем искать угол вырезки *a*, можно построить график изменения объема ведра от этого угла. "Горб" на графике подскажет точку или диапазон первого приближения, что существенно ускорит и упростит расчет.

Заканчивая разговор о среде Eureka, перечислим и некоторые другие подобные пакеты "программирования без программирования": MATCAD, MATLAB, FAZER, MUSIMP, REDUCE.

"ОКНА" В МИР ИНФОРМАЦИИ

Это заглавие – образный перевод с английского названия одного из самых известных программных продуктов Framework III, разработанного для персонального компьютера не менее известной фирмой Ashton-Tate. Перевод, близкий к дословному, гласит: "работа с обрамленной информацией, третья версия". Что же это за "обрамленная информация"?

На "слепке" с экрана в тот момент, когда компьютер работает под управлением среды Framework III, выделяется заштрихованная область (рис. 2.11). Это так называемый стол (desk).

Верхняя часть экрана содержит ключевые слова, отмечающие "блюда" главного меню. Такая разбивка экрана традиционна. Вспомним хотя бы пакет Eureka (см. рис. 2.2).

---

* Plotter (англ.) – графопостроитель.

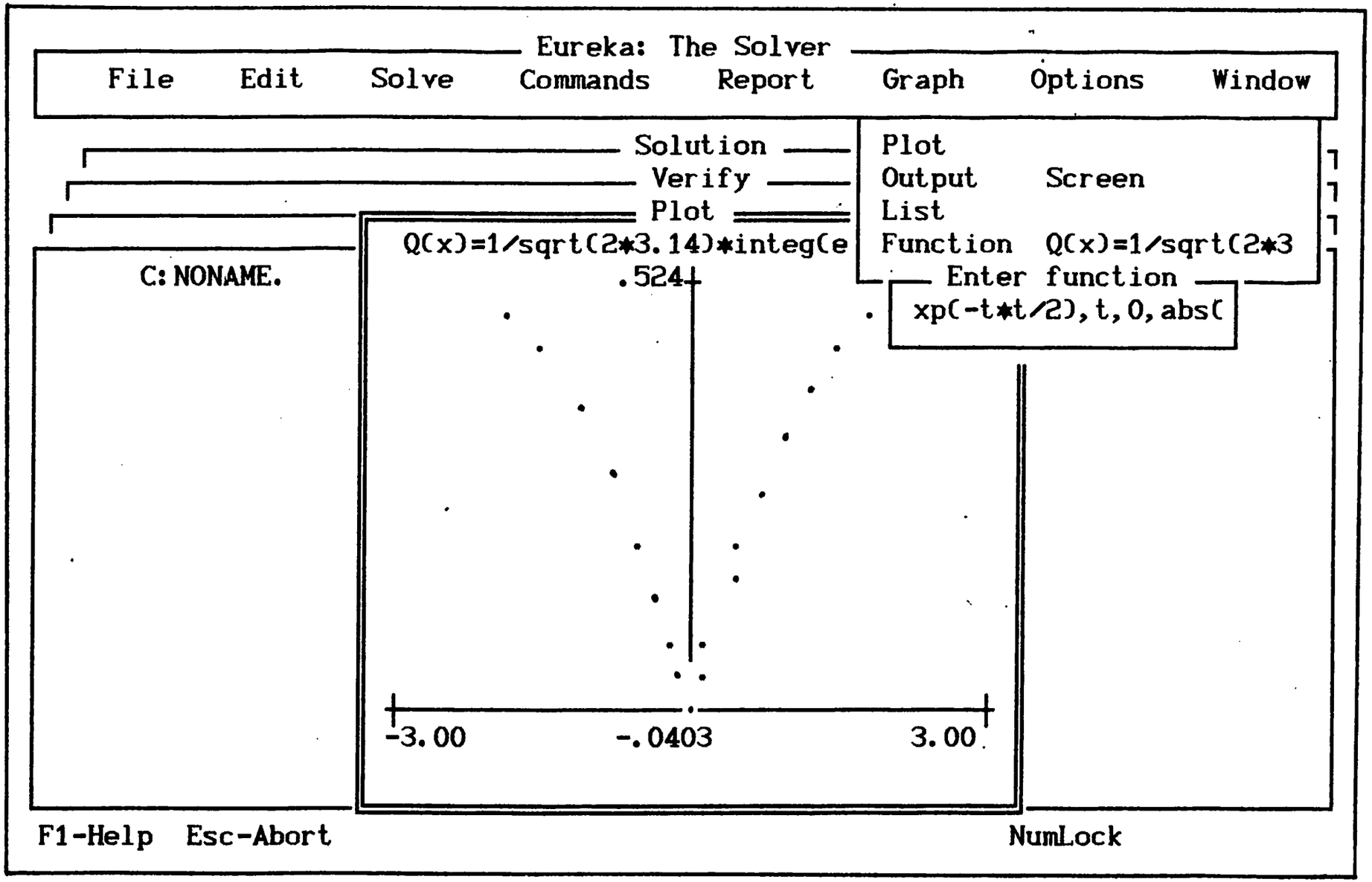


Рис. 2.10. Построение в среде Eureka графика функции

Apps Disk Create Edit Locate Frames Words Numbers Graph Print 11:56 am

Outline
Empty / Word Frame

Spreadsheet
Database
Width (# Cols/Fields) {4}
Height (# Rows/Records) {17}
Columns/Fields: Add {1}
Rows / Records: Add {1}

Macro / Abbreviation

3>

HELP3.DAT	46K
HELP4.DAT	2K
SETUP	17K
2.RES	37K
237A1.RES	5K
.236	26K
BRARY.FW3	31K
ADME	2K
NDBASE.FW3	5K
MPLE1.FW3	1K

<Library>

<A:>
>C:<

<A:\>

ANIMALS.EKA	1K
ANIMALS.RPT	1K
BIOFAB.LET	2K
FOSFAT.EKA	2K
HELP.EK	2K
KALIN1.LET	1K
KALPLAN.TXT	2K

A:\ Cabinet: 2/3 NUM

Create a spreadsheet of specified Width and Height

Рис. 2.11. Создание новой электронной таблицы в Framework III

На столе расположены три окна (Frame), которые открыл пользователь. Справа в двух окнах показано содержимое диска A (программисты сказали бы "каталог дискеты") и диска C ("каталог винчестера"). Каталоги в окнах содержат только названия файлов. Левое окно – это вспомогательное меню или подменю. В нем указано, какую работу можно поручить машине, создавая новые информационные объекты (позиция Create главного меню).

Подробное описание Framework III – дело долгое и неблагодарное. Работая в этой программной среде, пользователь всегда имеет доступ к исчерпывающей информации о том, как следует обратиться к машине, чтобы она помогла выполнить задуманную работу. Для этого на клавиатуре есть клавиша F1 (help – помощь). Стоит ее нажать и необходимая информация появится на экране.

Ограничим нашу встречу с Framework III знакомством с так называемыми электронными таблицами на примере двух конкретных задач.

Одну задачу мы уже решали на программируемом микрокалькуляторе (см. главу 1), а теперь попробуем проверить ответ с помощью персонального компьютера.

З а д а ч а. *На складе имеются резервуары четырех типов: круглый цилиндр длиной L, радиус сечения которого R* (см. рис. 1.13, *а*);

цилиндр длиной L, сечение которого – эллипс с большим радиусом a и малым радиусом b (см. рис. 1.13, *б*);

сфера радиуса R (см. рис. 1.13, *в*);

прямоугольный параллелепипед длиной L и шириной d (см. рис. 1.13, *г*).

Определить объем жидкости в резервуарах.

Проведем расчет на персональном компьютере в среде электронных таблиц.

Электронные таблицы предназначены прежде всего для обработки данных матричного характера (на это, кстати, намекает и название "Spreadsheet" – "распростертый лист бумаги", "простыня"). Типичный пример таких данных – годовая сводная ведомость заработной платы, где слева (в боковике) записаны фамилии работников, а в "шапке" (головке) – месяцы или кварталы. В нижней части подобной таблицы, как правило, какие-то суммарные или средние цифры, подытоживающие данные в графах. Соответствующую электронную таблицу легко построить в среде Framework III.

Кроме того, электронные таблицы позволяют изящно решать задачи, не имеющие выраженного табличного характера (например, наша задача – определение объема резервуаров).

Вернемся к рис. 2.11. На нем зафиксировано создание электронной таблица с заданной шириной и высотой (об этом свидетельствует надпись в нижней части рисунка). Электронная таблица появится на экране после нажатия клавиши RETURN. Подменю режима Create показывает, что пользователь дал машине указание создать новую электронную таблицу шириной (Width) в четыре столбца (Cols*) и высотой (Heigth) в 17 рядов (Rows).

Как правило, пользователь заранее не знает точных размеров готовой таблицы, поэтому машина оставляет ему возможность вставить (Add в режиме Create) или удалить (Remove в режиме Edit) любое число столбцов и рядов в процессе формирования таблицы.

Не случайно на рис. 2.11 оказались окна, показывающие содержимое архивной памяти ЭВМ (диски A и C). Дело в том, что созданию новой электронной таблицы обычно предшествует поиск ее аналога. Часто оказывается, что намного проще переделать похожую старую таблицу, предварительно скопировав ее с диска, нежели создавать новую (например, сводная ведомость зарплаты за февраль 1990 г. лишь незначительно отличается от такой же ведомости за январь этого же года).

Электронные таблицы получили свое название не только и не столько потому, что они реализованы на ЭВМ, а прежде всего из-за того, что их графы могут содержать как введенные числа и символы, так и значения, рассчитываемые машиной по формулам. Если пользователь изменит хотя бы одно число-аргумент, то его функции автоматически примут новые значения. Эта особенность отличает электронные таблицы от тех, которые можно создавать с помощью текстового процессора, превращающего ЭВМ в пишущую машинку с памятью.

Какие функции удается реализовать, работая с электронной таблицей пакета Framework III? Описать все из них практически невозможно прежде всего потому, что таких функций очень много: числовые, литерные, статистические, финансовые и др. По-видимому, никто не знает точного их количества, как и в случае с командами среды Eureka: разработчик уже забыл, а пользователь еще не запомнил. Впрочем, запоминать функции не требуется, поскольку пакет Framework III предусматривает обширные комментарии (help), где легко отыскать соответствующую подсказку.

Если к работе с пакетом Framework III только что приступили, то после нажатия клавиши F1 (help – помощь), на экране появляется картинка, копия которой приведена на рис. 2.12. Здесь собраны названия статей (INDEX OF TOPICS) с необходимой информацией.

---

* Сокращение от Columns (англ.) – колонки, столбцы.

```
Help                          INDEX OF TOPICS                          12 30 pm

                Learning Framework III
                Key Commands
                Using a Mouse for Navigation and Commands
                Menu Options
                Disk/Filing Operations
                Outlines
                Word Processing
                Spreadsheets and Databases
                Graphs
                Printing
                Customizing Framework III
                Networks and Mail
                Telecommunications
                Building Blocks and Syntax (FRED Programming Language)
                Index of FRED Functions (Alphabetical)
                FRED Functions (By Category)
                Acknowledgments

        Select a topic with ← or → and press RETURN;    Press ESC to exit
```

Рис. 2.12. Оглавление "помощи" пользователю при работе с Framework III

Перечислим их, снабдив каждую короткой аннотацией, чтобы составить хотя бы общее представление о пакете.

Изучение пакета Framework III (Learning ...). Статья с общими сведениями, касающимися пакета.

Ключевые команды и команды ключей (Key ...). На клавиатуре, кроме обычного набора клавиш, есть 10 (или 12) дополнительных, так называемых функциональных, клавиш-ключей F1 – F10. Работая в той или иной программной среде, они несут различную смысловую нагрузку. Например, F1 – вызов информационной помощи, остальные клавиши могут означать копирование, перенос, изменение размеров окон и столбцов таблиц и т. д.

Использование "мыши" для путешествия в мире информации и выполнения команд (Using ...). Умение работать с "мышью" – это одна из особенностей, отличающих пакет Framework III от его более ранних версий. Если у ЭВМ есть "мышь", то процесс общения с машиной изменяется коренным образом. Для выполнения той или иной команды не требуется нажимать клавиши клавиатуры в определенной комбинации (кстати, запомнить возможные комбинации весьма непросто). Необходимо немало поработать на ЭВМ, чтобы знать, например, что клавишу "+" нажимают для вхождения в окно, а "—" – для выхода. Вместо этого достаточно передвинуть "мышь" по поверхности стола так, чтобы управляемый ею курсор указывал на выбранное поле (к примеру, на надпись "Using a Mouse for Navigation and Commands"), и нажать левую кнопку "мыши".

Нижняя строка на рис. 2.12 подсказывает иной – "безмышиный" способ диалога с помощью клавиш управления курсором. В этом случае процесс управления складывается из нескольких действий: подвести курсор к выбранной строке надписи (при этом, она будет выделена) и нажать клавишу RETURN (Ввод). Клавиша ESC отменяет ранее заданную команду. Аналог этой клавиши – правая кнопка "мыши".

Дополнительные меню (Menu ...). Одно из них (для режима Create), показано на рис. 2.11. Подобные подменю есть и у остальных девяти режимов главного меню:

Apps (приложения, аппаратная часть компьютерной сети);

Disk (работа с дисковыми устройствами, временный и окончательный выход из Framework III в DOS и др.);

Edit (редактирование информационных объектов);

Locate (местонахождение);

Frames (обработка окон);

Words (обработка слов);

Numbers (обработка чисел);

Graph (построение графиков);

Print (распечатка на принтере).

Правее этого слова зафиксировано время, когда была нажата клавиша Print Screen (печать экрана) для получения рис. 2.11 – 11 ч 56 мин до полудня (am).

Вызвать то или иное подменю можно либо с помощью "мыши", либо, как и при работе в среде Eureka, одновременным нажатием управляющей клавиши (Ctrl) и клавиши с заглавной буквой режима: A, D, C, E, L, F, W, N, G или P.

Работа с файлами и дисководами (Disk ...). Это один из наиболее часто используемых режимов. Освоить его несложно, а для того, чтобы описать работу с ним, потребовалось бы слишком длинное отступление.

Набросок, план текста (Outlines ...). В этом режиме можно создавать и редактировать структурированные тексты, подобные книге, разбитой на главы и разделы. Команда Outline позволяет записать названия структурных составляющих текста, чтобы по ним можно было легко и быстро найти нужный фрагмент.

Текстовый процессор (Word ...). Позволяет превратить ЭВМ в интеллектуальную пишущую машинку. Статистика показывает, что персональные компьютеры бóльшую часть своего рабочего времени используются именно в этом качестве.

Электронные таблицы и базы данных (Spreadsheets ...). О таблицах рассказ пойдет чуть ниже. Базы данных не менее интересны, но объем книги не позволяет их рассмотреть.

Графики (Graphs ...). В этом режиме можно строить на экране дисплея разнообразные графики.

Печать (Printing ...). Полученные на экране информационные объекты нередко требуется извлечь из машины так, чтобы их можно было бы в любой момент посмотреть или кому-либо показать. Таблицы, графики, тексты и любую другую информацию можно отпечатать на принтере, нарисовать на графопостроителе и затем отослать по почте. Впрочем, пакет Framework III может обеспечить более современную пересылку информации. В компьютеризированном бюро документы печатаются только для архива, а для пересылки используется электронная почта.

О продаже пакета Framework III (Customizing ...).

Сети и почта (Networks ...). В этом режиме персональный компьютер можно превратить в интеллектуальный терминал компьютерной сети.

Телекоммуникации (Telecommunications ...). Пакет Framework III позволяет перекачивать информацию из файлов, созданных в других программных средах (например, в текстовых процессорах, базах данных, ранних версиях пакета Framework и т. д.).

Образование блоков и синтаксис языка FRED (Building ...). Команды этого языка позволяют управлять информационными объектами.

Заголовки функций языка FRED в алфавитном порядке (Index ...).

Функции языка FRED, рассортированные по типам (FRED ...). Описаны статистические, финансовые и другие функции.

Прочие уведомления для пользователей среды Framework III (Acknowledgements ...).

Из этой краткой характеристики видно, что пакет Framework III, позволяет создавать информационные объекты четырех типов: структурированный текст, электронная таблица, деловая графика и база данных.

Вернемся к задаче об объеме резервуаров. Для заполнения соответствующими формулами граф электронной таблицы необходимо знать числовые функции языка FRED. Вновь воспользуемся клавишей F1 (help – помощь). Традиционный набор

числовых функций, который с теми или иными небольшими изменениями есть в любом языке программирования, будь то Бейсик, Рапира, Паскаль, Фортран, ПЛ/1 и т. д., можно увидеть на рис. 2.13. Читатель может сравнить функции электронных таблиц пакета Framework III (см. рис. 2.13) с функциями пакета Eureka (см. рис. 2.9). Многие функции, перечисленные на этих рисунках, есть и у микрокалькуляторов.

Две последние строки рис. 2.13 подсказывают пользователю, как путешествовать по описанию пакета Framework III:

просмотр предыдущей страницы "помощи" (Previous Screen);

просмотр следующей страницы "помощи" (Next Screen);

вызов на экран названий статей "помощи" (Index of Topics – см. рис. 2.12);

справка о "нагрузках" функциональных клавиш клавиатуры (Key Help).

Ознакомившись с текстом "подсказки", нетрудно вернуться к работе, нажав клавишу ESC.

Кратко перечислим операции, необходимые для создания электронной таблицы.

Для того чтобы войти в программную среду Framework III, запускают программу "fw.exe", которая обычно хранится на винчестере. Прежде, чем создавать новую таблицу, что называется "от нуля", необходимо проверить, нет ли в архивной памяти похожей электронной таблицы. Если есть, то ее достаточно отредактировать.

Однако найти подходящую таблицу удается не всегда. В этом случае необходимо вызвать подменю режима Create (создать) главного меню и скорректировать цифры, отмечающие ширину и высоту новой таблицы. Этот этап зафиксирован на рис. 2.11. Рамка, которой управляют "курсорные" клавиши, обрамляет слово Spreadsheet, поэтому после нажатия клавиши RETURN на экране дисплея появится новая, пока что безымянная и пустая, таблица со столбцами, промаркированными от A до D, и рядами, пронумерованными от 1 до 17. Если заполнить ее и дать файлу имя "reserv", получится таблица, приведенная на рис. 2.14.

Чем же заполнена эта таблица? Ряды 1 и 2 хранят литерную константу – заголовок таблицы. Ряды 3 и 5 – элементы украшения. Ряд 4 отмечает названия параметров, собранных в таблице. Особого внимания заслуживает столбец D, заполненный не литерными (как в других случаях), а числовыми переменными. При этом графы таблицы D6 – D12 хранят независимые переменные, которые пользователь может изменять, а графы D14 – D17 – значения функций, заданных пользователем и вычисленных компьютером. Одна из них (в данном случае формула для расчета объема остатка жидкости в эллиптическом цилиндре) представлена в предпоследней строке рис. 2.14.

```
Help                    THE FRED LANGUAGE: Numeric Functions                12:40 pm

  @int(n)           Gives integer part of n; @int("chr") gives ASCII value
  @round(n, dp)     Rounds n to number of decimal places specified by dp
  @floor(n)         Gives the floor of n (largest integer not greater than n)
  @ceiling(n)       Gives the ceiling of n (smallest integer not less than n)
  @abs(n)           Gives the absolute value of n (converts - to + numbers)
  @sign(n)          Returns +1 if n is positive, -1 if negative, 0 if n is 0
  @mod(n, div)      Gives the modulus of n divided by div
  @sqrt(n)          Gives the square root of n
  @rand no param    Generates a random number 0-1 to 15 decimal places
  @log(n)           Gives the base 10 logarithm of n
  @ln(n)            Gives the natural logarithm of n
  @exp(n)           Gives e (2.7182818...) to the exponent of n
  @sin(n)           Gives the sine of n, where n is an angle (in radians)
  @cos(n)           Gives the cosine of n
  @tan(n)           Gives the tangent of n, where n is an angle (in radians)
  @asin(n)          Gives the arc sine of n
  @acos(n)          Gives the arc cosine of n
  @atan(n)          Gives the arc tangent of n
  @atan2(n1, n2)    Gives the 4-quadrant arc tangent of n1 and n2
  @pi no param      Gives the value of pi (3.1416...) to 15 digits

    Previous Screen      Next Screen      Index of Topics      Key Help
     Choose an option with ← or → and press RETURN;   press ESC to exit
```

Рис. 2.13. Числовые функции языка FRED пакета Framework III

Apps Disk Create Edit Locate Frames Words Numbers Graph Print 5:13 pm

<C:\FW3> Treserv Решение задачи о резервуарах. Пакет Framework IIII <Library>

	A	B	C	D
1	Решение задачи об объеме остатка жидкости в резервуарах			
2	Среда электронных таблиц (Spreadsheet) пакета Framework			
3	*----------*			
4	Параметр	Обозначение	Размерность	Значение
5	*----------*			
6	Длина резервуара	L	м	5
7	Большой радиус эллипса	a	м	5
8	Малый радиус эллипса	b	м	2.50
9	Радиус цилиндра	R	м	5
10	Радиус сферы	R	м	1
11	Ширина параллелепипеда	d	м	2
12	Высота слоя жидкости	h	м	.12300
13	Объем жидкости:	V	м^3	
14	в эллиптическом резервуаре		м^3	16.00473
15	в цилиндрическом резервуаре		м^3	22.40801
16	в сфере		м^3	.04558
17	в параллелепипеде		м^3	1.23000

<A:> >C:< <A:\> Treserv 1

D6*D7*D8*@acos((D8-D12)/ ax. Пакет Framework IIII.D14 Sfera 0/1

D6*D7*D8*@acos((D8-D12)/D8)-(D8-D12)*@sqrt(D7*D7*(1-(D8-D12)^2/D8/D8))

Formula/number editing; RETURN finishes

Рис. 2.14. Задача "о резервуарах" в электронных таблицах пакета Framework III

К редактированию формулы (именнно этот момент показан на рис. 2.14) приходится прибегать в двух случаях. Прежде всего это необходимо, когда обнаружена ошибка и ее требуется исправить. Значительно чаще приходится редактировать формулы, скопированные для ускорения работы из другой ячейки таблицы. Так, формулы объема остатка жидкости в цилиндрах двух типов (ряды 14 и 15) почти не отличаются, поэтому в графу D15 можно скопировать (тут поможет клавиша F8 (copy) формулы из графы D14). Потребуется только заменить переменные D7 – (a) и D8 – (b) на D9 – (R). Копировать можно не только отдельные графы, но и столбцы, ряды и целые прямоугольные зоны, выбирая их клавишей F6 (select). В ряде случаев машина при этом автоматически корректирует имена переменных, учитывая сдвиг по рядам и столбцам.

Заполнив таблицу константами и формулами, достаточно записать в графу D12 высоту остатка жидкости в резервуаре, чтобы можно было считать ответ в графах D14, D15, D16 или D17 в зависимости от того, какой резервуар нас интересует.

Графы электронной таблицы, кроме функций, способны хранить и альтернативные конструкции, например

"если (IF) верно заданное булево выражение, то делай одно, нет – делай другое".

Это может пригодится. и для нашей таблицы. С помощью альтернативы в таблицу можно вставить "защиту от дурака". Например, в таблице на рис. 2.14 нетрудно предусмотреть позицию, где будут появляться надписи "Исходные данные верны" или "Исходные данные неверны" в зависимости от значения введеного аргумента в графе D12 (ведь переменная в графе D12 не совсем независимая: она не может быть меньше нуля, а для сферы, к примеру, не должна превышать его диаметр (2R).

Если сравнить рис. 2.11 и 2.14, то легко заметить, что информационные окна, показывающие содержимое дисков A и C, переместились на экране влево. Пакет Framework III позволяет передвигать окна по дисплею (клавиша F3), изменять их размер (клавиша F4). При необходимости легко увеличить (или уменьшить) ширину столбцов электронных таблиц так, чтобы в их рамках умещалась длинная запись.

Вторая задача, которую мы решим с помощью электронных таблиц, имеет явно выраженный матричный характер и иллюстрирует некоторые статистические функции и графические возможности среды Framework III.

На рис. 2.15 приведена информация, представляющая интерес для тех, кто хочет узнать о финансовых и статистических функциях пакета Framework III. Аргументами являются не

```
Help      THE FRED LANGUAGE: Financial and Statistical Functions      12:39 pm

 Financial Function                        Calculates
 @pv(pmt, rate, periods)          Present value of payments
 @fv(pmt, rate, periods)          Future compound value of payments
 @npv(rate, flows)                Net present value of flows
 @pmt(prin, rate, periods)        Regular payment to amortize a loan
 @irr(guess, flows)               Internal rate of return of flows
 @mirr(risky, safe, flows)        Modified internal rate of return

 Statistical Function                      Action
 @avg(list)                   Averages the values in the list
 @count(list)                 Counts the numeric values in the list
 @max(list)                   Finds the maximum value in the list
 @min(list)                   Finds the minimum value in the list
 @std(list)                   Finds the standard deviation of values in list
 @sum(list)                   Sums the values in the list
 @var(list)                   Finds the variance of values in the list

 The statistical functions allow any number of parameters in their lists.
 Any non-numbers in a list are ignored.

   Previous Screen      Next Screen      Index of Topics      Key Help
    Choose an option with ← or → and press RETURN;   press ESC to exit
```

Рис. 2.15. Финансовые и статистические функции языка FRED пакета Framework III

отдельные переменные (значения, хранящиеся в графах таблицы), а списки (list), включающие столбцы, ряды и целые области, заключенные в прямоугольные рамки. Так, выражение sum(B2:F10) суммирует значения числовых переменных, хранящихся в рядах 2 – 10 столбцов B – F. При этом пустые зоны игнорируются.

На рис. 2.16 показан "слепок" с экрана, когда на нем отображается электронная таблица с именем "otpmkkpc", хранящая некоторые параметры одного из вариантов первой главы книги. В этой таблице особенно интересны результаты статистической обработки. На рис. 2.16 зафиксирован момент редактирования графы C15, в которой записано среднее (average) значение числа машинописных строк в первой главе рукописи.

Интересно и умение пакета Framework III строить графики и диаграммы в соответствии с содержимым электронных таблиц.

Весьма наглядна диаграмма (Bardiagram), показывающая изменение числа строк в разделах первой главы рукописи (рис. 2.17).

Пакет Framework III позволяет строить графические объекты и другого вида, например круговые диаграммы (их более образное название – Piediagram*.

Деловая графика помогает не "утонуть" в море цифр.

Сопоставив две простенькие задачи "о резервуарах" и "о книге" и потенциальные возможности пакета Framework III (занимающего, к слову сказать, 1 – 2 Мбайт на жестком диске), можно сделать вывод, что решение подобных задач в среде электронных таблиц – это "стрельба из пушки по воробьям". Действительно, справиться с решением помог бы даже программируемый микрокалькулятор.

Однако решение задач с помощью самостоятельного программирования на алгоритмических языках – это тупиковый путь инженерного и научного творчества. Рано или поздно рамки традиционных языков программирования для специалиста-непрограммиста, решающего прикладные задачи, станут "испанскими сапогами". Слишком уж легко запутаться в рутинных операциях отладки и тестирования программ, поиска и исправления ошибок, согласования работы периферии.

Искусство общения с ЭВМ – это не только (и не столько) программирование, сколько умение видеть характерные особенности задачи, позволяющие быстро выбрать для ее решения подходящие готовые программные средства. Мы попытались рассказать о двух из них: пакеты Eureka и Framework III.

---

* От a pie (англ.) – пирог.

```
Apps  Disk  Create  Edit  Locate  Frames  Words  Numbers  Graph  Print  3 40 pm
                                                                  <Library>
[otpmkkpc]
|        A                    B                      C          D       E
1   Характер 1 главы одного из вариантов рукописи книги "От ПМК к ПК"
2
3   N раздела     Название раздела          Число строк рисунков таблиц
4      1.1     Микрокалькуляторы вокруг нас     106        1       1
5      1.2     Что могут и чего не могут ПМК     78        0       0
6      1.3     Язык ПМК                         101        3       1
7      1.4     Некоторые вводные замечания      254        0       2
8                  о программировании
9      1.5     Продолжение разговора о          315        5       0
10                 программировании
11     1.6     Слово к владельцам ПМК           735        1       5
12                  МК-61 и МК-52
13  Максимальное значение                       735        5       5
14  Минимальное значение                         78        0       0
15  Среднее значение                          264.8333   2.8000   1.50
16    Всего                                    1589       10       9
                                                                  <A:>
                                                                  >C:<
                                                                  pmkkpc]
@avg(C4:C11)          ||        otpmkkpc.C15        ||Char:     12/1
@avg(C4:C11)
                    Formula/number editing; RETURN finishes
```

Рис. 2.16. Электронная таблица пакета Framework III с данными о варианте первой главы книги

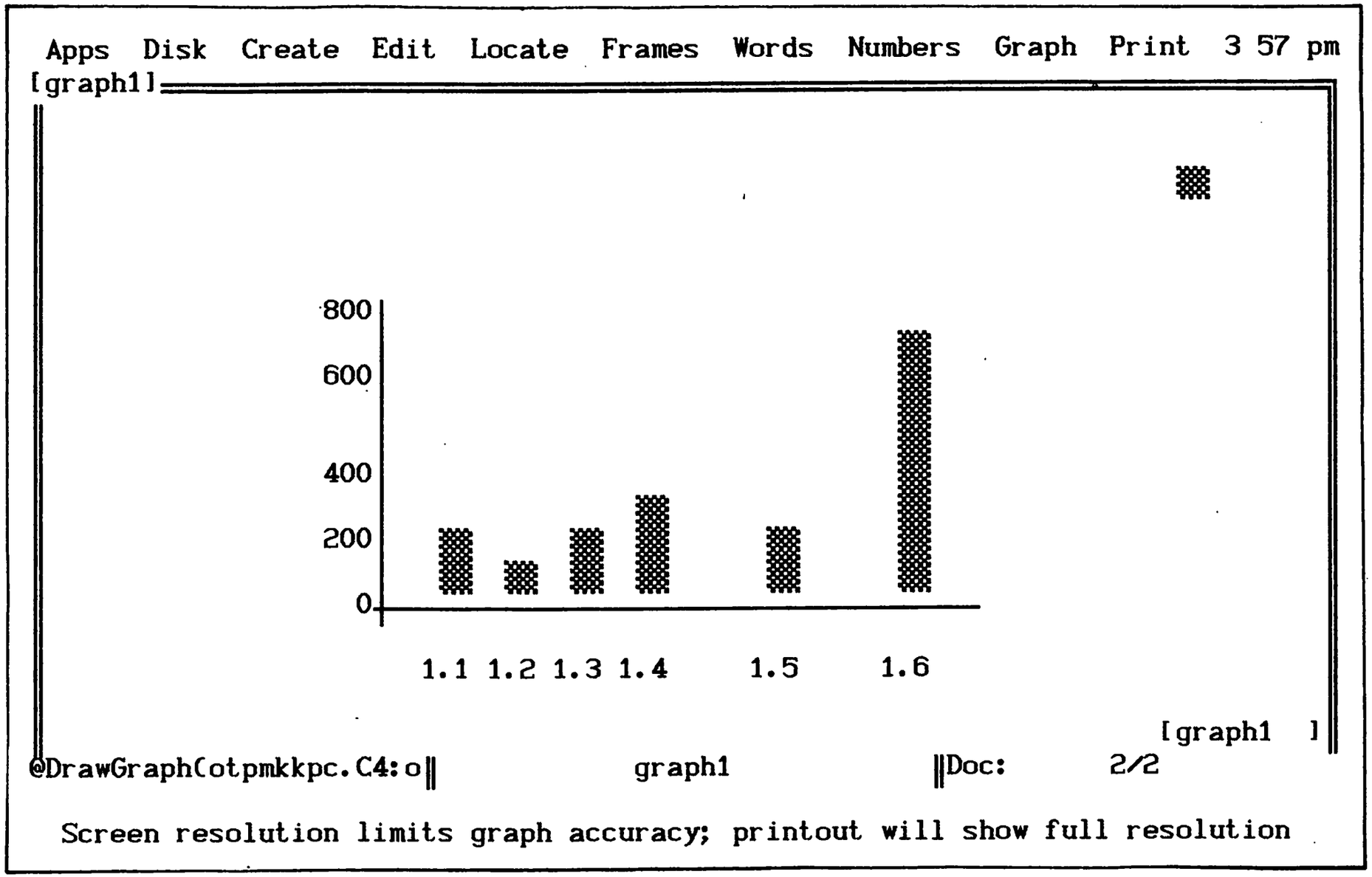


Рис. 2.17. Гистограмма числа страниц варианта первой главы книги

КАК МОЖНО ЖОНГЛИРОВАТЬ ФАЙЛАМИ

Рассказом о двух инструментальных системах Norton Commander и PCTOOLS мы завершаем повествование о программируемых микрокалькуляторах и персональных компьютерах. Серьезное знакомство с персональными компьютерами обычно начинается с инструментальных систем. Они позволяют, как говорится, "вдохнуть душу" в только что распакованный компьютер, т. е. компактно и аккуратно разместить в архивной памяти ЭВМ программные продукты, полуфабрикаты и субпродукты.

Работая с файлами, можно переписывать их с диска на диск с помощью директив операционной системы. Значительно удобнее использовать подходящие для этого инструментальные системы или, как их еще называют, надстройки над DOS. Преимущества инструмента особенно заметны тогда, когда копируемый файл требуется отыскать в дебрях многочисленных записей на архивном диске, подправить его, переименовать, а уж затем скопировать на другой диск.

На рис. 2.18 – картинка, которая может появится при обращении к инструментальной системе Norton Commander или сокращенно NC (читается "Нортон Коммандер" или "Нортон"). Название системы связано с именем ее разработчика Петера Нортона, слово "коммандер" показывает что система предназначена для управления файлами. Картинка на рис. 2.18 могла бы быть иной, так как NC – очень гибкий инструмент, который легко настроить по своему желанию.

Чаще всего используются команды копирования (copy) и перемещения (move) файлов из одной области архивной памяти ЭВМ в другую. После запуска NC разбивает экран дисплея пополам (см. рис. 2.18). Каждая из этих половинок действует как самостоятельный экран (например, может содержать информацию о корневых файлах и директориях).*

Объясним термин "корневые", вспомнив основные правила хранения информации в архивной памяти персонального компьютера.

Еще недавно персональный компьютер называли микроЭВМ, подчеркивая приставкой "микро" ограниченный объем памяти и малое быстродействие этих машин. Сейчас термин микроЭВМ приобрел иное значение и указывает скорее на габариты и низкую стоимость (в конвертируемой валюте) мощных современных персональных компьютеров.

Архивная память, которую раньше называли внешней, – это накопитель информации на жестком диске, или винчестер.

---

* Directory (англ.) – указатель (справочник, каталог).

```
                    D:\                                          C:\                    1:22p
   Name        Name        Name        Name        Name        Name
BAS                                    ANTIVIR     Ibmbio   com jetty   doc
IM                                     BETA        Ibmdos   com jetty   exe
LETER                                  CHESS       autoexe1 bat kart    dat
ME                                     DJ          autoexec bat mouse   com
PAINT                                  DOS         command  com status  me
SC4                                    DRHALO      config   sys t-2     fw2
SK                                     EKA         cpanel   com treeinfo ncd
SPECTR2                                FW2         dj       exe
STATG                                  FW3         dmdrvr   bin
ZAJCEVA                                GAMES       flaw1    map
treeinfo ncd                           GAMES1      hp08x12  fnt
                                       JET         hp08x16  fnt
                                       LEX         hp16x24l fnt
                                       LEXICON     hp16x24m fnt
                                       NC          hp16x24o fnt
                                       PCTOOLS     hp16x24s fnt
                                       PROLOG      hp24x48  fnt
                                       QB          hp24x48b fnt
                                       REBUS       idmkeyb  com
                                       ROLEX       jetto    exe
C:\>
1Help   2User   3View   4Edit   5Copy   6RenMov 7Mkdir  8Delete 9Menu   10Quit
```

Рис. 2.18. Обращение к NC – надстройке над DOS

Совсем недавно он представлял из себя самостоятельный блок, иной раз значительных размеров. Сегодняшний винчестер – это встроенная в компьютер металлическая коробочка, размером не более толстого словаря карманного формата.

Строение архива напоминает дерево: есть корень, ветви – директории и листья – файлы. Такое устройство позволяет владельцу компьютера не тонуть в море информации объемом в десятки, а иной раз, и сотни мегабайт.

На диске файлы получают своего рода прописку. Сравните две записи:

```
СССР, МОСКВА, МЯСНИЦКАЯ, д.3, кв.5, Иванов-отец
C:\MOSKVA\MIASNITSKAJA\D_3\KV_5\Ivanov.man
```

Перед объектами поиска (в первом случае – человек, во втором – файл), стоят директории, или указатели, позволяющие быстро вести поиск.

Вернемся к рис. 2.18. В двух его половинах перечислены прописными буквами имена директорий, а строчными – имена корневых (не входящих в директории) файлов. При этом одно из имен (здесь autoexec.bat) будет выделено курсорной рамкой, которую с помощью клавиш управления курсором можно передвигать в области C (справа) или D (слева), а также переносить из одной области в другую (клавиша Tab). Если нажать клавишу Enter, компьютер может выполнить различные команды. Все зависит от исходной ситуации, например:

если выделено имя директории (прописные буквы), то войдем в нее (были на Мясницкой улице, окажемся в доме 3);

если выделено имя файла (строчные буквы), то NC постарается его запустить.

В первом случае мы как бы карабкаемся по информационному дереву от корня, выбирая нужны ветви. Для обратного спуска достаточно указать обозначение "..", стоящее в заголовке любой некорневой директории.

Набор файлов и директорий, показанный на рис. 2.18, можно встретить практически на любом персональном компьютере. Инструментальная система NC позволяет группировать файлы и директории по размеру, времени записи, имени расширения в алфавитном порядке.

Перечислим содержимое дисков C и D.

На диске C:

ANTIVIR (антивир) – аптечка персонального компьютера (в этой директории скрыты названия "лекарств" – программных средств поиска и уничтожения компьютерных вирусов);

BETA (бэта) – программные средства, позволяющие компьютеру работать с буквами кириллицы;

CHESS (чесс) – набор файлов, превращающих компьютер в шахматного партнера;

DJ (дижет) + JET (жет) – группа файлов, управляющих работой принтера DeskJet.

DOS (ДОС) – набор файлов дисковой операционной системы;

DRHALO ("Доктор Хэло") – набор файлов, который позволяет с помощью "мыши" рисовать и чертить на экране компьютера;

EKA ("Эка" или "Эврика") – пакет Eureka;

FW2 (ФВ 2 или "Фреймворк 2") – вторая версия пакета Framework;

FW3 (ФВ 3 или "Фреймворк 3") – третья версия пакета Framework;

GAMES и GAMES1 (гэймз) – набор компьютерных игр (относятся к ним по-разному: с их помощью можно быстро освоить клавиатуру и "мышь", но в то же время игры – основной источник компьютерных вирусов и компьютерных психозов);

LEX и LEXICON (лекс и "Лексикон") – текстовый процессор, предназначенный в первую очередь для работы с кириллицей (разработан Е. Веселовым из ВЦ АН СССР);

NC – уже известный нам Norton Commander;

PCTOOLS (ПиСиТу – улз) – инструментальная система, по своим возможностям напоминающая NC;

PROLOG (Пролог) – пакет языка программирования Пролог;

QB (кью би или "Квик Бейсик") – среда программирования QuickBASIC, версия 4.0.

REBUS ("Ребус") – русифицированная версия системы управления базой данных dBASE;

ROLEX ("Ролекс") – программы, выводящие на дисплей изображение идущих наручных часов.

Файлы Ibmbio.com, Ibmdos.com, command.com, config.sys, cpanel.com и mouse.com сравнимы с системой зажигания автомобиля – они обеспечивают работу операционной системы MS-DOS.

Особо отметим файл autoexec.bat. Продолжая аналогию с автомобилем, этот файл можно сравнить со стартером. Без стартового файла при включении машины пользователю пришлось бы долго "заводить ее вручную", – давать машине разнообразные команды (например, на загрузку букв кириллицы, программ управления "мышью", запоминание адресов, нужных для работы файлов, и др.)

Стартовый файл записан на одной из первых дорожек магнитного диска, и ЭВМ после включения автоматически находит и запускает его. Нередко пользователи дополняют стартовый файл командой входа в инструментальную систему NC. Тогда после включения компьютера на экране появляется картинка, подобная изображенной на рис. 2.18.

На диске D:

BAS (бэз) – среда программирования QuickBASIC, версия 4.5;

IM (импоссибл мишн) – группа файлов, позволяющая играть с компьютером в Impossible Mission (ей, по правде говоря, место в директории GAMES или GAMES1, но на диске C уже нет места, и игра IM "выплеснулась" на диск D);

LETER (летер*) – под этим название скрываются файлы, с которыми работают, используя текстовый процессор (LEX, LEXICON и MultiEdit) (многие письма служебной переписки похожи (разнятся только имя адресата, дата и некоторые другие детали); поэтому удобно хранить их на диске в виде электронных бланков и, вызвав на дисплей, добавить данные, отредактировать и отпечатать любое число экземпляров без единой помарки**);

ME (малти эдит) – текстовый процессор MultyEdit;

PAINT (пэйнт) – группа файлов, с помощью которых на экране дисплея можно делать цветные рисунки;

SC4 (супер-кальк) – пакет программ для работы в среде электронных таблиц SuperCalc, версия 4;

SK – пакет программ делового человека, помогающий ему вести дневники, содержать в порядке записные книжки, списки телефонов, а также планировать свою работу;

SPECTR2 (спектр 2) – аналог пакета Framework, представляющий подсказку и меню на русском языке;

STATG – пакет программ для статистической обработки результатов экспериментов, а также для построения графиков и поверхностей на экране или на бумаге;

ZAJCEVA – группа программ инженера Л.А. Зайцевой, которая помогала авторам в работе с рукописью.***

В конце списка информации на дисках C и D записано имя файла treeinfo.ncd, который создан системой NC. В нем хранится информация о дереве всех файлов. Увидеть его со всеми ветвями-директориями позволяет система NC (рис. 2.19). Последняя строка на рис. 2.19 подсказывает пользователю возможные режимы работы с деревом информации.

Но вернемся к изображению, которое обычно появляется на экране после включения компьютера (рис. 2.18). Выбирая директории, пользователь отыскивает файл и входит в программную среду, будь то язык программирования, игра или текстовый процессор. Нижняя строка на экране подсказывает, что можно сделать, нажимая на функциональные клавиши:

---

* От англ. letter (письмо).

** Тексты рукописи этой книги также хранились в директории LETER, а после замечаний рецензента и правки редактора в них вносили необходимые изменения.

*** Пользуясь случаем, авторы выражают Л.А. Зайцевой свою признательность.

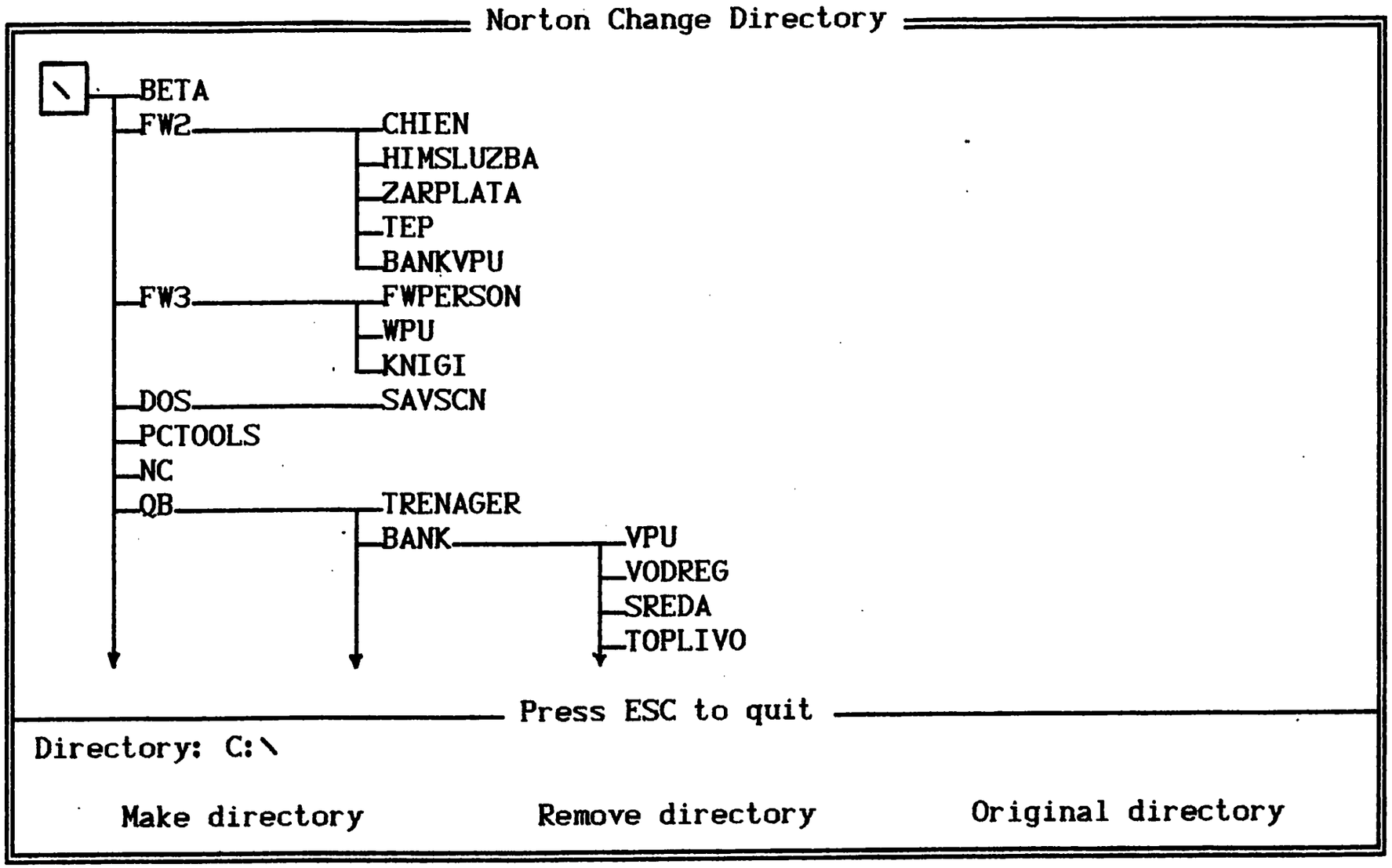


Рис. 2.19. Дерево структуры файлов, построенное с помощью NC

F1 – получить подсказку;

F2 – включить в работу свое индивидуальное меню;

F3 – просмотреть выбранный файл;

F4 – отредактировать выбранный файл (например, так был вызван на экран файл help.eka из пакета Eureka (см. рис. 2.9));

F5 – скопировать выбранный файл или группу файлов на новое место в архивной памяти ЭВМ (так в большинстве случаев и пополняют программное обеспечение компьютера, перенося программы с дискеты (накопитель A:) на винчестер (накопитель C:); команда используется и для распечатки файлов на принтере или для передачи данных на другие ЭВМ, объединенные в сеть.);

F6 – перенести файл или группу файлов на новое место на диске и (или) переименовать их;

F7 – создать новую директорию, т. е. вырастить новую "голую" ветку дерева, на которой затем могут появиться листочки-файлы;

F8 – уничтожить выбранный файл или группу файлов (операция считается весьма деликатной, поскольку приводит к необратимым последствиям; дружественная программа прежде, чем выполнять эту команду, несколько раз переспрашивает человека "Are you sure?" ("вы уверены?"); каждый раз после такого предупреждения проверьте себя еще раз, ведь иные программы или данные стоят дороже машины, а уничтожить их легко, нажав по ошибке всего несколько клавиш.);

F9 – вызвать на экран дополнительное меню, а при необходимости и новые подменю работы с NC;

F10 – завершить работу с NC.

Мы лишь слегка коснулись только основных возможностей NC. Лучший способ глубже ознакомиться с ними – поработать за клавиатурой персональной ЭВМ.

Системе NC едва ли уступает в популярности система PCTOOLS (разработчик – фирма Central Point Software) (рис. 2.20). Она отличается от NC диалогом с пользователем. В NC человек отдает машине команды с помощью функциональных клавиш, подобно тому, как это делается в программных продуктах Eureka, Framework, QuickBASIC, Turbo-Pascal. В системе PCTOOLS управляющими являются клавиши, помеченные буквами, выделенными в ключевых словах меню (см. три нижних строки на рис. 2.20), аналогично тому, как это сделано в текстовом процессе MSWord.

По своим возможностям обе системы близки. Может быть поэтому выбор той или иной из них чаще всего определяется личным вкусом или привычкой пользователя, а не какими-либо принципиальными соображениями.

```
PC Tools Deluxe R4.23                                          Vol Label=DISK1_VOL1
-----------------------------------File Functions-------------------Scroll Lock OFF
Path=C:\QB\128SOVET\GLAVA0\*.*
    Name      Ext      Size Attr   Date        Name      Ext      Size Attr   Date
 1  FIG15A    BAS       578 ...A  6/21/89      SOVET15   BAS       515 ...A  5/22/89
 2  FIG15B    BAS       566 ...A  1/01/80      SOVET18   BAS      1069 ...A  5/24/89
 3  FIG18     BAS       789 ...A  1/01/80      SOVET2    BAS       514 ...A  5/22/89
 4  FIG2      BAS       585 ...A  1/01/80      SOVET21   BAS       289 ...A  6/20/89
 5  FIG28     BAS       767 ...A  1/01/80      SOVET22   BAS       442 ...A  5/24/89
 6  FIG39     BAS       562 ...A  1/01/80      SOVET28   BAS      1002 ...A  5/22/89
 7  FIG5      BAS      1220 ...A  6/21/89      SOVET3    BAS       311 ...A  6/20/89
 8  FIG8A     BAS       329 ...A  6/21/89      SOVET33   BAS       630 ...A  6/16/89
 9  FIG8B     BAS       320 ...A  1/01/80      SOVET34   BAS       633 ...A  6/16/89
10  FIG9A     BAS       502 ...A  6/21/89      SOVET36   BAS      1647 ...A  4/27/89
11  FIG9B     BAS       723 ...A  1/01/80      SOVET39   BAS       362 ...A  6/16/89
    SOVET12   BAS       927 ...A  5/22/89      SOVET4    BAS       241 ...A  6/20/89
    SOVET14   BAS      1043 ...A  5/24/89      SOVET5    BAS      1098 ...A  6/20/89
+-------------------------------------------------------------------------------+
!  29 files LISTed    =    19675 bytes.   29 files in sub-dir =    19675 bytes. !
!  11 files SELECTed  =     6941 bytes.   Available on volume =  1132544 bytes. !
+-------------------------------------------------------------------------------+
!   Copy Move cOmp Find Rename Delete Ver view/Edit Attrib Wordp Print List     !
!Sort Help <+=SELECT F1=UNselect F2=alt dir lst F3=other menu Esc=exit PC Tools!
!  F8=directory LIST argument  F9=file SELECTion argument  F10=chg drive/path  !
+-------------------------------------------------------------------------------+
```

Рис. 2.20. Обращение к PCTOOLS – надстройке над DOS

Еще один пример, поясняющий схожесть и различия двух инструментальных систем. Если в NC клавиша RETURN (Ввод) служит для входа в директорию или для запуска программы, хранящейся на диске в виде файла с расширением ".exe", ".com", или ".bat", то в PCTOOLS нажатие этой клавиши указывает машине, какие файлы будут участвовать в дальнейших действиях: копировании, стирании и т. д. С помощью клавиши RETURN были отмечены одиннадцать файлов (см. рис. 2.20) объемом 6941 байт с адресом

C:\QB\128SOVET\GLAVA0\ .

По этому адресу разместились Бейсик-программы нулевой (так назвали ее авторы) главы книги В.Ф. Очкова и Ю.В. Пухначева "128 советов начинающему программисту" (М.: Энергоатомиздат, 1990).

Из рисунка 2.20 видно, что в поддиректории GLAVA0 "проживают" 29 файлов объемом 19675 байт, а на диске C остается 1132544 свободных байта.

Читателю, старающемуся разобраться с описанием инструментальной системы без компьютера, может показаться, что она не упрощает, а напротив, только усложняет работу с файлами. Для того чтобы убедится в обратном, достаточно хотя бы час поработать с системой.

К хорошему привыкаешь так быстро, что просто невозможно затем отказаться от такого удобного инструмента, как NC или PCTOOLS, и вернуться к "жонглированию" файлами командами DOS.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Абрамов С.А., Зима Е.В. Начала программирования на языке Паскаль. – М.: Наука. 1987. – 110 с.

Богдюкевич С.В., Шахвердов В.А., Баласанян В.Э. Программирование на микроЭВМ "Искра 226". – М.: Финансы и статистика. 1987. – 235 с.

Банди Б. Методы оптимизации: Вводный курс / Пер. с англ. – М.: Радио и связь. 1988. – 128 с.

Борковский А.Б. Англо-русский словарь по программированию и информатике (с толкованиями). – М.: Рус. яз.. 1989. – 335 с.

Боровин Г.К., Комаров М.М., Ярошевский В.С. Ошибки-ловушки при прграммировании на Фортране. – М.: Наука. 1987. – 141 с.

Бронштейн И.Н., Семендяев К.А. Справочник по математике для инженеров и учащихся вузов. 13-е изд., испр. – М.: Наука. 1986. – 544 с.

Брудно А.Л. Введение в программирование в содержательных обозначениях. – М.: Наука. 1965. – 148 с.

Брудно А.Л., Каплан. Олимпиады по программированию для школьников. – М.: Наука. 1985. – 96 с.

Брябрин В.М. Программное обеспечение персональных ЭВМ. – М.: Наука, 1989. – 271 с.

Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ / Пер. с англ.; 2-е изд., испр. – М.: Мир. 1985. – 332 с.

Вирт Н. Алгоритмы + структуры данных = программы / Пер. с англ. – М.: Мир. 1985. – 406 с.

Вирт Н. Систематическое программирование: Введение / Пер. с англ. – М.: Мир. 1977. – 185 с.

Вуд А. Микропроцессоры в вопросах и ответах / Пер. с англ. – М.: Энергоатомиздат. 1985. – 183 с.

Геворкян Г.Х., Семенов В.Н. Бейсик – это просо. – М.: Радио и связь. 1989. – 143 с.

Горстко А.Б., Кочковская С.В. Азбука программирования (Сер. Информатика для всех). – М.: Знание. 1988. – 140 с.

Грогоно П. Программирование на языке Паскаль / Пер. с англ. – М.: Мир, 1982. – 382 с.

Грэхем Р. Практический курс языка Паскаль для микроЭВМ / Пер. с англ. – М.: Радио и связь, 1986. – 198 с.

Гудман С., Хидетниеми С. Введение в разработку и анализ алгоритмов / Пер. с англ. – М.: Мир, 1981. – 366 с.

Дал У., Дейкстра Э., Хоор К. Структурное программирование / Пер. с англ. – М.: Мир, 1975. – 247 с.

Джонстон Г. Учитесь программировать / Пер. с англ. – М.: Финансы и статистика, 1989. – 356 с.

Дьяконов В.П. Справочник по алгоритмам и программам на языке Бейсик для персональных ЭВМ. – М.: Наука, 1987. – 240 с.

Йодан Э. Структурное проектирование и конструирование программ / Пер. с англ. – М.: Мир, 1979. – 415 с.

Кейтер Дж. Компьютеры – синтезаторы речи / Пер. с англ. – М.: Мир, 1985. – 237 с. (Сер. В мире науки и техники).

Керниган Б., Плоджер Ф. Элементы стиля программирования / Пер. с англ. – М.: Радио и связь, 1984. – 158 с.

Кетков Ю.Л. Диалог на языке Бейсик для мини- и микро ЭВМ. – М.: Наука, 1988. – 367 с.

Кнут Д. Искусство программирования для ЭВМ. Т.1. Основные алгоритмы / Пер. с англ. – М.: Мир, 1976. – 735 с.

Компьютеры: Справ. руководство. В 3-х т. /Под ред. Г.Хелмса; Пер. с англ. – М.: Мир, 1986.

Косневски Ч. Занимательная математика и персональный компьютер / Пер. с англ. – М.: Мир, 1987. – 192 с.

Кренкель Т.Э., Коган А.Г., Тараторкин А.М. Персональные ЭВМ в инженерной практике. – М.: Радио и связь, 1989. – 336 с.

Кучура Н.А., Ходош М.В., Цагельский В.И. Персональные ЭВМ Единой системы. Бейсик (Практ. руководство). – М.: Финансы и статистика, 1988. – 207 с.

Лоберг Р., Лутц Т. Домашний компьютер / Пер. с нем. – М.: Дет. лит., 1987. – 95 с. (Сер. Науч.-попул. лит.).

Маркелова Л.Н. Эксплуатация программоуправляемой вычислительной машины "Искра 226". – М.: Машиностроение, 1987. – 224 с.

Мир компьютеров в вопросах и ответах: В 2-х кн. / Пер. с япон. – М.: Мир, 1989.

Мичи Д., Джонстон Р. Компьютер-творец / Пер. с англ. – М.: Мир, 1987. – 254 с.

Моррил Г. Бейсик для персонального компьютера ИБМ / Пер. с англ. – М.: Финансы и статистика, 1987. – 206 с.

Основы компьютерной грамоты / А.Н. Жигарев, Н.В. Макарова. М.А. Путинцева; Под общ. ред. Н.В. Макаровой. – Л.: Машиностроение, 1987. – 153 с.

Очков В.Ф., Пухначев Ю.В. 128 советов начинающему программисту. – М.: Энергоатомиздат, 1990. (В печати).

Очков В.Ф., Пухначев Ю.В. 24 этюда на Бейсике. – М.: Финансы и статистика, 1988. – 112 с.

Перегудов М.А., Халамайзер А.Я. Бок о бок с компьютером. – М.: Высш. шк., 1987. – 207 с.

Перминов О.Н. Программирование на языке Паскаль. – М.: Радио и связь, 1988. – 219 с.

Персональный компьютер в играх и задачах. – М.: Наука. 1988. – 192 с. (Сер. Кибернетика - неограниченные возможности и возможные ограничения).

Персональные компьютеры: Информатика для всех. – М.: Наука, 1987. – 203 с. (Сер. Кибернетика - неограниченные возможности и возможные ограничения).

Программирование на языке Бейсик-плюс для СМ-4 / В.П. Семик., Б.Р. Монцибович., Д.П. Непочатых и др. – М.: Финансы и статистика, 1982. – 246 с.

Программное обеспечение микроЭВМ: Практ. пособие для инж.-пед. работников системы профтехобразования. В 11-ти кн./Под ред. В.Ф.Шаньгина. – М.: Высш. шк. 1986 – 1988.

Трейстер Р. Персональный компьютер фирмы IBM / Пер. с англ. – М.: Мир, 1986 – 210 с.

Уолш Б. Программирование на Бейсике / Пер. с англ. – М.: Радио и связь, 1988. – 335 с.

Уорт Т. Программирование на языке Бейсик / Пер. с англ. – М.: Машиностроение, 1981. – 255 с.

Фокс А., Фокс Д. Бейсик для всех / Пер. с англ. – М.: – М.: Энергоатомиздат, 1986. – 176 с.

Человек и компьютер // Наука и жизнь. 1986., № 2, 4, 12; 1987 № 6, 10, 12; 1988. № 4, 6, 10, 12; 1989. № 2, 4, 12.

Школа пользователя ЭВМ // Наука и жизнь. 1990. № 1, 3, 5, 7.

Языки программирования Ада, Си, Паскаль: Сравнение и оценка / Под ред. А.Р. Фьюэра, Н.Джехани; Пер. с англ. – М.: Радио и связь, 1989. – 386 с.

ОГЛАВЛЕНИЕ

Предисловие 3

Глава 1.
Программируемые микрокалькуляторы

Первое знакомство 5
Коротко о большом семействе 5
Учимся программировать 11
Начинаем работать на МК-61 35

Детали и тонкости 57
Алфавит карманной ЭВМ 57
Логические операции в игровых программах 65
Секреты микрокалькулятора 69
Для чего нужен тест 82

Библиотека программ 88
Универсальная программа 88
Азимут и дальность 94
Эти "случайные" неслучайные числа 97
Вечный календарь 100
Поможет структурная диаграмма 104
Микрокалькулятор играет в спортлото 107
Сложные программы 108
Микрокалькулятор-компьютер 111

Приложения 116
Приложение 1. Лексикон программируемых микрокалькуляторов "Электроника Б3-34, МК-61, -52" и их модификаций 116
Приложение 2. Коды программируемых микрокалькуляторов "Электроника МК-52 и -61" 120

Список рекомендуемой литературы 122

Глава 2.
Персональный компьютер

От микрокалькулятора к персональному компьютеру 125

Цикл типа "пока" 128

Альтернатива 130

Вложенные структуры 131

Цикл типа "до" 134

Золотое сечение 135

Цикл с параметром 137

Подпрограммы и процедуры 140

Новая машина – новые возможности 142

Великолепная семерка программирования 142

Можно ли играть на скрипке с одной струной? 149

Еще раз о подпрограммах и процедурах 153

В погоне за собственным хвостом 155

Кролики и минимумы 155

Когда задачу решают с конца 157

Программа без чисел и вычислений 162

Кит или кот? 164

Электронный советчик 167

Что в имени тебе моем? 169

Компьютер-секретарь 175

Программа, печатающая сама себя 177

Работаем с готовыми программами 179

Программирование без программирования 179

"Окна" в мир информации 195

Как можно жонглировать файлами 211

Список рекомендуемой литературы 220

Подписывайтесь на "Интеркомпьютер"!

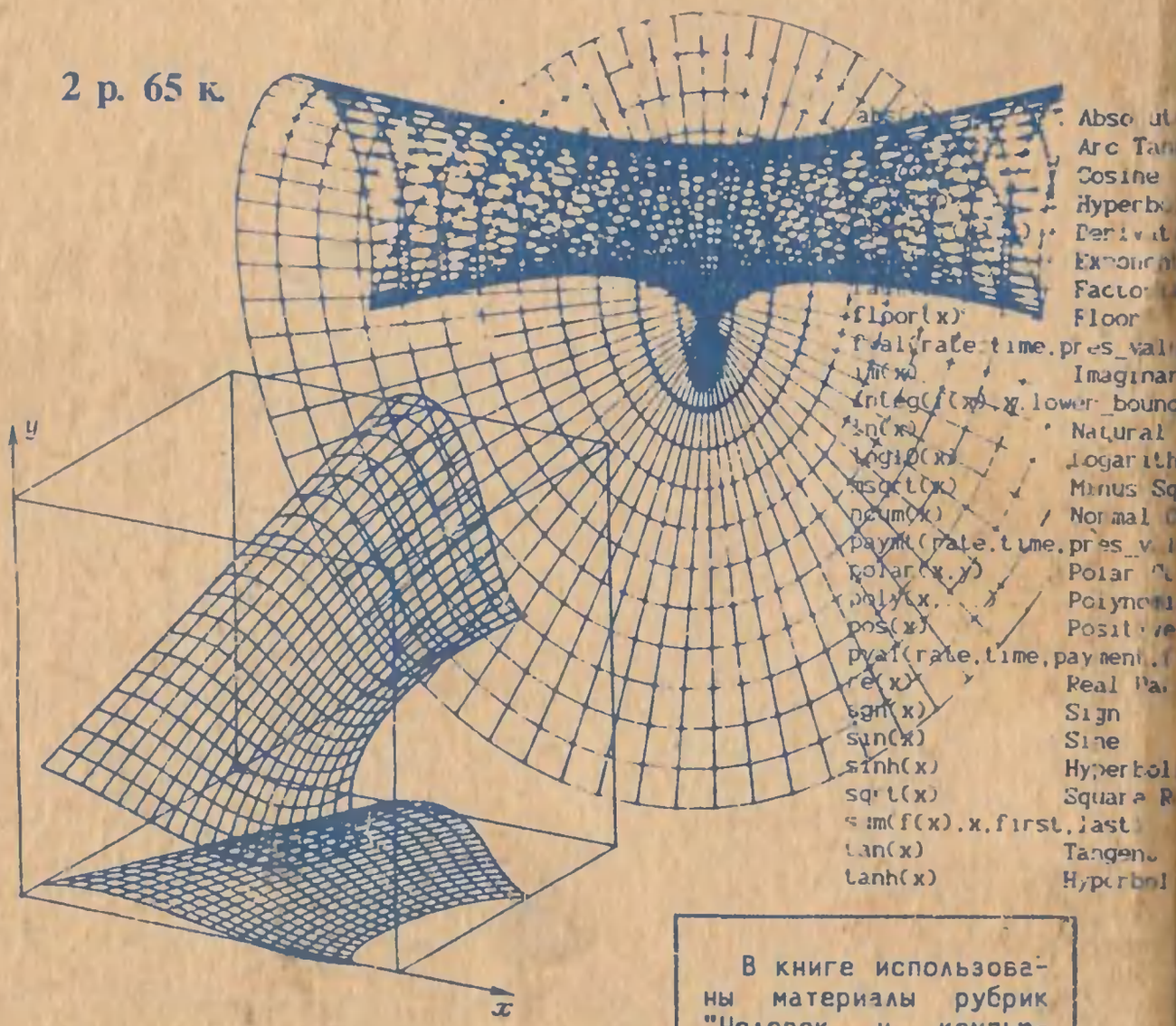
ИНТЕРКОМПЬЮТЕР - это информационный бюллетень, издаваемый совместным предприятием ИНТЕРЮНИТИ.
ИНТЕРКОМПЬЮТЕР познакомит вас с программными продуктами и техническими средствами, разрабатываемыми в СССР и за рубежом.
Вы узнаете о новостях мира вычислительной техники и тенденциях его развития, получите оперативную информацию о последних разработках и результатах тестирования новейших программ и устройств.
ИНТЕРКОМПЬЮТЕР будет держать вас в курсе важных событий и расскажет о достижениях, которые также находят отражение на страницах ведущих компьютерных изданий Byte, PC Magazine, PC World, Dr. Dobb's Journal, InfoWorld, PC Week.
Не менее актуальной будет и информация о состоянии компьютерного рынка в СССР.

ИНТЕРКОМПЬЮТЕР издается с ноября 1989 г.
В нынешнем году выйдет в свет 8 номеров.
Тираж 50000 экз. Цена 3 руб.

По вопросам подписки обращайтесь в ЦНТУ "Информ" по адресу:

111141, Москва, Зеленый просп., 7
Телефон: 176-68-88

2 р. 65 к.



В.Ф.Очков, В.А.Хмелюк

От микрокалькулятора к персональному компьютеру

В книге использованы материалы рубрик "Человек и компьютер", "Школа начинающего программиста" и "Школа пользователя ЭВМ" журнала "Наука и жизнь".

Программирование в кодах микрокалькуляторов, на алгоритмических языках Бейсик и Паскаль, работа в программных средах персональных компьютерах - основные темы книги. Для многих - это ступени на пути овладения вычислительной техникой.

Москва
Издательство МЭИ
1990 г.