

Давние читатели раздела сразу заметят в его нынешнем выпуске отклонение от традиции: вслед за фамилиями авторов заметок не указано, из каких они городов. Это сделано во избежание повторов: все заметки нынешнего выпуска присланы из Киева.

Письма оттуда всегда составляют изрядную долю в почте раздела. Неудивительно: в Киеве много научно-исследовательских институтов и других учреждений компьютерного направления, и в первую очередь — Институт кибернетики АН УССР им. В. М. Глушкова. Потому то и оказалось возможным образовать хорошую подборку из писем киевлян.

Когда она была составлена, обнаружился любопытный факт: половина отобранных писем так или иначе касается преподавания информатики. Отличная примета! Если те, кто вырабатывает и хранит столь нужные сегодня знания, берут на себя и заботы об их распространении — на успех компьютеризации можно рассчитывать.

Какой город будет представлен в нашем разделе следующим! Реальные на то шансы есть у Ленинграда, Горького — помногу писем приходит и оттуда. А может быть, еще в каком-то городе читатели, не сговариваясь, разом возьмутся за перо и пришлют столько заметок, что хорошая подборка сложится из них.

ЕСЛИ В ШКОЛЕ НЕТ КОМПЬЮТЕРА

Основы информатики и вычислительной техники, или, сокращенно, ОИВТ — так называется курс, преподавание которого началось в 1986/87 учебном году в общеобразовательных школах, ПТУ, техникумах и других средних специальных учебных заведениях нашей страны. Тем самым в орбиту информатики втягивалось 4,3 миллиона старшеклассников, обучающихся в 60 790 школах. Из этих школ только в 972 (1,6%) была материальная база для изучения нового предмета.

Возникает естественный вопрос: как же изучать ОИВТ, не имея ВТ (вычислительной техники)?

Разумный ответ оказался в посылке, недавно пришедшей в редакцию из Киева. В этой посылке были пособия по курсу ОИВТ, опирающиеся на программируемый микрокалькулятор БЗ-34 и простой школьный калькулятор МШК-2. Кроме того, там был ряд номеров детского журнала «Пионе-

рия», выходящего в Киеве на украинском и русском языках, в котором с 1985 года учитель С. Т. Кузнецов и кандидат физико-математических наук В. Б. Распопов ведут раздел «Компьютер и ты». И, наконец, книга «Формирование компьютерной грамотности учащихся», изданная в 1987 году издательством «Радянська школа» на украинском языке. Все эти материалы показывают, что микрокалькулятор, в особенности программируемый, в умелых руках становится прекрасным подручным средством для плавания в океане информатики. Конечно, ЭВМ лучше, спора нет, но если нет ЭВМ, то очень много можно сделать и с помощью калькулятора.

Пособие для 9-го класса В. Б. Распопова и А. Ф. Верланя «Основы программирования на микрокалькуляторах» знакомит с программированием и простейшими понятиями этой дисциплины (присваивание, цикл, условные и безусловные переходы

УДАЧУ МОЖНО СПРОЕКТИРОВАТЬ

По программе кружка информатики, который я вел для школьников, настало время перейти к вопросу о том, как разрабатывать программы.

Для изучения этого вопроса, предложил я ребятам, возьмем компьютерный фокус В. Пинаева («Наука и жизнь» № 12, 1987 г.). Вот его суть. Зритель задумывает и называет вслух целое трехзначное число, которое ЭВМ отгадывает за несколько попыток. Каждая попытка состоит в том, что компьютер сообщает очередную свою догадку и, если она неверна, зритель называет какое-нибудь слово. Фокусник набирает это слово на клавиатуре, и все повторяется сначала. С каждым разом догадки машины все ближе к задуманному числу, и вот наконец оно отгадано.

Секрет фокуса в том, что вместе со словом фокусник вводит тайную подсказку компьютеру. Пусть она будет такой: если вводимое слово содержит четное количество букв, это означает, что задуманное число больше выданного машиной, если нечетное — меньше. Четностью количества букв в слове фокусник может легко управлять, добавляя при необходимости в начале слова пробел.

В качестве первого шага к решению нашей задачи давайте четко сформулируем ее условие — поставим себе техническое задание. Это желательно делать всегда, даже если вы пишете программу только для личного пользования, поскольку помогает понять все тонкости выполняемой работы.

Итак, требуется написать программу, которая:

1. Выводит некоторое трехзначное число и спрашивает, угадала ли она.
2. Если ответ отрицательный, то про-

Введя эту программу в микрокалькулятор БЗ-34, нажмите клавиши В/О С/П. На индикаторе появится ряд единиц и среди них — одна восьмерка, которая станет быстро перемещаться от одного края индикатора до другого и обратно. Когда она подойдет к одному из краев, попробуйте остановить ее точно у края нажатием клавиши С/П. Так вы проверите свою реакцию.

и т. д.). Читателю пособия предоставляется выгодная возможность работать самостоятельно. Для этого есть шесть лабораторных работ. Некоторые из них являются миниатюрными исследованиями — такова, например, работа по определению быстродействия микрокалькулятора.

Второе пособие для 9-го класса «Основы информатики и вычислительной техники» (авторы А. Ф. Верлань и В. Н. Касаткин) продолжает цепочку основных понятий: информация, двоичное кодирование, алгоритм и алгоритмические языки. Эти же авторы под тем же названием разработали пособие для 10-го класса, по которому учащиеся знакомятся с двоичной арифметикой, математической логикой и языком программирования Бейсик. Наконец, следующее пособие для 10-го класса, «Основы применения вычислительной техники», написанное А. Ф. Верланем и В. Б. Распоповым, проводит аналогию между языком программируемого калькулятора и языком Бейсик. Ряд программ написан параллельно на обоих языках, а это дает возможность прочувствовать конструкции уже известные, но написанные на другом языке, оценить особенности программирования на сравниваемых языках.

Книга «Формирование компьютерной грамотности учащихся» содержит 17 статей учителей-практиков, использующих микрокалькулятор на уроках начиная с 4-го класса и во внеклассной работе.

Конечно, небольшая посылка не смогла бы вместить все книги, выпущенные центральными издательствами Украины в помощь школьным преподавателям новой

грамма просит ввести слово и выдает новое трехзначное число-догадку, большее предыдущего, если количество букв в слове четное, и меньшее, если оно нечетное. Затем она переходит к пункту 1.

Теперь перейдем к составлению алгоритма или, как говорят, программисты, проектированию программы. При этом будем пользоваться подобием алгоритмического языка, разработанного А. П. Ершовым и примененного в школьной информатике.

Если бы у нас была машина, способная самостоятельно угадывать трехзначное число, то все, что требовалось бы от нас, — это написать

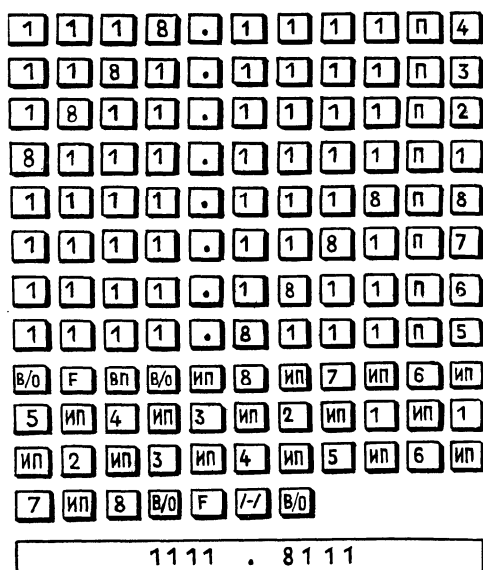
нач

угадать число

кон

К сожалению, такой машины у нас нет, и нам придется уточнить, что значит «угадать число».

Как в описании фокуса, так и в нашем техническом задании чувствуется явная цикличность действий программы: показать число, спросить слово, показать число,



дисциплины. Тем более что добрую половину посылки составил комплект журнала «Пионерия». В разделе «Компьютер и ты» здесь так же, как и в упомянутых книгах, речь идет о программируемом микрокалькуляторе, наиболее распространенном из вычислительных средств и наиболее доступном сегодня. Четкие рисунки программ (один из них представлен здесь) помогают избежать ошибок при вводе. Есть в разделе и познавательные задачи, и игры. По поводу игр можно лишь пожелать, чтобы исход в них не всегда был предreshen в пользу микрокалькулятора. Как только играющий обнаружит, что ему отведена роль статиста, вводящего в микрокалькулятор ходы и терпеливо ждущего собственного поражения, — интерес к игре пропадает...

спросить слово... Поэтому вполне естественно при уточнении воспользоваться циклом. Напрашивается и условие, при котором цикл нужно повторять — конечно же, когда число не угадано. Кроме того, опыт говорит, что перед циклом, как правило, приходится произвести некоторые действия, например, задать начальные значения каким-либо величинам, а после цикла — заключительные действия. Так что мы вполне можем записать в расширенном виде:

нач

предисловие

пока число не угадано **повторять**

определить следующее число-догадку

все

послесловие

кон

Что касается предисловия и послесловия, то по самой их сути они могут быть уточнены только после того, как будет составлена вся программа. Поэтому займемся вплотную определением числа-догадки.

СЕКРЕТ УСПЕХА - В ХОРОШЕМ АЛГОРИТМЕ

Когда составлен алгоритм решения задачи, написать по нему программу обычно не составляет труда. В частности, поэтому задачи первого тура олимпиад по школьной информатике, проводимых с 1986 года Институтом кибернетики АН УССР, имеют своей целью именно составление алгоритмов.

Первый тур олимпиады—заочный. В школы рассылаются задачи, примерно через три недели школьники присылают решения, авторы лучших решений приглашаются в Институт кибернетики на второй, очный тур, где от них требуется составление уже не только алгоритмов, но и основанных на них программ. Все эти программы по очереди запускаются на компьютере, результаты их

работы отображаются на дисплее, школьники в это время становятся зрителями и болельщиками, а зрелище получается весьма увлекательнее!

При подборе задач как отборочного, так и финального тура мы стремимся в каждую из них вложить какой-нибудь метод программирования или общий подход, чтобы школьник в поисках решения сам открыл этот метод или подход. Разбор такой задачи не просто пересказ решения, а маленькая лекция о том, где возникают подобные задачи и где применяются подобные методы. Это придает олимпиадам познавательную роль.

Задачи стараемся придумывать сами, но иногда обращаемся к фольклорным источникам — впрочем, точнее было бы сказать,

Наше техническое задание подсказывает нам, каким образом это можно сделать: определить следующее число-догадку == == == **показать число-догадку** **если** число не угадано **то** уточнить число

все

Удобный способ уточнения заключается в том, что сначала нужно выбрать интервал, в котором искомое число находится наверняка, а затем в качестве очередной догадки выдавать середину этого интервала. Если искомое число меньше нашей догадки, то интервал поиска заменяется своей левой половиной, если больше, то правой — и действия повторяются. Таким образом, интервал поиска каждый раз сокращается в два раза, покуда не сузится до одного числа — искомого.

В соответствии с только что сказанным определим, что значит «уточнить число»:

уточнить число == == == уточнить интервал
выбрать число-догадку в интервале А вся наша программа будет выглядеть: **нач** предисловие

пока число не угадано **повторять**

показать число-догадку

если число не угадано

то уточнить интервал

выбрать число-догадку в интервале

все

все

послесловие

кон

Обозначим концы интервала поиска А и В, а число-догадку С. Кроме того, пусть ответ человека на вопрос, угадано ли число, выражается значениями текстовой переменной «ответ». Пусть ее значениями будут буквы «д» или «н», соответственно кодирующие слова «да» и «нет». Теперь мы можем сказать, что фрагмент «уточнить интервал» устанавливает либо А, либо В равным С, а фрагмент «выбрать число-догадку в интервале» устанавливает С равным середине интервала (А, В).

Очевидно, что в предисловии должны

устанавливаться начальные значения переменных А, В, С и «ответ», а в послесловии нужды вообще нет.

Теперь уже можно записать окончательный проект программы:

нач

А := 99

В := 1000

С := 500

ответ := «н»

пока ответ = «н» **повторять**

показать С

спросить ответ

если ответ = «н»

то уточнить интервал (**изм** А, **изм**

В, **вх** С)

выбрать число в интервале (**вх** А, **вх** В, **изм** С)

все

все

кон

Вслед за заголовками «уточнить интервал» и «выбрать число в интервале» появились названия тех величин, которые будут использоваться в соответствующих фрагментах. Причем если в некотором фрагменте величина изменяется, она помечена **изм**, а если нет — **вх** (от слова «входная»). Распишем оба фрагмента более подробно: уточнить интервал (**изм** А, **изм** В, **вх** С)

нач

спросить слово

если в слове четное количество букв

то А := С

иначе В := С

все

кон

По сути, это дословная запись того, что мы уже обсуждали. Что касается фрагмента «выбрать число», то с ним еще проще: выбрать число в интервале (**вх** А, **вх** В, **изм** С)

нач

С := (А + В) / 2

кон

Здесь / означает деление нацело.

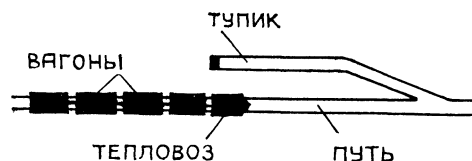
Теперь, казалось бы, можно вставить эти

что их влияние ощущается во всех предлагаемых ниже задачах.

В. БАРДАДЫМ.

СОРТИРОВКА ВАГОНОВ

На станцию, изображенную на рисунке, прибывает поезд. Его вагоны пронумерованы в произвольном порядке. Требуется составить алгоритм, располагающий вагоны в порядке убывания номеров. В тупик помещается только один вагон или тепловоз. Предполагается, что диспетчер, управляющий маневрами, видит номера всех вагонов поезда. Разрешается передвигать тепловозом вперед и назад любое количество вагонов, расцеплять и сцеплять соседние вагоны и тепловоз с вагоном, переводить стрелку. Затем требуется решить ту же задачу, предполагая, что диспетчер видит номера только двух ближайших к стрелке вагонов



и вагона в тупике. Номера остальных вагонов диспетчер не запоминает.

ЗАБОР ВОКРУГ РОЩИ

Рощу ценных деревьев следует оградить забором, имеющим вид замкнутой линии. Известны координаты всех деревьев. Описать алгоритм для ЭВМ с графопостроителем, рисующий план рощи с забором минимальной длины. Деревья изображаются точками. Забор может проходить вплотную к деревьям. Графопостроитель может выполнять команды НАНЕСИТОЧКУ (X, Y)

части на их законные места в «главной» программе. Но лучше этого не делать по нескольким причинам. Во-первых, главная программа и без того получилась достаточно сложной, и добавление к ней еще десятка строк только уменьшит ее понятность, а значит — затруднит внесение изменений при отладке и последующих улучшениях. Во-вторых, каждая из выделенных нами частей выполняет четко определенную функцию, и если потом мы захотим изменить алгоритм, по которому эта функция выполняется, у нас будет уверенность, что переделывать придется не всю программу, а только ее часть. Договоримся, что в будущей программе оба фрагмента выразятся в виде подпрограмм.

Составление проекта закончено. Давайте еще раз повторим основные этапы проделанной нами работы.

Во-первых, мы постарались четко определить задачу, которую будет решать наша программа (программисты говорят «составили спецификацию»). Заметим: для больших программных систем спецификация едва вмещается в несколько объемистых томов, и процесс ее написания заслуженно считается одним из самых сложных и чреватых ошибками этапов в разработке программы.

Во-вторых, мы применили так называемое нисходящее проектирование (или проектирование по принципу «сверху — вниз»). Сначала мы определили проект будущей программы в самых общих чертах, не обращая внимания на то, что отдельные его части в том виде, в каком они записаны, вообще не могут быть выполнены на ЭВМ. Затем мы постепенно уточняли все такие части, в свою очередь разбивая их на более конкретные и понятные машине, пока не получили проект, легко реализуемый на любом распространенном языке программирования.

Попробуйте по этому проекту написать задуманную нами программу на Бейсике и сравните ее с приведенным здесь вариантом. При сравнении постарайтесь вникнуть

```

100 REM ПРОГРАММА УГАДЫВАНИЯ ЧИСЛА
110 REM (СМ. "НАУКА И ЖИЗНЬ", 12, 1987)
120 REM
130 PRINT "*****"
140 PRINT "*"
150 PRINT "* ФОКУС-ПОКУС *"
160 PRINT "*"
170 PRINT "*****"
180 PRINT
190 A%=99
200 B%=1000
210 C%=500
220 ANSW="Н"
230 REM
240 REM ОСНОВНОЙ ЦИКЛ
250 REM
260 IF ANSW<>"Н" THEN 350
270 PRINT "ЭТО";C%"; УГАДАЛА? [Д/Н]";
280 LINE INPUT ANSW
290 IF ANSW<>"Н" THEN 340
300 REM УТОЧНИТЬ ИНТЕРВАЛ
310 GOSUB 500
320 REM ВЫБРАТЬ ЧИСЛО
330 GOSUB 800
340 GOTO 260
350 REM
360 STOP
370 REM
500 REM
510 REM ПОДПРОГРАММА "УТОЧНИТЬ ИНТЕРВАЛ"
520 REM ИЗМЕНЯЕТ А ИЛИ В В ЗАВИСИМОСТИ ОТ
530 REM УКАЗАНИЙ ФОКУСНИКА - ЧЕТНОСТИ
540 REM ДЛИНЫ СЛОВА. СЛОВО ОБОЗНАЧЕНО
550 REM WORD#
560 REM
570 PRINT "ТОГДА НАЗОВИТЕ ЛЮБОЕ СЛОВО ";
580 LINE INPUT WORD#
590 IF (LEN(WORD#) MOD 2) > 0 THEN 620
600 A%=C%
610 GOTO 640
620 REM ИНАЧЕ
630 B%=C%
640 RETURN
800 REM
810 REM ПОДПРОГРАММА "ВЫБРАТЬ ЧИСЛО".
820 REM ИЗМЕНЯЕТ С, УСТАНАВЛИВАЯ ЕГО
830 REM ПОСЕРЕДИНЕ ИНТЕРВАЛА (А, В)
840 REM
850 C%=(A%+B%)/2
860 RETURN
1000 END

```

во все тонкости. Например, в подпрограмме «уточнить интервал» лишний пробел и знак «;» в конце оператора PRINT помогают замаскировать пробел, вводимый фокусником в начале некоторых слов.

Когда эта программа была введена в компьютер, она заработала с первого пуска. И это было закономерным результатом ее разработки «сверху — вниз».

И. РОМАНЕНКО

и НАЧЕРТИТЬ ОТРЕЗОК (X1, Y1, X2, Y2), где (X,Y) — координаты точки, (X1, Y1) и (X2, Y2) — координаты концов отрезка.

ПОИСК ПОДПОСЛЕДОВАТЕЛЬНОСТИ

Дана последовательность N чисел. Ее подпоследовательностью будем называть любое подмножество множества ее членов, расположенных в порядке возрастания их индексов.

Составить алгоритм, вычисляющий для данной конечной последовательности различных чисел ее монотонно убывающую подпоследовательность наибольшей длины.

ЕСТЬ ЛИ СВЯЗЬ?

В Антарктиде расположены N метеорологических станций. Каждая станция соединена со всеми другими станциями линиями связи. Из-за стихийного бедствия многие линии связи оказались нарушенными. Написать алгоритм, определяющий, между

какими парами станций связь невозможна даже через цепочки других станций. Исправность линии связи между 1-й и K-й станциями можно выяснять с помощью логической функции ЕСТЬ СВЯЗЬ (1, K).

СЧИТАЕМ ВОРОН*

Написать программу, которая получает на входе число N и печатает на выходе фразу «N ворон», записывая числительные словами. $0 \leq N \leq 9999$. Примеры:

Вход	Выход
1	одна ворона
3	три вороны
8	восемь ворон
55	пятьдесят пять ворон
200	двести ворон
1234	одна тысяча двести
_____	тридцать четыре вороны

* (Автор задачи — Ю. Матиясевич).

КОМУ НУЖНА АМБАРНАЯ КНИГА?

В 1986 году наш раздел получил новое название «Человек и компьютер». Связано это было с тем, что в нем наряду с программами для микрокалькуляторов стали публиковаться программы для персональных компьютеров. И первая из таких программ под названием «Амбарная книга» поступила в обновленный раздел с киевским обратным адресом — см. заметку В. Соловьева «Как я заведовал складом химреактивов» («Наука и жизнь» № 2, 1986 г, стр. 106—107).

После публикации той программы прошло три года, а отклики на нее продолжают поступать. Читатели (чем она их приворожила?!) все улучшают и развивают ее, дополняя новыми сервисными участками. Теперь с ее помощью можно управлять данными не только по складу, но и по другим объектам. Вот как, например, могут выглядеть записи по отделу кадров:

СОЛОВЬЕВ В. РОД. 48
10.23, ПОСТ. НА РАБОТУ
71.02.17, ВЫГОВОР ЗА
ОПОЗД. 87.06.04.

Разбор программы, по мнению В. Чистякова из Москвы, А. Ярвица из Свердловска и других читателей, стал неплохим уроком программирования. В письмах пришли аналоги программы «Амбарная книга», написанные для других ЭВМ: Роботрон 1715 (И. Щербин-ская, г. Ленинград), СМ-4 (В. Кузнецов, г. Рига), ДВК-2М (А. Инушкин, г. Смоленск). Подробный разбор программы с указанием основных путей ее совершенствования сделал К. Александров из Ленинграда.

Учтя основные замечания читателей, выставляем на их суд переработанную программу «Амбарная книга» (версия для «Искры-226»). Ее особенности: 1. Хранение записей в упакованном виде на магнитном диске. 2. Использование оператора LINPUT. 3. Поиск как по одному, так и по двум ключевым словам.

Предлагаем читателям так доработать программу, чтобы по ней можно было вести поиск с любой комбинацией ключевых слов, применить более удобную форму записи массива на диск, более быстрый алгоритм поиска информации в книге и ее упаковки.

```

10 REM ПРОГРАММА "АМБАРНАЯ КНИГА" В.СОЛОВЬЕВ + НАУКА И ЖИЗНЬ
20 N=500:DIM N$(500)100,N#100:A$="СТРАНИЦЫ":R=0
30 DATA LOAD DC OPEN F A$:REM ОТКРЫТИЕ ФАЙЛА ДЛЯ СЧИТЫВАНИЯ
40 FOR I=1 TO N:DATA LOAD DC N$(I):IF END THEN 50:NEXT I
50 PRINT HEX(03);"В КНИГЕ ";I-1;" ЗАПОЛНЕН.СТРАНИЦ":GOTO 70
60 PRINT HEX(03);REM ОЧИСТКА ЭКРАНА ДИСПЛЕЯ
70 INPUT "1-НОВАЯ ЗАПИСЬ,2-ПРОСМОТР,3-ПОИСК ПО КЛЮЧЕВЫМ
СЛОВАМ,4-STOP":B:ON B GOTO 80,130,100,220
80 FOR I=1 TO N:IF N$(I)=" "THEN 90:NEXT I:PRINT "КНИГА
ПЕРЕПОЛНЕНА":GOTO 60
90 PRINT I;"-Я СТРАНИЦА":LINPUT N$(I):GOTO 60
100 INPUT "ПЕРВОЕ КЛЮЧЕВОЕ СЛОВО",S$(1)
110 S$(2)=" ":INPUT "2-Е СЛОВО",S$(2):IF S$(2)=" "THEN 130
120 INPUT "ЛОГИЧЕСКАЯ СВЯЗКА (И,ИЛИ,НЕ)",L$
130 FOR I=1 TO N:N#N$(I):IF N#=" "THEN 210:IF B=2 THEN 200
140 FOR J=1 TO 2:P(J)=0:IF S$(J)>" "THEN 150:P=P(1):GOTO 190
150 L=LEN(N#)-LEN(S$(J))+1:FOR Z=1 TO L:
IF S$(J)=STR(N#,Z,LEN(S$(J))) THEN 160:NEXT Z:GOTO 170
160 P(J)=1:REM J-Е СЛОВО НАЙДЕНО НА I-ОЙ СТРАНИЦЕ
170 NEXT J:REM СЛЕДУЮЩЕЕ КЛЮЧЕВОЕ СЛОВО
180 P=P(1)*P(2):IF L$="И"THEN 190:REM ЛОГИЧЕСКОЕ УМНОЖЕНИЕ:
P=P(1)+P(2):IF L$="ИЛИ"THEN 190:REM ЛОГИЧЕСКОЕ СЛОЖЕНИЕ:
P=P(1)-P(2):IF L$<>"НЕ"THEN 120
190 IF P<1 THEN 210:REM ЭТУ СТРАНИЦУ ПРОПУСТИТЬ
200 PRINT I;"-Я СТР.":LINPUT N$(I):IF N#N$(I) THEN 210:R=1
210 NEXT I:GOTO 60
220 IF R=0 THEN 290:REM В КНИГЕ НЕТ ИЗМЕНЕНИЙ
230 FOR I=1 TO N-1:IF N$(I)>" " OR N$(I+1)=" "THEN 250
240 N$(I)=N$(I+1):N$(I+1)=" ":GOTO 230:REM УПАКОВКА КНИГИ
250 NEXT I
260 SCRATCH F A$:DATA SAVE DC OPEN F (A$):REM ПЕРЕЗАПИСЬ
270 FOR I=1 TO N:IF N$(I)=" "THEN 280:DATA SAVE DC N$(I):NEXT I
280 DATA SAVE DC END:REM ЗАКРЫТИЕ ФАЙЛА ПОСЛЕ ЗАПИСИ
290 END

```

1024 СОВЕТА

(ПЯТЫЙ, ШЕСТОЙ И СЕДЬМОЙ БАЙТЫ С ИЛЛЮСТРАЦИЕЙ)

Мне нравится рубрика «1024» совета», недавно появившаяся в разделе «Человек и компьютер». Хочу предложить для нее сразу три байта. Все они, сколь бы странным это ни казалось, навеяны одной программой — помните заметку А. Лебедева, где речь шла о поиске места в микрорайоне для встроенного магазина? Заметка называлась «Что нам стоит дом построить» («Наука и жизнь» № 6, 1987 г., стр. 134 — 135). Программу я переработал и, на мой взгляд, улучшил.

33. Расшифровывай в программе смысл редких операторов и операторов, имеющих разное написание в разных диалектах языка (см. строку 1, начало строки 4 и конец строки 17).

34. Оформляй название программы не ремаркой, а оператором печати. Это позволит распознавать программу и по ее листингу, и при ее прогонке (см. строку 2).

35. Старайся избегать приемов программирования и операторов, допустимых не во всех диалектах языка. Например, оператор ввода с клавиатуры с комментарием (см. строки 3, 7 и 8) разбей на два: оператор печати и оператор ввода; не обозначай одним и тем же именем массив и простую переменную (см. строку 11) и т. д.

36. Резервируй в памяти ЭВМ место под массивы с некоторым запасом (см. строку 4). Это даст возможность при необходимости в непосредственном режиме увеличить число элементов массивов, ввести недостающие параметры и повторить расчет, запустив программу не с самого начала, исправив пропуск некоторых параметров в начальном диалоге.

37. Отводи под вещественные по сути величины целочисленные по форме переменные. Так, заменяя километры на метры (см. строку 7), рубли на копейки, тонны на граммы и т. д., можно избежать десятичных запятых и существенно сэкономить память ЭВМ.

38. Помни, что использование целочисленных переменных экономит память ЭВМ лишь в том случае, если эти переменные объединены в массивы. Имей в виду, что большинство Бейсик-трансляторов имеют однуарифметику и для вещественных, и для целочисленных переменных. Простые целочисленные переменные не ускоряют прогонку программы, а только засоряют память и листинг знаками процента.

39. Давай переменным «говорящие», но не очень длинные имена: OPT —

оптимальный, MIN — минимальный (см. строку 5).

40. Старайся не пользоваться принципом умолчания или по крайней мере не злоупотребляй им. Этот принцип по-разному толкуется в разных диалектах Бейсика, а в таких ортодоксальных языках, как Паскаль, его почти нет. Поэтому объявляй переменные в ремарках (см. строку 5), указывая шаг параметра цикла (см. строки 6 и 19), помещай конец программы (см. строку 24) и т. д.

41. Используй на Бейсике удачные находки других языков программирования. Так, в языке Си заголовок цикла с параметром может содержать операторы обнуления сумматоров, операторы присваивания. Использование таких приемов в других языках (см. строки 10, 12) не только делает программу более компактной, но и исключает некоторые ошибки: повторный вход в цикл без обнуления сумматора, например.

42. Проводя чистку цикла (см. строку 11), помни, что уже давно появились так называемые оптимизирующие трансляторы, берущие такую черновую работу на себя, оставляя человеку простор для творчества. Избегай чрезмерного усердия, не забывай о ясности программы (см. также совет 11 в № 2 за 1988 г., стр. 133).

43. Заменяя для ускорения счета возведение в целую степень на умножение, помни, что такая операция может иметь обратный эффект, если переменные индексные, да к тому же вставленные в выражения (см. строку 13).

44. При формировании текстовых сообщений не забывай о падежах и других грамматических формах (см. конец строки 15).

45. Оформляй ремарками концы структурных блоков (см. строки 16, 21 и 22) во избежание каких бы то ни было двусмысленностей при анализе программы.

46. При формировании таблиц на дисплее и на бумаге используй специальные зоны, попасть в которые по могают запятые в операторах печати (см. строки 18 и 20). Это избавит от использования табуляторов и форматов печати.

47. Если в трансляторе нет слова ELSE (иначе), то смело меняй полную альтернативу на две неполные (см. строки 21 и 22). При этом не придется использовать метку, которую многие считают признаком низкой культуры программирования (автор совета — Д. Ван Тассел).

48. Отмечай ремаркой малозаметные для человека особенности программы, пренебрежение которыми может сильно исказить результат (см. строку 20).

49. Не забудь указать в программе ее имя и внешние ресурсы, где она хранится (см. строки 23 и 24).

50. Пустые места в коротких строках заполняй дополнительными комментариями (см. строки 6, 9, 12, 14, 23 и 24).

51. При выводе результата не забудь распечатать и исходные данные (см. строки 18—23). Это, во-первых, позволит оформить полный протокол расчета, а во-вторых, призовет к дополнительному контролю исходных данных.

52. При первой прогонке программы давай ей для пробы простейшие примеры, решение которых заранее известно. Для нашей программы в качестве первого теста можно предложить пять равнозаселенных домов, расположенных в углах и внутри квадрата. Подобным приемом можно быстро выявить синтаксические и грубые смысловые ошибки.

53. Отделяй в листинге программы пробелами операторы на строке, служебные слова и переменные. Это повысит наглядность программы.

54. Размещение нескольких операторов на строке делает программу компактной. Она сможет полностью уместиться на экране дис-

```
1 CLS 'ГАСИЕНИЕ ЭКРАНА ДИСПЛЕЯ, КУРСОР СЛЕВА ВВЕРХУ
2 ? ' ПОИСК ДОМА В МИКРОРАЙОНЕ ДЛЯ ВСТРОЕННОГО МАГАЗИНА'
3 ? : ? ' ЧИСЛО ДОМОВ В МИКРОРАЙОНЕ' : INPUT N
4 BASE 1 : REM 1-МИНИМ.ИНДЕКС : DIM XX(N+5), YX(N+5), GX(N+5)
5 'J, J_OPT, K, N-CARDINAL (INTEGER, NO >0); S, S_MIN, X0, Y0-REAL
6 FOR J=1 TO N STEP 1 'ВВОД ИСХОДНЫХ ДАННЫХ
7 : ?J'-Й.ДОМ' : ?'X,Y В МЕТРАХ' : INPUT XX(J), YX(J)
8 : ?'ЧИСЛО ЕДОВОК В НЕМ' : INPUT GX(J)
9 NEXT J 'ДАЛЕЕ РАСЧЕТ ПРЯМЫМ ПЕРЕБОРОМ
10 S_MIN=1E99 : FOR J=1 TO N STEP 1 'ДОМА-КАНДИДАТЫ
11 : X0=XX(J) : Y0=YX(J) 'ЧИСТКА ЦИКЛА
12 : S=0 : FOR K=1 TO N STEP 1 'ПОДСЧЕТ ЦЕЛЕВОЙ ФУНКЦИИ
13 : S=S+GX(K)*SQR((X0-XX(K))^2+(Y0-YX(K))^2)
14 : NEXT K 'ДАЛЕЕ ВЫВОД ПРОМЕЖУТОЧНОГО РЕЗУЛЬТАТА
15 : ?'ОТ'J'-О ДОМА СУММАРНОЕ РАССТОЯНИЕ' S'ЧЕЛОВЕКОМЕТР(ОВ)'
16 : IF S<S_MIN THEN S_MIN=S : J_OPT=J 'END IF
17 NEXT J : PLAY' CDEFGAB' 'МЕЛОДИЯ ДО-РЕ-МИ-ФА-СОЛЬ-ЛЯ-СИ
18 CLS : ?'ПАРАМЕТРЫ ДОМОВ' : ?J', 'X', 'Y', 'G', 'МАГАЗИН'
19 FOR J=1 TO N STEP 1 'ЦИКЛ ВЫВОДА РЕЗУЛЬТАТА
20 : ?J, XX(J), YX(J), GX(J), 'НЕ ЗАБУДЬ ПОСЛЕДНЮЮ ЗАПЯТУЮ
21 : IF J=J_OPT THEN ?'ЕСТЬ' : REM END IF
22 : IF NOT(J=J_OPT) THEN ?'НЕТ' : REM END IF
23 NEXT J 'ПРОГРАММА НАПИСАНА 88.07.12, ХРАНИТСЯ
24 END 'НА ДИСКЕ N 2 ПОД ИМЕНЕМ 'LEBEDEV'
```

плея, что упростит ее разбор отладку и модернизацию. Если ты к тому же планируешь программу опубликовать, то ее компактность повысит шансы появления на страницах журналов, испытывающих дефицит бумаги.

55. Нумеруй строки своей программы, несмотря на то, что

наиболее совершенные версии Бейсика (Турбо-Бейсик, например) в номерах строк не нуждаются. Номера строк пригодятся при описании программ.

56. Меняя число пробелов перед первым оператором строки, можно выделить структуру программы «паскалевским»

манером (см. строки 7 и 8, 11 — 16 и 20 — 22). Если при трансляции экономная машина их выкидывает, то обмануть ЭВМ в ряде случаев удастся, поставив после номера строки знак — разделитель операторов.

М. ПЕРЕПЕЛИЦА

ПЕЩЕРА

После ввода программы этой игры в калькулятор МК-61 перед первой партией в регистр 3 вводится результат действий $9 \div 1/x$, а в регистр 7 — любое число меньше единицы, с несколькими знаками после запятой. Перед началом каждой партии следует набрать 7 F 10^x П1 Сх П6. Для начала игры нажимаем клавиши В/О С/П. Через 15 секунд на индикаторе появляется игровое поле: 8. С—Е (буквы С может не быть, партии непохожи друг на друга). Это пещера, в которой находится клад, обозначенный цифрой 8. Знаком минус обозначено место игрока, стремящегося добыть клад, буквой Е— чудовище ЕГГОГ, которое гонится за игроком, чтобы сожрать его. С — это символ двери, которую надо открыть.

Еще раз нажимаем С/П. Если двери перед игроком не было, то он передвинется на одну позицию влево, а если была, то игроку кратковременно показываются два целых числа с количеством цифр не более четырех. Перед каждым числом на индикаторе мимолетно вспыхивает частотол из единиц: 1,111111 1. Эта же комбинация будет на индикаторе после остановки — как сигнал к дальнейшим действиям.

Показанные числа надо запомнить и после остановки, когда на индикаторе видна шеренга единиц, ввести первое из этих чисел, затем нажать С/П. Если введенное число совпадает с показанным, то на индикаторе появится ноль. Далее следует ввести второе число и нажать С/П. Если и теперь все правильно, то на появившемся игровом поле знак дверей С

будет заменен цифрой 7 — дверь открыта. Предполагается, что при этом игрок стоит в дверном проеме; оттого знака минус в этой картине нет. За каждую удачно открытую дверь игроку начисляется три очка. Если хотя бы одно число введено неверно, то появляется игровое поле, в котором игрок и дверь остались на месте, а чудовище продвинулось на одну позицию влево. За неудачную попытку с игрока снимается три очка.

Так игра продолжается до конца. Если чудовище догнало игрока, то оно его пожирает. На индикаторе — ЕГГОГ, что говорит о том, что игра проиграна. Если игрок добрался до клада, то к набранной им сумме очков прибавляется 8 и итоговая сумма показывается на индикаторе как знак победы. Впрочем, если неудачных попыток было много, то сумма будет отрицательной (как видно, кладоискательство может быть и убыточным предприятием!).

Для владельцев МК-52 предлагается дополнение. После взятия клада из ППЗУ в программу записывается другой фрагмент по адресам 00-27. При этом сохраняется обстановка, предшествующая взятию клада. В этом варианте, как и в первом, необходимо преодолевать двери, открывая их тем же способом. Но после потери клада жизнь чудовищу не мила, и если игрок ошибается,

то чудовище делает шаг к нему навстречу. После перезаписи из ППЗУ игра начинается командой В/О С/П.

ОСНОВНАЯ ПРОГРАММА.
 00.Сх 01.ХП5 02.1т 3.ХП4 04.6
 05.ХПО 06.ПП 07.87 08.2 09.: 10.
 К{x} 11.ХП2 12.6 13.Х 14.ПП
 15.1 16.+ 17.F10^x 18.Х 19.ПХ1
 20.+ 21.ХП1 22.ПХ5 23.ПХ4
 24.— 25.F/x 26.ПХ1 27.ПХ4
 28.F10^x 29.5 30.Х 31.+ 32.ПХ5
 33.F10^x 34.+ 35.Кинв 36.ПХ6
 37.↔ 38.С/П 39.ПХ2 40.Fx<>0
 41.72 42.ПП 43.87 44.ХП8 45.ПП
 46.87 47.ХП9 48.ПХ3 49.B↑ 50.B↑
 51.ПХ8 52.B↑ 53.ПХ3 54.B↑
 55.ПХ9 56.B↑ 57.ПХ3 58.С/П
 59.ПХ8 60.— 61.Fx=0 62.80
 63.С/П 64.ПХ9 65.— 66.Fx = 0
 67.80 68.ПХ6 69.3 70.+ 71.ХП6
 72.КПХ4 73.FLO 74.06. 75.ПХ6
 76.8 77.+ 78.ХП6 79.С/П 80.ПХ6
 81.3 82.— 83.ХП6 84.КПХ5
 85.БП 86.22 87.ПХ7 88.Fт 89.+
 90.Fe^x 91.К{x} 92.ХП7 93.ВП
 94.5 95.К[x] 96.В/0.

СМЕННЫЙ БЛОК ДЛЯ ВТОРОЙ ЧАСТИ ИГРЫ. 00.7
 01.ХПО 02.1 03.ВП 04.6 05.ХПА
 06.ПХ1 07.ПХ0 08.F10^x 09.:
 10.ПХА 11.+ 12.К{x} 13.ХП2
 14.ПХО 15.ПХ5 16.— 17.БП
 18.24 19.Кноп 20.Кноп 21.Кноп
 22.БП 23.14 24.Fx<>0 25.75
 26.ПХ1 27.ПХ0 С адреса 28
 продолжается основная программа.

Е. БРЕДНЯ
 (ученик 10-го класса школы № 252).

• МАЛЕНЬКИЕ ХИТРОСТИ

При логических операциях в появляющемся на индикаторе числе иногда нет символа в последнем или нескольких последних разрядах. Для выяснения того, какой же знак там стоит — ноль, который не индицируется, если он стоит на последнем месте в дробной части числа, или 16-ричная цифра F, индицируемая отсутствием символа, исходное число необходимо логически умножить на 8,9999999. Если на месте отсутствующих цифр ничего не появится, то, значит, в исходном числе там стоял ноль ($0 \wedge 9 = 0$), а если появится цифра 9, то в исходном числе была цифра F ($F \wedge 9 = 9$).

К. ТУРКИН.