Издательство «Знание»

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Новое в жизни, науке, технике

Подписная научно-популярная серия

Издается ежемесячно с 1988 г.

Персональная ЭВМ «Ямаха MSX-2»

1989

8

Новое
в жизни,
науке,
технике

Подписная
научно-
популярная
серия

8/1989

Издается
ежемесячно
с 1988 г.

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

ПЕРСОНАЛЬНАЯ ЭВМ «ЯМАХА MSX-2»

В Номере:

Издательство
«Знание»
Москва
1989

ББК 32.973
П27

Авторы выпуска

УШАНОВ Сергей Николаевич — известный публицист. Основная направленность публикаций — пути развития научно-технического прогресса.

МЕШКОВ Николай Алексеевич — кандидат технических наук, старший научный сотрудник кафедры кибернетики МИЭМ, имеет несколько изобретений, награжден почетным знаком Минвуза СССР и ЦК ВЛКСМ «За успехи в научно-техническом творчестве молодежи». Победитель I Московского городского конкурса молодых ученых и специалистов по учебным и игровым программам.

КОЛОСОВ Юрий Владимирович — кандидат технических наук, доцент, специалист в области разработки средств микропроцессорной техники.

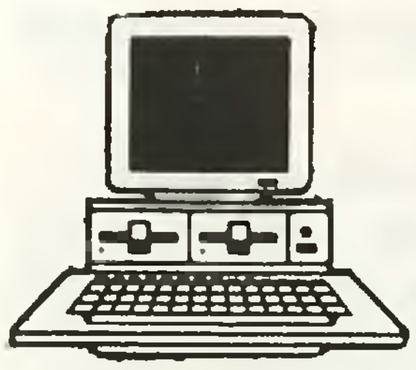
МАЛЫХИНА Мария Петровна — кандидат технических наук, доцент, программист.

ЧАСТИКОВ Аркадий Петрович — кандидат технических наук, доцент, специализируется в области информатики и вычислительной техники.

УТЕНКОВ Сергей Николаевич — инженер, журналист.

РЕДАКТОР Б. М. ВАСИЛЬЕВ

ISBN 5-07-000333-X



Мы собираемся дать краткую сводку чисто практических советов, призванных помочь вам овладеть навыками работы на персональном компьютере. Мы рассчитываем на читателя, слышавшего о компьютерах, может быть, даже видевшего их в действии, однако самостоятельно еще не работавшего на них. Таким читателем будет и «технарь», получивший образование в докомпьютерный период, и современный любознательный гуманитарий. Всех их объединяет общее желание научиться использовать компьютер в своей работе и до конца еще не преодоленный страх перед этой сравнительно новой и непривычной для нас техникой.

Н. МЕШКОВ, С. УШАНОВ

ЗНАКОМСТВО С ПЕРСОНАЛЬНЫМ КОМПЬЮТЕРОМ «Ямаха MSX-2»

Понимая все нетерпение читателя, раскрывшего эту книжку, впервые в жизни севшего за персональный компьютер (руки так и чешутся поскорей что-нибудь нажать!), мы тем не менее призываем вас вначале внимательно прочесть вступление.

Сознательно делая упор на практические проблемы, мы постараемся без особой надобности не перегружать голову читателя побочными сведениями. В отличие от многих других книг на такую тему, мы полностью опускаем историю и перспективы развития компьютерной техники, сравнительный анализ различных персональных компьютеров и т. п. Начинающему пользователю, на наш взгляд, совершенно не нужно знать, как устроен компьютер внутри и как, собственно говоря, он «считает» — в двоичной, восьмеричной или какой-нибудь еще системе счисления. Есть просто некий «черный ящик» и ряд правил работы с ним, позволяющих получить от него нужный вам полезный результат. Ваша задача — понять эти правила и научиться применять их на практике, а наша — помочь вам в этом.

Описывая свойства персонального компьютера, мы будем иметь в виду вполне конкретную его модель — японскую машину «Ямаха MSX-2». Эти компьютеры (и их предыдущие варианты «Ямаха MSX») входят в комплекты учебной вычислительной техники, закупаемые для вузов, школ нашей страны, и, следовательно, стали сегодня у нас относительно распространенными.

Сокращение MSX обозначает не марку компьютера, а некий стандарт выпускаемых на Западе «домашних» ЭВМ — не слишком дорогих, но обладающих сбалансированным набором качеств, обеспечивающих удовлетворение многих потребностей делового человека как на службе, так и дома. Если на компьютере стоит значок MSX — вы можете смело садиться за него после чтения этой книжки, даже когда перед вами оказалась не «Ямаха», а, скажем, «Филипс» или машина какой-то иной марки. Кроме того, стандарт MSX гарантирует так называемую совместимость снизу вверх, т. е. все, чему вы научились, работая с компьютером MSX, вы спокойно можете применить и к более мощным машинам стандарта MSX-2 (но не наоборот).

За основу для описания мы выбрали данные компьютера «Ямаха MSX-2», поскольку эта версия стандарта более перспективна. Впрочем, отличия не слишком существенны, так что почти все сказанное далее принесет пользу и тем, кто собирается работать на «младшей» модели «Ямаха MSX». Больше того, навыки, которые, как мы надеемся, приобретет наш читатель, он сможет применить и в общении с другими персональными компьютерами примерно того же класса — «домашними» вариантами ЭВМ «Коммодор», «Атари», «Синклер» и их отечественными собратьями вроде «Микроши», «Агата», БК-0010 и т. д. Приобретя некоторый опыт практической работы на компьютере, читатель без труда сумеет потом приспособиться к любой машине сходного класса сложности.

Итак, мы начинаем.

Познакомимся с компьютером

На рис. 1 показан состав комплекта вычислительной техники, который

Рис. 1

обычно и называют персональным компьютером (ПК). Собственно, ПК — это часть ящика, названного на рисунке системным блоком, а все остальное — периферийные устройства. Перечислим их.

Клавиатура. Позволяет пользователю (т. е. вам, читатель) вводить в ПК информацию. Обычно выглядит как набор клавиш стандартной пишущей машинки, дополненный для удобства работы добавочными клавишами (отдельно вынесены клавиши с цифрами и знаками арифметических действий, особые управляющие клавиши).

Монитор. Специальный высококачественный «телевизор», на экране которого ПК отображает буквы, цифры, рисунки. Может быть черно-белым или цветным. Цветной, разумеется, дает гораздо больше возможностей и удовольствия пользователю, однако он дороже.

Дисковод. Объемистая внешняя память ПК. Щели для двух дисководов видны на рисунке в правой части системного блока (в иных вариантах конструктивного исполнения дисководы располагаются отдельно от системного блока). Подлежащая длительному хранению информация записывается на гибкие магнитные диски, которые вставляются в щели дисководов. «Ямаха MSX» использует малоформатные гибкие диски диаметром 3,5 дюйма (89 мм)

«Мышь». Ручной манипулятор, передвигаемый по поверхности стола и позволяющий пользователю удобно и быстро оперировать элементами изображения на экране, посылать в ПК команды и т. д. Вместо «мыши» к системному блоку можно подключить и другие подобные манипуляторы: джойстики (поворотная рукоятка с кнопкой), световое перо («электронный карандаш», проводя которым по поверхности экрана можно как бы рисовать на ней), специальный планшет, выполняющий функции «мыши», и т. п. Эти манипуляторы заметно облегчают работу на ПК.

Принтер. Устройство, печатающее на бумаге изображение, которое пользователь видит на экране монитора, или невидимое, но хранимое в памяти машины. Принтер того типа, какими обычно снабжаются ПК, может воспроизвести на бумаге не только цифровые таблицы или текст, но также и любой сложный рисунок — график, диаграмму, даже копию фотоснимка, поскольку изображение составляется из множества мельчайших точек. Сложные и дорогие принтеры способны отпечатать на бумаге и цветное изображение.

Каждое из периферийных устройств подключено к системному блоку своим кабелем (сказанное относится, естественно, и к отдельно расположенным дисководам). Прежде чем включить ПК, внимательно рассмотрите рис. 1 и проверьте, верно ли подсоединена ваша «периферия». Если по неопытности или рассеянности вы забудете подключить какой-то не самый главный кабель, ничего страшного не случится: компьютер сам «почувствует» это и в нужный момент сделает вам вежливый выговор.

Теперь, еще по сути дела ничего не зная о ПК, вы можете включить его. Стоп! Ничего не трогайте!

ВНИМАНИЕ! Запомните правило: вначале всегда нужно включать питание периферийных блоков и только затем — системного. Выключать — в обратном порядке: сперва системный блок, а уж потом — периферийные. Соблюдение этой маленькой условности продлит жизнь вашему компьютеру.

Итак, включаем «периферию»: монитор, дисковод, если у него отдельный выключатель (предварительно убедитесь, что из щели не торчит забытый там магнитный диск: включать и выключать это устройство нужно без диска), принтер, если он понадобится. «Мышь» или джойстик выключателей питания, как правило, не имеют.

Включив наконец системный блок, вы уже запустили ПК в работу. Что делает при этом компьютер? Прежде всего он проверяет, установлен ли пользователем диск с исходными программами (вы услышите ворчанье мотора дисковода).

Если такого диска нет, ПК, как было задумано его проектировщиками, обращается к своей внутренней долговременной памяти (профессионалы называют ее постоянным запоминающим устройством — ПЗУ), считывает оттуда требуемую для начала работы информацию и переходит в режим так называемого резидентного БЕЙСИК-интерпретатора. Компьютер сообщит вам об этом примерно такой текстовой заставкой, появившейся на экране монитора:

```
MSX BASIC version 2.1
Copyright 1986 by Microsoft
Ok
```

Это значит, что ПК готов принимать и исполнять команды алгоритмического языка БЕЙСИК-MSX (версия 2.1), программа пошагового исполнения которых (интерпретатор) резидентно, т. е. постоянно, хранится в ПЗУ компьютера. Для дальнейшей работы вам доступен объем быстродействующей (оперативной) памяти ПК, равный нескольким десяткам тысяч символов. Готовность к такому режиму работы компьютер показывает завершающим заставку «Ok»(о'кей).

Полезно знать, что начать работу на ПК можно и по-другому. Если после включения периферийных блоков пользователь вставит в щель дисковода так называемый системный диск (на нем записана весьма сложная информация, образующая «дисковую операционную систему») и только затем включит питание системного блока, компьютер высветит на экране иную заставку:

```
MSX—DOS version 1.03
Copyright 1984 by Microsoft
COMMAND version 1.11
A>
```

Смысл этого сообщения таков: ПК прочел с диска, «усвоил» и готов применять дисковую операционную систему MSX-DOS. Пользователь при этом располагает очень широкими возможностями выбора «инструментария» — несколько алгоритмических языков, таких, как БЕЙСИК, ПАСКАЛЬ и т. д., различные служебные программы и многое другое. Находясь в рамках MSX-DOS, вы можете пользоваться более эффективным вариантом того же БЕЙСИКа (он именуется компилируемым), однако прибегать к нему на первых шагах работы с ПК вряд ли имеет смысл. Лучше набраться терпения и подождать, пока компьютер не станет вашим более близким знакомым, а срок этот зависит, разумеется, целиком от вас. Чем больше времени вы общаетесь с ПК, тем меньше будет пугающих проблем!

Первые диалоги

Вернемся к самому простому варианту — работе с резидентным БЕЙСИКом. Кроме информационной заставки, завершающейся успокоительным «Ok», вы видите на экране маленький светлый прямоугольник. Он называется курсором и указывает место, на котором после нажатия клавиши появится символ — цифра, буква или специальный знак.

Перед вами клавиатура (ее вид приведен на рис. 2). Если нажать помеченную буквой клавишу, эта буква тут же отобразится на том месте, где только

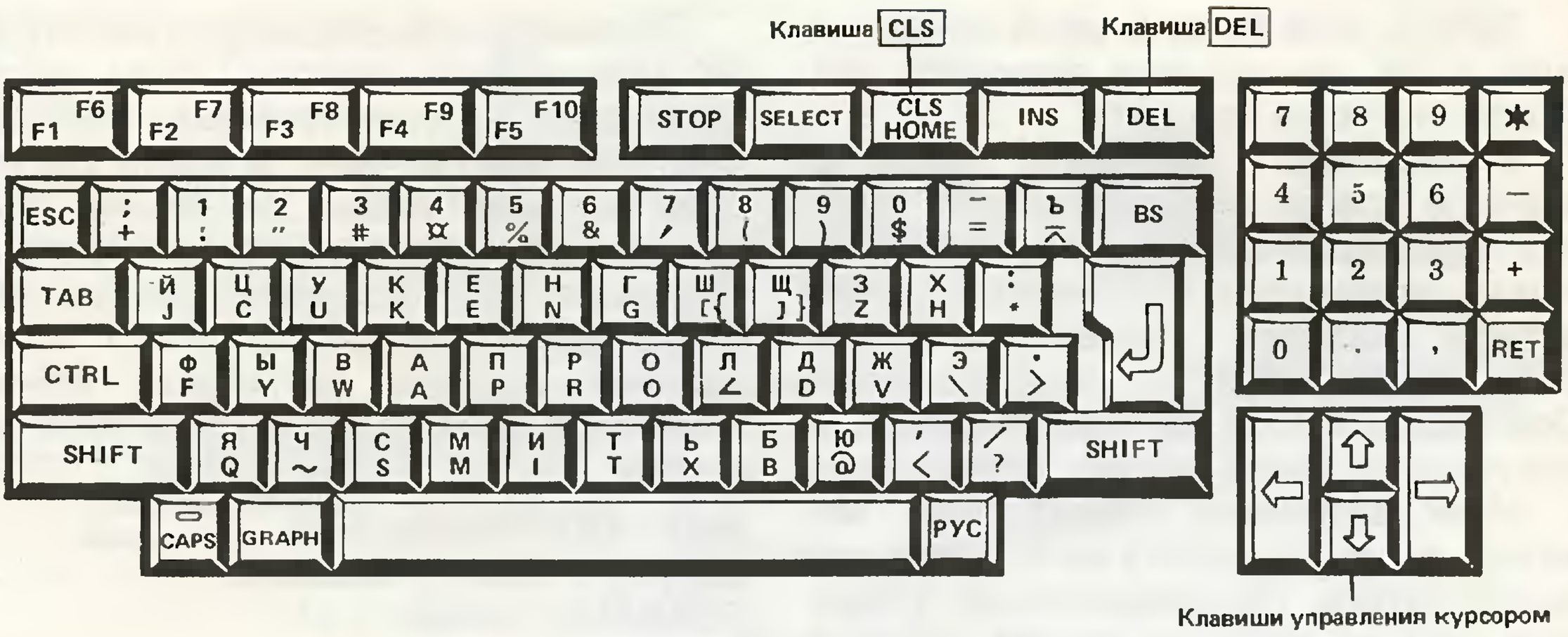


Рис. 2

что находился курсор, а сам он мгновенно перескочит на одну позицию вправо, указывая место появления следующего символа. Нажмите еще раз буквенную клавишу и держите ее нажатой — ПК через полсекунды поймет ваше намерение и начнет быстро повторять этот символ, заполняя им строку. Когда цепочка букв дойдет до правого края экрана, курсор сам перескочит на начало новой строки и начнется заполнение следующей строчки.

Каждая клавиша основной клавиатуры имеет по меньшей мере четыре значения: можно получить латинскую или русскую букву, прописную или строчную. Переключатель с латинского алфавита на русский и обратно — клавиша РУС, расположенная справа внизу. Прописные буквы (а также иные символы, изображенные в верхней части клавиш) будут отображены на экране, если предварительно нажать клавишу переключения регистра SHIFT и удерживать ее нажатой, пока вновь не потребуются строчные буквы. Чтобы облегчить работу на клавиатуре, предусмотрен «замок верхнего регистра» — клавиша CAPS, после однократного нажатия которой все буквы будут печататься прописными (удерживать ее не нужно). Перейти вновь на нижний регистр можно, еще раз нажав клавишу CAPS. Если среди прописных букв встретятся одна-две строчные, можно, не переключая регистра, воспользоваться клавишей SHIFT, которая на этот раз временно вернет нас к строчным буквам. Клавиша GRAPH, расположенная рядом с замком регистра CAPS, дает еще одну возможность: алфавитно-цифровые клавиши вызывают с ее помощью так называемые псевдографические символы — заготовки графического оформления таблиц, простых рисунков и т. д.

Отдельная малая цифровая клавиатура предназначена для печатания цифр и знаков арифметических действий. Под ней находится четверка клавиш, предназначенных для управления курсором. Нажимая одну из этих клавиш, вы можете перемещать курсор на экране в направлении, указанном стрелкой на клавише.

Блок алфавитных и цифровых клавиш основной клавиатуры окружен служебными клавишами, позволяющими выполнять специальные функции, зависящие от вида исполняемой компьютером программы, от режима его работы и т. д. Рассказывать об этих органах управления ПК мы станем по ходу изложения, а пока упомянем лишь о двух, имеющих аналоги на обычной пишущей машинке. Клавиша BS — «шаг назад» — позволяет вернуться к неверно напечатанному знаку и исправить опечатку. Большая клавиша с жирной изогнутой стрелкой по своему действию отчасти похожа на клавишу возврата каретки пишущей машинки. Пользоваться ею нам придется очень часто, и для краткости ниже мы станем обозначать ее как ВК. (Ее «дублер» есть и на цифровой клавиатуре — клавиша RET).

Письменный диалог с компьютером логично начать с простой арифметики,

тем более что многие начинающие пользователи ПК в глубине души считают его попросту большим арифмометром. Не станем тратить слова, переубеждая их, — очень скоро компьютер сам докажет всю вздорность подобного вывода.

Проверим способности компьютера на самом простом примере. Напечатайте на экране: 2+2=. ПК не даст ответа, хотя вы, наверное, предполагали, что, словно в калькуляторе, вслед за знаком равенства выскочит цифра 4. Нет, компьютер еще не знает ваших намерений и, чтобы дать ему понять, что вы обратились к нему с заданием, необходимо нажать клавишу ВК. Нажали? Читайте ответ компьютера: SYNTAX ERROR — Синтаксическая ошибка. ПК напоминает вам, что в обращении с ним нужен чрезвычайный педантизм и что все обычные для беседы между людьми неопределенности, недоговоренности и упования на догадливость собеседника вам следует раз и навсегда забыть. ПК, как мы договорились, работает в режиме языка БЕЙСИК, так что команду следовало оформить строго по правилам этого языка. В БЕЙСИКе нужное нам задание описывается оператором PRINT, т. е. вот что нужно напечатать на экране:

PRINT 2+2(ВК)

(«ВК», разумеется, на экране не отразится). Компьютер тут же даст верный ответ: в начале следующей строки появится число 4. (Заметьте, что знака равенства нет!) Чтобы не печатать всякий раз целое слово PRINT, ему предусмотрена короткая замена — знак вопроса (?).

Арифметические действия обозначаются здесь как обычно: +, −, /(знак деления в виде двух точек запрещен), * (звездочка заменяет косой крест знака умножения). Порядок выполнения четырех действий арифметики тоже обычный: вначале умножение и деление, затем сложение и вычитание. Изменить порядок можно одной, двумя и т. д. **парами** скобок — тоже по обычным правилам, только здесь все скобки должны быть круглыми: ...(...(...)...)... Компьютер перед вычислением непременно проверит правильность записи и, скажем, если число открывающих скобок не равно числу закрывающих, считать результат не будет, опять вежливо сообщив невнимательному пользователю о синтаксической ошибке.

Таким путем — работая в режиме калькулятора — вы можете вычислять весьма сложные алгебраические выражения, но полная длина командной строки не должна превышать 255 символов, иначе ПК снова укажет вам на нарушение правил. Командная строка — это все выражение, целиком завершаемое нажатием клавиши ВК. Если в одной строчке экрана умещается 80 символов, то выражение может быть длиной три экранных строки с небольшим. Никаких знаков переноса формулы применять не нужно: курсор сам будет переходить с конца одной строки экрана на начало следующей, разрывая математическое выражение в любом месте.

Если к перечисленным четырем действиям арифметики добавить еще операцию возведения в степень (она обозначается так: 2^3 — два в третьей степени и выполняется в цепочке действий самой первой), то у вас в руках окажется набор вычислительного инструмента для решения многих не слишком сложных расчетных задач.

Конечно же, серьезные расчеты, для которых, собственно, и целесообразен компьютер, — дело более сложное. Они, как правило, требуют не примитивного режима «калькулятора», а целой программы работы ЭВМ. С правилами составления программ вы познакомитесь ниже. Чтобы написать программу, необходимо узнать вначале кое-что об используемом для этой цели языке БЕЙСИК.

Язык БЕЙСИК-MSX: работа с числами

Пошаговый режим «калькулятора», или непосредственного исполнения команд, в реальной работе на компьютере почти не применяется (разве что для отладки программ). Гораздо интереснее истинно программный режим, когда вы предварительно записываете (лучше всего сразу на экране) программу, т. е. последовательность действий

вашего ПК, а затем компьютер сам выполняет ее команды поочередно, не заставляя вас на каждом шагу понуждать его к работе клавишей ВК.

Программа состоит из командных строк — тех, длина которых, как было сказано, не должна превышать 255 символов. Чтобы ПК понял, что вы предлагаете ему действительно программу, а не простой приказ выполнить цепочку алгебраических действий в пределах одной командной строки, каждая командная строка программы должна начинаться номером. Скажем, вот так:
321? 2+2

Нажатие клавиши ВК теперь не даст вам долгожданного результата — числа 4, а всего лишь перебросит курсор к началу следующей командной строки (здесь клавиша ВК только сообщает компьютеру, что одна строка программы закончена, нужно быть готовым к вводу следующей). Забегая немного вперед, покажем, как получить-таки ответ: в начале следующей строки напечатайте слово RUN и вновь нажмите клавишу ВК, чтобы ввести это командное слово в «мозг» машины. Теперь компьютер исполнит эту маленькую программу, как и прежде, он вначале следующей экранной строки выдаст число 4.

Номера строк должны находиться в интервале от 0 до 65 529 и могут быть только целыми. Компьютер выполняет программу последовательно, в порядке возрастания номеров командных строк.

Вслед за номером строки пишутся команды или операторы языка БЕЙСИК, отделяемые друг от друга двоеточием (:). Попробуйте повторить теперь свои расчеты, сделанные в режиме «калькулятора», разбив вычисления на отдельные нумерованные строки и каждый раз запуская всю программу командой RUN. Номера строк совсем не обязаны быть последовательными числами: важно лишь монотонное их возрастание. Программисты даже нарочно нумеруют строки так: 10, 20, 30 и т. д., чтобы при необходимости после можно было вставить между ними новые, если это вдруг потребуется. Несущественен и порядок записи строк. Вполне можно сперва написать 20-ю, а затем 10-ю или 5-ю строку (компьютер потом сам расположит их в нужном порядке), хотя поступать так не очень удобно. Будьте особенно внимательны, чтобы не приписать новой строке уже использованный номер! Машина поймет это как ваше желание выбросить прежнюю строку с этим номером и заменить ее новым вариантом. Чтобы избавить вас от утомительного печатания номеров строк, компьютер снабжен режимом автоматической нумерации, о чем мы еще скажем.

Опишем теперь более тонкие понятия, необходимые для грамотного выполнения вычислений.

Константы (постоянные). Они могут быть не только числовыми, но и так называемыми символьными, или строковыми. Пока будем говорить лишь о числовых, которые бывают следующих типов: целые, с фиксированной и с плавающей десятичной точкой. Целые константы — это просто целые числа из интервала от −32768 до 32767. (Десятичной запятой, которая в литературе по компьютерам именуется точкой и именно так обозначается на письме, в них нет. Число 25 — целая константа, число 25.0 — не целая!) Константы с фиксированной точкой — действительные числа, положительные или отрицательные, записанные обычным образом: 25.1 или 11.0, или −0.5 и т. д. Константы с плавающей точкой — положительные или отрицательные действительные числа, записанные в так называемой экспоненциальной форме: константа с фиксированной точкой (или мантисса), умноженная на число 10 в целой степени. Такой способ записи используется в научной литературе, однако обозначения здесь несколько иные: 2.543 E 06 (в более привычных обозначениях это $2{,}543 \cdot 10^6$) или 1.5 E −03 (т. е. $1{,}5 \cdot 10^{-3}$) и т. п. Пробелы между мантиссой, знаком E и показателем степени не нужны — они добавлены нами просто для удобства чтения числа читателем, непривычным к новой системе обозначений.

Записанные в экспоненциальной форме числовые константы могут принимать значения от 10^{-64} до 10^{63}.

Кроме того, можно пользоваться еще 16-теричными константами (в системе счисления с основанием 16), ко-

торые обозначаются приставкой &H (пример: &H98AC). Используемые здесь «цифры» — это 0,1,...,9 и дополнительные символы — латинские буквы A(10), B(11), C(12), D(13), E(14) и F(15). Восьмеричные константы обозначаются приставкой &O (пример: &123). Обратите внимание, что здесь O — буква, а не цифра. Компьютер всегда четко различает буквы и цифры, а вот пользователи порой путают. Поэтому программисты придумали специальное обозначение нуля, явно отличающее цифру от сходной по начертанию буквы: O — буква, 0 — цифра. И наконец, двоичные константы обозначаются приставкой &B (пример: &B01001).

Точность. Числовые константы могут храниться в памяти машины и обрабатываться либо с обычной точностью (6 значащих цифр до и после десятичной точки), либо с двойной (14 значащих цифр). Естественно, что двойная точность делает результаты расчетов более точными, однако за это приходится расплачиваться снижением скорости вычислений и менее экономным использованием объема памяти компьютера. Поэтому пользователь должен всякий раз выбрать, что же для него в данном случае важнее. По умолчанию (т. е. если не сделано специальных указаний) компьютер приписывает традиционной записи числовой константы двойную точность.

Сформулируем правила назначения степени точности строже: обычная точность используется в случае экспоненциального представления константы (с буквой E) или в случае, когда запись числа завершается восклицательным знаком (пример: 43.678!). Двойная точность используется при традиционной записи числовой константы (без специальных указаний на иное), либо при экспоненциальной форме, где буква E заменена буквой D, либо если запись числа завершается знаком # (примеры: 1234, 12.55, −15, 5.666D−07, 123.543#).

Переменные. В отличие от констант переменные не имеют постоянного, единожды определенного числового или иного значения, а получают его в нужный момент выполнения программы. Если какая-то переменная была введена программистом (компьютер узнает об этом по появлению имени переменной в том или ином месте программы), то первоначально ей автоматически присваивается нулевое значение.

Имя переменной — это любая выбранная программистом, т. е. просто вами, коль скоро вы взялись составлять компьютерную программу, цепочка символов, непременно начинающаяся с латинской буквы: A12345 — имя некоей переменной, 1A2345 — нет. Вслед за первой буквой можно написать любое число букв и цифр, однако компьютер различает переменные лишь по двум первым символам имени. Иначе говоря, для машины имена AB и ABBBBBBBBBBBA будут обозначать одну и ту же переменную. Помните об этом! Какой же резон писать буквы или цифры после второго символа? А вот какой: человек все-таки не машина, и ему легче будет различать имена переменных, если они как-то намекают на вкладываемый в переменную смысл. Поэтому программисты часто пишут не просто X5, а скажем, X5MAX — такая форма станет напоминать вам, что это какой-то максимум, хотя для компьютера имена X5 и X5MAX совершенно одинаковы.

Еще одно ограничение: внутри имени переменной не должно быть так называемых зарезервированных слов (их список включает команды, операторы языка БЕЙСИК, названия распространенных математических функций). То есть нельзя выбирать имя переменной вроде A125RUNMAX: его часть — слово RUN, которым вы уже пользовались как командой, является запрещенной. Конечно же, недопустимо и имя, состоящее из одного только зарезервированного слова. Из сказанного вы можете заключить, что компьютер все же читает имя переменной целиком, хотя потом в его «сознательной» памяти сохраняются лишь два первых символа (все экономия!).

Переменные, как и константы, могут принимать числовые и символьные (строковые) значения. Пока говорим только о числовых. В этом случае вновь возникает проблема компромисса между достигаемой точностью резуль-

татов расчета и затратами времени на вычисления, занимаемым объемом машинной памяти. И программисту даны на выбор три степени точности: минимальная — целочисленные переменные (обозначаются знаком % после имени: X5%), переменные обычной точности (ABBA5!) и двойной точности (BAA#). Видно, что обозначения типов переменных те же, что и для констант. И точно так же, если не сделано специальных указаний на обратное, любой переменной компьютер приписывает по умолчанию двойную точность.

Чтобы вам наглядно представить, зачем выдумана вся эта сложность с типами переменных и констант, скажем, что в памяти компьютера целочисленная переменная занимает 5 байт (байт — единица объема информации, равная 8 битам), переменная обычной точности — 7 байт, а двойной точности — 11 байт. Если в программе, где по сути требуются лишь целые значения, вы забудете указать это компьютеру соответствующим прибавлением знака %, машина, не понимающая смысла вычислений, механически станет считать все в режиме двойной точности, а значит, доступная вам емкость памяти (для записи текста программы и результатов) сократится ни много ни мало вдвое. Без какого-либо выигрыша в реально требуемой точности счета! Так что стоит задуматься.

Существует особый тип переменных, называемых массивами. Они обозначаются по-другому, например так: A(7) — элемент одномерного массива под номером 7, или B(2,5) — элемент двумерного массива, расположенный на пересечении строки номер 2 со столбцом номер 5, и т. д. Такие обозначения позволяют короче и осмысленнее представлять многочисленные переменные программы, экономить машинную память и свои силы. Число измерений («размерность») массива не должно превышать 255, а наибольшее число его элементов определяется доступной емкостью памяти.

Правила, по которым компьютер преобразует переменные одного типа в другой, вполне определенны, хотя несколько утомительны для первоначального чтения. Поэтому, упомянув о них, мы воздержимся от изложения их здесь, отослав интересующегося читателя к более полным руководствам.

Операторы. Могут быть арифметические операторы, операторы сравнения, логические и функциональные.

С большинством арифметических операторов (^, — ,*, /, + и —) вы уже знакомы. (Операторы перечислены в том порядке, в каком компьютер будет их выполнять, если они встретятся в расчете одновременно. Первый минус обозначает операцию перемены знака числа.) Но есть еще два особых арифметических оператора.

Оператор целочисленного деления: Y. (Иное обозначение: / — обратная косая.) Действует он так. Делимое и делитель прежде всего преобразуются в целые числа (путем отбрасывания дробных частей), затем выполняется деление и у частного тоже отбрасывается дробная часть. Примеры: 12.5Y5.7 даст в результате число 2,10Y7 — число 1, 25Y5 — разумеется, число 5. Обратите внимание, что числа при этом не округляются до ближайших по привычным правилам, а находится их целая часть.

Оператор MOD. Дает остаток в процедуре целочисленного деления. Пример: 10.8MOD 8.93 даст в результате число 2 (остаток от деления 10 на 8), 32 MOD4 — число 0.

Операция целочисленного деления выполняется вслед за умножением и делением обычного вида, а оператор MOD — за целочисленным делением.

Операторы сравнения: =(равно), <>(не равно), <(меньше), >(больше), <=(меньше или равно), >=(больше или равно). Они выполняются вслед за арифметическими операторами и обычно используются в условных конструкциях типа «если... то». Результат сравнения определяет собой дальнейшее выполнение программы. Например: если переменная X5=0, то перейти к выполнению строки программы под номером 1005. Компьютер проверит в нужный момент, равно ли значение переменной X5 нулю, и перейдет (либо не перейдет, если условие не выполнено) к исполнению строки программы под указанным номером.

Самыми последними будут выполнены логические операторы. Их шесть:

NOT, AND, OR, XOR, EQV и IMP. Чаще всего используются первые три — НЕ, И, ИЛИ. Детальное описание результатов действия логических операторов мы также пока отложим, чтобы сэкономить место для других сведений.

Функциональные операторы — это, проще говоря, определенные заранее функции, применяемые к тем или иным переменным. Язык БЕЙСИК-MSX позволяет программисту использовать и стандартные математические функции, и функции, определяемые самим пользователем. Приведем перечень стандартных функций, используемых при работе с числовыми переменными:

ABS(X) — дает абсолютную величину X.

ATN (X) — дает арктангенс X — в радианах, результат будет лежать в интервале от $-\pi/2$ до $\pi/2$. Числовая переменная X может быть любого типа, однако значение арктангенса всегда будет иметь двойную точность.

CDBL(X) — превращает X в число двойной точности.

CINT(X) — превращает X в целое число (путем отбрасывания дробной части). Если X выходит за пределы интервала от −32768 до 32767, компьютер выдает сообщение об ошибке переполнения «Overflow».

COS (X) — дает косинус X (X — в радианах). Значение косинуса имеет двойную точность.

CSNG(X) — превращает X в число обычной точности.

EXP (X) — дает значение функции e^X. Величина X не должна превосходить 145.06286085862, в противном случае компьютер выдает сообщение об ошибке переполнения — «Overflow».

FIX(X) — дает целую часть X (дробная часть отбрасывается. FIX(X)=SGN(X)* INT (ABS (X)). Отличие между сходными функциями FIX и INT проявляется при отрицательных X: FIX даст целое, большее или равное X, а INT — целое, меньшее или равное X.

INT (X) — дает наибольшее целое число, не превосходящее X.

LOG(X) — дает значение функции lnX. X не должен быть отрицательным.

RND(X) — дает «случайное» число из интервала (0,1). X определяет свойства получаемых последовательностей этих чисел.

SGN(X) — дает значение 1 при положительных X, значение 0 при X=0 и значение −1 при отрицательных X.

SIN(X) — дает синус X (X — в радианах). Значение синуса имеет двойную точность.

SQR(X) — дает квадратный корень X. X должен быть неотрицательным.

TAN (X) — дает тангенс X (X — в радианах). Значение тангенса имеет двойную точность.

Язык БЕЙСИК-MSX: работа с символами

Прежде чем переходить к рассказу о том, как составлять программы для компьютера, мы хотим выполнить свое прежнее обещание — сказать несколько слов о работе с символами. Говоря о константах и переменных, мы упоминали, что будем иметь в виду числа, правила обращения с которыми известны вам из обычной арифметики и алгебры. Но компьютер может работать не только с числами. Есть еще один важный тип переменных или констант, которые мы станем называть символьными (в других книгах их именуют также текстовыми, строковыми, а иногда даже литералами).

Возьмем простой пример: номер телефона, скажем, 1234567. Он составлен из цифр и на первый взгляд может показаться числом. Однако правила обращения с числами к телефонным номерам совершенно неприменимы. Действительно, ну какой смысл, например, складывать два телефонных номера или вычитать один из другого? Ясно, что разумного результата такая «телефонная арифметика» не даст. Номер телефона нужно рассматривать не как число, а именно как символьную величину — просто цепочку цифр, оперировать с которой следует по особым правилам. И такие правила есть. Скажем, отделяя первые три цифры — 123, вы получите номер обслуживающей этот телефон районной АТС. По нему можно отыскать в телефонном справочнике различные интересные сведения: подключен ли ваш телефон к системе автоматической междугородной связи, куда обращаться в случае поломки, где

принимают абонементную плату. В некоторых городах сам вид телефонного номера несет дополнительную информацию: например, телефоны учреждений семизначные, а квартирные телефонные номера состоят всего из шести цифр. То есть лишь взглянув на номер, вы можете сделать вывод о его владельце. Неважно, каковы именно правила в каждом частном случае, важно, что они существуют и не совпадают с условностями числовой арифметики и алгебры.

В языке БЕЙСИК-MSX символьная константа — это последовательность алфавитно-цифровых и иных символов, число которых не должно быть более 255. Компьютер станет обращаться с ней именно как с символьной, если вы проинформируете его об этом, заключив константу в кавычки. Примеры: «Вход воспрещен», «АВВА», «123ВА», «123-45-67», «1234567» (последний пример — символьная константа — быть может, номер чьего-то телефона, а та же запись без кавычек рассматривалась бы компьютером просто как число).

Проверьте сказанное самостоятельно: заставьте компьютер выполнить две простейшие программы:

? 2+ 2 (ВК)
? «2 + 2» (ВК)

В первый раз ответом на ваше задание будет появление на следующей строке экрана числа 4 — результата арифметического сложения двух чисел. Во второй раз компьютер напечатает в ответ вашу символьную константу неизменной: 2+2. Оператор PRINT (обозначенный как ?) печатает символьные константы, т. е. все, заключенное между двумя знаками кавычек, без изменения. Этим можно пользоваться для красивого оформления результата наших сложных вычислений:

? «Вот последние агентурные данные:
2 + 2 =»; 2 + 2 (ВК)
И компьютер даст развернутый ответ:
Вот последние агентурные данные:
2 + 2 = 4

Символьной константой может быть и пробел („ “) и даже полное отсутствие каких-либо знаков, так сказать, пустой символ („“), который тоже будет воспроизведен оператором PRINT совершенно точно: на экране ничего не появится, и курсор по-прежнему будет занимать первую позицию на строке (в случае печатания пробела он сдвинется на один шаг вправо).

Имя символьной переменной выбирается по общему правилу (первый символ — непременно латинская буква) и дополняется знаком $: A$, ABBA$, A123BBBBBB$ и т. д.

Символьные константы, значения символьных переменных можно «складывать» (или сцеплять, если использовать верный термин). Такая операция обозначается знаком + и дает в результате ее выполнения следующие друг за другом символьные «слагаемые», пригнанные без зазора, словно кирпичи в плотной кладке. Заставьте компьютер продемонстрировать вам действие этой операции сцепления:

? „Посторонним“ + „ “ + „вход“ + „воспрещен“ + „!“ (ВК)

Если вы напечатали задание верно, в результате получите ходовую фразу. Можно отпечатать ее на принтере и повесить на какой-нибудь двери. Попробуйте сами объяснить смысл второго «слагаемого» и то, зачем в составе третьего есть пробел, тогда как в первом и четвертом пробела нет.

Кроме того, уже знакомые нам по работе с числами операторы сравнения (равно, не равно, больше, меньше и т. д.) применимы и к символьным величинам. Помня все, что мы сказали выше об особом характере символьных величин, вы должны бы прийти в недоумение: как это можно сравнивать нечто, не имеющее численного значения? Действительно, что больше — слово «четыре» или номер «123»? Бессмысленный вопрос. Сравнение будет иметь смысл лишь после того, как символьным величинам припишут какие-то числа, с которыми уже можно оперировать по правилам арифметики.

И такие числа действительно приписаны всем символам, имеющимся на клавиатуре компьютера. Собственно говоря, компьютер как чисто арифметическое устройство вообще ничего не знает ни о буквах, ни о других знаках

вроде запятых, пробелов, кавычек и т. д. Он работает только с их номерами. Есть довольно объемистая таблица (256 символов), называемая таблицей кодов ASCII и служащая компьютеру алфавитом. Изготовители компьютеров договорились размещать в этой таблице буквы, цифры и все прочие знаки (в том числе и те, которые на экране вообще никак не отображаются, хотя машина их воспринимает и различает) в некоем стандартном порядке. То есть каждому знаку присвоили собственный номер — от нуля до 255. Прописная буква «А» своим номером будет отличаться от строчной «а», русская «А» от «А» латинской, несмотря на то что последние две выглядят одинаково (предусмотрительные проектировщики компьютеров иногда догадываются изображать сходные буквы русского и латинского алфавитов чуть-чуть по-разному, чтобы программист мог легко различить их на глаз).

Именно по этим номерам и производится сравнение символьных величин. Правила сопоставления таковы. Берутся первые символы двух сравниваемых величин. Большей считается та символьная величина, чей код первого символа оказался больше. Если коды первых символов одинаковы, для сравнения берутся два вторых символа, все снова повторяется и т. д. Когда в процессе подобного перебора одна из символьных величин, скажем слово, кончилась, процедура останавливается: более короткое слово считается «меньшим» более длинного («Арба» < «Арбалет»).

Примечание. Пробел — вполне законный символ компьютерного алфавита. Ему, между прочим, присвоен номер (в латинской части таблицы — 32). То же можно сказать о «нулевом», или «пустом», символе с номером 0. Поэтому, например, «ABBA » > «ABBA » (пробел дает «довесок»).

Цифры в таблице кодов ASCII расположены в их естественном порядке, поэтому слово «123» меньше слова «223», как и соответствующие числа 123 и 223. Латинские буквы размещены в порядке привычного алфавита — сначала все прописные, потом все строчные, так что слово «Ааа» будет меньше, чем «ааа». К сожалению, в русской части кодовой таблицы ASCII порядка несколько меньше, чем в латинской. Более или менее установившегося стандарта тут до сих пор нет, и поэтому вам придется прежде всего поинтересоваться, как расположены русские буквы в кодовой таблице вашего компьютера. Например, компьютеры «Ямаха MSX» и «Ямаха MSX-2» сделаны не совсем удобно: русские буквы у них размещены не в привычной алфавитной последовательности. Компьютеру, да и программисту это безразлично до тех пор, пока не захочется использовать программы сортировки вроде расставления фамилий в списке по алфавиту. Тогда придется изучить расположение символов в таблице ASCII, чтобы не наделать ошибок.

В заключение главы о работе с символами перечислим некоторые функции, переменными которых являются символьные величины:

ASC(X$) — дает в результате число, равное номеру первого символа X$ в таблице ASCII. X$ не должен быть пустым символом, иначе компьютер выдаст сообщение об ошибке.

BIN$(X) — дает в результате двоичную запись десятичного числа X. Внимание: вы получаете здесь не число, а символьную величину — цепочку двоичных цифр, и работать с ней далее нужно именно как со «словом». Преобразовать это «слово» в число позволяют специальные приемы.

CHR$(X) — дает символ, номер которого в таблице кодов ASCII равен X.

HEX$ (X) — дает в результате шестнадцатеричную запись десятичного числа X. Здесь, как и в случае функции BIN$(X), получается символьная величина.

INSTR (X$, Y$) — дает число, равное номеру символа, с которого внутри X$ содержится Y$. Пример: X$ = «компьютер», Y$ = «пью»; INSTR(X$, Y$) даст в результате число 4, поскольку слово «пью» начинается внутри слова «компьютер» с 4-го символа от начала. В другом варианте эта функция записывается как INSTR(N, X$, Y$). Здесь число N — номер символа, с которого следует начинать поиск слова Y$ внутри X$ (1<=N<=255).

LEFT$ (X$, N) — дает в результате усеченную символьную величину X$: сохраняются лишь N первых символов. Число N должно быть в интервале от 0 до 255.

LEN(X$) — дает число отдельных символов, составляющих символьную величину X$. Учитываются все символы, включая пробелы и «невидимые» знаки (не отображаемые на экране).

MID$ (X$, N, M) — дает усеченную символьную величину X$: сохраняется ее средняя часть длиной в M символов, начиная с символа номер N. N и M должны быть в интервале от 1 до 255. Если в записи функции число M опущено или если оно превышает число имеющихся в правой части X$ символов, эта функция дает просто правую часть символьной величины X$, начиная с символа под номером N.

OCT$ (X) — дает в результате восьмеричную запись десятичного числа X. Здесь, как и в случае функции BIN$ (X), получается символьная величина.

RIGHT$ (X$, N) — дает усеченную символьную величину X$: сохраняются лишь N последних символов. Число N должно быть в интервале от 0 до 255.

Команды языка БЕЙСИК-MSX

Теперь познакомившись с принципами обработки чисел и символьных величин (проще говоря, слов) на компьютере, нам для составления первых программ необходимо узнать и запомнить хотя бы самые распространенные команды языка БЕЙСИК. Без этого написать программу длиной более 1—2 командных строк практически невозможно.

Следуя избранной линии простоты изложения, авторы решили рассказывать о командах тоже попроще. Но язык общения с компьютером требует предельной строгости: важна любая запятая, точка, кавычка. Это часто вынуждает берущихся описывать команды вводить довольно громоздкие обозначения, охватывающие всевозможные варианты. В ходу даже такие мудреные термины, как метасимволы! Мы поступим иначе. В заголовке каждой статьи, посвященной одной команде, мы выпишем сразу несколько конкретных примеров (с реальными параметрами) записи этой команды. Постараемся, разумеется, не терять полноты описания, однако вместо обобщенных X или X$ станем пользоваться просто выбранными наугад числами или словами. Это позволит читателям наглядно представить себе вид верной записи, но не следует все-таки забывать, что параметры — числа и слова — здесь играют роль иллюстраций и что в своей программе нужно будет подставить вместо них соответствующие другие величины.

AUTO AUTO 15
AUTO 25,5 AUTO 35,

Эта команда включает автоматический нумератор командных строк. Набрав на клавиатуре AUTO и нажав клавишу ВК, вы увидите, что в начале следующей экранной строки появился номер 10. Можете печатать за ним команды своей программы. Следующее нажатие ВК даст вам номер второй командной строки — 20 и т. д. (Выше мы объясняли, почему программисты нумеруют строки программы десятками.) Команда AUTO 15 начинает нумерацию с 15, по-прежнему добавляя каждый раз по 10. AUTO 25,5 начнет нумерацию с 25, но добавлять станет уже не по 10 (исходный режим), а по 5, как было указано вами в команде. Если вы введете команду в таком «урезанном» виде: AUTO 35, (внимание: запятая осталась, но второго числа нет!) — тогда компьютер начнет нумерацию с 35, добавляя всякий раз приращение, установленное вами прежде (здесь — 5). Введя просто AUTO, вы вернетесь в исходный режим.

Может случиться так, что вы вспомнили о команде AUTO где-то в середине составления программы. При этом автоматический нумератор может выдать номер, который однажды уже был использован вами во время «ручной» нумерации. Компьютер отметит этот прискорбный факт звездочкой после опасного номера: 25*. Если вы не желаете, чтобы прежняя командная строка с тем же номером была уничтожена, немедленно после появления звездочки нажмите клавишу ВК (печать любого символа после звездочки воспринимается машиной как намерение записать измененный вариант старой строки).

Выключить автоматический нумератор можно, одновременно нажав клавиши CTRL и C (латинское) или CTRL и STOP. Последняя комбинация заслуживает того, чтобы ее запомнили: она позволяет выйти из самых разных режимов, почему-либо надоевших программисту.

CONT

Команда продолжения выполнения программы, прерванной командами типа STOP.

DELETE 35 DELETE 20–40

Уничтожает командные строки с указанными номерами (строку 35 или все строки с 20-й по 40-ю включительно). Если в программе нет строк с такими номерами, компьютер напечатает сообщение об ошибочном действии программиста.

LIST LIST 25 LIST 30–50
LIST–50 LIST 30–

Заставляет компьютер распечатать на экране программу или ее часть. При этом машина предварительно выполнит черновую работу — сотрет замененные вами строки, расположит номера строк в порядке возрастания. Просто LIST распечатывает программу целиком, LIST 25 — единственную строку номер 25, LIST 30–50 — строки с 30-й по 50-ю, LIST– 50 — начальную часть программы до 50-й строки включительно, LIST 30 — конечную часть, начиная с 30-й строки.

Распечатка программы — эту процедуру, как и ее результат, программисты порой называют на английский манер листингом — может быть прекращена одновременным нажатием клавиш CTRL и STOP. Чтобы вновь начать распечатку (сначала), придется опять послать команду LIST. Клавиша STOP дает возможность временно приостановить процесс распечатки. Нажав STOP еще раз, вы позволяете компьютеру продолжить прерванное дело.

LLIST

Эта команда подобна команде LIST, но включает в работу принтер, печатающий текст вашей программы на бумаге.

NEW

Стирает в памяти компьютера всю программу целиком — ее команды, значения переменных и констант. Ее следует посылать всякий раз, когда вы собираетесь составлять новую программу, а старую уже зафиксировали на бумаге или магнитном диске или просто желаете стереть бесследно. Прежде чем нажать ВК после NEW, подумайте: не стираете ли вы из памяти машины что-то нужное!

RUN RUN 50

Уже отчасти знакомая вам команда. Она заставляет компьютер выполнить составленную вами программу с первой до последней строки. Если возникает надобность исполнить лишь часть программы, применяют запись типа RUN50 — тогда программа будет выполняться со строки номер 50 до конца.

Команды, используемые в программе (операторы)

CLEAR

«Сбрасывает» значения всех использованных в программе переменных. Числовые переменные при этом приравниваются нулю, символьные — «пустому» символу.

DATA 1,25,3.17, Alarm, «1970, 1971»

Команда, обозначающая «банк данных» вашей программы. Вслед за ней перечисляются необходимые в программе константы, разделяемые запятыми. Предельная длина этого перечня — одна командная строка, то есть 255 символов. Если весь ваш список требуемых констант не уместился в командную строку, начните следующую нумерованную строку программы еще одним словом DATA и продолжайте перечисление. И т. д. Когда компьютеру понадобится по команде READ прочесть этот список, он станет двигаться по нему последовательно от начала к концу, перебирая командные строки, начинающиеся словом DATA, в порядке возрастания их номеров (номера совсем не обязаны быть последовательными: между этими строками могут располагаться иные части программы).

Список констант может содержать числовые константы в любом представ-

лении (с фиксированной десятичной точкой, с плавающей точкой, целочисленные — об этом дальше). В списке допустимы и символьные константы, причем здесь их даже не обязательно всякий раз выделять снаружи кавычками (кавычки обязательны лишь тогда, когда в составе символьной константы есть запятые, кавычки, двоеточия, начальные или конечные пробелы, которые требуется сохранить). Это объясняется тем, что указания, как трактовать тип константы — числовая она или символьная, хотя по виду это не всегда можно определить, — содержатся в команде использования такого списка (READ).

END

Команда, завершающая выполнение программы и возвращающая компьютер в режим исходного ожидания (знак Ok). Впрочем, в конце программы ее можно и не помещать — она там всегда подразумевается. END в середине программы будет останавливать ее выполнение, и этим приемом иногда пользуются при отладке программ, чтобы исполнить лишь кусок от начала до строки с END.

FOR...NEXT

FOR X=25 TO 125 STEP 5... NEXT X

Этот сложный на вид оператор называется оператором цикла. Он настолько важен и вам так часто придется им пользоваться, что поневоле мы вынуждены поговорить о нем более подробно. Да он только с непривычки кажется чрезмерно запутанным, по сути же ничего особо сложного в нем нет. Что делает компьютер, встретив в программе подобную конструкцию? Во-первых, он присваивает переменной X значение 25. Во-вторых, смотрит, какой задан шаг, и действительно находит в нашем примере, что шаг (STEP) выбран равным 5. В-третьих, определяет для себя на будущее последнее значение X (125). После этого начинаются собственно циклы: X=25, компьютер выполняет какую-то операцию, обозначенную здесь многоточием (скажем, вычисляет и печатает квадрат X или делает еще что-нибудь — звонит в звонок и т. п.) — 1-й цикл закончен. Затем берется новое значение X (старое плюс шаг), т. е. X=25+5=30, и все предыдущее повторяется еще раз — 2-й цикл закончен. И так далее, пока на одном из циклов значение X не окажется равным 125. Тогда компьютер, в последний раз проделав помещенное в программе вместо многоточия реальное задание, закончит эту круговерть и перейдет к выполнению следующей командной строки программы. Вот что такое оператор цикла.

Значение шага в операторе цикла может быть и отрицательным — тогда компьютер станет вычитать вместо прибавления его к X. Указание о величине шага может быть вообще опущено. В этом случае компьютер по умолчанию предполагает шаг равным 1. Можно опустить и обозначение переменной цикла после слова NEXT: машина тогда станет считать оператором цикла отрезок программы между этим NEXT и ближайшим к нему сверху FOR. Однако намеренно поступать так не рекомендуется: затраты труда на явное обозначение переменной невелики, а отсутствие четкого обозначения способно в иных случаях запутать и программиста и сам компьютер. Правда, сравнительно простые промахи программиста машина успешно находит. Если вы увидите на экране сообщение об ошибке «NEXT without FOR», знайте: в вашей программе оказалось слово NEXT, выше которого ни в одной строке нет необходимого ему FOR. (Конструкцию FOR...NEXT можно размещать не в одной, а в нескольких следующих подряд командных строках. Так обычно и делают для более красивого, а значит, и более понятного оформления текста программы.)

Некоторая тонкость. В нашем примере ситуация сравнительно простая: последнее значение X как раз совпадает с установленной нами границей изменения переменной (125). А что было бы, если б мы выбрали эту границу равной, например, 126? Переменная X, изменяемая от 25 пятерками, однажды примет значение 125, а затем 130, так и не попав в 126. Как ведет себя компьютер в такой ситуации? Вот как: он в каждом цикле **перед** исполнением задания, обозначенного многоточием, проверяет, не стало ли новое значение переменной X больше установленной

границы. Если не стало, задание выполняется и начинается новый цикл. Если же при такой повторяющейся проверке X окажется больше границы (меньше, когда шаг всякий раз уменьшает X), задание на этот раз **уже не выполняется,** компьютер заканчивает свою возню с данным оператором цикла и переходит к исполнению командной строки, следующей за строкой со словом NEXT. То есть результат нашего примера не изменится от замены границы 125 на 126.

Циклы вида FOR...NEXT могут быть вложены друг в друга. Внимательно следите — как при использовании вложенных одно в другое выражений со скобками, — чтобы каждое начало (FOR) имело соответствующий ему конец (NEXT). Высказанная выше рекомендация не опускать обозначения переменной цикла, даже когда формально это не требуется, и выбор различных имен для переменных различных вложенных операторов цикла помогут вам не запутаться и, следовательно, получить программу с меньшим числом ошибок. Переменная цикла, как правило, нужна только внутри оператора FOR...NEXT, и по завершении исполнения оператора она «умирает». Это иногда побуждает начинающих программистов «экономить» на буквах, обозначая все переменные циклов, скажем, через X. Результат такой неразумной экономии — запутанная программа, в которой уже не способен разобраться и сам ее автор. А это лучший способ наделать много ошибок.

GOSUB 250

Такая команда заставляет компьютер перейти к выполнению маленькой программы, входящей в состав всей вашей программы автономно. Из стремления к ясности и по возможности безошибочности программ хорошие программисты складывают их из отдельных «кирпичиков» — блоков, которые можно назвать подпрограммами. Эта команда отсылает компьютер к одной из таких подпрограмм, начинающейся с командной строки под номером 250. Закончив все, что требуется сделать в рамках подпрограммы (скажем, провести некую подготовительную работу, результаты которой будут затем — и может быть, даже неоднократно — использованы основной программой), компьютер встретит там завершающее слово RETURN и немедленно перейдет к выполнению той командной строки программы, которая следует сразу же за строкой с командой GOSUB. Подобные челночные операции, как и обработка циклов, могут быть вложены одна в другую, так что от программиста требуется немалое внимание для исключения ошибок.

GOTO 120

Подобная предыдущей команда перехода. Здесь компьютер переключает свое внимание на исполнение командной строки под номером 120. Выполнив, если к тому нет препятствий, инструкции названной 120-й строки, машина уже не возвращается назад, а переходит к командным строкам, следующим за 120-й. Если по какой-то причине команды 120-й строки в данный момент неисполнимы, компьютер ищет, перебирая следующие командные строки, ближайшую к строке 120 исполнимую строку и берется за нее, потом за следующую и т. д. Таким образом можно обойти группу строк программы, изменить порядок выполнения строк, отослав компьютер по программе вверх, и т. п.

IF... THEN... ELSE...

IF x<0 THEN? «—» ELSE? «+»

Это весьма важный оператор условного перехода. На нем тоже имеет смысл остановиться более подробно, поскольку он применяется чрезвычайно часто. Смысл оператора несложен: **если** (IF) — **затем** следует условие, объясняющее, что именно может тут случиться, — **то** (THEN) — указывается, что должно быть при этом условии сделано или к какой командной строке программы следует при этом перейти (ее номер); **в противном случае** (ELSE) — указывается, что делать, когда названное условие не выполнено или к какой строке программы следует перейти (номер строки).

В нашем примере словесное описание оператора выглядит так: если переменная X в данный момент оказывается меньше нуля, то компьютеру следует напечатать на экране знак —; в против-

ном случае (если X окажется не меньше нуля) — напечатать знак +. Обратите внимание, что знак + будет при этом печататься не всегда уместно. Например, когда x=0, условие оператора не выполняется и компьютер отпечатает знак +. Наша вина!

После слов THEN и ELSE может следовать целая цепочка операторов-указаний: сделать одно, затем другое, третье и т. д. Если же указывается номер программной строки, где описаны такие указания, то он, разумеется, должен быть единственным.

Проверив выполнение условия и исполнив затем одну из явных инструкций, содержащихся в операторе условного перехода (обе инструкции никогда не могут быть выполнены, ведь соблюдаться и не соблюдаться одновременно условие, если оно сформулировано грамотно, не может), компьютер ищет и начинает исполнять ближайшую — исполнимую — следующую строку программы. Когда инструкции заданы неявно (указан лишь номер командной строки), машина переходит к выполнению соответствующей строки программы и следующих за ней строк, больше не возвращаясь к оператору IF ... THEN ... ELSE.

Эти операторы условного перехода также могут быть вложены один в другой — насколько позволяет предельная длина командной строки (255 символов). И здесь от программиста требуется особое внимание, чтобы не запутаться в многочисленных разветвлениях.

Замечание. Альтернативная часть ELSE оператора условного перехода может быть опущена, останутся только IF и THEN. В этом случае компьютер проверяет выполнение условия и при положительном исходе проверки исполняет инструкции, следующие за THEN, а при отрицательном исходе сразу переходит к ближайшей следующей — исполнимой — строке программы.

ЯЗЫКИ программирования

М. П. МАЛЫХИНА,

А. П. ЧАСТИКОВ

В октябре 1963 г. представители фирм IBM и SHARP образовали Комитет под председательством Б. Розенблата и Дж. Радина по созданию совершенного языка программирования. Предполагалось создать такой язык, который сможет удовлетворить различные классы пользователей, решающих как научные, так и коммерческие (экономические), а также специальные задачи. В результате разработки эффективного «единственного» языка программирования ожидалось, что отпадет необходимость в таких языках, как ФОРТРАН, КОБОЛ и др.

Первой попыткой создания универсального языка стала разработка в начале 60-х годов фирмой System Development языка ДЖОВИАЛ (автор Ж. Шварц и др.). Но этот язык, использовавшийся исключительно для военных приложений, не смог претендовать на роль универсального.

Первоначально разработка ПЛ/1 ориентировалась на обеспечение соответствующих расширений ФОРТРАНА, он и назывался ФОРТРАН-IV. Скоро стало ясно, что совместимость с ФОРТРАНом сохранить не удастся, не удастся также достичь целей, определенных для нового языка.

Одним из аспектов начального этапа разработки языка ПЛ/1 была быстрота, с какой предполагалось завершить описание языка — декабрь 1963 г. В дальнейшем сроки были отодвинуты сначала на январь, а затем на февраль 1964 г. Причиной такого жесткого графика было стремление приурочить создание ПЛ/1 к выпуску семейства совместимых ЭВМ, известных теперь под названием IBM/360.

Первый официальный документ с описанием нового языка программирования был выпущен в марте 1964 г., второй — опубликован в июне 1964 г. В январе 1965 г. П. Рогоуэй и Дж. Радин подготовили окончательный отчет Комитета по созданию нового языка.

Новый язык разработчики нарекли NPL (аббревиатура от слов New Programming Language — новый язык программирования). Это название существовало вплоть до 1965 г., а затем, по предложению английских пользователей из Национальной физической лаборатории, язык стал называться ПЛ/1 (PL/1 — сокращение от Programming Language / One — язык программирования / первый).

Осенью 1964 г. в нью-йоркском центре программирования фирмой IBM было учреждено правление по контролю над новым языком, которое через два года переехало в Херсли (Англия). По мере того как члены

```
INPUT X                    INPUT X$
INPUT «ЖДУ ЧИСЛА И СЛОВА»; X, X$
```

Эта команда позволяет вам во время выполнения программы общаться с компьютером — вводить что-то с клавиатуры. Первый пример заставит машину приостановить исполнение программы и вывести на экран знак вопроса (?). Тем самым компьютер показывает, что он ждет от вас реакции. Составляя программу, вы в данном случае указали, что вводиться с клавиатуры будет число. Напечатайте на экране какое-то число и не забудьте нажать затем клавишу ВК (иначе как же компьютеру узнать, что вы действительно закончили ввод данных, а не просто думаете, какую еще цифру допечатать). Ожидание тут же кончится — программа приняла число и выполняется дальше, причем переменной X будет присвоено введенное вами числовое значение. Можно предусмотреть в этой команде ввод нескольких чисел: INPUT, X, Y, A, B. Тогда компьютер будет запрашивать вас до тех пор, пока не получит всю нужную информацию.

Попробуйте в описанном примере напечатать в ответ на призыв машины (?) не число, а слово. Нажмите ВК — что получилось? Ошибка произошла оттого, что в команде INPUT вы пользовались обозначением числовых переменных, а ввели символьную величину. Наш второй пример, где обозначение переменной указывает на ее символьный характер, позволяет вводить слова. Конечно, вы теперь можете напечатать и какое-то число, скажем, 1234567, но машина истолкует его как символьную величину и станет оперировать с ней соответственно (помните, что говорилось выше о номере телефона?). Так что будьте внимательны.

Третий пример самый сложный, но зато и самый нужный для реальной работы. Здесь перед обозначениями вводимых переменных помещен текст «подсказки» (он обязательно заключается в кавычки и завершается точкой с запя-

ПЛ-1

правления осуществляли свою деятельность, направленную на «очищение» и усовершенствование языка, стало очевидно, что ни один из документов по разработке языка, подготовленных в 1964 г., не мог служить основой, на которую можно было опереться в дальнейшей работе. Сотрудники лаборатории из Херсли выполнили немалую работу по становлению нового языка. Они не ограничились просто его «очищением» и расширением. Они взяли, можно сказать, сырой материал и на протяжении двух лет занимались его обработкой, оформлением и «облицовкой», в результате чего получился хорошо определенный, пригодный к широкому использованию язык программирования.

Характеризуя язык ПЛ/1, его часто сравнивают с известными предшественниками, причем говорят, что он сочетает в себе блочную структуру и динамическое распределение памяти АЛГОЛа, структуры записей и средства ввода-вывода КОБОЛа и арифметические операции ФОРТРАНа, а также располагает некоторыми дополнительными средствами: обработкой строк, списков и др.

Создание языка ПЛ/1 было направлено на более эффективное использование компьютерной техники нового поколения: язык ПЛ/1 был первым, с помощью которого можно было решать проблемы, возникающие при взаимодействии с операционной системой. Он лучше, чем какой-то другой язык, обеспечивал бóльшие удобства распределения памяти, управления задачами и обработки исключительных ситуаций. Например, пользователь мог инициировать асинхронное выполнение задач и управлять процессом решения, основываясь на таких факторах, как временные задержки или завершение другой задачи. ПЛ/1 привнес идею родовых функций, что означает использование одного имени функции для различных типов входных данных. К важной особенности ПЛ/1, отличающей его от других языков, относится широко используемая концепция условий по умолчанию, которая позволяет пользователям указывать не все атрибуты данных, функций и операторов, хотя со временем некоторые специалисты стали рассматривать эту особенность как слабость языка.

Другой особенностью языка было то, что пользователь мог работать только с теми его средствами, которые нужны для решения задачи. При этом пользователь может и не знать все средства языка.

ПЛ/1 стал первым универсальным языком программирования (точнее, первый, претендующий на такое название). Как и всякий новый язык, он не избежал болезней становления. Хотя первый компилятор с языка был создан в 1965 г., до 1967 г. не было «приличной» версии компилятора, которую можно было бы широко использовать. И только в начале 1968 г., когда сотрудниками фирмы IBM был разработан улучшенный вариант компилятора, стали очевидны достоинства нового язы-

той). В данном случае машина вступает с вами в настоящий диалог. Она приостанавливает выполнение программы, выводит на экран вежливое сообщение: ЖДУ ЧИСЛА И СЛОВА, — а затем выдает уже привычный вам вопросительный знак. Подсказку сочиняете вы сами, но когда ее напечатает на экране компьютер — все равно приятно. Будто с хорошим собеседником поговорили!

Отвечать на компьютерную подсказку нужно в том же синтаксисе, какой использован при записи команды INPUT: отделяя элементы ответа друг от друга запятыми и не заключая символьные величины в кавычки. Весь ответ целиком завершается, как обычно, нажатием клавиши ВК. Если вы где-то ошиблись — ввели, например, слово вместо числа и т. п., — компьютер попросит повторить все сначала:

```
? Redo from start
```

Если вы ввели недостаточное количество данных, — скажем, напечатали только число и нажали ВК, — машина выведет на экран двойной знак вопроса (??), намекая вам, что требуется еще нечто, названное в подсказке. Выполните это ее указание — и программа начнет работать дальше.

Выйти из режима команды INPUT можно, если одновременно нажать клавиши CTRL и C (латинское) или CTRL и STOP. После этого компьютер вообще прекращает исполнение прерванной программы и ждет ваших новых указаний.

ON... GOTO...
ONXGOTO 150, 210, 25
ON... GOSUB...
ON 2*X+5 GOSUB 1000, 2000

Это полезная команда-переключатель. Когда компьютер встречает ее в программе, он вычисляет значение выражения, записанного после слова ON (отбрасывая дробную часть, если получается нецелое число), и переходит к выполнению командных строк, перечи-

ЯЗЫКИ программирования

ка. Первоначально планы реализации ПЛ/1 на компьютерах семейств IBM/360 предусматривали создание семейства компиляторов, но их появилось только два: ПЛ/1 F и ПЛ/1 D. В дальнейшем компиляторы с ПЛ/1 были разработаны для ряда вычислительных систем, выпускаемых фирмами Burroughs, Control Data, Honeywell и др. В нашей стране язык ПЛ/1 был успешно реализован на вычислительных машинах серии ЕС ЭВМ.

С марта 1968 г. и до середины 70-х годов фирма IBM опубликовала ряд документов по усовершенствованию языка ПЛ/1 с целью его стандартизации (версии 0,1,2 и ООК).

В то же время в процесс развития языка включились Комитеты ANSI и ESMA (Европейская ассоциация изготовителей машин), опубликовавшие в феврале 1975 г. проект стандартного языка ПЛ/1, который был принят в октябре 1976 г.

Наряду с полуформальным определением языка, осуществленным сотрудниками IBM в Херсли, основательную работу по формальному его определению проделали сотрудники венских лабораторий той же фирмы. С помощью предложенного ими метода формального определения языков программирования впервые был описан такой сложный язык высокого уровня, как ПЛ/1. В основе метода ВЯО (Венского Языка Определений) лежат работы Маккарти, Элгота, Лэндина. Первая версия формального определения языка появилась в 1966 г., вторая — в 1968.

После своего «выхода в свет» ПЛ/1 был определен некоторыми специалистами как важнейший язык 70-х годов. Однако вскоре стало очевидным, что он не в состоянии «бороться» с КОБОЛом и ФОРТРАНом, а затем, в конце 70-х годов, уступил место БЕЙСИКу и ПАСКАЛЮ.

Одна из причин неоправданных надежд на широкое применение ПЛ/1, как отмечает Г. Хеллс, объясняется его сложностью (многие реализации языка содержат более 200 ключевых слов). Сложность привела к низкой читабельности и затруднила автоматическую проверку программ. Другая причина: язык ПЛ/1 появился после того, как ФОРТРАН, КОБОЛ, АЛГОЛ тоже повсеместно были признаны, и многие пользователи, программирующие на этих языках, не нуждались в оригинальных возможностях ПЛ/1.

И все же всплеск активности ПЛ/1 был отмечен в годы широкого использования компьютеров IBM/360 и IBM/370, а также совместимых серий машин, созданных в других странах. Кроме того, история языка ПЛ/1 получила свое продолжение.

В 1966 г. для существовавшего языка для обработки формул ФОРМАК (FORMAC) на основе ПЛ/1 была выпущена версия под названием ПЛ/1-ФОР-

сленных после GOTO по такому правилу: при значении выражения, равном 1, — к строке номер 150, при значении, равном 2, — к строке номер 210, а при значении, равном 3, — к строке номер 25. Если значение выражения оказалось нулевым или превысило число вариантов выбора (в данном примере их три), машина просто переходит к следующей исполнимой командной строке программы. Когда оцениваемое выражение имеет запрещенное значение (отрицательное или превосходящее 255), компьютер прерывает выполнение программы и на экране увидим сообщение об ошибке.

После переключения на одну из выбранных командных строк исполнение программы пойдет от нее «вниз». Чтобы заставить компьютер вернуться к следующей за командой ON... GOTO... строке программы, используют вариант ON... GOSUB ... Здесь переключатель отсылает машину к первым строкам той или иной подпрограммы, по исполнении которой происходит автоматический возврат. Разумеется, подпрограммой можно назвать и единственную командную строку, если она завершается признаком подпрограммы (командой RETURN).

PRINT

С этой распространенной командой вы уже более или менее знакомы. Напомним, что все, заключенное в кавычки и стоящее после слова PRINT, компьютер воспроизводит на экране без малейших изменений, а вместо имен числовых или символьных переменных печатает их значения, имеющие место в тот момент, когда в программе встретилась команда PRINT.

Кое-что об особенностях пунктуации в этой команде. Если в списке печатаемых командой переменных разделить их имена точкой с запятой (пример: PRINT A$; B$; C$), компьютер соответствующие слова напечатает слитно. Скажем, когда A$=«ЭТОТ», B$=«ВАРИАНТ», C$=«НЕУДОБЕН», на экране получится вот что:

ПЛ-1

МАК, которая не без успеха использовалась на ЭВМ третьего поколения до начала 80-х годов.

В начале 70-х годов появились первые микропроцессоры. В 1972 г. компанией МАА (Microcomputers Applications Associates) был разработан язык ПЛ/М (PL/M — Programming Language for Microcomputers — язык программирования для микрокомпьютеров) как язык системного программирования для 8-разрядных микропроцессоров фирмы Intel. ПЛ/М берет свое начало от языка написания компилятора ХПЛ, который, в свою очередь, происходит от полного языка ПЛ/1. Консультант фирмы Г. Килдалл написал первый компилятор с языка ПЛ/М. В 1974 г. он же разработал первую версию операционной системы CP/M (Control Programm for Microcomputers — управляющая программа для микрокомпьютеров), на которую опирался резидентный компилятор ПЛ/М. Язык ПЛ/М стал инструментальным языком фирмы Intel и сыграл большую роль в отказе от применения языков ассемблерного типа. В дальнейшем многие ведущие фирмы — производители микропроцессорной техники — создали свои варианты языка ПЛ/М. К настоящему времени разработано несколько таких версий: ПЛ/3, ПЛ/65, ПЛ/М-80, ПЛ/М-86, ПЛ/М-286. Некоторые из этих версий реализованы на отечественных микро-ЭВМ семейства СМ1800.

Литература

Скотт Р., Сондак Н. ПЛ/1 для программистов / Пер. с англ. — М.: Статистика, 1977.

Корнева Л. А. Эволюция языка ПЛ/1 // Прикладная информатика. — 1982. — Вып. 1.

Малютин Э. А., Малютина Л. В. ПЛ/1 для начинающих. — М.: Финансы и статистика, 1985.

Хелмс Г. Л. Языки программирования / Пер. с англ. — М.: Радио и связь, 1985.

Компьютеры: Справочное руководство / Пер. с англ. — В 3 т. — Т. 2. — Л.: Мир, 1986.

Фролов Г. Д., Олюнин В. Ю. Практический курс программирования на языке PL/1. — М.: Наука, 1986.

Дейтел Г. Введение в операционные системы / Пер. с англ. — В 2 т. — М.: Мир, 1987.

ЭТОТ ВАРИАНТ НЕУДОБЕН.

Действительно, читать не совсем удобно. Чтобы исправить положение, нужно запрограммировать пробелы между словами: PRINT A$; « »; B$; « »; C$. Тогда компьютер отпечатает результат более красиво:

ЭТОТ ВАРИАНТ НЕУДОБЕН.

При печати чисел после каждого добавляется пробел, а положительные числа будут иметь еще и дополнительный пробел спереди (вместо не обязательного знака плюс), так что здесь специально заботиться о введении в программу пробелов не требуется.

Если разделить имена печатаемых командой PRINT переменных не знаками «точка с запятой», а просто запятыми, компьютер ведет себя иначе. Он «мысленно» разбивает поле экрана на колонки шириной по 14 символов, и значение каждой переменной из разделенного запятыми списка печатает в начале новой колонки, создавая что-то вроде простенькой таблицы. То есть выполнение команды PRINT A$, B$, C$ с прежними значениями этих символьных переменных теперь даст такой результат:

```
ЭТОТ          ВАРИАНТ       НЕУДОБЕН
<-------------><------------><------------
14 символов   14 символов   14 символов
```

Когда команды PRINT расположены в соседних командных строках, не завершенных никакими знаками, или разделены двоеточиями, компьютер печатает результаты, каждый раз начиная с новой экранной строчки. Часть программы:

```
30 PRINT A$
40 PRINT B$
50 PRINT
60 PRINT C$
```

даст результат:

```
ЭТОТ
ВАРИ-
АНТ
НЕУДО-
БЕН
```

Отметьте появление пустой экранной строки, если в операторе PRINT вообще нет никаких имен переменных. При желании напечатать то же самое в одну строчку командные строки 30 и 40 следовало завершить знаком «точка с запятой», строку 50 выбросить из программы, а также позаботиться о пробелах между словами:

```
30 ? A$; « »;
40 ? B$; « »;
60 ? C$
```

Результат: ЭТОТ ВАРИАНТ НЕУДОБЕН

Здесь мы еще раз напомнили вам о том, что слово PRINT для краткости можно заменять в этих случаях знаком вопроса (?).

READ

Мы уже упоминали эту команду, говоря об операторе DATA. Они всегда должны существовать в программе совместно: под обозначением DATA хранится набор нужных для выполнения программы данных, а по команде READ компьютер начинает пользоваться этими данными. После слова READ приводится разделяемый запятыми список переменных, значения которых будут взяты из «банка» DATA. Чтобы не остановить выполнение программы из-за ошибки, нужно внимательно следить за тем, согласуются ли типы данных в DATA с соответствующими именами переменных в команде READ. (Скажем, если первой считываемой переменной является X — обозначение числовой величины, — а в списке DATA первым идет какое-то слово — символьная константа, — ЭВМ обнаружит ошибку и прервет исполнение программы.)

Порядок пользования «банком данных» DATA таков. Встретившийся в ходе выполнения программы оператор READ присваивает своей первой переменной первое значение, содержащееся в списке констант DATA. Затем то же самое повторяется для второй (замените всюду в только что приведенной фразе «первый» на «второй»), третьей переменной и т. д. Поскольку программист может так составить программу, что в ней окажется несколько операторов DATA и несколько READ, для легкости понимания процедуры представьте себе, что все строки с DATA объединены и их данные следуют друг за другом без перерывов. Подобным же образом объединены все операторы READ, и перечисленные в них переменные сведены в единый список с сохранением порядка следования, но без посторонних включений. Тогда процедура, выполняемая компьютером по команде READ, становится совершенно понятной.

Если в списках DATA почему-либо окажется больше констант, чем того требуют переменные из перечня READ, лишние данные попросту не используются и на исполнение программы это отрицательно не влияет. Если же, наоборот, какие-то переменные не могут найти соответствующих значений (данные перебраны все, а список обслуживаемых оператором READ переменных еще не исчерпан), компьютер расценит это как явный огрех программиста, прервет выполнение программы и напечатает сообщение об ошибке.

REM

Вероятно, самая простая из команд. Обозначает, что все следующие за ней символы в пределах данной командной строки не являются частью программы, не требуют анализа и исполнения, поскольку это лишь вспомогательный текст — заметки, написанные программистом для себя самого или своих коллег, которые станут когда-то разбираться в этой программе. Хотя комментарий, обозначенный как REM, при выполнении программы не учитывается, в распечатке текста программы по команде LIST или LLIST он будет сохранен (иначе какой была бы его ценность, если заметки нельзя потом прочесть?).

Командная строка с комментарием, как и все строки программы, снабжается своим номером, хотя при выполнении программы компьютер, конечно, ее пропускает.

Есть и другой вариант оформления комментария: вместо начинающего строку слова REM можно использовать просто одиночную кавычку — апостроф ('). Она же позволяет и дописать комментарий в конце иной программной строки, причем двоеточия, обычно разделяющего различные операторы, здесь не требуется.

RESTORE RESTORE 125

Это команда используется в процессе считывания данных (см. операторы DATA и READ). Как было объяснено выше, передвигаясь по списку переменных оператора READ, компьютер считывает данные из «банков» последовательно, возврата назад нет. Однако иногда такой возврат может показаться программисту нужным. В этих случаях применяется команда RESTORE. Она возвращает внимание машины вновь к началу списка констант, и следующий за RESTORE оператор READ станет присваивать переменным значения, которые уже были однажды использованы (вначале первое, потом второе и т. д.). Такой прием бывает нужен для того, чтобы можно было неоднократно пользоваться каким-то полезным списком данных, не печатая его в программе несколько раз. Более того, если «банков данных» DATA в программе несколько — а вы можете разбить общий список констант произвольным образом на части, обозначить каждую часть, начинающуюся оператором DATA, как отдельную командную строку со своим собственным номером, — оператор RESTORE способен возвращать процесс считывания данных к любому месту (не только к началу). Для этого вслед за словом RESTORE печатается номер строки программы, куда следует вернуться. Разумеется, эта строка должна начинаться командой DATA (впрочем, если такое условие и будет случайно нарушено, компьютер начнет поиск «вниз» по программе и примет за точку отсчета первый встретившийся ему оператор DATA). Однако если вы по оплошности укажете номер строки, которого вообще нет в вашей программе, компьютер прекратит ее выполнение и выдаст сообщение об ошибке.

STOP

Команда прекращения выполнения программы. Ее помещают в любом месте программы, и машина при этом печатает такое сообщение: Break in ... (следует номер командной строки, где встретилось слово STOP). Работу программы можно возобновить, введя команду CONT.

«Проба пера» (чему же мы научились?)

Вы, вероятно, уже несколько утомились от чтения этих казуистических законов из жизни компьютеров. Хотя мы и выбирали, кажется, самое необходимое и не слишком сложное, непривычному человеку трудно сразу понять, что все

это вещи очень полезные и что на самом-то деле он сам уже может составить простенькую компьютерную программу. Не верите? Давайте попробуем.

Но прежде чем начать печатать на экране текст нашей самой первой программы, нужно кое-что вспомнить, а что-то узнать заново о возможностях компьютера как пишущей машинки. Без этого двигаться дальше нельзя: нажали клавишу не с той буквой — а как исправить опечатку, не знаем...

Экранный редактор. Сейчас мы кратко опишем возможности того, что называется экранным редактором языка БЕЙСИК-MSX. Экранный — поскольку работает в пределах всего экрана, становящегося «рабочей страницей» вашего компьютерного блокнота. Редактор — потому что выполняет задачи редактирования текста (исправление опечаток, переделка слов, «фраз» и т. п.). Конечно, истинный редактор — это вы, вы ведь редактируете текст программы, а компьютер лишь дает вам для этого удобные средства.

О клавишах управления курсором (четыре крупные клавиши со стрелками) вы уже знаете. Нажимая их, можно поместить курсор в любое место экрана — как бы двигая каретку и крутя резиновый вал обычной пишущей машинки, перемещающий лист бумаги вверх-вниз. Клавиша BS — «шаг назад» — позволяет вернуться к только что напечатанному символу и поправить сделанную опечатку (тоже как на пишущей машинке). Что же делать, если опечатку заметили не сразу? Клавишами управления курсором следует наложить его на неверно отпечатанную букву или иной знак и «перебить», нажав нужную клавишу с буквой. Сделав так, вы заметите, что как бы печатаете поверх уже имеющегося текста, причем старый вариант начисто стирается курсором. Так можно вносить поправки. Правда, тут есть хитрость: машина «запоминает» то, что напечатано на экране, не сразу — лишь после нажатия клавиши ВК. Иногда исправлений в командной строке оказывается столь много, что легче отпечатать всю строку вновь. Поставьте в начале новой строки старый номер и напечатайте поправленный текст (не забывайте сделать в заключение главное — ВК). Компьютер немедленно «забудет» прежний вариант строки программы, заменив его новым (пока лишь «в уме»).

Для быстроты и удобства редактирования текста программы предусмотрен ряд специальных служебных клавиш. Клавиша, помеченная как CLS/HOME, на верхнем регистре (SHIFT или CAPS), очищает экран от всех записей, а также помещает курсор в левый верхний угол экрана. Нажатие той же клавиши на нижнем регистре перебрасывает курсор в левый верхний угол экрана — на исходную позицию (команда HOME — вроде приказа собаке: «Место!»).

Клавиша INS включает режим вставки пропущенных символов, слов, целых «фраз». Когда вы нажмете ее, курсор уменьшится, чтобы напоминать вам, что включен этот режим, и каждый вновь отпечатанный вами символ станет появляться в месте расположения курсора, тогда как все прежние символы, находящиеся справа от него, будут сдвигаться вместе с курсором, освобождая место для новых. Выключить режим вставки можно повторно нажав клавишу INS. Курсор при этом примет прежние размеры и вновь начнет стирать старые знаки во время печатания по ним новых.

Клавиша DEL выполняет в некотором смысле обратную задачу: нажимая ее, вы увидите, что часть программной строки, расположенная справа от курсора, начинает сдвигаться влево, причем каждый раз ее начальный символ как бы «съедается» курсором. Нажав эту клавишу много раз, можно уничтожить в такой «черной дыре» всю правую часть командной строки.

Чтобы мгновенно уничтожить часть командной строки, находящуюся справа от курсора, следует одновременно нажать клавиши CTRL и E (латинское). Чтобы уничтожить всю такую строку целиком — нажать одновременно клавиши CTRL и U (курсор может находиться в любом месте строки). Командную строку можно стереть из памяти машины и иначе: отпечатайте ее номер и нажмите ВК. Компьютер заменит старую строку новой, но новой нет — она пуста.

Если нужно дополнить строку программы новыми символами, а вести курсор к нужной точке экрана шагами слишком долго, нажмите одновременно CTRL и N — курсор мгновенно окажется в конце командной строки и можно печатать ее продолжение.

Есть и некоторые иные редакторские функции, однако на первое время вам хватит перечисленных, а по мере большего знакомства с компьютером вы постепенно освоите и другие. Любую задачу редактирования текста программы можно решить, как правило, несколькими способами, используя различные средства экранного редактора и различные их сочетания. Каждый программист со временем вырабатывает некий собственный стиль, свои приемы, представляющиеся ему наиболее естественными и удобными. Так что пробуйте, запоминайте, выбирайте!

Составление программы. Давайте в качестве «пробы пера» попытаемся составить несложную программу диалога компьютера с его хозяином, капризного такого диалога — пусть наш компьютерный собеседник будет привередливым, придирающимся к мелочам и неточностям ответов человека. На этом примере мы опробуем в деле тот скромный арсенал программных средств, который был предварительно описан выше. Постараемся — пусть даже в ущерб краткости и быстроте исполнения нашей программы, зато в пользу ее понятности читателю и из соображений демонстрации как можно большего числа известных нам команд — в этой простой программе показать, что такое программирование для компьютера. Надеемся, что вы заразитесь страстью писать свои собственные программы, заставляющие машину делать все нужное вам — от длинных расчетов до компьютерных игр. По ходу дела мы будем при необходимости вводить не упоминавшиеся ранее команды или приемы программирования, коротко характеризуя суть понадобившихся новинок. Это еще раз напомнит вам, что из сказанного выше вы узнали далеко не все даже о простейшем языке программирования БЕЙСИК и что чтение более детальных описаний, а главное, постоянная самостоятельная работа на компьютере должны стать вашими обычными занятиями. Это как изучение человеческого языка: предела познанию его тонкостей нет, и каждый выбирает для себя тот конечный уровень, который отвечает практическим потребностям.

Мы станем вместе с вами составлять компьютерную программу, комментируя свои действия и объясняя ожидаемые результаты. Иллюстрации — командные строки — мы будем писать в том виде, в каком они появятся на экране вашего компьютера, и вы, набирая текст программы с клавиатуры, постарайтесь в точности копировать наши примеры (любая оплошность, вроде забытой или поставленной не на своем месте ничтожной запятой, способна напрочь испортить программу).

Если не приказано иное, компьютер «Ямаха MSX-2» после включения устанавливает для себя такой экранный режим: 24 строки по 40 символов в каждой (последняя 24-я строка содержит список обозначений функциональных клавиш, которыми мы пока не пользуемся). Поэтому и мы выберем длину нашей «экранной» строки ровно в 40 символов, чтобы вам легче было выделить текст программы из текста пояснений, а также чтобы облегчить перепечатку программы и сравнение с помещенным здесь «эталонным» вариантом.

Итак, мы собираемся составить программу, которая во время своей работы будет имитировать несложный диалог машины и человека. Компьютер будет задавать своему хозяину вопросы (разумеется, письменно — говорить он пока не обучен), принимать письменные же ответы, «обдумывать» их, капризно придираться к вам, кокетничать, переспрашивать и т. д. А вы в ответ на его озорство должны терпеливо выполнять все, что он просит, иногда намеренно нарушая правила, чтобы посмотреть на реакцию машины.

С чего же начать? Прежде всего нужно очистить экран от следов вашей предыдущей работы — всегда приятнее творить на чистом листе, чем на уже исписанной странице. Нажмите клавишу SHIFT и, не отпуская ее, клавишу CLS/HOME. Экран чист, курсор занял исходную позицию в левом верхнем углу

экрана. Экран вы подготовили, однако в памяти машины могла остаться предыдущая программа, что сильно попортит нам все дело. Чтобы очистить память, воспользуемся командой NEW (и непременно — ВК). Вот теперь компьютер действительно готов к работе над программой. Можно еще включить режим автоматической нумерации строк, но программа наша ожидается небольшой, так что не составит труда перенумеровать строки вручную. Это в данном случае даже удобнее: номера будут у нас нести определенную смысловую нагрузку, как вы увидите, и идти поэтому группами, так что в режиме AUTO пришлось бы часто менять параметр команды (начальный номер строки).

Начинаем программу, как водится, с заголовка:

```
10 REM Программа «Капризный собес
едник»
20 'Будет откликаться на имя: Иван
Петрович Сидоров
```

Были показаны сразу два варианта оформления комментариев к программе — тех сообщений, которые нужны программисту, но не компьютеру. Еще раз напомним: каждая командная строка программы завершается нажатием клавиши ВК, а переход с одной строки экрана на другую машина делает автоматически (никаких знаков переноса использовать нельзя!). При этом фразы или слова могут разрываться не совсем удобным для чтения образом (как разорвано слово «собеседник» в строке 10). Избежать этого можно, лишь перенося все слово целиком на следующую строчку экрана. Это делается клавишей пробелов, но не клавишами управления курсором и ни в коем случае не нажатием ВК (компьютер поймет это как окончание строки 10).

```
30 CLS:LOCATE 0,12:BEEP
```

Это первая исполняемая компьютером командная строка. Оператор CLS очищает экран (на всякий случай, ведь мы не знаем, что там было напечатано прежде). Новая для нас команда LOCATE помещает курсор в 13-сверху строку (нумерация строк начинается с нуля) — примерно посередине экрана, у самого левого его края (1-й символ строки нумеруется нулем, 2-й — единицей и т. д. до 39). Каманда BEEP (звуковой сигнал «Бип!») станет предшествовать в нашей программе каждому обращению компьютера к хозяину (компьютерный вариант уличного «Эй, послушайте!»). Заметьте, что операторы внутри командной строки, как уже говорилось, разделяются двоеточиями и что в конце строки двоеточия нет.

```
40 INPUT «Здравствуйте, здравствуйте!
Только я с незнакомцами не разговари-
ваю. Как вас зовут»; NAME$
```

Вот первая реплика компьютера в нашей микропьесе. Знака вопроса, требуемого русской грамматикой, в конце реплики нет потому, что оператор INPUT добавляет этот знак от себя. Если б мы тоже поставили его, получилось бы два: ?? — а это неграмотно. Задав (написав) вопрос, машина ждет вашего ответа. Вы должны теперь напечатать что-то на экране и ввести ответ клавишей ВК (не забывайте про ВК!). Машина обозначит его как символьную переменную NAME$. Язык БЕЙСИК требует использования в именах переменных латинских букв, вот почему мы выбрали английское слово, смысл которого напоминает нам, что это чье-то имя.

Чтобы диалог был естественным и забавным, компьютер должен проанализировать ваш ответ и начать с вами ворчливую перебранку, добиваясь своей цели. А его цель — узнать, что хозяина зовут Иван Петрович Сидоров, успокоиться и программу закончить. Для простоты предположим, что машина анализирует ответ всего лишь по одному первому символу. Мы увидим, что и в таком случае возникает немало вариантов, освоив которые можно програмировать и более сложные ситуации. Но прежде «обслужим» программой прямое попадание в цель:

```
50 IF NAME$=«Иван Петрович Сидоров»
TH
EN GOTO 500
500 BEEP:BEEP:BEEP
510 CLS:LOCATE 15,12:? «Уррра-а-а!»
520 END
```

То есть, если вы сразу напечатали искомое имя, компьютер переходит к ко-

мандной строке 500 (номер выбран с запасом, чтобы до него уместилось вполне достаточно еще не придуманных нами строк), троекратным писком салютует законному хозяину, очищает экран и в середине его пишет радостное «Уррра-а-а!», после чего по команде END заканчивает выполнение программы. Строка проверки условия IF ...THEN не содержит упоминавшейся ранее при описании этого оператора части ELSE. А это значит, что, когда условие не выполняется, машина просто переходит к следующей командной строке. В этой строке, номер ее будет 60, программа начинает самое интересное — изучение неправильного ответа и соответствующее реагирование на различные варианты. А ответ, без сомнения, будет в этот момент неправильным, ведь при правильном ответе компьютер просто не достигнет строки 60: после 50-й он немедленно перескочит на 500-ю, просалютует, как предписано, исполнит команды строки 510 и в соответствии со строкой 520 завершит на этом программу.

Мы договорились, что машина станет анализировать ответ по одному символу. Вначале нужно его отделить:

```
60 Z$=LEFT$ (NAME$, 1): Z=ASC (Z$)
```

Первый оператор здесь «отрезает» от введенной вами цепочки самый левый символ*, а второй оператор определяет номер этого символа в таблице кодов ASCII. Зачем это понадобилось? Вот зачем: компьютеру все 256 символов таблицы ASCII одинаково милы, хотя для человека они неравнозначны. В соответствии со своими пристрастиями и представлениями о родстве и смысле символов люди сгруппировали их в таблице. Отдельной группой идут цифры, отдельной латинские буквы и отдельной русские, большие и малые. Этой-то систематизацией мы теперь и воспользуемся, чтобы представить в нашей программе компьютер «разумным». (Кстати, пусть вас не смущает, что 60-ю строку программы мы напечатали после 500-й: потом машина расставит их как надо.)

Поскольку компьютер уже знает код первого символа ответа, нужно научить машину «понимать» его смысл. Если заглянуть в таблицу кодов ASCII, применяющуюся в компьютере «Ямаха MSX-2», то мы увидим, что цифры там имеют коды от 48 до 57, большие латинские буквы — от 65 до 90, малые латинские — от 97 до 122, а русские буквы — от 192 до 223 (малые) и от 224 до 254 (большие). Значит, в зависимости от того, в какой диапазон угодила наша числовая переменная Z, можно конструировать вид следующей реплики «умной» машины. При этом, чтобы создать впечатление компьютера, глубоко вникающего в суть слов, следует предусмотреть и тот случай, когда отвечающий машине человек (который, конечно же, не столь разумен) начинает ответ с какого-то совершенно абсурдного символа — запятой или знака параграфа.

Сделаем главное: сконструируем анализатор «смысла» ответа (простейший — по первому символу):

```
70 IF Z>=48 AND Z<=57 THEN X=1:GOT
O 1000' Цифра
80 IF Z >=65 AND Z <=90 OR Z>=97 AND
Z<=122 THEN X=2:GOTO 1000' Лат. буква.
90 IF Z>=192 AND Z<=223 THEN X=3: GOTO 1000' Русская малая
100 IF Z>=224 AND Z<=254 THEN X=4: GOTO 1000' Русская большая
110 X=5: GOTO 1000' Абсурдный знак
```

Анализатор работает так. Строка 70 заставляет машину проверить, попадает ли Z в интервал 48-57 (коды цифр). Логический оператор AND (И) означает одновременное выполнение записанных по обе стороны от него условий. То есть если одновременно Z>= 48 и Z<=57, иначе говоря, если 48<=Z<= <=57, следует перейти к строке 1000, имея при себе новую переменную X, которой присвоено значение 1. В строке под номером 1000 (она нами еще не придумана) будет стоять «переключатель», направляющий дальнейшие действия компьютера соответственно значению переменной переключения X.

---

* Строго говоря, оператор LEFTS в этой строке излишен, поскольку команда ASC сама отбирает лишь первый символ. Однако мы решили оставить два оператора, чтобы структура программы была более демонстрационна.

Совершенно аналогично действуют и остальные командные строки (80—100). В конструкции IF ... THEN здесь нет части ELSE, поэтому при невыполнении поставленного условия компьютер перейдет не в направлении, указанном оператором ELSE, а просто к следующей строке программы. Смысл действий, выполняемых в каждой строке, указан комментарием, кратко напоминающим программисту, что именно ищется на этом этапе.

Логические операции строки 80 посложнее: проверяется, не попадает ли Z в один из двух интервалов (65 — 90, 97 — 122). Эту задачу решает оператор OR (ИЛИ). Чтобы не ошибиться при записи этой строки, нам пришлось заглянуть в справочники, где мы выяснили, что в соответствии с принятым порядком выполнения логических операций вначале исполняется AND, а потом OR. Так что наша запись верна. (В противном случае результат выполнения такой цепочки операций мог быть совсем иным.)

Теперь о строке 110. Если компьютер дошел до нее, это означает, что Z не попадает ни в один из названных нами числовых интервалов, т. е. первый символ ответа человека и не цифра, и не латинская буква, и не русская (какой-то иной символ, вроде §, названный в комментарии абсурдным). Проверять конструкцией IF ... THEN больше нечего, поэтому переменной переключения X сразу присваивается значение 5 и машине предписывается немедленно перейти к строке 1000.

```
1000 ?:?:BEEP:ON X GOSUB 1100, 1200
,1300, 1400, 1500
```

Это наш переключатель. Соответственно значению X он направляет компьютер к одной из пяти подпрограмм, описывающих реакцию машины на пять предусмотренных нами вариантов ответа. Два следующих подряд оператора PRINT (в сокращенном обозначении) дают нам просто пропуск двух экранных строк, чтобы отделить дальнейшие сообщения от предыдущих. BEEP — привычный уже нам сигнал предупреждения: „Внимание, буду говорить!“. По смыслу его следовало бы поместить непосредственно перед началом печатания компьютерного ответа, но подпрограмм у нас ожидается целых пять, значит, команду BEEP пришлось бы повторять тоже 5 раз, поэтому мы сочли более удобным и экономным «вынести ее за скобку». Компьютер работает очень быстро, так что заметного разрыва между сигналом «Бип!» и началом печатания на экране ответа машины вы не ощутите, хотя за это микроскопическое время компьютер успеет исполнить много команд.

Пришло время записать подпрограммы, к которым машина станет переходить под управлением переменной X:

```
1100' Реакция на цифру
1110? «Что это у вас имя какое стран
ное — начинается цифрой? Или вы
тоже компьютер? Не хочу говорить с
компьютерами. Прощайте!»: RETURN
1200' Реакция на латинскую букву
1210?«Зачем вы пишете не по-русски?
Вы иностранец? Я с иностранцами не
знакомлюсь. Прощайте!» RETURN
1300' Реакция на русскую малую букву
1310?«Почему ваше имя пишется с ма
лой буквы? Себя не уважаете, что ли?
Не могу говорить с вами. Прощайте!»:
RETURN
1400' Реакция на русскую большую
букву
1410?«Простите, как вы сказали?»:?
NAME$; «?»:?«О, какое прекрасное
имя! К сожалению, вы — не тот, кто мне
нужен. Извините. До свидания.»: RE
TURN
1500' Реакция на абсурдный знак
1510?«Нет, это просто издевательство!
Таких имен вообще не бывает! Прощай
те!»: RETURN
```

В подпрограмме, начинающейся строкой 1400, где компьютер отменно любезен, есть особенности. Во-первых, каждая из трех частей реплики будет начинаться с новой строки, во-вторых, использована маленькая хитрость. Машина, будто не расслышав, повторяет сообщенное вами имя — печатается переменная NAME$, — присовокупив к нему знак вопроса.

Исполнив одну из подпрограмм, компьютер по завершающей каждую подпрограмму команде RETURN возвращается к строке, следующей за командной строкой 1000 («переключателем»).

Поскольку там у нас еще ничего нет, машина «зависнет» — замрет, перестав реагировать на любые ваши запросы с клавиатуры. Давайте выведем ее из такого тупика:

```
1010 G$ = INKEY$
1020 IF G$ = « » THEN 1010 ELSE 30
```

Здесь мы использовали незнакомый вам оператор INKEY$. Он заставляет машину принять с клавиатуры один символ. Если никаких клавиш не нажимали, считается, что принят «пустой» символ (INKEY$=«»). Итак, вместо попадания в безысходный тупик компьютер в соответствии с командной строкой 1010 проверяет, не была ли нажата какая-либо клавиша после его реплики. Строка 1020 заставляет эту процедуру проверки повторяться до тех пор, пока вы, устав от ожидания, не коснетесь клавиатуры. Что бы вы ни нажали — машина немедленно перейдет к программной строке 30 и начнет исполнение всей программы с самого начала: очистит экран, пискнет, напишет на 13-й строчке «Здравствуйте...» и т. д. по уже описанному образцу. Таким образом, компьютер не прекратит вас спрашивать и пререкаться с вами до тех пор, пока вы не введете имя «Иван Петрович Сидоров». Только тогда программа завершится.

Можно сделать поведение машины более натуральным, ограничив ее терпение, скажем, пятью безуспешными попытками. Для этого в начале программы нам придется установить «счетчик» — оператор цикла, а в строке 1020 возвращать компьютер уже не к прежней, а к этой новой начальной строке программы:

```
25 FOR N=1 TO 5
1020 IF G$=«» THEN 1010 ELSE NEXT N
```

Новая строка 25 встанет перед строкой 30 (теперь вы понимаете, зачем мы нумеруем строки программы столь «неплотно»?), а прежняя строка 1020 заменится ее новым, исправленным вариантом. При такой записи программы компьютер переспросит вас ровно 5 раз и вновь «зависнет», не найдя за строкой 1020, куда бы ему перейти, когда значение переменной цикла N стало равным 6. Сконструируем еще одно аккуратное завершение программы:

```
1030 CLS:LOCATE 0,12:BEEP:? «Я устал. Пять раз спрашивал вас, и пять раз вы отвечали невпопад. Больше говорить с вами не хочу.»
1040 END
```

Последняя реплика будет напечатана на трех строчках очищенного экрана: с 13-й по 15-ю (LOCATE 0,12 переводит курсор на начало 13-й экранной строки.

Машина выполняет все команды по человеческим меркам очень быстро. Настолько быстро, что диалог приобретает излишне стремительный темп. Чтобы добавить в характер машины больше привычных нам качеств собеседника, можно запрограммировать «задумчивость» компьютера. Пусть, например, после того как в ответ на первый вопрос вы ввели неверное имя, компьютер несколько секунд над ним «размышляет». Вставим для этого между строками 50 и 60 «задержку»:

```
55 FOR R=1 TO 10000
56 NEXT R
```

Компьютер будет теперь метаться между строками 55 и 56 ровно 10000 раз, и на это уйдет несколько секунд. Вы можете подобрать время «раздумий» по своему вкусу: попробуйте поставить вместо 10000 другое число — 5000, или 15000, или какое-нибудь еще. Можно смоделировать тут же и раздражительность собеседника (кто же любит, когда к нему пристают во время раздумий!):

```
55 FOR R=1 TO 20000
56  G$=INKEY$
57 IF G$ THEN BEEP:? «Не смейте мне мешать! Видите — думаю!»
58 NEXT R
59  CLS:LOCATE 0,12
```

Здесь наш компьютер стал тугодумом: цикл FOR ...NEXT должен повторяться целых 20000 раз. Если вам надоело ждать результата и вы нажали какую-то из клавиш, машина довольно грубо огрызнется и продолжит раздумья. В заключение, чтобы очистить экран, изрядно замусоренный репликами компьютера, применена командная строка

59, готовящая «чистый лист» для последующих ответов машины.

Несколько необычна на этом участке программы строка 57. Оператор IF G$ — это сокращенный вариант полной записи: IF G$> «», т. е. «если с клавиатуры поступил не пустой символ». Остальное же, надеемся, стало вам уже привычным.

Вы теперь можете вполне самостоятельно в меру своей фантазии придумать и отразить в программе «Капризный собеседник» любые приглянувшиеся вам качества «говорящего» робота. Суть вам должна быть ясна — дерзайте.

По окончании черновой работы — когда экран монитора усеян остатками неверно напечатанных строк программы, строки еще идут не в надлежащем порядке и т. п. — наберите команду LIST и нажмите клавишу ВК. Компьютер тут же приведет текст программы в окончательный красивый вид. Теперь, чтобы испытать программу в работе, достаточно послать команду RUN.

Вам нужно понимать, что практическое программирование — род искусства, поэтому никаких жестких правил составления программ (кроме, разумеется, правил синтаксиса используемого вами компьютерного языка) нет и не может быть. Программу, выполняющую те же задачи, которые мы поставили перед собой, конструируя нашего «капризного собеседника», можно написать и по-другому. Не исключено, что иной вариант получится даже лучше — компактнее, изящнее. Скажем, частое использование оператора GOTO теоретики программирования считают дурным тоном, утверждая, что без него можно обойтись в любом случае и что чем чаще он используется программистом, тем программист хуже. Взглянув на строки 70—110 нашей программы, мы сами устыдились обилия там этих самых GOTO. Впрочем, главная наша цель — показать принципы составления программ и использовать в своем примере как можно больше из упоминавшихся нами операторов и приемов практического программирования — все-таки была, как нам кажется, достигнута. А далее — все в ваших руках. Практикуйтесь, работайте, совершенствуйте стиль методом проб и ошибок. Постарайтесь, однако, запомнить главное: чем стройнее будет выглядеть ваша программа, чем меньше будет в ней путаницы, тем меньше шансов просмотреть ошибку.

Мы старались даже в первом вашем общении с программированием дать вам представление о принципе структурности программ. О том, как следует расчленять поставленную задачу на части и последовательно реализовывать их в блоках программы. Той же цели служит и принятый нами способ нумерации командных строк. Вы видите, что «анализатор» — ядро программы — имеет строки, пронумерованные иначе, чем дальнейшие ответвления, блоки подпрограмм вынесены в конец и пронумерованы так, что по номерам начальных строк легко определить их связь с переменной переключения и т. д. Все это совсем не обязательно — можно было бы после составления и опробования программы заставить компьютер заново перенумеровать все строки последовательными числами 1, 2, 3 и т. д. Действиям машины это не повредит ни в малейшей степени. Но мы не советуем так поступать. Когда-нибудь через год или через десять лет вам, может быть, захочется вспомнить составленную вами программу. И вот тут-то помогут все эти приемы: и напоминающие смысл ваших действий комментарии в командных строках, и несущая смысловую нагрузку нумерация строк, и даже удачно (по соображениям мнемоники) выбранные имена переменных. Этим качеством не стоит пренебрегать ради сомнительной «экономии букв».

Как рисовать на экране

Внимательно изучив сказанное выше о возможностях компьютерного языка БЕЙСИК-MSX, познакомившись, быть может, с другими книгами, где этот язык описан хоть и сложнее, зато более подробно, вы сможете начать общение с компьютером и на практике понемногу набирать опыт. Работа с числами и символами, конечно же, чрезвычайно полезна с практической точки зрения (суть многих профессий «ум-

ственного» характера как раз в этом и состоит). Однако среди многочисленных способностей персонального компьютера никак нельзя не упомянуть его рисовальный талант. Рисунки требуются людям во многих случаях: даже оформление научной статьи не обходится без них (и чаще всего они оказываются наиболее трудоемкими компонентами). Использование компьютера как помощника в этом деле просто неоценимо. По свидетельству тех, кто освоил работу на компьютере для подготовки текстов и рисунков, производительность труда возрастает буквально в десятки раз. Главное достоинство — легкость внесения любых исправлений, автоматизация часто повторяющихся процедур и целых фрагментов рисунка, возможность очень быстро получить копию подготовленного на экране рисунка (при помощи принтера).

Область «художественных» способностей компьютера чрезвычайно широка, но в рамках отведенного нам места придется коснуться лишь немногого. Далее мы опишем несколько самых распространенных и сравнительно простых команд языка БЕЙСИК-MSX, позволяющих начинающему программисту снабдить свои расчетные результаты или подготовленный на компьютере текст простыми рисунками — диаграммами, графиками, схемами.

Вот эти команды в порядке их важности.

SCREEN

Устанавливает основные параметры работы с экраном. Вслед за словом SCREEN могут быть поставлены шесть цифр (через запятые). Их смысл таков.
Первая цифра:
0 — текстовой режим экрана (24 строчки по 40 или по 80 символов в каждой).
1 — текстовой режим (24 строчки по 32 символа).
2 — графический режим высокого разрешения (256×192 точки).
3 — многоцветный графический режим (64×48 крупных «точек»).
4 — графический режим (высокое разрешение, улучшенные варианты графических «заготовок» — спрайтов).
5 — графический режим (256×212 точек изображения, каждая может быть окрашена палитрой в 16 цветов, выбираемых из 512 доступных «красок»).
6 — графический режим (512×212 точек, палитра в 4 цвета из 512 «красок»).
7 — графический режим (512×212 точек, палитра в 16 цветов из 512 «красок»).
8 — графический режим (256×212 точек, окрашенных в любой из 256 цветов).
Вторая цифра: 0, 1, 2, 3. Устанавливает размер спрайтов.
Третья цифра: 0 — нажатием клавиш не будет сопровождаться звуковым сигналом (писком), иная цифра — включение писка.
Четвертая цифра: 1 — низкая скорость передачи данных на кассетный магнитофон, если он используется (1200 бод), 2 — высокая скорость передачи (2400 бод).
Пятая цифра: 0 — если используется принтер стандарта MSX (он может печатать графические символы), иная цифра — в противном случае (тогда эти символы будут заменены пробелами).
Шестая цифра: вид развертки электронного луча в мониторе (0 обычная, 1 — с чередованием строк, 2 — специальная без чередования строк, 3 — специальная с чередованием строк).

WIDTH 35

Устанавливает на экране «поля» и размер символов. В режиме экрана 0 число, стоящее за словом WIDTH, может быть равным от 1 до 80 (интервал 1 — 40 дает крупные символы, а 41—80 дает мелкие). В режиме экрана 1 допустимый интервал станет у́же: 1—32.

CLS

Знакомая команда очистки экрана (она действует и в текстовом, и в графическом режимах).

COLOR 1, 3, 5

Устанавливает соответственно трем перечисленным через запятые числам три цвета: цвет переднего плана (знака, окрашенной точки рисунка и т. п.), цвет фона и цвет окантовки изображения (границы экрана). Эти числа должны лежать в диапазоне 0—3 или 0—15 (лишь в режиме экрана 8 диапазон расширяется до 0—255).

Номер цвета	Название	Состав: R	B	G
0	Прозрачный	0	0	0
1	Черный	0	0	0
2	Зеленый	1	1	6
3	Светло-зеленый	3	3	7
4	Темно-синий	1	7	1
5	Светло-синий	2	7	3
6	Темно-красный	5	1	1
7	Лазурный	2	7	6
8	Красный	7	1	1
9	Светло-красный	7	3	3
10	Темно-желтый	6	1	6
11	Светло-желтый	6	4	6
12	Темно-зеленый	1	1	4
13	Пурпурный	6	5	2
14	Серый	5	5	5
15	Белый	7	7	7

Буквами R, B, G в приведенной таблице обозначены цветовые компоненты (красный, синий, зеленый), из которых складываются другие цвета (указаны относительные интенсивности компонент).

В режиме экрана 5 и 7 можно пользоваться всеми 16 перечисленными в таблице цветами. В режиме экрана 6 доступны лишь цвета с номерами 0 — 3, а в режиме 8 — с номерами от 0 до 255. Номер цвета в последнем случае рассчитывается по формуле: 32*G+4*R+B, здесь R, G, B — относительные интенсивности компонент, как в приведенной выше таблице. Диапазоны допустимых значений интенсивностей: 0 — 7(R и G), 0 — 3 (B).

Та палитра, что описана в нашей таблице, выбирается компьютером по умолчанию, т. е. без специальных указаний. Однако при желании вы можете составить свою собственную палитру цветов, воспользовавшись еще одной командой:

COLOR=(...) COLOR=(2, 5, 1, 3)

Она описывает состав (в компонентах R, G, B) цвета вашей собственной палитры. В приведенном примере, показывающем образец записи этой команды, первое число (2) — номер цвета в палитре, а второе, третье и четвертое числа — интенсивности компонент. Диапазоны допустимых значений интенсивностей: 0 — 7 (R и G), 0 — 3 (B). Записав 16 таких команд, получите полную новую палитру.

Чтобы вновь вернуться к исходной палитре, после того как вам наскучит пользоваться собственным набором цветов, пошлите команду COLOR, не сопровождая слово никакими числами.

«Рисовальные» команды. Все описанные выше команды, как вы могли заметить, еще не дают возможности что-то нарисовать на экране. Они лишь подготавливают эту процедуру: устанавливают соответствующий режим экрана, дают вам в руки цветовую палитру. Приступим теперь к рисованию.

Собственно, без применения специальных написанных программистами-профессионалами сложных «художнических» программ, позволяющих в буквальном смысле слова рисовать на экране (скажем, с помощью светового пера или «мыши») наше рисование — на уровне простейших команд языка БЕЙСИК — есть не что иное, как кропотливый расчет координат каждой точки будущего рисунка и окрашивание ее затем в выбранный нами цвет. У традиционных, некомпьютеризованных живописцев такая манера именуется пуантилизмом (от французского слова, обозначающего точку).

Основная команда компьютерного пуантилизма:

PSET(...),...
PSET STEP(...),...

В скобках после слова PSET указываются координаты рисуемой точки, например: (56, 74). Начало экранной системы координат находится в левом верхнем углу экрана — точка (0, 0). Ось X идет слева направо, и значения координат — целочисленные от 0 до 255 (режимы экрана 2, 3, 4, 5 и 8 — см. описание команды SCREEN) или от 0 до 511 (режимы экрана 6 и 7). Ось У идет сверху вниз, и значения координат — от 0 до 191 (режимы экрана 2, 3 и 4) или от 0 до 211 (режимы экрана 5, 6, 7 и 8). В варианте команды PSET STEP (...) в скобках указываются **относительные** координаты, проще говоря изменения X и У по отношению к координатам точки, нарисованной на предыдущем шаге. То есть запись PSET STEP (5, 8) отвечает возникновению на экране точки, которая будет на 5 «делений» правее (ось X направлена вправо) и на 8 «делений» ниже (ось У направлена вниз!) предыдущей точки.

После скобок через запятую указы-

вается номер цвета рисуемой точки. Это позволяет не только рисовать, но и стирать уже нарисованные на экране фигуры: если, например, вам нужно удалить красную точку, расположенную на голубом фоне, достаточно будет просто заменить ее голубой — точка исчезнет.

LINE (...) — (...),...

Эта команда рисует отрезок прямой. В первых скобках указывают координаты начала, во вторых — координаты конца отрезка. Затем через запятые перечисляют вспомогательные параметры: номер цвета, каким должен быть нарисован отрезок, и одно из двух специальных обозначений — буквы B (латинская) или BF.

Добавление буквы B заставит компьютер нарисовать уже не отрезок, а прямоугольник, диагональю которого служил бы наш отрезок (но он теперь невидим), а стороны параллельны границам экрана. Добавление букв BF соответствует построению такого же прямоугольника, однако на этот раз он будет закрашен цветом, номер которого определен командой.

CIRCLE (...),...,...

Команда построения окружностей, эллипсов, дуг. В скобках указываются координаты центра (абсолютные или относительные с добавлением слова STEP, как было объяснено выше), затем через запятые перечисляются: радиус, номер цвета, начальный и конечный углы в радианах (для дуг), отношение осей (для эллипсов) — вертикальной к горизонтальной, т. е. высоты к ширине. Углы при построении дуг отсчитываются от горизонтальной оси против хода часовой стрелки (эти величины могут быть не только целочисленными).

PAINT (...),...,...

Эта команда позволяет ускорить неквалифицированные малярные работы, досаждающие любому художнику. Она заставляет компьютер закрасить нужным цветом любую область на экране, ограниченную замкнутой кривой. Представить ее действие наглядно проще всего так: вообразите, что в точке экрана с координатами, которые указаны как обычно в скобках, опрокинулся невидимый пузырек с краской того цвета, номер которого записан через запятую после скобок. Краска начнет равномерно растекаться по всем направлениям, останавливаясь лишь тогда, когда на пути встречается «преграда» — точка экрана, окрашенная ранее в «цвет границы» (его номер записывается через запятую после номера закрашивающего цвета). Если краска начала растекаться из точки, находящейся внутри замкнутого контура, цвет которого мы указали в команде как «цвет границы», через некоторое время вы получите на экране целиком закрашенную область. Если граница оказалась незамкнутой (не только чисто геометрически, а и по цвету), жидкая «краска» найдет эти дырки, вытечет наружу, продолжая расплываться дальше. В этом случае вполне возможно, что растекающаяся краска сможет постепенно заполнить все пустые места экрана.

Примечание. Отдельный «цвет границы» можно указывать только в режиме 3, 4, 5, 6, 7 или 8 экрана. В режиме 2 «цветом границы» всегда считается тот цвет, каким производится закрашивание.

DRAW«...» DRAW «C5U32R80L1»

Это самая сложная, но и самая мощная из рисовальных команд. Иногда ее называют даже не просто командой, а графическим макроязыком. Она позволяет в виде цепочки специальных символов описать довольно сложные перемещения рисующей точки («пера») по экрану, запрограммировав таким образом процесс изображения весьма сложных фигур, состоящих из отрезков прямых.

Процедура рисования нужной фигуры предварительно разбивается на этапы, или отдельные шаги: шаг вправо, влево, вверх, вниз, по какой-то из диагоналей, шаг с возвратом, поворот и т. д. Затем эти шаги последовательно описываются отдельными операторами команды DRAW. После того, исполняя команду DRAW, компьютер так же отдельными шагами (но очень быстро) рисует запрограммированную вами фигуру. Получается несравненно скорее и удобнее, нежели описывать построение

фигуры на нижнем языковом уровне — многочисленными ее точками.

Операторы этой интересной команды могут снабжаться такими приставками: B (означает просто движение рассматриваемой нами как кончик пера точки по экрану; следа за собой точка при этом не оставляет — «перо» сухо) или N (означает, что после выполнения следующего за такой приставкой оператора «перо» немедленно возвращается в предыдущую свою позицию — это позволяет избежать специального программирования обратного пути).

Операторы движения:

Un — перемещение на n «делений» вверх.
Dn — перемещение на n «делений» вниз.
Ln — перемещение на n «делений» влево.
Rn — перемещение на n «делений» вправо.
En — перемещение на n «диагоналей» вправо вверх.
Fn — перемещение на n «диагоналей» вправо вниз.
Gn — перемещение на n «диагоналей» влево вниз.
Hn — перемещение на n «диагоналей» влево вверх.

Слово «диагональ» здесь обозначает действительно диагональ элементарного квадрата на экране (с горизонтальной и вертикальной сторонами, равными одному «делению» осей координат). Значит, перемещение на пять «диагоналей» вправо вверх, например, отвечает мысленному перемещению на пять «делений» вправо и на пять «делений» вверх.

Еще один оператор: M x, y — перемещение «пера» в точку с координатами x и y. Это вариант задания абсолютных координат. Если перед числами x и y поставлен знак минус или плюс (M+5, +1, или M−3, +2), то в этом случае компьютер рассматривает числа x и y как относительные координаты.

Иные операторы:

Sn — устанавливает масштабный коэффициент изменения длины всех последующих перемещений «пера», действующий до его отмены таким же оператором. Коэффициент равен n/4, *n* может выбираться из диапазона 1—255. (Особый случай n=0 просто выключает действие масштабного коэффициента, то есть приравнивает его единице.) Не распространяется на оператор M, если в нем использованы абсолютные координаты точки.

An — устанавливает постоянный угол поворота (против часовой стрелки) всех последующих перемещений «пера» до отмены поворотов таким же оператором. Значения числа n могут быть следующими: 1 (поворот на 90°), 2 (на 180°), 3 (на 270°) и 0 (без поворота).

Cn — устанавливает номер цвета, которым должно рисовать наше «перо» (в интервале от 0 до 15).

Команду DRAW можно записывать и в неявном виде, например, так: вначале описать в программе символьную переменную, представляющую собой цепочку операторов движения «пера» и иных из набора этой команды (A$= «...»), а затем послать команду DRAW «XA$;» (здесь X — не переменная, а просто обязательная компонента синтаксиса, т. е. воспроизводимая в программах всегда неизменной). Это позволяет разбивать общую процедуру на этапы и, в частности, дает возможность пользоваться цепочками операторов длиной более чем 255 символов.

Во всех перечисленных операторах n, x и y могут быть не только числовыми константами, но и именами переменных (числовых). В этом случае перед именем переменной в операторе ставится предупреждающий машину знак= , а после имени — еще и точка с запятой. Пример: DRAW «M+=XA; , −=YB;», здесь XA и YB — имена переменных.

Вначале вам покажется, будто столь длинное описание команды с ее многочисленными операторами, каждый из которых тоже описан не коротко, не облегчит пользование графикой на практике. Но это не так. У вас под рукою всегда должен быть маленький справочник — хотя бы эти страницы, — однако пользоваться им в работе вы станете часто лишь на первых порах. Потом главное из всего обширного множества правил и условностей языка БЕЙСИК запомнится само, и обращаться к подобным описаниям при-

дется только в спорных или непонятных случаях. Ведь даже длинный список операторов движения в команде DRAW будет заучен вами без усилий после нескольких сеансов работы у экрана компьютера.

Заключение

Мы рассказали вам о том, что такое ваш персональный компьютер. Там, где описание было более детальным, подробности относились к конкретной машине «Ямаха MSX-2» и к конкретному алгоритмическому языку высокого уровня БЕЙСИК-MSX (версия 2.1). В тех местах, где излагались некие общие принципы работы с компьютером, содержится информация «широкого применения»: за какой бы персональный компьютер сходного класса сложности вы потом ни сели, он больше не покажется вам совершенно незнакомым.

Мы надеемся, что, получив предварительные представления не только о составе и возможностях персональных компьютеров, но и о горизонтах, которые открывает перед вами владение каким-либо из компьютерных языков, вы с гораздо большей охотой и с намного меньшим, чем прежде, страхом станете работать за клавиатурой. Еще бы: это позволяет переложить на плечи машины столько надоедливой рутинной работы — от длинных расчетов и нудных однообразных графических работ до пугающих своей грандиозностью текстовых, вроде составления алфавитного указателя к толстой книге.

Если у нас в будущем появится такая возможность, мы постараемся потом познакомить вас с персональным компьютером более близко, сделав в основном упор на совершенно не затронутые здесь темы — работа с мониторами, лентами и дисками, использование принтеров, использование сложных готовых программ для полноценной профессиональной работы непрограммирующего специалиста, описав иные «способности» компьютера (такие, как генерация звуков), пользование спрайтами и т. д. Впрочем, способностей этих столько, что всего и не перечислишь...

Литература

1. Дьяконов В. П. Справочник по алгоритмам и программам на языке БЕЙСИК для персональных ЭВМ. — М.: Наука, 1987. — 240 с.

2. Кетков Ю. Л. Диалог на языке БЕЙСИК для мини- и микро-ЭВМ. — М.: Наука, 1988. — 368 с.

3. Уолш Б. Программирование на БЕЙСИКе. — М.: Радио и связь, 1987. — 336 с.

4. Фокс Д., Фокс А. БЕЙСИК для всех. — М.: Энергоатомиздат, 1986. — 176 с.

ПЕРЕВОДЫ

ПРИОРИТЕТНЫЕ НАПРАВЛЕНИЯ РАЗВИТИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ В США

Отдел вычислительной техники и технологии Национальной Академии наук США опубликовал обзор, в котором, в частности, указаны наиболее перспективные направления разработок средств вычислительной техники, требующие значительных финансовых вложений.

Среди них указывались:

1. Сверхнадежные компьютерные системы, которые будут безотказно работать в течение 20 и более лет. Эти системы представляют особую ценность для исследований в космосе и в местах повышенной опасности. Ввиду того, что в настоящее время вычислительная техника несовершенна, сверхнадежность фактически означает дублирование на микроэлектронном уровне. Эти системы должны быть снабжены средствами искусственного интеллекта (ИИ) для обнаружения неполадок и перестройки запоминающих устройств (ЗУ) и информационных каналов для работы в новых условиях.

2. Сверхкомпьютеры с быстродействием в триллион (10^{12}) оп/с, т. е. по крайней мере в тысячу раз быстрее существующих компьютеров. Это потребует значительного прогресса в микроэлектронике, в создании кристаллов памяти, содержащих миллиарды бит, новой архитектуры компьютеров с параллельным использованием сотен и даже тысяч процессоров.

3. Телефоны-переводчики. Они позволят вести беседу по телефону на разных языках. Эти системы потребуют значительных разработок в области ИИ, так как будут обладать независимым (от говорящего) распознаванием продолжительной речи, причем с большим словарным запасом; естественно звучащими системами речевого синтеза, моделирующими речевые особенности говорящего; системами машинного перевода, способными работать с любой речью, включая жаргонные, профессиональные, разговорные и многозначные слова, незаконченные фразы и т. д.

В Японии уже начались работы над семилетним проектом (стоимостью 120 млн. долл.) создания телефона, который будет переводить с япон-

ПЕРЕВОДЫ

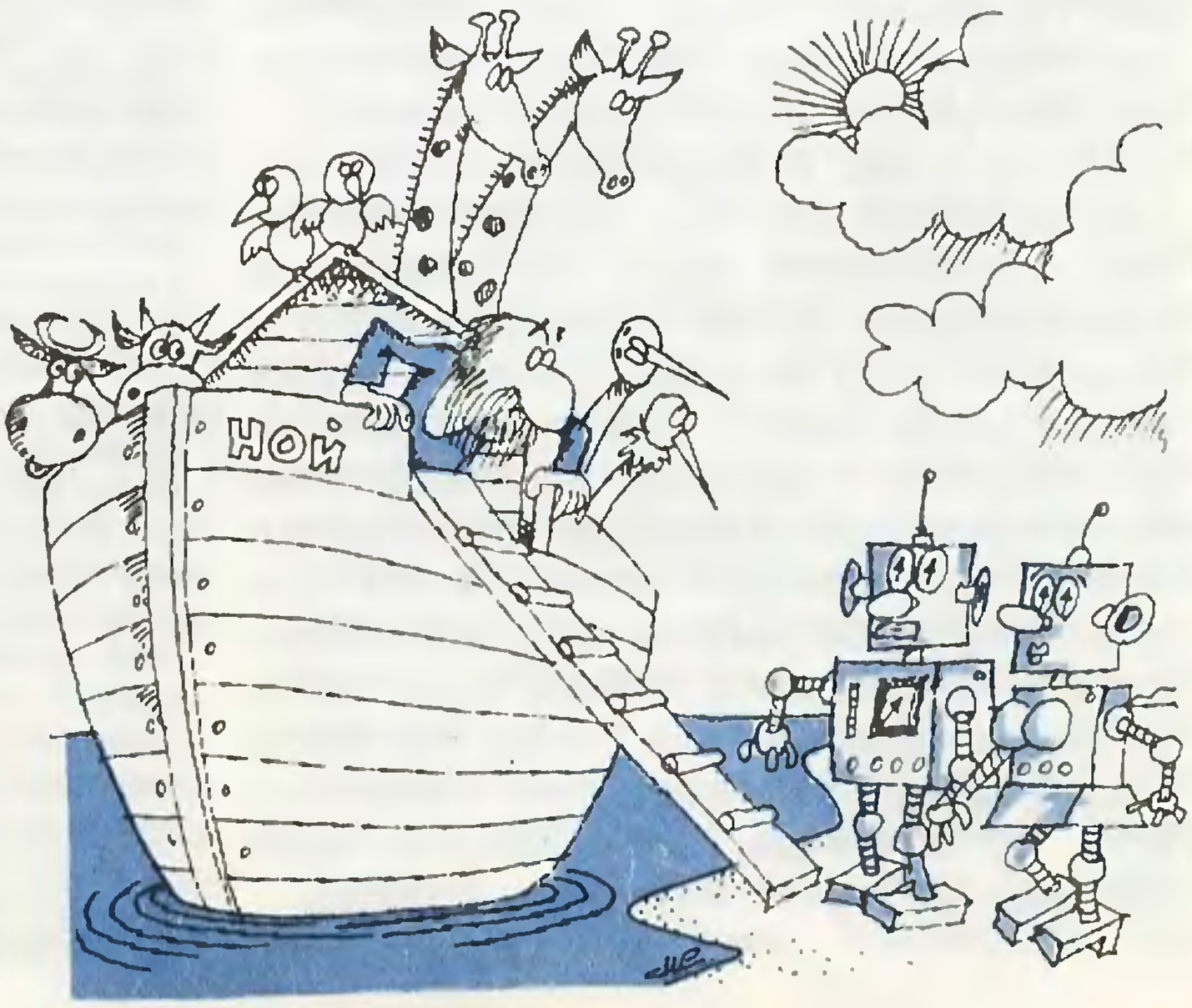
Умные молекулы для думающих компьютеров

Возможно, биология в большей мере близка к решению проблемы будущих «думающих компьютеров», чем электроника. На состоявшемся в декабре 1988 г. международном симпозиуме в Японии исследователи США удивили участников идеями получения «интеллектуальных» веществ, новых видов чувствительных устройств и, возможно, в недалеком будущем искусственного мозга.

Михаэл Конрад из Wayne State University (Детройт) говорил о веществах, названных им «хай — IQ вещества», молекулы которых в компьютерах будущего смогут распознавать изображение. На симпозиуме биоэлектронных и молекулярных устройств Стюарт Хамерофф из Аризонского университета продемонстрировал работу на компьютерах, основанных на структурах, названных им «микрососуды»: мелких волокнах протеина, которые находятся в клеточной цитоплазме. Оба ученых заявили, что процесс познания заключен в молекулярной деятельности нейронов мозга. На симпозиуме говорилось, что недалек тот день, когда биомолекулярные устройства будут состоять из органических веществ, входящих в электронную схему. Исследователи уже работают над микросхемами, обрабатывающими информацию, полученную от зрительных процессоров, которые содержат бактериородопсин — светочувствительное вещество, связанное со зрительным пигментом, родопсином, находящимся в глазах человека.

Второй, более предпочтительный класс устройств, будет состоять только из искусственно полученных протеинов. Эти устройства будут использоваться для обнаружения химических веществ. Исао Карибе из Токийского технологического института разработал «микросхему свежести», которая с 1991 г. будет закладываться в пакет вместе с продаваемой рыбой. Микросхема будет обнаруживать пахнущие вещества, образующиеся при гниении рыбы, и будет сообщать покупателю (видимо, по изменению цвета пакета) о том, что рыба испорчена.

Долгосрочная задача Хамероффа — создание искусственного мозга, полученного из цитоплазмы методом генной инженерии. Заглядывая в далекое будущее, он видит возможность программирования в этих структурах характера человека. Еще одно аналогичное применение



— искусственная цитоплазма, которая будет копировать матрикс микрососудов. Такая структура, по мнению Хамероффа, положит начало новым возможностям, связываемым исследователями с интеллектом. В XXI веке ученые, возможно, научатся образовывать из этих матриксов «мозг» в самокорректирующихся калькуляторах или в интеллектуальных текстовых процессорах.

Другое ближайшее применение биомолекулярных устройств, предложенное Тоуесакой Маризуми из Токийского технологического института — искусственный нос, — наиболее подходящий орган для первых попыток создания искусственной нервной системы. По предсказанию Хамероффа, искусственный нос должен появиться в ближайшее десятилетие.

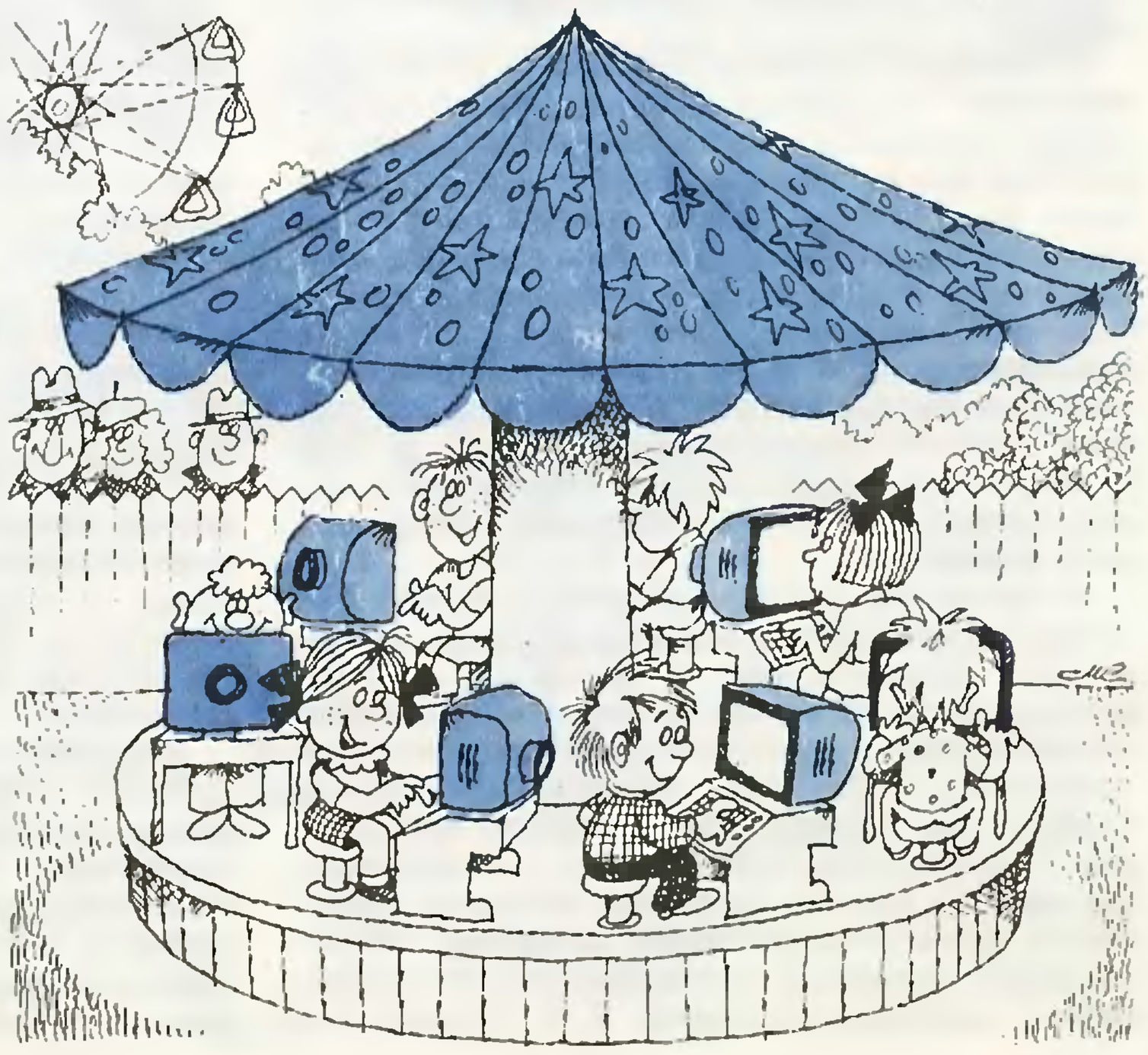
New Scientist,
24/31 декабря 1988, с. 29
С. А. Утенков

БЛЕСТЯЩАЯ МЫСЛЬ

В США поступила в продажу нейронная программа «Модель мозга», которая рекламируется следующим образом. Если у Вас при чтении этих строк слова вдруг начнут расплываться и превратятся в массу бессвязных мыслей, то Вы, видимо, сходите с ума. 100 биллионов нейронов Вашего мозга не выстроились в необходимые синаптические цепочки, благодаря которым и существует разум и интеллект. Но не беспокойтесь, наша программа моделирует функции мозга.

Программа — это пособие по моделированию нейросвязей (нейросетей), их взаимодействию в мозге и использованию для построения искусственного интеллекта компьютерных систем. Программа на нейросети моделирует память, сознание и понимание на любом компьютере, работающем в системе MS—DOS. Пользователи могут собирать 1,2 тыс. нейронов в простые схемы и формировать из них более сложные системы. Программа снабжена сигнализацией: появляется световой сигнал, если удается добиться синаптического соединения — прообраза блестящей мысли.

Newsweek,
3 октября
1988, с. 3



МИКРОПРОЦЕССОРЫ

ЭВМ

РОБОТОТЕХНИКА

Эта статья для инженерно-технических работников и студентов, для которых описан опыт применения одного из простейших отечественных средств микропроцессорной техники в производственной и учебной практике, а также в быту. Предлагаем вам рекомендации из опыта автора по использованию отечественных микрокалькуляторов для управления различными, в основном тепловыми, процессами. Такие задачи наиболее часто встречаются в практике.

Ю. В. КОЛОСОВ

УПРАВЛЯЮЩИЕ КАЛЬКУЛЯТОРЫ — ЭКСПЕРИМЕНТАТОРУ И ПРАКТИКУ

В настоящее время отмечается широкое распространение калькуляторов как наиболее доступного и массового средства вычислительной техники. Все большую популярность получают программируемые калькуляторы, которые целесообразно применять при необходимости многократного решения однотипных задач. Соответственно в последнее время появилось значительное количество публикаций, как отечественных, так и переводных, связанных с использованием калькуляторов.

В последнее время мы все чаще встречаемся с терминами «микропроцессоры», «микропроцессорная техника». А всегда ли мы знаем, что это такое? Как они выглядят и где их можно использовать? А доступна ли эта техника, например, школьникам, которые в настоящее время изучают дисциплину «Информатика и вычислительная техника» и обычно умеют работать по крайней мере с калькуляторами, хотя часто и не понимают, а зачем нужно быстро считать? А может быть, эта техника полезна домохозяйкам, которые в свои школьные годы никак не могли встретиться с упомянутой дисциплиной и, скорее всего, относятся к ней с опасением?

К настоящему времени имеется большое количество публикаций, посвященных вопросам использования калькуляторов в расчетной инженерной и учебной практике. Например, «Радиотехнические расчеты на микрокалькуляторах» Я. К. Трохименко, Ф. Э. Любич, «Микрокалькуляторы с программным управлением в учебной лаборатории» М. Ф. Поснова, Н. Н. Поснова — математические расчеты, расчеты в области механики, сопромата и других математических дисциплин, «Решение задач по химии с использованием программируемых микрокалькуляторов» В. Д. Майборода и др., «Прикладные программы для Б3-21» А. Н. Цветков — расчеты финансов, статистика и т. д. Эти издания ориентированы на использование калькуляторов типа Б3-21, Б3-34, МК-56, МК-54, МК-52, МК-61.

В настоящее время начинает получать все более широкое распространение и другой класс так называемых программируемых управляющих калькуляторов, например, из отечественных это МК-46, МК-64, из зарубежных — HP-41C фирмы Hewlett Packard, T1-59 фирмы Texas Instruments. К достоинству подобных устройств относится возможность реализации инженерных законченных систем при решении задач управления процессами и технологиями, что может быть использовано в учебной и бытовой практике (приготовление пищи, управление режимами выращивания культур в теплицах, как комнатных типа «Флора», «Тульпс», так и садовых, управление комнатными инкубаторами), наконец, в промышленности, особенно при отладке или управлении новыми технологиями.

К одному из последних и достаточно удачных изданий, рассказывающих об использовании управляющих калькуляторов, относится «Программирование на микрокалькуляторе МК-64» И. Н. Антипова, которое может служить хорошим пособием для изучения методики программирования при работе с этими калькуляторами. Попутно следует отметить, что в принципе это неправильное применение управляющего калькулятора, ибо его назначение управлять, а не вычислять. При эксплуатации калькулятора МК-64 в режиме вычислений не используются главные его технические особенности, которыми он и отличается от большой гаммы вычислительных калькуляторов, а именно интерфейсы связи с внешним объектом, аналого-цифровой преобразователь, что, кстати говоря, и превращает его в микропроцессорный контроллер. Все выше сказанное не упрек автору, ибо пособие действительно хорошее и доступное пониманию.

И наконец, следует еще раз подчеркнуть, что в качестве управляющих могут использоваться только микрокалькуляторы типа МК-46 и МК-64, содержащие интерфейсные устройства связи с объектом и соответствующие разъемы для подключения внешних устройств. Все остальные же программируемые, но неуправляющие калькуляторы могут быть использованы для решения задач

управления только после соответствующих доработок.

Традиционно проектирование микропроцессорных систем включает в себя: разработку аппаратного обеспечения (состав элементов или устройств и схем их соединения или включения) и разработку программного обеспечения (составление алгоритмов и программ).

Аппаратное обеспечение системы управления

Обычно узел управления выполняется в виде замкнутой системы. Объект — это оборудование, прибор, аппарат, в котором выполняется определенный технологический процесс. Устройство управления подключается к объекту и выполняет все задачи по управлению технологическим процессом.

В качестве устройства управления будем рассматривать программируемый управляющий калькулятор типа МК-64.

Программируемый управляющий калькулятор МК-64 относится к микропроцессорным устройствам управления. Его можно использовать для управления медленно текущими процессами с реализацией простейших алгоритмов управления. При этом он выполняет следующие функции:

— опрос дискретных датчиков типа включено — выключено;

— опрос аналоговых датчиков, например, датчиков температуры — термометров сопротивления, термопар и др.;

— установку заданного значения регулируемой величины;

— выдачу команд для управления объектом.

С помощью программируемого управляющего калькулятора можно решать различные технологические и организационные задачи:

— контроль и индикацию измеряемого параметра, например температуры;

— поддержание заданного значения параметра, например температуры, — реализация позиционного (нелинейного) регулятора;

— программное во времени изменение параметра, например температурного режима;

— реализацию циклических процессов управления: включение, выключение, выдержка времени и т. д.;

— реализацию линейного закона регулирования (алгоритма П-регулятора).

Структура и комплектность системы

В общем случае управляемый технологический объект (оборудование) содержит исполнительно-регулирующие устройства и измерительно-информационные средства (датчики параметров). В тепловых технологических объектах бытового назначения, характеризуемых относительно низкими эксплуатационными температурами (до 200—250°С), в качестве измерительных средств целесообразно использовать, например, промышленные металлические термометры сопротивления. Иногда используются и полупроводниковые термометры (термисторы) из-за их более высокой чувствительности. Однако нелинейность их характеристик и, что особенно неудобно, их невзаимозаменяемость создают определенные трудности при работе. Из промышленных термометров сопротивления может быть рекомендован платиновый термометр ТСП, градуировка 22 (100П). Выбор последнего целесообразен из-за его идентичности термометрам международной градуировки, чем обеспечивается их взаимная совместимость как по аппаратуре управления, так и по средствам измерения.

В качестве исполнительно-регулирующих устройств тепловых объектов удобнее всего применять наиболее надежные тиристорные силовые блоки.

На рис. 1 приведена блок-схема рассмотренной системы. Как видно из рисунка, для стыковки объекта (совместно с датчиками) и устройства управления, в качестве которого в рассматриваемой системе использован калькулятор МК-64, требуются блоки согласования: нормирующие преобразователи, силовой блок и блок управления калькулятором.

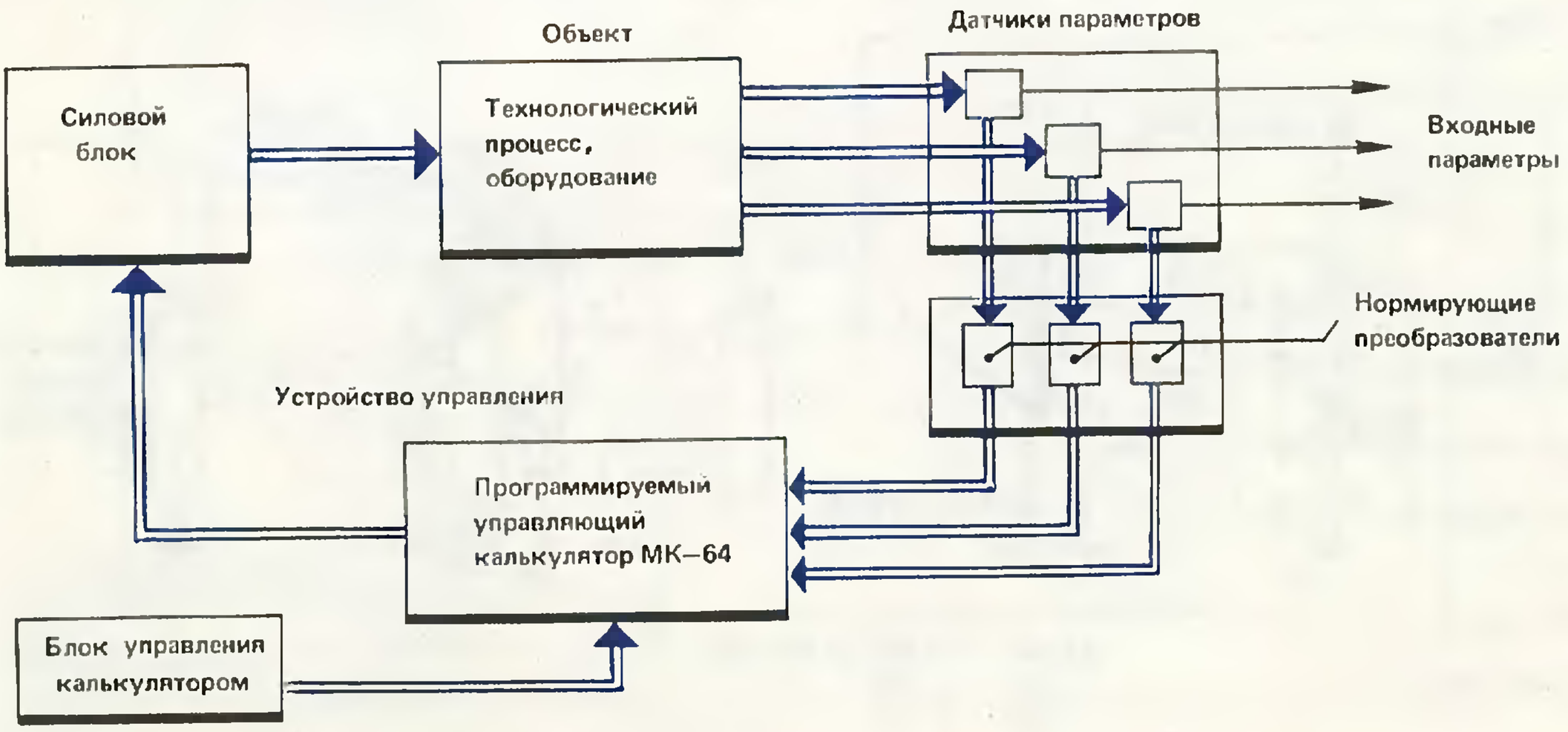


Рис. 1

Блоки согласования и управления

В настоящее время в практике применения средств вычислительной техники для управления процессами и оборудованием выбраны единые метрологические параметры, которые приняты в качестве нормированных сигналов. Чаще всего это сигнал напряжением 0—10 В (иногда 0—5 или 0—1 В). Промышленные датчики параметров обычно имеют на выходе ненормированные сигналы, например, термометры сопротивления в омах, термопары в милливольтах и т. д., возникает необходимость в использовании нормирующих преобразователей, обеспечивающих унификацию (преобразование) сигналов различных датчиков.

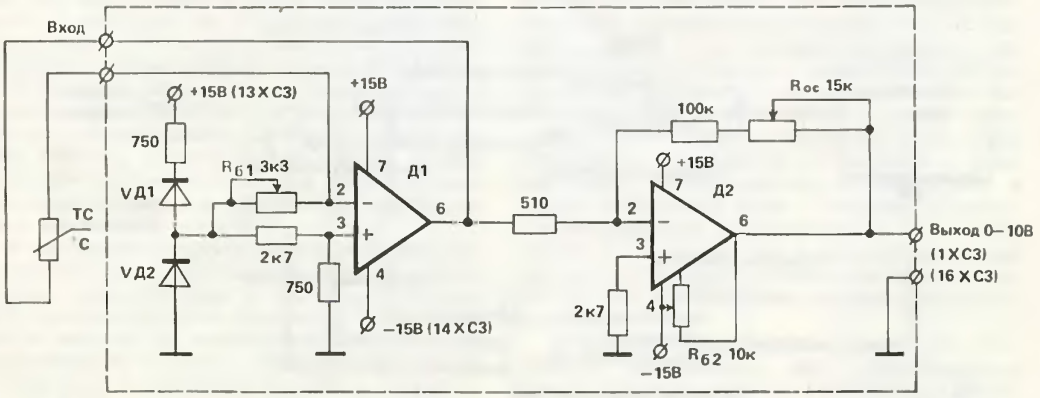
Отечественной промышленностью выпускаются нормирующие преобразователи Ш78 и Ш79 (первые для омических датчиков сопротивления, вторые для потенциометрических датчиков). Эти приборы отличаются достаточно крупными габаритами и значительной стоимостью, превосходящей габариты и стоимость программируемых калькуляторов. Из-за этого их использование вряд ли оправдано. Для малогабаритных и сравнительно дешевых микропроцессорных устройств управления рядом зарубежных фирм — «Кореси» (Франция), «Евротерм» (Англия, ФРГ, США) — налажен выпуск малогабаритных нормирующих преобразователей для наиболее распространенных тепловых датчиков: термометров сопротивления и термопар, иногда конструктивно объединенных в одном комплекте (корпусе) с датчиком [1].

Для термометра сопротивления, например платинового, градуировки 100 П, совместимой, как уже отмечалось выше, с международной градуировкой, может быть предложена схема простого нормирующего преобразователя (рис. 2). На рисунках надписи типа (15ХС3) обозначают: 15 номер контакта разъема ХС3 на калькуляторе МК-64, имеющем три 16-контактных разъема (ХС1, ХС2, ХС3).

Схема выполнена на базе двух операционных усилителей, один из которых выполняет роль преобразователя сопротивления в напряжение, а второй — усилителя напряжения. На VD1, VD2 — стабилизатор опорного напряжения [2]. Сравнительно простая схема преобразователя предопределяет его незначительную стоимость и габариты.

Безусловно, его несколько худшие, по сравнению с Ш78, метрологические характеристики могут быть оправданы именно при работе с простейшими управляющими калькуляторами типа МК-64, отличающимися также более низкими, сравнительно, например, с микро-ЭВМ К1-20, метрологическими характеристиками и быстродействием. В случае же применения относительно быстродействующих микро-ЭВМ может быть предложен программный способ улучшения метрологических характеристик, предоставляющий возможность многократных измерений и усреднения измеряемого параметра. При работе систем с обратной связью, как в рассматриваемом случае (рис. 1), периодические флюктуации мало сказываются на работе управляемой системы, так как система непрерывно корректируется, а управляемая величина систематически усредняется. Другими словами, случайные ошибки при периодических измерениях отфильтровываются низкочастотной системой.

При изготовлении схемы преобразователя потребуется выполнить ее настройку на заданный диапазон измеряемых температур при принятой градуировке термометра. Настройку осуществляют, фиксируя показания штатного термометра в холодной (при 0°С) и кипящей (100°С) воде. Удобнее воспользоваться магазином сопротивления для имитации сопротивления термометра. Настройка схемы выполняется по этапам. Вначале производится балансировка операционного усилителя Д1 переменным сопротивлением $R_{б1}$ на его входе. Для этого на магазине устанавливается сопротивление, соответствующее нулевой температуре (100 Ом для ТСП 22 град), на выходе — в точке 6 — должно быть 0 В. Затем производится балансировка второго операционного усилителя переменным сопротивлением $R_{б2}$. При этом на выходе нормирующего преобразователя (рис. 2) устанавливается тоже нулевое напряжение. Далее выполняется окончательная настройка, для



Д1; Д2 к 140 УДБ (к 140 УД7)
VД1 Д 809
VД2 Д 223

Рис. 2

чего на магазине устанавливается сопротивление, соответствующее предельной температуре, на которую рассчитывается данный преобразователь (например, 139 Ом при пределе измерения 100°С или 177 Ом при 200°С и т. д.). При этом с помощью переменного сопротивления R_{oc} на выходе нормирующего преобразователя устанавливается напряжение 10 В.

На рис. 3 приведена схема силового блока управления, выполненная на базе управляющего тиристора типа КУ202Н. Для защиты калькулятора введена развязка силовых цепей тиристорной части схемы и слаботочных цепей калькулятора с помощью тиристорного оптрона.

Работа калькулятора в режиме управления сопровождается его очередным запуском после каждого опроса датчиков. Блок управления и выполняет процесс перезапуска калькулятора в режиме управления, так как по техническим характеристикам МК-64 для его запуска после каждой очередной выдачи сигнала управления внешними устройствами требуется формировать повторный сигнал «Пуск». Схема блока управления приведена на рис. 4.

Для предотвращения отключения исполнительного устройства от импульсной помехи, возможной при перезапуске калькулятора на входе силовой схемы управления, введена интегрирующая цепочка (33 к, 5,0 мкФ — рис. 3). Для удобства включения силового блока к разъему XC2 калькулятора предусмотрена перемычка 15XC3—15XC2, обеспечивающая подключение цепи питания — 27 В на разъем XC2. Никаких дополнительных регулировок и настроек при изготовлении силового блока не требуется.

При разработке схемы блока управления калькулятором была учтена следующая особенность работы МК-64. При каждом запуске калькулятора по входу «Пуск» происходит сброс триггеров, формирующих выходные управляющие сигналы, что нарушает организацию непрерывного управления процессами. В этом случае для сохранения выходных управляющих сигналов потребуется вводить в схему специальные запоминающие триггеры, а это нелогично — сбрасываем штатный триггер и его компенсируем дополнительным триггером. Поэтому был использован второй, также предусмотренный в калькуляторе МК-64, способ запуска через развязывающие диоды по входам «Вх.инф.Р9...Вх.инф.Р12». При этом способе запуска не происходит сброса триггеров, формирующих выходные управляющие сигналы.

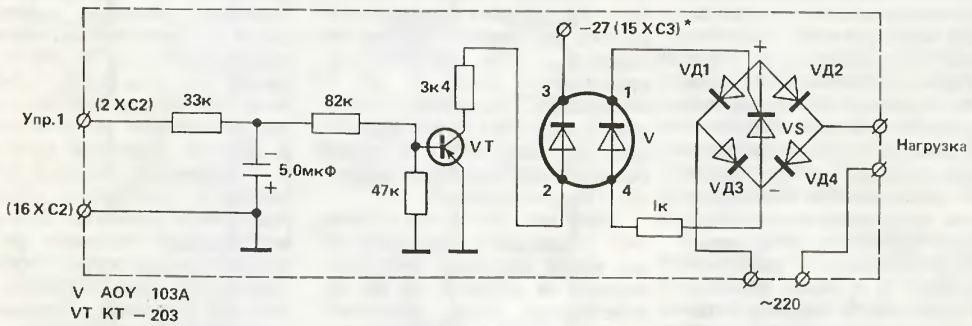


Рис. 3

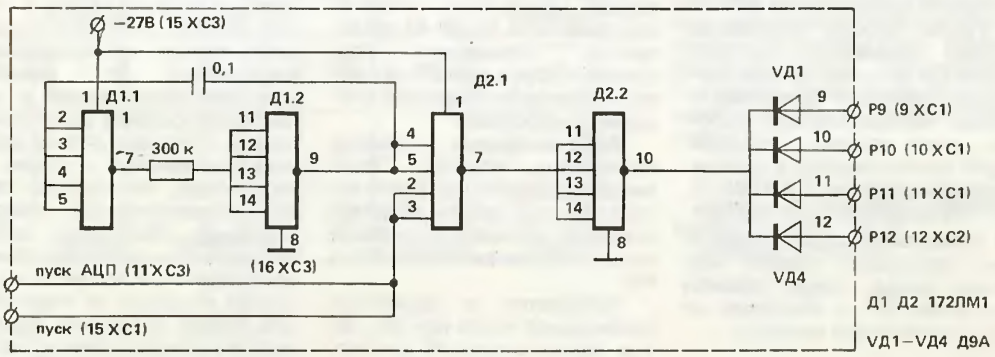


Рис. 4

Для обеспечения перезапуска калькулятора в режиме непрерывного управления используется схема генератора, выполненная на микросхеме Д1, обеспечивающая непрерывное формирование импульсного запуска. Поступающие на «Вх.инф.Р9...Вх.инф.Р12» импульсы запуска могут привести к искажению информации, принимаемой по этим же входам на каждом очередном цикле измерений, поэтому на вход микросхемы Д2.1 подается сигнал «Пуск АЦП», запрещающий прохождение импульсов запуска на время приема входной информации от АЦП. Одновременно этот сигнал подается на вход «Пуск»: для сброса триггеров выходных управляющих сигналов перед каждым очередным циклом формирования нового управляющего воздействия.

Все описанные внешние устройства согласования подключаются к калькулятору МК-64 с помощью штатных штеккерных разъемов, входящих в комплект калькулятора. Соответственно на схемах устройств согласования отражены (см. в скобках) номера контактов и номера одного из трех штеккерных разъемов калькулятора, на которые выводятся необходимые точки соединения. И само собой разумеется при необходимости все устройства согласования могут быть отключены и калькулятор может быть использован в обычной расчетной практике.

Основные правила и режимы работы на МК-64 [3]

Настоящая статья предназначена для читателя, имеющего навыки работы с программируемыми калькуляторами.

К основным режимам работы программируемого управляющего калькулятора относится режим программирования и режим автоматической работы. На полях страницы приведено командное обеспечение МК-64.

В режиме программирования осуществляется набор требуемых программ. Для перехода в режим программирования необходимо выполнить следующую последовательность действий: а) возврат в нулевой адрес — клавиша В/0; б) переход в режим программирования — это Р, ПРГ.

В режиме автоматической работы производится процесс вычисления и управления. Для перехода в режим автоматической работы необходимо выполнить: а) переход в режим «автоматическая работа» Р АВТ; б) возврат в нулевой адрес В/0.

После набора программы обычно выполняется ее контроль. Он включает в себя следующие операции: а) переход в режим «автоматическая работа», далее возврат в нулевой адрес и снова переход в «режим программирования» Р АВТ В/0, Р ПРГ; б) клавишей «шаг вправо» Ш проверяются коды команд; в) при проявлении ошибки вернуться назад, нажав клавишу шаг влево ШГ, и повторить набор исправляемой команды.

Служебные коды и комбинации команд

Команда безусловного перехода. Команда возврат в нулевой адрес В/0 при переходе в режимы программирования и автоматическая работа вводится только при условии выполнения этих режимов с естественного первого адреса. Если эти режимы начинаются с другого (не первого) адреса, то переход к этим адресам осуществляется командой БП, а следующей клавишей набирается код шага новой адресации, например, БП, 3 означает безусловный переход к адресу 34 — коду клавиши 3.

Заметим кстати, что в перечне командного обеспечения нет кодов операций, оканчивающихся на цифру 0, но им соответствуют коды, оканчивающиеся на цифру 6. Поэтому при необходимости адресации к ячейкам памяти 10, 20, 30 и т. д. вместо этих кодов записываются адреса 06, 16, 26 и т. д.

Команда перехода к подпрограмме выполняется аналогично команде безусловного перехода, изменяется только первая нажимаемая клавиша, например ПП 3. На МК-64 реализуется вложение подпрограмм друг в друга. Всего допускается до пяти уровней вложения подпрограмм.

Ввод исходных данных и служебных команд. После ввода программы до начала автоматической работы вводятся исходные данные и служебные коды в рабочие регистры с Р2 по Р9.

Набирается на индикаторе необходимое число или код, затем соответствующий регистр, например, 11010001 ...Р 9.

После ввода данных в регистры может быть произведена их проверка вызовом этих данных из соответствующих регистров на индикатор, например, F 9 на индикаторе засвечивается 1101 0001.

Занесение допусковых величин. Допусковые (заданные) значения регулируемой величины заносятся в кольцевой регистр стековой памяти — всего их шесть — для обеспечения допускового контроля информации и соответствующей организации управления по шести каналам. При этом следует учитывать, что допусковые значения первого канала располагаются в шестом стековом регистре, т. е. в регистре, который заполняется в последнюю очередь при наборе допусков. Допусковое значение включает три последние цифры полного восьмиразрядного кода, например, 10 000 300. Введение допускового значения осуществляется следующим образом: набирается допуск, затем он вводится в стековый кольцевой регистр, например, 10 000 300 ... Р⇄.

Ввод и масштабирование измеряемой информации. Информация по первому каналу вводится во второй регистр памяти, по второму каналу в третий регистр и т. д. Следует иметь в виду, что измеряемая информация от внешних устройств записывается в три последних разряда регистровой памяти и поэтому для ее удобного отображения и чтения на индикаторе желательно вводить соответствующее масштабирование. Последнее можно осуществлять двумя способами: программным в процессе вычислений и вывода на индикатор или лучше введением множителя в очередной регистр, который принимает соответствующую информацию. Например,

КОМАНДНОЕ ОБЕСПЕЧЕНИЕ МК–64

код опер.	обозначение клавиши
01	P 0
02	P 0
03	P e^{LX}
04	0
05	F B↑
06	B↑
11	P 1
12	F 1
13	P ln
14	1
15	F ↔
16	↔
21	P 2
22	F 2
23	P П
24	2
25	F X
26	X
31	P 3

КОМАНДНОЕ ОБЕСПЕЧЕНИЕ МК – 64

код опер.	обозначение клавиши
32	F 3
33	P e^x
34	3
35	F ÷
36	÷
39	P нол
49	P x⩾0
59	P x=0
41	P 4
42	F 4
43	P
44	4
45	F 1/x
46	•
69	P x<0
78	c/n
79	P x≠0

КОМАНДНОЕ ОБЕСПЕЧЕНИЕ МК – 64

код опер.	обозначение клавиши	
51	P	5
52	F	5
53	P	1–1
54		5
55	F	X^2
56		1–1
61	P	6
62	F	6
63	P	ВП
64		6
65	F	$\sqrt{}$
66		ВП
71	P	7
72	F	7
73	P	Cx

КОМАНДНОЕ ОБЕСПЕЧЕНИЕ МК – 64

код опер.	обозначение клавиши	
74		7
75	F	Cx
76		Cx
81	P	8
82	F	8
83	P	cos
84		8
85	F	–
86		–
91	P	9
92	F	9
93	P	sin
94		9
95	F	+
96		+

измеряется напряжение 3,24 В — оно записывается и далее высвечивается на индикаторе 0,000324. Введя множитель 1.10^4, получаем на индикаторе 3.24. Масштабирование осуществляется следующим образом: набирается множитель, затем выбирается соответствующий регистр, например, для введения множителя $1 \cdot 10^4$ в регистре 2—1 ВП 4 ... Р 2.

Выбор и ввод кода эксперимента. Код эксперимента включает 8-разрядное число, где первый разряд означает количество опрашиваемых внешних устройств, второй разряд 1 асинхронный режим работы; третий разряд 0; четвертый разряд 0 — работа без допускового контроля или 1 работа с допусковым контролем; пятый, шестой и седьмой разряды произвольные, восьмой 1, 2 или 3 означает характер вывода на цифропечать и даже при отсутствии такого вывода обязательно набираются, так как это является условием повторения эксперимента, т. е. обеспечения непрерывного процесса управления. Например, код 11010001 означает опрос одного датчика с допусковым контролем и обеспечением непрерывного процесса.

Ввод кода эксперимента осуществляется следующим образом: набирается код эксперимента, затем выбирается 9 регистр, например, 11010001 ... Р 9.

Код повторения эксперимента. Код повторения эксперимента, иначе обеспечивающий режим непрерывного управления, включает следующую комбинацию команд:

В↑ ВП 1 0 ↔С/П

Программное обеспечение систем управления

Как уже отмечалось выше, программное обеспечение зависит от характера решаемой задачи. При этом можно использовать разный комплект из вышерассмотренного набора аппаратных средств.

Контроль температурных параметров относится к одной из наиболее распространенных задач, встречающихся как в производственной, так и в бытовой практике.

Удобнее всего в качестве датчика температуры использовать промышленные термометры сопротивления. В случае применения платинового термометра ТСП градуировки 22 можно применить вышеприведенную схему нормирующего преобразователя. Для других типов термометров, например медных (ТСМ, градуировки 23, 24) или платиновых других градуировок (ТСП, градуировки 20, 21), потребуется осуществить настройку той же схемы по вышеописанной методике. Наконец, при отсутствии промышленного термометра в качестве датчика можно использовать медную проволоку. Для удобства ее сопротивление (т. е. длину) следует подобрать. Например, в холодной воде R°100 Ом, в горячей (100°С) — 139 Ом.

Рассмотренный блок управления калькулятором должен быть подключен только при необходимости организации непрерывных измерений после запуска калькулятора. Если же ставится задача одиночных или разовых измерений, то блок управления не нужен. В последнем случае для каждого очередного измерения осуществляется запуск восьмикратным нажатием кнопки ПУСК.

Контроль одного параметра. Для решения задачи измерения и визуального контроля одного измеряемого параметра следует набрать следующую программу.

Адрес	Команда	Код команды	Содержание команды
01	F 2	22	Индикация измеряемого параметра
02	В↑	06	Код повторения эксперимента
03	ВП	66	
04	1	14	
05	0	04	
10	←→	16	
11	С/П	78	
12	БП	58	Возврат в начало программы (В адрес 01)
13	Р О	01	

В приведенной программе ячейка 10 следует за ячейкой 05, это не ошибка, а специфика микрокалькулятора МК-64, в котором при адресации ячеек памяти используют шестиричную систему.

Команда F 2 в начале программы обеспечивает вызов измеряемого параметра из регистра 2 для его визуальной индикации на время задержки при повторении эксперимента.

Комбинация команд БП Р 0 обеспечивает безусловную переадресацию или безусловный переход на адрес, указанный кодом команды Р 0 −01.

После набора и контроля набранной программы следует ввести служебные команды. Сначала код эксперимента 1100 0001 в регистр 9, обеспечивающий контроль одного параметра и режим повторения эксперимента. Затем множитель в регистр 2, обеспечивающий выдачу измеряемого параметра в нормированном виде. Например, при использовании нормирующего преобразователя с выходным сигналом 0—10 В введение множителя $1 \cdot 10^4$, а именно 1 ВП 4 ... Р 2, обеспечивает индикацию показаний в вольтах. Для нормирующего преобразователя, рассчитанного на диапазон температур 0—100°С, введение множителя $1 \cdot 10^5$ или 1 ВП 5 ... Р 2 обеспечивает индикацию непосредственно в градусах, что особенно удобно.

Наконец, желательно проверить числа, введенные в регистры 9 и 2, вызовом содержимого этих регистров — должно высветиться 1100 0001 в регистре 9 и 10000 для множителя $1 \cdot 10^4$ или 100000 для множителя $1 \cdot 10^5$ в регистре 2.

Нажав клавиши Сх В/0, подготовить начало процесса. Запустить процесс, нажав кнопку ПУСК. При этом на табло высвечивается измеряемая величина.

Контроль двух и более параметров. При необходимости решения задачи измерения и контроля двух параметров программа выглядит следующим образом.

Адрес	Команда	Код команды	Содержание команды
01	F 2	22	Индикация 1-го параметра
02	В↑	06	Код повторения эксперимента
03	ВП	66	
04	1	14	
05	0	04	
10	←→	16	
11	С/П	78	
— — — — — —			
12	F 3	32	Индикация 2-го параметра
13	В↑	06	Код повторения эксперимента
14	ВП	66	
15	1	14	
20	0	04	
21	←→	16	
22	С/П	78	
23	БП	58	Возврат в начало программы
24	Р О	01	

Из приведенной программы видно, что фрагмент ее для измерения и индикации второго параметра отличается только вызовом измеряемого параметра из регистра 3.

Служебные коды для этого случая таковы: код эксперимента 2100 0001, где первая цифра 2 означает опрос двух датчиков; соответственно и мно-

жители вводятся в два регистра — 2 и 3, например, 1 ВП 5 ... Р 2, 1ВП5 ... Р 3.

Задача измерения двух параметров может встретиться, например, для контроля влажности воздушной среды психометрическим методом, включающим измерение температуры «сухого» и «влажного» термометров с дальнейшим пересчетом этих данных в показатели относительной влажности среды. Такую задачу приходится часто решать в агротехнической практике, например в парниках, теплицах и овощехранилищах, наконец, при эксплуатации сушильного оборудования.

Микрокалькулятор МК-64 можно с успехом использовать для многоточечных измерений (не более семи). В этом случае семь раз будет повторяться фрагмент программы, включающий индикацию очередного измерения и код повторения эксперимента. Разумеется, что в этом случае для экономии программируемой рабочей памяти целесообразнее повторяющийся фрагмент программы реализовать в виде подпрограммы, записанной только один раз. Пример такой организации программы.

Адрес	Команда	Код команды	Содержание команды
01	F 2	22	Индикация первого параметра
02	ПП	68	Вызов под-
03	÷	36(40)	программы
04	F 3	32	Индикация второго параметра
05	ПП	68	Вызов
10	÷	36(40)	подпрограммы
11	F 4	42	Индикация третьего параметра
12	ПП	68	Вызов
13	÷	36(40)	подпрограммы
14	F 5	52	Индикация четвертого параметра
15	ПП	68	Вызов
20	÷	36(40)	подпрограммы
21	F 6	62	Индикация пятого параметра
22	ПП	68	Вызов
23	÷	36(40)	подпрограммы
24	F 7	72	Индикация шестого параметра
25	ПП	68	Вызов
30	÷	36(40)	подпрограммы
31	F 8	82	Индикация седьмого параметра
32	ПП	68	Вызов
33	÷	36(40)	подпрограммы
34	БП	58	Возврат в
35	Р 0	01	начало программы
40	В↑	06	Подпрограмма
41	ВП	66	повторения
42	1	14	эксперимента
43	0	04	
44	↔	16	
45	С/П	78	
50	В/О	48	

Результат каждого последующего измерения будет помещаться в очередной регистр памяти, начиная с регистра 2 по 8. Соответственно и первая цифра кода эксперимента изменится на 7, т. е. код будет 7100 0001.

Во всех приведенных случаях измерения могут быть организованы как по запросу, т. е. при каждом очередном восьмикратном нажатии клавиши ПУСК выполняется очередное одно измерение и его индикация, так и в режиме непрерывных циклических измерений, одно за другим с небольшим периодом (порядка 2—6 с) индикации. При организации первого режима отпадает необходимость в изготовлении блока управления калькулятором, который используется для организации только непрерывного эксперимента.

Повышение точности измерений

Для повышения достоверности и точности измерений может быть предложено программное обеспечение для выполнения многократных измерений одного и того же параметра с дальнейшим усреднением результата измерений.

Адрес	Команда	Код команды	Содержание команды
01	F 2	22	Индикация очередного измерения
02	В↑	06	
03	ВП	66	
04	1	14	
05	0	04	Код повторения
10	↔	16	эксперимента
11	С/П	78	
12	В↑	06	
13	F 8	82	Суммирование
14	+	96	результатов
15	Р 8	81	измерений
20	F 7	72	Подсчет числа
21	1	14	опросов и возв-
22	—	86	рат на очеред-
23	Р 7	71	ное изменение
24	Р Х-0	59	при незавершен-
25	Р 0	01	ном количестве опросов
30	F 8	82	Определение
31	В↑	06	средне-арифме-
32	F 6	62	тического от
33	÷	36	результата мно-
			гократных измерений
34	С/П	78	Останов

Вводим служебные коды, как и для случая контроля одного параметра — код эксперимента в регистр 9 и множитель параметра в регистр 2, а также дополнительные коды: ноль в регистр 8 и $1/n$ в регистры 6 и 7, где n — заданное число измерений, подлежащих усреднению.

Следует помнить, что при использовании последней программы перед каждой очередной се-

рией измерений с усреднением результата следует обновить дополнительные коды в регистрах 6, 7 и 8, так как информация в них меняется в процессе работы программы. Однако при желании можно несколько модифицировать программу, тогда как коды в регистрах 6, 7 и 8 будут сохраняться и не потребуется их обновления перед каждой очередной серией измерений.

Представленная программа составлена для использования в режиме запроса. Можно при необходимости ее модифицировать и для работы в режиме непрерывных циклических измерений. В этом случае для разделения циклов между ними целесообразно ввести искусственную программную задержку.

Описанная методика измерения может быть рекомендована и для решения задачи интегральной оценки параметра, подвергающегося флюктуационным изменениям во времени, часто встречающимся в исследовательской практике.

При организации измерения других физических величин возникает необходимость в разработке новых устройств преобразования для каждого конкретного датчика параметра. Изменится также коэффициент масштабирования и алгоритм расчета по переводу сигнала измеряемого параметра в его реальные единицы измерения. А остальные устройства и программы могут сохраняться без изменения.

В следующем выпуске этой рубрики читайте: программируемый таймер, поддержание заданной температуры, МК-64 управляет температурным режимом.

Литература

1. Колосов Ю. В. Цифровые регуляторы на микропроцессорной базе // Сб. трудов МТИ. — 1984. — № 54.

2. Гутников В. С. Интегральная электроника в измерительных устройствах. — Ленинград: Энергия, 1980.

3. Микрокалькулятор «Электроника МК-64»: Техническое описание и инструкция по эксплуатации.

ского на английский язык.

4. Обучающиеся компьютеры и роботы. Многие проблемы программирования могут быть преодолены, если машины смогут обучаться, как человек: на основе наблюдений, практической работы, по книгам.

Должны быть созданы компьютеры, которые могли бы, например, прочесть главу из учебника высшей школы и ответить на вопросы по этому тексту. Это потребует большого прогресса в робототехнике, машинном зрении, понимании естественного языка.

5. Самопроизводящиеся системы. Эта задача может иметь практическое применение, когда возникнет необходимость доставки на какую-то планету Солнечной системы целого предприятия. Вместо того, чтобы запускать в космос весь завод, можно отправить контейнер с деталями и оборудованием, снабженный «генетическим кодом», позволяющим производить сборку на месте. Эти системы также потребуют значительного прогресса в робототехнике, в области ИИ и так называемых «систем, способных к воспроизводству».

Sciense. — № 4872. — 1988. — 16 октября. — Т. 241. — С. 1436.

С. А. Утенков

Европа соглашается финансировать программы распознавания.

Из всего огромного потока требований на выделение средств в более признанные области европейская Комиссия ЕЭС утвердила программы, на которые будут выделены средства на втором этапе исследовательской программы по информационной технологии.

Среди них: «Сандиал» (для распознавания продолжительной речи); «Санстар» (разработка интерфейсов); «Полиглот» (преобразование речи в текст на нескольких языках); программа по распознаванию речи среди шумов, например среди потока автомобилей.

New Scientist, август 1988, с. 38.

П27 **Персональная** ЭВМ «Ямаха MSX-2». — М.: Знание, 1989. — 48 с. — (Новое в жизни, науке, технике. Сер. «Вычислительная техника и ее применение»; № 8).
ISBN 5-07-000333-X

20 к.

Знакомим читателя с популярной в нашей стране персональной ЭВМ японского производства — «Ямаха-MSX». Брошюра будет также полезна пользователям, получившим в свое распоряжение и другие типы ПЭВМ и впервые приступившим к освоению компьютера и методов программирования. Во второй статье приведен опыт применения программируемого калькулятора в системах малой автоматизации.

Материал рассчитан на широкий круг читателей.

2404000000 **ББК 32.973**

Научно-популярное издание

ПЕРСОНАЛЬНАЯ ЭВМ «ЯМАХА MSX-2»

Гл. отраслевой редактор *Л. А. Ерлыкин*
Зам. гл. отраслевого редактора *Г. Г. Карвовский*
Редактор *Б. М. Васильев*
Мл. редактор *Н. А. Васильева*
Художники *В. Н. Конюхов* и *К. Н. Мошкин*
Худож. редактор *М. А. Гусева*
Техн. редактор *А. М. Красавина*
Корректор *В. И. Гуляева*

ИБ № 9804

Сдано в набор 24.04.89. Подписано к печати 21.06.89. Т-01450. Формат бумаги 70 × $100^{1}/_{16}$. Бумага офсетная № 2. Гарнитура журнально-рубленая. Печать офсетная. Усл. печ. л. 3,90. Усл. кр.-отт. 8,45. Уч.-изд. л. 4,57. Тираж 68 221 экз. Заказ 235. Цена 20 коп. Издательство «Знание». 101835, ГСП, Москва, Центр, проезд Серова, д. 4. Индекс заказа 894708. Ордена Трудового Красного Знамени Калининский полиграфический комбинат Государственного комитета СССР по делам издательств, полиграфии и книжной торговли. 170024, г. Калинин, пр. Ленина, 5.

Цена 20 коп.

Индекс 70195

Адрес подписчика:

Пч 27 43

Издательство «Знание»

Подписная научно-популярная серия

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Изобретено в Европе, запатентовано в Америке, изготовлено в Японии.

Поговорка

Вычислительные системы для головного мозга — то же самое, что рычаг для мышц

Чарльз Лехт, президент фирмы

Ни одна армия не может уничтожить идею, время которой пришло

В. Гюго

Наш адрес: СССР, Москва, Центр, проезд Серова, 4