ISSN 0234—0453.

ИНФО

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

3-4 '92

Всем руководителям школ, ПТУ, техникумов, учителям и методистам

Научно-техническое предприятие ***«Альтернатива»*** НИИ ИВТ АПН поможет вам резко поднять методический и технический уровень использования отечественных терминальных классов.

Мы предлагаем комплексные учебные пакеты программ

для классов КУВТ-86 (любого типа):

1. Широко известный учебный пакет по информатике ***«Альтернатива-1»***. Включает универсальную систему пересылки по сети, текстовые, графические и музыкальные редакторы, систему управления базами данных, электронные таблицы, клавиатурные тренажеры, исполнители, графический конструктор, пакет игровых программ.

Учебный пакет по информатике ***«Альтернатива-2»***. Включает системы программирования ассемблер, Форт, Си, Турбо-Паскаль, Турбо-Пролог.

3. Систему AUTOR. Это простая в изучении система, предназначенная для создания учебных и демонстрационных программ в кодах БК-0010.

4. Комплекты из 12 манипуляторов «джойстик», совместимых с нашими программами. Их использование резко повышает удобство работы с программами и снижает износ клавиатуры.

для классов УКНЦ («Электроника 0511»):

Пакет ***«Альтернатива-УКНЦ»***, который включает:

пересылаемую по сети в течение 10—15 с операционную систему RT-11, обеспечивающую быструю пересылку и работу в РМУ любых программ центральной машины;

систему программирования Турбо-Паскаль, транслирующую в коды, с графическими возможностями;

текстовые и музыкальные редакторы, систему управления базами данных, Турбо-Пролог, пакет игровых программ:

для классов «Корвет» (любых типов и модификаций):

Пакет ***«Альтернатива-Корвет»***, который включает:

пересылаемую по сети в течение 10—15 с операционную систему CP/M, обеспечивающую быструю пересылку и работу в РМУ любых программ центральной машины;

систему программирования Турбо-Паскаль, транслирующую в коды;

текстовые и музыкальные редакторы, систему управления базами данных, пакет игровых программ, клавиатурные тренажеры, исполнители.

для классов УКНЦ.01 (на основе БК-0011 или БК-0011М):

Пакет ***«Альтернатива — БК-0011»***, который является адаптацией пакета ***«Альтернатива- 1»*** на этот тип классов.

Мы гарантируем:

высокий профессиональный уровень разработок;

комплектование пакетов подробными описаниями, облегчающими их быстрое освоение.

Только у нас:

очень низкие цены;

такие комплексные пакеты для всех типов стандартных классов;

столь подробная и полная документация;

столь высокая репутация как разработчиков и поставщиков (заверения о высоком качестве наших разработок можно получить в кабинете информатики любого ИУУ РФ).

Оптовым покупателям (роно, гороно, гуно, ИУУ) делается скидка.

Ждем заявок по адресу: 656049 Барнаул, а/я 3475, *«Альтернатива»*.

Научно-методический журнал.
Учрежден Министерством образования РФ
и коллективом редакции

Издается с августа 1986 г.
Выходит шесть раз в год
Москва «Школа-Пресс» 1992

ИНФ 3-4'92 май—август

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

Содержание

Методика

Матюшкин-Герке А. Учебно-прикладные задачи в курсе информатики 3
Бурцев С., Ефремов А., Ефремов Д., Зорич А. Комплект учебных миров КуМир 11
Струкова Л. Проектирование дидактического процесса в операционной среде «ЭСКИЗ» 16
Губанов А., Гузеев В. Проблемы разработки ЭС для профориентации и профотбора 19
Урнов В. Второй этап 23

КВТ

Ковтун М. Программирование в системе Express Pascal 25
Желтов А. Семейство «Автор» — гипермедиа для терминальных классов 30
Девятов А. Программирование 9-игольчатых матричных принтеров семейства Epson 33
Григорьев С. Пролог-Д в ПЗУ БК-0010 38
Майоров С., Дмитриева О. Тема «Операционные системы» на базе КУВТ-86 39
Soft-парад 40
Половина И. Введение функции пользователя в работающую программу 41

Клуб «Корвет»

Ахманов С., Нечаев А., Рой Н., Скурихин А. Архитектура «Корвета» 44
Ахманов С., Нечаев А., Скурихин А. Архитектура процессора КР580ВМ80А 51
Редченко В. «Заплатка» на программе 53
Дергаленко О. Инструментальные оболочки для КУВТ «Нейва» («Корвет») 54

Клуб «Агат»

Степанов М., Самолысов В. Дополнительные возможности графики ПЭВМ «Агат» 59
Литвинов В. Препроцессор интерпретатора языка Бейсик ПЭВМ «Агат» 64

Клуб УКНЦ

Гусев А. Операционная система ФОДОС 68
Крамер Е., Жигунов В. Использование периферийного процессора 77
Жигунов В., Носов М., Селиверстова И. Графическая обучающая система для УКНЦ 79
Калейдоскоп УКНЦ 83
Ларин В. Использование драйвера квазидиска 84

Клуб БК

О многочастотной кодовой модуляции 87
Еще раз о защите 89
СМ 6337+БК 91
Каталожная информационно-поисковая система 92
Форматировка результатов вычислений в Бейсике БК 93

Педагогический опыт

Шерпаев Н. Демонстрация вычислений 94
Чубров Е., Сливина Н., Демушкин А. Компьютер и изучение математики 96
Пинаев В. Ханойские перемешанные башни 97

Внеклассная работа

Очков В. Среда программирования 99
Воронова М. Вступительный экзамен по ОИВТ 104

Молодежная инициатива

109

Зарубежный опыт

Нарыкова И. Компьютерное моделирование в Великобритании 116

Информация

Видеотоварищество 121
Заседание Координационного совета 122

Справочный листок

Разъясняем нормативные документы 123

Обложка *С. Плюща*

Редактор отдела *А. Кравцова*
Научный редактор *Н. Копытина*
Зав. редакцией *Н. Игнатова*
Художественный редактор *Л. Коновалова*
Корректоры *Л. Яковлева, В. Антонова*

Сдано в набор 25.03.92. Подписано в печать 29.05.92. Формат 70×100¹/₁₆. Бумага офсетная. Печать офсетная. Усл. печ. л. 10,40. Усл. кр.-отт. 42,65. Уч.-изд. л. 13,00. Тираж 56 290 экз. Заказ 329. Цена 4 руб.
Издательство «Школа—Пресс»

Почту направлять по адресу: 119034, Москва, Смоленский б-р, д. 4.
Издательство «Школа—Пресс». Журнал «Информатика и образование».
Адрес редакции: Лефортовский пер., д. 8.
Телефон: 272-26-71.

Ордена Трудового Красного Знамени Чеховский полиграфический комбинат Министерства печати и информации Российской Федерации.
142300, г. Чехов, Московской обл.

А. МАТЮШКИН—ГЕРКЕ

Учебно-прикладные задачи в курсе информатики

В принятой нами концепции (см., например, [1]) курс ОИВТ должен включать разделы:

основы алгоритмизации и программирования,

структура и функционирование средств вычислительной техники,

решение задач с прикладным содержанием,

эволюция средств программного обеспечения ЭВМ,

вычислительная техника в народном хозяйстве; компьютерная культура современного специалиста.

В этой статье, как видно из заглавия, речь пойдет о третьем из перечисленных разделов, и отнюдь не сами по себе рассматриваемые в ней задачи, а именно следующие за их решением обсуждения и выводы являются главным. Нужно не просто тем или иным способом эти задачи решить, а показать и пояснить на их основе функциональные закономерности, связанные с применением вычислительной техники в различных областях человеческой деятельности.

Отступим на некоторое время от академического стиля изложения и позволим себе небольшое неформальное введение.

Как известно, герой знаменитого кинофильма «Небесный тихоход» майор Булочкин, упав с высоты 6000 м без парашюта, не только остался жив, но даже смог снова летать. Было бы интересно узнать, возможно ли такое на самом деле или же подобное случается только в кино.

В формализованном виде соответствующая задача будет выглядеть так.

Задача № 1. Падение тела с учетом сопротивления воздуха.

Составить программу, позволяющую получить табличную зависимость от времени значений пройденного пути, скорости и ускорения тела, движущегося под действием силы тяжести и силы сопротивления воздуха, пропорциональной квадрату скорости. Начальную скорость считать равной 0.

Выполнить эту программу для следующих исходных данных: масса тела $m=80$ кг, коэффициент сопротивления $L=0{,}5$ кг/м. (Обоснования такого выбора коэффициента L можно найти, например, в [2], но сейчас, на первых порах, лучше всего принять его на веру.)

Приступим к делу. Попробуем промоделировать на компьютере процесс падения нашего «беспарашютиста». Для этого нам придется кое-что (не пугайтесь, совсем немного!) вспомнить из школьного курса физики.

Формула:

$$F=F_т+F_{сопр} \qquad (1)$$

говорит о том, что на падающее тело в нашем случае действуют две силы: сила тяжести $F_т$ и сила сопротивления воздуха $F_{сопр}$. Выражение для первой из них:

$$F_т=m\times g, \qquad (2)$$

где m — масса тела, а $g=9.81$ — ускорение свободного падения, по всей видимости, не вызовет затруднений ни у читателей, ни у их учеников.

В условии дано, что сила сопротивления воздуха прямо пропорциональна квадрату скорости, с которой падает тело:

$$F_{сопр}=-L\times V^2. \qquad (3)$$

Обозначив через a ускорение, с которым движется падающее тело, мы (согласно 2-му закону Ньютона) сможем записать:

$$F=m\times a. \qquad (4)$$

Из (1) — (4) мы получаем основную

из нужных нам формул:

$$a=g-\frac{L}{m}\times V^2, \qquad (5)$$

которую, обозначив L/m через k, будем использовать в виде:

$$a=g-k\times V\times V. \qquad (6)$$

Разумеется, в отличие от ранее встречавшихся нам в школьном курсе физики задач, величина a не будет здесь постоянной. Однако с учетом того, что за малый промежуток времени от t до t+Dt изменение a тоже будет небольшим, мы вправе использовать следующее приближенное равенство:

$$V(t+Dt)=V(t)+a(t)\times Dt, \qquad (7)$$

где V(t) и V(t+Dt) — это значения скорости в моменты времени t и t+Dt соответственно, а a(t) — ускорение в момент времени t.

Среднюю скорость на этом промежутке времени примем приближенно равной полусумме (среднему арифметическому) значений V(t) и V(t+Dt):

$$V_{ср}=(V(t)+V(t+Dt))/2. \qquad (8)$$

Тогда путь, пройденный телом за время от t до t+Dt, выразится так:

$$DS=V_{ср}\times Dt, \qquad (9)$$

а обозначив соответственно через S(t) и S(t+Dt) расстояния, пройденные от начала движения до момента времени t и до момента времени t+Dt, получим:

$$S(t+Dt)=S(t)+DS. \qquad (10)$$

Как нетрудно было догадаться, мы собираемся при моделировании процесса падения разбить его на множество мелких участков, на каждом из которых будет правомерно применение формул (6) — (10).

В начальный момент времени (t=0) и пройденный путь S, и скорость движения V равны 0. Поэтому, согласно формуле (6), ускорение a будет равно:

$$a(0)=g-k\times 0\times 0=g.$$

Применяя теперь формулу (7), получим:

$$V(Dt)=0+g+Dt.$$

Отсюда

$$V_{ср}=(0+g\times Dt)/2$$

и

$$DS=((g\times Dt)/2)\times Dt,$$
$$S(Dt)=S(0)+DS=((g\times Dt)/2)\times Dt.$$

Таким образом, начав с момента времени t=0, мы получили все характеристики нашего процесса уже для t=Dt.

Вновь посчитаем ускорение, теперь уже в этот новый момент времени: a(Dt)=g—k×(g×Dt)×(g×Dt).

Зная a(Dt), тем же способом, что и раньше, найдем V(Dt+Dt):

$$V(Dt+Dt)=V(Dt)+a(Dt)\times Dt$$

(V(Dt) и a(Dt), стоящие здесь справа от знака равенства, только что были найдены), после чего рассчитаем новые значения $V_{ср}$, DS и S(Dt+Dt).

Тем самым будут получены ускорение, скорость и пройденный путь на момент времени 2×Dt. Аналогично будут проводиться расчеты для t=3×Dt, t=4×Dt и т. д.

Предлагаемая обычно учащимися блок-схема этого алгоритма приводится на рис. 1, а текст соответствующей программы (на языке Бейсик) — на рис. 2.


```
10 PRINT "ВВЕДИТЕ ИСХОДНЫЕ ДАННЫЕ"
20 INPUT "КОЭФФИЦИЕНТ L (КГ/М)=?";L
30 INPUT "МАССА M (КГ)=?";M
40 INPUT "ШАГ Dt (СЕК)=?";D
50 G=9.810001
60 K=L/M
70 T=0
80 S=0
90 V=0
100 A=G
130 PRINT T,A,V,S
140 IF S=>6000 THEN 210
150 T=T+D
160 A=G-K*V*V
170 V1=V+A*D
180 S=S+(V+V1)*D/2
190 V=V1
```

```
200 GOTO 130
210 PRINT "РАСЧЕТ ОКОНЧЕН"
220 END
```

Опытным глазом сразу можно заметить многие недостатки этой программы. Но вот стоит ли пытаться еще до ее запуска добиться, чтобы учащиеся поняли суть этих недостатков и нашли способы их устранения? Нам представляется, что в подобных случаях гораздо полезнее дать ребятам возможность самим обнаружить свои недоработки в процессе работы программы, а затем уже вместе с ними искать пути их исправления. Ведь априорные объяснения на фоне и без того достаточно трудного материала многими учениками просто не смогут быть восприняты.

Итак, запустим нашу программу, задав ей L=0,5, m=80 и Dt=0,1. Через несколько минут, выдав предварительно чуть больше полутора тысяч строк, программа закончит свою работу, и из оставшейся на экране информации мы узнаем, что все падение продолжалось около 154 с, а скорость приземления (точнее, приводнения) была равной примерно 39,6 м/с. Маловато данных для того, чтобы ответить на поставленный перед формулировкой условия задачи вопрос. А то, что могло бы нам в этом помочь, промелькнуло на экране так быстро, что мы просто не успели за ним уследить.

Да и с чисто формальных позиций вряд ли можно считать эту задачу решенной. Ведь на экране осталась лишь малая доля от общего числа информационных строк, сформированных программой.

Повторный запуск программы со значением Dt=1 принципиальных изменений в ситуацию не внесет. При попытке же использовать еще большие значения Dt (например, 5 или 10) мы получим в результате совершенную чушь: чередование больших по абсолютной величине положительных и отрицательных значений ускорения, соответствующие им изменения скорости то в сторону уменьшения, то в сторону увеличения и в конце концов останов программы по «переполнению».

Помочь здесь ребятам докопаться до сути этого феномена довольно легко. Неприменимость формулы (7) при больших значениях достаточно хорошо поддается объяснению.

Не так уж сложно оказывается и навести учащихся на способ разрешения выявившегося противоречия: обилие выводимой информации при малых значениях Dt и нарушение адекватности модели при больших его значениях. Здесь оказывается достаточным выполнять вычисления с малым Dt, но выводить на экран только отдельные из получаемых результатов (например, каждый десятый или каждый сотый и т. п.). Более того, можно организовать программу так, чтобы она выводила таблицу на экран блоками по 20 строк. В дальнейшем это окажется для нас весьма удобным. Предусмотренный в программе запрос о необходимости продолжения вывода таблиц успешно может заменить собой условие ее окончания по достижении переменной S значения 6000.

Что же касается обилия значащих цифр в выдаваемых результатах, а также небольших, но неуместных здесь погрешностей при вычислении значений t, то с этим учащиеся должны были научиться справляться еще при изучении начальных разделов курса (использовать, например, форматный вывод).

Усовершенствованный вариант программы приводится на рис. 3, а одна из выдаваемых ею видеограмм — на рис. 4. Из этой таблицы мы, в частности, можем усмотреть, что при падении с высоты 75 м скорость в конечный момент равнялась бы примерно 30—33 м/с.


```
10 PRINT "ВВЕДИТЕ ИСХОДНЫЕ ДАННЫЕ"
20 INPUT "КОЭФФИЦИЕНТ L (КГ/М)=?";L
30 INPUT "МАССА M (КГ)=?";M
40 INPUT "ШАГ Dt (СЕК)=?";DT
50 INPUT "ШАГ ТАБУЛЯЦИИ=Dt*N; N=";N
60 G=9.810001
70 T=0
80 S=0
90 V=0
100 K=L/M
110 A=G-K*V*V
120 W$="###.#  #.##  ##.##  ####.#"
130 PRINT"   T    A       V         S"
140 FOR I=1 TO 20
150 PRINT USING W$;T;A;V;S
160 FOR J=1 TO N
170 T=T+DT
180 A=G-K*V*V
190 V1=V+A*DT
200 S=S+(V+V1)*DT/2
210 V=V1
220 NEXT J
230 NEXT I
240 PRINT USING W$;T;A;V;S
250 INPUT "ПРОДОЛЖИТЬ (D/N)";U$
260 IF U$="D" OR U$="d" THEN 130
270 END
```

```
   T     A       V        S
 0.0   9.81    0.00     0.0
 1.0   9.34    9.64     4.9
 2.0   7.90   18.25    18.9
 3.0   6.05   25.14    40.8
 4.0   4.27   30.19    68.6
 5.0   2.85   33.65   100.6
 6.0   1.83   35.91   135.3
 7.0   1.15   37.34   172.2
 8.0   0.71   38.23   210.0
 9.0   0.43   38.78   248.5
```

```
 10.0   0.26   39.11    287.5
 11.0   0.16   39.31    326.7
 12.0   0.10   39.43    366.1
 13.0   0.06   39.51    405.6
 14.0   0.03   39.55    445.1
 15.0   0.02   39.58    484.7
 16.0   0.01   39.59    524.2
 17.0   0.01   39.60    563.8
 18.0   0.00   39.61    603.4
 19.0   0.00   39.61    643.1
 20.0   0.00   39.61    682.7
ПРОДОЛЖИТЬ (D/N)?
```

Но ведь именно с такой высоты (Бруклинский мост) один из американских каскадеров совершил прыжок в воду! Сравнение этой скорости с получившейся у нас величиной 39,6 м/с позволяет считать описанный в кинофильме эпизод вполне возможным.

Небольшая и достаточно очевидная корректировка программы: ввод значения начальной скорости падающего тела вместо присваивания ей значения 0, а также учет изменений во времени коэффициента L позволят использовать эту программу и для моделирования прыжка с последующим раскрытием парашюта.

Задача № 2. Прыжок с парашютом.

Составить программу, позволяющую получить табличную зависимость от времени значений пройденного пути, скорости и ускорения тела, движущегося под действием силы тяжести и силы сопротивления воздуха, равной произведению квадрата скорости на коэффициент, равномерно изменяющийся за первые 5 с от значения L1 до значения L2, а затем остающийся постоянным.

Выполнить ее для следующих исходных данных: m=80 кг, L1=0,5 кг/м, L2==100 кг/м, V0=35 м/с.

Действительно, для решения задачи достаточно в предыдущей программе вместо оператора 20 включить:

```
20 INPUT "L1 (кг/м)="; L1,
25 INPUT "L2 (кг/м)="; L2,
```

а вместо оператора 100:

```
100 K1=L1/M
103 K2=L2/M
105 K=K1
```

оператор 90 записать в виде:

```
90V=V0
```

и добавить оператор 85:

```
85 INPUT «НАЧ. СКОРОСТЬ V0 (м/сек)==»; V0.
```

Для учета же изменений коэффициента K по времени использовать:

```
107 DK=((K2—K1)/5)×Dt
173 IF T>5 THEN 180
177 K=K+DK
```

Попробуем теперь решить более трудную задачу.

Задача № 3. Расчет конечной скорости падающего тела.

С высоты 40 м (балкон 10-го этажа) один участник эксперимента отпускает падать надутую камеру от мяча. Его помощник засекает время падения по секундомеру и получает результат 8.0 с.

Требуется узнать, с какой скоростью произойдет приземление этой камеры.

От предыдущих эта задача отличается тем, что ни вес камеры, ни коэффициент сопротивления нам неизвестны. Тем не менее она вполне решаемая. Один из подходов к ее решению (далеко не самый изящный, но зато хорошо понятный учащимся) мы сейчас и осуществим.

Заметим прежде всего, что в наших расчетах величины L и m участвуют только через посредство своего отношения k==L/M. Заменим оператор 20 программы рис. 3 на:

```
20 INPUT «КОЭФ-Т k"=; K
```

и удалим из нее операторы 30 и 100. Пустим эту программу «на счет» и в ответ на ее запрос введем в качестве значения k число 0. Шаг Dt примем равным 0.1 с, а «шаг табуляции» N — равным 4 (для того, чтобы последняя строка выводимой таблицы соответствовала как раз значению t=8). Мы получим:

```
   T        A        V        S
 0.00     9.81     0.00     0.00
 . . .    . . .    . . .    . . .
 8.00     9.81    78.48   313.92
```

Видно, что за 8 с при k=0 тело пролетит гораздо больше указанных в условии 40 м. (Этого и следовало ожидать, ведь равенство нулю коэффициента k означает полное отсутствие сопротивления.)

Вновь запустим нашу программу, но теперь уже зададим k=1. В последней строке выводной таблицы мы будем иметь:

```
8.00    0.00    3.13    24.45
```

Очевидно, что в этот раз значение k мы взяли слишком большим. Продолжая наш вычислительный эксперимент, мы будем задавать k значения 0.5, 0.3, 0.4 и т. д. до тех пор, пока в итоговой таблице против значения t=8 мы не получим в графе «Расстояние» (S) числа, достаточно близкого к 40.

Так, например, при k=0.36 будет S==39.99. Точность вполне приличная.

Рассмотрим теперь полученную таблицу. Мы видим, что начиная с расстояния 3—4 м

скорость падения практически стабилизируется. Это наводит на мысль о возможной экспериментальной проверке наших расчетов. Достаточно замерить высоту отскока камеры при ее падении с высоты 40 м и с высоты 3,5 м. Совпадение этих величин (или, по крайней мере, их хорошая близость друг к другу) будет свидетельствовать в пользу выбранной нами модели.

Прежде чем приступать к обещанному обсуждению, рассмотрим еще одну группу задач, объединяемых нами под условным названием «Задачи о бассейне».

Во многих сборниках задач по математике до сих пор встречаются задачи такого типа: «В бассейн проведены две трубы. Через одну из них бассейн наполняется за 3 ч, а через вторую опорожняется за 2 ч. За какое время опорожнится наполненный до краев бассейн, если одновременно открыть обе трубы?»

По мысли составителей, эту задачу следует решать так: «За один час через первую трубу поступит 1/3 от общего объема бассейна, а через вторую вытечет 1/2 этого объема. Значит, в итоге за один час количество воды уменьшится на 1/2 — 1/3=1/6 всего объема. Следовательно, весь бассейн опорожнится за 6 ч».

На самом деле это, конечно, не так. Если поступление воды в бассейн действительно можно считать равномерным, то уж для ее вытекания подобное предположение принять нельзя. Чем выше уровень воды, тем сильнее будет происходить ее истечение. На это обстоятельство указывал еще в довоенные годы известный популяризатор науки Я. И. Перельман (см. [3]). Как мы увидим, в условиях приведенной задачи бассейн вообще никогда до конца и не опорожнится. Уровень воды в нем стабилизируется на некоторой высоте, зависящей от формы и размеров бассейна. Оставаясь в рамках традиционных школьных курсов математики и физики, такие задачи решить нельзя.

Покажем, что и здесь с появлением компьютеров (на худой конец,программируемых калькуляторов) наши возможности существенно расширяются.

Начнем с простейшей в этой группе задачи.

Задача № 4. Бассейн выполнен в форме прямоугольного параллелепипеда с длиной A, шириной B и высотой C. В его днище имеется снабженное задвижкой отверстие с площадью сечения S.

Вертикальная составляющая скорости частиц воды, вытекающих через это отверстие, определяется формулой Э. Торричелли:

$$V=\sqrt{2\times g\times h}, \qquad (11)$$

где g — ускорение свободного падения, а h — высота столба воды над отверстием.

Составить программу, позволяющую определить, за какое время опорожнится наполненный до краев бассейн, и, кроме того, выдающую на экран табличную зависимость h от времени t.

Приступим к решению этой задачи.

За элементарный промежуток времени Dt из бассейна вытечет объем воды DQ= =S×V×Dt, тем самым уровень ее понизится на величину Dh=—DQ/(A×B).

Для упрощения вычислений удобно ввести вспомогательную величину

$$L=-\frac{S\times\sqrt{2\times g}}{A\times B}, \qquad (12)$$

тогда цикл расчета будет выглядеть следующим образом:

$$Dh=L\times Dt\times\sqrt{h};\ h=h+Dh;\ t=t+Dt. \qquad (13)$$

Как и для задачи 1, первые варианты составленных учащимися программ будут нуждаться в определенной их доводке. При первых же запусках программы выясняется, что из-за того, что вычисления в программе не являются абсолютно точными, использовать равенство нулю в качестве условия окончания программы не удается, а получать на экране отрицательные значения этой переменной не хочется. Ясно, что надо особо оговорить условия завершения программы. Одним из возможных тут может быть дополнительное задание малой величины e, такой, чтобы по достижении переменной h этого значения мы вправе были считать бассейн опорожнившимся.

Сделанные пояснения, на наш взгляд, вполне достаточны для того, чтобы читателям стала понятна логика представленной на рис. 6 (уже «усовершенствованной») программы решения этой задачи.

Запустим программу, задав ей следующие исходные данные: A=B=C=1 (м), S= =0.001 (кв. м.), Dt=10 (с), N=3 и E= =0.001 (м). Результаты ее работы приводятся на рис. 7. Мы видим, что при заданных условиях бассейн опорожнится примерно за 430 с.

```
10 INPUT "ДЛИНА И ШИРИНА
                    БАССЕЙНА (М) =";A,B
20 INPUT "ВЫСОТА БАССЕЙНА (М) =";C
30 INPUT "ПЛОЩАДЬ ОТВЕРСТИЯ
                          (КВ.М) =";S
40 INPUT "E (К УСЛОВИЮ
                      ЗАВЕРШЕНИЯ) =";E
50 INPUT "ШАГ Dt =";DT
60 INPUT "ЧИСЛО ШАГОВ
                   ПЕРЕД ПЕЧАТЬЮ =";N
70 G=9.810001
80 L=-SXSQR(2XG)/AXB
90 H=C
```

```
100 T=0
110 W$="####.###      ##.###"
120 CLS
130 LOCATE 1,1
140 PRINT "      T          H"
150 FOR I=1 TO 20
160 FOR K=1 TO N
170 T=T+DT
180 DH=L*SQR(H)*DT
190 H=H+DH
200 IF H<E THEN 220
210 NEXT K
220 PRINT USING W$;T,H
230 IF H<E THEN 320
240 NEXT I
250 IF H<E THEN 320
260 LOCATE 22,1
270 PRINT "ПРОДОЛЖАТЬ (D/N)";
280 U$=INPUT$(1)
290 LOCATE 22,1
300 PRINT "                              "
310 IF U$="D" OR U$="d" THEN 120
320 LOCATE 22,1
330 PRINT "ВРЕМЯ ОПОРОЖНЕНИЯ
                        БАССЕЙНА =";T
340 LOCATE 22,40
350 PRINT "ОКОНЧАНИЕ РАСЧЕТА";
355 LOCATE 22,40
360 END
```

8

```
     T          H
  30.000      0.870
  60.000      0.749
  90.000      0.637
 120.000      0.534
 150.000      0.440
 180.000      0.355
 210.000      0.279
 240.000      0.211
 270.000      0.153
 300.000      0.104
 330.000      0.064
 360.000      0.034
 390.000      0.013
 420.000      0.001
 430.000     -0.000
ВРЕМЯ ОПОРОЖНЕНИЯ БАССЕЙНА = 430
                    ОКОНЧАНИЕ РАСЧЕТА
```

Выполнение программы с теми же исходными данными, кроме значения Dt, которое мы зададим равным сначала 2, а затем 0.5 с, не займет много времени, но даст зато хороший повод обсудить с учащимися причины незначительных расхождений в результатах.

Следующая задача (она приводится чуть ниже) является своего рода промежуточной между только что решенной и той, с которой мы начали эту часть статьи. Здесь уже в условии участвуют обе трубы, но для той, через которую вода выливается, дается пока еще не время опорожнения, а все та же площадь сливного отверстия S.

Итак, задача № 5: «К бассейну, описанному в задаче № 4, добавлена на уровне верхнего его края труба, через которую при закрытом выходном отверстии он наполняется за T1 с.

Составить программу получения табличной зависимости уровня воды h от времен t при условии, что при t=0 открываются и выходное отверстие, и наполняющая труба, а начальное значение уровня составляет H0».

Коль скоро при отсутствии стока бассейн через вторую трубу наполняется за T1 с, то «вклад» этой трубы в изменение уровня воды за время Dt остается неизменным и равным высоте бассейна, деленной на время его заполнения и умноженной на Dt:

$$Dh1=(C/T1)\times Dt.$$

Изменение уровня за время Dt при условии, что работают обе трубы, будет, следовательно, равным Dh+Dh1, где Dh — величина, полученная нами при решении предыдущей задачи.

Сравним друг с другом эти два слагаемых. Первое из них, как видно из (12) и (13), будет отрицательным и при изменении h от 0 до C будет меняться по абсолютной величине от 0 до Dh (max), где

$$Dh(max)=\frac{S\times\sqrt{2\times g\times C}}{A\times B}\times Dt,$$

а второе — оставаться равным одному и тому же положительному числу $Dh1=(C/T1)\times Dt$.

Если $Dh1<Dh(max)$, то при каком-то значении h, заключенном между 0 и C, наступит состояние равновесия: Dh1+Dh=0. (Вот и обещанное объяснение, почему полного опорожнения в этом случае так и не произойдет.) Если же $Dh1>Dh(max)$, то сумма Dh1+Dh всегда будет строго больше 0 и, значит, бассейн, независимо от начального уровня, заполнится до краев и излишек воды будет из него выливаться уже не через трубу, а через борт.

Без учета этой второй возможности нам достаточно было бы изменить в предыдущей программе только один оператор:

$$180\ Dh=(C/T1+L*SQR(h))\times Dt,$$

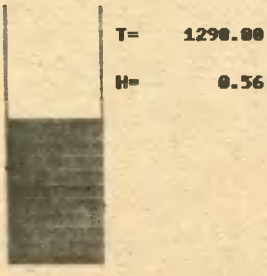
добавить оператор ввода для T1 и, если нам будет угодно, модифицировать условие окончания работы программы. Но поскольку действительно скорость притока воды может оказаться больше максимальной скорости ее оттока, нам придется учесть в программе и эту возможность. Полностью один из вариантов такой программы дается на рис. 8.

```
10 INPUT "ДЛИНА И ШИРИНА БАССЕЙНА
                          (М) =";A,B
20 INPUT "ВЫСОТА БАССЕЙНА (М) =";C
30 INPUT "ПЛОЩАДЬ ОТВЕРСТИЯ
                       (КВ.М) =";S
40 INPUT "E (К УСЛОВИЮ
                   ЗАВЕРШЕНИЯ) =";E
50 INPUT "НАЧАЛЬНОЕ ЗНАЧЕНИЕ
                      УРОВНЯ =";H0
```

```
60 INPUT "ВРЕМЯ ЗАПОЛНЕНИЯ
                    (БЕЗ СТОКА) =";T1
70 INPUT "ШАГ Dt =";DT
80 INPUT "ЧИСЛО ШАГОВ ПЕРЕД
                        ПЕЧАТЬЮ =";N
90 G=9.810001
100 L=-S*SQR(2*G)/A*B
110 M=C/T1
120 X=A*B*C
130 Y=S*T1
140 H1=X*X/(2*Y*Y*G)
150 H=H0
160 T=0
170 IF C<H1 THEN H2=C ELSE H2=H1
180 REM H2 - "уровень равновесия"
190 W$="####.#     ##.###"
200 CLS
210 PRINT "      T          H"
220 FOR I=1 TO 20
230 FOR K=1 TO N
240 T=T+DT
250 DH=(M+L*SQR(H))*DT
260 H=H+DH
270 IF H>C THEN H=C
280 IF ABS(H-H2)<E THEN 370
290 NEXT K
300 PRINT USING W$;T,H
310 NEXT I
320 LOCATE 22,1
330 PRINT "ПРОДОЛЖАТЬ (D/N)";
340 U$=INPUT$(1)
350 IF U$="D" OR U$="d" THEN 200
360 LOCATE 22,1
370 PRINT USING W$;T,H
380 LOCATE 23,5
390 PRINT "ОКОНЧАНИЕ РАСЧЕТА"
400 LOCATE 22,40
410 END
```

Может, видимо, представить интерес и совершенно другой ее вариант (см. рис. 9). Выдаваемая им видеограмма представлена на рис. 10 и, полагаем, дополнительных пояснений не требует.

```
10 INPUT "ДЛИНА И ШИРИНА БАССЕЙНА
                        (М) =";A,B
20 INPUT "ВЫСОТА БАССЕЙНА (М) =";C
30 INPUT "ПЛОЩАДЬ ОТВЕРСТИЯ
                    (КВ.М) =";S
40 INPUT "E (К УСЛОВИЮ
                ЗАВЕРШЕНИЯ) =";E
50 INPUT "НАЧАЛЬНОЕ
            ЗНАЧЕНИЕ УРОВНЯ =";H0
60 INPUT "ВРЕМЯ ЗАПОЛНЕНИЯ
                (БЕЗ СТОКА) =";T1
70 INPUT "ШАГ Dt =";DT
80 INPUT "ЧИСЛО ШАГОВ
                ПЕРЕД ПЕЧАТЬЮ =";N
90 w$="####.##"
100 G=9.810001
110 L=-S*SQR(2*G)/A*B
120 M=C/T1
130 X=A*B*C
140 Y=S*T1
150 H1=X*X/(2*Y*Y*G)
160 H=H0
170 T=0
180 IF C<H1 THEN H2=C ELSE H2=H1
190 REM H2 - "уровень равновесия"
200 PRINT "PRESS 'ENTER'"
210 U$=INPUT$(1)
220 SCREEN 1
230 CLS
240 LOCATE 3,10
250 PRINT "T="
260 LOCATE 5,10
270 PRINT "H="
280 LINE (5,5)-(5,105)
290 LINE -(55,105)
300 LINE -(55,5)
310 X0=104-(100*H)/C
320 FOR I=104 TO X0 STEP -1
330 FOR J=6 TO 54
340 PSET (J,I)
350 NEXT J
360 NEXT I
370 LOCATE 20,1
380 PRINT "PRESS 'ENTER'"
390 U$=INPUT$(1)
400 LOCATE 20,1
410 PRINT "                    "
420 X1=104-(100*H)/C
430 FOR K=1 TO N
440 T=T+DT
450 DH=(M+L*SQR(H))*DT
460 H=H+DH
470 IF ABS(H-H2)<E THEN 700
480 IF H>C THEN 700
490 NEXT K
500 LOCATE 3,15
510 PRINT USING W$;T
520 LOCATE 5,15
530 PRINT USING W$;H
540 X2=104-(100*H)/C
550 IF X1>X2 THEN 620
560 FOR I=X1 TO X2
570 FOR J=6 TO 54
580 PSET (J,I),0
590 NEXT J
600 NEXT I
610 GOTO 670
620 FOR I=X1 TO X2 STEP -1
630 FOR J=6 TO 54
640 PSET (J,I)
650 NEXT J
660 NEXT I
670 X1=X2
680 IF H<E THEN 700
690 GOTO 430
700 H=C
710 LOCATE 20,1
720 STOP
730 SCREEN 2
740 SCREEN 0
750 END
```



Ну а как же быть с «классической» задачей (см. задачу № 6 в конце статьи), в которой ни слова не говорится ни о каких размерах выходного отверстия, а дается лишь время заполнения и время опорожнения?

Разумеется, надо свести ее к уже решен-

ной, а именно к задаче № 5. Но как узнать используемую там величину S? Читатель, равно как и ученик, внимательно проследивший за решением задачи № 3, сразу предложит вновь прибегнуть к вычислительному эксперименту: используя программу для задачи № 4, нащупать такое значение S, при котором рассчитываемое время опорожнения практически совпадало бы с задаваемым в условии.

Так, например, если при отсутствии притока воды бассейн с размерами $1\times1\times1$ опорожняется за 300 с, то после нескольких прогонов программы (рис. 6) мы получим для S приближенное значение 0,00144 м^2, а для случая T=436 (сравните с результатами решения задачи № 4) будет S=0.001.

Если теперь мы зададим время опорожнения (при отсутствии притока) T=300 с, а время заполнения (при закрытом сливном отверстии) T1=240 с, то, узнав только что указанным способом размер S=0.00144, мы, используя затем программу (рис. 7), выясним, что уровень воды стабилизируется примерно на высоте 0.45 м.

Если же по-прежнему T=300 с, а T1=150 с, то стабилизация будет иметь место на уровень верхнего края бассейна.

Зачем же вообще в курсе информатики нужны такие трудные и к тому же лишенные особых внешних эффектов задачи? Может быть, лучше было бы позаниматься с ребятами программами изображения на экране распускающихся розочек или дать им поиграть в какую-либо забавную игру? Попробуем объяснить.

Известно, что за прошедшие десятилетия вычислительная техника использовалась и у нас в стране, и за рубежом, мягко говоря, не очень эффективно. Одна из причин этого — непонимание пользователями объективных закономерностей, относящихся к включению в ту или иную область деятельности какого-либо нового фактора, позволяющего резко усилить физические или интеллектуальные возможности исполнителей. В полной мере и, пожалуй, прежде всего это касается ЭВМ. Пытаясь сделать с их помощью то же самое, что делалось без них, к тому же практически теми же старыми методами, мы, образно говоря, использовали ЭВМ как микроскоп для забивания гвоздей. Так было и с АСУ, и с применением вычислительных машин для решения научно-технических задач да и во многих других случаях.

Хотелось бы, чтобы подрастающее поколение не повторяло наших ошибок. Для этого необходимо, чтобы учащиеся не только освоили азы формальных операций с компьютерами, не смотрели на него, как на забавную игрушку, но и (прежде всего!) поняли те особенности, которые связаны как с новыми возможностями, открывающимися при применении ЭВМ, так и с новыми требованиями к специалистам, прямо или косвенно эту технику использующим.

Мы убеждены, что, несмотря на малый объем курса, в нем все-таки возможно сформировать у ребят начальные представления о том, что для успешного использования ЭВМ в любой области деятельности главным является понимание безусловной необходимости коренных изменений ее содержания и методов и (обязательно!) умение и готовность эти изменения осуществить. Самая высокая техника программирования и блестящее владение спектром новейших средств обработки информации, великолепное знание архитектуры компьютеров окажутся в лучшем случае (!) бесполезными, если эти знания, умения и навыки не найдут себе правильной точки приложения.

Именно на решение этой проблемы и ориентирован рассматриваемый здесь раздел курса. Закрепление навыков алгоритмизации и программирования носит тут вспомогательный характер. Главными же являются методологические (не путать с методическими!) аспекты.

Обязательно (и весьма настойчиво) следует обратить внимание обучаемых на следующие обстоятельства.

Рассмотренные задачи не могли быть нами решены без использования ЭВМ. Иными словами, мы, пусть пока на учебных примерах, увидели, что вычислительная техника позволяет значительно расширить круг доступных решению задач.

Эти задачи были нами решены методами, совершенно отличными от тех, которые используются в традиционном школьном курсе физики. Значит, компьютеры не только позволили нам существенно расширить наши возможности, но и потребовали применения принципиально новых методов (пошаговое моделирование, вычислительный эксперимент).

В какой мере сформулированная выше учебная цель будет достигнута, существеннейшим образом зависит, конечно, и от внутренней убежденности учителя в ее значимости. Эта убежденность в свою очередь появляется обычно как следствие обстоятельной и разносторонней подготовки. К сожалению, большинство наших учителей не имели возможности такую подготовку получить. Но это уже предмет особого разговора.

Мы уже отмечали, что использовавшиеся нами программы отнюдь не представляли собой шедевров программирования. Разумеется, вполне реально довести до понима-

ния учащихся и более изящные способы решения. Например, вместо «ручного нащупывания» искомого значения параметра в задачах № 3 и 6 можно поручить эту работу самой программе. Мы и рекомендуем обратиться к таким программам (см., например, задачи № 7, 10—13 в конце статьи), но... во-первых, не сразу, а лишь после того, как сущность проводимого вычислительного эксперимента будет хорошо осознана учащимися в процессе работы с программами более примитивными, но зато и гораздо более «прозрачными». А во-вторых, уделить этим отшлифованным программам достаточно скромное место. Не следует стремиться обучить умению составлять такие программы всех без исключения. Во многих случаях тут целесообразно будет ограничиться демонстрацией уже готовых программ и предложить проделать аналогичные разработки только наиболее сильным ученикам, и притом по их собственному желанию. Сам по себе такой подход будет способствовать правильному формированию у обучаемых своего рода шкалы значимости, что должно впоследствии помочь им выработать иммунитет против чрезмерного преувеличения роли одних аспектов прикладной информатики по сравнению с другими.

Необходимо помнить, что школьный курс информатики является общеобразовательным, и главные акценты в нем следует расставлять, ориентируясь прежде всего на будущего рядового пользователя, а отнюдь не на профессионального программиста.

Продолжение следует.

С. БУРЦЕВ, А. ЕФРЕМОВ, Д. ЕФРЕМОВ, А. ЗОРИЧ

Комплект учебных миров КуМир

Памяти А. П. Ершова

Проиллюстрируем работу Строителя с помощью следующего алгоритма. Предположим, что Строитель стоит перед столбиком. Задача — забраться на этот столбик. Если высота столбика больше 1, то необходимо произвести строительные работы. В этом случае сначала попросим Строителя снимать кубики до тех пор, пока можно будет сделать шаг. Затем скомандуем ему сделать шаг, развернуться и сложить обратно все кубики, которые были сняты.

```
алг вперед
дано ! Строитель перед столбиком лицом к нему
надо ! Строитель на столбике лицом в том же направлении
нач
. нц пока высота__спереди — высота >1
.  . взять
. кц
. шаг__вперед
. кругом
. нц пока высота__спереди > 0
.  . взять
. кц
. кругом
кон____
```

Примеры задач для Строителя.

В следующих задачах предполагается, что на поле Строителя лежат кубики, причем на одной клетке лежит не больше одного кубика.

1. Определить, сколько кубиков лежит в ряду южнее Строителя.

2. Строитель у западной стены. На полу в ряду Строителя лежит кубик. Принести его к западной стене.

3. На полу лежит кубик. Строитель в юго-западном углу. Найти расстояние (количество шагов) до кубика.

4. Найти какой-нибудь кубик, находящийся на минимальном расстоянии от северо-восточного угла (расстояние между двумя клетками есть количество шагов, за которое можно дойти из одной клетки в другую).

При помощи Строителя можно моделировать задачи, связанные с линейными таблицами. Предположим, что на поле Строителя кубики есть только в одном горизонтальном ряду.

5. Найти среднюю высоту столбиков.

6. Найти высоту самого низкого столбика.

7. Найти количество столбиков максимальной высоты.

Окончание. Начало см.: ИНФО. 1991. № 2.

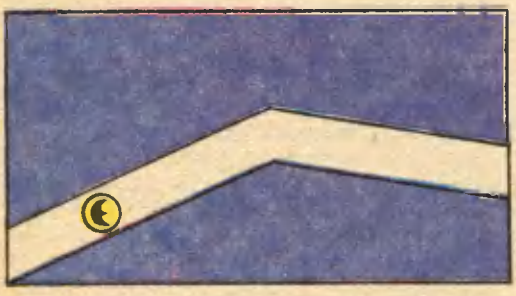
7. Мир Вездехода

Вездеход движется в поле, снабженном прямоугольной системой координат OXY. Поле может иметь произвольные размеры, на нем находятся прямолинейные стены произвольной длины, расположенные произвольным образом. Вездеход имеет конечные размеры и может перемещаться по полю, не наезжая на стены. Он «помнит» свои координаты и направление своего движения. На Вездеходе имеется локатор, с помощью которого можно измерить расстояние до ближайшей стены в направлении движения Вездехода.

Вездеход круглый, локатор показывает направление движения. Вот список команд Вездехода:

```
вездеход              вперед(цел а)   цел угол
налево(цел а)         цел поз_х       цел расст
направо(цел а)        цел поз у
```

Команда «вездеход» устанавливает исполнителя в начальное положение. По командам «налево» и «направо» Вездеход поворачивается на указанное число градусов, по команде «вперед» — сдвигается. Из команд обратной связи отметим команду «расст», возвращающую расстояние до ближайшей стены в направлении движения Вездехода.



Конструктор мира Вездехода может ставить новые стены, убирать старые, а также ставить Вездеход на новое место и поворачивать его в нужном направлении. При изменении обстановки в поле Вездехода появляется курсор, который служит для постановки и удаления стен. Вот список команд Конструктора:

```
курсор_влево(цел а)    начать_постановку
курсор_вправо(цел а)   начать_удаление
курсор_вверх(цел а)    кончить
курсор_вниз(цел а)     сбросить
курсор_в(цел а1, а2)   уст_вездеход
                       уст_направление

очистить
записать
цел текущая_поз_х
цел текущая_поз_у
цел число стен
```

Подведя с помощью первых пяти команд курсор в нужную точку (т. е. к одному из будущих концов новой стены или существующих концов старой), можно начать ставить или удалять стену с помощью следующих двух команд. Подведя курсор к другому концу стены, можно окончить постановку или удаление командой «кончить». По команде «сбросить» процесс создания или удаления стены прерывается. По команде «очистить» удаляются все стены, а по команде «записать» запоминается текущее состояние обстановки. Содержание других команд ясно из названий.

Приведем примеры, иллюстрирующие возможности Вездехода. Они иллюстрируют разные стили управления Вездеходом и разные возможности его использования. Первый пример чисто алгоритмический.

Имеется коридор с одним изгибом, и Вездеход стоит перед западным входом в коридор. Мы проведем Вездеход через коридор «на ощупь», не измеряя расстояний. Обратная связь осуществляется проверкой того, совпадает ли действительное смещение Вездехода с предписанным. Вот алгоритм:

```
алг проход
 дано ! Вездеход у входа в коридор
 надо ! пройти коридор
нач
. до_упора
. нц пока можно_вверх
.  . налево (90); до_упора; направо (90);
до_упора
. кц
. нц пока поз_х<240
.  . направо (90); до_упора; налево (90);
до_упора
. кц
кон____
алг лог можно_вверх
 дано ! Вездеход стоит в направлении
на запад
 надо ! проверить, можно ли сместиться
вверх
 нач цел у
 . у:=поз_у; налево (90); вперед (1);
направо (90)
 . если у=поз_у
 .  . то знач :=ложно
 .  . иначе знач :=истинно
 . все
 кон____
 алг до_упора
  дано ! Вездеход в любом положении
  надо ! сместиться вперед до ближайшей
стены
 нач
 . вперед (500)
 кон____
```

Второй пример требует некоторых математических знаний. Это вычисление площа-

ди с помощью Вездехода. Теперь Вездеход стоит на месте, а информацию об окружающей обстановке он получает с помощью измерения расстояний. Вездеход находится внутри выпуклого многоугольника. Он подсчитывает его площадь, разбивая многоугольник на треугольники, одна из вершин которых находится в точке, где он стоит, а две другие — на сторонах многоугольника. Алгоритм имеет вид:

```
алг треугольная_площадь
 дано ! Вездеход внутри звездного многоугольника
 надо ! вычислить площадь этого многоугольника
нач вещ пл, d_f, рад1, рад2, pi, цел i
. pi :=3.14159
. d_f :=sin (10*pi/360)
.пл:=0
.рад2:=расст
. нц для i от 1 до 72
. . рад1 :=рад2; налево (5); рад2 :==расст
. . пл :=пл+рад1*рад2/2*d_f
. кц
кон_____
```

В заключение приведем несколько задач для Вездехода.

1. Все стены в поле вертикальны. Вездеход находится у левого края поля. Надо пройти к правому краю.

2. В поле есть коридор, по которому можно пройти «не двигаясь назад», т. е. без уменьшения x-координаты. Вначале Вездеход стоит у левого края. Требуется пройти к правому краю.

3. Вездеход внутри выпуклого многоугольника. Сместиться в крайнюю левую точку этого многоугольника.

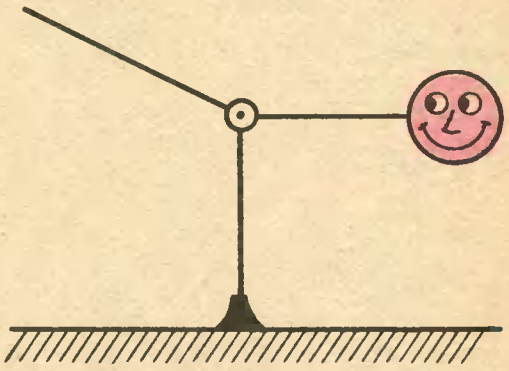
4. Пусть стены образуют выпуклый многоугольник, и пусть Вездеход находится внутри этого многоугольника. Определить: а) периметр многоугольника; б) число углов; в) площадь многоугольника.

5. Пусть стены образуют выпуклый многоугольник, и пусть Вездеход находится вне этого многоугольника. Определить площадь многоугольника.

6. Пусть Вездеход стоит перед стеной (т. е. его локатор показывает расстояние до какой-то точки этой стены). Вычислить: а) минимальное расстояние до этой стены; б) угол, под которым видна стена; в) длину стены.

8. Мир Двунога

Как уже говорилось, Двуног — это существо с тремя конечностями: двумя ногами и тяжелой головой на очень длинной шее. Рук у Двунога нет. Вместо туловища у него центральный шарнир, вокруг которого могут независимо вращаться ноги и шея. Живет Двуног в плоском (двумерном) мире. Задача состоит в том, чтобы научить Двунога двигаться: научить его садиться, вставать, шагать, подниматься по лестнице, словом, научить его передвигаться примерно так, как это делает человек. Передвигается Двуног по поверхности земли, образующей некоторый рельеф — ступеньки, ямы, препятствия и т. п.



Расскажем подробнее об анатомии и физиологии Двунога. Мы уже говорили, что вместо ступней у Двунога присоски, которые могут включаться и выключаться. Внутри присосок находятся шарниры. Присоски могут быть прижаты или отпущены, а шарниры — зафиксированы или свободны. Когда присоски отпущены, они могут отрываться от земли. Но Двуног имеет развитое чувство самосохранения, поэтому последняя присоска никогда не отрывается. Используя присоску и шарнир в ней, Двуног может, например, стоять на одной ноге, как фигурист, исполняющий «ласточку». Напомним, что голова у Двунога очень тяжелая, а ноги и шея ничего не весят, так что Двунога на рисунке тянет свалиться вперед, а точнее, направо — куда наклонена голова.

Что будет, если Двуног на рисунке, стоя «в ласточке», освободит шарнир в присоске? Конечно же, он начнет падать туда, куда его тянет голова,— направо. Падает он как единое целое: ноги и шея не меняют своего положения друг относительно друга, они жестко фиксированы в центральном шарнире.

Мы не рассказали еще ничего о нервной системе Двунога. Между тем Двуног обладает прекрасным вестибулярным аппаратом — он умеет определять абсолютную высоту своих конечностей. Это позволяет ему, например, отличать яму от ровной дороги.

Наконец, надо сказать несколько слов об окружающей Двунога среде. Потолки, стены,

пол, лестницы и т. д. состоят из одного и того же жесткого материала, который мы будем называть «земля». Все, что состоит из земли, мы будем называть рельефом. Двуног, вообще говоря, не может изменять рельеф, например провалиться или вырыть яму. Кроме того, Двуног не может отрываться от рельефа — прыжки Двуногу недоступны. В любой момент времени Двуног касается земли хотя бы одной из присосок.

Команды, управляющие движением Двунога:

```
голову__прямо
поднять__левую__ногу (цел i)
поднять__правую__ногу (цел i)
опустить__левую__ногу (цел i)
опустить__правую__ногу (цел i)
голову__налево (цел i)

голову__направо (цел i)
цел высота__левой
цел высота__правой
цел высота__головы
отпустить (цел i)
```


Команда «голову__прямо» заставляет Двунога поднять голову и выпрямить шею, насколько это возможно. (Ему может, например, помешать потолок.) Следующие шесть команд управляют вращением конечностей Двунога: они поворачивают соответствующую конечность на i градусов в нужном направлении. Следующие три команды позволяют использовать вестибулярный аппарат Двунога — с их помощью определяется абсолютная высота левой и правой ступни и головы соответственно. Команда «отпустить» отпускает у Двунога присоски и освобождает все шарниры, кроме центрального, на время, указанное в аргументе. Присоску, которая все же остается прижатой, Двуног выбирает сам так, чтобы он мог вращаться вокруг шарнира, находящегося в этой присоске.

В Мире Двунога рельеф состоит из прямоугольных блоков. Конструктор Мира Двунога может устанавливать дополнительные блоки, убирать прямоугольные куски блоков и ставить Двунога в нужное положение.

Примеры задач для Двунога.

1. Двуног стоит на ровной дороге; надо сесть на землю (лечь на землю) и затем встать. Двуног висит на потолке, прицепившись одной из ног. Надо пройти по потолку от одного края экрана до другого.

2. Двуног стоит на ровной дороге. Разрешается поставить его в любое начальное положение. Надо пройти от левой стороны экрана до правой, используя только следующие предписания: «голову__налево», «отпустить».

3. Двуног стоит на ровной дороге. Надо: а) сдвинуться вправо на заданное расстояние; б) померить расстояние между двумя стенами. Двуног стоит перед лестницей. Надо померить высоту от пола до потолка. Приготовить рельеф и написать программу, позволяющую Двуногу забраться с пола на потолок.

4. Двуног стоит на ровной дороге перед прямоугольной ступенькой неизвестной высоты. Надо забраться на ступеньку. (Замечание. Программа должна быть универсальной и работать независимо от высоты ступеньки.) Двуног стоит на пересеченной местности. Надо нарисовать карту рельефа (запомнить вершины всех блоков-прямоугольников, составляющих рельеф и находящихся в пределах досягаемости Двунога).

9. Мир Редактора

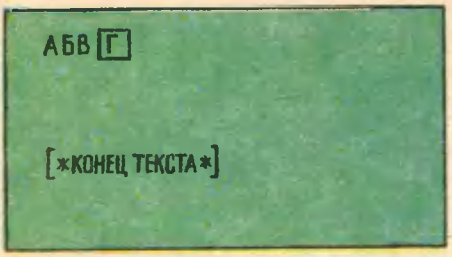
Редактор умеет создавать и изменять тексты. Реальным прототипом этого исполнителя является программа «редактор текстов». Но в отличие от этой программы в Мире Редактора школьник может писать алгоритмы обработки текста, а не делать это вручную. Надо отметить, что базовые возможности «редактора текстов» и Редактора совпадают. Поэтому можно сказать, что Редактор — это программируемый «редактор текстов».

Вообще говоря, в КуМире есть встроенный «редактор текстов», который используется при написании алгоритмов и без знакомства с которым работа с КуМиром невозможна. Существует множество программ «редактор текстов», например «МикроМир», «Тор», «WordStar» и др. Отметим, что вызов команд Конструктора Мира Редактора при ручном изменении обстановки совпадает с вызовом соответствующих команд в «МикроМире» и во встроенном редакторе КуМира. Кроме того, команды Редактора входят в базовый набор любого «редактора текстов».

Таким образом, познакомившись с Миром Редактора, школьник будет в некоторой степени подготовлен к работе с программой «редактор текстов». Это несомненно будет полезно для него, поскольку обработка текстов с помощью компьютера — одно из основных применений персональных ЭВМ.

Редактор отмечает место возможного изменения символа в тексте с помощью курсора. Символ, на котором стоит курсор, называется текущим; текущей также называется соответствующая строка. Все изменения, вносимые в текст, производятся в месте, тем или иным образом связанном с положением курсора. Редактор умеет перемещать курсор и изменять текст. У Редактора имеются, правда очень ограниченные, возможности запоминания и вспоминания — бу-

фер Редактора имеет вместимость один символ. Изображение Мира Редактора показано на рис. 4.



Текст, с которым работает Редактор, представляет собой последовательность строк фиксированной длины. Разумеется, если текст не помещается на экране целиком, отображается только его часть.

Подлежащий обработке текст (точнее, его видимая часть) ограничен рамкой (на рисунке — внутренней), это место на экране называется текстовым окном Редактора. Строка «[* конец текста *]», задает границу текста. В светящемся прямоугольнике справа в негативном изображении появляется содержимое буфера.

Вот список команд Редактора:

```
вперед
назад
вверх
вниз
в__начало__строки
в__конец__строки
в__начало__текста
в__конец__текста
установить__курсор(цел поз)
установить__текущий(лит знак)
записать__строку(лит строка)
вставить__пробел
удалить__символ

вставить__строку
удалить__строку
запомнить
установить__буфер(лит символ)
вспомнить
лог в__начале__строки
лог в__конце__строки
лог в__начале__текста
лог в__конце__текста
цел позиция
лит текущий
лит буфер
```

Первые девять команд — это команды перемещения курсора: при этом команда «установить__курсор» устанавливает курсор в выбранное положение в текущей строке. Команда «установить__текущий» заменяет текущий символ на символ «знак», а по команде «записать строку» Редактор заменяет каждый из символов, начиная с текущего, соответствующим символом аргумента «строка». По команде «запомнить» в буфер помещается текущий символ, а при выполнении команды «вспомнить» текущий символ заменяется содержимым буфера. Последние семь команд осуществляют обратную связь и позволяют определить положение курсора, текущий символ и содержимое буфера.

Пример. Составим алгоритм для замены одного символа на другой в текущей строке.

```
алг замена__в__строке(лит символ1, символ2)
    дано ! курсор в начале строки и не в конце текста
    надо ! все вхождения символ 1 в текущей строке заменены
         ! на символ 2, курсор в начале следующей строки
нач
. если текущий=символ 1
. . то установить__текущий (символ2)
. все
. нц пока не в__конце__строки
. . вперед
. . если текущий=символ1
. . . то установить__текущий (символ2)
. . все
. кц
. вниз; в__начало__строки
кон
```

Используем этот алгоритм для решения следующей задачи: заменить все вхождения одного символа другим во всем тексте. Вот требуемая программа:

```
алг замена__в__тексте(лит символ1, символ2)
    дано ! некоторый текст
    надо ! все вхождения символ1 в тексте
нач ! заменены на символ2
. в__начало__строки; в__начало__текста
. нц пока не в__конце__текста
. . замена__в__строке (символ1, символ2)
. кц
кон
```

Задачи для Редактора:

1. Найти в тексте заданный символ, строку.

2. Переместить курсор в строке на место последнего символа, отличного от пробела.

3. Заменить внутри строки подряд идущие пробелы одним.

4. Выровнять текст: а) по правому краю; б) по левому краю; в) по правому и левому краю.

5. Подсчитать в тексте: а) число слов; б) число предложений; в) число различных слов.

6. Найти самое длинное и самое короткое слово в тексте.

7. Зашифровать текст: а) записывая каж-

дую строку в обратном порядке; б) записывая каждое слово в обратном порядке.

8. Возможно, некоторые существительные с сочетаниями -ци- и -цы- в тексте написаны с ошибками. Составьте алгоритм, исправляющий эти ошибки.

Литература

1. *Варсанофьев Д. В., Кушниренко А. Г., Лебедев Г. В.* Е-практикум — программное обеспечение школьного курса информатики и вычислительной техники // Микропроцессорные средства и системы. 1985. № 3. С. 27—32.

2. *Варсанофьев Д. В., Дымченко А. Г.* Е-86 // Информатика и образование. 1986. № 1. С. 72—74.

3. *Леонов А. Г., Эпиктетов М. Г.* Практикум Е-87. М.: ВИНИТИ АН СССР, № 4279-В88 (депонировано). 8 с.

4. *Бурцев С. В., Ефремов А. В., Ефремов Д. В., Зорич А. В., Кушниренко А. Г.* Комплект учебных миров КуМир — новое поколение развивающего программного обеспечения для школьников: Сборник общества «Знание». В печати.

5. *Эпиктетов М. Г.* Учебно-производственная среда на основе школьного алгоритмического языка // Материалы V Всесоюзного семинара «Разработка и применение программных средств ПЭВМ в учебном процессе». М.: ИПИ АН СССР, 1989.

6. *Кушниренко А. Г., Лебедев Г. В., Сворень А. Р.* Основы информатики и вычислительной техники. М.: Просвещение, 1990.

7. *Isaacs P. M., Cohen M. F.* Controlling dynamic simulatioh with kinematic constraints, behavior functions and inverse dynamics. Computer Graphics, 1987, Vol. 21, n 4.

Л. СТРУКОВА

Проектирование дидактического процесса в операционной среде «ЭСКИЗ»

Дидактические возможности компьютера справедливо интерпретировать как возможности, способные поддерживать и организовывать процессы преподавания и учения. Виды деятельности преподавателя (дидактическое проектирование, организация дидактического процесса, дидактический анализ) при компьютерном обучении сохраняются. Однако меняется их содержание, с одной стороны, и происходит их взаимопроникновение, с другой.

Говоря о взаимопроникновении, необходимо учитывать, что преподаватель на компьютерном занятии использует готовое программное средство, в котором уже поставлены определенные цели, отобрано содержание, определены требования к знаниям и умениям, реализованы конкретные методы, смоделированы определенные формы организации обучения. Использование программ учебного назначения вносит в учебный процесс опыт, идеи разработчика программы, т. е. происходит суперпозиция двух подходов.

Мы считаем, что проектирование эффективного программного средства невозможно без осознанного выбора и применения современной теории обучения в ее целостном виде, без учета специфических особенностей компьютера, с одной стороны, и владения компьютером как инструментом — с другой.

При разработке педагогических программных средств мы базируемся на трех основных положениях [1].

1. Компьютеризация обучения — это процесс, изменяющий существующие подходы к обучению, процесс создания новой системы, имеющей свои цели, теоретическую базу, свою методику организации, функционирования и оценки.

2. Общие цели обучения — развитие творческого мышления и индивидуальных способностей человека, формирование профессиональных знаний, умений, навыков, формирование потребностей и умений к самообучению, самооценке.

3. Наличие взаимосвязи общеобразовательных и профессиональных знаний, причем общеобразовательные знания выступают в качестве основы для развития творческого мышления, овладения общенаучными методами.

Дидактической основой, адекватной

этим исходным позициям, является теория проблемного обучения [2]. Вопрос выбора и сознательного применения тех или иных методов обучения, выбора способов организации обучения — вопрос творчества преподавателя, основанного на конкретных его знаниях. Научить его проектированию дидактического процесса — одна из сложных задач педагогики.

Мы попытались создать такое программное средство, которое бы помогло преподавателю выбрать наилучший метод и соответствующую форму организации обучения с учетом целей обучения, уровня подготовленности учащихся, профессиональной подготовки преподавателя и содержания учебного материала, а также позволило бы проиллюстрировать на конкретном содержании реализацию метода и формы обучения в виде готовой программы учебного назначения.

Наилучшим средством для разработки такого вида программ являются, на наш взгляд, инструментальные программные средства, которые рассчитаны на пользователей, не являющихся профессионалами в компьютерной области.

Нами была выбрана в качестве средства операционная среда для преподавателей и студентов «Эскиз», версия 1.3, разработанная в НИИ ВО АПН СССР.

Система предназначена как для преподавателей, так и для учащихся. Различие состоит лишь в последовательности их работы. Остановимся подробнее на особенностях работы обеих групп пользователей.

Известно, что выбор метода обучения в наибольшей степени зависит от четырех факторов: дидактической цели, особенностей содержания учебного материала, учебных возможностей учащихся, методической подготовленности преподавателя [3]. Этим в большей степени и определялась последовательность этапов работы преподавателя с системой.

1-й этап. Тестирование подготовленности преподавателя.

Тест проверяет наличие базовых знаний в области теории проблемного обучения. Тест включает вопросы, условно разделенные на три уровня. Третий уровень — узнавание объектов, свойств,— требует от преподавателя знания определений, правил. Второй уровень предполагает репродуктивные действия путем самостоятельного воспроизведения по памяти. Первый уровень содержит вопросы, знание которых отражает владение проблемным обучением как теорией и требует от преподавателя продуктивных действий для трансформации уже имеющихся знаний.

В зависимости от результата преподаватель может быть направлен к этапу ликвидации пробелов в опорных знаниях и умениях либо к этапу выбора метода обучения.

2-й этап. Ликвидация пробелов в опорных знаниях и умениях.

Для этих целей реализована контролирующая программа с элементами обучения. Программа содержит задания трех уровней. Обучение по программе начинается с заданий третьего уровня и заканчивается самым сложным — первым. В случае ошибочных ответов или частично правильных дается полный правильный ответ. Затем преподаватель переходит к повторному тестированию.

3-й этап. Тестирование подготовленности преподавателя.

4-й этап. Выбор метода обучения.

Каждый метод обучения реализуется через сочетание различных бинарных методов, объединяющих в себе два вида деятельности — преподавание и учение. Основные правила бинарных методов формулируются исходя из трех главных функций преподавания и трех главных функций учения. Для преподавания это: а) передача учащимся информации; б) побуждение к учебной деятельности; в) управление процессом учения. Для учения: а) восприятие учебной информации; б) усвоение новых и повторение ранее усвоенных понятий и способов действий; в) применение знаний.

В процессе диалога строится логическая цепочка из следующих факторов: дидактическая цель — характер учебного материала — уровень обученности учащихся, которая и определяет метод обучения. Каждый фактор имеет несколько уровней: высокий, средний, низкий. Дидактическая цель высокого уровня — усвоение знаний на уровне творческого применения, среднего уров-

ня — усвоение знаний на уровне применения по образцу, низкого уровня — усвоение знаний на уровне воспроизведения. Выбор уровня предыдущего фактора определяет выбор уровня следующего фактора, т. е., определив дидактическую цель высокого уровня, необходимо определить характер учебного материала высокого и среднего уровней, так как данная цель не может быть достигнута на содержании, где соотношение новых и опорных знаний очень мало.

Определив общий метод обучения, бинарные методы обучения, а также определив приемы и способы их реализации, преподаватель переходит к следующему этапу.

5-й этап. Выбор формы организации обучения, соответствующей методу обучения.

Каждой группе методов обучения соответствует своя группа форм организации обучения. Например, группе продуктивных методов, к которой относятся диалогический, эвристический и исследовательский методы, соответствуют следующие формы организации обучения: эвристическая беседа, дидактическая игра, творческая самостоятельная игра, проблемный эксперимент. Каждая форма реализуется с помощью определенной совокупности способов и приемов деятельности преподавателей и учащихся. Например, дидактическая игра как форма организации обучения требует от преподавателя сообщения правил и условий игры, выбора ролей, управления игрой. От учащегося — последовательного решения проблемы, эмоционального общения.

Методы преподавания и учения, а следовательно и их способы и приемы, реализуются в педагогическом программном средстве с той лишь разницей, что методы преподавания — через структуру программы, а методы учения — через ее состав. Поэтому выбор метода и формы обучения определяет конкретную учебную программу со специфическим составом и структурой. Программы, разработанные для этой системы, построены на химическом содержании.

6-й этап. Работа с педагогическим программным средством.

В качестве примера приведем программу, реализующую общий метод — диалогический, бинарный метод — стимулирующе-поисковый в форме дидактической игры.

Программа построена по принципу путешествия. Представьте, что вы на сказочной поляне. Всего несколько шагов — и вы за белокаменной стеной сказочного града. Каждый шаг — вопрос. Правильный ответ дает возможность сократить путешествие и увеличивает количество очков. Все вопросы логически соединены между собой и охватывают одну тему по химии.

К преимуществам системы относится мощная информационная поддержка. Преподаватель на любом этапе диалога может воспользоваться помощью или пояснениями, которые содержат необходимый фактологический материал. Кроме того, преподавателю будет предъявлена схема, где цветом будет выделен тот путь, который совершил преподаватель, выбирая метод и форму организации обучения.

Для учащихся последовательность работы с системой строится следующим образом.

1-й этап. Тестирование уровня подготовленности учащихся.

Были выделены три уровня обученности учащихся: высокий, средний, низкий. Критерием принадлежности к одному из этих уровней является коэффициент соотношения опорных и новых элементов знания. Для каждого уровня были определены новые и опорные знания и сформулирована совокупность заданий, которые и составили содержание теста.

2-й этап. Ликвидация пробелов в опорных знаниях и умениях.

Программа для ликвидации пробелов построена по принципу от простого к сложному, что позволяет наиболее слабым учащимся подняться на более высокий уровень. Затем система направляет учащегося на повторное тестирование.

3-й этап. Тестирование уровня подготовленности учащихся.

4-й этап. Работа с педагогическим программным средством.

В зависимости от уровня подготов-

ленности система порекомендует конкретное педагогическое программное средство. Диапазон программных средств достаточно широк: от контролирующих программ с элементами обучения до игровых программ, требующих от учащегося продуктивных действий.

В заключение отметим, что система разработана для IBM PC-совместимых машин, работает в среде DOS 2.0 и выше. К недостаткам системы можно отнести зависимость системы от операционной среды «Эскиз», что накладывает определенные ограничения на объем оперативной памяти и емкость винчестера.

Литература

1. *Шакирова Д. М., Струкова Л. А., Суфиярова Ф. Р.* Системный подход к компьютеризации обучения в профессиональной школе // Создание автоматизированных систем и комплексов для высшей школы на базе персональных ЭВМ: Материалы XIII Советско-французского семинара. Т. I. Казань, 1988. С. 45—52.

2. *Махмутов М. И.* Проблемное обучение: основные вопросы теории. М.: Педагогика, 1975. 368 с.

3. *Махмутов М. И.* Методы проблемно-развивающего обучения в средних профтехучилищах. М., 1983. 63 с.

А. ГУБАНОВ, В. ГУЗЕЕВ
МОПИ им. Н. К. Крупской

Проблемы разработки ЭС для профориентации и профотбора

В новых социально-экономических условиях, складывающихся в стране, необходимость массовой работы по профориентации и профотбору становится все более насущной. Естественно воспользоваться известными тестовыми методиками, накопленными прикладной психологией. И вот здесь взявшегося за эту задачу специалиста поджидают многочисленные трудности. Назовем среди них две: как выделить параметры личности, наиболее значимо влияющие на профессиональный отбор, и как упростить обработку результатов тестирования по батарее тестов.

Авторы, вступив на эту стезю, довольно быстро убедились, что при массовых обследованиях без компьютера задача практически неразрешима. Но, сделав батарею тестов на машине (в программировании участвовала И. А. Пинчук и авторы), так, как это делают обычно,— автоматическое предъявление стимульного материала с ручной обработкой, и проведя серию профориентационных обследований мы поняли, что необходима автоматизация интерпретации результатов. Итак, мы пришли к проблеме разработки экспертной системы психодиагностики для профориентации и профотбора. Использование экспертных систем в значительной степени позволяет снять остроту вопроса, обеспечив всех желающих научно обоснованными рекомендациями относительно выбора профессиональной деятельности на основе анализа индивидуальных возможностей каждого.

Вернемся к нашим проблемам. Первая — сознание оптимальной батареи психодиагностических методик. В значительной степени ее решение зависит от того, какие конкретные задачи ставятся перед разрабатываемой экспертной системой: либо она позволяет дать рекомендации пользователю относительно одной, отдельно взятой профессии, либо позволяет ему сориентироваться в многообразии существующих профессий. Поскольку в первом случае проблема решается конкретным анализом требований к психологическим возможностям человека, предъявляемым данной профессией, постольку состав психодиагностических методик здесь будет определяться относительно узким набором профессионально значимых для этой области деятельности психологических качеств. Во втором случае, прежде чем перейти к непосредственному подбору методик, необходимо каким-то образом разбить все многообразие профессий на классы. Профессии, относящиеся к одному классу, должны предъявлять примерно одинаковые требования к психологическим качествам человека. Только после разработки такой классификации можно переходить к подбору психодиагностических методик.

В разработанной нами экспертной системе эта проблема решалась следующим образом. При составлении батареи методик мы исходили из задачи ориентации пользователя во всем множестве имеющихся профессий. Поэтому в батарею были включены семь опробованных и достаточно хорошо известных психодиагностических методик, позволяющих дать сравнительно всестороннее психологическое описание человека. Это описание включает в себя характеристику его психофизиологических возможностей, уровня развития важнейших психических функций, характеристику направленности мотивации в совместной и профессиональной деятельности, особенности характера и поведения.

Первая из входящих в батарею методик (СВП) предназначена для выявления таких психофизиологических характеристик, как сила нервной системы, скорость и устойчивость реакции. Процедура состоит в том, что тестируемому предъявляют на экране монитора световые стимулы в виде белого квадратика на черном фоне с временными интервалами, принимающими случайные значения от сотен миллисекунд до нескольких секунд. От испытуемого требуется в ответ на появление стимула как можно быстрее нажать определенную клавишу. Выдается сто световых стимулов, и фиксируется время реакции на каждый из них. Сила нервной системы определяется как выраженное в процентах отношение суммарного времени реакции на первые тридцать стимулов к суммарному времени реакции на последние тридцать. Очевидно, чем ниже сила нервной системы, тем ниже этот показатель, так как возрастает время реакции на последнюю серию стимулов из-за значительного переутомления. Время реакции определяется как среднее время реакции по всем предъявленным стимулам, а устойчивость реакции обратно пропорциональна дисперсии времени реакции. Очевидно, что, чем реакция устойчивее, тем меньше дисперсия времени реакции и больше рассматриваемая величина.

Вторая методика (ОБРАЗ) предназначена для определения уровня развития образной памяти. Тестируемому на экране поочередно предъявляют 15 геометрических изображений на три секунды каждое. После этого ему вновь предъявляют эти же изображения, но перемешанные с пятнадцатью новыми. Нажатием на определенную клавишу тестируемый должен ответить, видел ли он показываемое изображение в первой серии. Процент правильно опознанных изображений указывает на уровень развития образной памяти у тестируемого

Третья методика (РУДИК) предназначена для выявления уровня развития мышления. Она позволяет оценить логичность мышления на арифметическом материале. Процедура состоит в предъявлении тестируемому на экране числовых последовательностей, в которых пропущены в разных местах по два соседних числа. От тестируемого требуется вставить эти числа так, чтобы не нарушалась логика построения ряда. Допускается ввод чисел в любом порядке. Всего дается двадцать числовых рядов, а время выполнения задания ограничено десятью минутами. Количество правильно восстановленных рядов указывает на уровень развития логичности мышления.

Тест КАРТА ИНТЕРЕСОВ предназначен для выявления интересов тестируемого по полю из 29 направлений профессиональной деятельности — таких, как геология, медицина, строительство, авиация и т. д. Опросник данной методики содержит 174 утверждения, по шесть на каждое направление. Процедура проста: поочередно предъявляется каждое из утверждений, а тестируемый должен по пятибалльной шкале указать, в какой мере он хотел бы заниматься тем, что описано в утверждении. По каждому направлению подсчитывается суммарный балл. Ясно, что он будет максимальным у наиболее предпочитаемых направлений профессиональной деятельности. На экран выводятся в виде гистограммы баллы по всем направлениям в двух полуплоскостях: предпочтительные профессии — вверх зеленым, нежелательные — вниз красным.

Пятая методика — это компьютеризированный вариант теста ОРИЕНТИРОВОЧНАЯ АНКЕТА, определяющего доминирующую направленность мотивации человека в совместной деятельности на себя, на общение, на решение задачи.

Компьютеризированный вариант одного из распространенных ТЕСТОВ АЙЗЕНКА (шестая методика) позволяет описать личность тестируемого по таким параметрам, как экстраверсия и нейротизм, а также содержит шкалу, позволяющую оценить, насколько тестируемый был искренен, отвечая на вопросы тестов.

Наконец, седьмая методика представляет собой компьютерную версию ТЕСТА КЕТТЕЛЛА, который позволяет описать личность испытуемого по 16 параметрам, и широко используется практическими психологами.

В целом наша экспертная система осуществляет исследование личности по 28 параметрам, список которых приводим полностью: 1. Время реакции. 2. Устойчивость реакции. 3. Сила нервной системы.

4. Логичность мышления. 5. Качество образной памяти. 6. Профессиональный интерес. 7. Степень ориентации мотивации в совместной деятельности на себя. 8. Степень мотивации в совместной деятельности на задачу. 9. Степень ориентации мотивации в совместной деятельности на взаимодействие. 10. Экстраверсия — интроверсия. 11. Нейротизм. 12. Достоверность тестирования. 13. Открытость. 14. Интеллект. 15. Стремление к доминированию. 16. Эмоциональная стабильность. 17. Беспечность. 18. Добросовестность. 19. Смелость. 20. Чувствительность. 21. Подозрительность. 22. Аутизм (мечтательность). 23. Проницательность. 24. Гипотимия (чувство вины). 25. Гибкость поведения. 26. Самодостаточность. 27. Самоконтроль. 28. Фрустрированность (напряженность).

Практика профотбора и профориентации показала, что наиболее удачным решением проблемы классификации множества существующих профессий является выбор предмета труда в качестве основания. В этой классификации, широко используемой психологами, выделяют пять основных групп профессий. К первой группе, условно называемой «человек — техника», относятся те профессии, где в качестве предмета труда выступают сложные технические объекты: слесарь, ткачиха, водитель, инженер-механик, пилот и т. д. Во вторую группу «человек — знаковая система» включаются профессии, в которых предмет труда выступает в знаковой форме, например: наборщик, кассир, программист. В третью группу «человек — художественный образ» попадают профессии, имеющие дело с художественным образом или его частью: профессии маляра, парикмахера, фотографа, музыканта и др. К группе «человек — природа» относятся профессии, связанные с животными или растениями, например: зоотехник, ветеринар, агроном, пчеловод и многие другие. Наконец, профессии пятой группы имеют непосредственное отношение к людям — предметом труда является человек; называется группа «человек — человек». Понятно, что сюда относятся педагоги, врачи, продавцы и т. д.

Очевидно, что общий предмет труда в каждой группе профессий предъявляет примерно одинаковые требования к психологическим характеристикам человека, пытающегося овладеть одной из них. Поэтому, отталкиваясь от предложенной классификации, можно подойти к решению проблемы выделения профессионально значимых психологических качеств. По-видимому, среди них можно выделить две группы. В первую войдут более или менее универсальные качества, сформированность которых является необходимым условием для овладения большинством имеющихся профессий. Вторую группу составят специфические качества, сформированность которых является необходимым условием успешного овладения какой-нибудь одной профессией или группой сходных профессий, имеющих, например, общий предмет труда.

Опрос экспертов показал, что из 28 рассматриваемых параметров личности при оценке возможностей тестируемого в овладении специальностями одного из пяти выделяемых классов профессий 22 параметра следует определить как профессионально значимые, а шесть — как вспомогательные. Из 22 профессионально значимых параметров пять (такие, как самоконтроль, профессиональный интерес, добросовестность) оценены как универсальные, 11 — как специфические (например, для профессий типа «человек — человек» это экстравертированность, ориентация в мотивации совместной деятельности на общение и др.; для профессий группы «человек — знаковая система» — логичность мышления; для профессий группы «человек — художественный образ» — чувствительность и т. д.); оставшиеся шесть параметров носят промежуточный между универсальными и специфическими характер, так как определяют пригодность к двум-трем группам профессий.

Теперь перейдем к вопросам технической реализации разработанных методик. Начнем с тех ограничений, которые мы себе установили. Прежде всего, техническая база — «Ямаха-2» в локальной сети класса. Нам нужна максимальная простота эксплуатации системы в сети, возможность ее уточнения и доводки по мере уточнения норм при росте исследуемой выборки. В связи с этим мы согласились работать в рамках вшитого Бейсика, базовой сетевой программы и без использования операционной системы, загружаемой с дискеты. Не исключали возможность использования промежуточных твердых носителей информации. При всех этих барьерах, установленных вполне сознательно, проблема оказалась разрешимой.

Среди четырех основных форм представления знаний экспертов, рассматриваемых в литературе по искусственному интеллекту, мы остановились на правилах пролукций. Это естественный выбор, если учесть, что психологи имеют частично формализованные процедуры интерпретации тестовых баллов. На выходе мы хотели бы дать испытуемому не только рекомендации по выбору профессии, но и достаточно развернутую характеристику его личности. Поскольку иссле-

дование ведется по большому числу параметров, то заложить в базу данных все варианты готовых характеристик, как это сделано в компьютерном варианте теста Личко, не представляется возможным — общее число вариантов далеко превышает 2^{16}. Остается только модульный принцип построения характеристики. В базе данных содержится 50 текстовых модулей, отбор и конструирование которых сами по себе оказались непростой задачей инженерии знаний. Таким образом, в базе данных содержатся 50 небольших текстовых файлов. База знаний включает правила двух видов: непосредственно по значениям параметров личности, компонующие ее характеристику текстами из базы данных, и более сложные, формирующие рекомендации по выбору профессии — эти правила работают с группами параметров и числовыми множествами, представляющими собой второй результат работы правил первого вида. Все правила реализованы в виде алгоритмических конструкций «альтернатива», иногда с глубоким вложением.

Мы отказались от включения в систему свойственного другим экспертным системам механизма пояснений, так как тестируемому нет нужды знать, как психолог делает свои выводы, а профессионал, который, может быть, будет работать с системой, имеет непосредственно все численные значения параметров. Впрочем, при остром желании все правила можно извлечь прямо из текста программы, которую мы не считаем нужным защищать ни от чтения, ни от копирования.

Итак, рассмотрев возможные подходы к решению проблем, возникающих при разработке экспертных систем для профориентации и профотбора, опишем возможности созданной нами экспертной системы ППДМ (пакет психодиагностических методик). Она осуществляет автоматическое тестирование с использованием описанной батареи психодиагностических методик и автоматическую интерпретацию полученных данных по 28 приведенным выше параметрам. Результаты интерпретации выводятся на печать в виде связного текста объемом в две — четыре страницы и включают в себя характеристику достоверности полученных данных, характеристику нервной системы, памяти, мышления, мотивации совместной деятельности, личностную характеристику и рекомендации по профессиональному выбору. Эти рекомендации составляются на основании методики КАРТА ИНТЕРЕСОВ, позволяющей оценить, к какой из пяти названных групп профессий испытывается максимальная склонность, и данных по профессионально значимым для этой группы профессий параметрам — универсальным, специфическим, промежуточным. В рекомендации указывается, к какой из групп профессий есть наибольший интерес и какие из психологических особенностей тестируемого могут затруднить ему овладение специальностью из указанной группы профессий, если такие особенности имеются. Пример выдаваемого документа приведен после списка литературы.

Управляющая программа системы предлагает два варианта работы: на одиночной машине с дисководом и в сети компьютерного класса. В первом случае вся процедура полностью автоматизирована от первой заставки на экране до последнего восклицательного знака на бумаге. Во втором случае все методики, входящие в систему, могут рассматриваться как автономные программы и при желании или необходимости использоваться поодиночке — каждая из них выдает на экран названия и численные значения всех параметров, в том числе и тех, которые не обрабатываются экспертной системой в силу их малой значимости, но отслеживаются собственно тестами. Для обработки всех параметров часть из них (названные 28) переносятся с экрана на промежуточный носитель — специальный бланк, также печатаемый на принтере, и после прохождения всей батареи тестов вручную вводятся в машину с дисководом (ввод занимает примерно полминуты). Бланки в нужном количестве печатаются управляющей программой. Она же отслеживает рассылку тестов по сети.

Опытная эксплуатация системы показала ее полезность для психолого-диагностических служб, центров профориентации и переквалификации, школ и педагогических вузов. За справками можно обращаться в Молодежный центр «Мир» при МОПИ им. Н. К. Крупской по адресу: *107005 Москва, ул. Радио, 10-а.*

В. УРНОВ
генеральный директор AIST AB

Второй этап

В настоящее время практически завершен первый этап информатизации образования и сферы управления в рамках экспериментального «Московского проекта». В московских школах установлены 130 компьютерных классов IBM-совместимой техники.

Проанализировав состояние дел в этих школах и оценив возможности использования поставленной техники, Московский департамент образования совместно с НЦПСО, фирмой IST и фирмой AIST AB предлагают программу второго этапа этого проекта.

Учитывая зарубежный опыт информатизации образования, мы пошли по пути создания высокоэффективной технологии разработки программно-методических сред для учителя и ученика, которая описана в «ИНФО» № 5 1991 г. Эта технология на первом этапе проекта дала положительные результаты — многие московские учителя эффективно работают в среде TeachCAD и создают собственные разработки, реально внедряя компьютеры в учебный процесс. По итогам первого этапа мы считаем, что эта уникальная технологическая линия в образовании должна иметь развитие как в нашей стране, так и среди стран-участниц проекта.

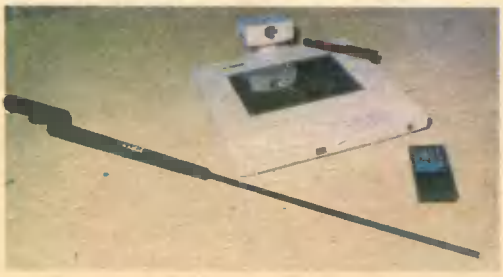
Для координации совместной работы и максимального использования ресурсов всех участников второго этапа в Москве организована фирма AIST AB.

По международному размаху проект, назовем его условно «Этап-II», воистину безграничен!

В Японии (фирма Sharp) производится уникальная панель, позволяющая выводить при помощи обычного эпидиаскопа на белый экран более 20 000 оттенков весьма и весьма качественного VGA изображения, которое может достигать по диагонали 4-х метров с обычного монитора или, что тоже пока уникально, с бытового видеомагнитофона.

В США фирмой Proxima выпускается устройство Cyclops, сканирующее это цветное изображение и позволяющее при помощи лазерной указки или дистанционного беспроводного лазерного устройства (lazerpointer) управлять им на огромном экране, не подходя к компьютеру.

В Европе (Швеция, Бельгия и, что осо-



бенно радует, Россия) фирмами IST и AIST продолжаются работы над ядром вышеописанной аппаратной среды — программной средой TeachCAD.

Подробно останавливаться на этом языке программирования для непрограммиста-учителя я не буду (см. «ИНФО», № 5, 1991 г.), но несколько основных принципов приведу.

В руки учителя и ученика попадает предельно гибкое «непрограммистское» средство создания собственных учебных курсов с их немедленной интерактивной демонстрацией на класс любых размеров без потери качества и скорости изображения полноценного компьютера, с дополнительной возможностью подключения любого видеоизображения при помощи видеомагнитофона. В ближайшей перспективе — возможность использования сканеров для получения цветных изображений и распознаваемых текстов на не более мощных, чем AT, компьютерах и CD-ROM для хранения высококачественных изображений.

Это полная технология замкнутого цикла производства профессиональных пакетов обучающих курсов школами — учителями и учениками — с немедленным реальным внедрением и методико-психологической «обкаткой» прямо в школе, не говоря уже о том, что подобное решение полностью снимает массу чисто медицинских проблем (зрение, психологическая нагрузка) у ребятишек, вынужденных в ином случае проводить достаточно много времени, уставившись в экран монитора.

Самое главное здесь то, что сама идея позволяет наполнить новым смыслом уроки информатики на любом уровне. Ученики и учителя информатики и других предметов получают уникальную возможность: не теряя времени на изучение языков программирования, творчески работать, получая профессиональный результат. Результат, необходимый не только для работы в своей школе, но и в других школах (напомним, TeachCAD — полный компилятор); результирующий EXE-файл может быть стандартным способом использован на любой IBM-совместимой технике). А так как исходный текст программ TeachCAD в терминах «фон», «картинка», «текст», «действие» тоже сохраняется, то при наличии у адресата-учителя этой технологии сохраняется возможность быстро внести свои собственные изменения и/или дополнения в предложенный ему курс.

Таким образом, описанная выше технология позволяет всем заинтересованным сторонам — от учителя и ученика до профессионального программиста — включиться в решение поставленной сверхзадачи — задачи информатизации образования.

Уважаемые читатели!
Новый телефон редакции
272-26-71

М. КОВТУН

Программирование в системе Express Pascal

Еще одна простая программа

Попробуем создать еще одну простую программу — решения квадратного уравнения. Она будет вводить три коэффициента (a, b и c) и выдавать два корня уравнения. Никаких проверок (является ли уравнение квадратным, имеет ли оно вещественные корни) выполняться не будет.

```
var  a, b, c  : real;
              { коэффициенты уравнения }
     D, SD    : real;
          { дискриминант и квадратный }

          {              корень из него }
     x1, x2   : real;
                    { корни уравнения }
begin
  Write('Введите коэффициенты
                 уравнения a, b и c: ');
  Readln ( a, b, c );
  D := Sqr(b) - (4*a*c);
  SD := Sqrt(D);
  x1 := ( -b + SD ) / ( 2*a );
  x2 := ( -b - SD ) / ( 2*a );
  Writeln;  Writeln('Корни
              квадратного уравнения:');
  Writeln('    x1 = ', x1 );
  Writeln('    x2 = ', x2 );
end.
```

Введите программу в компьютер и попробуйте решить с ее помощью несколько уравнений (как это делать, рассказывалось в предыдущем номере).

А теперь разберемся, как она работает. Первые четыре строки описывают используемые переменные. Новое здесь — тип real (real — это предопределенный идентификатор, обозначающий вещественный тип). Значения переменных этого типа — вещественные числа (их еще называют действительными), т. е. такие, которые могут иметь дробную часть. Диапазон возможных значений вещественных — от 2×10^{-39} до 2×10^{38} (приблизительно); количество значащих десятичных цифр — около 12.

Продолжение. Начало см.: Информатика и образование. 1992. № 1, 2.

Напомним, что такое значащие цифры. Это понятие появляется при выполнении приближенных вычислений. Любой прибор, измеряющий физическую величину, дает некоторую ошибку, величина которой зависит от погрешности прибора. Например, если мы измерили вольтметром напряжение и получили величину 37,25 В, причем известно, что вольтметр имеет погрешность 0,1 %, то верить можно только первым трем цифрам результата — четвертая уже содержит ошибку. Истинное значение напряжения может быть 37,26 В, или 37,2373 В, или каким-нибудь еще — точного значения мы все равно не получим. Те цифры, которым можно верить в приближенном числе, и называются значащими.

В компьютере вещественное число представляется с помощью конечного числа цифр, поэтому только немногие числа будут представлены точно, большинство же — только приближенно. Ведь память компьютера состоит из конечного (хотя и очень большого) числа элементов, а на любом отрезке числовой оси имеется бесконечно много вещественных чисел. Поэтому, какой бы способ представления вещественных чисел мы ни выбрали, всегда найдется число, которое в данном компьютере данным способом не может быть представлено точно.

Способ представления вещественных чисел, используемый в системе Express Pascal, обеспечивает 12 верных цифр. Обратите внимание — 12 верных цифр, считая от первой значимой, а не после десятичной точки. Например, если вы получили в результате вычислений число 123456789.01200000, нельзя утверждать, что последние пять его цифр именно нули; истинное значение их может быть и другим, но компьютеру не по силам вычислить эти цифры. Четвертый после десятичной точки знак здесь уже может содержать ошибку. Но если результат есть 123.456789012, то верить можно девяти цифрам после десятичной точки.

Нужно заметить, что приближенные вычисления на компьютере — весьма тонкая и деликатная наука. Этой теме посвящено много серьезных математических исследований. Ведь в процессе вычислений ошибки могут *накапливаться*, и из-за этого ошибка в результате может намного превосходить ошибки в исходных данных. Хороший пример того, на какие подводные камни можно наскочить, дает наша программа решения квадратного уравнения.

Попробуйте решить уравнение с коэффициентами 1, 4, 3. Ответ будет точным: 1 и —3. Но попробуйте взять коэффициенты 1, 10000000, 1; корни будут — 10000000 и 0! Но ведь из теоремы Виета следует, что ни один корень квадратного уравнения с ненулевым свободным членом не может равняться нулю. Действительно, если мы вычислим оба корня с точностью до 20 знаков, то получим 9999999.99999990000000 и 0.00000010000000000000001000000. Если эти значения округлить до 12 знаков, то получим 10000000 и 0.0000001 — и именно такой результат нам хотелось бы получить. Не подумайте, что 0.0000001 — слишком маленькое число для компьютера и поэтому он не может не вычислить точно. Попробуйте решить уравнение с коэффициентами 1, 4e—18, 3e—36 (что означает такая запись, рассказано ниже) — результат получится абсолютно точным. Попробуйте чуть-чуть изменить коэффициенты 1, 3.999999999e—18, 3e—36 — соответствующим образом изменится ответ. Так что компьютер успешно справляется с гораздо меньшими числами. Откуда же возникает такая грубая ошибка при коэффициентах 1, 10000000, 1? Она не в программе и даже не в реализации вещественной арифметики. Плох выбранный алгоритм решения квадратного уравнения. Он будет давать заметную ошибку всегда, когда корни уравнения сильно различаются (абсолютная величина корней не существенна, важна именно величина отношения между корнями). Кстати, разработка алгоритма, который при любых коэффициентах квадратного уравнения вычислял бы его корни с точностью, обеспечиваемой машинной арифметикой,— задача на уровне курсовой работы студента, специализирующегося в области вычислительной математики.

После зарезервированного слова begin идут уже знакомые операторы вывода запроса и ввода коэффициентов. Но здесь мы будем вводить уже не целые, а вещественные числа. Правила записи вещественных чисел такие.

Если вещественное число не имеет дробной части, то оно может быть записано по правилам записи целых чисел.

Дробное вещественное число может быть записано как десятичная дробь, которая может начинаться со знака и в которой целая часть отделяется от дробной точкой (обратите внимание — точкой, а не запятой!). Если в записи вещественного числа присутствует точка, то перед ней и после нее должны быть цифры (нельзя написать, например, 2. или .01 — нужно писать 2.0 и 0.01 соответственно).

Любое вещественное число может быть записано в экспоненциальной нотации, т. е. в виде целого или дробного числа, за которым следует буква E (латинская; строчная или прописная) и целое число (возможно, со знаком). Экспоненциальная нотация означает следующее: число, стоящее перед буквой E, должно быть умножено на 10 в степени число, стоящее после буквы E. Например, 1.23e3 обозначает число 1230; 0.9e—5 обозначает число 0.000009. Экспоненциальная нотация предназначена для записи очень больших и очень маленьких чисел. Попробуйте сосчитать нули в числе 0.00000000000000000000093 — и тогда вы оцените преимущества записи 0.93e—20. В экспоненциальной записи часть числа, стоящая перед буквой E, называется *мантиссой*, а часть числа, стоящая после буквы E, называется *порядком*.

Внутри записи вещественного числа не могут появляться пробелы и другие разделители: например, запись «2.1 e 3» недопустима.

Примеры записи вещественных чисел:

```
  14299          16.9453    3e11
 +14299          —2517.0+2.311E—12
 —123456789123456789  0.0123    0.234e+21
```

В большинстве европейских стран и в США при записи десятичных дробей принято отделять дробную часть от целой не запятой (как у нас), а точкой. Это соглашение используется и во всех языках программирования. По мере распространения компьютеров и в нашей стране все чаще и чаще используется точка вместо запятой.

Как и в случае целых чисел, вещественные константы в программе записываются по тем же правилам, что и вводятся с клавиатуры.

В строках, описывающих переменные, находятся тексты, заключенные в фигурные скобки. Это комментарии. Они никак не обрабатываются компилятором и нужны для того, чтобы облегчить чтение и понимание программы человеком. Комментарий должен начинаться открывающей фигурной скобкой «{» и заканчиваться закрывающей фигурной скобкой «}»; внутри комментария могут стоять любые знаки, кроме закрывающей фигурной скобки. Комментарий можно поместить в любую точку Паскаль-программы, в которой может находиться пробел. Можно продолжить его на несколько строк; например, нашу программу мы могли бы начать с комментария

```
{       Эта программа
          вычисляет
    корни квадратного уравнения  }
```

Но мы рекомендуем все-таки ставить открывающую и закрывающую комментарий фигурные скобки в каждой строке и писать так:

```
{       Эта программа                }
{         вычисляет                  }
{  корни квадратного уравнения       }
```

Такая привычка позволит избежать ошибок наподобие следующей:

```
var  a, b, c : real;     { коэффициенты
                           уравнения
     D, SD   : real;     дискриминант и
                       квадратный корень
                                из него
     x1, x2  : real; корни уравнения }
```

Здесь описания переменных D, SD, x1, x2 попали в комментарий и стали невидимыми для компилятора. Такая ошибка очень неприятна: на первый взгляд все в порядке, но компилятор говорит, что идентификатор D не определен.

В комментарии мы рекомендуем включать сведения, необходимые для понимания того, что и как делает программа, но только те, которые не очевидны из текста программы. В частности, при описании переменных стоит указывать в комментарии, что будет в них находиться. Но, например, комментарий

```
write('Привет!');  {Вывести "Привет!"}
```

только загромождает программу.

После операторов ввода коэффициентов идут четыре строки, содержащие операторы присваивания. Эти операторы и вычисляют корни уравнения. Отличие этих операторов присваивания от тех, которые мы обсуждали раньше, только в том, что выражение, стоящее в правой части, является вещественным арифметическим выражением. Как и целое арифметическое выражение, вещественное строится из констант, ссылок на переменные и вызовов функций с помощью знаков операторов и круглых скобок. В вещественном выражении некоторое подвыражение может быть целым. Например:

```
var  k, m, n : integer;
     x, y, z : real;
begin
  z := (2*k + m div n) / (x * y);
end.
```

Здесь целым является подвыражение (2*k + m div n). В Паскале подвыражения вычисляются как целые до тех пор, пока это возможно; затем целое значение преобразуется в вещественное, и далее операции выполняются над вещественными числами.

В вещественных выражениях знаки операций **div** и **mod** могут появляться только внутри целых подвыражений. Для обозначения операции деления вещественных чисел используется знак «/». Если же операция «/» применяется к целым операндам, то они перед выполнением операции преобразуются в вещественные, после чего выполняется операция деления вещественных чисел.

В операторе присваивания в левой части может быть указана вещественная переменная, а в правой находиться целое выражение. В таком случае выражение будет вычислено как целое, а затем его результат будет преобразован в вещественное представление, и оно будет присвоено переменной, указанной в левой части.

Вообще, если в каком-то месте Паскаль-программы требуется вещественное значение, а указано целое, то целое значение автоматически преобразуется в вещественное. Обратное преобразование (вещественного значения в целое) никогда не выполняется автоматически. Таким образом, оператор присваивания, в левой части которого указана целая переменная, а в правой части находится вещественное выражение, является ошибочным. Однако при необходимости вещественное значение может быть преобразовано в целое с помощью стандартных функций.

Теперь о функциях. В Паскале они очень похожи на обычные математические: получают *аргументы* и вырабатывают *результат*. Вызов функции на Паскале записывается в такой форме:

имя функции (*аргумент*, ..., *аргумент*)

В качестве аргументов (или параметров) функции должны быть указаны выражения. Количество аргументов и их тип зависят от функции.

В первом из операторов присваивания нашей программы

```
D := Sqr(b) - (4*a*c);
```

используется функция Sqr. Она должна иметь один аргумент целого или вещественного типа; ее результатом является квадрат ее аргумента; тип результата будет целым, если аргумент был целым, и вещественным, если аргумент был вещественным.

Во втором операторе присваивания

```
SD := Sqrt(D);
```

используется функция Sqrt. Она должна иметь один аргумент вещественного типа; ее результат — квадратный корень из ее аргумента, он тоже имеет вещественный тип.

Обе эти функции являются *стандартными*, т. е. они созданы разработчиками компилятора и вы можете пользоваться ими, не заботясь об их описании. Вы можете описать в программе и свои функции; о том, как делать это, мы расскажем позже.

Обратите внимание на то, что аргументом функции может быть не только переменная, но и выражение. Например, вместо

первых двух операторов присваивания мы могли бы написать один:

```
SD := Sqrt ( Sqr(b) - (4*a*c) );
```

Мы не сделали так лишь потому, что собираемся в будущем усовершенствовать нашу программу, добавив проверку знака дискриминанта.

Еще одно замечание об операторах, вычисляющих корни. Можно было бы не вычислять отдельно квадратный корень из дискриминанта, а вместо этого последние два оператора присваивания заменить на

```
x1 := ( -b + Sqrt(D) ) / ( 2*a );
x2 := ( -b - Sqrt(D) ) / ( 2*a );
```

Это было бы хуже исходного варианта тем, что квадратный корень из дискриминанта вычислялся бы дважды (а операция извлечения квадратного корня довольно длинная). Конечно, в такой короткой программе, как наша, этого замедления работы вы не заметили бы; однако если бы программа должна была решить подряд тысячу квадратных уравнений, замедление было бы существенным.

Можно было бы еще ускорить нашу программу, вынеся в отдельный оператор вычисление произведения 2*a.

Последние три строки в разделе операторов нашей программы выполняют вывод результатов на экран. Сейчас вещественные числа будут выводиться в экспоненциальном формате и занимать 17 позиций: одна позиция — знак числа, 12 — мантисса, одна — буква «e», еще одна — знак порядка, две позиции — порядок.

Если вам не нравится такой формат вывода, можете изменить его. Для этого после любого параметра процедуры Write (и Writeln) можно поставить двоеточие и число. Эта комбинация задает количество позиций, отводимых на экране для выводимого значения. Если такое число не задано, процедура Write отведет ровно столько позиций, сколько необходимо для записи значения; если число позиций задано, то значение будет дополнено слева необходимым числом пробелов; но если указанное число позиций меньше необходимого, Write нарушит заданные рамки и напечатает значение целиком.

Такая возможность удобна для вывода большого количества чисел в виде аккуратной таблицы. Попробуйте, например, выполнить такую программку:

```
var  x11, x12, x13, x21, x22, x23,
                x31, x32, x33 : integer;
begin
  x11:=1;      x12:=31;     x13:=-111;
  x21:=22;     x22:=0;      x23:=1;
  x31:=12345;  x32:=-321;   x33:=76;
  writeln ( x11, x12, x13 );
  writeln ( x21, x22, x23 );
  writeln ( x31, x32, x33 );
end.
```

Числа, выводимые на экран, слипнутся, и вы не сможете ничего понять. Но замените операторы вывода

```
writeln ( x11:7, x12:7, x13:7 );
writeln ( x21:7, x22:7, x23:7 );
writeln ( x31:7, x32:7, x33:7 );
```

и на экране появится аккуратная таблица.

Кроме того, для вещественных чисел (и только для них) можно задать число десятичных знаков, поставив после числа позиций еще одно двоеточие, а после него — число десятичных знаков. Такой способ вывода удобен, когда заранее можно оценить порядок выводимых чисел. Например, если вы собираетесь решить квадратные уравнения, корни которых лежат в интервале от 0.1 до 10, и вам нужно знать три знака после точки, операторы вывода можно заменить на следующие:

```
Writeln('   x1 = ', x1:6:3 );
Writeln('   x2 = ', x2:6:3 );
```

Попробуйте и посмотрите, что получится.

Количество позиций и количество десятичных знаков может задаваться не только константами, но и произвольными целыми арифметическими выражениями. Это позволяет писать программы, способные изменять формат вывода в зависимости от исходных данных.

При выводе вещественных чисел всегда производится округление до последнего выводимого знака.

Теперь давайте посмотрим, как система будет реагировать на ошибки. Они могут встретиться в любой программе, и процесс их исправления (отладка) — обычный этап в разработке программы. Ошибки можно разделить на следующие категории:

- Ошибки в записи программы (синтаксические). Их замечает компилятор и сообщает о них.
- Ошибки, которые обнаруживаются системой при выполнении программы (например, деление на 0 или извлечение квадратного корня из отрицательного числа). Они диагностируются системой, и компилятор поможет вам в их обнаружении.
- Ошибки в алгоритме (например, вычисление дискриминанта по формуле Sqr(b)+2×a×c, а не Sqr(b+4×a×c).

Их можно обнаружить только путем внимательного анализа программы; компилятор здесь ничем помочь не сможет. (В самом деле, откуда компилятору знать, как должно решаться квадратное уравнение и, вообще, что данная программа решает квадратное уравнение?)

Здесь мы обсудим, как использовать компилятор для поиска ошибок первых двух категорий. Попробуем делать ошибки и смотреть, как на них реагирует компилятор. (Здесь будет описываться поведение системы Express Pascal; поведение системы Turbo Pascal очень похоже, поведение других систем может отличаться внешне, но по сути будет тем же).

Давайте сначала в первой строке нашей программы заменим двоеточие на точку с запятой

```
var  a, b, c  ; real;
                { коэффициенты уравнения }
```

и попробуем скомпилировать ее. Компилятор обнаружит ошибку и выдаст сообщение:

```
Error 62: ":" expected
Press Esc.
```

(«Ошибка 62: ожидается «:». Нажмите Esc»). После того как вы нажмете Esc, система автоматически перейдет в режим редактирования текста, и курсор в тексте будет установлен на знак «;». Все, что вам остается сделать,— исправить эту литеру и снова запустить компиляцию.

Несколько более сложным может оказаться поиск ошибки, вызванной искажением имени. Замените в первой строке программы b на w

```
var  a, w, c  : real;
                { коэффициенты уравнения }
```

и попробуйте скомпилировать программу. Компилятор выдаст сообщение об ошибке:

```
Error 73: Unknown identifier
Press Esc.
```

(«неизвестный идентификатор»). После нажатия Esc система перейдет в режим редактирования текста и курсор будет установлен на имя b в операторе Read ln (a, b, c). Действительно, идентификатор b не был определен; но чтобы найти истинную причину ошибки, придется вернуться к разделу описания переменных.

Особенно трудно находить ошибки, вызванные несбалансированностью зарезервированных слов **begin** и **end**. Обычно в хорошо написанной программе расстояние между парными словами **begin** и **end** может достигать нескольких десятков строк (в плохо написанной программе они могут оказаться еще дальше). Обнаружить это компилятор может, как правило, очень нескоро, зачастую только в конце программы. А найти причину ошибки можно только просмотрев все **begin** и **end**. Чтобы избежать таких ошибок, мы рекомендуем при наборе программы сразу после слова **begin** вводить слово **end**, а уж потом вставлять между ними операторы.

Теперь об ошибках, которые могут возникнуть в период выполнения программы. Первый их вид — ошибки в формате данных. Попробуйте, например, на запрос «Введите коэффициенты уравнения a, b, c:» ввести число с запятой вместо точки: «1 1,2 0,1». Внизу экрана появится сообщение:

```
Run-Time Error 15.  PC=....#:
                          Invalid number format
Press Esc.
```

(Ошибка времени исполнения 15. Счетчик адреса=...: Неправильный формат числа). Выдаваемое в этом сообщении значение счетчика адреса может быть использовано для поиска оператора, при выполнении которого была обнаружена ошибка (как это сделать, рассказано в документации). Конечно, в нашем случае сразу видно, что мы ошиблись при вводе; но при отладке сложной программы информация о том, при выполнении какого оператора возникла ошибка, может оказаться весьма полезной.

Другой вид ошибок, возникающих в период исполнения программы,— применение операций к недопустимым аргументам. Например, квадратный корень из отрицательного числа извлечь нельзя. Задайте нашей программе коэффициенты, при которых дискриминант будет отрицательным (например, 1 2 3), и будет выдано сообщение об ошибке:

```
Run-Time Error 16.  PC=....#:
                          Invalid argument
Press Esc.
```

(Неправильный аргумент). Это сообщение должно заставить вас проверить введенные коээфициенты. Если они таковы, что квадратное уравнение не имеет корней, то ничего не поделать. Но если вы видите, что корни должны быть (а программа не смогла их найти из-за ошибки в формуле вычисления дискримината, но вы этого пока не знаете), то следует искать ошибку в программе. Компилятор поможет найти оператор, при выполнении которого возникла ошибка,— это оператор извлечения квадратного корня из дискриминанта. Раз коэффициенты такие, что корни должны быть, а квадратный корень из дискриминанта не извлекается,— значит, программа неправиль-

но вычисляет дискриминант. После этого рассуждения вы анализируете оператор, вычисляющий дискриминант, и находите ошибку.

Мы рассмотрели здесь только несколько примеров возникновения ошибочных ситуаций. Начав работать с компьютером, вы встретите много других. Полный список сообщений об ошибках есть в документации; там же вы найдете рекомендации по их поиску и устранению. (Если же у вас нет документации потому, что вместо того, чтобы купить систему, вы просто скопировали ее у кого-то, то поделом вам.)

Продолжение следует.

А. ЖЕЛТОВ,
программист сектора «Корвет» научно-технического предприятия «Альтернатива»

Семейство «Автор» — гипермедиа для терминальных классов

Прошло достаточно много времени с тех пор, как первые КУВТ-86 и «Ямахи» пришли в школы; с тех пор многое изменилось. Изменился и подход к информатизации образования. Первоначальную установку на обучение программированию в школе постепенно вытеснил акцент на подготовку грамотного пользователя. Впрочем, действия руководства народного образования во многом непредсказуемы, и, как нам стало известно, есть планы введения (по крайней мере в московских школах) предмета «Программирование».

В любом случае широкой областью применения компьютеров в школе является информатизация общеобразовательных предметов. В последнее время за рубежом широко стали применять для этой цели т. н. системы гипермедиа (hypermedia). Познакомились с ними и наши преподаватели и ученики — в рамках «пилотного» (система LinkWay) и «московского» (система TeachCAD) проектов. Однако упомянутые системы реализованы только на IBM PC-совместимых компьютерах. В этом плане между пользователями такой техники и пользователями отечественных КУВТов наметился серьезный разрыв — ведь гипермедиа действительно позволяет создавать учебные программы на качественно новом уровне.

Сократить этот разрыв, а быть может, где-то и свести к нулю поможет проект «Автор», реализуемый нашим предприятием. Проект предусматривает создание систем гипермедиа для всех основных типов отечественных терминальных классов — КУВТ-86, УКНЦ и «Корвет», КУВТ на основе БК-0011 и БК-0011 М. Значительная часть проекта уже выполнена, и системы семейства «Автор» (оригинальное название AUTOR3) уже поставляются пользователям КУВТ-86, КУВТ-86 М и КУВТ УКНЦ. 01. Система «Автор-Корвет» подготавливается к поставке, в ближайшее время будут завершены работы по созданию аналогичной системы для УКНЦ.

Первоначально система «Автор» создавалась для разрешения довольно специфичной проблемы, возникающей при разработке учебных программ: учебная программа, обучающая или контролирующая, должна быть не только красиво и качественно написана, но еще и методически правильно построена. Программисты обычно не владеют методикой преподавания, а методисты не обладают достаточно высокой программистской квалификацией, необходимой для использования в программе графики, музыки, мультипликации. Собрать вместе двух таких специалистов — уже само по себе достаточно трудное дело; а ведь им еще надо найти общий язык...

Мы разрешили эту проблему так: программисты создают систему, позволяющую пользователю-непрограммисту создавать высокоэффективные педагогические программные средства, максимально использующие возможности компьютера, и таким образом делают свою часть работы; а методисты работают с этой системой и создают готовый программный продукт. Но, как часто бывает, после создания системы выяснилось, что область ее применения значительно шире, чем представлялось вначале.

Помимо тех возможностей, которые система предоставляет для комплексной информатизации практически любых учебных кур-

сов, ее можно использовать, скажем, для реализации педагогического принципа «Learning through teaching» (обучая других, обучаюсь сам) — подробнее см., например, статью «TeachCAD — союз ученика и учителя» в «ИНФО» № 5 за 1991 г. Кроме того, с помощью системы можно создавать базы графических и музыкальных фрагментов («электронные галереи» и «музыкальные шкатулки»), игры, основанные на принципе гипертекста (например, аналогичные «Larry» на IBM PC) и многие другие программы и программные системы.

Разработка «Автора» была начата в 1989 г., практически одновременно (и совершенно независимо) с разработкой системы «LinkWay» Лэрри Керайети для фирмы IBM. Коллектив разработчиков «Автора» (О. О. Войцехович, К. И. Воропаев и Д. А. Котов) ознакомился с LinkWay только в 1991 г.

Системы во многом похожи, но есть между ними и существенные различия. Прежде всего, «Автор» не содержит внутреннего языка, аналогичного языку Script. Большинство задач, для решения которых в LinkWay приходится писать программу на Script, в «Авторе» решаются с помощью стандартных средств. Наша система не содержит средств для редактирования текстовых полей в подготовленных с ее помощью программах; впрочем, необходимость в таких средствах вызывает сомнение. Кроме того, «Автор» построен по принципу «открытой архитектуры», и, если его стандартных возможностей для решения каких-либо задач окажется недостаточно, пользователь имеет возможность подключить свой собственный модуль, написанный на ассемблере либо на языке высокого уровня (в терминологии «Автора» такие программы именуются «кодовыми кадрами»).

Чтобы читатель получил более ясное представление о системе «Автор», рассмотрим сначала принципы, на которых основана система,— принцип гипертекстового построения программ и принцип WYSIWYG, тем более что их используют и LinkWay, и TeachCAD, и другие.

Гипертекстовое построение программ получило в последнее время довольно широкое распространение, однако общепринятого определения гипертекста пока не существует. Приведем одно из многих:

«Гипертекст представляет собой комбинацию текстов, созданную применительно к возможностям ЭВМ обрабатывать и отображать информацию. Гипертекст может отличаться от обычного текста порядком следования материала, элементы гипертекста могут размещаться в виде иерархического дерева или сети организации (он может иметь несколько уровней краткости изложения и детализации материала), способом представления материала (его использование может обеспечить воспроизведение движущихся и преобразующихся иллюстраций) и т. д.» [2].

Работу программы, организованной в гипертекстовой форме, можно сравнить с чтением энциклопедии. Если роман или статью обычно читают последовательно, страница за страницей от начала и до конца, то с энциклопедией работают, как правило, по-другому: открывают оглавление, находят нужную статью и изучают ее. Встретив ссылку на другую статью, изучают ее и т. д. В энциклопедии имеется множество перекрестных ссылок, и читатель (в зависимости от того, что и насколько подробно он желает изучить) выбирает из множества возможных путей получения информации один-единственный (аналогия заимствована из [4]).

Так же построена и гипертекстовая программа. Она состоит из множества отдельных модулей, причем порядок, в котором расположены эти модули, как правило, не имеет значения. При запуске программы исполняется первый модуль; в нем содержатся ссылки на один или несколько других. Если ссылка одна, то после его исполнения происходит переход на модуль, указанный в ссылке; если ссылок несколько, то переход осуществляется по какому-либо признаку, например по правильному либо неправильному ответу ученика на заданный вопрос: если ответ верен, то осуществляется переход, например, на модуль с именем «А», если неверен — то на модуль с именем «Б». Взаимодействие программы с пользователем (в приведенном случае — получение и анализ ответа ученика) в разных системах организовано по-разному; о механизмах взаимодействия с пользователем, заложенных в «Автор», будет сказано позднее.

Программные модули постоянно сменяют друг друга, как кадры на экране при просмотре кинофильма. Мы воспользовались этой аналогией и назвали модули системы «Автор» кадрами.

В настоящее время принцип гипертекста получил дальнейшее развитие. Теперь под гипертекстом понимают не только комбинацию текстов; современные гипертекстовые системы, называемые еще системами гипермедиа, кроме собственно текста поддерживают вывод графики (LinkWay, TeachCAD), а некоторые — еще и исполнение музыкальных фрагментов (AUTOR3 и «Автор-Корвет»). В системе «Автор-Корвет», кроме того, предусмотрены специаль-

ные средства для использования мультипликации.

Как показала практика, гипертекст весьма удобен для построения учебных программ. На его основе можно создавать не только чисто информационные программы с несколькими уровнями детализации материала, но и обучающе-контролирующие программы, в значительной мере подстраивающиеся под конкретного ученика, с возможностью более подробного изучения именно тех разделов курса, в которых знания ученика недостаточны.

Другой основной принцип, заложенный в систему «Автор»,— WYSIWYG (what you see is what you get — что вы видите, то и получаете; применительно к данному случаю — интерактивное программирование в образах конечной программы).

Как вы помните, «Автор» создавался для пользователей-непрограммистов, значит, использование в системе какого-либо внутреннего языка программирования сразу исключалось. Необходимо было найти такой механизм создания конечных программ, который был бы прост, удобен и эффективен. После многочисленных экспериментов он был найден — теперь процесс работы с «Автором» больше напоминает не программирование, а конструирование программы, сборку ее из составных частей.

Например, для того чтобы вывести какое-либо графическое изображение на экран, не требуется описывать процесс его появления с помощью операторов, как в языках программирования; достаточно сделать рисунок в графическом редакторе, затем при помощи системного редактора-компоновщика поместить его в нужное место на экране. Точно так же, в диалоге с пользователем, проходит весь процесс создания программы: формирование новых модулей-кадров, их объединение в структуры, задание условий, при выполнении которых происходит переход на другие кадры, и т. д.

Необходимо заметить, что системы, построенные по принципу WYSIWYG, отличают не только простота и удобство использования, но и то, что программа создается непосредственно в образах конечного результата. Говоря другими словами, вид экрана при проектировании программы практически не отличается от того, что мы видим во время исполнения готовой программы. Безусловно, это значительно облегчает процесс программирования.

Итак, пользователь системы «Автор» конструирует на экране дисплея свою программу, организованную по принципу гипертекста — в виде множества программных модулей-кадров, связанных друг с другом ссылками. Однако есть возможность создавать программы, порядок исполнения которых не является жестко заданным; специальные структуры управления позволяют организовывать диалог с пользователем и в зависимости от его действия изменять ход выполнения программы.

Основным средством общения с программой является переход от одного кадра к другому по положению стрелки-указателя на экране. При создании каждого модуля-кадра на экране можно выделить несколько зон (мы условно назвали их ответами), и за каждой из них закрепить имя любого из кадров, составляющих программу. При исполнении программы на экране дисплея появляется стрелка-указатель, которую можно перемещать при помощи манипуляторов «мышь», «джойстик» или при помощи клавиш управления курсором. При нажатии на кнопку манипулятора либо клавишу **ВВОД** программа проверяет, находится ли стрелка-указатель в какой-либо из заданных областей («ответов»), и если это так, то запускается программный модуль-кадр, имя которого было закреплено за данной областью в процессе создания программы.

Каждому ответу можно присвоить две степени активации: если задана одна из них, то переход будет происходить только тогда, когда указатель находится в данной области экрана и нажата кнопка манипулятора либо клавиша **ВВОД**; если задана другая, то для перехода достаточно, чтобы указатель просто попал в заданную область-ответ.

Как пример использования данного типа интерфейса можно представить себе следующую учебную программу по географии: первоначально на экране компьютера находится карта мира; ученик помещает указатель на какой-либо континент и подтверждает свой выбор, нажимая на клавишу. Происходит переход на кадр, выводящий на экран подробное изображение континента; теперь можно выбрать любую из стран и получить на экране карту страны и подробную информацию о ней: население, площадь, форма управления и т. д.

Такой гипертекстовый компьютерный атлас достаточно просто создается средствами системы «Автор»; основная трудность заключается в создании изображений континентов и стран при помощи графического редактора и вводе информации о странах в программу. Впрочем, если воспользоваться принципом «Learning through teaching» и доверить часть работы ученикам, то можно и достаточно быстро создать программу, и закрепить у учеников навыки работы с графическим редактором (да и знания по географии тоже).

Второй тип стандартного интерфейса пользователя системы «Автор» — переход по введенной строке символов. Это довольно традиционный тип интерфейса, но, как показала практика, потребность в нем возникает довольно часто.

Чтобы оценивать знания ученика, накапливать статистическую информацию в процессе работы, производить несложные вычисления и т. д., в системе «Автор» имеются специализированные переменные, именуемые статистиками. Используя структуры управления (статистические кадры в AUTOR3 и структура CASE в «Автор-Корвет»), статистики можно изменять и отслеживать их значения. Если значения статистик удовлетворяют заданным при создании программы требованиям, порядок исполнения программы соответствующим образом изменяется.

В состав «Автора», кроме собственно редактора-компоновщика, при помощи которого создаются программы, входят и другие инструментальные средства, предназначенные для подготовки графических и музыкальных фрагментов, кодовых вставок и т. д. Поставочный комплект системы, как правило, состоит из:

редактора-компоновщика, предназначенного для создания готовых исполняемых программ с расширением .COM, .BIN и .SAV соответственно для «Корвета» и остальных типов КУВТов, использующих операционную систему RT-11;

графического редактора, предназначенного для создания используемых в программе графических фрагментов;

системы подготовки и распечатки на принтере графических фрагментов;

музыкального редактора;

набора готовых программных модулей (кодовых кадров), предназначенных для решения наиболее часто встречающихся задач;

системы программирования МИКРО11, применяемой для создания пользователем кодовых кадров (только КУВТ-86 и его разновидности);

систем программирования Express Pascal, ассемблер и, возможно, Пролог — для создания кодовых вставок в системе «Автор-Корвет»;

системы подготовки спрайтов (графических примитивов для мультипликации) (только в системе «Автор-Корвет»).

Состав системы «Автор» для УКНЦ до конца еще не определен, но будет в целом аналогичным приведенному выше.

Надеюсь, цель, которую я ставил перед собой, достигнута и читатели получили представление о гипермедиа вообще и системах семейства «Автор» в частности. Если есть желание узнать о них больше, рекомендую ознакомиться с литературой, список которой приведен ниже.

ЛИТЕРАТУРА

1. *Воропаев К. И.* Система «AUTOR3»: Техническое описание и руководство пользователя. Барнаул, 1992.

2. *Нельсон Т.* Информационные системы будущего // Информационный поиск: Сб. материалов. М.: Воениздат, 1970.

3. *Субботин М. М.* Новая информационная технология: создание и обработка гипертекстов // НТИ. Сер. 2. 1988. № 5.

4. Введение в систему LinkWay: (Комплект документации на ПО, поставляемое в рамках «пилотного проекта»).

А. ДЕВЯТОВ

Программирование 9-игольчатых матричных принтеров семейства Epson

Данная статья предназначается в первую очередь пользователям ПК «Корвет», однако будет полезна всем, имеющим дело с Epson-совместимыми и близкими к ним принтерами. Полное описание системы команд принтера Epson довольно велико, поэтому в журнальном варианте оставлены лишь наиболее употребимые команды.

Матричные принтеры являются неотъемлемой частью любого аппаратно-программного комплекса на базе персонального компьютера. Наиболее доступны и поэтому распространены принтеры, головка которых содержит 9 иголок.

Текст или графическое изображение, формируемые матричным принтером, состоят из горизонтальных строк, набранных из близко расположенных вертикальных колонок, состоящих в свою очередь из отдельных точек.

Несмотря на внешнюю простоту, принтер является весьма сложным устройством. По сути это специализированная микро-ЭВМ с собственным микропроцессором, ПЗУ и ОЗУ, каналом ввода данных и управления, внешними исполнительными устройствами в виде двух шаговых двигателей и печатающей головки.

Признанным лидером в сфере разработки и производства матричных принтеров является фирма Seiko Epson (Япония). Именно ей удалось в свое время разработать удачный набор команд для программирования режимов таких принтеров и продать большое количество своих принтеров на рынке компьютерной техники.

В результате соблюдение протокола команд фирмы Epson стало почти обязательным для всех производителей принтеров, претендующих на сбыт своей продукции на рынке.

Ниже описаны набор команд принтеров семейства Epson, способы задания режимов печати из программ пользователя, написанных для ПК «Корвет».

Характеристики принтеров семейства Epson

Матричные принтеры характеризуются следующими основными техническими параметрами: скоростью печати, размером основного буфера печати, наличием или отсутствием режима печати, близкого по качеству к шрифту портативной пишущей машинки, наличием или отсутствием «зашитого» в ПЗУ принтера знакогенератора русского алфавита (кириллицы).

В табл. 1 приведены основные технические характеристики распространенных в нашей стране 9-игольчатых матричных Epson-совместимых принтеров.

шрифт напоминает шрифт портативной пишущей машинки.

Аппаратно русифицированные принтеры с записанным в ПЗУ знакогенератором кириллицы обеспечивают печать смешанных текстов с английским и русским алфавитами. Отдельные модели могут различаться типом кодировки кириллицы, что обычно приводит к полному несоответствию распечатки и исходного текста при использовании стандартных способов печати из операционной системы (например, с помощью программы PIP или переключения в режим дублирования выдачи с помощью команд CTRL P). Обойти это препятствие позволяют прикладные программы печати, осуществляющие перекодировку текста перед выдачей его на принтер.

Ниже будут приведены стандартные таблицы кодировок символов, а также способ управления процессом перекодировки при печати, которую можно реализовать непосредственно в операционной системе.

Встроенный тест и печать в шестнадцатиричном формате. Все типы принтеров имеют встроенную самопроверку и возможность печати в шестнадцатиричном формате.

Самопроверка инициируется, если при включении питания была нажата клавиша FORM FEED (или LINE FEED) на верхней панели управления принтера. Она выводит на печать все символы, определенные во внутреннем знакогенераторе, и позволяет проверить правильность функционирования принтера, не прибегая к помощи компьютера.

Некоторые принтеры (FX-800, FX-850 и др.) в режиме самопроверки перед распечаткой алфавита выводят полезную информацию о текущей установке DIP-переключателей (DIP — аббревиатура от Dual Inline

Таблица 1

Тип принтера	Скорость печати, зн/с	Знакогенератор кириллицы	Режим печати NLQ	Размер буфера печати, Кбайт
FX-850/1050	264	Нет	Есть	8
FX-800/1000	200	Есть	Есть	8
LX-800	180	Нет	Есть	3
FX-80	160	Нет	Нет	2
RAVI-8010M	180	Есть	Есть	4
Compute Mate-160	160	Есть	Есть	8

Скорость печати указана для черновикового режима при плотности 10 зн/дюйм.

Многие принтеры могут печатать как в черновиковом режиме (draft), так и в режиме качественной печати (Near Letter Quality, или сокращенно NLQ). Во втором случае

Package), расположенных на основной плате внутри принтера и отвечающих за инициализацию тех или иных режимов печати при включении питания.

Печать в шестнадцатиричном формате инициируется, если при включении питания были нажаты клавиши FORM FEED и LINE FEED, после чего принтер принимает передаваемые ему коды и печатает как сами эти коды, так и соответствующие им символы. Соответствие устанавливается на основе кодировки знакогенератора, записанного в ПЗУ принтера. Режим полезен при отладке собственных прикладных программ, написанных для обслуживания принтера.

Управление принтером из программ пользователя на «Корвете»

Из программ пользователя на печать выдаются обычно либо массивы данных, либо текстовая информация, либо графическая копия экрана.

Есть два способа обращения к принтеру: стандартный, через базовую систему ввода-вывода (BIOS); обращение непосредственно по физическому адресу адаптера принтера, минуя BIOS. Применение стандартного способа для распечатки русского текста на принтерах со знакогенератором кириллицы в ПЗУ дает нужный результат только в том случае, если оказываются согласованы три фактора: тип кодировки кириллицы в данной версии BIOS ОС компьютера; наличие или отсутствие в BIOS перекодировочной таблицы для выдачи данных на печать, а также ее содержание; тип кодировки кириллицы в принтере. Поэтому при написании универсальных программ, рассчитанных на работу с принтером произвольного типа, надежнее использовать второй способ — написание собственного драйвера принтера с обращением по физическому адресу устройства.

Однако в этом случае есть опасность потери работоспособности программы при ее переносе в другую версию ОС CP/M, поскольку физические адреса устройств не являются стандартизованным параметром и могут меняться от версии к версии.

Стандартные способы работы с принтером. На нижнем уровне ОС, где осуществляется непосредственная связь между программной и аппаратной частями компьютера, в стандартной версии ОС CP/M есть две функции для общения с принтером: одна — для выдачи символа на печать, другая — для проверки готовности принтера.

Функция № 5 BIOS: LIST

Входные параметры: регистр C процессора должен содержать код символа, выводимого на принтер.

Выходные параметры: нет.

Действие: символ из регистра C выводится на параллельный интерфейс принтера в соответствии с протоколом. Если принтер не готов, то функция находится в состоянии ожидания готовности.

Функция № 15 BIOS: LISTST

Входные параметры: нет.

Выходные параметры: регистр A (аккумулятор процессора) содержит 255, если принтер готов к приему данных, и 0, если не готов.

Действие: чтение состояния линии данных BUSY параллельного интерфейса.

Функция № 15 является «подмножеством» функции № 5, т. е. вызывается последней в процессе выполнения. Способы доступа к этим функциям (как и к описанным далее) определены в конкретных языках программирования.

Драйвер принтера в ОС CP/M-80 на ПК «Корвет» использует эти функции BIOS; протокол обмена соответствует протоколу интерфейса Centronix в его наиболее простой версии. Опознавание готовности внешнего устройства к приему данных осуществляется с помощью функции LISTST, отсутствие сигнала BUSY является признаком готовности принтера.

При вызове функции LIST сначала проверяется готовность принтера к приему данных. В случае его занятости функция ожидает готовности; при ее наступлении код передаваемого символа записывается в порт параллельного интерфейса, а затем программно формируется строб записи данных с отрицательной логической полярностью и длительностью около 20 мкс.

С помощью этого драйвера передаются любые коды от 0 до 255. Символы с кодами от 0 до 127, представляющие нижнюю часть таблицы символов и совпадающие с нижней половиной стандарта ASCII, передаются без изменений; верхняя половина таблицы, т. е. символы с кодами от 128 до 255, передается после перекодировки в соответствии со специальной перекодировочной таблицей (она будет описана ниже).

Базовая дисковая операционная система (BDOS), состоящая из процедур обмена с дисками и другими логическими устройствами, в отличие от BIOS представляет собой аппаратно независимую часть операционной системы CP/M-80. Для работы с внешним печатающим устройством BDOS имеет функцию № 5 LIST, выполняющую вывод символа на логическое устройство LST:.

Функция № 5 BDOS: LIST

Входные параметры: регистр C содержит 05H, регистр E — код выводимого символа.

Выходные параметры: нет.

Действие: вывод на LST: символа, код которого задан в регистре E.

Бейсик, Паскаль и Си имеют стандартные программные средства для вывода информации на печать с помощью описанных выше функций BDOS и BIOS.

В интерпретаторе Бейсик ПК «Корвет» есть две команды выдачи на печать: LPRINT — выдача на принтер произвольных выражений, LLIST — печать текста программы. Любой режим работы принтера может быть запрограммирован с помощью команды LPRINT. В приводимом примере принтер инициализируется, включается режим «элита», ширина левого поля устанавливается равной 12 символам, после чего печатается сообщение.

```
10 LPRINT CHR$(27);"@";:
                        REM Инициализация
20 LPRINT CHR$(27);"M";:
                      REM Установка "элиты"
30 LPRINT CHR$(27);"1";CHR$(12);:
                      REM левое поле=12 зн.
40 LPRINT "Elite-sized characters"
```

Оператор LPRINT следит за выдачей кода CR (возврат каретки). Если после последнего CR будут выданы 70 символов, LPRINT самостоятельно выдаст дополнительные коды CR (0DH) и LF (0AH, перевод строки). Поэтому использование LPRINT для работы в графических режимах весьма неудобно. В некоторых версиях Бейсика есть оператор WIDTH, позволяющий изменять длину строки при выдаче информации на внешнее устройство (консоль или принтер). Увеличение длины до 255 символов в некоторых случаях облегчает печать графики. К сожалению, в Бейсик-Корвет 1.0 длину строки можно менять лишь в пределах от 15 до 64 символов.

На языке компилятора Паскаль/MT+ в разделе описания переменных объявляется переменная-файл (например, printer) типа TEXT. С помощью операторов ASSIGN (printer, 'LST:') и RESET(printer) (или REWRITE(printer), что в данном случае эквивалентно) вывод в нее направляется на логическое устройство LST:, связанное через BIOS с адаптером принтера. Вывод реализуется с помощью операторов WRITE(переменная-файл, выражение) и WRITELN(переменная-файл, выражение). В приводимом примере включается режим печати сжатых подчеркнутых символов и выводится сообщение «Hello!».

```
Program TestPrinter;
var
  printer : text;
begin
  assign(printer, 'LST:');
  reset(printer);
  write(printer,chr(15));{ condensed }
  write(printer,chr(27),'-',chr(1));
                          { underlined }
  writeln(printer, 'Hello !');
  close(printer);
end.
```

В языке Express Pascal есть логические устройства LST:, LPT: и PRN:, которые связаны с драйвером параллельного порта принтера. Для вывода информации на принтер можно использовать любое из них.

В разделе описания переменных объявляется переменная-файл (например, printer) типа TEXT. С помощью операторов ASSIGN(printer, ⟨имя логического устройства⟩) и REWRITE(printer) вывод в нее направляется на указанное логическое устройство, связанное через BIOS с адаптером принтера. Вывод реализуется с помощью операторов WRITE(переменная-файл, ...) и WRITELN(переменная-файл, ...). Пример аналогичен предыдущему.

```
Program TestPrinter;
var
  printer       : text;

begin
  assign(printer, 'LST:');
  rewrite(printer);
  write(chr(15));          { включить
                         уплотненный режим }
  write(chr(27),'-',1);    { включить
                            подчеркивание }
  writeln(printer, 'Hello !');
end.
```

Работа с принтером в среде языка компилятора Aztec Си похожа на работу с дисковым файлом.

Одним из возможных способов является применение функций буферированного ввода-вывода. В этом случае пользователь должен объявить указатель на макроопределение FILE (например, FILE*printer), затем присвоить этому указателю значение, полученное в результате открывания файла с именем LST: на запись (printer= =fopen('LST:', 'w')). Имя LST: является зарезервированной синтаксической конструкцией, присвоенной логическому устройству вывода на принтер, поэтому оно распознается компилятором среди других имен файлов.

После открывания такого файла можно применять любые функции буферированного ввода-вывода, используемые обычно для записи в поток, связанный с дисковым файлом: fprintf (printer, format, argl, ...), fputs (printer), fwrite (buffer, size, count, printer), aputc(printer). Пример аналогичен предыдущему.

```
#include "libc.h"
#define CR  13
#define ESC 27

main()
  {FILE *printer;

   printer=fopen("LST:", "w");
   fprintf(printer, "%c%c%c%c\n",
                      15, ESC, '-', 1);
   fputs("Hello !", printer);
```

```
 aputc(CR, printer);
 fclose(printer);
}
```

Для 8-разрядных машин с операционной системой CP/M-80 существуют и другие компиляторы Си. Приведенный пример подходит для компиляторов, в которых определено логическое устройство LST:, и компьютеров, содержащих в BIOS подпрограммы работы с принтером.

Управление принтером на физическом уровне. Если применение стандартного драйвера принтера становится неудобным, пользователь может написать свой собственный драйвер, адресуясь непосредственно по физическим адресам аппаратного адаптера принтера.

Адреса регистров адаптера принтера ПК «Корвет» в ОС CP/M v. 2.2, BIOS v. 2.0 приведены в табл. 2.

Таблица 2

Регистр	Адрес	Назначение
Регистр данных	FB30H	Передача данных
Регистр состояния	FB38H	Проверка готовности
Регистр управления	FB33H	Формирование строба

Готовность принтера к приему данных проверяется чтением из регистра состояния и проверки бита 2: если он равен 1, принтер готов к приему данных. В этом случае передаваемый байт данных в инвертированном виде записывается в регистр данных, после чего программно формируется управляющий строб (последовательной записью в регистр управления чисел 0BH и 0AH).

В соответствии с техническими описаниями Epson-совместимых принтеров управляющий строб должен иметь длительность не менее 0,5 мкс, задержку относительно момента выставления данных на шину — не менее 0,5 мкс. Нетрудно видеть, что для 8-разрядных компьютеров эти условия выполняются автоматически при применении любых языков программирования, в том числе и ассемблера.

Некоторые принтеры, не принадлежащие к семейству Epson, требуют прямого (а не инвертированного) представления байта данных. Это относится, например, к термопечатающим устройствам отечественного производства семейства «Электроника».

Для адекватной печати русского текста на русифицированных принтерах в случае несовпадения кодировок кириллицы в компьютере и принтере пользователю следует создать и разместить в своей программе собственную перекодировочную таблицу, через которую необходимо пропускать передаваемый текст.

Подпрограмму передачи символа или массива символов непосредственно через параллельный порт лучше писать на ассемблере — это позволит достичь максимальной скорости передачи. Выигрыш в скорости особенно наглядно проявляется при выдаче больших массивов данных (до 2 Кбайт) при печати графических изображений в режиме учетверенной плотности.

Ниже приведен пример простой реализации драйвера печати на Бейсике (собственно драйвер записан в строках 200—250). Его алгоритм легко переписать на Паскале или Си.

```
10  S$="My own printer driver"
20  FOR I=1 TO LEN(S$)
30    D=ASC(MID$(S$,I,1)): GOSUB 200
40  NEXT I
50 'D=13: GOSUB 200
60  END
70  REM Подпрограмма выдачи символа
                  с кодом D на принтер
200   S=PEEK(&HFB38):
        REM Чтение регистра состояния
210   IF (S AND 4)<>4 THEN 200:
       REM Проверка готовности принтера
220   POKE &HFB30, 255-D:
          REM Запись инвертированного
                байта в регистр данных
230   POKE &HFB33, &HB:
                REM Формирование строба
240   POKE &HFB33, &HA:
                REM Формирование строба
250   RETURN
```

Продолжение следует.

В следующих номерах будут опубликованы таблицы кодировок (основная, альтернативная, КОИ-8, принтера RAVI-8010M) и некоторые команды управления принтером.

С. ГРИГОРЬЕВ

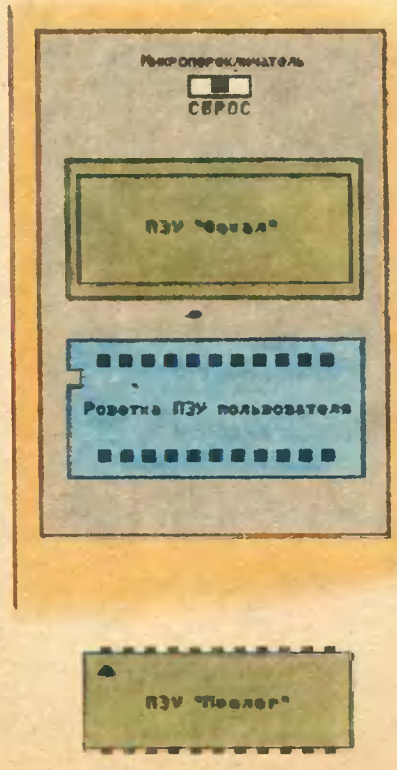
Пролог-Д в ПЗУ БК-0010

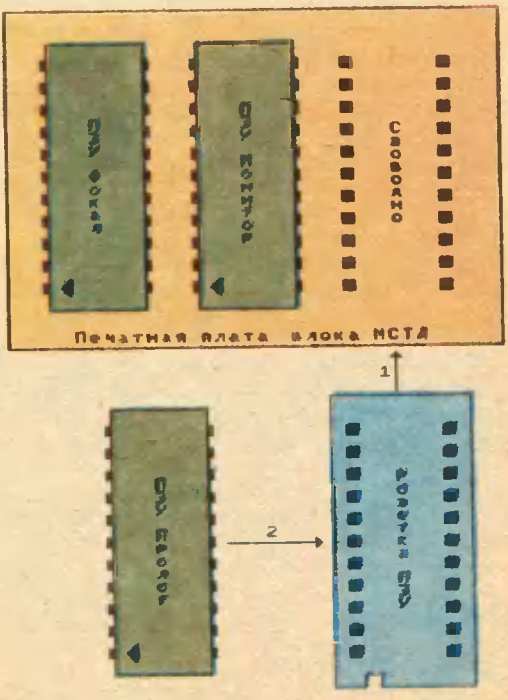
Язык логического программирования Пролог, как показала практика наших публикаций [1, 2], вызывает интерес. Программное обеспечение ПЭВМ «Электроника БК-0010» (правильнее сказать, программное обеспечение КУВТ-86(87) содержит систему Пролог, однако она не была доступна широкому кругу пользователей, особенно обладателям домашних БК-0010.

Совместно с ТЦ ППП «Модус» нами разработана система Пролог-Д, хранящаяся в ПЗУ, непосредственно подключаемом к компьютеру. Входной язык системы полностью совпадает с входным языком, описанным в [2], а возможности (объем исходного текста и глубина рекурсии) значительно шире за счет того, что система хранится в ПЗУ, а не в ОЗУ. Нет, как вы сами понимаете, и проблем с надежностью считывания программы.

К БК-0010 (Фокал) ПЗУ «Пролог» подключается очень просто: устанавливается в розетку ПЗУ пользователя, расположенную в специальном отсеке под отодвигающейся крышкой (рис. 1). При этом важно не перепутать порядок ножек микросхемы. Ножка с номером 1 отмечена на микросхеме стрелкой, а клемма розетки с номером 1 отмечается вырезом на розетке.

Несколько сложнее установить ПЗУ «Пролог» в БК-0010.01 (Бейсик). На месте розетки пользователя в этой машине находятся впаянные микросхемы ПЗУ «Бейсик»; крышка, закрывающая ПЗУ, заклеена. В данном случае ПЗУ «Пролог» устанавливается в блок МСТД в соответствии с рис. 2. На печатной плате блока есть свободное место, на которое припаивается розетка ПЗУ типа РС-24, а в нее вставляется микросхема ПЗУ (последняя не допускает пайки). Розетки РС-24 продаются в магазинах «Радиолюбитель», «Юный техник» и т. п. Впаивать розетку необходимо с учетом номеров клемм, так же как впаяны другие микросхемы на печатной плате МСТД. На рис. 2 цифрами 1 и 2 показана последовательность работ. Для работы блок МСТД подключается к разъему МПИ.





Для запуска системы Пролог необходимо перейти в монитор и после появления приглашения «?» запустить программу нажатием клавиши П.

Для приобретения микросхемы ПЗУ «Пролог» достаточно обратиться в ТЦ ППП «Модус». Стоит она вместе с инструкцией пользователя 200 рублей.

Литература

1. *Григорьев С., Морозов М.* Давайте попробуем Пролог // Информатика и образование. 1987. № 4.

2. *Григорьев С.* Программирование на Прологе-Д // Информатика и образование. 1990. № 4—6; 1991. № 1.

Адрес для справок: ТЦ ППП «Модус», 142500, Московская обл., г. Павловский Посад, ул. Кузьмина, д. 33.

Тел. 8-243-2-22-43 для Москвы, 8-096-43-2-22-43 — для остальных городов.

С. МАЙОРОВ, О. ДМИТРИЕВА
СШ № 400, Москва

Тема «Операционные системы» на базе КУВТ-86

Раздел программы по компьютерной грамотности включает знакомство со следующим программным обеспечением: редактор текстов, графический редактор, электронные таблицы, база данных, экспертные системы. На настоящий момент возможности локальных сетей классов учебной вычислительной техники позволяют организовать практическое знакомство с этими программами на рабочих местах учащихся. Такой подход обеспечивает подготовку учебника как пользователя конкретного программного средства и не дает целостной картины о работе пользователя с ЭВМ как с многоуровневой системой. Тема «Операционные системы», по нашему мнению, имеет полное право на существование в курсе информатики и основ вычислительной техники средней школы. Остановимся на этом подробнее. Пользовательские навыки работы с ПЭВМ будут полностью сформированы только в том случае, если помимо уровня прикладного программного обеспечения будет рассмотрен уровень операционной системы, как комплекса программ, предназначенного для планирования и организации процесса обработки, ввода-вывода и управления данными, распределения ресурсов, подготовки и отладки программ.

Несомненно, наличие в большинстве ЭВМ, используемых в школах, ПЗУ с тем или иным транслятором облегчает «жизнь» преподавателя, но при этом у учащихся формируется ложное представление о понятии системы программирования. Они вкладывают в него лишь правила символической записи алгоритмических конструкций и не связывают его с необходимостью «физически» иметь транслятор того языка, на котором они хотят программировать. Таким образом, программный принцип работы ЭВМ без изучения правил и принципов работы операционных систем не раскрывается полностью. Тема «Программное обеспечение», которая включает в себя понятие базового программного обеспечения ЭВМ, типы языков программирования, понятие о технологии разработок программных средств является чисто теоретической и не имеет практической поддержки в работе за машиной.

Обучая учащихся работе с прикладным программированием обеспечением, учитель должен буквально «на пальцах» ввести понятие «файл», объяснить, зачем нужно имя файла и почему оно должно быть индивидуальным, не говоря уже о расширении имени. Ясно, что говорить об этих понятиях в отрыве от объяснения файловой системы нельзя.

Изучение раздела «Операционные системы» целесообразно проводить последовательно по темам. На наш взгляд, это позволит выработать правильное представление о правилах работы и принципах построения любой операционной системы.

Тема 1. Понятие операционной системы и место операционной системы в среде программного обеспечения ЭВМ. Функции операционной системы.

Тема 2. Краткая историческая справка. Понятие файла. Файловая система: минимальный адресуемый элемент — блок; понятие файла, устройства файловой и нефайловой структуры, имя устройства, логические и физические имена устройств, спецификация файла, формат записи спецификации, групповая спецификация, каталоги.

Тема 3. Базовый состав операционной системы: модульный принцип организации операционной системы, драйверы устройств, монитор, первичный загрузчик, утилиты.

Тема 4. Командный язык операционной системы: понятие команды, понятие ключа,

входные и выходные файлы, формат командной строки.

Тема 5. Команды операционной системы.

Подобная последовательность окажется полезной только в том случае, если на рабочих местах учащихся есть возможность организовать работу с операционной системой.

Проблему организации работы на рабочем месте ученика (РМУ) БК-0010 по правилам операционной системы решает педагогическое программное средство ЭМУЛЯТОР RT11, которое является сетевым монитором для КУВТ-86 на базе БК-0010. Монитор создан для обеспечения работы на ученических местах по правилам операционной системы RT11 с ограниченным набором команд и предназначен для поддержки преподавания темы «Операционные системы». Он может быть использован для организации работы со стандартной сетью ДВК — БК, позволяет осуществить загрузку и запуск программ на ученических местах в формате. BIN и .ACU и производить работу в среде Бейсик—БК.

ЭМУЛЯТОР позволяет регулировать доступ к диску и выполнение отдельных команд для каждого ученического места по желанию учителя, который на приоритетном уровне может вмешиваться в процесс обмена.

Можно для всех РМУ установить различные режимы: выполнение команды без подтверждения со стороны преподавателя; выполнение команды с подтверждением со стороны преподавателя; выполнение команды для данного РМУ запрещено.

Файловая система монитора аналогична файловой системе RT11. В формате файловой системы RT11 выдержаны индивидуальные и групповые спецификации файлов, имена физических и стандартных логических устройств. Команды монитора могут быть выполнены для устройства по умолчанию (DK:), которое задает учитель на рабочем месте преподавателя перед началом работы в среде ЭМУЛЯТОРА на РМУ. Рабочим может быть любое устройство файловой структуры, в том числе, несомненно, и логический диск. Использование команды DIR с ключом /DEL позволяет на РМУ выводить список удаленных файлов, что может являться демонстрацией состояния файла «пустой».

Возможность просмотра каталога системного диска позволяет говорить о базовом составе и модульном принципе организации операционной системы.

Командный язык ЭМУЛЯТОРА совпадает с командным языком RT11 и позволяет задавать командную строку в формате: ⟨КОМАНДА⟩/⟨КЛЮЧ⟩ ⟨СПЕЦИФИКАЦИИ ВХОДНЫХ ФАЙЛОВ⟩ ⟨СПЕЦИФИКАЦИЯ ВЫХОДНОГО ФАЙЛА⟩, что дает возможность на РМУ получать различную информацию о системе и управлять системой.

Защита от удаления, переименования и копирования файлов на системном устройстве позволяет сохранить работоспособность реальной RT11 на рабочем месте преподавателя.

ЭМУЛЯТОР, конечно, не обладает всеми возможностями RT11, но это программно-педагогическое средство обеспечивает полную программную поддержку преподавания темы «Операционные системы». Это подтверждает опыт работы с ЭМУЛЯТОРОМ на уроках ОИВТ в VIII—XI классах СШ № 400 Москвы.

Для тех, кто заинтересовался нашей разработкой, сообщаем телефон для справок в Москве: 272-26-71.

Авторы статьи «Преподавание предмета» Основы технического творчества», опубликованной в ИНФО, № 2, 1992, работают в Московском техникуме космического приборостроения. Контактные телефоны: **261-66-87, 261-62-22, 261-52-20.**

Soft-парад

Первым откликом на объявление о soft-параде стало письмо А. Вернигоры из томской «Ассоциации пользователей УКНЦ». Он предложил свою разбивку категорий парада и первые оценки. В частности, лучшим разработчиком программ для УКНЦ он считает НЦ ПСО (Москва). К сожалению, основное внимание в предлагаемом перечне уделено хорошим программам. Рекламировать их нужно, но не менее важно и предостерегать против плохих!

Программа	Оценка	Комментарий
Математика. Исследование графиков функций	2	Обучение и контроль
Физика-7	0	Тесты; учитель-физик оценил низко
Русский язык	2	Головоломки, очень популярны у младших школьников
КОКОС	0	Работает только на РМП
KEY BASIC, KEY OS, KEY K13	0	Обучающие по Бейсику, ОС, редактору текста. Привязаны к РМП, в Бейсик-курсе неудачно подобраны задания
История-6, 7	0	Кроссворды
ZERO (кооп. "Интерфейс")	1	Пакет для учителя информатики
Механика (Физика-9)	1	Неудачное оформление
Алгоритмы и исполнители	1	Практикум по алгоритмическому языку

КУВТы		
КУВТ УКНЦ (13 рабочих мест)		2-3 РМУ обычно не работают. Едва ли не самой уязвимой является самая простая деталь - выключатель питания.

Программа	Оценка	Комментарий
ОС, сетевое программное обеспечение, утилиты		
ADOSFB	2	RT-11 с русскоязычной диагностикой
NETSVD.SAV	2	Сетевой монитор
SC.SAV	2	Файловый монитор
MZFORM.SAV	2	Программа форматирования дискет
Трансляторы		
PAS.SAV	2	Турбо-Паскаль
LOGO.SAV	2	Интерпретатор МИТ-Лого
Прикладные программы (базы данных, различные редакторы и т.п.)		
EDK.SAV	2	Текстовый редактор
WRITER.SAV	2	Система подготовки текстов
MELOM.SAV	2	Музыкальный редактор
MUSIC.SAV	2	"Проигрыватель" мелодий
AHEAD.SAV	1	Графический редактор
PAINT.SAV	2	Графический редактор
GREDIT.SAV	1	Графический редактор
Инструментальные средства создания ПЛС		
SAE	2	Автоматизированная обучающая система
UNTEST.SAV	-2	Система подготовки опросных программ
Обучающие и контролирующие программы		
Пакет Бейсик-практикума	2	

И. ПОЛОВИНА
Пермский пединститут

Введение функции пользователя в работающую программу

В настоящее время широкое распространение получили обучающие программы, в частности по математическим дисциплинам. Особенностью этих программ является частая необходимость ввода вида функции пользователем (например, при построении графиков, решении уравнений и т. п.). Если ПЭВМ имеет дисковод или электронный диск, то это легко сделать.

Суть предложения в следующем. Вся программа разбивается на две части. Первая (назовем ее управляющей) находится в ОЗУ и предлагает пользователю ввести вид исследуемой функции, например:

ВВЕДИТЕ ФУНКЦИЮ F(X)-

Вторая часть (основная программа) находится на дискете либо на электронном диске, в ее работе и будет использовано описание функции в виде функции пользователя (первой строкой этой программы).

В управляющей программе открываются два файла последовательного доступа: один — для записи данных, например SBORKA.BAS, другой — для считывания; в этой роли выступает основная программа — OSNOBN.BAS. Первой строкой в файл SBORKA записывается строка с функцией пользователя, далее — строки из основной программы OSNOBN, т. е. происходит как бы процесс «склеивания» (для этого необходимо преду-

смотреть в основной программе место для строки функции пользователя). А чтобы сразу запустить собранную таким образом программу на счет, в конце управляющей программы должен стоять соответствующий оператор запуска, зависящий от типа операционной системы ПЭВМ:

```
CHAIN "<устройство>SBORKA.BAS"
```

для ПЭВМ с операционной системой типа ОС ДВК,

```
RUN "<устройство>SBORKA.BAS"
```

для ОС MSX.

Ниже приводится листинг программы, реализующий предложенный алгоритм для преподавательской машины «Ямаха».

```
10 REM управляющая программа
20 CLS
30 MAXFILES=2
40 K$="10 DEF FNY(X)="
50 LOCATE 5,14
60 INPUT "ВВЕДИТЕ ФУНКЦИЮ F(X)=";F$
70 IF LEN(F$)=0 THEN STOP
80 T$=K$+F$
90 OPEN "A:SBORKA.BAS" FOR
                          OUTPUT AS #1
100 OPEN "A:OSNOBN.BAS" FOR
                           INPUT AS #2
110 PRINT #1,T$
120 IF EOF(2) THEN 160
130 LINE INPUT #2,T$
140 PRINT #1,T$
150 GOTO 120
160 CLOSE
170 RUN "A:SBORKA.BAS"

20 REM основная программа OSNOBN.BAS
30 REM строка 10 оставлена для функции
                              пользователя
40 X=FNY(5): REM пример использования
                                 функции
50 PRINT X
```


ДОСКА ОБЪЯВЛЕНИЙ

Прошу откликнуться тех, кто работает с малышами. Хотела бы обменяться опытом, методиками, программами. Ищу информацию о курсах для преподавателей информатики в младших классах.

690105, Владивосток, ул. Русская, 98, кв. 84. М. Л. Золотайко.

Предлагаю для обмена свыше 100 программ для IBM PC.

450024, Уфа, ул. Правды, 21, кв. 181. Ю. А. Малякин.

Хочу переписываться, обмениваться программами и информацией с владельцами ПК «Криста», модель 2.

656053, Алтайский край, г. Барнаул, ул. 2-я Северо-Западная, 37. В. Бушаев.

Куплю новые программы для ПЭВМ «Апогей-БК01».

127599, Москва, ул. Маршала Федоренко, 4, корп. 2, кв. 370. П. Ю. Солдатов.

Ищу коллег — владельцев ПЭВМ «Апогей-БК01».

117465, Москва, ул. Теплый Стан, 9/2, кв. 63. А. А. Попов.

Отзовитесь, пользователи ПК «Партнер»! Давайте объединимся в заочный клуб! Буду рад ответить всем.

342430, Украина, Донецкая обл., г. Комсомольское, ул. Зои Космодемьянской, 54. М. А. Жирохов.

Обменяюсь информацией и программами с владельцами БК-0010.01. Имею 480 программ.

397222, Воронежская обл., Грибановский р-н, с. Ключики. В. Н. Севрюков.

Предлагаю переписку и сотрудничество владельцам БК-0010 (-01), интересующимся и занимающимся доработками БК (расширение ОЗУ, подключение дисковода и т. д.), написанием соответствующих программ (в том числе Диск-Монитора и ДОС).

220033, Минск-33, а/я 112. А. Н. Загорский.

Учителям информатики, работающим с КУВТ УКНЦ 01.01 (БК-0011М), предлагаю переписываться и обмениваться программами.

346330, Ростовская обл., г. Донецк, ул. Кольцевая, 11, кв. 2. С. А. Бредин.

Борис Бортник, создатель DIDAY и др.! Помогите, пожалуйста, разобраться с организацией рулонного сдвига!

410030, Саратов, ул. Антонова-Саратовского, 44/62, кв. 229. А. В. Олейник.

Разработана программа, позволяющая исполнять программы для ПМК серии Б3-34 и выше (МК-52, МК-61) на ПЭВМ типа ДВК и др.; программа, позволяющая об-

легчить программирование на языке ПМК; создан банк программ для ДВК на основе алгоритмов, реализованных на ПМК. Стоимость поставки равна сумме стоимости дискеты (если запись должна быть произведена на дискету формата МХ: 40 дорожек, прошу присылать дискету в связи с невозможностью самостоятельного форматирования) и почтовых расходов.
614084, Пермь, ул. Куйбышева, 163, кв. 24. М. Н. Бахарев.

Обмениваюсь программами для БК-0010 и «Спектрума». Имею около 300 различных программ.
117415, Москва, просп. Вернадского, 69, кв. 60. К. Толмазов.

Ищу программы для «Вектора-06Ц», желаю переписываться с владельцами этого компьютера.
686611, Магаданская обл. Чаунский р-н, пос. Апапельгино, ул. Северная, 8, кв. 7. Д. Петренко.

Хотел бы переписываться и обмениваться программами с пользователями БК-0010. Имею много игровых и прикладных программ.
327034, Николаев, ул. Южная, 74а, кв. 3. В. В. Котов.

Хочу переписываться с радиолюбителями-коротковолновиками, заинтересованными в компьютеризации любительской радиосвязи. Могу предложить программы в кодах «CW QSO» и «RTTY QSO» собственной разработки.
249400, Калужская обл., г. Людиново, ул. Московская, 9, кв. 50. Г. А. Тяпичев.

Предлагаю обмен программами для БК-0010.01. В моем фонде около 1500 файлов, в основном игровые программы. В письмо прошу вложить каталог.
150023, Ярославль, ул. Рыкачева, 4/7, кв. 18. А. В. Новиков.

Хотел бы обмениваться программами и переписываться с владельцами «Вектора-06Ц». Обладаю определенным количеством интересных программ и информацией для ПК. Каталог — в кротчайшие сроки.
330001, Запорожье, ул. Победы, 42, кв. 34. В. Г. Печерский.

Обмен программами для БК.
141120, Московская обл., г. Фрязино, пр. Десантников, 11, кв. 321. В. А. Зигунский.

Хотел бы переписываться и обмениваться программами с владельцами БК-0010.01.
430031, Саранск, ул. Гожовская, 10, кв. 57. М. А. Панков.

Хочу переписываться и обмениваться программами с владельцами ПК «Вектор-06Ц». Отвечу всем, кто мне напишет.
155000, Ивановская обл., Гаврилов Посад, ул. К. Маркса, 12, кв. 5. А. А. Курашин.

Имею более 200 программ для БК-0010.01. Предлагаю переписку и обмен программами.
362035, СОССР, Владикавказ, ул. Леваневского, 279/2, кв. 16. Э. Т. Дзуцев.

Прошу помочь с программами для «Микроши»; обменяю выпуски программ 12 и 13 на 1—11; готов обмениваться распечатками.
Башкирская ССР, Туймазинский р-н, п. Субханкулово, 8-й м-н, д. 4, кв. 106. О. Латыпов.

Хочу переписываться и обмениваться программами с владельцами «Микроши» и «Кристы».
452604, БашССР, Туймазинский р-н, пос. Субханкулово, мкрн. 8, д. 3, кв. 34. Э. Давлетшин.

Владельцев ПЭВМ «Вектор-06Ц» приглашаю к сотрудничеству. Имею более 500 программ.
270015, Одесса, ул. Академика Королева, 51/1, кв. 50. Б. Васильев.

Хочу обмениваться программами и описаниями к программам для «Спектрума».
674520, Читинская обл., Оловяннинский р-н, пос. Ясногорск, ул. Энергетиков, 9, кв. 79. И. В. Бородихин.

Ищу владельцев ПК «Вектор-06Ц» для обмена информацией и программами. Возможны какие-либо совместные разработки.
339000, Донецкая обл., г. Макеевка, мкрн. Солнечный, 8, кв. 128. Б. Б. Линник.

Хотел бы переписываться с владельцами и пользователями ПК «Поиск». Предлагаю программы на кассетах и дискетах. Заинтересован в обмене программами и информацией.
197198, С.-Петербург, ул. Съезжинская, 13, кв. 36. А. Н. Бессалов.

Готовы оказать помощь по адаптации «старых» Бейсик-программ обладателям «Корвета» с Бейсиком версии 2.0. Постараемся ответить на вопросы. Имеем собственные игровые и прикладные программы.
265100, Ровенская обл., г. Дубно, ул. Чапаева, 34, СШ № 8. А. Н. Дмитриев.

С. АХМАНОВ, А. НЕЧАЕВ, Н. РОЙ, А. СКУРИХИН

Архитектура «Корвета»

Потренируемся в управлении инверсией. Пример 7 покажет, как воздействуют биты 4 и 5 видеорегистра на вывод знаков на экран (клавиши F1, F2 и F3 управляют этими битами).


```
{пример 7}
program TestInv;
const
      VideoRamBase   = $FC00;
{начало видеопамяти в конфигурации 1CH}
      VideoRegister  = $FB3A;
      MaxAddr        =  1023;
      F1             =    80;
      F2             =    81;
      F3             =    82;
      ESC            =    58;
var     Adr: integer;
 { для хранения относительного адреса}
        a,s,m: byte;
          { для хранения кода символа}
begin                 { тело программы }
  ClrScr;
  Adr := 0;
  repeat
    ReadKeyExt(a,s,m);
    case s of
      F1:  mem[VideoRegister] :=
     { выключаем инверсию, не трогая }
            (mem[VideoRegister] AND
                          $CF) OR $20;
       {остальные биты видеорегистра }
      F2:  mem[VideoRegister] :=
                (mem[VideoRegister] AND
                          $CF) OR $10;
      F3:  mem[VideoRegister] :=
           mem[VideoRegister]  OR $30;
    ESC:   exit;
             { если ESC, заканчиваем }
    else
      if a>31 then
         begin
           mem[VideoRamBase+Adr] :=a;
     {запись этих кодов в видеопамять}
           if Adr < MaxAddr then
                          Adr := Adr+1
           else Adr := 0;
         end;
    end;
  until false;
end.
```

Продолжение. Начало см.: Информатика и образование. 1992. № 1, 2.

Отсюда не очень ясно, что делает клавиша F3,— кажется, что она ничего не меняет. Чтобы наглядно продемонстрировать ее действие, изменим программу так, чтобы клавиши F4 и F5 сдвигали содержимое экрана вверх и вниз соответственно. Набрав программу, разберитесь, как зависит процесс сдвига экрана от того, какая из клавиш F1 — F3 была перед этим нажата.

```
{пример 8}
program TestInv;
const
      VideoRamBase   = $FC00;
      VideoRegister  = $FB3A;
      MaxAddr        =  1023;
      OurRow     = VideoRamBase + 64*8;
      F1             =    80;
      F2             =    81;
      F3             =    82;
      F4             =    83;
      F5             =    84;
      ESC            =    58;
      SPACE          =    32;
var     Adr: integer;
        a,s,m: byte;
begin                 { тело программы }
  mem[VideoRegister] :=
 {на всякий случай выключаем инверсию}
                (mem[VideoRegister] AND
                          $CF) OR $20;
  for adr := VideoRamBase to
                VideoRamBase+MaxAddr do
                     mem[Adr] := SPACE;
                      { очищаем экран }
  for adr := OurRow to OurRow+20 do
                   mem[Adr] := ord('*');
  mem[VideoRegister] :=
                      { включаем инверсию }
     (mem[VideoRegister] AND $CF)
                                OR $10;
  for adr :=OurRow +20 to OurRow+40 do
     mem[Adr] := ord('*');
  mem[VideoRegister] :=
                  { выключаем инверсию }
     (mem[VideoRegister] AND $CF)
                                OR $20;
  for adr := OurRow+40 to OurRow+60 do
     mem[Adr] := ord('*');
  repeat
    ReadKeyExt(a,s,m);
    case s of
```

```
        F1:  mem[VideoRegister] :=
        { переключаем режим, не трогая }
             (mem[VideoRegister] AND
                          $CF) OR $20;
         {остальные биты видеорегистра }
        F2:  mem[VideoRegister] :=
                (mem[VideoRegister] AND
                          $CF) OR $10;
        F3:  mem[VideoRegister] :=
             mem[VideoRegister]  OR $30;
        F4:  begin
               for adr := 0 to
                          MaxAddr-64 do
                 mem[VideoRamBase+Adr]:=
               mem[VideoRamBase+Adr+64];
               for adr := MaxAddr-64 to
                             MaxAddr do
                 mem[VideoRamBase+Adr]:=
                             SPACE;
             { очищаем последнюю строку}
             end;
        F5:  begin
               for adr := MaxAddr-64
                          downto 0 do
              mem[VideoRamBase+Adr+64]:=
                  mem[VideoRamBase+Adr];
               for adr := 0 to 63 do
                 mem[VideoRamBase+Adr]:=
                             SPACE;
             { очищаем последнюю строку}
             end;
       ESC:  exit;
      end;
    until false;
  end.
```

А теперь подумаем, как можно управлять курсором. У «Корвета» курсором обычно служит инверсное знакоместо. Если текст на экран выведен в обычном, прямом, виде, проблем нет: для организации курсора достаточно приемов, использованных в примере 7. Если же часть текста выведена в инверсном виде, возникает необходимость модифицировать курсор в зависимости от его местоположения (делать его инверсным или прямым). Таким образом, прежде чем высветить курсор, нужно определить состояние бита инверсии данного знакоместа. Для этого воспользуемся регистром 38H. В программе 9 курсор «слушается» стрелок влево и вправо; ESC, как обычно, выход из программы.

```
{пример 9}
program TestInv;
const
      VideoRamBase   = $FC00;
      VideoRegister  = $FB3A;
      MaxAddr        =  1023;
      OurRow     = VideoRamBase + 64*8;
      Reg38H         = $FB38;
      InvBit         =     8;
      LEFT           =    68;
      RIGTH          =    70;
      ESC            =    58;
      SPACE          =    32;
var    Adr: integer;
       a,s,m: byte;
       PosCursor,OldPosCursor:integer;
procedure InvToggle(Pos:integer);
         { включает/выключает курсор }
var c:byte;
    inv:boolean;
begin
     mem[Vide Register] :=
                  { включаем "память" }
        mem[VideoRegister] OR $30;
     c := mem[OurRow + Pos];
 {прочитали символ из текущей позиции}
     Inv := (mem[Reg38H] AND
                       InvBit) = InvBit;
                  { состояние инверсии }
     if  Inv then
         mem[VideoRegister] :=
                { выключаем инверсию }
           (mem[VideoRegister] AND
                          $CF) OR $20
    else
        mem[VideoRegister] :=
                { включаем инверсию }
          (mem[VideoRegister] AND
                          $CF) OR $10;
    mem[OurRow + Pos] := c;
  { вернули символ в инверсном виде }
end;
begin                  { тело программы }
 mem[VideoRegister] :=
{на всякий случай выключаем инверсию}
    (mem[VideoRegister] AND
                          $CF) OR $20;
 for adr := VideoRamBase to
              VideoRamBase+MaxAddr do
    mem[Adr] := SPACE;
                       { очищаем экран }
 for adr := OurRow to OurRow+20 do
    mem[Adr] := ord('*');
 mem[VideoRegister] :=
                { включаем инверсию }
    (mem[VideoRegister] AND
                          $CF) OR $10;
 for adr := OurRow+20 to OurRow+40 do
    mem[Adr] := ord('*');
 mem[VideoRegister] :=
              { выключаем инверсию }
    (mem[VideoRegister] AND
                          $CF) OR $20;
 for adr := OurRow+40 to OurRow+60 do
    mem[Adr] := ord('*');
 PosCursor := 0;
 OldPosCursor := 0;
 InvToggle(PosCursor);
 repeat
   ReadKeyExt(a,s,m);
   case s of
    LEFT:  if PosCursor > 0 then
                        Dec(PosCursor);
    RIGTH: if PosCursor < 63 then
                        Inc(PosCursor);
   end;
   if PosCursor <> OldPosCursor then
     begin
       InvToggle(OldPosCursor);
  { погасили курсор в прежнем месте }
       InvToggle(PosCursor);
       { зажгли курсор в новом месте }
       OldPosCursor := PosCursor;
       { запомнили, где это сделали }
     end;
 until s = ESC;
            { если ESC, заканчиваем }
 mem[VideoRegister] :=
    { при выходе выключаем инверсию }
 (mem[VideoRegister] AND $CF) OR $20;
end.
```

Следующие два примера продемонстрируют использование возможности читать состояние сигнала VBlank (бит 1 регистра 38H). Сначала попробуем синхронизоваться с разверткой монитора. Алгоритм работы программы прост: дожидаемся обратного хода вертикальной развертки и выводим символ Ж в верхнюю строку экрана. Снова дожидаемся, теперь уже прямого хода развертки, затем делаем небольшую задержку (ее можно изменять) и, наконец, выводим в то же место экрана новый символ — X.

Клавиши **ВВЕРХ** и **ВНИЗ** плавно меняют величину задержки. Наберите эту программу и, меняя задержку, получайте ее воздействие на экран. Не правда ли, интересный эффект?


```
{пример 10}
program TestBlank;
const
      VideoRamBase  = $FC00;
      VideoRegister = $FB3A;
      OurPos        = VideoRamBase + 32;
      Reg38H        = $FB38;
      BlankBit      =     2;
      UP            =    72;
      DOWN          =    66;
      ESC           =    58;
      SPACE         =    32;
var     ValDelay : integer;
                    {величина задержки}
        a,s,m: byte;
          { для хранения кода символа}

Function TestBlank:boolean;
      { проверяет вертикальный бланк }
begin
     TestBlank := (mem[Reg38H] AND
                        BlankBit) = 0;
end;

procedure Del(count:integer);
var i:integer;
begin
  for i := 0 to count do;
end;
begin                  { тело программы }
  ClrScr;
  ValDelay := 8;
  repeat
    repeat
    until TestBlank;
    {ждем начала вертикального бланка}
    mem[OurPos] := ord('Ж');
    repeat
    until Not TestBlank;
     {ждем конец вертикального бланка}
    Del(ValDelay);
    mem[OurPos] := ord('X');
    if keypressed then
                       ReadKeyExt(a,s,m)
    else s := 0;
    case s of
     UP:   Inc(ValDelay);
     DOWN:Dec(ValDelay);
     ESC: exit;
    end;
  until false;
end.
```

Теперь посмотрим, какой модификации ваш компьютер. Ранее говорилось, что у разных моделей может быть разная длительность горизонтальной развертки и, как следствие, разная длительность кадра. Отличаются они незначительно, примерно на 0,1 %, но различие все же можно зафиксировать. Для этого кроме компьютера нам понадобятся часы.

Наберите следующую программу и запустите ее, точно зафиксировав время начала работы. Ровно через час (3600 с, с точностью до секунды) нажмите **ПРОБЕЛ**. Выведенное число даст частоту кадров, если его поделить на 3600. Частота, близкая к 50,08 Гц,— признак новой модели, 50,02 Гц — старой.

Обратите внимание на то, что все сказанное об обратном ходе справедливо и для графического экрана.

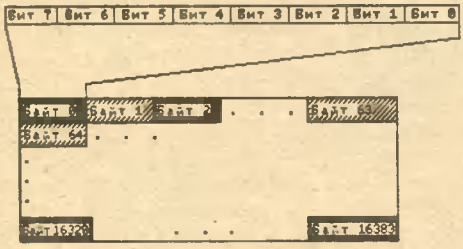
```
{пример 11}
program TestScan;
const
      Reg38H        = $FB38;
      BlankBit      =     2;
var   Count : LongInt;{счетчик кадров}

Function TestBlank:boolean;
begin
     TestBlank := (mem[Reg38H] AND
                        BlankBit) = 0;
end;

begin                  { тело программы }
  ClrScr;
  WriteLn('Засеките время
                    и нажмите ПРОБЕЛ');
  if ReadKey = '' then ;
  WriteLn('Пошел счет...
        Через 3600 с нажмите ПРОБЕЛ');
  Count := 0;
  repeat
    repeat
    until TestBlank;
    {ждем начала вертикального бланка}
    repeat
    until Not TestBlank;
     {ждем конец вертикального бланка}
    Inc(Count);
  until KeyPressed;
  WriteLn('Показания
                    счетчика = ',Count);
end.
```

Графический дисплей

Основа графического дисплея (его упрощенная функциональная схема приведена на рис. 1) — графическое запоминающее устройство. Контроллер ЭЛТ, как и в случае А/Ц дисплея, формирует все необходимые управляющие сигналы, в том числе и адрес отображаемой ячейки памяти. Байт, считанный им из видеопамяти, поступает в сдвиговый регистр и преобразуется в восемь последовательных (по горизонтали) точек экрана (старший бит отображается слева!).

Напомним, что графический экран «Корвета» состоит из 512×256 точек; значит, для его заполнения необходимо (512×256)/8=16384 ячеек памяти. Именно такой объем адресного пространства занимает графическая память.

Правила отображения графической памяти на экран аналогичны правилам А/Ц дисплея, т. е. в левом верхнем углу отображается нулевой байт; номера отображаемых ячеек возрастают (через каждые 8 точек экрана) слева направо и сверху вниз. В каждой строке отображается 512/8=64 байта (рис. 2).



Но этого достаточно только для одноцветного изображения, а у «Корвета», как мы знаем, каждая точка графического экрана может отображаться любым из восьми цветов. Следовательно, нужен не один, а три бита на точку!

Все верно. Поэтому существует не один, а три одинаковых банка графической памяти по 16 Кбайт, причем в режиме отображения все они работают синхронно. Каждый банк образует свою графическую плоскость, в формировании любой точки экрана каждая плоскость участвует одним своим битом, и, если мы хотим зажечь точку, например, желтого цвета, нам необходимо соответствующий этой точке бит плоскости 0 установить в 0, а биты плоскостей 1 и 2 в 1. Для этого можно:

считать байт из первой плоскости;

изменить бит, соответствующий заданной точке;

записать байт обратно в первую плоскость;

считать байт из второй плоскости;

изменить бит, соответствующий заданной точке;

записать байт обратно во вторую плоскость;

считать байт из третьей плоскости;

изменить бит, соответствующий заданной точке;

записать байт обратно в третью плоскость.

Именно так происходило бы зажигание точки, если бы архитектура дисплея соответствовала рис. 1; времени на эти манипуляции потребовалось бы очень много. Для ускорения процесса вывода графической информации «Корвет» имеет некоторые аппаратные ухищрения (они повышают эффективность работы с графикой раз в 10): во-первых, запись байта осуществляется сразу во все три плоскости; во-вторых, записываются только те биты, которые следует изменить.

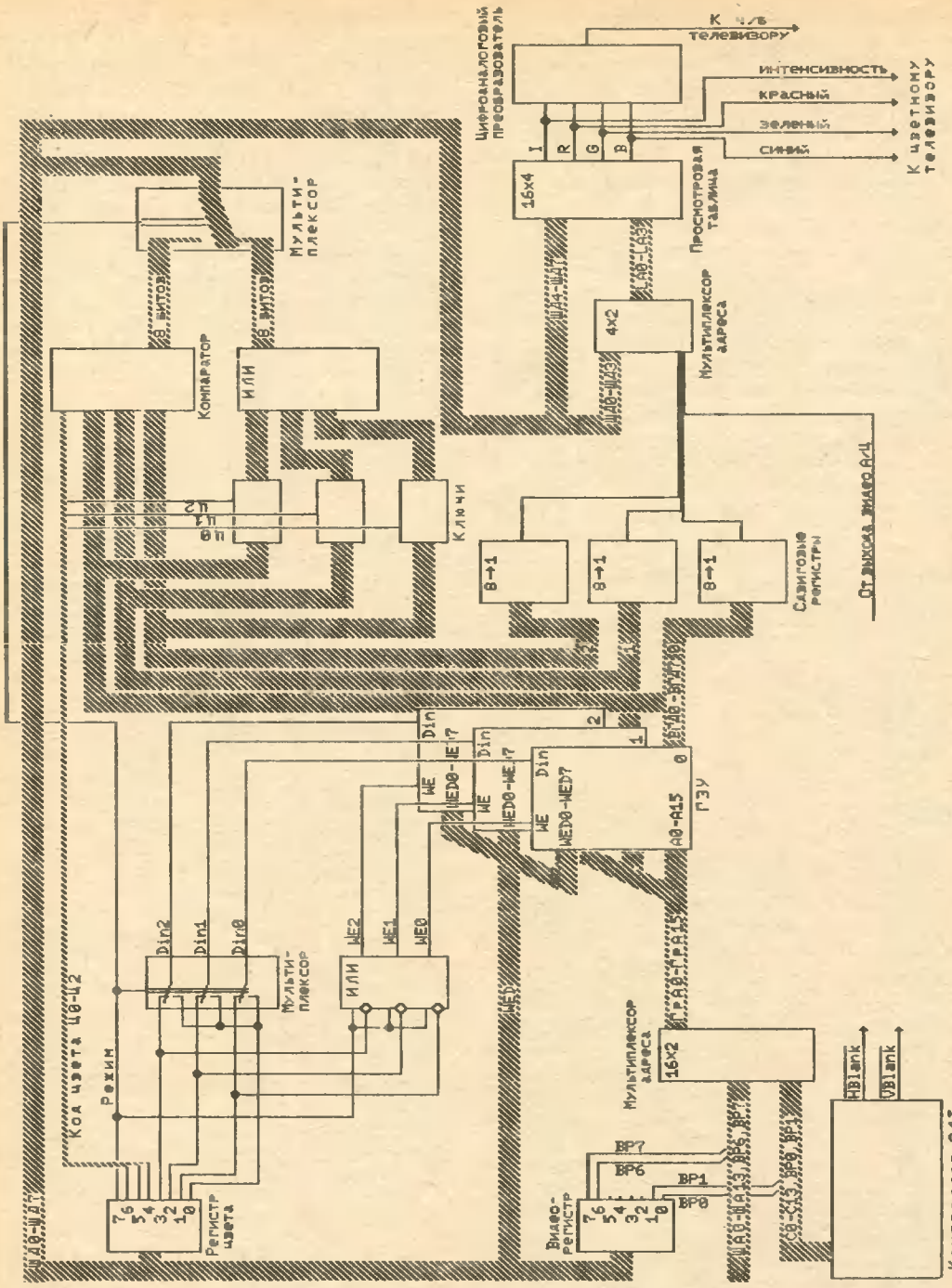
Оба механизма реализуются сходными способами — с использованием образцов-масок. Во втором случае, чтобы не считывать данные из ГЗУ перед их записью, а прямо записывать только нужные биты, используется маска разрешения записи. Ее роль играет байт данных, поступающий из процессора. Нули в его битах блокируют запись, единицы разрешают, так что в соответствующем байте ГЗУ при записи меняются только некоторые биты.

А чтобы они изменились нужным образом и одновременно во всех трех цветовых плоскостях, используется образец — регистр цвета. Поскольку адрес отдельной точки экрана в каждой плоскости один и тот же, достаточно задать в этом регистре цвет, которым она должна быть нарисована (другими словами, задать, в каких плоскостях в соответствующий бит будет записана 1, а в каких 0).

Таким образом, для высвечивания точки выполняется не девять операций, а только две — занесение кода в регистр цвета и запись байта с соответствующей единичкой в ГЗУ. Если же учесть, что цвет, как правило, меняется не часто, а среди соседствующих по одному байту восьми точек высвечиваются порой и две, и три, и все восемь, то оказывается, что в большинстве случаев одной операцией воздействуют на несколько точек. Отсюда и заметный выигрыш по времени.

Регистр цвета

```
Бит
 7 - режим (1 - цвета, 0 - плоскостной)
 6     Режим цвета: код цвета для
 5   - сравнения; плоскостной режим:
 4     разрешение чтения плоскостей
 3     Режим цвета: код цвета для за-
 2   - писи; плоскостной режим: разре-
 1     шение записи в плоскость (0 -
       да, 1 - нет)
 0 - не играет роли в режиме цвета; в
     плоскостном режиме - записываемые
     данные
```



Функциональная схема графического дисплея. Здесь Din — записываемые графические данные (data in); WE — сигнал разрешения записи (write enable); WED — маска разрешения записи (write enable data); ВР — сигналы с видеорегистра; ГрА — адрес в ГЗУ; LA — адрес в просмотровой таблице (от look up table address); ВГД — видеографические данные.

Компаратор сравнивает значения троек битов, поступающих с трех плоскостей ГЗУ, со значениями битов кода цвета.

Мультиплексор адреса просмотровой таблицы в зависимости от ситуации пропускает в качестве адреса к таблице либо биты с шины данных (при обращении к таблице процессора), либо биты с выходов плоскостей ГЗУ и А/Ц (при осуществлении процесса отображения данных на экране).

К сожалению, такой режим доступа к ГЗУ (он называется цветовым) не позволяет обращаться к отдельным байтам (а это бывает необходимо, если мы, например, используем одну из плоскостей для хранения не графических, а обычных данных). Поэтому реализован и другой режим (плоскостной). Он включается записью 0 в бит 7 регистра цвета.

В плоскостном режиме биты 1—3 регистра цвета превращаются в биты разрешения записи в отдельные плоскости: значение 0 (внимание!) разрешает запись в соответствующую плоскость, а 1 — запрещает.

Данные, поступающие из процессора, воздействуют на процесс записи так же, как в режиме цвета, т. е. разрешают или запрещают запись в биты заданного байта ГЗУ (это несколько неудобно, но зато позволяет не усложнять схему дисплея); какая именно цифра, 0 или 1, записывается, определяет содержимое бита 0 регистра цвета.

Приведем пример: запись в плоскостном режиме произвольного байта (обозначим его С) по адресу А в плоскости 0 ГЗУ:

установить регистр цвета (бит 7=0, бит 0=0, бит 1=0, биты 2—3=1);

записать по адресу А 11111111В (в результате ячейка А плоскости 0 ГЗУ обнулится);

установить регистр цвета (бит 7=0, бит 0=1, бит 1=0, биты 2—3=1);

записать по адресу А байт С (биты, где записаны 0, не изменятся, а где 1 — изменятся на 1, в результате будет записан байт С).

Итак, с записью в ГЗУ мы разобрались. При чтении оттуда тоже есть свои хитрости, в частности два режима обращения — плоскостной и цвета.

Чтение в графических операциях требуется довольно редко, чаще всего при заливке (закрашивании нарисованного контура). Поскольку эта операция является еще и самой длительной, чтение в режиме цвета оптимизировано в расчете на ее выполнение: оно выдает результат сравнения кода цвета заданной точки экрана и содержимого битов 4—6 регистра цвета (если код цвета точки совпадает с битами 4—6, то соответствующий бит читаемого байта будет 1, в противном случае — 0). Таким образом процессор получает результат сравнения сразу по восьми точкам.

Основную работу по закраске контура будут делать всего две операции:

чтение байта из графической памяти и проверка выхода на границу; если 0 (отсутствие границы), то

запись байта в графическую память (закрашивание). Если же считанный байт не 0, значит, достигнута граница контура.

При чтении в плоскостном режиме биты 4—6 регистра цвета выполняют функцию разрешения чтения: если бит установлен в 1, чтение из соответствующей плоскости разрешено, если 0 — запрещено. Одновременно можно разрешить чтение из нескольких плоскостей; в этом случае получаемый байт есть результат операции поразрядного ИЛИ между считанными из каждой плоскости байтами.

И наконец, последний механизм воздействия на изображение. Это просмотровая таблица (на рис. 3 она изображена в правом нижнем углу), позволяющая изменять соответствие реальных цветов цветовым кодам. Например, коду 5 (логическому цвету 5) может быть поставлен в соответствие любой из восьми реальных цветов, в том числе и черный.

Просмотровая таблица — это 16 4-разрядных ячеек памяти. На нее поступает четыре потока информации: из трех плоскостей ГЗУ и А/Ц дисплея. Из этих четырех битов формируется адрес, определяющий номер ячейки таблицы, содержимое которой будет выведено на экран в очередной его точке. Адрес формируется так:

бит 0 — плоскость 0 ГЗУ;
бит 1 — плоскость 1 ГЗУ;
бит 2 — плоскость 2 ГЗУ;
бит 3 — А/Ц дисплей.

Содержимое каждой ячейки таблицы, в свою очередь, задает реальный цвет:

бит 0 — включение голубого цвета;
бит 1 — зеленого;
бит 2 — красного;
бит 3 — интенсивность (1 — полная, 0 — половинная).

Если мы хотим получить некий цветовой эффект, изменяя содержимое просмотровой таблицы, нам необходимо записать новое значение в ячейку, соответствующую требуемому коду логического цвета, через специальный регистр (назовем его регистром просмотровой таблицы). Он имеет адрес FBH на странице управляющих регистров. В его биты 0—3 записывается номер ячейки просмотровой таблицы, в биты 4—7 — код физического цвета, записываемый в ячейку (бит 7 задает интенсивность (яркость) изображения, бит 6 — наличие красного, бит 5 — зеленого, бит 4 — синего цвета).

Так, если мы хотим, чтобы точка, которой назначен логический цвет 5, была зеленого цвета, нужно записать в регистр просмотровой таблицы 00100101В. Если мы хотим сделать ее ярко-зеленой, нужно записать 10100101В.

Теперь мы знакомы со всеми тонкостями устройства графического дисплея. Впрочем, есть у него еще одна особенность. ГЗУ мо-

жет быть собрано из микросхем двух типов: емкостью 16 Кбит либо 64 Кбит. В последнем случае объем ГЗУ достигает 192 Кбайт. Это очень полезное добавление, которое можно использовать, в частности, в качестве электронного диска. К сожалению, оно встречается довольно редко, поскольку стоимость микросхем емкостью 64 Кбит более высока.

В ГЗУ увеличенного объема можно организовать четыре независимые графические страницы; каждая из них может хранить свое изображение, причем переключаются они мгновенно. Более того, можно одну страницу отображать на экране, а на другой готовить следующий кадр.

Как управлять страницами графической памяти, можно понять из рис. 3. Биты 0 и 1 видеорегистра (он уже встречался нам при описании А/Ц дисплея) задают номер отображаемой страницы, а биты 6 и 7 — к какой странице в данный момент обращается процессор.

Пора приступать к примерам. Напомним, что обращение к ГЗУ требует переключения карты памяти, вследствие чего переписывать последующие программы на Бейсике нельзя, а при работе с Express Pascal следует соблюдать определенные правила. Для простоты условимся запускать программу только из среды Express Pascal (компиляция в память). Кроме того, на время переключения карты памяти необходимо запрещать прерывания.

При обращении к ГЗУ нам понадобится регистр цвета. Он относится к группе регистров «только для записи» и в конфигурации 1CH (конфигурация CP/M) имеет адрес 0FABFH.

Для начала — традиционное развлечение: зажигание отдельных точек (звездное небо), но не обычным путем, через функции языка программирования, а непосредственным воздействием на ГЗУ.

```
{пример 12 , звездное небо}
program Stars;
const
      GrRamBase      = $4000;
      SysRegister    = $FA7F;
        { адрес системного регистра }
      ColorRegister  = $FABF;
                       { регистра цвета }
      SysGrRegister  = $FF7F;
            {адрес системного регистра
                      в конфигурации $3C}
      SysConfig      =   $1C;
                      {конфигурация CP/M}
      GraphConfig    =   $3C;
 {конфигурация при включенной графике}
      ESC            =    58;
      SPACE          =    32;
      iDI            =   $F3;
      iEI            =   $FB;
      Convert:array[0..7] of byte =
            ($80,$40,$20,$10,8,4,2,1);
    {для преобразования номера точки }
var       Addr : integer;
          x,y : integer;
                        { для координат }
          Color: byte;
          Mask : byte;

begin                   { тело программы }
  ClrScr;
  ClrGScr;
  Randomize;
  repeat
     x := Random(512);
     Color := Random(8);
     y := Random(256);
     mem[ColorRegister] :=
                  $80 OR (Color shl 1);
                                  {цвет}
     Mask := Convert[x and 7];
     Addr := GrRamBase+y*64+(x shr 3);
     inline (iDI);
              { запретили прерывания }
     mem[SysRegister] := GraphConfig;
     mem[Addr] := Mask;
     mem[SysGrRegister] := SysConfig;
    inline (iEI);
              { разрешили прерывания }
  until KeyPressed;
end.
```

Продолжение следует.

С. АХМАНОВ, А. НЕЧАЕВ, А. СКУРИХИН

Архитектура процессора КР580ВМ80А

На этом можно закончить с однобайтными пересылками. В качестве отступления еще раз отметим, что «пересылка» в данном случае слово не совсем точное. Все перечисленные команды не пересылают данные, как, например, письма — изымая их из одного места и помещая в другое, а копируют, записывают в новое место, оставляя в неприкосновенности оригинал.

Приступим к рассмотрению команд, пересылающих сразу по два байта. Многие из них схожи с перечисленными выше, но есть и новая, очень важная, подгруппа команд, обеспечивающая использование стека — обмен со стеком, его загрузка, загрузка указателя стека SP. Кроме того, появляются команды, не только пересылающие, но и обменивающие значения.

N п/п	Мнемоника и длительность выполнения	Код	Действие
12	LXI RR,NN 10 тактов	0 0 R R 0 0 0 1 N N N N N N N N N N N N N N N N	Загрузка 16-разрядного числа в регистровую пару
13	LHLD NN 16 тактов	0 0 1 0 1 0 1 0 N N N N N N N N N N N N N N N N	Пересылка в регистровую пару HL 16-разрядного числа из ячейки памяти
14	SHLD NN 16 тактов	0 0 1 0 0 0 1 0 N N N N N N N N N N N N N N N N	Пересылка 16-разрядного числа из регистровой пары HL в память
15	PUSH RR 11 тактов	1 1 R R 0 1 0 1	Пересылка содержимого регистровой пары в стек
16	POP RR 11 тактов	1 1 R R 0 0 0 1	Пересылки 16-разрядного числа из стека в регистровую пару
17	SPHL 6 тактов	1 1 1 1 1 0 0 1	Пересылка содержимого HL в SP
18	XCHG 4 такта	1 1 1 0 1 0 1 1	Обмен 16-разрядными числами между HL и DE
19	XTHL 18 тактов	1 1 1 0 0 0 1 1	Обмен содержимым между HL и стеком

12. Загрузка 16-разрядного числа в регистровую пару. Мнемоника команды:

```
LXI RR,NN
```

Ее код состоит из трех байтов:

```
00RR0001
NNNNNNNN
NNNNNNNN
```

Первый байт определяет тип операции (биты 7, 6, 3, 2, 1, 0) и имя регистровой пары (биты 5 и 4).

Пример:

```
LXI H,3210H
```

или, в цифровом виде,

```
00100001B=21H
00010000B=10H
00110010B=32H
```

загрузит в HL число 3210H.

В мнемонике команды используется только имя старшего регистра пары, например, H обозначает HL; однако SP обозначается полностью:

```
LXI SP,NN
```

13. Пересылка 16-разрядного числа из ячеек памяти в регистровую пару HL:

```
LHLD NN
```

(load HL direct — прямая загрузка в HL). При ее выполнении содержимое ячеек памяти с адресами NN и NN+1 пересылается в регистровую пару HL.

Посмотрим, что произойдет при выполнении команды

```
LHLD 5678H
```

или, в цифровом виде,

```
00101010B=2AH
01111000B=78H
01010110B=56H
```

Пусть ячейка памяти 5678H содержит число 01H, а ячейка 5679H — число 02H. Тогда в HL запишется число 0201H, т. е. в L будет храниться 01H, а в H — 02H.

14. Пересылка 16-разрядного числа из регистровой пары HL в память:

```
SHLD NN
```

(store HL direct). При выполнении, к примеру, команды

```
SHLD 5678H
```

или, в цифровом виде,

```
00100010B=22H
01111000B=78H
01010110B=56H
```

содержимое регистра L помещается в ячейку памяти с адресом 5678H, а содержимое H — в ячейку с адресом 5679H; если в HL хранилось число 1011H, то в ячейке 5678H будет записано 11H, а в 5679H — 10H.

Как вы, наверное, обратили внимание, во всех командах, имеющих отношение к 16-битным числам, сначала загружается младший байт числа (он обозначается LSB от англ. least significant byte — младший значащий байт), а потом уже старший (MSB — most significant byte — старший зна-

Продолжение. Начало см.: Информатика и образование. 1992. № 1, 2.

чащий байт). Это соглашение строго соблюдается во всех командах.

15. Пересылка содержимого регистровой пары в стек.

Здесь прежде всего напомним, что хотя стек есть не какое-либо особенное устройство, а лишь особым образом используемая область ОЗУ, и обмен с его ячейками можно было бы выполнять и обычными командами, для обращения к нему всегда используется SP и только SP. Причина проста — в КР580ВМ80 аппаратно реализован механизм, «заставляющий» содержимое SP всегда указывать на вершину стека, т. е. то место в памяти, где находится «последний патрон».

Итак, команда

```
PUSH  QQ
```

(от push — толкать, отсюда и жаргонное «выталкивать в стек») пересылает в стек содержимое одной из регистровых пар BC, DE, HL или AF (PSW), обозначенных в этой мнемонике QQ. Код команды:

```
11QQ0101
```

Имя регистровой пары кодируется значениями битов 5 и 4 следующим образом:

```
BC - 00
DE - 01
HL - 10
PSW - 11
```

При выполнении этой команды происходят довольно сложные действия. Рассмотрим процесс выталкивания в стек содержимого HL (обратите внимание, что в мнемонику входит только имя старшего регистра):

```
PUSH H
```

Сначала содержимое регистра L помещается в ячейку памяти с адресом, равным содержимому регистра SP минус 2 (обозначим (SP)—2), а содержимое регистра H — в ячейку с адресом (SP)—1; затем содержимое SP уменьшается на 2. Если в HL хранилось число 0102H, а в SP — 9999H, то после выполнения команды в ячейке 9998H будет число 01H, в ячейке 9997H — число 02H, а (SP) будет равно 9997H; (SP) указывает на ячейку памяти с последним загруженным в стек числом.

Еще один пример. Пусть в аккумуляторе содержится число 55H, в регистре флагов — 43H, а в SP — 1005H. После выполнения команды PUSH PSW в ячейке 1004H будет число 55H, в ячейке 1003H — 43H, а в SP — 1003H.

16. Пересылка 16-разрядного числа из стека в регистровую пару:

```
POP QQ
```

(не звук ли пробки, выскакивающей из бутылки, стал прообразом этой мнемоники?). При ее выполнении число из вершины стека (адрес которой всегда хранится в SP) помещается в регистровую пару, (SP) увеличивается на 2.

Попробуем выполнить команду

```
POP H
```

Пусть в SP записано число 2345H, в ячейке памяти с адресом (SP), т. е. 2345H, записано 33H, а в ячейке с адресом (SP)+1, т. е. 2346H, записано 44H. После выполнения команды в регистр L попадет 33H, в H — 44H, а в SP будет храниться 2347H; (SP) указывает на новую вершину стека.

17. Пересылка (HL) в SP:

```
SPHL
```

Содержимое регистровой пары HL просто копируется в регистр SP.

18. Обмен двухбайтными словами между регистровыми парами HL и DE:

```
XCHG
```

(корявое сокращение слова exchange — обмен). То, что было в HL, попадет в DE, а то, что было в DE, попадет в HL.

19. Обмен содержимым между регистровой парой HL и стеком:

```
XTHL
```

(возможно, от exchange top HL).

Допустим, что в SP хранилось число 1234H; в вершине стека, т. е. в ячейках памяти с адресами 1234H и 1235H — числа 33H и 22H соответственно; в L — число 55H; в H — 44H. Тогда после выполнения команды XTHL в ячейках памяти с адресами 1234H и 1235H будут храниться числа 55H и 44H соответственно, а в регистровой паре HL — число 2233H. Содержимое SP не изменится.

На этом можно подвести черту под командами пересылки. Отметим, что на регистр флагов F воздействует только одна из них — POP PSW.

Продолжение следует.

В. РЕДЧЕНКО

«Заплатка» на программе

Подпрограммы в машинных кодах, часто включаемые в Бейсик-программы, выглядят в них эдакими «заплатками» из другой ткани, зато заметно увеличивают их эффективность. Использование таких подпрограмм дело непростое, требующее определенной квалификации, однако многие любители освоили его. Ну а включение подпрограмм в чужие программы в машинных кодах? Кто осмелится на такое?

Оказывается, не так страшен черт. Докажем это на примере.

В хорошем текстовом редакторе для «Корвета» WMR есть один существенный недостаток: нельзя перемещать курсор клавишами дополнительного поля. Попробуем исправить его, перекодировав все эти клавиши (в том числе DEL, INS, CLS).

Клавиша 2 должна перемещать курсор вниз. В инструкции к WMR («Диалог. Пакет 3») можно узнать, что для этого она должна генерировать код 1E (здесь и далее коды и адреса шестнадцатиричные), соответствующий управляющему символу УПР+ +Ч. В «Руководстве системного программиста» находим, что код клавиши 2 хранится в ячейке с адресом F6AC. Записать туда 1E можно двумя командами ассемблера:

```
MVI A,1E
STA F6AC
```

В кодах они записываются так: 3E, 1E, 32, AC, F6.

Примерно такие же цепочки кодов команд нужны для перекодировки остальных клавиш. Добавим, кроме того, еще две команды, включающие нижний регистр русских букв (для этого в ячейки F72D и F72E нужно записать нули). Вся программа будет выглядеть так.

```
3E 10 32 A0 F6
3E 06 32 A6 F6
3E 1E 32 AC F6
3E 01 32 B2 F6
3E 19 32 B8 F6
3E 12 32 BE F6
3E 17 32 C4 F6
3E 08 32 CA F6
3E 15 32 D0 F6
3E 13 32 D6 F6
3E 05 32 DC F6
3E 14 32 E2 F6
3E 1A 32 E8 F6
3E 07 32 EE F6
3E 00 32 2D F7
3E 00 32 2E F7
C9
```

Последний код — команда RET.

Теперь оформим ее в виде отдельного файла, который назовем KEYWM.COM.

Чтобы записать эти коды на диск, используйте программу POWER.COM (команды DS и SAVE) или DU.COM, в которой можно быстрее и с меньшей вероятностью ошибки набирать и записывать информацию, чем в POWER, которая все равно, однако, понадобится для создания файла на диске (команды READ, SAVE).

Приступим. Загружаем DU.COM: A>DU. Выбираем пустую дорожку на диске, задаем номер дорожки: T43. Задаем номер сектора: S17. Нажимаем D, на экран выводится дамп этого сектора. Запишем в него коды, начиная с адреса 00 (лучше небольшими группами):

```
CH00,3E,10,32,A0,F6,3E,06,32,A6,F6<BK>
```


10 кодов занесены (пока это только копия сектора в ОЗУ!). Чтобы посмотреть их в дампе, нажмем еще раз D. Убедившись, что все в порядке, действуем дальше — вводим очередную порцию кодов (теперь уже с адреса 0A):

```
CH0A,3E,1E,32,AC,F6,3E,01,32,B2,F6<BK>
```

И т. д. Когда занесены все коды, нужно заполнить остальные ячейки сектора кодом 00 (это не обязательно, но желательно — потом с таким сектором удобнее будет работать).

Записываем сектор на диск: W. Выйдем из DU в систему: X. Загрузим POWER и дадим команды

```
A>READ 43 17 4000 1
A>SAVE KEYWM.COM 4000 1
```

Убедившись, что на диске появился файл KEYWM.COM, попробуем с ним работать. Загрузим его:

```
A>KEYWM
```

Теперь, если попытаться набрать команду

```
A>WMR PROBA.TXT
```

(PROBA.TXT — файл, который мы будем создавать в редакторе), то будут набираться русские буквы нижнего регистра, поэтому команду надо набирать, нажав одновременно клавиши АЛФ и РГ и удерживая их.

Набирая текст, передвигая курсор, удаляя символы или вставляя новые, проверим, как работает редактор. Если что-то нас не устраивает, можно внести изменения в файл KEYWM.COM, если все в порядке, проделаем следующее:

объединим файлы WMR.COM и KEYWM.COM;

заставим добавленные коды работать в виде подпрограммы.

Первая задача решается просто:

```
A>COPY WM,COM=WMR.COM,KEYWM.COM[VOR]
```

Получим объединенный файл WM.COM из 77 секторов: 76 — WMR.COM, 1 — KEYWM.COM.

При загрузке в ОЗУ WM.COM будет располагаться начиная с адреса 0100, 77-й сектор — с адреса 2700, а последний код (09) — по адресу 2750.

Где-то в начале программы, до того как редактор начнет работу с текстом, надо организовать переход к подпрограмме по адресу 2700. Для этого годится любая из команд JMP и CALL, занимающих три байта каждая, но где ее разместить? Надо использовать одну из трех возможностей:

команда JMP;

команда CALL;

три «пустые» ячейки подряд — NOP.

Используем дизассемблер SID.COM:

```
A>SID WM.COM
```

Дадим команду L. На экране — команды WM.COM. Нам повезло: первая же команда — JMP 0269, она располагается, начиная с адреса 0100, т. е. в ячейках 0100, 0101, 0102, и в кодах имеет вид C3 69 02, в чем нетрудно убедиться, дав команду D и посмотрев дамп. В эти ячейки нужно записать команду перехода к нашей подпрограмме JMP 2700, которая в кодах имеет вид C3 00 27. Это можно сделать с помощью POWER.COM. Надо загрузить файл с адреса 4100 (командой LOAD) и с помощью команды DS в ячейки 4100, 4101, 4102 записать вышеупомянутые коды. Затем в адрес 6750 и далее записать три кода: C3 69 02. Эти три кода заменят в нашей подпрограмме C9 00 00 и обеспечат правильный вход в основную программу.

Осталось только сохранить измененную программу командой SAVE.

Редактор работает. Если вам не нравится, что по выходе из него кодирование клавиш сохраняется (а также остается включенным нижний регистр русских букв), можно сделать (таким же образом) программку KEY.COM, изменяющую все коды на стандартные.

WM.COM хорошо работает в МикроДОС разных версий, передается системой SEND по локальной сети на РМУ, где работает с любыми текстами. Известный редактор ED.COM в этой системе не работает на РМУ с файлами, загруженными с диска.

Учтите, что в CP/M-80 не переключается верхний регистр и не работают клавиши DEL, INS, CLS (вместо них можно использовать УПР+T, УПР+Z, УПР+G). Черные клавиши дополнительного поля работают нормально. Можно было бы усложнить программу и сделать ее универсальной, но в данном случае эта задача не ставилась. Целью автора было обратить внимание читателей на следующее:

рядовой пользователь может делать весьма полезные вещи относительно простыми средствами;

не следует бояться программ в кодах — их тоже можно читать, писать, модифицировать;

чем больше мы, пользователи «Корвета», будем общаться (в частности, через «ИНФО»), тем легче и интереснее нам будет работать.

О. ДЕРГАЛЕНКО

Инструментальные оболочки для КУВТ «Нейва» («Корвет»)

Мы уже дожили до того времени, когда компьютеры в учебных заведениях стали использовать не только на уроках информатики, но и при обучении далеким от вычислительной техники дисциплинам. Это стало возможно благодаря появлению специальных обучающих и контролирующих программ, позволяющих учителю, например, английского языка, обладающему лишь элементарными компьютерными навыками, использовать ПЭВМ на своих уроках. Эффективность обучения с применением компьютера хотя бы только для проведения контрольных опросов, как правило, выше, чем при традиционном безмашинном обучении; автор утверждает это, основываясь на своем богатом опыте.

Как правило, для создания подобных обучающих и контролирующих программ требуется целый коллектив разработчиков,

включающий программиста и методиста, хорошо владеющего предметом, для которого создается программа. Кроме того, желательно присутствие психолога и дизайнера. В нашей стране много замечательных преподавателей, хороших программистов, профессиональных психологов и дизайнеров, но нечасто встретишь такой коллектив в одном учебном заведении, например в школе. Поэтому пока существует не так уж много хороших обучающих и контролирующих программ.

Многие преподаватели сами могут разработать методику обучения или контроля знаний по своему предмету, но для переложения ее на компьютер требуется квалифицированный программист. В таких случаях могут помочь автоматизированные обучающие системы (АОС). Свою часть работы программисты и дизайнеры сделали, создавая их; педагог может сделать свою часть самостоятельно. АОС представляет собой набор инструментальных средств для создания специальных объектов — вопросников — и средства их интерпретации, осуществляющие диалог ученика с компьютером.

«Учительская» часть АОС похожа на специализированный текстовый редактор для создания и изменения вопросников. Вопросник же содержательно может соответствовать одному уроку по какому-либо предмету, или зачету по какой-то теме, или объяснению темы и т. п. Структурно вопросники состоят из кадров. Кадр — это та порция информации, которая выдается ученику на экран за один раз (например, вопрос или определение какого-либо понятия). Если кадр содержит вопрос, то интерпретатор вопросника ждет, пока ученик наберет на клавиатуре или выберет из предложенных правильный, по его мнению, ответ. Затем интерпретатор, пользуясь правилом, заложенным в него учителем, принимает решения о правильности ответа и о том, какие действия предпринимать дальше, например какой кадр загрузить следующим.

Ниже предлагается обзор известных автору АОС для КУВТ «Нейва» («Корвет», «Форманта»).

BAZA v. 3.2 (пакет ОБУЧ-2) (автор А. Пиунов, г. Каменск-Уральский). Система состоит из двух программ, написанных на Бейсике: BAZAP4 для РМП и BAZAU4 для РМУ. Кроме того, в комплект поставки входят несколько вопросников. BAZAU4 представляет собой интерпретатор вопросников, а BAZAP4, кроме того, содержит инструментальные средства для создания вопросников. Данную систему нельзя, строго говоря, назвать АОС, так как она не позволяет создавать обучающие вопросники.

Каждый кадр вопросника содержит один вопрос, который выдается интерпретатором ученику на экран, и единственный шаблон ответа. Шаблоном (или образцом) называется строка, с которой должен сравниваться ответ ученика. Если введенный учеником ответ совпадает с шаблоном, он принимается как правильный, и ученику сообщается об этом. Если ответ не совпадает с шаблоном, то выдается сообщение: «Вы не знаете правильный ответ!» — и на экран выводится шаблон. Крупным недостатком системы является то, что сравнение идет «жестко», т. е. строчные и прописные буквы интерпретатор считает различающимися, все пробелы учитываются.

Вопросы ученику выдаются последовательно или выбираются из заданного диапазона случайным образом. В случае случайного выбора вопросы могут повторяться. На РМУ интерпретаторы рассылаются программой STS, загружаются с магнитофона или средствами сетевой операционной системы (СОС) «Корнет». Неудобен процесс загрузки и сохранения вопросников на ГМД или магнитной ленте — ученик должен давать команды Бейсика LOAD и SAVE.

После опроса можно просмотреть итоговую информацию: сколько было сделано попыток работать с вопросником, на сколько вопросов был дан правильный ответ, а на сколько — неправильный.

В силу того что BAZA написана на Бейсике, она является открытой системой, т. е. пользователь может вносить свои изменения, улучшающие, с его точки зрения, систему. Но это доступно только владеющим программированием.

ТРЕНАЖЕР (пакет ОБУЧ-3) (авторы М. Бердюгин, А. Мельников, СГМИ, г. Екатеринбург). В состав системы входят инструментальная часть для РМП TRENRMP, интерпретатор вопросников TRENRMU и специализированная контролирующая система для проверки англо-русского перевода с возможностью редактирования словаря. Аналогично системе BAZA, ТРЕНАЖЕР написан на Бейсике и позволяет создавать только контролирующие вопросники. Каждому вопросу может быть придано от 1 до 4 вариантов ответов, из которых только один правильный. Ученик не набирает свой ответ, а выбирает один из предложенных. После того как все вопросы исчерпаны, выдается итоговая информация.

Недостатком по сравнению с BAZA является отсутствие в инструментальной части интерпретатора, т. е. возможности сразу же проверить создаваемый вопросник. Редактирование вопросников, как и в BAZA, осуществляется покадрово вводом вопроса или ответов заново.

Вопросник, создаваемый системой, имеет весьма прозрачную структуру (как и у

BAZA), так что ученику нетрудно подсмотреть правильный ответ (достаточно нажать клавишу **СТОП** и просмотреть память).

АОКС «Хорда» v. 2.4 (авторский коллектив п/р О. Дергаленко, г. Каменск-Уральский). Автоматизированная обучающая и контролирующая система «Хорда» является полной АОС (позволяет создавать и обучающие, и контролирующие вопросники). Можно также создавать вопросники, в которых обучающий и контролирующий режимы присутствуют одновременно. Структуру вопросника, сценарий диалога компьютера с учеником учитель определяет сам. Вопросник может содержать циклы, произвольные переходы от одного кадра к другому. Кадр (16 строк по 64 символа в строке) может быть вопросом или сообщением, не требующим ответа ученика (например, объяснением какого-либо понятия). Ответ может выбираться учеником из предложенных вариантов или вводиться в произвольной форме. В последнем случае учитель может задать от 10 шаблонов ответов, среди которых могут быть правильные, неправильные или нейтральные. Нейтральные ответы используются при запросе у ученика дальнейших действий (например, повторить все сначала или закончить работу). В зависимости от ответа следующим будет загружен на экран один из 10 кадров, заданных в шаблонах.

Например, обучающий вопросник можно сконструировать следующим образом. Ученику объясняется какой-то материал, т. е. текст и рисунки выводятся на экран. Затем по пройденному материалу ему задаются контрольные вопросы. Если ученик правильно отвечает на вопросы, ему выдается следующая порция информации, иначе вновь выдается пройденный материал для повторения.

Можно построить вопросник так, что ученику будут прощаться мелкие ошибки, неточности, незнание правил правописания. Сравнение ответа ученика с шаблоном происходит всегда «мягко», т. е. пробелы игнорируются, строчные и прописные символы не различаются. Кроме того, в шаблоне можно использовать специальные символы, еще более «смягчающие» сравнение.

В вопросниках можно использовать рисунки, созданные с помощью псевдографических символов шрифта или графического редактора GRAF.

Большим преимуществом перед BAZA и ТРЕНАЖЕРом является наличие загружаемого шрифта и инструмента для создания шрифта под конкретную дисциплину. Благодаря этому в вопроснике можно использовать специальную символику дисциплин, например математические значки, элементы электрических схем, а также создавать вопросники на языках государств СНГ.

Вопросник АОКС должен полностью помещаться в ОЗУ, что ограничивает его размер примерно 24000 символами. Если этого недостаточно, материал можно разбить на несколько вопросников и загружать их автоматически один за другим, не прерывая обучения.

В процессе диалога с учеником АОКС может формировать на диске протокол сеанса работы, куда записываются имя ученика, имя вопросника, номера выдававшихся кадров, ответы ученика и оценки, данные системой. Наличие протокола позволяет легко разрешать конфликтные ситуации после контрольных опросов. Кроме того, система формирует журнал, куда записывает информацию об ученике и количество вопросов, на которые он дал правильные ответы. Вопросник и протокол кодируются, что исключает возможность подсмотреть правильный ответ.

Недостатком АОКС является отсутствие какой-либо подсказки при работе с редактором вопросников; впрочем, система его команд в основном совпадает с имеющимися в популярных текстовых редакторах МикроМир, Е и легко запоминается.

В состав пакета входят редакторы шрифтов и вопросников, два интерпретатора вопросников, программа просмотра и печати вопросников, программа проверки вопросников, программа просмотра и печати протокола, кодировщик вопросников и другие полезные утилиты.

АОКС является полностью закрытой системой, т. е. пользователь не может расширить возможности программ, входящих в ее состав. Но структура вопросника, кстати, весьма простая, описана в документации, что позволяет пользователю писать собственные программы работы с вопросниками.

Для работы на РМУ в составе КУВТ необходима СОС «Корнет».

АОС «Радуга» (МП «Микс» НИИЯФ МГУ, Москва). По своей идеологии «Радуга» очень близка к АОКС, точнее, АОКС похожа на «Радугу», так как «Радуга» появилась значительно раньше. Она включает несколько интерпретаторов, что обеспечивает ей более широкие возможности, чем АОКС. АОС использует оконный интерфейс, т. е. пользователь может оформлять в виде отдельных окон сам вопрос или сообщение, варианты ответов в меню, реакции на ответ ученика. Каждому окну можно задать цвет фона, при этом цвет символов изменяется автоматически так, чтобы они хорошо читались. Используются также загружаемые шрифты. Редактор вопросников позволяет осуществлять полноэкранное покадровое редактирование. Система команд редактора не-

сколько тяжеловата, но этот недостаток компенсируется встроенной подсказкой. Редактор позволяет копировать кадры не только из текущего загруженного вопросника, но и из других вопросников с диска.

Редактор шрифтов позволяет изменять очертания всех 256 символов (в АОКС нельзя изменить символы с кодами от 0 до 31). Вопросник, как и в АОКС, обычно должен умещаться в память полностью, однако в состав АОС входят интерпретаторы, способные подгружать кадры вопросника с диска по мере необходимости. При этом РМУ полностью занято обслуживанием интерпретатора, зато размер вопросника ограничивается только емкостью диска.

АОС «Радуга» имеет собственные сетевые средства. Без СОС «Корнет» АОКС запустить на РМУ нельзя, а «Радугу» можно.

«Радуга» сопровождается очень добротной, подробной, ориентированной на новичка документацией; единственный ее недостаток — то, что она описывает версию АОС для КУВТ «Ямаха», и, хотя есть глава об отличиях реализации для «Корвета», некоторые вопросы остаются открытыми. В частности, каким образом формировать графические файлы, если приводимые команды для Бейсика «Ямахи» отсутствуют в Бейсике «Корвета»?

«Радуга» является более открытой системой, чем АОКС. В документации описывается не только структура вопросника, кстати сравнительно сложная, но и процедуры доступа к нему из программ, написанных на Паскале; однако исходные тексты этих процедур отсутствуют в поставке.

В состав «Радуги» кроме редакторов и интерпретаторов входят программа печати вопросников, программа формирования линейных вопросников и несколько вопросников, демонстрирующих возможности си-

Основные характеристики пакетов

	BAZA	ТРЕНАЖЕР	АОКС	«Радуга»
Размер вопросника, байты	5000	1148	24 000	20 000
Число кадров в вопроснике, максимально	Зависит от размера	10	99	99
Размер вопроса, строки	4[1]	2	16	30[2]
Число вариантов ответа	10	4	10	30[2]
Размер варианта ответа, знаки	250[1]	255	30	30[2]
Назначение вопросника	Только	контроль	Контроль и	обучение
Загружаемые шрифты	—	—	+	+
Интерпретация ответов:				
сравнение с образцом	+	—	+	+
числовые с округлением	—	—	—	+
несравнение с образцом	—	—	—	+
выбор из меню	—	+	+	+
Изменение диалога в зависимости от ответа ученика	—	—	+	+
Изменение диалога в зависимости от количества правильных ответов	—	—	+	—
Полноэкранное покадровое редактирование	—	—	+	+
Копирование кадров из текущего вопросника	—	—	+	+
Копирование кадров из другого вопросника	—	—	—	+
Подсказка в редакторе	+	+	—	+
Прозрачность вопросника	+	+	—	+
Автоматический переход от одного вопросника к другому	—	—	+	+[3]
Протокол работы	+	—	+	+
Журнал работы	—	—	+	—
Встроенные сетевые средства	—	—	—	+
Подключение графических изображений	—	—	+	+
Среда: РМП	Бейсик	Бейсик	CP/M	CP/M
РМУ	Бейсик	Бейсик	«Корнет»	Нет
Печать вопросников	—	—	+	+
Работа с магнитофоном	+	—	—	—
Расширяемость	+	+	—	+
Совместимые АОС на других ЭВМ	—	—	—	+

[1] Общий размер вопроса и шаблона ответа — 250 символов.
[2] Указан максимально возможный; зависит от размера вопроса, вариантов ответов и реакций.
[3] Можно загружать разные вопросники из разных кадров.

стемы.

Версии «Радуги» реализованы для КУВТ «Корвет», «Ямаха» и для ПЭВМ типа IBM PC.

Что же выбрать? Какая из описанных систем лучше? Это зависит, конечно, от ваших потребностей и возможностей. Если требуется создавать лишь контролирующие вопросники линейной структуры, вполне приемлемы BAZA и ТРЕНАЖЕР; к тому же они самые дешевые. Пожалуй, предпочтение следует отдать системе BAZA как более понятной и надежной. Для создания обучающих систем потребуются АОКС или «Радуга». «Радуга» обладает более широкими возможностями конструирования вопросников и лучшим интерфейсом, чем АОКС. АОКС же представляется (возможно, это пристрастное мнение одного из разработчиков АОКС) более легкой в обращении, обладающей более широким набором полезных утилит. Интерпретаторы вопросников АОКС позволяют изменять диалог ученика с компьютером в зависимости от числа правильных или неправильных ответов. К тому же АОКС несравненно дешевле (200 рублей против 5000 рублей «Радуги»). Но для не имеющих СОС «Корнет» «Радуга» — единственно подходящая система.

М. СТЕПАНОВ, В. САМОЛЫСОВ

Дополнительные возможности графики ПЭВМ «Агат»

Стандартные версии интерпретатора языка Бейсик ПЭВМ «Агат» предоставляют пользователю единственный графический оператор PLOT, который можно без каких-либо затруднений использовать на занятиях по информатике. Этот оператор имеет ясный геометрический смысл (построение точки, отрезка или ломаной), кроме того, для его использования не требуется никакой подготовительной работы. Иначе обстоит дело с группой операторов DRAW, XDRAW, SCALE, ROT. В документации дано не слишком ясное их описание, из которого следует, что для применения этих операторов необходима длительная подготовка.

Цель настоящей статьи:

дать ясное описание операторов и подготовительной фазы работы;

предложить преподавателю текст инструментальной программы, предназначенной для упрощения подготовительной фазы;

описать методику использования этих операторов на уроках информатики;

описать класс простых задач, естественным образом решаемых с помощью оператора DRAW.

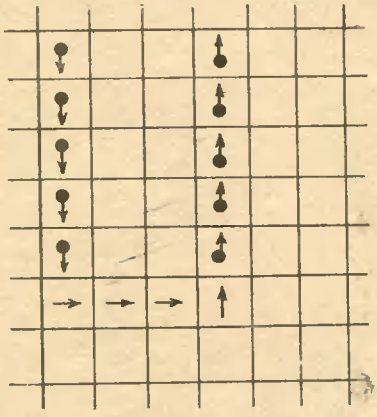
Описание операторов и подготовительной работы

Оператор DRAW служит для вычерчивания линий на экране в графическом режиме высокого разрешения. Линия вычерчивается воображаемым пером без отрыва. При движении перо может оставлять след (закрашивать пройденную точку-клеточку экрана) или не оставлять его.

Подготовка к использованию оператора DRAW состоит в создании информационного массива, задающего движение пера, и в размещении этого массива в физической памяти компьютера. Разобьем этот процесс на этапы.

1. Указание направления движения пера с помощью стрелок на клетчатой бумаге. При этом можно использовать 8 типов стрелок (табл. 1), с каждой из которых связан числовой код (рис. 1).

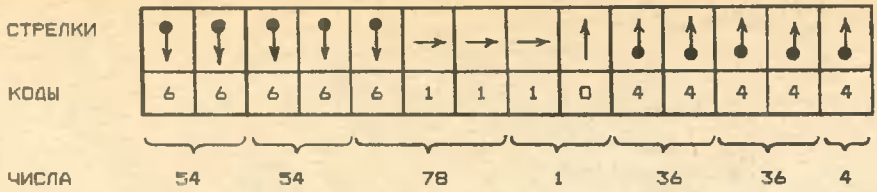
ТИП СТРЕЛКИ	КОД	ЗАКРАШИВАЕТСЯ ЛИ КЛЕТКА, С КОТОРОЙ УХОДИТ ПЕРО
↑	0	НЕТ
→	1	НЕТ
↓	2	НЕТ
←	3	НЕТ
↑	4	ДА
→	5	ДА
↓	6	ДА
←	7	ДА



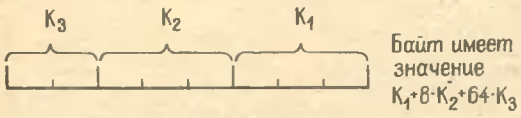
2. Замена последовательности стрелок на последовательность кодов. Последовательность стрелок, задающая движение пера, располагается в линию, и каждая стрелка заменяется соответствующим кодом (рис. 2).

3. Преобразование последовательности кодов в более компактную последовательность чисел для загрузки в байты физической памяти. Для преобразования выбираются три первых, еще не преобразованных кода — K_1, K_2, K_3. Если $K_3 \leqslant 3$, то все три кода заменяем одним числом, вычисляемым по формуле $n = k_1 + 8k_2 + 64k_3$. Если же $k_3 > 3$, то одним числом заменяем только два кода k_1 и k_2: $n = k_1 + 8k_2$.

Смысл этих преобразований заключается в распределении кодов K_1, K_2, K_3 по битам



байта, в котором будет помещено число (легко видеть, что $n<256$). Под коды K_1 и K_2 отводится по 3 бита, а под код K_3 остается только 2 бита, поэтому он не может превосходить 3 (рис. 3).



В нашем примере последовательность из 14 кодов заменится на 7 чисел (рис. 2). К последовательности чисел добавляется ноль. Он служит указателем того, что описание образа закончено.

4. Размещение последовательности чисел в физической памяти компьютера. В память компьютера может быть загружена информация сразу о нескольких образах, вычерчиваемых воображаемым пером. Вся эта информация располагается в специальной таблице, представляющей собой совокупность последовательных байтов. Адрес начального байта таблицы должен быть записан в виде шестнадцатиричного числа, запись которого затем разбивается на пару старших и пару младших цифр. Число, записываемое парой младших цифр, следует заслать в ячейку с адресом ¤E8, а число, записываемое парой старших цифр,— в ячейку ¤E9. Например, если начальный байт таблицы имеет адрес ¤1500, то следует выполнить команды

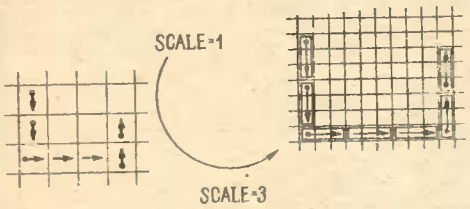
```
POKE ¤E8, 0
POKE ¤E9, ¤15
```

Если в программе будут использоваться m образов, то 2m+2 начальных байта таблицы предназначаются для специальных целей. В начальный байт всегда засылается число m, в следующий байт всегда будем засылать ноль. Далее в каждую пару байтов засылаем значения, определяющие смещение до непосредственного описания очередного образа от начального байта таблицы. Каждое смещение записывается в шестнадцатиричной системе счисления и размещается в паре соседних байтов в порядке: младшие цифры, старшие цифры, по аналогии с записью адреса таблицы (рис. 4).

Чтобы использовать команду DRAW, необходимо предварительно загрузить в физическую память компьютера таблицу описания образов, а также начальный адрес таблицы в ячейки ¤E8 и ¤E9. Загрузка может быть проведена с помощью монитора, с помощью команд POKE в прямом диалоге, с помощью специальной программы загрузки и, наконец, с помощью той же самой программы, в которой затем будет использоваться оператор DRAW.

Перейдем к описанию команд DRAW, XDRAW, SCALE и ROT. Команды DRAW

и XDRAW заставляют перо вычерчивать нужный образ, однако перед их выполнением следует установить масштаб (SCALE) и ориентацию (ROT) образа относительно экрана. Команда SCALE=M ($0 \leqslant M \leqslant 255$), дает перу указание по каждой стрелке проходить не одну точку, а количество точек, равное данному числу M (рис. 5).



Команда ROT=R ($0 \leqslant R \leqslant 255$) заставляет перо чертить образ относительно исходной точки, повернутый на угол, равный $R \cdot \pi/32$. Следует иметь в виду, что при малых масштабах воспроизведения образа оператор ROT может работать плохо. Лучше использовать масштабы не менее 10.

Оператор DRAW заставляет перо чертить образ тем цветом, который включен оператором COLOR (обычно COLOR=15). Записывается оператор DRAW в виде:

DRAW $N_{обр}$ AT X, Y

При его исполнении перо вычерчивает образ с номером $N_{обр}$, описанный в таблице. За исходную берется точка экрана с координатами X, Y.

Оператор XDRAW не связан с оператором COLOR. Перо, проходя по контуру образа, не закрашивает точки экрана, а инвертирует их. т. е. черные точки превращает в белые, а белые — в черные.

Приведем пример программы, которая строит несколько образов различных масштаба и ориентации.

```
10 REM ЗАГРУЗКА АДРЕСА ТАБЛИЦЫ
20 POKE 8E9,15
30 POKE 8E8,0
40 REM СОДЕРЖИМОЕ ТАБЛИЦЫ
50 DATA 1,0:REM ЧИСЛО ОБРАЗОВ
60 DATA 4,0:REM СМЕЩЕНИЕ ОБРАЗА
                              В ТАБЛИЦЕ
70 DATA 54,54,78,1,36,36,4,0:REM
                          ОПИСАНИЕ ОБРАЗА
80 REM ЗАГРУЗКА ТАБЛИЦЫ
90 FOR AD=8150Ø TO 8150Ø+11
100 READ B
110 POKE AD,B
120 NEXT AD
130 REM С ОБРАЗОМ МОЖНО РАБОТАТЬ
140 HGR=1:COLOR=15
150 FOR R=Ø TO 64
160 SCALE=R/3:REM УСТАНОВКА МАСШТАБА
170 ROT=R:REM ПОВОРОТ ОБРАЗА
180 DRAW 1 AT 128,128:REM ВЫЧЕРЧИВАНИЕ
                                  ОБРАЗА
190 NEXT R
200 GET A$:TEXT=0
```

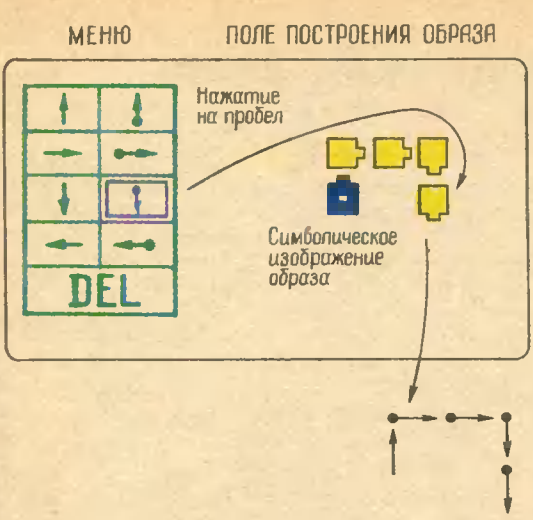
Поскольку создание таблицы довольно хлопотное занятие, а ее загрузка требует обращения к такому специфическому оператору, как POKE, применение оператора DRAW на уроках информатики осложнено. Ниже приводится текст программы, которая позволит упростить получение таблицы для достаточно простых образов и загрузит эту таблицу в память компьютера.

```
10 DIM KD(100),X(100),Y(100),B(50)
20 REM ПОСТРОЕНИЕ МЕНЮ
30 MGR=2:COLOR=2
40 FOR YY=0 TO 100 STEP 20
50 PLOT 0,YY TO 40,YY
60 NEXT YY
70 PLOT 0,0 TO 0,100
80 PLOT 40,0 TO 40,100
90 PLOT 20,0 TO 20,80
100 FOR NX=0 TO 1
110 FOR NY=0 TO 4
120 NP=NY*2+NX+1:IF NP=10 THEN NP=9
130 ON NP GOSUB 1000,1100,1200,1300,
              1400,1500,1600,1700,1800
140 NEXT NY,NX
150 NX=0:NY=0:X(0)=84:Y(0)=56:K=0
160 COLOR=5:PLOT 84,56 TO 88,56
            TO 88,60 TO 84,60 TO 84,56
170 COLOR=1:GOSUB 1900:GOSUB 1000
180 REM РАБОТА С МЕНЮ
190 GET S$:UK=ASC(S$)
200 IF UK=13 THEN 510
210 IF S$="_" THEN 360
220 IF UK=26 AND NY<4 THEN MX=NX  :
                          MY=NY+1:GOTO 270
230 IF UK=21 AND NX<1 THEN MX=NX+1:
                          MY=NY  :GOTO 270
240 IF UK=25 AND NY>0 THEN MX=NX  :
                          MY=NY-1:GOTO 270
250 IF UK=8  AND NX>0 THEN MX=NX-1:
                          MY=NY  :GOTO 270
260 GOTO 190
270 REM ПЕРЕМЕЩЕНИЕ ПО МЕНЮ
280 CR=0:CF=2
290 FOR I=1 TO 2
300 NP=NY*2+NX+1:IF NP=10 THEN NP=9
310 COLOR=CR:GOSUB 1900
320 COLOR=CF:ON NP GOSUB 1000,1100,
    1200,1300,1400,1500,1600,1700,1800
330 NX=MX:NY=MY:CR=1:CF=1
340 NEXT I
350 GOTO 190
360 REM ПОСТРОЕНИЕ ТОЧКИ ОБРАЗА
370 IF NY=4 AND K>0 THEN GOSUB 2000:
              K=K-1:GOSUB 2000:COLOR=5:
                  GOSUB 2100:GOTO 190
375 IF NY=4 AND K=0 THEN 190
380 IF NY=0 THEN DX=0 :DY=-1
390 IF NY=1 THEN DX=1 :DY=0
400 IF NY=2 THEN DX=0 :DY=1
410 IF NY=3 THEN DX=-1:DY=0
420 X(K+1)=X(K)+7*DX:Y(K+1)=Y(K)+7*DY
430 IF X(K+1)>120 OR X(K+1)<42 THEN 190
440 IF Y(K+1)>120 OR Y(K+1)<8 THEN 190
450 KD(K)=4*NX+NY
460 COLOR=4-NX:GOSUB 2100
470 PLOT X(K)+2+3*DX,Y(K)+2+3*DY
480 IF NX=1 THEN GOSUB 2200
490 K=K+1:COLOR=5:GOSUB 2100
500 GOTO 190
510 REM ПЕРЕКОДИРОВКА
520 IF K=0 THEN 190
```

```
530 K=K-1:NK=0:NB=0
540 B(NB)=KD(NK):NK=NK+1
550 IF NK>K THEN 610
560 B(NB)=B(NB)+8*KD(NK):NK=NK+1
570 IF NK>K THEN 610
580 IF KD(NK)<>1 OR KD(NK)<>2 OR
      KD(NK)<>3 THEN NB=NB+1:GOTO 540
590 B(NB)=B(NB)+64*KD(NK):NK=NK+1
600 IF NK<=K THEN NB=NB+1:GOTO 540
610 NB=NB+1:B(NB)=0
620 POKE 232,0:POKE 233,48
630 POKE 12288,1:POKE 12289,0
640 POKE 12290,4:POKE 12291,0
650 FOR I=0 TO NB
660 POKE 12292+I,B(I)
670 NEXT I
680 HGR=2:COLOR=15
690 SC=1:R=1:XT=127:YT=127:PR=0
700 SCALE=SC
710 ROT=R
720 DRAW 1 AT XT,YT
730 GET S$
735 IF PR=0 THEN COLOR=0:
              DRAW 1 AT XT,YT:COLOR=15
740 IF S$="R" OR S$="P" THEN R=R+1:
                        IF R=64 THEN R=0
750 IF (S$="<" OR S$=",") AND SC>0
                        THEN SC=SC-1
760 IF (S$=">" OR S$=".") AND SC<255
                        THEN SC=SC+1
770 IF S$="B" OR S$="6" THEN PR=0
780 IF S$="S" OR S$="C" THEN PR=1
790 IF ASC(S$)=26 AND YT<255
                        THEN YT=YT+1
800 IF ASC(S$)=21 AND XT<255
                        THEN XT=XT+1
810 IF ASC(S$)=25 AND YT>0
                        THEN YT=YT-1
820 IF ASC(S$)=8 AND XT>0 THEN XT=XT-1
830 IF ASC(S$)=13 GOTO 850
840 GOTO 700
850 REM ВЫДАЧА НА ЭКРАН
860 TEXT=0:HOME:M=9:K=0
870 NC=INT(NB/10)
880 OS=NB-10*NC
890 FOR I=0 TO NC
900 PRINT :PRINT ")_____DATA_";
910 IF I=NC THEN M=OS
920 FOR J=0 TO M
930 PRINT B(K+J);
935 IF J<>M THEN PRINT ",";
940 NEXT J
950 K=K+10:PRINT
960 NEXT I
970 END
1000 PLOT 10, 4 TO 10,16
1010 PLOT  9, 6 TO 11, 6
1020 RETURN
1100 PLOT 30, 4 TO 30,16
1110 PLOT 29, 6 TO 31, 6
1120 PLOT 29,16 TO 31,16
1130 PLOT 28,15 TO 32,15
1140 PLOT 28,14 TO 32,14
1150 PLOT 28,13 TO 32,13
1160 PLOT 29,12 TO 31,12
1170 RETURN
1200 PLOT  4,30 TO 16,30
1210 PLOT 14,29 TO 14,31
1220 RETURN
1300 PLOT 24,30 TO 36,30
1310 PLOT 34,29 TO 34,31
1320 PLOT 25,28 TO 27,28
1330 PLOT 24,29 TO 28,29
1340 PLOT 24,30 TO 28,30
1350 PLOT 25,32 TO 27,32
1370 RETURN
1400 PLOT 10,44 TO 10,56
1410 PLOT  9,54 TO 11,54
1420 RETURN
1500 PLOT 30,44 TO 30,56
1510 PLOT 29,54 TO 31,54
1520 PLOT 29,44 TO 31,44
1530 PLOT 28,45 TO 32,45
1540 PLOT 28,46 TO 32,46
1550 PLOT 28,47 TO 32,47
1560 PLOT 29,48 TO 31,48
1570 RETURN
1600 PLOT  4,70 TO 16,70
1610 PLOT  6,69 TO  6,71
1620 RETURN
1700 PLOT 24,70 TO 36,70
1710 PLOT 26,69 TO 26,71
1720 PLOT 33,68 TO 35,68
1730 PLOT 32,69 TO 36,69
1740 PLOT 32,71 TO 36,71
1750 PLOT 33,72 TO 35,72
1760 RETURN
1800 PLOT  8,84 TO  8,96
1810 PLOT  8,84 TO 11,84
1820 PLOT  8,96 TO 11,96
1830 PLOT 12,85 TO 12,95
1840 PLOT 18,84 TO 18,96
1850 PLOT 18,84 TO 23,84
1860 PLOT 18,96 TO 24,96
1870 PLOT 18,90 TO 21,90
1880 PLOT 28,84 TO 28,96
1890 PLOT 28,96 TO 33,96
1892 PLOT 33,95
1895 RETURN
1900 REM РАМКА
1910 XX=NX*20+2:YY=NY*20+2:L=16:
                  IF NY=4 THEN L=36:XX=2
1920 PLOT XX,YY TO XX+L,YY
    TO XX+L,YY+16 TO XX,YY+16 TO XX,YY
1930 RETURN
2000 REM DEL
2010 COLOR=0
2020 FOR I=-1 TO 3
2030 PLOT X(K)-1,Y(K)+I
                        TO X(K)+3,Y(K)+I
2040 NEXT I
2050 RETURN
2100 PLOT X(K),Y(K) TO X(K)+4,Y(K)
       TO X(K)+4,Y(K)+4 TO X(K),Y(K)+4
                              TO X(K),Y(K)
2110 RETURN
2200 FOR I=1 TO 3
2210 PLOT X(K)+1,Y(K)+I
                        TO X(K)+3,Y(K)+I
2220 NEXT I
2230 RETURN
```

Дадим краткие указания по пользованию программой. Программа имеет стадии. 1. Создание образа. 2. Манипуляция созданным образом. 3. Завершающая стадия.

При создании образа используется меню (рис. 6), в котором изображены все 8 типов клеток. Нижняя клетка соответствует операции удаления из образа последней стрелки. По меню можно перемещать указатель. Нажатие на пробел вызывает установку очередной стрелки или удаление последней стрелки



из образа — в зависимости от положения указателя в меню. Признаком того, что образ создан, является нажатие на клавишу ввода. (При создании образа не используйте стрелку ↑ из меню более одного раза подряд, так как иначе в массиве может появиться нулевой байт, воспринимаемый как завершение описания образа.)

В фазе манипуляции образом используются следующие клавиши:

< — уменьшение образа;

> — увеличение образа;

P R — поворот образа

— перемещение образа

Б В — переход к новому образу после стирания старого (без следа)

C S — переход к новому образу без стирания старого (след)

— переход в завершающую стадию

В завершающей стадии на экран выдаются числа, которые следует загрузить в память ПЭВМ для создания соответствующего образа. После этого программа заканчивает свою работу, при этом в память компьютера уже загружена соответствующая таблица, начиная с адреса ¤ 3000. Благодаря этому можно сразу начать работу с этим образом, предварительно удалив из памяти программу-редактор (NEW) и набрав простую программу работы с образом. В этой программе следует использовать вторую страницу ГВР (HGR=2).

```
10 REM ОТРЕЗОК
20 DATA 1,0,4,0,4,0
30 POKE ¤E8,0:POKE ¤E9,¤10
40 FOR AD=¤1000 TO ¤1005
50 READ B
60 POKE AD,B
70 NEXT AD
80 NEW
```

Подготовительная работа, проводимая перед использованием оператора DRAW, может быть организована либо с помощью программы-редактора, либо на базе выделения сравнительно узкого класса образов.

Приведенную выше программу-редактор следует рассматривать как заготовку, которую преподаватель может переработать, ориентируясь на свои нужды. Например, можно предусмотреть в редакторе средства для создания таблицы, содержащей описание нескольких образов, и загрузки этой таблицы в память компьютера с произвольного адреса. Использование редактора в исходной форме допускает два варианта. Первый, уже описанный, заключается в построении образа и последующем удалении редактора из памяти. Если же необходимо произвести перезагрузку таблицы с другого адреса, то можно удалить редактор, а затем, используя данные, выведенные на экран, набрать строки новой программы, которые позволят перезагрузить таблицу. Наконец (и в том и в другом случае) набирается та часть программы, в которой будет производиться манипуляция созданным образом.

Продолжение следует.

В. ЛИТВИНОВ

Препроцессор интерпретатора языка Бейсик ПЭВМ «Агат»

Написать серьезную программу на Бейсике «Агата» — задача не из легких, если речь не идет о сотне-другой операторов. Решить ее поможет технология программирования, основанная на препроцессировании, модульном программировании и абстрактных типах данных, предлагаемая ниже.

Введение

В дисковой операционной системе ПЭВМ «Агат» имеется команда исполнения из текстового файла EXEC ⟨имя файла⟩. Она позволяет готовить и корректировать программу на Бейсике с помощью любого текстового редактора, например с помощью текстового редактора системы «Школьница», а представление программ в виде текстовых файлов в свою очередь позволяет применить к ним препроцессирование (макрообработку).

Знакомство с препроцессором в языке Си [1, 2] привело меня к мысли написать примерно такой же препроцессор и для интерпретатора языка Бейсик ПЭВМ «Агат». Кроме того, работа с [3] и модульное программирование на Бейсике показали существование еще одного вида программирования в абсолютных адресах — программирования, так сказать, в «абсолютных номерах строк», а также своеобразной фрагментации поля номеров строк программы. Поэтому было решено, что к функциям препроцессора, подобным имеющимся в языке Си, необходимо добавить возможность передачи управления на метки, а также конструкцию ELSE в операторе IF. Разумеется, лучше всего написать препроцессор на ассемблере, однако по причине недостатка времени я остановил свой выбор на языке Рапира. В результате препроцессор был написан за полтора месяца; правда, он работает не очень быстро, но разве плохо в течение тех 2—5 минут, пока идет препроцессирование, оторваться от экрана, размяться? Кроме того, вряд ли ту же работу вручную можно выполнить быстрее. Чем меньше исходная программа, тем быстрее будет получен результат, поэтому необходимо стремиться к модульному программированию [3].

Описание препроцессора

Препроцессором называется программа для внесения различного рода изменений в исходный текст на одном из языков программирования в автоматическом режиме. В отсутствие препроцессора эта работа обычно производится вручную с помощью текстового редактора. Следовательно, препроцессор повышает производительность труда программиста, сокращает число описок при составлении больших программ. Применять же препроцессор для маленьких программ нерационально. Это будет похоже на исполнение чижика-пыжика симфоническим оркестром.

Описываемый препроцессор имеет следующие возможности:

включение в программу текстов из внешних файлов, замену имен текстами, удаление имен (простые макроопределения);

условное препроцессирование;

обработка передач управления на метки и автоматическая нумерация строк;

расширение оператора IF конструкцией ELSE;

собственные комментарии.

Включение из файлов и простые макроопределения. Для каждого массива в программе может потребоваться примерно такая подпрограмма распечатки его элементов:

```
8000 FOR I=1 TO NEL
8010 PRINT I;":";MAS(I)
8020 NEXT
...
```

Если массивов много, то удобно, используя препроцессор, написать простое (для данного случая — даже очень) макроопределение и записать его как файл на диск:

```
...
.&PTARR FOR I=1 TO &NARR
        PRINT I;":";&ARR
        NEXT
```

В нужном месте препроцессор согласно указаниям программиста включит это макроопределение из файла в текст составляемой программы и заменит имена, начинающиеся символом &, на конкретные, т. е. произведет настройку имен.

Условное препроцессирование. В зависимости от значения некоторого признака можно включать в составляемую программу тот или иной фрагмент текста, например ту или иную подпрограмму, описание данных и т. д. Например, для небольшого числа элементов массива лучше применять сортировку методом пузырька, для большого — быструю сортировку. Написав в программе

```
...
#IF МАЛЭЛ
#INC ПУЗЫРЬКОВАЯ
```

```
#ELSE
#INC БЫСТРАЯ
#END
...
```

мы заставим препроцессор включить либо ту, либо другую подпрограмму — в зависимости от признака МАЛЭЛ.

Обработка передач управления на метки и автоматическая нумерация строк. Вот фрагмент программы.

```
...
10220 GOSUB 980
10230 GOSUB 990
10240 GOSUB 1000
10250 GOTO 400
...
```

С расчетом на использование препроцессора этот фрагмент запишется так:

```
...
GOSUB ВЫЧИСЛЕНИЕ_ЧИСЛИТЕЛЯ
GOSUB ВЫЧИСЛЕНИЕ_ЗНАМЕНАТЕЛЯ
GOSUB ВЫЧИСЛЕНИЕ_ВЫРАЖЕНИЯ
GOSUB ПЕЧАТЬ
...
```

Ненамного длиннее и намного понятней. А вообще-то самая краткая запись программы — машинный код.

Расширение оператора IF конструкцией ELSE позволяет увеличить наглядность программы и соблюсти одно из требований структурного программирования к оператору ветвления. Вот примеры.

```
...
00780 IF NUM<HG THEN A=NUM:GOTO 3890
00790 B=NUM:GOTO 5200
...
...
IF NUM<HG THEN A=NUM:GOTO НАЛЕВО
          ELSE B=NUM:GOTO НАПРАВО
...
```

Собственные комментарии. Препроцессор не передает собственные комментарии в готовую программу, тем самым экономя оперативную память. Комментарии существуют только в файлах на диске. В условиях нарастающей важности специфики программных модулей это просто необходимо.

Используя препроцессор, можно специфицировать модуль достаточно подробно; например:

```
**********************************
* Модуль MCOMP вычисляет модуль  *
* комплексного числа.            *
* вход: X - действительная часть, *
*       Y - мнимая часть.        *
* выход:Z - модуль числа.        *
* локальные переменные: нет.     *
**********************************
.MCOMP Z=SQR(X*X+Y*Y)
      RETURN
**********************************
* конец модуля MCOMP             *
**********************************
```

В загрузочный модуль будет вынесено

```
...
1010 Z=SQRT(X*X+Y*Y)
1020 RETURN
...
```

Работа препроцессора

На вход препроцессора поступает текстовый файл (исходный модуль), каждая строка которого обрабатывается определенным образом.

Если первый символ строки *, то это комментарий, и строка игнорируется.

Если первый символ — #, то это команда препроцессора, и она соответствующим образом выполняется.

Если первый символ — точка, то это означает, что идентификатор, следующий без пробела сразу за ней,— метка. Она учитывается, как обычно, в таблице меток вместе с соответствующим номером строки. Остаток строки обрабатывается как строка операторов Бейсика.

Любой другой символ означает, что это строка операторов Бейсика. Строке присваивается текущий номер, разрешаются ссылки на метки (т. е. вместо имен меток проставляются соответствующие им номера строк), конструкция IF заменяется эквивалентным представлением и готовая строка (в случае ELSE — две) на Бейсике выводится в другой текстовый файл. Текущий номер строки увеличивается на текущий шаг. Процесс повторяется до исчерпания исходного модуля. В результате получается текстовый файл с готовой программой (загрузочный модуль) на Бейсике, которую можно загружать и интерпретировать.

Команды препроцессора

Команды начинаются символом #в первой позиции строки. Сразу за этим символом без пробела следует имя команды, а затем через пробелы один или два операнда. При составлении исходных модулей можно указывать только три первых символа имени команды.

Команда INCLUDE.

Формат:

```
#INCLUDE <имя файла>
```

По этой команде обработка строк текущего файла приостанавливается, и начинается обработка строк файла, имя которого указано в операнде. Обработка продолжается до исчерпания этого файла. В этом файле также могут встречаться команды препроцессора. Уровень вложенности INCLUDE не должен превосходить 6.

Если имя файла начинается подчерком (например, __TEST), то это означает, что на входе препроцессора — загрузочный модуль, и он целиком без изменений переносится в выходной файл. Это очень важный

момент: появляется возможность использовать ранее созданные загрузочные модули. Для нумерации новых присоединяемых модулей необходимо прибегать к команде препроцессора NUMBER во избежание пересечения номеров строк.

Команда DEFINE.

Формат:

```
#DEFINE <имя> <фрагмент текста>
```

После этой команды каждое вхождение имени, начинающегося с символа &, из первого операнда, будет заменено фрагментом текста из второго операнда. Фрагмент текста начинается с любого символа, кроме пробела, и продолжается до конца строки. Замена будет производиться во всех последующих строках, содержащих метки и операторы на Бейсике. Замена производится до разрешения ссылок. Вложенные замены недопустимы. В строках комментариев и команд препроцессора замена не производится. Если второй операнд отсутствует, то имя просто удаляется из строки.

Имя определено, если оно встретилось в команде DEFINE и при этом присутствовал второй операнд.

Команда IF.

Формат:

```
#IF <имя>
```

Это команда условного препроцессирования.

Если имя определено, то все последующие строки обрабатываются как обычно до соответствующей команды ELSE или ENDIF. Если имя не определено (а имя можно сделать неопределенным, указав его в команде DEFINE без второго операнда), то все последующие строки до ELSE или ENDIF игнорируются. Возможна вложенность команд IF практически неограниченной глубины.

Команда ELSE.

Формат:

```
#ELSE
```

Необязательна. Может соответствовать некоторой команде IF и является альтернативой ей. Если в искомой IF имя не определено, то обрабатываются строки между ELSE и соответствующей ENDIF. Если же имя в IF определено, то строки между ELSE и ENDIF игнорируются.

Команда ENDIF.

Формат:

```
#ENDIF
```

Обязательно должна соответствовать каждой команде IF. Прекращает действие последней.

Команда NUMBER.

Формат:

```
#NUMBER <номер> <шаг>
```

Служит для установки текущего номера строки и шага нумерации. Оба операнда — числа. Нельзя использовать шаг, равный 1, если в модуле имеются конструкции ELSE. В начале работы препроцессора текущий номер и шаг устанавливаются равными 10.

Пояснения к примеру

Для демонстрации рассмотрим пример программирования на Бейсике с применением средств препроцессора. Представлена программа POSTFIX, преобразующая инфиксное арифметическое выражение в постфиксное [3]. В ней использовался абстрактный тип данных — стек, оформленный в виде отдельного настраиваемого модуля [4].

Модуль настраивается путем изменения фактических имен и констант в файле POSTFIX__STACK__DEF, например, добавлением в начале каждого фактического имени буквы W. Далее этот файл может быть записан на диск с новым именем, допустим: POSTFIX__STACK__WDEF. Чтобы создать еще один стек, в программу необходимо ввести строки

```
...
#INC POSTFIX_STACK_WDEF
#INC STACK_TYP
GOSUB WCREATE
...
.AD 1
```

В дальнейшем при работе с новым стеком необходимо учитывать, что имена его операций начинаются с буквы W.

Отладка программы

Как видим, каждая операция в абстракции стека реализована в виде отдельного модуля и имеет свои входные, выходные и локальные переменные.

Распечатка во время выполнения программы имени модуля, содержимого входных переменных в начале работы модуля и выходных в конце дает достаточно полную информацию о ходе выполнения (хотя ничто не мешает таким же образом печатать и локальные переменные). Разумеется, сказанное касается и всех других модулей, а не только реализующих стек, например модуля POSTFIX__PRCD.

Препроцессор позволяет легко вставлять и удалять отладочные операторы.

Рассмотрим модуль STACK__PUSH, реализующий операцию размещения в стеке. Расположим после метки .&PUSH следующие строки:

```
#IF TRACE
  PRINT "***&PUSH***"
  PRINT "&TP=";&TP;" MAXSTACK=";
                    &MAXSTACK;" &X=";&X
#ENDIF
```

а перед строкой с RETURN

```
#IF TRACE
  GOSUB &PTSTK
  PRINT "&TP=";&TP;" OVER=";OVER
#ENDIF
```

Тогда, разместив в начале программы POSTFIX строку

```
#DEFINE TRACE YES                    (1)
```

мы тем самым определим TRACE и заставим препроцессор сгенерировать операторы отладочной печати в нужном месте. Чтобы отключить печать, нужно или удалить строку (1), или убрать из нее YES.

Можно ввести признак отладки для каждого модуля в отдельности и свести эти признаки в файл отладки, который будет иметь примерно такой вид:

```
* ФАЙЛ POSTFIX_TRACE_DEF
#DEF TRACE_EMPTY
#DEF TRACE_PUSH YES
      ...
#DEF TRACE_PRCD YES
```

В основной же программе можно включать и отключать отладку с помощью команды

```
#INC POSTFIX_TRACE_DEF
```

Манипулируя операндами в командах DEFINE файла POSTFIX__TRACE__DEF, можно выбирать модули, подлежащие трассировке.

Заключение

Весьма важный совет: если вы используете в работе препроцессор, будьте последовательны. Не торопитесь. Не вносите исправлений в загрузочный модуль. Это еще может себя оправдать, если модулей несколько штук. А если их многие десятки? Тогда мелкий выигрыш во времени отладки одного модуля может обернуться крупным проигрышем при их взаимоувязке.

Для тех, кто заинтересовался описанной темой, у кого появились вопросы, дополнения,— мой адрес: 307239, Курская обл., г. Курчатов, 6-й микрорайон, ул. Садовая, 4, кв. 67. Литвинову Виктору Ивановичу.

ЛИТЕРАТУРА

1. *Хендрикс Д.* Композитор языка Си для микроЭВМ. М.: Радио и связь, 1989.

2. *Дансмур М., Дейвис Г.* Операционная система UNIX и программирование на языке Си. М.: Радио и связь, 1989.

3. *Лэнгсам Й., Огенстайн М., Тенебайм А.* Структура данных для персональных ЭВМ. М.: Мир. 1989.

4. *Лисков Б., Гатэг Дж.* Использование абстракций и спецификаций при разработке программ. М.: Мир. 1989.

А. ГУСЕВ

Операционная система ФОДОС

3.3. Просмотр содержимого файла на экране дисплея

Для просмотра содержимого файла на экране дисплея используется команда ***TYPE (TY)***. Имеет смысл просматривать файлы, содержащие текстовую информацию. Не имеет смысла смотреть файл *BASIC.SAV*, так как это набор машинных инструкций. Что из себя представляет файл, можно узнать из его расширения:

SAV Файл отображения памяти, загрузочный модуль, являющийся программой в машинных кодах, может быть запущен на выполнение. Например, *BASIC.SAV* — компилятор языка Бейсик. Запускается из ОС командой *RUN* (см. далее).

BIN Файл, содержащий программу в машинных кодах, или файл отображения памяти. Используется компилятором языка *BASIC*.

OBJ Объектный модуль, библиотеки подпрограмм. Используются программой *LINK* и *LIBR*. Файл содержит машинные коды.

SYS Системный файл. Драйверы ОС. Файл содержит машинные коды.

BAD Файл, содержащий плохие блоки. Если на диске имеются плохие (испорченные) блоки, то ОС считает, что там записан файл *FILE.BAD*. Таким образом исключается возможность записи файлов на плохой участок диска. При появлении на диске плохих блоков его следует отформатировать.

BAK Резервный файл. При работе с редактором текста старый файл не стирается с диска, а сохраняется и ему присваивается расширение *BAK*. Файл содержит текстовую информацию.

COM Командный файл. Часто используемые команды ОС пользователь может записать в файл с расширением *COM* и запустить его на выполнение. При этом ОС будет считывать команды из этого файла, как будто их набирает пользователь с клавиатуры. Пример: ***BASIC.COM***. Загружает компилятор языка Бейсик на рабочие места учеников. Файл содержит текстовую информацию.

DAT Файл данных. Результат работы программ может быть записан на диск. Как правило, содержит текстовую информацию.

DOC, DPR, HLP, LST, WRT — файлы с этими расширениями содержат описания, документацию, справочную информацию. Расширение *LST* указывает на файл, содержащий листинг (информацию) о результатах работы трансляторов. Текстовой редактор *EDIC* записывает результаты своей работы в файл с расширением *LST*. Текстовой редактор *WRITER* создает файл с расширением *WRT*. Все файлы текстовые.

FOR Файл, содержащий текст программы на языке Фортран. Содержит текстовую информацию.

PAS Файл, содержащий текст программы на языке Паскаль. Содержит текстовую информацию.

MAC Файл, содержащий текст программы на языке Макроассемблер. Содержит текстовую информацию.

Если в команде *TYPE* указать только имя файла, без расширения, то ОС будет искать файл с указанным именем и с расширением *LST*. Пример использования команды *TYPE*:

```
MZØ>TY INSTR
```

Распечатать на экране терминала содержимое файла *INSTR.LST*.

Если требуется распечатать файл с расширением, отличным от *LST*, то надо указать имя файла с его расширением. Например:

```
MZØ>TY BASIC.COM
```

Распечатать на экране содержимое файла *BASIC.COM*.

```
MZØ>TY BASIC.SAV
```

Команда выводит на дисплей содержимое файла *BASIC.SAV*. При этом экран заполнится бессмысленным набором символов, так

Продолжение. Начало см.: ИНФО. 1992. № 2

как в файле *BASIC.SAV* содержатся машинные коды. Давать такую команду не следует, так как в результате ее выполнения дисплей может перейти в другой режим работы, что потребует перезагрузить ЭВМ.

В команде *TYPE*, в именах файлов можно использовать знак групповой операции. Например:

```
MZØ>TY INS*
```

Распечатать на экране все файлы, имеющие расширение *LST*, и с именами, начинающимися на *INS*. При этом вывод на экран будет непрерывным.

Следующая команда выводит на экран все файлы с именем *BASIC*:

```
MZØ>TY BASIC.*
```

Распечатать на экране все файлы с именем *BASIC* и любым расширением. Однако вывод на экран содержимого файла *BASIC.SAV* может привести к плохим последствиям (см. выше). Чтобы исключить данный (данные) файл из операции, следует применять ключи. Далее приводятся ключи команды *TYPE*:

1. /NOQUERY (/NOQ) — вывод на экран без запроса. Можно не указывать, так как ОС принимает этот ключ по умолчанию.

2. /QUERY (/Q) — печать с запросом. Перед печатью каждого файла ОС запрашивает разрешение на включение его в операцию. Например:

```
MZØ>TY/Q BASIC.*
```

Распечатать на экране все файлы с именем *BASIC* и любым расширением, перед печатью каждого файла запрашивать о включении его в операцию. Ответ пользователь вводит, как и при копировании: *Y* — печатать файл, *N* — не печатать (*N* можно не набирать, а сразу нажимать на ВВОД). Результат работы будет следующий:

```
        Files copied
DK:BASIC.COM    to TT:   ?Y
DK:BASIC.SAV    to TT:   ?N
DK:BASIC.BIN    to TT:   ?
DK:BASIC.HLP    to TT:   ?Y
```

Запросы ОС аналогичны запросам при работе с командой *COPY*. Это связано с тем, что вывод файла на экран и копирование файла на (*tt*) устройство *TT*: (дисплей, см. выше) — одно и то же действие. Итак, файлы *BASIC.COM* и *BASIC.HLP* будут выведены на экран, а файлы *BASIC.SAV* и *BASIC.BIN* нет. Информация о *DK*: появилась в результате того, что в команде *TYPE* не было указано устройство, а только имя файла. Если указать *MZ:BASIC.**, то вместо *DK*: будет стоять *MZ*:.

3. /COPIES:n (/COP:n) — распечатать *n* копий. Команда выводит 4 раза подряд содержимое файла *TEXT.LST*, находящегося на диске *MZ*1:

```
MZØ>TY/COP:4 MZ1:TEXT
```

4. /DELETE (/DEL) — после вывода на экран содержимого файл удаляется с диска. Удаление производится без запроса.

5. /NEWFILE (/NEW) — включить в операцию файлы, имеющие текущую дату создания.

3.4. Вывод содержимого файла на печатающее устройство

Для вывода содержимого файла на принтер применяется команда ***PRINT (PRI)***. В команде *PRINT* используются все ключи команды *TYPE*. Действие команды *PRINT* аналогично копированию файлов на устройство *LP*: (см. выше).

3.5. Переименование файлов

Для изменения имени файла используется команда ***RENAME (REN)***. Данная команда изменяет имя файла, не изменяя его содержимого. Формат команды:

MZO>REN [/ключ] ⟨старое имя⟩ ⟨новое имя⟩

Например:

```
MZØ>REN TED RED
```

файлу *TED* дать новое имя *RED*.

```
MZØ>REN MZ1:*.LST MZ1:*.HLP
```

Всем файлам, хранящимся на диске *MZ*1: и имеющим расширение *LST*, придать расширение *HLP*.

При изменении имени файла можно изменить и дату его создания. Для этого применяется ключ */SETDAT (/SETD)*. Например:

```
MZØ>REN/SETD GRT GGG
```

переименовать файл *GRT* в *GGG* и установить текущую дату создания.

3.6. Удаление файлов

Для удаления файлов используется команда ***DELETE (DEL)***. Удалять можно только незащищенные файлы. Одновременно можно указывать несколько имен файлов, но не более 6. Удаление производится с запросом, т. е. ключ */QUERY* можно не указывать. Например:

```
MZØ>DEL TEST

        Files deleted
DK:TEST.SAV       ?Y
```

```
DK:TEST.BIN       ?N
DK:TEST.LST       ?
DK:TEST.ASC       ?Y
```

Будут удалены файлы: *TEST.SAV* и *TEST.ASC*.

При попытке удалить защищенный файл ОС выдаст сообщение о том, что операцию выполнить невозможно. Например, если файл *TEST.SAV* защищен, то результат выполнения предыдущей команды был бы следующим:

```
MZØ>DEL TEST

?PIP-W-Protected file DK:TEST.SAV
        Files deleted
DK:TEST.BIN       ?N
DK:TEST.LST       ?
DK:TEST.ASC       ?Y
```

Файл *TEST.SAV* защищен от удаления и в операцию не включается.

С командой *DELETE* можно использовать ключи:

1. */SYSTEM* (*/SYS*) — включить в операцию системные файлы.
2. */NOQUERY* (*/NOQ*) — удаление без запроса. Файлы, указанные в команде, удаляются без запроса. Защищенные файлы в операцию не включаются.

3.7. Защита файлов

Для присвоения файлу атрибута защищен от удаления применяется команда *PROTECT* (*PRO*). Например:

```
MZØ>PRO MZ1:RUS.ASC
```

защитить файл *RUS.ASC*, находящийся на устройстве *MZ1:*, от удаления.

Команда *DIR* после размера файла теперь будет выдавать символ *P*.

```
MZØ>DIR MZ1:RUS.ASC

  25-Dec-91
RUS  .ASC  5P   Ø2-Sep-86
   1 Files,     5 Blocks
  518 Free blocks
```

Для снятия атрибута защищен от удаления используется команда *UNPROTECT* (*UNPRO*). Например, снять защиту с файла *RUS.ASC*

```
MZØ>UNPRO MZ1:RUS.ASC
```

Теперь команда *DIR* после размера файла не будет выдавать символ *P*.

```
MZØ>DIR MZ1:RUS.ASC

  25-Dec-91
RUS  .ASC  5   Ø2-Sep-86
   1 Files,     5 Blocks
  518 Free blocks
```

4. Запуск файлов на выполнение

4.1. Запуск программных файлов

Файл, содержащий машинные коды, можно загрузить в память машины и запустить на выполнение. Для этого используется команда *RUN* (*RU* или *R*): Например:

```
MZØ>RUN MZ1:TEST
```

загрузить с устройства *MZ1:* и запустить на выполнение файл *TEST.SAV*.

Если в команде *RUN* не указать расширение в имени файла — ОС будет искать файл с расширением *SAV*, имя устройства — ОС будет искать файл на устройстве по умолчанию.

Для запуска на выполнение файла, находящегося на системном диске и содержащего программу, набрать только его имя.

После того как пользователь набрал строку и нажал на клавишу ⟨ВВОД⟩, ОС начинает ее анализировать. Если это команда, то она выполняется, если нет, то ОС ищет на системном диске файл с набранным именем и расширением *SAV*. Затем этот файл загружается и запускается на выполнение.

Примеры использования команды *RUN*:

```
MZØ>RUN MZ1:TIR
```

запустить на выполнение программный файл *TIR.SAV*, находящийся на диске *MZ1:*.

```
MZØ>RUN PF
```

запустить на выполнение программный файл *PF.SAV*, находящийся на устройстве по умолчанию *DK:* (в данном случае с *MZØ:*).

```
MZØ>RUN SS.BIN
```

запустить на выполнение программный файл *SS.BIN*.

```
MZØ>RUN TEST.LST
```

запустить на выполнение файл *TEST.LST*. Так как это текстовой файл, то операция выполняться не будет.

```
MZØ>R TEST
```

запустить на выполнение программный файл *TEST*, находящийся на системном диске.

```
MZØ>BASIC
```

запустить на выполнение программный файл *BASIC.SAV*. ОС проанализирует набранную строку и определит, что *BASIC* не является командой и будет пытаться запустить на выполнение файл *BASIC.SAV*.

```
MZØ>RUN REN
```

запустить на выполнение программный файл *REN.SAV*.

4.2. Запуск командного файла

Для загрузки компилятора языка Бейсик на РМУ (рабочие места учеников) надо набрать порядка 10 строк команд ОС, что очень неудобно для пользователя. В процессе набора могут возникнуть ошибки. Поэтому необходимые строки записываются в виде от-

дельного файла *BASIC.COM* (расширение *COM* указывает на командный файл). ОС считывает из файла строки и выполняет их. Для запуска командного файла нужно набрать символ *a* (клавиша Ю), @ затем его имя. Например:

```
MZØ>@BASIC
```

запустить на выполнение командный файл *BASIC.COM*.

Командные файлы создаются с помощью текстового редактора. На системном диске находится командный файл *STARTS.COM*. Он запускается при загрузке ОС и содержит команды, необходимые для настройки параметров ОС. Из файла *STARTS.COM* запускается программа установки даты — *DAY.SAV*.

5. Установка параметров системы

Команда ***SET*** изменяет характеристики драйверов и некоторые системные параметры. Например, если при загрузке операционной системы не выполнить команду *SET TT SCOPE*, то клавиша ⟨ЗАБОЙ⟩ не будет стирать исправляемый символ, а будет отображать его на экране. Драйвер командной строки, позволяющий восстанавливать набранные команды нажатием на клавишу ⟨СТРЕЛКА ВВЕРХ⟩, тоже включается командой *SET*. Для установки параметров следует набрать команду и нажать на ⟨ВВОД⟩. Например:

```
MZØ>SET TT SCOPE
```

Далее приводятся варианты использования команды *SET*

SET TT SCOPE — разрешение использования клавиши ⟨ЗАБОЙ⟩. Эту команду следует включать в стартовый командный файл.

SET SL ON — включение драйвера командной строки *SL*.

SET SL OFF — выключение драйвера *SL*.

SET SL LET — включение режима установки командных строк для драйвера *SL*.

SET RD ON — включение драйвера «квазидиска» *KD*. При этом теряется возможность использования цветов и работы с графическим редактором, текстовым редактором *WRITER*, компилятором языка Бейсик.

SET RD OFF — отключение «квазидиска». Вся информация на нем теряется. После этой команды можно работать с цветами.

SET RD INIT — инициализация электронного диска (без использования стандартных средств ОС).

SET USR NOSWAP — запрещение свопинга. После выполнения этой команды операции с файлами производятся быстрее, но уменьшается доступный объем ОЗУ ЦП.

SET USR SWAP — разрешение свопинга.

SET MZ NTRK=40 — установка сорока дорожек на диске.

SET MZ NTRK=81 — восемьдесят одна дорожка на диске.

SET — параметры драйверов можно узнать с помощью программы *HANDLE*. Для этого надо набрать эту команду и через пробел имя драйвера. Например:

```
MZØ>HANDLE SL
```

Вывести информацию о драйвере *SL*.

6. Создание рабочей и системной дискет

6.1. Создание рабочей дискеты

Как правило, в комплекте с УКНЦ поставляются 20 дискет, на некоторых из них записан минимум необходимого программного обеспечения.

Остальные — чистые. Если попытаться прочитать каталог такой дискеты командой *DIR*, то на экран будет выведено сообщение:

```
Error reading directory
        ( ошибка чтения каталога )
```

Дискета не готова к работе, так как она еще не размечена. Для записи информации ее надо **отформатировать**, т. е. «разбить» на дорожки и сектора.

Форматирует диск программа ***TESTMZ***. При форматировании вся информация, хранящаяся на диске, будет уничтожена. Для начала форматирования следует набрать:

```
MZØ>TESTMZ
```

Если на системном диске файла *TESTMZ* нет, то его следует скопировать или запустить с другого диска (см. команду *RUN*). После запуска программы форматирования (в целях предосторожности) следует поставить диск, который нужно форматировать, и удалить из дисковода все другие. На экране появится меню №1:

```
Т Е С Т      Н Г М Д

---> УСТАНОВКА ПАРАМЕТРОВ
ФОРМАТИРОВАНИЕ
ЗАПИСЬ
ЧТЕНИЕ
ВЫБОР УСТРОЙСТВА
ФУНКЦИОНАЛЬНЫЙ ТЕСТ
```

Клавишей управления курсора ⟨**СТРЕЛКА ВНИЗ**⟩ установите указатель → на строку форматирования и нажмите на ⟨**ВВОД**⟩. На экране появится меню №2:

```
     ФОРМАТИРОВАНИЕ

УСТРОЙСТВО (Ø,1,2,3) Ø :
```

Наберите номер кармана дисковода, в котором установлен диск и нажмите на ВВОД. Для более надежной работы дискету следует форматировать в том кармане, где он будет использоваться в дальнейшем. (Как правило, системный диск — в нулевом кармане, а рабочий — в первом.) После выбора

кармана появится запрос на параметры форматирования:

```
РАЗМЕТКА СТАНДАРТНАЯ ? (Д,Н)
```

Ввод символа Д (да) обозначает форматирование по стандартной разметке. Если ответить Н (нет), то программа будет запрашивать параметры форматирования. На каждый запрос следует набирать нужный параметр и нажимать на клавишу ⟨ВВОД⟩. Просто нажатие на клавишу ⟨ВВОД⟩ сохраняет старые параметры.

Рассмотрим форматирование со стандартной разметкой (ответ Д). Получив ответ, программа начинает форматировать диск. При этом на экран выводятся номера форматируемых дорожек. По завершении форматирования происходит чтение с указанием номеров дорожек. Если форматирование закончилось без ошибок, то выдается запрос на продолжение:

```
ПОВТОРИТЬ ПРИ ТЕХ ЖЕ ПАРАМЕТРАХ (Д,Н,К)
```


Ответ Д означает повторное форматирование (надо форматировать несколько дискет, или операция форматирования завершилась с ошибкой), ответ Н — изменение параметров для нового форматирования, ответ К — закончить работу и перейти в меню # 1.

В случае обнаружения ошибок при форматировании следует повторить команду (не вынимая дискеты, ответить Д). При повторном форматировании могут быть устранены ошибки, и одновременно оно позволяет уменьшить вероятность появления в дальнейшем плохих блоков. Однако если при 2-м или 3-м проходе ошибки остаются, то дискету применять не следует. При обнаружении ошибок программа информирует об этом сообщением:

```
ОШИБКА СТОРОНА : ВЕРХ
или
ОШИБКА СТОРОНА : НИЗ
```

Завершив форматирование, следует ввести ответ К, перейти в меню # 1, нажать на клавишу М, а затем на ⟨ВВОД⟩. На экране появится приглашение ОС.

После форматирования диск готов к работе частично. Он физически размечен, но каталог еще не создан. При попытке прочитать каталог диска командой *DIR* на экран будет выведено сообщение:

```
?-DIR-F-Invalid directory
                (неверный каталог)
```

Для создания каталога диск надо инициализировать командой ***INITIALIZE (INIT)***. Например:

```
MZ0>INIT MZ1:
```

инициализировать диск *MZ1:*.

При этом ОС выдаст запрос на подтверждение операции:

```
MZ1: / Initialize; Are you sure ?
```

(вы согласны инициализировать *MZ1:*).

Необходимо ответить *Y*. При любом другом ответе команда выполняться не будет.

Инициализировать можно не только новый диск, но и уже бывший в употреблении. При инициализации уничтожается информация о всех файлах, находившихся на дискете без использования команды *DELETED*. Однако ОС страхует пользователя от случайной инициализации диска. Если на диске находились защищенные от удаления файлы, то ОС задаст вопрос:

```
Volume contains protected files;
                    Are you sure ?
```

(на диске имеются защищенные файлы).

Необходимо ответить *Y*. При любом другом ответе команда выполняться не будет.

После инициализации дискета полностью готова к работе. Команда *INITIALIZE* позволяет использовать следующие ключи:

1. ***/BADBLOCKS(/BAD)*** — проверить диск на наличие плохих блоков и записать на их место файлы *FILE.BAD*.

2. ***/RESTORE(/RES)*** — восстановить каталог диска, для которого была выполнена команда *INITIALIZE*.

3. ***/VOLUMEID(/VOL)*** — определить идентификатор диска (имя диска) и имя владельца. ОС потребует ввести имя диска (*volume ID:*), а затем имя пользователя (*Owner:*). Имена набираются латинскими буквами. Например:

```
MZ0>INIT/VOL MZ1:
MZ1: / Initialize; Are you sure ? Y
Volume ID: disk-01
Owner    : Ivanow
```

Присвоить диску имя *disk*—01. Пользователь *Ivanow*.

6.2. Создание системной дискеты

Системный диск отличается от рабочего наличием файла монитора и системного загрузчика. Для создания системной дискеты надо сначала подготовить рабочий диск, т. е. отформатировать дискету и проинициализировать ее. На готовый диск скопировать файлы монитора, драйверов и системных программ с имеющегося системного диска. Системный диск должен обязательно содержать файлы:

монитора *RT11CD.SYS* (или другого);

драйверов: *MZ.SYS, TT.SYS, LP.SYS, MC.SYS*;

свопинга *SWAP.SYS*.

На системном диске обязательно долж-

ны находиться системные программы «утилиты»:

DIR.SAV (программа распечатки каталога);

PIP.SAV (программа копирования);

DUP.SAV (программа обслуживания внешних устройств).

Однако даже после этого копирования диск еще не является системным, так как на нем нет начального загрузчика. Для копирования начального загрузчика необходимо использовать команду *COPY* с ключом *BOOT*.

Далее приводится пример создания системного диска на устройстве *MZ1:*. Для этого необходимо, чтобы диск с ОС был установлен в верхнем кармане дисковода. При создании системного диска на *MZ0:* необходимо загрузиться с нижнего кармана дисковода, а в командах устройство *MZ0:* поменять на *MZ1:*, и наоборот.

1. Отформатировать и проинициализировать диск.

2. Выполнить последовательность команд:

```
MZØ>COP MZØ:*.SYS MZ1:
```

Копируем системные файлы.

```
MZØ>COP MZØ:(DIR,PIP,DUP,
                STARTS,DAY) MZ1:
```

Копируем системные программы, стартовый командный файл и программу установки даты. Для создания более хорошей среды ОС следует еще скопировать необходимые программы (см. разделы описания программного обеспечения).

```
MZØ>COP/BOOT MZ1:RT11CD MZ1:
```

Копируем загрузчик ОС *RT11CD* на диск *MZ1:*.

После этого диск считается системным. С него можно загрузить ОС.

7. Локальная сеть УКНЦ

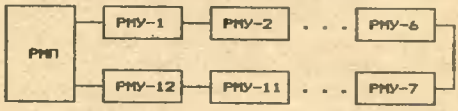
7.1. Структура локальной сети

В состав КУВТ УКНЦ входят «Электроника МС 0511.02» (РМП) и «Электроника МС 0511.01» (РМУ), объединенные в кольцевую локальную сеть со скоростью передачи 57 Кбод.

Использование кольцевой локальной сети имеет преимущества по сравнению с радиальной (используется в КУВТ-86), так как подключение новых РМУ не представляет затруднений. Достаточно разорвать в удобном месте кольцо, впаять дополнительную соединительную колодку и подключить ее к адаптеру, имеющемуся на каждом компьютере.

К преимуществам кольцевой сети относятся также: простая и недорогая связь (физическая) между устройствами, возможность передачи данных всем РМУ одновременно, простой контроль ошибок при передаче, гарантированное время передачи для каждого устройства. Однако кольцевая сеть имеет и недостатки: надежность ее функционирования зависит от надежности каждого РМУ (в каждом РМУ происходит ретрансляция данных), довольно сложна процедура начальной загрузки кольца и др.

Схематично ЛВС можно представить так:



Каждая станция имеет номер (РМУ-1 — номер 1 и т. д.). РМП, как правило, имеет максимальный номер — 63. Чем больше станций в сети, тем длиннее ожидание при обращении к ней. Стандартное программное обеспечение работает с количеством РМУ не более 16.

Номер ЭВМ в ЛВС устанавливается переключателями, находящимися на адаптере ЛВС (для доступа к адаптеру ЛВС надо открыть крышку внизу ЭВМ). Установку номера надо производить при выключенном питании ЭВМ, так как номер станции считывается один раз, при включении питания.

Номер устанавливается в двоичной системе счисления. Направление по стрелке — установка соответствующего бита в единицу, обратное направление — установка бита в ноль. Всего имеется шесть переключателей (максимальное число в двоичной системе счисления — 111111, в восьмеричной — 77, в десятичной — 63). Переключатель, соответствующий младшему биту, выделен красным цветом. Например, для установки номера станции 9 (в двоичной системе счисления — 1001) надо установить переключатели в положение:

7.2. Загрузка на РМУ Бейсик-системы

Бейсик-система УКНЦ предназначена для работы учащихся на РМУ с языком высокого уровня — Бейсик. Бейсик-система позволяет:

загружать транслятор языка Бейсик на РМУ;

составлять программы, запускать их на выполнение, редактировать созданные программы;

записывать программы на диск;

считывать программы с диска;

выводить листинг программы и данные на печатающее устройство и дисплей РМП.

Бейсик-система поставляется вместе с классом УКНЦ на дисках. Однако использовать ее в заводском варианте неудобно. В Московском городском институте усовершенствования учителей произведена необходимая доработка пакета. Бейсик-система состоит из следующих частей:

дискового транслятора языка Бейсик — *BASIC.SAV*. Находится на системном диске;

транслятора языка Бейсик для РМУ — *BASIC.BIN*. Находится на рабочем диске;

загрузчика системы на РМУ — *BOOT6.BIN*. Находится на системном диске;

программы загрузки системы на РМУ — *FTS9.SAV*. Находится на системном диске;

программы связи класса — *NETUK.SAV*. Предназначена для передачи программ на языке Бейсик между РМП и РМУ. Находится на системном диске;

драйвера локальной сети — *MC.SVS*. Находится на системном диске;

командного файла для начала загрузки Бейсик-системы на РМУ — *BASIC.COM*. Должен присутствовать на системном и рабочем дисках;

командного файла для запуска программы связи без предварительной загрузки Бейсик-системы — *NET.COM*. Должен присутствовать на системном и рабочем дисках.

Внимание! Данная система требует наличия системного диска в верхнем кармане дисковода, а рабочего — в нижнем. Номер центральной машины (РМП) должен быть максимальным!!

Данная расстановка файлов позволяет использовать один системный диск и несколько рабочих. Программы, созданные учениками, записываются на рабочий диск. Для загрузки Бейсик-системы необходимо:

1. Включить необходимые ЭВМ.

2. Загрузить ЭВМ с системной дискеты, установленной в верхнем кармане, и в ответ на приглашение *MZ0⟩* набрать *©BASIC:*

```
MZØ>©BASIC
```

На экран будут выводиться строки командного файла. После появления строки:
Файл для загрузки в РМУ⟩*BASIC*
следует нажать на ⟨ВВОД⟩ и дождаться появления строки:
Команда ⟨Ж, З, Ф, К⟩
Следует набрать символ Ж (ждать) и нажать на ⟨ВВОД⟩. На экран будут выводиться номера РМУ, ожидающие загрузки. Номера выводятся в десятичной системе счисления:
РМУ ждущие загрузки
01 02 04 05 08 12
На РМУ при этом курсор со строки ЗАГРУЗКА ИЗ СЕТИ переходит в начало следующей. В процессе загрузки может появиться сообщение: ОШИБКА УРОВНЯ ПЕРЕДАЧИ
В этом случае проверить надежность соединения кабеля локальной сети и разъема ЭВМ. Если это не поможет, то следует выключить все РМУ и затем включать их по одной. Если при включении очередной машины неоднократно появится сообщение ошибка уровня передачи, то следует отказаться от использования данной ЭВМ. В процессе такой проверки допускается однократное появление сообщения об ошибке.

После получения всех необходимых запросов нажмите на ⟨ВВОД⟩ и на запрос
КОМАНДА ⟨Ж, З, Ф, К⟩
следует набрать символ З (загрузка) и нажать на ⟨ВВОД⟩. Начнется загрузка Бейсик-системы. При этом на РМУ будут выводиться номера передаваемых блоков. После окончания загрузки на РМП будет выдано сообщение:
РМУ ЗАГРУЖЕНЫ
Нажать на клавишу ⟨ВВОД⟩ и после появления строки
КОМАНДА ⟨Ж, З, Ф, К⟩
набрать символ К (конец) и нажать на ⟨ВВОД⟩. После этого загрузится и запустится на выполнение программа связи класса — *NETUK*.
При этом на экране появится меню:

Программа связи класса УКНЦ (1986.12.14)
Продолжение
Подтверждение запросов на запись
Подтверждение запросов на чтение
Запрещение записи
Запрещение чтения
Блокировка сообщений
Сброс
Загрузка Бейсик-системы
Конец

Для перехода в режим связи требуется выбрать строку «Продолжение» (она выбрана по умолчанию — в ней стоит курсор) и нажать на ⟨ВВОД⟩. После этого на экране появится строка:

Программа связи класса УКНЦ (1986.12.14)

Теперь с РМУ можно производить обмен с внешними устройствами: загружать с диска или записывать на диск программы на Бейсике, выводить данные и программы на печатающее устройство и экран РМП. При обращении к внешним устройствам на экран РМП будут выводиться сообщения. Для выхода в меню надо нажать на клавишу ⟨ВВОД⟩. Далее рассматриваются остальные строки меню:

Подтверждение запросов на запись

После выбора этой строки и нажатия на клавишу ⟨ВВОД⟩ появится следующее меню:

Программа связи класса УКНЦ (1986.12.14)

Подтверждение запросов на запись

Для всех каналов

Выход

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

17

Сброс

Для запрета записи на внешние устройства РМУ подвести курсор к нужному номеру и нажать на ⟨ВВОД⟩. При этом в строке появится символ: *. Если подвести курсор к строке «Для всех каналов» и нажать на клавишу ⟨ВВОД⟩, то будет установлено запрещение записи для всех ЭВМ.

Строка «Сброс» отменяет установленные параметры. Для выхода в корневое меню следует подвести курсор к строке «Выход» и нажать на клавишу ⟨ВВОД⟩. Теперь, при попытке записи, на РМП появится сообщение и запрос на продолжение. При нажатии на клавишу *Y* (*Yes*) запись будет произведена, а на клавишу *N* (*No*) — нет. На РМУ, выдавшем запрос, появится сообщение об ошибке.

Подтверждение запросов на чтение

Подтверждение запросов на чтение реализуется аналогично режиму «Подтверждение запросов на запись». Одновременно можно указать подтверждение на чтение и запись.

Запрещение записи

Запрет записи без подтверждения. Любая запись запрещена.

Запрещение чтения

Запрет чтения без подтверждения. Любое чтение запрещено.

Блокировка сообщений

Сообщения на экран не выводятся.

Сброс

Сброс всех установленных режимов.

Загрузка Бейсик-системы

При выборе этой строки меню будет производиться загрузка Бейсик-системы на РМУ, т. е. автоматически запустится командный файл *BASIC.COM* (все повторится).

Конец

Выход в ОС.

После выхода в ОС можно войти в систему связи, минуя загрузку Бейсик-системы. Для этого надо запустить на выполнение командный файл *NET.COM:*

```
MZØ>@NET
```

Для загрузки Бейсик-системы на рабочее место преподавателя следует использовать командный файл *BAS.COM:*

```
MZØ>@BAS
```

Транслятор языка Бейсик загружается на РМП. Программы в процессе работы будут считываться с рабочего диска (*MZ1:*).

7.3. Работа с внешними устройствами в Бейсике

7.3.1. Вывод результатов работы и листинга программ на принтер и дисплей РМП

Листинг программы, находящийся на РМУ, можно вывести на печатающее устройство. Для этого на РМУ требуется ввести команду:

```
SAVE "LP:"
```

Записать программу на устройство *LP:*, а так как *LP:* — это принтер, то текст программы будет выведен на печать. При этом принтер должен быть подключен к ЭВМ преподавателя и включен. Если у вас печатающее устройство D100M или СМ6337, то перед загрузкой Бейсик-системы следует запустить на выполнение программу D100M:

```
MZØ>D1ØØM
```

Для вывода листинга программы на дисплей РМП на РМУ следует ввести команду:

```
SAVE "TT:"
```

Для вывода результатов работы на печатающее устройство с РМУ в ученические программы следует включать фрагменты типа:

```
5Ø OPEN "LP:" FOR OUTPUT
6Ø PRINT# "Эта строка будет
      вывыдена на принтер"
7Ø CLOSE
```

7.3.2. Запись/чтение программы с диска

Для записи программ на магнитный диск следует использовать команды: *SAVE* или *CSAVE*. Например:

```
SAVE "GAME"
```

Записать программу на диск под именем *GAME*. При этом файлу присвоится расширение *ASC*. Файл *GAME.ASC* является текстовым. В нем находятся строки программы.

```
CSAVE "GAME"
```

Записать программу на диск под именем *GAME*. При этом файлу присвоится расширение *COD*. Файл *GAME.COD* содержит коды операторов.

Для чтения программ с диска исполь-

зуются команды: *LOAD* или *CLOAD*. Например:

```
LOAD "GAME"
```

Загрузить программу *CAME.ASC*.

```
CLOAD "GAME"
```

Загрузить программу *GAME.COD*.

Команды *CLOAD* и *CSAVE* выполняются быстрее, чем *LOAD* и *SAVE*. Рекомендуется загружать программы командой *CLOAD*, а записывать двумя: *CSAVE* и *SAVE*. Таким образом, на диске хранятся две копии файла, и если он не прочитается командой *CLOAD*, то можно загрузить страховочную копию командой *LOAD*.

7.4. Сетевой монитор

Сетевой монитор *KLassNET* позволяет запускать на РМУ программы, работающие под управлением ОС. Он состоит из следующих частей: *KLNET.SAV*, *NETACT.SAV*, *NETSLV.SAV*, *READON.LYF*, *NODISP.LAY*.

Для запуска сетевого монитора требуется запустить на выполнение программный файл *KLNET:*

```
MZØ>KLNET
```

После этого программа выдаст запросы на установку режимов работы:

Укажите количество станций (не более 15):

> — Следует набрать число РМУ и нажать на клавишу ⟨ВВОД⟩.

Определите количество блоков непрерывного обмена:

> — Оптимальное число 8.

Введите спецификацию файла задачи:

> — Следует нажать на клавишу ⟨ВВОД⟩.

После этого на экране появится таблица, отражающая ход работы. Затем надо по очереди включать РМУ. Перед включением очередной машины дождитесь появления таблицы на предыдущей. Сетевой монитор *KLassNET* можно запустить с помощью командного файла *K.COM*. Для этого надо набрать:

```
MZØ>@K
```

Далее приводится порядок работы на РМУ:

1. Перед включением машины убедитесь в том, что на других машинах нет сообщения: **загрузка из сети.**

2. Включите свою ЭВМ и видеомонитор. На экране появится сообщение: **загрузка из сети.**

3. Дождитесь появления таблицы.

4. Клавишами ⟨↑⟩ или ⟨↓⟩ установите нужный диск (для игр — диск В).

5. Нажмите на клавишу ⟨ВВОД⟩.

6. После появления меню нажмите два раза на клавишу ⟨АР2⟩ (в левом верхнем углу клавиатуры).

7. Клавишами управления (стрелками) подведите курсор к нужной программе (игре).

8. Нажмите на клавишу ⟨ВВОД⟩.

9. Клавишей ⟨←⟩ установите курсор на команде ⟨СТАРТ⟩.

10. Нажмите два раза на клавишу ⟨ВВОД⟩.

Если на экране появляется знак ⟨↑⟩, значит, ваша ЭВМ передает запрос на центральную машину. Знак ⟨↓⟩ информирует, что идет прием данных. Если случайно выбран не тот диск, то нажмите четыре раза на клавишу ⟨АР2⟩.

7.5. Сетевая система

Сетевая система *SOSNET* позволяет загружать ОС на РМУ. После загрузки каждое РМУ работает под управлением ОС *RT1RMU*. Состав системы:

— ОС, работающая на РМП
— ОС, работающая на РМУ
— Драйвер системного диска
— Драйвер локальной сети
— Драйвер удаленного диска
— Драйвер «квазидиска» для РМУ
— Программа загрузки и связи
— Быстрый загрузчик
— Программа создания загрузчика

— *RT1RMP.SYS*
— *RT1RMU.SYS*
— *II.SYS*
— *ME.SYS*
— *VD.SYS*
— *RD.SYS*
— *SOSNET.SAV*
— *QUICK.SAV*
— *GETMEM.SAV*

Для загрузки системы надо:

1. Загрузить центральную ЭВМ с дискеты система-сеть.

2. Запустить командный файл *R.COM:*

```
MZØ>@R
```

Командный файл создаст электронный диск на РМП, который для РМУ будет являться системным. Затем запустится на выполнение программа SOSNET. На экране появится сообщение о начале работы. Включите все РМУ. На экране РМП начнут появляться сообщения о РМУ, ждущих загрузки. После прихода всех сообщений нажмите на клавишу ⟨ВВОД⟩. Появится командная строка:

Команда ⟨Ж, З, Ф, К⟩

Введите команду З (загрузка) и нажмите на клавишу ⟨ВВОД⟩. После появления на РМУ приглашения *VD0*⟩ можно начинать работу. В процессе работы можно перезагружать РМУ. При этом будет производиться индивидуальная загрузка. Полное описание сетевой системы находится в файле *SOSNET.LST*.

Внимание! Если при загрузке SOSNET происходит зависание РМУ, то необходимо проверить номер центральной машины. Все переключатели на сетевом адаптере должны быть установлены в положение ВКЛ. После новой установки переключателей следует выключить, а затем включить ЭВМ. Проверьте реальный номер РМП. Для этого перейдите в режим отладки (режим 6) и наберите символ Т. На дисплее должно появиться сообщение:

Тест станции # 77.

Появление другого номера указывает на то, что неисправны переключатели адаптера локальной сети. Следует заменить адаптер.

Со стороны РМУ устройства распределяются следующим образом:

VD0: — системный диск. С него загружена ОС *RT1RMU*. Он находится в памяти РМП. Его размер ограничен. Таким образом, при обращении с РМУ к *SY:* или *VD0:* произойдет обращение к электронному диску на РМП.

VD1: — при обращении с РМУ к *VD1:* происходит обращение к магнитному диску *MZ0:* (верхний карман дисковода). Таким образом, *VD1*: соответствует *MZ0*:

VD2 — соответствует диску *MZ1*:

VD3 — соответствует диску *MZ2*:

VD4 — соответствует диску *MZ3*:

На РМУ можно выполнять все команды ОС, за исключением инициализации дисков. Программа *DUP* из-за ограниченного объема системного электронного диска не включена в систему. Кроме того, это позволяет застраховаться от случайного уничтожения информации на дисках.

Помимо устройства *VD* пользователю доступен драйвер терминала для РМУ — *TT:*, устройство удаленной печати *LP:*. На РМУ можно пользоваться электронным диском, но при этом теряется возможность работы с цветами. Включение электронного диска осуществляется командой: *SET RD ON*. После включения электронного диска его надо проинициализировать, но так как программы инициализации на системном диске нет, то сделать это средствами ОС нельзя. Для этого надо использовать команду: *SET RD INIT*. При этом драйвер электронного диска сам себя инициализирует.

Для печати файла с РМУ следует пользоваться стандартной командой операционной системы *PRINT*. Для вывода содержимого файла на дисплей РМП следует скопировать данный файл на устройство *LP6:*. Например:

```
VDØ>COP RD:TEST.LST LP6:
```

Внимание! При работе с графическими редакторами, транслятором языка Бейсик и текстовым редактором WRITER работа с электронным диском RD не допускается.

Более подробное описание сетевой системы находится в файле *SOSNET.LST*.

Е. КРАМЕР, В. ЖИГУНОВ

Использование периферийного процессора

Программирование периферийного процессора (ПП), как правило, вызывает трудности у начинающих программистов. Даже опытный программист, имеющий техническое описание УКНЦ, испытывает затруднения: в этой работе много программных тонкостей, требующих серьезного изучения. Но дело стоит того; именно в ПП скрыты самые интересные для программиста вещи. Размещение программ в ПП позволяет решить многие проблемы, связанные с программированием сложных задач, так как доступ к внешним устройствам (параллельному порту, магнитофону, программируемому таймеру, некоторым клавишам и др.) и многим регистрам возможен только со стороны ПП.

Данная статья предназначена для пользователей, знакомых с макроассемблером. При ее чтении необходимо дополнительно пользоваться документацией из комплекта поставки УКНЦ «Работа с внешними устройствами» У1.00031-01 90 01. Ниже приводится текст программы на языке макроассемблера, которая позволяет размещать программы в периферийном процессоре.

```
.MACRO .ENDPP
  MOV   #FLAG/2,@#177010
  CLR   @#177014          Обнуление флага
  RTS     PC
.ENDM
```

Массив параметров — последовательность ячеек, имеющая для каждого уст-

ройства фиксированную длину. Массив параметров готовится программой центрального процессора (ЦП) и адрес его передается через канал в ПП.

```
MASSIW::
MØM:    .BYTE   Ø
MØS:    .BYTE   Ø
M1M:    .BYTE   Ø
M1S:    .BYTE   Ø
M2:     .WORD   Ø
M3:     .WORD   Ø
M4:     .WORD   Ø
```

```
TT:                              Канал 2
   TSTB  @#176674        Регистр состояния
                                 (источник)
   BPL   TT                 Опрос готовности
   RETURN
```

Передача адреса массива параметров через канал осуществляется посылкой четырех байтов в следующем порядке: первый байт — 8 младших разрядов адреса, второй — 8 старших разрядов адреса, третий и четвертый байты равны 377.


```
PUTMP:
  MOV   #MASSIW,RØ
  CALL  TT                  Опрос готовности
                          Передача адреса МП
  MOVB  RØ,@#176676     Первый байт: 8
                    младших разрядов адреса
  SWAB  RØ
  CALL  TT
  MOVB  RØ,@#176676      Второй байт: 8
                  старших  разрядов адреса
  CALL  TT
  MOVB  #377,@#176676       Первый байт
                          завершения: 377
  CALL  TT
  MOVB  #377,@#176676       Второй байт
                          завершения: 377
  CALL  TT
  RETURN
```

Получив первые два байта, содержащие адрес массива параметров, и третий байт, отличный от нуля, управляющая программа в ПП пересылает массив параметров в свою память, используя механизм прямого доступа в ОЗУ ЦП. Определив тип устройства, она выполняет операцию с ним в соответствии с заданными параметрами. Передача данных между устройством и ОЗУ ЦП выполняется также с использованием механизма прямого доступа. После завершения операции управляющая программа записывает в младший байт первого слова массива параметров код ответа и читает из канала четвертый байт, сообщает таким образом о выполнении операции ЦП.

Если канал находится в неопределенном состоянии, то для приведения его в исходное состояние необходимо передать в него последовательность из четырех байтов, состоящую из трех нулей и кода 377.

```
INICAN:
Приведение канала в исходное состояние
176676 - регистр  данных связь ЦП и ПП
      CLR     RØ
      MOV     #3,R1
1$:   CALL    TT
      MOVB    RØ,@#176676          Посылка
                                 трех нулей
      SOB     R1,1$
      CALL    TT
      MOVB #377,@#176676     Посылка кода
                                      377
      RETURN
```

При записи информации на устройство программа должна проанализировать готовность источника (седьмой бит) для передачи информации. Если седьмой бит равен 1, то устройство готово принять очередной байт данных. После того как программа запишет очередной байт данных на регистр данных источника, устройство снимает готовность, а устанавливает его вновь только тогда, когда будет готово принять на регистр данных источника очередной байт данных.

```
TT:                              Канал 2
   TSTB @#176674         Регистр состояния
                                 (источник)
   BPL  TT                  Опрос готовности
   RETURN
```

Прежде чем выполнять команду записи, необходимо определить область памяти ПП, в которую можно будет записать программу.

```
FREEMEM:
   MOVB #1,MØS          1 - код "выделить
                                  память"
   MOVB #32,M1M          Тип устройства:
                                32 - ОЗУ ПП
   MOV  #-1,M3          -1 - выделение
                       максимальной памяти
   CALL PUTMP                 Передача МП
   CLR  R3
   CLR  R5
   MOV  M2,ADRPP
   MOV  M3,R3
   MOV  M2,R5
   RETURN
```

Формирование массива параметров для передачи области ОЗУ ЦП в ОЗУ ПП. Передается область ОЗУ ЦП, заключенная в логические скобки.

```
WRITE:
   MOVB #Ø2Ø,MØS          Команда записи
   MOVB #32,M1M             Устройство:
                                  ОЗУ ПП
   MOV  ADRPP,M2          Адрес ОЗУ ПП
   MOV  #SAVOZU,M3        Адрес ОЗУ ЦП
   MOV  #MEM-SAVOZU/2,M4   Число слов
   CALL PUTMP
   RETURN
```

Запуск программы в ОЗУ ПП. Стартовый адрес программы содержится в М2.

```
STARPP:
   MOVB #30,M0S          Команда СТАРТ
   MOVB #32,M1M      Устройство: ОЗУ ПП
   MOV  ADRPP,M2       Стартовый адрес
                                 ОЗУ ПП
   CALL PUTMP
   RETURN
SAVOZU:
;
;                          Тело программы
;
    .ENDPP       Выставление флага
MEM:

BEGIN:     Приведение канала в исходное
                                 состояние
      CALL INICAN       Выполняется один
            раз в начале: определение
             свободной области ОЗУ ПП
      CALL FREEMEM  Выполняется
          один раз: пересылка адреса МП
      CALL WRITE    Пересылка адреса МП
      CALL STARPP
      MOV  #177777,FLAG
1$:   TST  FLAG
      BNE  1$                  Если флаг
                            равен 0 - выход
      EMT  350
ADRPP: .WORD 0
FLAG:  .WORD 0
       .END BEGIN
```

В. ЖИГУНОВ, М. НОСОВ, И. СЕЛИВЕРСТОВА

Графическая обучающая система для УКНЦ

Что это такое? Это простая и удобная вещь, при помощи которой можно автоматизировать процесс обучения и контроля знаний по практически любым учебным дисциплинам. Циклы работы разбиты на уроки. Урок состоит из набора кадров для демонстрации, обучения и контроля. Кадры связываются в единый урок через задаваемые возможные варианты ответов.

В зависимости от выбранного ответа происходит переход на тот или иной кадр урока (например, для повтора или подробного объяснения). Есть и возможность вызывать любые командные файлы, чтобы выполнять команды операционной системы.

Что она может? Да почти все, что может УКНЦ:

создавать графику и текст на одном кадре;

воспроизводить любые звуковые эффекты и мелодии, создаваемые как в самой системе, так и в музыкальном редакторе;

использовать цветные спрайты (маленькие картинки) и их перемещения по экрану для оживления и иллюстрирования;

перемещать элементы изображения и отдельные части экрана, причем очень быстро;

использовать в качестве области ответа не только текст или числа, но и любое изображение (например, на уроке географии);

писать крупным шрифтом жирного начертания, который гораздо лучше стандартного и сохранит зрение учащихся;

загружать уроки на РМУ.

И для всего этого совершенно не надо знать языков программирования! Даже ученики смогут наработать уроки для учителя по его сценарию (например, в качестве домашнего задания).

Система запоминает количество допущенных ошибок и может выдавать их после опроса. При превышении заданного количества ошибок (его можно изменять в процессе опроса, добиваясь тем самым адаптации к возможностям учащихся) сеанс работы прекращается.

Чем она отличается от других АОС? Во-первых, простотой — лишь шестью управляющими клавишами пользуется учитель и тремя — ученик. Вся работа по созданию урока ведется через удобную систему меню на русском языке. Все режимы работы всегда перед глазами, и нет нужды учить какие-либо команды, чтобы управлять этой системой. И школьники, и студенты, и преподаватели осваивают ее за очень короткое время.

Во-вторых, богатыми возможностями для демонстрации и объяснения учебного материала. Использование системы ограничивается фактически фантазией педагога, а не ее возможностями.

В-третьих, экономностью: созданные уроки занимают на дискете очень мало места.

Как ее использовать? На основе этой системы можно создавать обучающие и контролирующие уроки, разнообразные меню и справочники, архивы и библиотеки, электронные учебники, информационные базы текстовых и графических данных, рекламные ролики и компьютерные клипы, живые и красочные отчеты о проделанной работе, разработку внешнего вида ваших программ, печатные издания с полутоновой графикой (например, указания для лаборатор-

ных работ или выпуски стенгазеты).

Поскольку различные кадры легко связываются воедино независимо от процесса их создания, очень просто подменять одно содержание другим при сохранении прежней структуры и взаимосвязи. Поскольку кадры можно связывать командными файлами, в них можно включать любые команды операционной системы.

Сколько же нужно дискет на эту роскошь? Совсем немного. Кадр используется в среднем полминуты и занимает 3—5 блоков на дискете. Для опроса всего класса в урок будет входить 15—30 кадров. Так что на дискете поместятся 15—20 уроков, а если информацию сжать, то гораздо больше.

Справится ли локальная сеть? Если делать кадры длительностью полминуты и более, то даже при работе с дисководом больших задержек не будет. Еще более убыстрить работу можно, использовав предварительную загрузку мест учеников вашими уроками (записав их перед уроком на квазидиск, входящий в состав системы). Ну а если использовать электронный диск (выпускается предприятием «Техноком», емкость 1 Мбайт), то его быстрота, бесшумность и надежность сделают вашу работу легкой и приятной. Один кадр длиной 5 блоков загружается на РМУ примерно за 3 с, так что в учебном классе с двенадцатью РМУ время загрузки будет составлять 36 с.

Как делать кадры? Редактор кадров открывается меню, выбрать из которого можно клавишами **ВВЕРХ, ВНИЗ** и **ВВОД**. Выбрав, например, режим «Точка», подведите появившийся графический маркер в нужное место и нажмите **ВВОД** — появится точка. Столь же проста работа и в других режимах. Для линии нужно указать начало и конец линии, для прямоугольника — диагональные вершины, для дуги — центр, начальную и конечную точки.

Кроме «Точки», «Линии», «Дуги» и «Прямоугольника» есть режимы:

«След» (за маркером будет идти след его перемещения);

«Цвет» (выбор текущего цвета и палитры цветов);

«Кисть» (закраска замкнутого контура любым цветом);

«Спрайт» (чтение и показ спрайта, созданного в редакторе CSEG);

«Макро» (можно оформить любую последовательность команд как макрокоманду, а затем вызывать ее сколько угодно раз в любом месте экрана. С помощью этого режима формируются сложные действия, например движение, звуковые эффекты);

«Метки» (можно ставить на кадре до 10 меток, а затем возвращаться к ним графическим маркером);

«Копия» (копирует любую часть экрана в любое другое место);

«Удвоить X» (растягивает любую часть экрана по оси X);

«Удвоить Y» (растягивает любую часть экрана по оси Y);

«Звук» (можно задавать любые тональности и длительности звучания);

«Ждать» (задержка от 1 мс до 1 мин, например чтобы дать время на прочтение текста);

«Текст» (ввод больших или малых русских или латинских букв, цифр и специальных символов);

«Ответ» (указание любой части экрана как варианта ответа; этот режим позволяет запускать командные файлы, например, для включения в систему других программ);

«Слой» (указание номера слоя для его рисования и редактирования; под слоем понимается некий дополнительный кадр, который может вызываться из основного кадра, причем очень быстро);

«Переход» (указание области экрана, при выборе которой учеником происходит переход на заданный слой и его показ);

«Выбор» (на экране показывается маркер в виде заданного вами спрайта, а ученик клавишами **ВПРАВО, ВЛЕВО, ВВЕРХ, ВНИЗ** подводит его к нужной области и нажимает **ВВОД**);

«Кадр» (для перехода к другому кадру из этого слоя);

«Лупа» (постоянный показ в левом верхнем углу увеличенной в 8 раз части экрана вокруг графического маркера);

«Линейка» (для показа расстояний, знакомест, отметки расположения);

«Трансл» (для перевода небольшого кадра в файл типа .LST, чтобы его можно было показать командой монитора TYPE с сохранением вывода графики и текста).

Почти все команды перечислены; освоить их, как вы видите сами, не представит труда.

Редактор кадров работает как на РМП, так и на РМУ, что позволяет ученикам самостоятельно строить программы обучения и контроля.

Состав системы.

1. Цветной спрайтовый редактор CSEG, позволяющий легко создавать небольшие картинки — спрайты. Работа с увеличенным изображением и одновременный показ спрайта так, как он будет выглядеть на экране, позволяют легко рисовать сложные и красивые мелкие части изображения. Максимальный размер спрайта 32×32.

Режимы редактора (выбираются через меню): выбрать текущий цвет; поставить

точку текущим цветом; вести за маркером след текущим цветом; сменить формат экрана (80, 40, 20 или 10 символов в строке); записать спрайт в буфер, считать его оттуда, наложить, исключить; записать на диск, считать с диска спрайт или буфер; изменить шаг движения маркера; заполнить спрайт выбранным цветом.

Этот редактор используется также при разработке игровых программ, например, для создания фаз движения фигурок и записи их в буфер, где все они могут быть видны одновременно.

2. Текстовый редактор NED, простой, но мощный, дополнен системой меню. Теперь даже неподготовленный человек без труда сможет работать с текстами: он избавлен от необходимости учить команды редактора. Квалифицированный же пользователь сможет работать быстро, так как традиционные команды сохранены.

Основные возможности редактора: смена регистров и алфавитов уже набранного текста; работа с макрокомандами; многократный повтор действия или макрокоманды; поиск в тексте по задаваемому образцу; работа в режиме вставки вводимых символов или их замены; задание правой границы текста для автопереноса слов при вводе; выравнивание текста по правой границе; чтение из дополнительных файлов, запись в них части текста (или печать); вставка даты и времени коррекции файла и автоматическое их изменение и многое другое.

В данной системе редактор используется для создания командных файлов.

3. Драйвер квазидиска с поддержкой цветной графики VN.SYS позволяет использовать часть экранной памяти для организации в ней квазидиска размером 60 блоков, что очень сильно ускоряет работу с системой, особенно в сети. Его отличие от известного драйвера KD.SYS в том, что он полностью сохраняет возможность работы с цветной графикой благодаря экономии видеопамяти, что достигается в режиме 40 символов в строке.

4. Сетевые средства. Система загружает операционную систему RT-11 на РМУ, загружает заданные программы на конкретные РМУ, поддерживает обмен по сети, позволяет просматривать каталоги дисков при выборе файлов.

С РМП видно, какое РМУ что сейчас делает (чтение, запись, загрузка). Система может разрешать конкретным ученикам работать только с определенными устройствами (с дискетами, электронными или логическими дисками, квазидисками и принтерами). Таким образом, можно быть уверенными в том, что ваши программы и данные не будут потеряны по чьей-либо вине или оплошности.

5. Демонстрационный урок по химии (тема «Химическая связь») — сокращенный вариант из имеющегося набора уроков. В нем использованы движение спрайтов, звуковые эффекты и другие возможности системы. Наилучший эффект достигается при использовании цветного монитора, но и на черно-белом разные цвета, отображаемые градациями яркости, выглядят весьма информативно. Тем более что при разработке наших уроков мы ориентировались на черно-белые мониторы.

6. Операционная система RT-11 с меню выбора и запуска средств системы. После того как вы вставите дискету в дисковод и запустите загрузку с диска, операционная система выйдет в меню средств обучающей системы. Оно также создано в редакторе обучающей системы и может быть иллюстрацией к возможностям ее использования.

После завершения работы с выбранным пунктом меню возникает снова; пункт ВЫХОД позволит вам выйти в режим команд монитора.

Дополнительные возможности. Быстрая загрузка операционной системы на РМП (в три раза быстрее обычной); настройка системы на национальные шрифты и алфавиты; работа с джойстиком (параллельно с клавиатурой); сжатие кадров почти в два раза для экономии места ни дискете, причем с моментальной распаковкой при использовании; печать изображения с экрана (с полутоновой графикой) для создания методических и информационных листков и брошюр (программа MN); создание собственных меню в нижней служебной строке или вывод туда сообщений при помощи программы MN. Так были сделаны меню для команд операционной системы, редакторов NED и DESS, дополнительная диагностика для редактора графической обучающей системы.

Недостатки. Естественно, у каждой программы есть свои плюсы и минусы. И как обычно, недостатки системы являются следствием ее достоинств. Попробуем это показать.

Существует масса критериев, по которым можно оценивать результат. Возьмем самые основные.

Обычно потребителя в первую очередь волнует надежность. Передача кадров на места учеников осуществляется под управлением программы сетевого монитора, работающей в рамках операционной системы. Сама программа сетевого монитора отличается высокой эффективностью и качеством передачи данных. Нарушить ее работу может только полная неработоспособность сети. Что касается рамок операционной системы, то это означает, что передача сопровождается мас-

сой файловых операций. На практике это приводит к интенсивной работе накопителя, отсюда повышенные требования к нему и используемым дискетам. Таков минус. Но то, что каждое РМУ становится настоящим местом работы с дисководами, принтером и операционной системой, т. е. настоящей ПЭВМ,— плюс.

При переходе от кадра к кадру (в зависимости от выбранного ответа) система может либо непосредственно переходить на требуемый кадр, либо использовать командные файлы, в которых указаны имена нужных кадров. Завершение одного кадра вызывает загрузку следующего. Во время урока часто случается, что несколько РМУ одновременно запрашивают очередные кадры. Загрузка кадра (несжатого) занимает до пяти секунд. Очевидно, что скорость передачи информации по локальной сети в учебных классах на базе УКНЦ существенно увеличить не удастся; остается искать программные способы. Например, можно сжимать файлы. Можно также использовать драйвер квазидиска, входящий в состав системы. Он позволит один раз загрузить кадры на РМУ, а затем многократно вызывать их, не используя накопитель. Еще более ускорить загрузку может использование драйвера квазидиска на РМП (здесь может работать и KD.SYS). Еще можно использовать электронный диск.

Для работы с системой в полном объеме начинающему пользователю требуется получить некоторые навыки работы с текстовым и спрайтовым редакторами. При создании кадров в редакторе LL вопросов обычно не возникает благодаря наглядной системе меню. И бесспорно, освоить Бейсик гораздо сложнее, чем графическую обучающую систему.

При создании сложных и адаптирующихся обучающих курсов их структура может быть довольно запутанной. Поэтому приходится сначала составлять сценарий обучения на бумаге, т. е. продумать все взаимосвязи и переходы, и только потом перекладывать на ЭВМ. Кроме того, операционная система RT-11 стандартно допускает примерно до 190 файлов в каталоге дискеты. Поэтому лучше создавать на дискете логические диски под файлы каждого урока (используя драйвер LD.SYS).

Существует много способов ответа на вопрос в обучающих системах: от выбора одного варианта из нескольких предложенных («угадайка») до ввода ответа в произвольной форме (при этом система должна анализировать введенные данные и сравнивать их с эталонными). Первый способ как раз и реализован у нас. Он прост, понятен, преподавателю легко создать варианты ответов, а ученику нетрудно передвигаться от варианта к варианту. Однако это приводит к переносу центра тяжести системы именно на обучающую часть, а не на контрольную. Второй же вариант позволяет делать мощные опросы и экзамены, способные ужаснуть любого двоечника, но на обучение остается весьма мало средств. К тому же создание обучающих программ на основе подобных систем значительно сложнее и требует гораздо более подготовленных преподавателей.

Остается заверить, что система будет развиваться и улучшаться и все недостатки будут устраняться в следующих версиях.

Взгляд преподавателя. В этих коротких заметках — попытка дать анализ применения новой графической обучающей системы (ГОС, разработка предприятия «Техноком») на уроках химии в школе № 852 Москвы. Мы попытались создать уроки по наиболее трудным разделам химии при помощи описанной выше обучающей системы.

Главное достоинство ГОС заключается в том, что она позволяет составлять обучающие программы-компьютерные уроки по наиболее трудным разделам химии. Такие уроки, в отличие от привычных тренажеров, включают в себя как обучающую (дидактическую), так и проверочную части. По дидактическим целям эти уроки можно отнести к урокам-практикумам, поскольку они помогают обучаемому выработать практические навыки и умения.

Каждый урок начинается с наглядного образного представления теоретического материала, для чего используются динамические возможности компьютерной графики и звукового сопровождения, имеющиеся в данной ГОС. Создание движущихся изображений на экране дает возможность использовать ГОС для составления программ-демонстраций природных явлений, динамического представления разнообразных процессов. Это представляет большой интерес для учителей химии, поскольку позволяет «оживить» схемы производств различных веществ, модели взаимодействия атомов и реакций между молекулами и т. п.

Во время урока учащиеся работают только с клавиатурой и экраном монитора. Информация подается в темпе, доступном среднему ученику. Любой фрагмент урока можно рассматривать произвольно долго.

Затем следует тест на правильность понимания данного материала. Тест включает проверочные задачи, усложняющиеся по мере работы на уроке. При неправильном решении проверочной задачи ученик либо возвращается к исходному теоретическому материалу, либо переходит к объяснению допущенной ошибки. Взаимосвязь переходов при необходимости может быть легко изменена.

Что привлекает в таком уроке?

1. Учащиеся сами выбирают темы работы, время просмотра отдельного кадра урока.

2. Работа по изучению нового материала ведется только с экраном видеомонитора.

3. Учитель с помощью ГОС может менять фрагменты готового урока по своему плану, с учетом дидактических задач.

4. Работа на таком уроке настраивает учащегося на самостоятельное изучение материала, активизирует познавательную деятельность ученика.

5. Возможность подходить индивидуально, дифференцированно к обучению, так как ГОС позволяет сделать очень наглядный, гибкий и образный «электронный» учебник, «листая» который каждый обучаемый выбирает себе урок по желаниям и возможностям. Подобный учебник может использоваться как тренажер по химии для желающих повторить пройденный материал или еще раз проработать самостоятельно трудную для восприятия тему. Такой опыт уже имеется, он показал, что программы-демонстрации весьма помогают учащимся в освоении труднейших тем курса химии.

Работа над дидактической частью наших уроков подсказала еще одно любопытное применение ГОС, полезное для уроков информатики: демонстрация возможностей ПЭВМ УКНЦ (различные звуковые эффекты, быстрая графика, цвета и их мгновенная замена, тексты различными шрифтами и др.).

Более того, наиболее способные учащиеся, проявляющие интерес к работе на ПЭВМ, могут при помощи ГОС самостоятельно составлять разнообразные обучающие программы по любым предметам и разделам школьного курса. Этим достигается многое: ученик расширяет свои умственные возможности, глубже овладевает изучаемым материалом, систематизирует свои знания, познает возможности УКНЦ. Работа с системой дает возможность учащимся осваивать современные методы обработки текстовой и графической информации на ЭВМ.

Использование учителем-предметником ГОС на своих уроках позволяет разнообразить формы и методы обучения, оживить преподавание предмета, а главное, позволяет не только ученику, но и учителю другими глазами взглянуть на содержание обучения. Ибо, как сказал известный химик А. Н. Несмеянов,

«Главное, что должно дать образование и о чем часто забывают,— это не «багаж» знаний, а умение владеть этим «багажом». Это и есть главная цель любого образования.»

Калейдоскоп УКНЦ

Как в диалоговом режиме изменить строку работающей Бейсик-программы? Например, заменить определение функции в первой строке следующей программы:

```
10 DEF FN Y(X)=SIN(X/2)
20 REM Основная часть программы
30 ...
```

А. Моряхин (пос. Октябрьский Архангельской обл.) предлагает использовать для этого диск (магнитный или электронный — драйвер KD). Суть приема ясна из примера.

```
10 INPUT "Введите функцию";FUN$
20 FUN$="10 DEF FN Y(X)="+FUN$
30 OPEN "PROM.ASC" FOR OUTPUT
40 ?#"20 GOTO 80"
50 ?#FUN$
60 CLOSE
70 MERGE"PROM",R
80 REM Основная часть программы
90 ...
```

При работе в сети следует позаботиться об индивидуальной нумерации текстовых файлов, поступающих с РМУ.

Е. Ягунов (п. Болшево Московской обл.) знает, что делать, когда на запрос чтения файла с РМУ выдается ответ «файл не найден», в то время как файл на диске есть. Оказывается, достаточно набрать на РМУ минимальную программу (например, 0?⟨ВК⟩) и записать ее на диск (SAVE «О⟨ВК⟩). Все файлы сразу находятся.

А. Б. Розенфельд (село Ново-Егорьевское, Алтайский край) предлагает усовершенствование к программе печати графики, опубликованной в «Калейдоскопе УКНЦ» в № 5 — 91, значительно повышающее качество печати. Нужно заменить в исходной программе две строки

```
1030 LPRINT CHR$(27);"@";
1230 LPRINT CHR$(27);"J";CHR$(24);
```

и добавить

```
1235 LPRINT CHR$(13);
```

Пример программы, управляющей режимом подчеркивания терминала (так же можно управлять инверсией, включением дополнительной клавиатуры и т. п.), прислал В. Б. Африканов (г. Балаково, Саратовская обл.). Для включения подчеркивания в регистр данных терминала нужно занести коды 33 и затем 244, для выключения — 33, 277 и 244. Коды для других операций приведены в заводском описании «Работа с внешними устройствами» (У1.00031-01 90 01—Лу).

```
10 P%=&0244     'Код включения
                          подчеркивания
20 V%=0%        'Второй код сейчас
                          не нужен
30 GOSUB 60000'Включить подчеркивание
40 PRINT "..."'Печать текста
50 P%=&0277     'Первый код выключения
60 V%=&0244     'Второй код выключения
70 GOSUB 60000'Выключить подчеркивание
80 END
60000 IF PEEK(&0177564)=0%TH60000'Тер-
                 минал готов? Нет - ждать
60010 POKE &0177566,27% 'Передать
                          код 33
60020 IF PEEK(&0177564)=0%TH60020
60030 POKE &0177566,P%  'Передать
                          первый код
60040 IF V%=0TH60070    'Второй код
                          не нужен?
60050 IF PEEK(&0177564)=0%TH60050
60060 POKE &0177566,V%  'Передать
                          второй код
60070 RETURN
```

В. С. Осипов (г. Курган) сообщает, что для изменения формата экрана нужно использовать кодовую последовательность 33 246 n, где n=61 для включения формата 80 знаков в строке, n=62 для формата 40 знаков в строке, n=63 — 20 знаков, n=64 — 10 знаков. Эти коды в документации не приведены.

В. ЛАРИН

Использование драйвера квазидиска

Драйвер KD.SYS позволяет превратить 64 Кбайт (из 92 Кбайт) видео-ОЗУ УКНЦ в квазидиск (электронный диск) с малым временем доступа и емкостью при стандартной инициализации 116 блоков.

УКНЦ имеет три плана видео-ОЗУ, каждый из которых используется для хранения своего цвета (красного, зеленого, синего) и имеет объем 32 Кбайт. Драйвер квазидиска KD.SYS использует два из них для хранения информации, а третий остается для работы видеомонитора в монохромном режиме.

Для работы с квазидиском необходимо выполнить следующие команды:

```
LOAD KD          (загрузка драйвера в ОЗУ)
SET KD ON        (включение драйвера)
INIT/NOQ KD:  (создание каталога
                          квазидиска)
```

Чтобы убедиться в готовности к использованию созданного квазидиска, можно подать команду

```
DIR KD:
```

и посмотреть созданный каталог на экране.

Драйвер отключается командой

```
SET KD OFF
```

при этом теряется вся информация на устройстве KD.

Приведем пример использования драйвера. При работе в Бейсик-сети инициатором обмена с РМП является РМУ. Если для урока необходимо на каждое ученическое место загрузить один и тот же файл, а такое встречается достаточно часто, то с каждого РМУ должна быть подана команда

```
LOAD "имя файла",R
```

С диска неоднократно считывается один и тот же файл, при этом столько же раз работает дисковод. Учитывая ненадежность и невысокое быстродействие дисководов, желательно считывание файла производить с квазидиска, предварительно его создав. Допустим, на уроке всеми учащимися должен быть использован файл UROK1.ASC.

Лучше всего перед уроком в текстовом редакторе создать командный файл KD.COM

такого содержания:

```
LOAD KD
SET KD ON
INIT/NOQ KD:
COPY DK:(UROK1.ASC,NET.COM) KD:
ASS KD: DK:
@NET
```

Командный файл NET.COM предназначен для установки программы связи NETUK и содержит следующие команды:

```
UNLOAD MC
SET MC ADDR=77
LOAD MC
R NETUK
```

Он, естественно, тоже должен быть сделан заранее в текстовом редакторе.

После создания командного файла KD.COM необходимо загрузить на РМУ Бейсик, выйти в операционную систему и запустить файл KD.COM командой

```
@KD
```

По окончании его работы на РМП устанавливается программа связи NETUK, и дальнейшая работа идет как обычно. Но после команды с РМУ

```
LOAD "UROK1",R
```

загрузка файла UROK1.ASC идет с квазидиска.

Поскольку драйвер квазидиска имеет емкость 116 блоков, то на него можно скопировать (четвертая команда в файле KD.COM) и другие необходимые для урока файлы.

Работая с квазидиском, дисковод можно вообще отключить (но не забудьте, что при перезагрузке РМП, сбоях электропитания, нажатии клавиши СТОП информация на квазидиске не сохраняется).

ЧТО МОЖЕТ ЭВМ?

GPSS — старый язык в новом обличии

Моделирование различных процессов на компьютере давно перестало быть прихотью программистов и превратилось в мощный аппарат анализа и синтеза. Для описания моделей реальных систем создано достаточное количество языков моделирования. Одним из первых широкое распространение получил язык моделирования дискретных процессов GPSS.

И вновь, спустя много лет, долгожитель получил наивысший рейтинг среди языков аналогичного класса. А возвратил былую популярность некогда самому мощному средству моделирования удачный симбиоз с персональными компьютерами, получившими в результате технологического прогресса цветную графику, огромную память и высокое быстродействие.

В результате простой и легко изучаемый язык, позволяющий рассчитать пропускную способность городского транспорта, запланировать подвоз сырья на склады предприятия, составить удобное расписание занятий и даже выбрать наиболее выигрышную стратегию для игры в «Спортлото», стал доступен и понятен даже новичкам в программировании.

И емкость, и скорость

Конкурировать с японскими электронщиками нелегко. Ведь в то время, пока на некоторых европейских фирмах ведутся еще только разработки 1-Мбитной запоминающей БИС, конструкторы японской фирмы Mitsubishi Electric разрабатывают меры по очередному снижению времени доступа в уже существующих статических ЗУ с произвольной выборкой аналогичной емкости. Их последним достижением являются образцы двух СЗУПВ с временем доступа к любому элементу не более 35 нс.

Одна из микросхем M5M51001 состоит из 1 Мслова по 1 биту каждое. Другая, M5M51004, имеет 256K 4-битных слов. Кристаллы ЗУ размером 6,1×15, 84 мм встроены в стандартные корпуса с 28 выводами.

С апреля 1991 г. фирма выпускает ежемесячно по 30 тыс. высокоскоростных запоминающих устройств со стоимостью одного бита, равной двум сотым цента.

Дистанционная клавиатура

Очень удобная вещь! Вы можете набрать на ней какой-нибудь текст и по телефону (в «клавиатуру» встроен акустический модем) передать в свой компьютер. А также получить от компьютера ответ, запомнить его в ОЗУ «клавиатуры» и прочитать на дисплее размером в одну строку (40 знаков). Или записать на магнитную ленту с помощью подключенного диктофона. Наконец, распечатать на прилагаемом карманном принтере. Ну, а если до компьютера ближе, чем до телефона, можно подключиться прямо к нему через разъем RS-232.

Называется описываемое устройство TX-1000. Оно весит 450 г, способно передавать информацию не только на компьютер, но и на факс или другое TX-1000. Емкость ОЗУ — около 40 Кбайт, длина строки принтера — 40 символов.

Прибор пригоден и в качестве записной книжки, хотя для этой цели дороговат — 750 долларов. Зато в планах фирмы «РФР Лтд», продающей его в Москве, организация сетевого центра — компьютера на АТС, на который могли бы «сбрасывать» информацию владельцы TX-1000, так что получится нечто вроде электронной почты.

Bernulli набирает высоту

Фирма Iomega, известная своими накопителями на гибких сменных магнитных дисках, использующими эффект Бернулли (о них мы рассказывали в № 6—90, с. 128), выпустила новую модель — Bernulli Transportable 90. На дискету диаметром 5,25 дюйма этот накопитель записывает 90 Мбайт данных!

Право быть клиентом нужно заработать

В компьютерных сетях обычно есть центральный компьютер. Он обслуживает «периферийные» машины, предоставляет им ресурсы своих внешних устройств (дисковая память, лазерный принтер), обеспечивает информационный обмен. Такой компьютер называют файл-сервером, поскольку основная его задача — хранение и рассылка файлов.

По мере роста числа подключенных компьютеров файл-серверу приходится все труднее, поскольку в классической модели сети он должен брать на себя все заботы по обслуживанию. Если, скажем, какой-то из «периферийных» компьютеров выдает заказ на выборку информации из централизованной базы данных, файл-сервер должен считать файл, содержащий нужные данные, выделить их, сформировать из них новый файл и послать заказчику.

В последнее время все большее распространение получает сетевая архитектура, названная «клиент—сервер». Ее отличие в том, что центральный компьютер берет на себя минимум забот, лишая «периферийные» статуса нахлебников и заставляя их более активно участвовать в работе сетевых программ. В примере с базой данных это означает, что сервер посылает по запросу «сырой» файл, а «периферийная» машина сама выделяет из него нужные данные.

О многочастотной кодовой модуляции

Магнитофон — слабое место БК-0010: низкая плотность записи и низкая надежность считывания информации делают работу с длинными файлами довольно мучительной. Файл длиной 15,5 Кбайт, т. е. занимающий всю память БК в обычном режиме, загружается почти 2 мин и занимает ленту длиной около 5 м! А на таком отрезке ленты весьма вероятно присутствие дефектов, делающих чтение невозможным. Про расход ленты нечего и говорить — на одной кассете МК-60 можно записать (на двух дорожках) не более 512 Кбайт, что с учетом необходимости дублирования информации и места, занимаемого паузами и оглавлениями файлов, дает реальную емкость не более 200 Кбайт на кассету.

Что представляет собой формат магнитной записи БК-0010? Прежде всего, запись производится в так называемом ЧМ-формате — с частотной модуляцией, т. е. информационная последовательность рассматривается побитно, причем двум возможным состояниям каждого бита (0 и 1) соответствуют две различные частоты импульсов (на БК — 3120 и 1560 Гц соответственно).

В начале файла записывается 10000(8) «нулевых» импульсов — так называемая установочная (или настроечная) последовательность. По ней компьютер находит начало файла и подстраивает драйвер чтения под конкретную скорость воспроизведения данного магнитофона.

Затем записывается маркер — особая последовательность импульсов, не встречающаяся в файлах. Благодаря маркеру компьютер может отличить начало файла от встретившихся случайно последовательностей импульсов и точно установить фазу считываемого сигнала — ведь полярность выхода магнитофонов может быть различной.

После маркера следует оглавление файла. Оно состоит из 10 машинных слов: 1-е — адрес файла, 2-е — длина файла в байтах, остальные 16 байтов — имя файла. Оглавление обеспечивает идентификацию и правильную загрузку файла.

Далее следует тело файла — собственно информационная последовательность, имеющая указанную в оглавлении длину. По окончании тела файла на МЛ записывается еще одно слово — контрольная сумма (КС) файла: результат сложения всех байтов файла с учетом разряда переноса. При подсчете КС учитывается только тело файла, оглавление и сама КС в расчет не входят. Наличие КС на МЛ позволяет после загрузки файла с достаточной уверенностью определить, не возникли ли ошибки при чтении — вероятность случайного совпадения КС для файлов среднего размера не превышает 0,002 %.

Информационная последовательность нулей и единиц представлена на МЛ последовательностью коротких и длинных импульсов, разделенных синхроимпульсами, равными по длительности нулевому информационному импульсу. Предполагалось, что это повысит надежность чтения информационных импульсов, но эффект оказался обратным — для получения при таком способе записи той же плотности информации среднюю частоту следования импульсов пришлось увеличить примерно в 1,67 раза, а это в свою очередь привело к снижению надежности вследствие низкого качества магнитных лент и значительных фазовых искажений бытовых магнитофонов. Кроме того, при единичном сбое компьютер легко путает последовательность информационных импульсов с последовательностью синхроимпульсов; вся информация после такого сбоя теряется — читаются только нули.

Как улучшить работу БК с магнитофоном? Простейший путь повышения плотности записи (уменьшение скорости МЛ при записи и/или увеличение ее при чтении) чреват снижением надежности. Можно исключить синхроимпульсы, что позволит увеличить плотность в 1,67 раза, а при некоторых дополнительных мерах и в два раза. Такой прием с успехом применен С. Ивашинниковым в копировщиках «ТУРБО». А нет ли других резервов?

Вспомним, что при частотной модуляции информационная последовательность кодируется побитно, причем каждому из двух состояний бита соответствует своя частота. Два состояния — две частоты. А если кодировать не отдельный бит, а пару? Два бита имеют четыре возможных состояния — 00, 01, 10 и 11. Припишем каждому из состояний свою частоту. Если среднюю частоту полученных импульсов выбрать равной средней частоте ЧМ-формата, плотность записи повысится вдвое — ведь каждый импульс теперь кодирует не один, а два бита и при равном числе импульсов количество кодируемой информации удваивается. Способ этот, описанный А. Молчановым в «ИНФО» № 3 — 91, может быть назван многочастотной кодовой модуляцией (МКМ), так как каждый импульс на МЛ не просто отображает, а кодирует определенное слово из 2 битов. МКМ реализована в популярных копировщиках HELP4S и HELP7, обеспечивает увеличение плотности записи в 3—4 раза при заметном повышении надежности.

Однако существующие программы, работающие с МКМ-форматом, хороши прежде всего для архивного хранения информации. Хотя они и позволяют не только копировать, но и запускать загруженные в МКМ-формате файлы, работать с ними неудобно — нужно сначала загрузить копировщик (длиной от 10000(8) до 22000(8)) и только потом можно загрузить и запустить файл. HELP7Z позволяет отчасти исправить положение — он записывает перед файлом МКМ-формата программу-загрузчик с автозапуском, которая уже грузит и запускает файл. Но этот способ хорош лишь для исполняемых программ — ну какой, скажите, смысл «запускать» текстовый файл? Таким образом, остается множество сфер деятельности, в которых МКМ-формат пока неудобен: работа с магнитофоном из текстовых и графических редакторов, языков высокого уровня, программ пользователя, формирующих собственные файлы, и т. п.

Кроме всего прочего, HELP4S и HELP7 идентифицируют МКМ-форматность файла с помощью символов-спецификаторов, вводимых в имя файла (14—16-й символы имени). Однако именно эти символы используются для спецификации во многих программах — Бейсике-MSX, файлах данных Фокала, блоках файлов графредактора P1.0M Бакерина и т. д. Записывать такие файлы с помощью HELP4S и HELP7, естественно, нельзя. Еще одним недостатком этих программ является то, что оглавление файла и КС записываются на МЛ с повышенной скоростью в ЧМ-формате, что заметно снижает общую надежность чтения.

Автором статьи разработан способ записи на МЛ, позволяющий устранить перечисленные недостатки.

В основе способа лежит 4-частотная МКМ, причем установочная последовательность и оглавление файла записываются в обычном ЧМ-формате и могут читаться на БК без всяких дополнительных средств; отличие от формата А. Молчанова в том, что запись ведется с обычной скоростью — 1200 бод. Имя исходного файла (в отличие от всех известных способов, включая HELP7 и ТУРБО) не искажается — идентификатором МКМ-файла служит не символ имени, а длина файла; она принимается на 100000(8) превышающей истинную. КС записывается после тела файла в МКМ-формате, благодаря чему надежность ее чтения заметно повышается.

В состав разработанного пакета программ входят три основных программы — копировщик, каталогизатор и резидентный драйвер.

Копировщик KMKM позволяет загружать файлы как в стандартном, так и в МКМ-формате без объявления имени, записывать на МЛ заданное число копий в выбранном формате, работать с одним или двумя магнитофонами в ручном и автоматическом режимах. Копировщик имеет весь ассортимент известных сервисных функций — поиск на МЛ, изменение любых параметров файлов, ввод и удаление автозапуска. Автозапуск вводится корректный, т. е. исключающий запуск неправильно загруженного файла, и перемещающий — автозапуском может быть снабжен файл любой длины, рассчитанный на работу по любому адресу, при этом длина файла возрастает лишь на длину блока автозапуска (206(8)). Копировщик имеет встроенный символьный дамп и возможность запуска загруженных кодовых программ, причем запуск не является принудительным, а осуществляется вручную после выхода из копировщика. Кроме того, копировщик позволяет «оснастить» любой текстовый файл программой чтения текста READER.4. Загрузка файла в KMKM производится с визуальным контролем, а возможность зависаний при загрузке, порча копировщика или случайная запись копий с ошибкой полностью исключены. Тип загружаемого файла (ЧМ или МКМ) определяется автоматически. Максимальная длина копируемых файлов при размещении KMKM в ОЗУ пользователя может достигать 60000(8); длина KMKM равна 6400(8).

Каталогизатор TMKM позволяет проверять файлы на наличие ошибок чтения. В процессе работы TMKM формирует протокол проверки, который можно просмотреть

на экране, записать на МЛ, распечатать на принтере МС 6313 (или аналогичном). Возможны фиктивное чтение файлов, редактирование протокола, ввод заголовков и номера первого файла, загрузка протоколов с МЛ для просмотра или продолжения (с сохранением нумерации файлов). Процесс чтения файла индицируется. Длина ТМКМ — 3500(8).

Резидентный драйвер ДМКМ предназначен для обеспечения работы любых программ пользователя, обращающихся к магнитофону через стандартное системное прерывание EMT 36 и не изменяющих в процессе работы вектор 30, в том числе из Фокала, Бейсика, пускового монитора и МСД, почти всех известных на БК редакторов текста, графических редакторов, ассемблер-систем и т. п. Возможна запись на МЛ как в стандартном формате БК, так и в МКМ-формате; при чтении тип файла определяется автоматически. Обеспечивается загрузка (или запись) файлов по адресу и с именем, указанными в блоке параметров EMT 36, а после загрузки (или записи) файла обеспечивается установка системных признаков в соответствии с принятым для EMT 36 стандартом (т. е. полностью эмулируется работа EMT 36 БК-0010). Кроме того, имеется возможность загрузки первого найденного файла без объявления имени (в любом формате). Никаких дополнительных сообщений драйвер на экран не выдает и работу подавляющего большинства программ пользователя не нарушает. Перехват прерывания EMT 36 в ДМКМ выполнен таким образом, что при его запуске на фоне других системных программ, перехватывающих прерывание по вектору 30 (отладчиков, драйверов принтеров и т. п.), он не нарушает их работу, являясь как бы «суперрезидентной» программой. Длина ДМКМ, включая блок для выхода в коды, Фокал или Бейсик, всего 2000(8).

В КМКМ и ДМКМ принят односкоростной способ записи — в ЧМ-формате с обычной скоростью — порядка 1200 бод, а в МКМ-формате — со скоростью около 4000 бод, выбранной из соображений достаточной плотности записи, высокой надежности и пониженных требований к магнитофону и МЛ. Все три программы написаны в перемещаемом формате, могут работать как в ОЗУ пользователя (ДМКМ — с учетом свободной зоны, не используемой другими программами), так и в дополнительном ОЗУ или ПЗУ по любому адресу.

Разработанный пакет, в отличие от существующих копировщиков, является функционально полным, обеспечивая все необходимые операции при работе с файлами. Сейчас ведется разработка специальной программы для восстановления испорченных файлов, записанных в МКМ-формате, и загрузчика.

Ю. ЗАЛЬЦМАН

Еще раз о защите

Многие пользователи БК-0010 хотели бы защитить свои программы на Бейсике от внесения изменений, а оригинальные алгоритмы, используемые в этих программах, от изучения другими программистами. Для этого используются хитрости с вызовом подпрограмм в машинных кодах или внесение в исходный текст символов, препятствующих его распечатке по команде LIST. Я предлагаю более радикальное решение этой проблемы.

Транслятор с Бейсика перед выполнением программы компилирует ее в «шитый» код; после этого можно уничтожить исходный текст программы (оставив необходимые для ее выполнения «шитый» код и системную область). Восстановить исходный текст по «шитому» коду крайне трудно даже для опытного программиста, а в случае большой и сложной программы это практически невозможно.

Предлагаемая здесь программа BASCOD выполняет все необходимые действия по преобразованию отлаженной Бейсик-программы в файл с «шитым» кодом, который можно загружать и запускать в мониторе как обычную программу в машинных кодах. Программа BASCOD полностью перемещаема. Текст написан на ассемблере (Микро-11 или более ранние версии).

```
MOV #3052,R1        ;Начало текста
MOV #1000,SP
MOV @#2002,R0       ;Адрес конца
                    ;       Бейсик-программы
```

```
        BIT #1,R0
        BEQ 1
        INC R0
1:      CMP (R0)+,#154754  ; * Поиск первого STOP
        BNE 1              ; *
        MOV R0,-(SP)
        MOV #104014,(R1)+  ; * Запись программы
                           ; *              останова
        MOV #12706,(R1)+   ; *
        MOV #1000,(R1)+    ; *
        CLR (R1)+          ; *
3:      CMP @R0,#154722    ;Начало цикла
                           ;           перекодировки
        BEQ 10
        CMP @R0,#154754
        BNE 30
        MOV #3052,@R0      ;STOP заменен на Выход
30:     CMP (R0)+,#156156  ; * Проверяется
        BNE 3              ; *         подозрение на
        CMP @R0,@SP        ; *         использование
        BGE 3              ; *  текстовой константы
        CMP @R0,#3052      ; *
        BLE 3              ; *
        CMP 2(R0),#400     ; *
        BHI 3              ; *
        TST 2(R0)          ; *
        BEQ 3              ; *
        MOV @R0,R2         ;Найдена текстовая
        MOV R1,(R0)+       ;  константа, перемещаем
        MOV (R0)+,R3       ;  ее в начальные адреса
                           ;                  кода
4:      MOVB (R2)+,(R1)+
        SOB R3,4
        BR  3
10:     MOV R0,R2          ;Найден END
        MOV #3052,(R0)+    ;END заменен на Выход
        BIT #1,R1
        BEQ 5
        INC R1
5:      MOV #1000,R0       ; * Перенос программы
        MOV PC,R3          ; *   начального запуска
        ADD #216,R3        ; *
20:     MOV (R3)+,(R0)+    ; *
        TST @R3            ; *
        BNE 20             ; *
        MOV #12704,(R0)+   ; *
        MOV @SP,(R0)+      ; *
        MOV #134,@R0       ; *
        MOV (SP)+,R0
6:      MOV #377,(R1)+     ; * Забитие бывшего
        CMP R1,@#2002      ; *                текста
        BLT 6              ; *
7:      MOV #14,R0         ;Начинается запись файла
        EMT 16             ;      с перекодированной
                           ;                программой
        MOV #232,R0
        EMT 16
        MOV #25,R0
        EMT 16
        EMT 16
        MOV #320,R1
        MOV #2,(R1)+
        MOV #1000,(R1)+
        MOV @#2030,@R1
        SUB #774,(R1)+
        MOV R1,-(SP)
        MOV #100724,R3     ; * Ввод имени файла
        JSR PC,@#100460    ; *
        MOV (SP)+,R1       ; *
        MOV #5020,R2       ; *
        EMT 10             ; *
        DEC R1             ; *
14:     MOVB #40,(R1)+     ; *
        DEC R2             ; *
        BPL 14             ; *
15:     MOV #320,R1
        EMT 36
        CLR R2             ;Вопрос "Повторять?"
        MOV PC,R1
        ADD #22,R1
        EMT 20
        EMT 6
        CMP R0,#116
        BNE 15
        JMP @#100000       ;Выход
        .B:25.A:Еще копию?
        .E
A:      EMT 14             ;Программа начального
                           ;                запуска
        MOV #146404,@#34
        MOV #100274,@#4    ;<СТОП> заменен на Выход
        MOV #232,R0
        EMT 16
        INC R0
        EMT 16
        MOV #2000,SP
        HALT
        END
```


Для правильной работы BASCOD необходимо выполнить некоторые подготовительные действия:

1. Отлаженную Бейсик-программу привести в следующий вид:

DIM ... (все операторы DIM, которые есть в программе)

STOP

...

Все остальные операторы. Среди них не должно быть операторов DIM и END. Все операторы END, которые встречаются в тексте, надо заменить на STOP

END

Наличие STOP между описанием массивов и остальными операторами (перед самым первым, если нет массивов) и END самым последним в программе обязательно!

2. Задать с помощью CLEAR верхнюю границу доступной Бейсику памяти. В любом случае надо оставить свободным место, куда будет загружена BASCOD, но желательно установить границу как можно ближе к фактическому концу программы.

3. Выполнить команду RUN. При этом возможно сообщение об ошибке 7 (переполнение памяти). Тогда следует увеличить в CLEAR верхнюю границу и повторить запуск. Если же памяти хватило, транслятор сформирует «шитый» код и начнет выполнение программы, но сразу остановится на операторе STOP в начале текста.

Теперь можно загружать и запускать BASCOD, например, так:

```
BLOAD "BASKOD".R.&037300
```

Если все сделано правильно, на экране появится вопрос «ИМЯ?». Нужно ввести имя для файла с «шитым» кодом и записать на

магнитофон нужное число копий. В дальнейшем этот файл можно будет загружать в мониторе командой M или M1000 и запускать командой S или S1000. Разумеется, запуск будет удачным, только если ПЗУ с Бейсиком не отключено.

Если же что-то не получилось, стоит проверить, правильно ли выполнены подготовительные действия.

Преобразование Бейсик-программ не исключает использования подпрограмм в машинных кодах, но для компоновки «шитого» кода и подпрограмм в один файл потребуется отладчик. Размещать подпрограммы в кодах можно не только после Бейсик-программы, но и на том месте, где раньше располагался исходный текст. Эта область заполнена кодом 377_8, и ее начало находится где-то после текстовых констант, начинающихся с адреса 3062_8. Операторы DEF USR для таких подпрограмм должны располагаться до первого STOP вместе с описаниями массивов. Для компоновки «шитого» кода и подпрограмм следует в любом отладчике загрузить преобразованную Бейсик-программу, найти начало свободной области, разместить в ней подпрограммы в кодах, занести для каждой из них в ячейку с адресом 2100_8+ +⟨удвоенный номер соответствующей функции USR⟩ адрес подпрограммы в памяти и записать файл на магнитофон.

Надеюсь, что программа BASCOD будет полезна многим пользователям, работающим с вильнюсским Бейсиком-86.

М. ЕЖОВ

СМ 6337+БК

Работающим на отечественных IBM-совместимых ПЭВМ хорошо известен принтер СМ 6337. Чтобы подключить его к БК, необходимо немногое — соединительный кабель и программное обеспечение.

Контакты разъема УП БК	Обозначение сигнала	Наименование сигнала	Контакты разъема ИРПР-М СМ 6337
A16	ВД00	D0	3
A13	ВД01	D1	5
B12	ВД02	D2	7
B10	ВД03	D3	9
B5	ВД04	D4	11
B7	ВД05	D5	13
B6	ВД06	D6	15
A7	ВД07	D7	17
A28	ВД08	STROBE	1
A23	ВВ08	BUSY	21
B29	ВВ14	PE	23
A30	ВВ15	ERROR	28
A11	Общий		2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24

Схема кабеля приводится в таблице. Некоторые сигналы принтера, например ACKNLG, не задействованы — они не нужны при работе. К БК подсоединены линии PE и ERROR (сигналы об ошибках), однако в большей части своих программ эти сигналы я не использую — для уменьшения объема программы, да и нет в них особой необходимости.

На СМ 6337 необходимо установить тип интерфейса ИРПР-М, т. е. поставить переключатель S5.1 в положение «0» («на себя»).

К сожалению, интерфейс принтера не совместим со стандартным, каковым обычно считают интерфейс, необходимый для работы с подпрограммой печати, зашитой в ПЗУ Бейсика. Однако в целом интерфейсы очень схожи (используются лишь различные уровни сигналов), поэтому несложно доработать любую программу, имеющую в своем составе драйвер принтера, для работы с СМ 6337. Мной были доработаны такие программы, как ОТЛАДЧИК.9, BRED и др. Доработанные версии получили соответственно названия ОТЛАДЧИК.9П, BRED.6337 и т. д. Доработка заключалась в замене «родного» драйвера принтера на разработанный мной драйвер 6337. Это достаточно просто. Хуже, если заменить драйвер невозможно, например, когда программа находится в ПЗУ (в первую очередь это относится к Бейсику). Для этого случая мной написаны специальные программы.

Существует четыре подхода к написанию таких программ — аппаратный, программно-аппаратный и два чисто программных. Расскажу лишь о программных. Первый — «поимка» вызова драйвера, очень медленный, а главное, порождающий очень объемные программы метод. Второй — написание подпрограмм в кодах, которые будут вызываться Бейсиком. Использовать их ненамного более сложно, чем оператор LPRINT, они

невелики и быстро работают. Пользовательский интерфейс навеян работой на ДВК-2 в среде BASIC-OSDVK. Вывод программы производится оператором записи SAVE вот так:

```
SAVE "LP:"
```

А подпрограмма печати перехватывает обращение к магнитофону и выводит «записываемое» на принтер.

Приведу одну из самых простых программ, работающих с 6337.

```
      MOV  #177714,R4 Адрес порта ввода-вывода
BEG:  MOV  '>',R0     Вывод ">"
      EMT  16
      MOV  PC,R1      Вычисление адреса буфера
      ADD  (PC)+,R1    текста, расположенного
      .@BUF+2          сразу после программы
      MOV  R1,R0      Ввод строки (до 63 симв.)
      MOV  #5077,R2
      EMT  16
1:    MOV  #400,(R4)  Сброс STROBE
      MOVB (R0)+,R3   Выделение следующего
                                 символа строки
      BIC  #177400,R3 Выделение младшего байта
      COMB R3         Инверсия передаваемого
                                 байта
      MOV  R3,(R4)    Передача на принтер
2:    BIT  #400,(R4)  Проверка сигнала BUSY
      BEQ  2           (ожидание готовности
                                 принтера)
      CMP  R0,R1      Конец строки?
      BNE  1
      BR   BEG        Возврат к началу программы
BUF:  END             Начало буфера текста
```

В программное обеспечение принтера входят как уже упомянутые доработки известных программ, так и оригинальные разработки — печать текстов различных форматов (на ассемблере, Бейсике, Фокале), печать дампов, твердой копии экрана, подпрограммы для работы в Бейсике, многие другие.

Заинтересовавшиеся могут обратиться по адресу: 432012, Ульяновск, пр. Гая, 23а, 44. Тел. 37-10-57.

П. НИКОНОВ

Каталожная информационно-поисковая система

Профессиональные пользователи БК всю основную информацию об имеющихся у них программах хранят обычно в так называемых каталожных файлах, которые создаются с помощью программ CATALOG, CATALOGM, VERIFY2, VERIFY+ и т. п. Однако разрозненность этих файлов (свой для каждой кассеты) мешает систематизации каталога в целом.

Восполнить этот пробел призвана описываемая ниже система, позволяющая создать единый сводный каталог из множества каталожных файлов отдельных кассет и производить с ним необходимые манипуляции: редактировать, сортировать, искать нужную запись и пр.

Для каждого фиксируемого в архиве файла системой сохраняются и обрабатываются параметры, перечисленные в таблице.

Параметр	Значение
Имя файла	16 символов
Адрес загрузки	0—177777(8)
Длина файла	0—177777(8)
Контрольная сумма	0—177777(8)
Номер кассеты	0—99
Сторона кассеты	A, B
Номер записи на стороне	0—99

Система единовременно позволяет хранить в памяти БК параметры более 1000 (!) файлов без обмена с магнитофоном.

Ядром системы является файл CAT, к которому по мере надобности могут подзагружаться файлы CAT.LOAD, CAT.EDIT, CAT.SORT и CAT.FIND. CAT обеспечивает:

просмотр списка (клавишами **ВВЕРХ, ВНИЗ**);

параллельную просмотру печать списка;

запись списка в просмотренном и исправленном виде обратно на МЛ;

дозагрузку остальных частей системы.

CAT.LOAD обеспечивает подстыковку файлов в формате VERIFY+ к уже имеющимся в памяти БК по запросу имени файла. При успешном чтении запрос повторяется. По клавише **СТОП** можно выйти в монитор системы.

CAT.EDIT обеспечивает процесс редактирования одной строки списка, перемещение по списку, вставку пустой строки, удаление целой строки.

CAT.SORT обеспечивает сортировку каталога по всем параметрам (как в неубывающем, так и в невозрастающем порядке), причем при сортировке по именам русский алфавит отсортировывается в естественном порядке.

CAT.FIND обеспечивает поиск по модели

с возможностью печати найденных файлов на принтере. Это, пожалуй, самый важный файл системы. В режиме редактора строки вводится модель поиска (для всех или для части параметров). Она может включать инверсные символы «?» и «*» (они, в отличие от обычных символов «?» и «*», вводятся клавишами СТРЕЛКА ВВЕРХ и СТРЕЛКА ВНИЗ): первый заменяет любой символ, второй — любую (в частности, «нулевую») группу символов. Например, модель *С* задает поиск всех имен, содержащих символ С, модель ??С* — имен, в которых символ С стоит в третьей позиции.

А. ЛОБАНОВ

Форматировка результатов вычислений в Бейсике БК

Результаты вычислений в Бейсике получаются зачастую не только с чрезмерным числом знаков, но и с ошибками. Например, результатом сложения 76 и 0.14 будет 76.140000000000001, а деление 8912 на 100 даст 89.11999999999999. Оператор же формата представления числа при выводе отсутствует.

Использование функции DEF FNA(X)==INT(X*Q+.5)/Q, стандартно применяемой для форматировки результатов вычислений с заданной точностью Q, не позволяет преодолеть отмеченные недостатки.

Между тем перевод результата вычислений P в строковую форму и дальнейшая работа с ней позволяют достаточно легко организовать процедуру форматировки представления числа без округления.

В строке 60 приведенной программы определена пользовательская функция, переводящая результаты вычислений в «строковый образ» FNP$(P), корректирующая «шалости» функции MID$() и успешно справляющаяся с такими каверзными числами, как 2.9999999999999. Многозадачность этой функции делает ее внешний вид весьма эффектным.

Как бы то ни было, ее применение полностью оправдано там, где удобство представления чисел важнее их достоверности, например при организации таблиц, корреляционных последовательностей и т. п.

Формат представления чисел в данной программе определяется параметром Q; число нулей в нем устанавливает число цифр, сохраняемых после запятой при печати числа.

Особую роль играют функции STRING$ в строке 160: они дополняют столбцы таблицы до требуемой ширины и в конечном счете формируют все поле таблицы.

В программе предусмотрено выравнивание чисел в графе по правому краю; для выравнивания по левому краю достаточно поменять местами функции STRING$ и FNP$ в строке 160. Уменьшаемое в первом аргументе функции STRING$ (в нашей программе 12) задает ширину графы.

Недостатком предлагаемого способа является медленная работа пользовательской функции. Но, как говорится, за неимением гербовой...


```
1 ' ФОРМАТИРОВКА (БЕЗ ОКРУГЛЕНИЯ!)
        РЕЗУЛЬТАТОВ ВЫЧИСЛЕНИЯ ПЕРВОГО
         ДЕСЯТКА СТЕПЕНЕЙ ЧИСЕЛ пи, e,
                              1n пи, 1n 10
10 CLS
20 INPUT "ТОЧНОСТЬ";Q
30 ?
40 ?
50 ?
60 DEF FNP$(P)=""+
      MID$(STR$(FIX(P+SGN(P)*5D-7)),1,
     LEN(STR$(FIX(P+SGN(P)*5D-7)))-1)+
 "."+MID$(STR$(FIX(Q*(ABS(P+SGN(P)*5D-
         7-FIX(P+SGN(P)*5D-7))+1))),3,
      LEN(STR$(FIX(Q*(ABS(P+SGN(P)*5D-
        7-FIX(P+SGN(P)*5D-7))+1))))-3)
70 A=1
80 B=1
90 C=1
100 D=1
110 FOR K=1 TO 10
120 A=A*PI
130 B=B*EXP(1)
140 C=C*LOG(PI)
150 D=D*LOG(10)
160 ? STRING$(12-LEN(FNP$(A)),32);
  FNP$(A);STRING$(12-LEN(FNP$(B)),32);
  FNP$(B);STRING$(12-LEN(FNP$(C)),32);
  FNP$(C);STRING$(12-LEN(FNP$(D)),32);
                                FNP$(D)
170 NEXT K
180 END
```

А. ГУСЕЙНОВ

Н. ШЕРПАЕВ
г. Брянск

Демонстрация вычислений

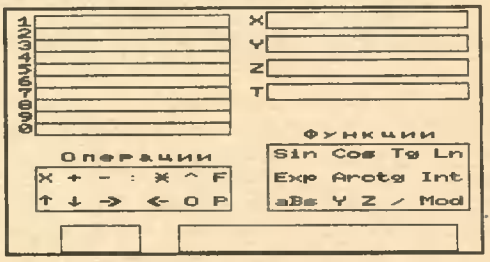
В статье предлагается для обсуждения программа «Стековый микрокалькулятор», которая наглядно отображает изменение состояния регистров стековой и адресуемой (неподвижной) памяти микрокалькулятора при выполнении вычислений. Она может использоваться непосредственно для вычислений, для обучения учащихся работе с программируемым микрокалькулятором в автоматическом режиме, а также при изучении демонстрационных и инструментальных программ.

Программа написана на Бейсике для «Ямахи»; имеется вариант программы для УКНЦ, возможен перевод программы для любого персонального компьютера школьного типа.

При запуске программы на экране компьютера изображаются в виде прямоугольников индикаторы для ячеек стековой памяти X, Y, Z, T и для ячеек адресуемых регистров памяти: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0. Под ними указывается список возможных операций и функций. В нижней части экрана выделены два «окошка» для индикации названия выполняемой операции или функции (в ее символьном изображении) и числового результата вычислений. Этот дополнительный регистр памяти служит для записи значений вычисляемых операций и функций перед их передачей в регистр X.

Модель микрокалькулятора предполагает выполнение следующих операций:

X — включение микрокалькулятора, очистка X;
+ — сложение;
— — вычитание;
: — деление;
× — умножение;
∧ — возведение в степень;
F — пересылка результата выполнения операции или функции в регистр X;
0 — циклический сдвиг в стеке;
P — вызов предыдущего значения;



<вверх> — сдвиг вверх в стеке (T→Z→→Y→X);
<вниз> — свдиг вниз в стеке (X→Y→→Z→T);
<влево>
<цифра> — пересылка содержимого регистра X в указанную ячейку;
<вправо>
<цифра> — пересылка содержимого указанной ячейки в регистр X.

Операндами для выполнения арифметических операций являются значения числовых величин, занесенные в регистры X и Y.

Вычисление значений функции осуществляется при нажатии на клавишу, указанную заглавной буквой в названии функции на экране компьютера. Аргументом функции является числовое значение из регистра X.

Изменение состояния стека при вычислениях происходит по правилам, используемым в программируемом микрокалькуляторе.

Изучению модели микрокалькулятора можно посвятить уроки следующего содержания: стековая память, числа с плавающей запятой и функции, адресуемые регистры.

Сначала учащиеся знакомятся с правилами выполнения простейших вычислений.

Для имитации включения модели микрокалькулятора следует нажать клавишу <X>, в противном случае нажатия на другие клавиши не воспринимаются. При нажатии клавиши <X> прямоугольник, изображающий

регистр памяти X (индикатор), окрашивается в светло-зеленый цвет.

Ввод числа производится нажатием цифровых клавиш, знака числа и десятичной точки. Ошибочное нажатие клавиш не воспринимается.

Для ввода числа 3,14 следует нажать клавиши <3>, <·>, <1>, <4>. После ввода указанных символов воспринимаются лишь цифровые клавиши, в частности не воспринимается нажатие клавиш <—> или <·>.

Для вычисления значения числового выражения (2,3—0,8) /0,5 следует нажать клавиши: <0>, <·>, <8>, <вниз>, <X>, <2>, <·>, <3>, <—>, <F>, <вниз>, <X>, <0>, <·>, <5>, <Y>, <:>, <F>. При этом состояние стека меняется согласно таблице

Клавиши <0>,<.>,	<8>,<вниз>,<X>,<2>,<.>,	<3>,<->
X	0 0. 0.8	0.8 2 2. 2.3 2.3.
Y	0.8	0.8 0.8 0.8 0.8 0.8 0.8

Клавиши <F>,<вниз>,<X>,<0>,<.>,<5>,<Y>,<:>,<F>									
X	1.5	1.5		0	0.	0.5	1.5	1.5	3
Y		1.5	1.5	1.5	1.5	1.5	0.5	0.5	0

Учитель без особого труда может подобрать для учащихся примеры, аналогичные рассмотренным.

На следующих уроках учащиеся знакомятся с представлением чисел в форме с плавающей точкой, т. е. в показательной форме, а также с вычислением значений элементарных функций. На этих же уроках рассматривается схема изменения состояния стека при выполнении операции возведения в степень.

Для обсуждения с учащимися можно выбрать примеры вычисления значений выражений $30 \wedge 10$ и $50 \wedge 10+\exp(50)$.

Приведем таблицы состояния стека при выполнении этих примеров.

Таблица вычисления 30^10

Клавиши <1>,<0>,<вниз>,<X>,<3>,<0>,<^>,<F>							
X	1	10	10	3	30	30	5.9049E+14
Y			10	10	10	10	10

Таблица вычисления 50^10+exp(50)

Клавиши <1>,<0>,<вниз>,<X>,<5>,<0>,<^>,<F>							
X	1	10	10	5	50	50	9.765625E+16
Y			10	10	10	10	10
Z							

Клавиши <вниз>,	<X>,	<5>
X	9.765625E+16	5
Y	9.765625E+16	9.765625E+16 9.765625E+16
Z	10	10 10

Клавиши <0>,	<E>,	<F>
X	50	50 5.18470552E+21
Y	9.765625E+16	9.765625E+16 9.765625E+16
Z	10	10 10

Клавиши <+>,	<F>
X	5.18470552E+21 5.18470552E+21
Y	9.765625E+16 10
Z	10 10

Затем учащиеся могут перейти к самостоятельному вычислению значений следующих выражений:

1. $8 \wedge 8+\sin(\ln(15 \wedge 15))-5.5$;
2. $\operatorname{arctg}(\lg(4)+12 \wedge 18)$;
3. $(10 \wedge 1.7-\exp(33)) \wedge 2$;
4. $(52 \wedge 8-\sin(6.8)) \wedge 3$;
5. $(1/8-\lg(5)) : (\sqrt{7}+6 \wedge 12)$;
6. $13 \wedge 15-6\cos(15)$;
7. $(\ln(5)-\operatorname{arctg}(0.001))\sqrt{(15)}$;
8. $(5-10 \wedge 25) \wedge 2+\arccos(0.5)$.

Далее учащиеся переходят к решению примеров с использованием адресуемых регистров памяти. Передача числа из регистра X в любой из адресуемых регистров осуществляется нажатием клавиши <влево> и цифровой клавиши, соответствующей номеру адресуемого регистра. Аналогично вызов числа из адресуемого регистра в регистр X осуществляется нажатием клавиши <вправо> и соответствующей цифровой клавиши.

Например, передача числа 13 в ячейку 0 осуществляется нажатием клавиш: <1>, <3>, <влево>, <0>. После нажатия указанных клавиш число 13 будет изображено в ячейках 0 и X. Очистка регистра X осуществляется нажатием клавиши X. Далее вызов числа 13 из ячейки 0 на индикатор X осуществляется нажатием клавиш: <вправо>, <0>.

Использование адресуемых регистров можно рассмотреть с учащимися на примере вычисления значения числового выражения:

$(1.35 \wedge 2+5.31 \wedge 2) : (1.35 \wedge 2-5.31 \wedge 2)$.

Для вычисления его значения можно использовать последовательность клавиш: <2>, <вниз>, <X>, <1>, <·>, <3>, <5>, <^>, <F>, <влево>, <1>, <X>, <5>, <·>, <3>, <1>, <^>, <F>, <влево>, <2>, <вниз>, <X>, <вправо>, <1>, <+>, <влево>, <0>, <X>, <вправо>, <2>, <вниз>, <вправо>, <1>, <—>, <вниз>, <вправо>, <0>, <:>, <F>.

Желательно обсудить с учащимися способ решения этого примера без использования адресуемых регистров.

Далее можно перейти к более сложным вычислениям, требующим использования как стековой памяти, так и адресуемых регистров.

Е. ЧУБРОВ, Н. СЛИВИНА, А. ДЕМУШКИН

Компьютер и изучение математики

Проводимые в различных учебных заведениях эксперименты позволяют определить, в каких ситуациях компьютер может расширить возможности педагога, подсказать новые методические подходы. Исследования, проводимые в школе-лицее № 1027 при Московском энергетическом институте совместно с кафедрой высшей математики этого вуза, позволили опробовать различные формы и методы применения ЭВМ при изучении математики в школе. Мы развиваем три программно-методических направления.

Путеводитель. Идея этого средства основана на том, что компьютерная программа может обладать бóльшими выразительными возможностями по сравнению с книгой и представляет более удобный доступ к информации. Путеводитель построен так, что предъявляет информацию в максимально наглядной, выразительной форме. Сегодня реализована первая глава — планиметрия. Каждое понятие (а в этой главе их около 150) представлено яркой, красочной картинкой, которая, используя минимум слов, формул и обозначений, вводит в суть понятия. Пользуясь меню, учащийся может получить дополнительные справки (определения, формулировки и т. п.). С помощью того же меню можно перейти к другим понятиям, которые логически связаны с данным.

Следует понимать, что путеводитель не призван заменить книгу-справочник. Он не претендует на абсолютную полноту, а предназначен в первую очередь тем учащимся, которые слабо владеют основами курса, не сформировали смысловые связи между понятиями, испытывают трудности в построении чертежа, адекватного условиям задачи. В то же время в путеводитель включена информация, выходящая за рамки стандартного курса, а также различные любопытные сведения.

Ведется работа над следующими главами путеводителя — стереометрией, тригонометрией, алгеброй.

Тренажеры. Программы этого типа помогают учиться решать задачи. В первую очередь это задачи стандартного типа, метод решения которых может быть удовлетворительно алгоритмизирован. В то же время тренажер не ограничивает учащегося единственным способом решения, а обеспечивает ему значительную свободу поиска пути к правильному ответу. Одновременно компьютер предоставляет дополнительные средства: теоретические справки, таблицы и т. п. Иными словами, компьютер берет на себя второстепенные действия, чтобы они не отвлекали ученика от идеи решения задачи.

Тренажер, как правило, имеет три режима. В режиме справки ученик может найти сводку необходимых теоретических сведений, а также посмотреть «эталонное» решение характерных примеров. В режиме репетитора учащийся действует сам, а компьютер играет роль терпеливого, внимательного педагога. Он контролирует каждый шаг решения, разъясняя ошибки и предлагая советы, а затем анализирует окончательный ответ. Наконец, в режиме самостоятельной работы ученику предоставляется максимальная свобода. Здесь компьютер просто организует среду, работая в которой удобно решать задачу.

Например, тренажер графики помогает научиться строить графики довольно сложных функций методом элементарных преобразований (сдвиг, сжатие/растяжение, отражение). Учащийся выбирает исходный график из широкого набора базовых функций (x^2, $\sin x$, 2^x, ...), а затем указывает машине, какие преобразования следует выполнять, чтобы построить график заданной ему функции. В любой момент он может посмотреть формулировки нужных правил (как построить графики $f(x\pm a)$, $f(x)\pm a$, $f(k\cdot x)$, $k\cdot f(x)$, $f(|x|)$, $|f(x)|$, исходя из графика $f(x)$), а также примеры решения задач. Учащийся может по своему желанию менять область координатной плоскости, изображенную на экране. Более того, проделав ряд преобразований над исходным графиком, он может попросить машину выписать уравнения тех кривых, которые он в действительности построил. Это помогает ученику выявить ошибку и понять связь между преобразованием формулы и трансформацией графика.

Анализ курса элементарной математики позволил выделить несколько десятков тем, при изучении которых тренажер необходим или желателен.

Математический эксперимент. Эксперимент, проведенный собственными руками, является незаменимым методом на уроках физики, химии, биологии. А возможно ли что-то подобное при изучении математики? Наш опыт показал, что экспериментальный подход стал возможным и нужным после того, как появился компьютер — инструмент проведения эксперимента.

Что такое эксперимент в математике? Это в первую очередь вычисления и построение графиков. Если математический

объект уже реализован в компьютере, иными словами, если мы имеем программу, которая готова немедленно посчитать то, что нам нужно, или построить график той кривой, уравнение которой мы задали, то у нас есть инструмент исследования. Подобная программа была разработана, она получила название «Формула».

Осталось только помочь учащемуся организовать эксперимент. Для этого разрабатываются указания, которых мы называем компьютерными уроками. Такой урок — это серия заданий, каждое из которые выполняется легко и быстро. Например, изучаются свойства логарифмической функции. Задание может быть таким:

Постройте график y=log x. Пересекает ли график ось ординат? Укажите множество x, при которых функция log x определена. При каких x функция пересекает ось абсцисс? Лежит выше (ниже) этой оси? Является ли функция монотонной?

Это только одно задание. Оно выполняется мгновенно, так как учащийся легко получает ответы, рассматривая изображение на экране. Цель вопросов — обратить взгляд ученика на определенную часть графика и подтолкнуть его к правильной интерпретации того, что он увидел. Данный урок включает двадцать одно задание, которые можно выполнить за 45 мин. Проведя «направленную серию экспериментов», учащийся в состоянии самостоятельно прийти к ответу на контрольные вопросы — обычные вопросы из учебника, например: «Сравните a и b, если $\log_{1/2}a > \log_{1/2}b$».

В настоящее время разработано 20 подобных уроков, работа над ними продолжается. Тот же метод мы применяем в вузе при обучении студентов математическому анализу. «Формула» очень полезна также при изучении информатики и методов вычислений. С ее помощью удается отделить изучение численного алгоритма от программистской рутины, которая его обычно сопровождает. Тем самым внимание студента концентрируется на существе метода.

Обратите внимание, что в отличие от «классических» обучающих программ недавнего прошлого, строившихся по принципу: кадр информации — вопрос машины — ответ обучаемого — переход к следующему кадру, описанные системы представляют собой среду, в которой активной, ведущей диалог стороной является ученик.

Если вас заинтересуют наши разработки, обращайтесь по адресу: 111578, Москва, ул. Молостовых, 10а, школа-лицей № 1027, научно-производственный центр «МЭЛС», тел. 300-15-31, 945-92-16.

В. ПИНАЕВ

Ханойские перемешанные башни

Практически ни одна книга по программированию не обходится без упоминания о классической задаче «Ханойские башни». Давайте перечитаем условие этой головоломки: «Имеются три вертикальных стержня на подставке; на одном из них — пирамида из кружков разного диаметра. Требуется переложить кружки, сохранив их порядок. При этом не разрешается больший кружок класть на меньший».

А что если убрать из условия задачи выделенное слово **пирамида?** Тогда можно будет разместить кружки на первом стержне в произвольном порядке. Сохраним прежнее условие игры: нельзя класть большой кружок на маленький. Цель игры: восстановить правильный порядок кружков на одном из трех стержней (т. е. уложить кружки сверху вниз в порядке возрастания диаметров).

То, что в исходном положении порядок нарушен, не должно нас смущать — ведь это сделано до начала игры. А далее мы будем строго придерживаться правил. Кстати, перемешивание — это начальный этап многих игр (сравните: кубик Рубика, игра в пятнадцать, всевозможные карточные игры и тому подобное). Назовем такую перемешанную башню n-башней. А разрешима ли головоломка для n-башни?

Прежде разберемся с более простой задачей. Пусть требуется собрать пирамиду из кружков, нанизанных на два стержня, при свободном третьем. Причем кружки расположены по возрастанию диаметров, но с возможными пропусками. Например, на первом стержне A расположены кружки (в порядке сверху вниз): 1, 3, 4, 6, а на втором стержне B — 2, 5, 7, 8 (циф-

ры соответствуют размерам кружков).

Предположим, что самый крупный N-й кружок находится на стержне A. Тогда, для того чтобы собрать пирамиду на стержне C, необходимо:

1) собрать на стержне B пирамиду без N-го кружка;

2) переместить N-й кружок на пирамиду C;

3) переместить пирамиду из $(N-1)$ кружка с B на C.

Сформулированное правило дает нам рекурсивную процедуру сборки пирамиды из двух упорядоченных частей. (Вообще говоря, следует добавить еще одно правило для случая, когда имеется всего один кружок. Но это совсем просто: перемещаем этот кружок на нужный стержень.) Заметим, что действие 1 — это та же задача, но с меньшим количеством кружков. Действие 3 — это перемещение обычной Ханойской башни по известному алгоритму.

Теперь вернемся к нашей n-башне. На стержне A находятся кружки в произвольном порядке. Стержни B и C пока пусты. Постараемся рассортировать кружки со стержня A на стержни B и C в виде двух упорядоченных частей, из которых соберем нашу пирамиду. Для этого будем перекладывать со стержня A кружки на B и C, пока это возможно. При этом мы переложим не менее двух кружков. Если нам повезет, то мы переложим все кружки с A на B и C и получим две упорядоченные части. А как действовать в таком случае, мы уже знаем.

Пример. На стержне A находятся кружки 4, 8, 3, 1, 5, 6, 2, 7. Последовательно перемещаем кружки: 4 — на стержень B, 8 — на C, 3 — на B, 1 — на B, 5 — на C.

Пусть очередной кружок диаметра D нельзя положить сверху ни на B, ни на C. Найдем место для вставки этого кружка на стержнях B и C. Проведенная условная граница отделит кружки диаметра, меньшего D. Вернемся к нашему примеру: на стержне A находятся кружки в порядке сверху вниз 6, 2, 7, на B — 1, 3, 4, на C — 5, 8. Сейчас для верхнего кружка 6 пирамиды A отделяем условно на пирамиде B кружки 1, 3, 4, а на пирамиде C отделяем кружок 5. Мы выделили две упорядоченные части на B и C. По сформулированному выше правилу «соединим» их в одну пирамиду на стержне B. После этого можно беспрепятственно переместить кружок с диаметром D на стержень C. Действуя таким образом, можно «рассортировать» n-башню на две упорядоченные части. А собирать их мы уже умеем.

Таким образом, мы сконструировали алгоритм «воссоздания» пирамиды по n-башне. Возможна и другая расстановка для нашей игры. Перемешанные кружки разделим на три произвольные кучки, а затем кружки каждой нанижем в произвольном порядке на соответствующие стержни. Цель игры и правила неизменны. Оказывается, и в этом случае головоломка всегда имеет решение. Дальнейшим развитием может быть многомерная Ханойская башня. Но, впрочем, здесь уже пропадает интригующая простота и изящность Ханойской башни.

В. ОЧКОВ
Московский энергетический институт

Среда программирования QBasic

Есть у меня шестерка слуг
Проворных, удалых.
И всё, что вижу я вокруг,
Всё узнаю от них.
Они по зову моему
Являются в беде.
There names are What,
and Why, and When, and
How, and Where, and Who.

Эпиграф статьи — стихотворение из сказки о длинном хоботе любопытного слоненка, которое автор когда-то учил на школьных уроках английского, запоминая слова «что — what», «почему — why» «когда — when», «как — how», «где — where» и «кто — who», и от которого в памяти осталось лишь переводное маршаковское начало и оригинальный киплинговский конец. Здесь для нас важны как форма, так и содержание.

Сначала о форме. Текст статьи, как и эпиграф, двуязычен — очень часто в нем встречаются английские термины посреди русских фраз. Увы, хотя гонения на кибернетику и стали историей, отставание в вычислительной технике — реальность сегодняшнего дня, и компьютеры долго еще не станут говорить на чистом русском.

Теперь о содержании эпиграфа.

Читатель, конечно, знаком с языками программирования (GW-BASIC, Паскаль, Фортран, Рапира, коды микрокалькулятора и т. п.). Здесь же речь пойдет не о языке, а о среде программирования. Это означает, что программист, сидящий за дисплеем, не остается наедине с бесчисленными операторами, командами и служебными словами языка. Ему помогает «шестерка слуг проворных, удалых». Вот их имена в среде QBasic: Файл, Редактирование, Просмотр, Поиск, Запуск, Отладка, Параметры и Справка.

Включив IBM PC с дисковой операционной системой MS-DOS пятой версии и запустив файл QBASIC.EXE, вы войдете в программную среду QBasic, признаком чего будет цепочка слов в верхней части дисплея.

Теперь можно обычным образом, нажав клавишу Alt, с помощью клавиш-стрелок выбрать из этого меню нужный пункт-слово и нажать клавишу Enter (традиционно используемую для таких целей в программных средах фирмы Microsoft). Можно, впрочем, пользоваться и мышью. Из выбранного слова выпадет колонка с именами новых «слуг» — вспомогательное меню, слова которого, в свою очередь, могут оказаться заголовками новых меню и т. д. Многоточие за названием «блюда» свидетельствует о том, что это не команда, а название нового меню.

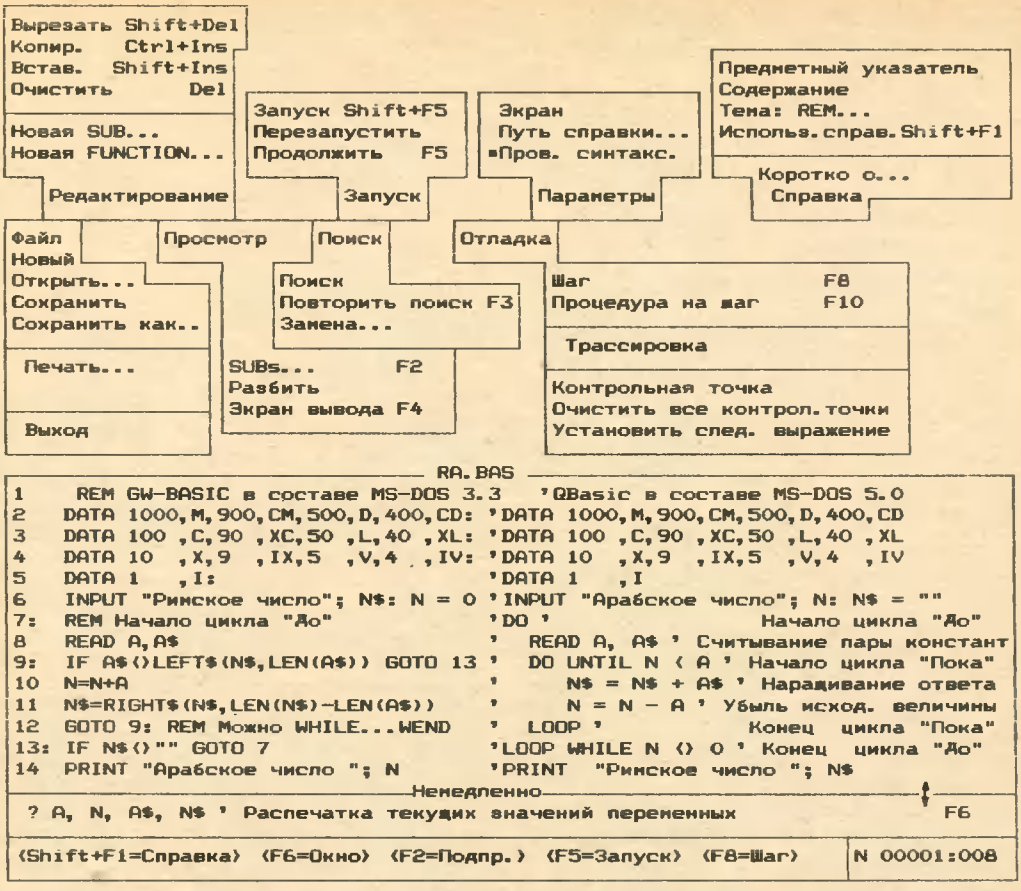
Чтобы дать представление о возможностях среды QBasic, на иллюстрации «проявлены» вспомогательные меню; для экономии места они развернуты не только вниз от главного меню, но и вверх. Рассказ о них и составит сердцевину нашего повествования.

Файл. Команды, собранные здесь,— окно среды QBasic в окружающий мир; они осуществляют связь компьютера с дисководами для жестких и гибких дисков, с принтером, с DOS. Режим «Новый» очищает ОЗУ от старой программы. Новая программа после этого может быть введена двумя способами: набором текста на клавиатуре или вызовом с диска. Чтобы считать программу с диска, нужно открыть программы («Открыть...») и убедиться, что в появившемся списке есть ее имя; если его нет — сменить диск или каталог на нем.

Но вот имя найдено. На него наводится курсор, нажимается клавиша Enter — и программа появляется на экране дисплея.

Режим «Сохранить» аналогичен оператору SAVE в обычном Бейсике. Не надеясь на памятливость человека, QBasic зорко следит за тем, чтобы, в архивной памяти оставалась копия программы, хранившейся в ОЗУ, и, если оператор захочет очистить ОЗУ командой «Новый» или «Выход», напомнит ему о необратимости такой операции и предложит записать программу на диск.

Режим «Печать...» распечатывает на прин-



тере программу (всю, только подпрограммы или только то, что выделено на экране дисплея) или информацию из окна непосредственного режима.

«Выход» — это прощание с QBasic и возврат в DOS.

Редактирование. Посмотрите на стол «некомпьютеризированного» редактора: там, конечно, лежат ножницы и клей. В арсенале есть и ножницы («Вырезать»), и клей («Вставить»), и копировальная машина («Копировать»). Чтобы из текста программы вырезать для копирования или переноса какой-то участок, его нужно сначала выделить цветом, нажимая клавиши перемещения курсора и притопив предварительно клавишу Shift. Затем выделенный текст можно из программы «Вырезать» и «Вставить» в любое место, а можно сделать штемпель («Копировать»), орудовать которым («Вставить») можно сколько угодно. Редактор среды QBasic, конечно, не такой мощный, как MS Word, но вполне приемлемый для работы с текстами, и не только программ. Автор статьи писал ее в среде QBasic, отключив предварительно контроль за синтаксисом строк — иначе машина оказывала бы медвежьи услуги, сообщая об «ошибке» (несоответствии правилам QBasic) после каждого перевода строки.

Просмотр. QBasic вооружен таким мощным средством построения программ, как разбивка их на процедуры и функции с именами и локальными переменными, что существенно отличает его от предшественника — языка GW-BASIC, входящего в состав MS-DOS 3.30. Об этой ценной особенности среды QBasic речь впереди. Сейчас отметим, что выполнение позиции меню «SUBs...» выводит на экран названия всех задействованных в программе процедур и функций, а также и самой программы. Если требуется отредактировать или просто просмотреть какую-то процедуру или функцию, то достаточно на ее имя навести курсор, нажать на клавишу Enter — и нужный участок программы перед глазами.

Команда «Разбить» разбивает экран с текстом программы на два окна, что позволяет редактировать программу, поглядывая в то же время на какую-то отдаленную ее часть. Перебрасывается курсор из одного

окна в другое аккордом Shift+F6.

Команда «Экран вывода» позволяет выйти из окна редактирования в окно с результатами работы программы и посмотреть, что она натворила. Нажатие на любую клавишу возвращает в окно редактирования.

Поиск. Здесь хранятся два очень нужных инструмента — «Поиск...» и «Замена...», позволяющие вести поиск и при необходимости заменять определенную цепочку символов на новую. Командой «Замена...», кстати, можно «вылечивать» Бейсик-программы, написанные на ранних версиях этого языка, от «детской болезни» — коротких (в два байта) и поэтому неудобных имен переменных: A, A0, T и т. д. Дал команду «Замени цепочку «T» на цепочку «Temperatura» — и QBasic за доли секунды сделает программу более понятной. Делая замену, система может переспрашивать: «В этом месте менять или пропустить?», чтобы застраховаться от, например, замены буквы T в служебных словах INPUT, PRINT.

Запуск. Дело вроде бы простое, но и тут возможны нюансы. Программу можно запустить с самого начала («Запуск»), продолжить ее выполнение («Продолжить») после прерывания аккордом Ctrl+Break или встречи с оператором STOP. Команда «Перезапустить» позволяет выполнять программу пошагово.

Отладка — один из самых ответственных моментов при создании новых программ и модернизации старых. Сам английский термин debugging (отладка) заслуживает того, чтобы о нем рассказать подробнее. A bug — по-английски жучок, а приставка «de» означает «удаление», так что термин можно перевести как «обезжучивание». По легенде, один из первых компьютеров стал барахлить из-за того, что в его электронных внутренностях запутался таракан, закоротил контакты, обуглился и напрочь замкнул их, внеся тем самым изменение в электронную схему машины — вот отсюда и произошло debugging.

Какие инструменты отладки есть у среды QBasic? Через режим «Установить следующее выражение» программа запускается с отмеченного курсором места. Можно указать операторы, на которых будет прерываться выполнение программы («Контрольная точка»). Все это, естественно, нужно будет убрать после отладки командой «Очистить все контрольные точки». «Трассировка» аналогична привычной команде TRON, но реализована более удобно — выполненные операторы отмечаются цветом букв и фона.

Параметры. Тут хранятся три инструмента настройки среды QBasic. «Экран» позволяет выбрать цвет фона, букв и т. д. «Путь справки...» указывает дорогу (D:\BASIC\QBASIC.HLP, например), на которой компьютер найдет файл со справочной информацией. «Проверка синтаксиса» заставит QBasic следить за правильностью синтаксиса программ. Если, например, программист напишет INPUT и переведет курсор на новую строку, то машина откажется работать дальше, напоминая, что за этим словом нужно приписать еще что-нибудь.

Справка. Режим помощи программисту — это сердцевина среды программирования QBasic. Как почти во всех программных продуктах, обращение к нему идет через нажатие клавиши-«выручалочки» F1. Но характер экранной подсказки будет зависеть от конкретного момента работы со средой QBasic. Если, например, программист набрал какое-то служебное слово (READ, COLOR, PLAY и т. д.) и забыл, какие параметры нужно за ним проставить, то после нажатия F1 на экране появится полное описание требуемого оператора и несколько типичных примеров его использования, один из которых можно будет перенести в текст программы, а затем отредактировать, приспособив к конкретному программному сюжету. Контекстная помощь устроена так, что после нажатия F1 на экран выдается информация по служебному слову или команде меню, намеченной к выполнению,— в зависимости от того, какое место отмечено курсором. На приведенном изображении экрана курсор находится в первой строке, на восьмой позиции (см. нижний правый угол рисунка — N 00001:008) программы, поэтому за словом «Тема» в меню «Справка» записано REM. Если сейчас программист нажмет F1, он узнает все об этом операторе.

Начинающим осваивать QBasic адресованы позиции «Предметный указатель» и «Содержание» в меню «Справка»: с их помощью можно найти и просмотреть описание любого раздела (структура программ, создание независимых модулей, синтаксис операторов и многое другое).

Теперь расскажем о собственно программировании в таких комфортных условиях. Первый пример — рядом с меню: справа программа, переводящая арабские числа в римские, написанная на QBasic, слева — переводящая римские числа в арабские, написанная на GW-BASIC.

QBasic не смотрит свысока на своего предшественника GW-BASIC. Если у вас есть библиотека программ на этом языке, то при переходе к QBasic она не пропадет. Возможны только небольшие заминки, связанные, например, с тем, что переменная IS допускаемая в GW-BASIC, в QBasic не может использоваться, так как в нем есть служебное слово IS («есть»).

Две особенности QBasic бросаются в глаза прежде всего.

Во-первых, строки в этом языке могут не нумероваться. Если в GW-BASIC перед введенной командой не было номера строки, она сразу выполнялась; в QBasic для непосредственного выполнения команд предусмотрено специальное окно «Немедленно». Конечно, при желании номера строк можно и ставить, и использовать в операторах перехода (GOTO, GOSUB) — эти возможности остались в основном для обеспечения совместимости со старыми версиями Бейсика.

Во-вторых, в QBasic можно обойтись не только без номеров строк (даже выполняющих по совместительству должность меток, например GOTO 9), но и без операторов условного и безусловного перехода. Для нелинейных алгоритмов предусмотрены специальные структурные управляющие конструкции.

Прежде всего это циклы «до» (итерация с постпроверкой) и «пока» (итерация с предпроверкой). Козьма Прутков в одном из своих афоризмов вопрошал: «Что полезнее — солнце или месяц?», сам же и отвечая: «Месяц. Ибо солнце светит днем, когда и без того светло, а месяц — ночью!» Аналогично можно ответить и на вопрос: «Что важнее — цикл «пока» или цикл «до»?» Цикл «пока», конечно: операторы тела цикла «до» выполняются как минимум один раз, что иногда приводит к ошибкам, операторы же цикла «пока» могут не сработать ни разу, что должно отвечать и задуманному алгоритму, и мудрой пословице «Семь раз отмерь (сначала проверь условие завершения циклических действий), а потом отрежь (выполни тело цикла)».

Синтаксис цикла в среде QBasic запомнить проще простого: делай (DO) до тех пор, пока (UNTIL или WHILE) не LOOP-нет терпение — пока не изменит свое значение логическое выражение, управляющее циклом.

```
DECLARE SUB F (X, Y)   '      См. внизу
INPUT "A,  B,  точность "; A, B, Eps
CALL F(A, Y1)'       Вызов процедуры F
DO WHILE B - A > Eps '  Начало цикла
   X = (A + B) / 2 '            Пополам
   CALL F(X, Y)' Расчет Y в середине
   IF Y1 * Y >= 0 THEN 'Альтернатива
      A = X '                Левое плечо
         ELSE B = X ' Правое плечо
   END IF '          Конец альтернативы
LOOP '                    Конец цикла
PRINT "Y = 0 при X = "; X '     Ответ
SUB F (Arg, Func) '        Заголовок
                             процедуры
  A = Arg ' Пересылка в локальных
                             переменных
  B = A * A '     Вторая локальная
            '                переменная
  Func = B - 3 '          ' F(X)=X*X-3
END SUB '              Конец процедуры
```

Программа поиска корня уравнения методом половинного деления содержит еще одну алгоритмическую конструкцию — полную альтернативу (IF...THEN...ELSE...END IF), а также крупное для работающих на «обычном» Бейсике новшество — локальные переменные: A и B в самой программе и A и B в процедуре с именем F расчета значения анализируемой функции — величины совершенно разные и независимые. «Вассал моего вассала не мой вассал» — девиз феодальной эпохи; «Переменная моей процедуры — не моя переменная» — девиз программ на развитых языках программирования.

Это свойство приближает QBasic к таким почтенным языкам, как QuickBASIC, Паскаль, Си, и позволяет программисту писать подпрограммы, не заботясь о разделении переменных. Этому же служит и механизм формальных параметров в заголовках процедур.

Но если функция пользователя объявлена через DEF FN, а не через DECLARE FUNCTION, ей доступны все переменные основной программы (в данном случае включая и Constant).

```
DEF FNY (X) '  Анализируемая функция
  FNY = X * X - Constant
END DEF
DEF FND (X) = 2 * X ' Ее производная
INPUT "X начальн., точность"; X, Eps
Constant = 3
DO ' Начало цикла
   X1 = X - FNY(X) / FND(X)
 IF ABS(X1 - X) < Eps THEN EXIT DO
   X = X1
LOOP ' Конец цикла
PRINT "Y=0  при X ="; X
```

В QBasic-программе поиска корня алгебраического уравнения методом Ньютона, приведенной выше, использован еще один цикл — с выходом из середины. Вот эту-то конструкцию (вопреки утверждению Дейкстры о том, что для реализации произвольного алгоритма необходимы конструкции следования, повторения и выбора) и следует считать необходимой и достаточной для создания любой программы. Ею можно моделировать любые циклы, ветвления, выборы; незаменимость же ее в том, что выходов из цикла может понадобиться несколько (если, например, значение логической переменной, управляющей циклом, меняется вне цикла и есть опасение пропустить ее изменение с «да» на «нет» в нужный момент; такое возможно, когда в программе задействованы элементы параллельного программирования или когда логическая переменная определяется показаниями встроенных часов компьютера или состоянием пе-

риферии), и в таком, например, сверхструктурированном языке, как Паскаль, выход из середины цикла реализуется либо с опорой на метку (а это признак низкой культуры программирования), либо через организацию процедуры. Паскалю не повезло!

Многие программисты считают необходимой для «приличного» языка возможность рекурсивного вызова процедуры. В старых версиях Бейсика рекурсия использовалась по принципу «если нельзя, но очень хочется, то можно» (см., например: ИНФО. 1991. № 4). В QBasic рекурсия разрешена; в качестве примера приведем решение традиционной задачи о Ханойских башнях.

```
DEFINT N ' Все переменные на N целые
DEFSTR X-Z '     Остальные - литерные
DECLARE SUB PUTDISK (N%,X$,Y$,Z$)
INPUT "Число дисков"; NØ
PUTDISK NØ, "A", "B", "C"
END

SUB PUTDISK (N%,X$,Y$,Z$)
DEFINT N: DEFSTR X-Z
IF N = 1 THEN
            PRINT X; "-"; Z
         ELSE
      PUTDISK N - 1, X, Z, Y
      PRINT X; "-"; Z
      PUTDISK N - 1, Y, X, Z
   END IF
END SUB
```

А на десерт — несколько насыщенных операторами выбора программ вычисления подоходного налога с граждан. Десерт будет под стать языку — богатым, из двух блюд.

Первое — вычисление налога с гражданина США. Что бросается в глаза? Во-первых, «ихняя» налоговая система нацелена на укрепление семьи — если супруги захотят развестись, то государство сразу накажет их рублем, точнее, долларом. Во-вторых, в США не дерут с налогоплательщика 60 % от заработка, превышающего определенную сумму; 33 % — вот тот предел, на который может отважиться государство, не опасаясь подавить активность своих кормильцев. В-третьих, при высоких доходах ставка налога снижается до 28 %: высокие доходы свидетельствуют о высокой деловой квалификации, такому человеку выгоднее доверить самому распоряжаться деньгами, чем отнимать их.

```
' Источник: David I. Schneider.  Mic-
' rosoft QuickBASIC: An  Introduction
' to Structured  Programming.  Dellen
' Publishing Company, San .Francisco,
' 1989.
DECLARE FUNCTION TaxMarried (adjPay)
DECLARE FUNCTION TaxSingle (adjPay)
INPUT "Грязные ($ в неделю) "; adjPay
PRINT "Налог с холостого  (незамуж-
      ней) = "; TaxSingle(adjPay); "$
PRINT "Налог с женатого  (замужней)
        = "; TaxMarried(adjPay); "$"
END

FUNCTION TaxMarried (adjPay)
' Расчет федерального налога с  жена-
'                          того (замужней)
 a = adjPay
 SELECT CASE a' Заголовок конструкции
              '                    "Выбор"
   CASE Ø TO 62: T = Ø
   CASE 62 TO 657: T = .15 * (a - 62)
   CASE 657 TO 1501: T = 89.25 +  .28
                                * (a - 657)
   CASE 1501 TO 3695: T  =  325.57  +
                       .33 * (a - 1501)
   CASE IS >= 3695: T = 1049.59 + .28
                        * (a - 3695)
 END SELECT'Конец конструкции "Выбор"
TaxMarried = T
END FUNCTION

FUNCTION TaxSingle (adjPay)
' Расчет федерального налога с холос-
'                          того (незамужней)
a = adjPay
SELECT CASE a
   CASE Ø TO 21: T = Ø
   CASE 21 TO 378: T = .15 * (a - 21)
   CASE 378 TO 885: T = 53.55 + .28 *
                              (a - 378)
   CASE 885 TO 2028: T = 195.51 + .33
                          * (a - 885)
   CASE IS >= 2028: T = 572.7 + .28 *
                              (a - 2028)
 END SELECT
TaxSingle = T
END FUNCTION
```

Второе — налог в СССР. Да, такой страны уже давно нет, но попытка угнаться за переменами в налоговом законодательстве без компьютера (а «ИНФО» готовится к печати «вручную» и отстает от времени минимум на три месяца) все равно обречена на неудачу, так что пусть программа имеет хотя бы историческую ценность, напоминая о былой весомости рубля.

«Некруглость» чисел в ее формулах не допускает скачков «налоговой» функции.

Подробнее о среде программирования QBasic, входящей в состав русифицированной версии MS-DOS 5.0, можно прочесть в следующих книгах:

1. *Кетков Ю. Л.* Языки GW-, Quick- и Turbo BASIC. М.: Финансы и статистика, 1992.

2. *Очков В. Ф., Пухначев Ю. В.* Уроки для пользователей IBM PC. М.: Финансы и статистика, 1992.

3. *Очков В. Ф.* Языки программирования GW-BASIC и QBasic: сравнительное описание. М.: Энергоатомиздат, 1992.

```
DECLARE SUB NALOG (g, nal)
 INPUT "Грязные (руб)"; g
 NALOG g, nal
 PRINT "Чистые ="; g - nal; "руб., налог ="; nal; "руб"
END

SUB NALOG (g, nal)
' Газета "Аргументы и факты", N 33 (514) за 1990 г.
 SELECT CASE g
  CASE 0 TO 101'                  Да здравствует профсоюз, запрещающий
  nal = 0'                      своим членам поступать на такую работу!
  CASE 101 TO 130'                                        И на такую тоже.
  nal = .29 * (g - 100)'
  CASE 131 TO 150 '                          А за такую нужно приплачивать
  nal = 8.7 + .3 * (g - 130) '                        гуманитарной помощью.
  CASE 151 TO 700'                    Как этот закон уживется с растущей
  nal = 14.7 + .13 * (g - 150)'                                 с инфляцией?
  CASE 701 TO 900'                          Это прожиточный минимум для
  nal = 86.2 + .15 * (g - 700)'                 уважающего себя мужчины,
  CASE 901 TO 1100'                   имеющего одну жену и одного ребенка.
  nal = 116.2 + .2 * (g - 900)'     "А то что ж: один в семи комнатах
  CASE 1101 TO 1300'          расселился, штанов у него 40 пар, а другой
  nal = 156.2 + .3 * (g - 1110) '    шляется, в сорных ящиках питание
  CASE 1301 TO 1500'                                        ищет", - ремарка
  nal = 216.2 + .4 * (g - 1300)'    Полиграфа Полиграфовича Шарикова.
  CASE 1501 TO 3000'                Они спутали заработанную плату со
  nal = 296.2 + .5 * (g - 1500)'                сверхприбылью монополии.
  CASE IS > 3000'                 Комментарий, не пропущенный в печать
  nal = 1046.2 + .6 * (g - 3000)'   внутренним цензором программиста
 END SELECT '       Это хорошо, что градации кончились на 60 %. А то я
END SUB'                  бы такое сказал. И цензор меня бы не удержал.
```


М. ВОРОНОВА

Вступительный экзамен по ОИВТ

Электронно-вычислительные машины все настойчивее входят в нашу повседневную жизнь, становясь неотъемлемой частью всех экономических и производственных процессов. В условиях контракта по найму работников все чаще встречается обязательное требование о владении как минимум навыками работы на компьютере. Не удивительно, что все серьезные учебные заведения все больше внимания уделяют подготовке специалистов всех профилей с такими навыками.

В связи с этим начиная с 1991 г. в порядке эксперимента в Государственной академии управления им. С. Орджоникидзе введен вступительный экзамен по основам информатики и вычислительной техники на всех факультетах и отделениях для выявления общего уровня подготовки абитуриентов по этой дисциплине.

В 1990 г. на подготовительных курсах при академии началась подготовка учащихся по ОИВТ. Забегая вперед, отмечу, что эта мера была действенна: процент поступивших среди абитуриентов, посещавших эти курсы и изучавших предмет «Информатика», был значительно выше, чем среди непосещавших их. Это, видимо, объясняется тем, что, во-первых, еще не во всех школах хорошо налажена система обучения данному предмету, а во-вторых, учебники, по которым ведется преподавание, не освещают всех вопросов, ответы на которые необходимо знать для сдачи экзамена по информатике.

Поэтому мы думаем, что читателям будет интересно ознакомиться с программой экзамена в Государственной академии управления по основам информатики и вычислительной технике и рекомендациями при подготовке к его сдаче. Отмечу особо, что при подготовке необходимо обратить внимание на разделы 3, 4, 5, 7 предложенной ниже программы, поскольку данные темы являются наиболее

трудно усваиваемыми при самостоятельном изучении. Читая лекции на подготовительных курсах при академии, я отметила следующую закономерность: большинство учащихся практически не умеют строить блок-схемы алгоритмов. При том, что на вступительном экзамене абитуриент может написать программу на любом освоенном языке программирования, очень важно именно на алгоритме проверить логическую цепочку рассуждений и их правильность. Поэтому большое внимание необходимо уделить именно этой части программы. Совсем нет в школьных учебниках материалов по структурной организации и функционированию ЭВМ.

В предложенном списке рекомендуемой литературы есть учебники [1, 3, 17], освещающие этот раздел программы. При подготовке к экзамену также следует обратить внимание на арифметику в позиционных системах счисления и действия с кодами двоичных чисел. Наиболее часты здесь ошибки из-за невнимательности.

К сожалению, в обзорной статье невозможно осветить все ключевые вопросы методики подготовки к вступительному экзамену по ОИВТ, поэтому за дополнительными рекомендациями и разъяснениями вы можете обращаться на подготовительные курсы при Государственной академии управления.

Программа вступительного экзамена 1991 г. составлена в соответствии с программами общеобразовательных средних школ по предмету «Основы информатики и вычислительной техники» и состоит из 8 разделов. В 1992 г. вступительный экзамен будет проводиться письменно. В него будут включены задачи из разделов 3, 4, 7, 8, но подготовку целесообразно проводить по предлагаемой программе. Для описания вычислительных процессов при решении экзаменационных задач абитуриент может применять любой освоенный язык программирования.

Абитуриент должен показать осведомленность о предмете информатики, стандартной конфигурации и принципах построения персонального компьютера, о назначении, технических характеристиках и потребительских свойствах отдельных устройств, об областях применения компьютера и составе его программного обеспечения.

Абитуриент должен знать арифметические и логические основы построения компьютера, основы алгоритмизации вычислительных процессов и язык программирования.

Абитуриент должен уметь представлять информацию в разрядной сетке компьютера, выполнять арифметические действия над числами в заданной форме представления (с фиксированной и плавающей запятой), вычислять значения логических функций по заданным значениям переменных, упрощать логические функции, описывать алгоритмы и программировать вычислительные процессы.

Программа

Раздел 1. Введение в информатику.

Предмет информатики. Место информатики среди других наук. Понятие об информации, ее видах и измерении. Вычислительная техника (ВТ) как средство обработки информации. Роль информатики и ВТ на современном этапе развития общества.

Раздел 2. Общий принцип построения компьютера.

Сущность процесса автоматизации вычислений. Понятие о команде, программе, программном обеспечении, алгоритме. Принцип программного автоматического управления ЭВМ. Общая схема ЭВМ. Классификация ЭВМ по назначению, производительности, потребительским свойствам. Современное состояние и перспективы развития компьютерной техники и ее программного обеспечения.

В данном разделе также необходимо показать знание принципов фон Неймана, представления информации в ЭВМ, умение оперировать понятиями машинного слова, информационной единицы, дать основные характеристики ЭВМ, их особенности и возможности, а также знать архитектуру ЭВМ и ее функциональных узлов (процессор, монитор, клавиатура, принтер и другие виды устройств ввода-вывода).

Раздел 3. Арифметические основы ЭВМ.

Позиционные системы счисления (десятичная, двоичная, шестнадцатеричная). Перевод чисел из одной системы счисления в другую. Двоичная арифметика. Машинные коды чисел: прямой, обратный, дополнительный. Действия с кодами двоичных чисел. Представление в ЭВМ целых и вещественных чисел. Машинные единицы информации.

В данном разделе абитуриент должен знать определение позиционных систем счисления и знать их отличие от непозиционных, должен знать алгоритм перевода числа из одной системы в другую (его целой и дробной частей), представление чисел с плавающей и фиксированной точкой, представление символьной информации (код ASCII).

Раздел 4. Логические основы ЭВМ.

Основные понятия и область применения алгебры логики. Элементарные логические операции. Логические функции. Построение таблиц истинности. Законы (аксиомы) алгебры логики и их следствия. Вычисление логических выражений при заданных наборах переменных. Упрощение логических выражений (на основе законов алгебры логики и их следствий). Логические элементы ЭВМ.

При изучении данного раздела абитуриент должен владеть понятиями комбинационной схемы и цифрового автомата, уметь строить логические схемы и минимизировать их, показать способы составления совершенной дизъюнктивной нормальной формы (СДНФ) и совершенной конъюнктивной нормальной формы (СКНФ) и этапы синтеза комбинационных схем.

Раздел 5. Структурная организация и функционирование ЭВМ.

Функциональные узлы ЭВМ: регистры, сумматоры. Назначение узлов ЭВМ. Функциональные устройства ЭВМ. Организация памяти и процессора. Понятие об арифметикологическом устройстве и устройстве управления. Организация автоматического исполнения программы. Назначение и характеристика внешних устройств ЭВМ: дисплей, клавиатура, принтер, накопители на магнитных дисках и лентах.

Абитуриенту необходимо знать определение узла ЭВМ, показать знание принципа действия регистров, счетчиков, дешифраторов, сумматоров и дать их определение; показать место процессора, памяти и периферийных устройств в ЭВМ.

Раздел 6. Программное обеспечение ЭВМ.

Базовое (системное) и прикладное программное обеспечение. Назначение и структура операционной системы. Общие сведения о пакетах прикладных программ. Понятие о трансляторах.

Абитуриент должен использовать полученные знания о файлах системы, составе операционной системы, супервизоре, утилитах, операционных режимах обработки информации на ЭВМ, системе управления базами данных, диалоговой обработке текстов, текстовых редакторах, электронных таблицах, обработке графической информации на ЭВМ.

Раздел 7. Алгоритмизация вычислительного процесса.

Понятие алгоритма. Свойства алгоритма. Способы представления алгоритмов. Формализованная запись алгоритма. Примеры алгоритмов. Формальное исполнение алгоритмов.

Этапы подготовки и решения задач на ЭВМ. Этап анализа постановки задачи. Примеры анализа постановки задачи. Разработка алгоритмов. Программирование. Трансляция. Отладка. Эксплуатация программы.

Типы данных: понятие, примеры. Понятие константы, переменной, массива. Характеристики величин (имя, значение, тип). Понятие выражения, классификация выражений. Строковые выражения.

Назначение и представление в алгоритмах основных операций (присваивания, ввода, вывода, ветвления, повторения действий), примеры. Базовые алгоритмические конструкции. Использование комментариев в алгоритме. Примеры линейных алгоритмов. Примеры разветвляющихся алгоритмов. Виды циклов. Тело цикла, пример цикла. Примеры циклических алгоритмов. Использование массивов в алгоритмах.

Абитуриент должен знать понятие идентификатора простой переменной, числовой и символьной константы, индексированной переменной, а также может продемонстрировать знания алгоритмов сортировок с помощью простых включений, простого выбора и простого обмена.

Раздел 8. Программирование вычислительного процесса.

Современные языки программирования (ЯП). Их типы. Элементы ЯП. Арифметические и логические операции ЯП. Идентификаторы переменных. Описание простых переменных в программе. Описание массивов в программе. Стандартные функции в ЯП: классификация, назначение, примеры. Операторы присваивания, ввода, вывода, безусловного перехода, условного перехода, выбора, цикла, цикла с предусловием, оператор размерности, функции, оператор-подпрограмма, графики. Управление выводом.

Реализация алгоритмических конструкций в ЯП. Примеры линейной, разветвляющейся, циклической программы. Использование комментариев в программе.

Отладка и тестирование программы. Виды ошибок, средства их устранения.

Типовые вопросы и задачи для подготовки к вступительным экзаменам по дисциплине «Основы информатики и вычислительной техники»

К разделу 2.

1. Архитектура и характеристики мини-ЭВМ. Принцип «общей шины».
2. Алгоритм работы процессора.
3. Основные принципы построения ЭВМ.
4. Принципы произвольного доступа к памяти.
5. Логическая структура ЭВМ.
6. Единица измерения памяти ЭВМ.
7. Архитектура и характеристики микро-ЭВМ. Структурная схема микропроцессора.
8. Память и процессор в структуре ЭВМ.
9. Три класса ЭВМ.
10. Принципы иерархии памяти.
11. Представление информации в ЭВМ.
12. Принцип хранимой программы и т. д.

К разделу 3.

1. Перевести число из одной системы счисления в другую:

а) из десятичной в двоичную: 31, 75.

б) из двоичной в шестнадцатеричную: 10010, 101

в) из шестнадцатеричной в десятичную: 1A7. 1

г) из шестнадцатеричной в двоичную: 2BE. A1

д) из десятичной в шестнадцатеричную: 357. 08

е) из двоичной в десятичную: 110111. 001

2. Выполнить арифметические действия

над числами в двоичной и шестнадцатеричной системе счисления:

а) 1011011 + 10111 б) 10100001 — 110 в) 110 × 101
г) 7B41 + F5A д) 7B41 — F5A

3. Выполнить арифметические действия над числами A и B в форме с плавающей точкой в формате слова (32 разряда), используя под изображение мантиссы 6 шестнадцатеричных разрядов:

а) $A=+18\frac{3}{4}$, $B=+19\frac{1}{4}$ б) $A=+18\frac{3}{4}$, $B=-7\frac{1}{2}$

в) $A=+12\frac{7}{16}$, $B=-17\frac{3}{16}$ г) $A=-14\frac{7}{16}$, $B=-6\frac{1}{16}$

5. Представление чисел в разрядной сетке компьютера:

а) представить в разрядной сетке полуслова (16 разрядов) целые числа A и B в форме с фиксированной запятой.

A=+163, B=—112;

б) представить в разрядной сетке слова (32 разряда) в форме с плавающей точкой (с учетом смещенного порядка) число A: +1541; —1541; —0.07; +0.07.

в) записать двоичное изображение числа в память компьютера в форме с плавающей точкой в разрядной сетке слова, если задана его мантисса m_A и смещенный порядок p^* в шестнадцатеричной форме:

$m_A=+0.3FA1$, $p^*=44$;
$m_A=-0.3FA1$, $p^*=44$;
$m_A=-0.285C3$, $p^*=3F$;
$m_A=+0.285C3$, $p^*=3F$

г) произвести действия с кодами двоичных чисел:

A=+0,1011, B=—0,1111 } в дополнительном коде

A=+0,1110, B=—0,1010 } в обратном коде

К разделу 4.

1. Построить таблицу истинности работы устройства, описанного логической функцией:

$$F(x_1, x_2, x_3)=x_1\cdot x_2\cdot\bar{x}_3\vee\bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_3$$

2. Упростить логические выражения:

$$(X\vee Y\vee Z)\cdot(X\vee\bar{Y}\vee\bar{Z})$$

3. Вычислить значение логического выражения

$$XY\vee\bar{X}Z\bar{Y})\cdot(X\vee\bar{Y}\vee Z)$$

4. Составить логическую функцию работы устройства по заданной таблице истинности. Полученную логическую функцию упростить.

x_1	x_2	x_3	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

5. Построить логическое устройство, имеющее 3 входа: a, b, c и 1 выход P по булевой функции, используя минимальное количество логических элементов, составив таблицу истинности.

$$P=(a\vee b\vee\bar{c})\ (a\vee\bar{b}\vee c)\ (\bar{a}\vee\bar{b}\vee c)\ (\bar{a}\vee b\vee c)$$

К разделам 7, 8.

1. Составить алгоритм и написать программу для задачи:

а) Пусть задана матрица $A=(a_{ij})$ ($i=1, 2, ..., n$; $j=1, 2, ..., n$) и вектор B ($b_1, b_2, ..., b_n$).

Составить схему для вычисления вектора D=B*A по формуле:

$$d_i=\sum_{k=1}^{n}a_{ik}b_k\quad(i=1, 2, ..., n)$$

2. Дано X. Вычислить Y, если:

$$y=\begin{cases}0 \text{ при } x\leqslant 0,\\ x^2-x \text{ при } 0<x\leqslant 1,\end{cases}$$

$x^2-\sin\pi x^2-1$ в противном случае.

3. Составить схему, содержащую оператор цикла для вычисления значений Y_n величины Y по формуле:

$$Y_n=\sqrt{\frac{\sum_{i=0}^{k}(x_{n+1}-x_{n-i})^2}{(k+1)\ (k+2)}}$$

для $n=1, 2, ..., 100$, где

$$k=\begin{cases}n-1, \text{ если } 1\leqslant n\leqslant 11\\ 10, \text{ если } 11<n<90\\ 100-n, \text{ если } 90\leqslant n\leqslant 100\end{cases}$$

4. Дана целочисленная матрица 10×10. Найти номера столбцов, содержащий максимальный элемент матрицы.

5. Дана символьная матрица 10×10. Найти номер последнего по порядку столбца, содержащего наименьшее число букв Ш и Щ.

6. Вычислить

$$\sum_{i=1}^{n} \frac{i!}{x^i}$$

7. Вычислить сумму шаров в пирамиде из n слоев, если количество шаров в слое равно квадрату порядкового номера слоя.

8. Определить количество и порядковые номера отрицательных чисел в массиве из десяти элементов A(10).

9. Рассортировать в порядке возрастания элементы матрицы A(3,4).

Следует отметить, что в данной теме основной интерес представляет цепочка логических рассуждений при построении алгоритма задачи, поскольку большая часть абитуриентов не умеет писать универсальных алгоритмов, пригодных для написания программы на любом языке.

К подтеме «Типы данных».

Какие из следующих последовательностей символов являются именами переменных и какого типа:

1. а) a—1; б) 2a; в) A(i+1); г) A_i
2. а) sin x; б) sin (x); в) a(i); г) A1
3. а) x; б) x1; в) A1(5); г) —a5
4. а) объем; б) max 15; в) B(j+i); г) 12a

10. Составить схему алгоритма вычисления $y=x^2+sin\ x$, где x изменяется от 0,1 до 0,9 с шагом 0,05.

11. В одномерном массиве A(10) вычислить произведение отрицательных элементов, имеющих нечетные индексы.

12. В двумерном массиве A(10,4) найти наибольший элемент и его номер.

13. В одномерном массиве поменять местами наибольший и наименьший элементы.

14. В одномерном массиве произвольных чисел A(10) найти наибольший элемент из отрицательных.

15. В одномерном массиве из 20-ти произвольных чисел определить сумму и количество элементов, которые меньше —5 и номера которых кратны 4.

16. В двумерном массиве A(5,5) вычислить суммы элементов столбцов.

17. Составить схему алгоритма вычисления $y=x^2+\sqrt{z}$, где x— переменная с индексом, заданная значениями x_1, ..., x_{50}, переменная z — задается своими начальным $z_н$ и конечным $z_к$ значениями, а также шагом изменения h_z. Значение переменной z вычисляются по рекурентной формуле $z=z+h_z$. Необходимо рассмотреть все варианты вычисления, когда для каждого значения x перебираются все значения y.

18. В двумерном массиве произвольных чисел A(10,10) найти наибольший элемент главной диагонали.

19. В двумерном массиве произвольных чисел A(10,10) найти наименьший элемент побочной диагонали.

20. В двумерном массиве произвольных чисел B(4,7) взаимно поменять местами 2-й и 5-й столбцы.

21. В двумерном массиве произвольных чисел B(4,7) определить номера столбцов, в которых среднее арифметическое его элементов меньше, чем среднее арифметическое элементов массива.

Литература

1. *Каган Б. М.* Электронные вычислительные машины и системы. М.: Энергоатомиздат, 1985.
2. *Уорт.* Программирование на языке Бейсик. М.: Машиностроение, 1981.
3. *Фролов Г. Д., Кузнецов З. И.* Элементы информатики. М.: Высшая школа, 1989.
4. *Абрамов С. А., Зима Е. В.* Начала информатики. М.: Наука, 1989.
5. *Светозарова Г. И., Мельников А. А., Козловский А. В.* Практикум по программированию на языке Бейсик. М.: Наука,
6. *Стогній А. А.* Программное обеспечение персональных ЭВМ (справочное пособие). Киев: Наукова думка, 1989.
7. *Брябрин В. М.* Программное обеспечение персональных ЭВМ. М.: Наука, 1989.
8. *Громов Г. Р.* Национальные информационные ресурсы. М.: Наука, 1989.
9. *Королев Н.* Структура ЭВМ и их математическое обеспечение. М.: Наука, 1985.
10. *Уолш Б.* Программирование на языке Бейсик. М.: Радио и связь, 1987.
11. *Кушниренко А. Г., Лебедев Г. В., Сворень Р. А.* Основы информатики и вычислительной техники. Проб. учебник для средних учебных заведений. М.: Просвещение, 1990
12. *Каймин В. А. и др.* Основы информатики и вычислительной техники: Проб. пособие для X—XI кл. средних школ. М.: Просвещение, 1989.
13. Основы информатики и вычислительной техники / Под ред. В. М. Моңахова, А. П. Ершова. М.: Просвещение, 1985 — 1 часть, 1986 — II часть.
14. *Растригин Л. А.* С компьютером наедине. М.: Радио и связь, 1990.
15. *Черемных С. В., Гиглавый А. В., Поляк Ю. Е.* От микропроцессоров к персональным ЭВМ. М.: Радио и связь, 1988.
16. *Блэнд Г.* Основы программирования на языке Бейсик в стандарте MSX. М.: Финансы и статистика, 1989.
17. *Попков А. И.* Введение в практическую информатику. М.: Радио и связь, 1990.

Азартные игры...

между людьми запрещены Уголовным кодексом, но с ПМК можно играть во что угодно. С математической точки зрения они интересны тем, что с вами играет Его Величество Случай, вращающий колесо Фортуны в пятом измерении, делая исход партии непредсказуемым. С психологической полезны тем, что без особых затрат можно увидеть, к чему приводит жажда легкой наживы. А с игровой привлекают — ну конечно, азартом! Если он у вас есть, вводите программу С. Чегодаева (Санкт-Петербург) «Ставка».

	0	1	2	3	4	5	6	7	8	9
0	ПП	91	К[x]	П4	ПП	91	К[x]	П5	ИП4	−
1	Fx≠0	04	К!x!	ИП9	:	ИПВ	ХУ	−	Fx<0	26
2	FBx	ИПВ	×	К[x]	ВП	27	Cx	ПС	ИП4	ИП7
3	:	ИП5	+	С/П	Fx≥0	28	П0	ИПС	+	Kx≠07
4	ИПА	ИП0	−	Fx≥0	28	ПП	91	П2	ИП4	−
5	ИП5	ИП2	−	×	Fx<0	62	ИП0	/−/	П0	ИПС
6	/−/	ПС	ИПВ	ИПС	+	ПВ	Fx≠0	84	ИПА	ИП0
7	+	ПА	Fx≠0	86	С/П	FL1	00	ИПА	ИПВ	−
8	Fx≠0	89	Fx≥0	86	ВП	.	0	$F\text{tg}^{-1}$	С/П	Fπ
9	С/П	ИПД	В↑	КСч	ПД	ИП9	×	1	+	В/О

Поставив переключатель «Р — Г» в положение «Г», введите константы:

```
100 П7, 13 П9, 3 F 1/X П8
```

(последнее число можно менять; чем оно меньше, тем рискованнее играет П ЛК).

Правила игры таковы: ПМК выводит на экран два числа (от 1 до 13 включительно). Цель игрока — угадать, попадет ли третье число, которое тоже вычислит ПМК, между первыми двумя, и сделать ставку, не превышающую сумму, принадлежащую игроку в данный момент. ПМК тоже делает ставку, но по своему алгоритму. Если третье число оказалось между двумя первыми, то суммы ПМК и игрока увеличиваются на размеры их ставок; в противном случае суммы соответственно уменьшаются. Если вы не хотите делать ставку, то наберите 0.

Игра идет в несколько этапов. Их число перед началом заносится в Р1. Этапом не считается тот кон, в котором и ПМК, и игрок сделали нулевые ставки (в таком случае программа выдает новые два числа, задающие промежуток). После заданного числа этапов ПМК подводит итоги: 00 — победа ПМК, ЕГГОГ — ваша, число π — боевая ничья. Впрочем, игра может кончиться и досрочно, если ваши денежки кончатся.

Перед каждой игрой следует ввести сумму денег игрока (примерно 100) в ПА, сумму ПМК (примерно столько же) в ПВ, число этапов (10—30) в П1. Теперь нажмите В/О С/П. На индикаторе — числа-границы, разделенные запятой; в РУ — ставка ПМК. Введите свою ставку и нажмите С/П. В случае ошибки (ставка отрицательна) числа повторяются. Если все введено правильно и ставки ненулевые, после останова в РХ будет сумма игрока, в РУ — сумма ПМК.

Нажимайте С/П и — успеха вам!

На всякий случай — контрольный пример. Суммы равны 100, число этапов — 2, число риска — 3F 1/X; В/О С/П; ПМК выдает «8.06», его ставка — 0, набираем тоже 0 С/П; «13.7», ставка ПМК — 46, ставим 50 С/П, на экране результат: наша сумма — 50 (проиграли!), сумма ПМК — 54; С/П; «8.11», ставка ПМК — 0, ставим 0 С/П; «10.05», ставка ПМК — 20, ставим 40 С/П; выигрыш, наша сумма — 90, у ПМК — 74, на экране — ЕГГОГ. На этот раз победа за нами (хоть и с потерей червонца).

Любителям посокращать программы С. Тарасов предлагает задействовать регистр Е для команды КППЕ вместо трех обращений ПП 91. Возможно использование адресно-кодовой связки

```
24. ВП  25. 27,
```

при этом надо заменить фрагмент

```
82. Fx≥0  83. 86
```

на

```
82. Fx<0  83. 25
```

и выбросить участок

```
84. ВП  85. .
```

В итоге вы получите сокращение на пять команд, что, впрочем, еще не предел.

Любители карточных игр благоволят обратить внимание на программу «Очко» А. Криволапова из Санкт-Петербурга. Ведущий, роль которого играет ПМК, сдает карты поочередно вам и себе. Вы подсчитываете сумму (которая не должна превысить 21; валет — 2 очка, дама — 3, король — 4, туз — 11, остальные карты по их номиналу). Победит тот, кто наберет больше очков, не перейдя границу 21 очка. «Очко» — 21.

```
   0   1   2   3   4   5   6   7   8   9
0 Cx  П4  П5 КППЕ В↑ ИП5  +  П5  ХУ  С/П
1 Fx=0 03  3  П2 КППЕ ИП4  +  П4 FL2  14
2 ИП4 ИПС  -  Fx≥0 30 ИП5 ИП4  -  Fx≥0 53
3 ИП5 ИПС  -  Fx<0 40 ИП4 ИП5  -  Fx≥0 53
4 ИП1 ИПД  +  П1 ИП0 ИПД  -  П0 ИП4 ВП
5  .  БП  00 ИП0 ИПД  +  П0 ИП1 ИПД  -
6 П1 ИП4 ИП3 С/П БП  00 ИП9 В↑ КСч П9
7  1   0   x   2   +  К[x] В/О
```

Вводим константы

```
66 ПЕ  22 ПС,
```

в Р0 — деньги игрока, в Р1 — ПМК, РД — ставка, Р9 — произвольное число, Р3 — сигнал победы игрока «_____»:

```
5.5555555 Кинв К[x] ВП 7 ПЗ
                    ("--------")
```

Начинаем: В/О С/П. На экране — карта, в РУ — сумма очков. Если нужна еще карта, нажмите С/П (на индикаторе не должно быть нуля); если карта не нужна, наберите 0 С/П. ПМК начнет набирать карты себе и затем выдаст результат: ЕГГОГ — победа ПМК, в РУ — деньги игрока, в РХ — количество очков ПМК (продублировано в Р4); или «_____» — ваша победа, в РУ — количество очков ПМК, ваши очки продублированы в Р5. Состояние банка — в регистрах 0 и 1.

В процессе игры можно изменять ставку, изменяя ставку, изменяя содержимое регистра РД (для продолжения игры после этого нужно нажать С/П).

Теперь — примеры. Произвольное число 0,1234567, сумма денег — по 100, ставка — 20. В/О С/П: «3» С/П «11» С/П «11» ХУ «25» — перебор, 0 С/П (теперь карты набирает ПМК) «_____» — победа! ХУ «26» — у ПМК перебор больше, ИПО «120». Новая партия: С/П «8» С/П «4» С/П «11» — перебор, ХУ «23» 0 С/П ЕГГОГ — мы проиграли, ХУ «10» — только что выигранные деньги вернулись к ПМК, ХУ «22» — у ПМК меньший перебор.

Изменим ставку — пойдем ва-банк: 100 ПД С/П «3» С/П «6» С/П «4» С/П «11» ХУ «24» — снова не повезло, 0 С/П «ЕГГОГ» ХУ «0» — мы проигрались, ХУ «17» С/П...

ПМК продолжит играть в долг, т. е. деньги в вашем кошельке — регистре будут со знаком «минус».

Вот и все на сегодня. Напоминаем, что администрация КЛИПа с нетерпением ждет от вас доработок предложенных программ, ваших собственных разработок и идей. Не забудьте оценить «Ставку» и «Очко» по десятибалльной системе для нашего хит-парада: игра, сервис, общая оценка.

До новых встреч!

В. МОСКОВЦЕВ

Сеть MSX-2

Несколько лет назад в лицее № 542 был оборудован дисплейный класс на основе базового КУВТ «YAMAHA MSX-2» со стандартной сетью Network version 3.0. Очень скоро выяснился существенный недостаток этого класса — наличие дисководов только на учительской машине. Поставляемая сеть хотя и имеет достаточно большое количество функций, но не поддерживает работу как дисковой операционной системы, так и дискового расширения Бейсика. При этом некоторые функции, как например, функция BRECEIVE, были реализованы с ошибкой.

Класс невозможно было использовать при изучении таких языков программиро-

вания, как Паскаль, Си и ассемблер. Нельзя было использовать и большинство обучающих программ, написанных к этому моменту, так как они либо нуждались в подгружаемых модулях, либо работали только в среде DOS.

Эти и некоторые другие соображения привели к необходимости создания сетевого пакета, поддерживающего работу DOS и дискового Бейсика на ученических машинах. К его основным достоинствам можно отнести следующие:

пакет полностью эмулирует (имитирует) набор стандартных системных функций и операций дискового Бейсика, включая операции над двоичными файлами, переход из операционной системы в Бейсик с сохранением всех файлов и обратно. При этом все операции производятся на внутреннем виртуальном диске, что позволяет обеспечить большую скорость работы и избежать конфликтов при работе с файлами, когда несколько пользователей пытаются одновременно читать и писать в один файл.

При этом размер виртуального диска может быть изменен в зависимости от требований пользователя, что позволяет программам использовать различные объемы как процессорной памяти, так и видеопамяти (например, I-LISP и Л-ГРАФ, описанные в «ИНФО». 1991. 2, 6);

пакет имеет модульную структуру, что делает его довольно гибким и с равным успехом позволяет использовать его в качестве инструментального средства;

пакет позволяет осуществлять обмен файлами между машинами, при этом каждая из них может выступать в роли сервера (головной машины, с которой производится обмен), а также имеется возможность защиты файлов от случайной или преднамеренной порчи;

пакет поддерживает терминальный режим, позволяющий управлять с учительской машины работой как всего класса в целом, так и отдельного терминала. В этом режиме содержимое экранов учителя и ученика совпадают и нажатие клавиш на учительской машине вызывает такую же реакцию на экране ученической, как и нажатие клавиш на последней. Это фактически соответствует ситуации, когда учитель подходит к ученику и работает на его машине;

пакет позволяет управлять работой ученических машин в пакетном режиме, с учительской машины можно перезапускать машины учеников или пересылать на них сообщения.

В качестве отдельного модуля к представленному пакету была написана программа, позволяющая преподавателю создавать меню системного уровня и задавать последовательность выполнения командного файла.

```
Занятие N6

1) Провести тест
2) Показать результаты
   теста
3) Разослать Л-ГРАФ
4) Переслать сообщение
   об окончании урока
5) Сохранить результат
   урока
6) Перезапустить машины
   учеников
7) Вернуться в основное
   меню

Что вы хотите ? _
```

Основной принцип работы сети заключается в следующем.

Сначала инициализируются ученические машины:

1. На ученические машины пересылается копия ПЗУ, поддерживающего работу MSX-DOS и содержащего дисковый Бейсик. На ученической машине в копию вносятся изменения, исправляющие ошибки, незаметные на учительской машине и не дающие возможность запускать на ученических некоторые программы, например, M80.COM.

2. Затем пересылается копия участка памяти содержащего MS-DOS,и некоторые системные переменные.

3. Инициализируется виртуальный диск нужного размера.

4. Пересылаются файлы согласно списку инициализации. При этом есть следующие возможности:

переслать файл на все машины или только на одну из них, согласно ее номеру в сети, например:

```
COMMAND.COM
0 DBG.COM
```

файл пересылается на все ученические машины

```
2 MYFILE.TXT
```

файл пересылается только на вторую машину;

переименовать файл, т. е. после пересылки он будет записан нв виртуальном диске ученика под другим именем, например:

```
EXAMPL1.LOG EXAMPL.LOG
```

файл EXAMPL1.LOG будет переслан на ученические машины и записан там под именем EXAMPL.LOG.

```
START.BAT AUTOEXEC.BAT
```

файл START, BAT будет переслан на ученические машины под именем AUTOEXEC. BAT и, как любой файл начальной установки, начнет выполняться сразу же после пересылки.

Если файл START.BAT содержит строку

```
LOGO EXAMPL.LOG
```

то сразу после пересылки будет запущен Л-ГРАФ;

обычно эти две возможности используются одновременно. Например:

```
1 TEST1.LOG TEST.LOG
2 TEST2.LOG TEST.LOG
```

Тогда на ученических машинах будет записан файл с именем TEST.LOG, но на первой машине он будет копией файла TEST1.LOG, а на второй копией файла TEST2.LOG.

После инициализации на сервере запускается монитор, который по очереди опрашивает ученические машины, обрабатывает их запросы на прием, передачу или удаление файлов, просмотр каталогов дисков. Протокол работы выводится на экран, однако его можно распечатать и на принтере. Если в результате работы обучающей программы создается файл, содержащий некоторую информацию о ходе работы, то он может быть сохранен на внутренних дисках под одинаковыми именами, а на сервере под разными. Избежать наложения файлов помогут два символа в имени файла SN, которые на сервере будут восприниматься как номер машины.

Хочу поблагодарить Н. М. Леонову, М. А. Зайцеву и других преподавателей информатики лицея № 542 за помощь в подготовке статьи и содействие при отладке пакета.

М. ЮСОВ,
студент МИФИ

Графический редактор для УКНЦ

В сентябре 1990 г. в нашей школе был установлен КУВТ УКНЦ. Программное обеспечение, поставляемое с классом, имело минимум прикладных программ. Не было среди них и графического редактора. Это и привело к идее написать недостающие программы своими силами.

Редактор создавался в несколько этапов. Вначале была написана «оболочка» для операторов графики, затем интерфейс пользователя и в завершение — блок записи на диск созданной Бейсик-программы.

Во время работы обнаружилась ошибка вильнюсского Бейсика. При частом обращении к функции POINT она начинает выдавать очень «страшные» числа вместо положенного интервала от 0 до 8.

Принцип работы программы достаточно прост, и при необходимости ее можно перевести на другой диалект Бейсика. Для этого необходимо внести небольшие изменения. В блоке записи необходимо изменить формат операторов работы с файлами, а в блоке формирования строк Бейсик-программы написать полные имена операторов Бейсика.

Если редактор используется для оформления программ, то в целях экономии памяти можно изменить формат записываемых строк вставив в строку оператор DATA и заменив имя оператора графики числовым кодом от 0 до 4. Также необходимо будет убрать все круглые скобки и знак «—». В этом случае для вы-

```
10 CLEAR 10000
20 DIM RR(300)
30 COLOR 8,1,1
40 SCREEN 0
50 WIDTH 80,0
60 SCREEN 3
70 CLEAR
80 RR(1)="1'(c) SHP 1991.
          Mirag V4.00"
90 COLOR 8,1,1
100 RR(2)="2SC3"
110 SO%=2
120 LINE (0,0)-(639,240),8,B
130 LINE (1,1)-(638,239),8,B
140 S%=5
150 P%=5
160 L%=8
170 X%=8
180 Y%=8
190 CK%=8
200 U1%=8
210 U2%=8
220 COLOR 8,6,6
230 ? AT(32,1)"  M i r a g  "
240 PSET (X%,Y%),8
250 TH=STRINGH(79,32)
260 GOSUB 1760
270 COLOR 8,1,1
280 ? AT(41,0)"  Line  "
290 ? AT(57,0)
      "Текущая строка:"SO%
300 IF SO%>345 TH GOS 2090
310 PY%=Y%
320 PX%=X%
330 GOSUB 1810
340 IF IR="A" TH 620
350 IF IR="B" TH 640
360 IF IR="C" TH 660
370 IF IR="D" TH 680
380 IF IR=" " TH 570
390 IF IR=CHRR(9) TH 700
400 IF IR=CHRR(5) TH 730
410 IF IR="F" TH 990
420 IF IR="J" TH 1530
430 IF IR="U" TH 1620
440 IF IR="S" TH 1140
450 IF IR="M" TH 1690
460 IF IR="T" TH 2150
470 IF IR="P" TH 1410
480 IF IR="H" TH GOS 1760
490 IF IR="Y" TH 2370
500 GOTO 330
510 C%=POINT(X%,Y%)
520 IF C%>8 OR C%<0 TH 510
530 PSET (PX%,PY%),P%
540 PSET (X%,Y%),CK%
550 P%=C%
560 GOTO 310
570 GOSUB 1830
580 GOSUB 1940
590 LINE (U1%,U2%)-(X%,Y%),L%
600 P%=L%
610 GOTO 700
620 IF Y%+S%>238 TH Y%=3%
    EL Y%=Y%+S%
630 GOTO 510
640 IF Y%-S%<3 TH Y%=238
    EL Y%=Y%-S%
650 GOTO 510
660 IF X%+S%>638 TH X%=8
    EL X%=X%+S%
670 GOTO 510
```

```
680 IF X%-SX<3 TH X%=638
    EL X%=X%-SX
690 GOTO 510
700 U1%=X%
710 U2%=Y%
720 GOTO 270
730 ? AT(0,0)THAT(0,0)" Вы
      желаете записать файл ?
      [Y=Да/K=Конец] "
740 GOSUB 1810
750 IF IN="Y" TH 990
760 IF IN="K" TH END
770 GOSUB 1760
780 GOTO 330
790 ? AT(0,0)TH
800 NN=""
810 ? AT(0,0)" Имя файла :"NN
820 GOSUB 1810
830 IF IN=CHRN(13) TH 860
840 NN=NN+IN
850 GOTO 810
860 NN=MIDN(NN,1,6)+".ASC"
870 OPEN NN FOR OUTPUT
880 FOR MZ=0 TO SOX
890 ? #RN(MX)
900 NEXT MX
910 OLN=STRN(SOX+1)
920 ? #OLN+"IFINKEYN="+CHRN(34)
      +CHRN(34)+"TH"+OLN
930 CLOSE
940 ? AT(0,0)" Записан файл "
      NN"; Для продолжения
      нажмите [пробел] "
950 GOSUB 1810
960 ? AT(0,0)THAT(0,0)
      " Выполнить записаную
      программу ? [Y=Да] "
970 GOSUB 1810
980 IF IN="Y" TH LOA NN,R
    EL 260
990 COLOR 1,1,8
1000 ? AT(41,0)"  Box   "
1010 GOSUB 1810
1020 IF IN="F" TH 1060
1030 IF IN="B" TH 1090
1040 IF IN="E" TH 1130
1050 GOTO 1010
1060 LINE (U1%,U2%)-(X%,Y%),
     L%,BF
1070 GOSUB 1830
1080 GOSUB 2000
1090 LINE (U1%,U2%)-(X%,Y%),
     L%,B
1100 GOSUB 1830
1110 GOSUB 1970
1120 P%=L%
1130 GOTO 270
1140 COLOR 1,1,8
1150 ? AT(41,0)" Circle "
1160 KZ=1/1.56
1170 COLOR 8,5,5
1180 ? AT(33,0)"R ="RX
1190 GOSUB 1810
1200 IF IN=" " TH 1360
1210 IF IN="0" TH 1270
1220 IF IN="K" TH 2230
1230 IF IN="+" TH 1310
1240 IF IN="-" TH 1330
1250 IF IN="E" TH 1400
1260 GOTO 1190
1270 RX=0
1280 COLOR 8,5,5
1290 ? AT(33,0)"R =      "
1300 GOTO 1180
1310 RX=RX+SX
1320 GOTO 1180
1330 IF RX-SX<=0 THEN 1190
1340 RX=RX-SX
1350 GOTO 1180
1360 COLOR 8,1,1
1370 CIRCLE (X%,Y%),RX,LX,0,0,
     KZ
1380 GOSUB 1830
1390 GOSUB 2030
1400 GOTO 270
1410 COLOR 1,1,8
1420 ? AT(41,0)" Paint "
1430 GOSUB 1810
1440 IF IN="E" TH 1520
1450 KZ=VAL(IN)
1460 P%=KZ
1470 PSET (X%,Y%),1
1480 PAINT (X%,Y%),KZ,LX
1490 PSET (X%,Y%),8
1500 GOSUB 1830
1510 GOSUB 2060
1520 GOTO 270
1530 COLOR 1,1,8
1540 ? AT(0,0)"Цвет ?"
1550 GOSUB 1810
1560 LX=VAL(IN)
1570 IF LX>9 TH 1530
1580 COLOR 8,1,1
1590 ? AT(0,0)"Цвет ="LX
1600 LINE (1,1)-(638,239),LX,B
1610 GOTO 330
1620 COLOR 1,1,8
1630 ? AT(9,0)"Шаг ?"
1640 GOSUB 1810
1650 SX=VAL(IN)
1660 COLOR 8,1,1
1670 ? AT(9,0)"Шаг ="SX
1680 GOTO 330
1690 FOR VX=3 TO 636 ST SX
1700 LINE (VX,3)-(VX,238),5
1710 NEXT VX
1720 FOR VZ=3 TO 234 ST SX
1730 LINE (2,VZ)-(637,VZ),5
1740 NEXT VZ
1750 GOTO 330
1760 COLOR 8,1,1
1770 ? AT(0,0)THAT(0,0)"Цвет ="
     LZAT(9,0)"Шаг ="SZAT(17,0)
     "X ="XZAT(25,0)"Y ="YZ
     AT(33,0)"R ="RZAT(41,0)
     "  Line   "
1780 COLOR 1,1,8
1790 ? AT(0,1);"Memory: "
       FRE(" ")
1800 RETURN
1810 IN=INKEYN
1820 IF IN="" TH 1810 EL RET
1830 SOX=SOX+1
1840 KZN=STRN(KZ)
1850 XN=STRN(XX)
1860 YN=STRN(YX)
1870 U1N=STRN(U1X)
1880 U2N=STRN(U2X)
1890 KN=STRN(KX)
1900 LN=STRN(LX)
1910 RN=STRN(RX)
1920 SN=STRN(SOX)
1930 RETURN
1940 RN(SOX)=SN+"LIN("+U1N+","+
     U2N+")-("+XN+","+YN+"),"+LN
1950 GOSUB 2440
1960 RETURN
1970 RN(SOX)=SN+"LIN("+U1N+","+
     U2N+")-("+XN+","+YN+"),"+
     LN+",B"
1980 GOSUB 2440
1990 RETURN
2000 RN(SOX)=SN+"LIN("+U1N+","+
     U2N+")-("+XN+","+YN+"),"+
     LN+",B"
2010 GOSUB 2440
2020 RETURN 1120
2030 RN(SOX)=SN+"CI("+XN+","+YN
     +"),"+RN+","+LN+",0,0,"+KZN
2040 GOSUB 2440
2050 RETURN
2060 RN(SOX)=SN+"PA("+XN+","+YN
     +"),"+KN+","+LN
2070 GOSUB 2440
2080 RETURN
2090 COLOR 8,5,5
2100 ? AT(5,1)" Внимение !!!
     Записано максимальное
     количество операторов. "
2110 ? AT(5,2)" Нажмите два
     раза любую клавишу для
     перехода в режим записи. "
2120 GOSUB 1810
2130 GOSUB 1810
2140 RETURN 790
2150 BEEP
2160 COLOR 8,5,5
2170 ? AT(0,0)THAT(0,0)
       " Перезапуск ?!
         Вы уверены ?!    "
2180 GOSUB 1810
2190 IF IN="Y" TH 30
2200 ? AT(0,0)TH
2210 GOSUB 1760
2220 GOTO 330
2230 BEEP
2240 COLOR 8,7,7
2250 ? AT(32,1)"K ="KZ"     "
2260 GOSUB 1810
2270 IF IN="+" TH 2320
2280 IF IN="-" TH 2340
2290 IF IN="E" TH 1190
2300 IF IN=" " TH 1360
2310 GOTO 2260
2320 KZ=KZ+0.01
2330 GOTO 2250
2340 IF KZ-0.05<0 THEN 2250
2350 KZ=KZ-0.01
2360 GOTO 2250
2370 COLOR 8,6,6
2380 ? AT(52,1)"Цвет курсора ?"
2390 GOSUB 1810
2400 CKX=VAL(IN)
2410 COLOR 8,1,1
2420 ? AT(52,1)
       " Цвет курсора ="CKX
2430 GOTO 330
2440 HDX=LEN(RN(SOX))
2450 UN=""
2460 FOR DZX=1 TO HDX
2470 ZZN=MIDN(RN(SOX),DZX,1)
2480 IF ZZN=" " TH 2500
2490 UN=UN+ZZN
2500 NEXT DZX
2510 RN(SOX)=UN
2520 RETURN
```

Используемая клавиша	Действие клавиши в режиме *Line*
ТАБ	Установка начала линии или начала диагонали прямоугольника.
P+⟨*n*⟩	Закраска фигуры, ограниченной линиями с «текущим» цветом, цветом *n* (*n*[0...8]).
J+⟨*n*⟩	Изменение «текущего» цвета рисования на цвет *n* (*n*[0...8]).
U+⟨*n*⟩	Изменение шага перемещения графического курсора (*n*[0...9]).
	Перемещение графического курсора в соответствующем направлении.
УПР+E	Выход в режим записи файла на диск. Необходимо подтвердить правильность команды (Y/N). В случае утвердительного ответа (Y) редактор запрашивает имя записываемого файла. Если запись проведена успешно, редактор предлагает запустить записанную программу; в случае отрицательного ответа редактор возвращается в режим *line*, после чего можно продолжить создание изображения.
S	Включение режима рисования кругов и эллипсов (*Cirle*).
F	Включение режима рисования прямоугольников (*Box*).
ПРОБЕЛ	Рисование линии из точки, отмеченной клавишей ТАБ, в текущую точку (в точку, где находится курсор).
H	Обновление служебной строки.
Y	Изменение цвета графического курсора (норм=8).
M	Нанесение разметочной сетки с текущим шагом курсора.
T	Перезапуск редактора.
	Действие клавиши в режиме Circle
E	Отмена режима.
+	Увеличить радиус на «текущий» шаг перемещения курсора.
—	Уменьшить радиус на «текущий» шаг перемещения курсора.
0	Обнулить радиус.
K	Установка коэффициента сжатия круга. В этом режиме используются следующие клавиши: ⟨*E*⟩ — отмена; ⟨+⟩ — увеличить коэффициент (вытянуть эллипс по вертикали); ⟨—⟩ — уменьшить коэффициент (вытянуть эллипс по горизонтали); ПРОБЕЛ — нарисовать эллипс.
ПРОБЕЛ	Нарисовать круг.
	Действие клавиши в режиме *Box*.
B	Нарисовать прямоугольник-«рамку». Начало диагонали: точка, отмеченная клавишей ⟨ТАБ⟩, конец — текущая точка нахождения курсора.

F	То же, что и ⟨B⟩, только закрашенный прямоугольник.
E	Отмена режима

вода рисунка нужна простая подпрограмма-дешифратор. Этот способ записи предоставляет ряд преимуществ. Первое — значительно экономится память, второе — сокращается время компиляции, третье — рисунок можно переносить в любую часть экрана.

Программа отлично работает на РМУ, на РМП ее работа затруднена по причине недостатка памяти. Но при необходимости ее можно запустить, уменьшив числа в операторе CLEAR до 1000, DIM до 50, предварительно выгрузив драйверы устройств командами ОС:

UNLOAD⟨dev⟩ или SET⟨dev⟩OFF.

В заключение мне хочется поблагодарить преподавателей Л. Федякину (СШ № 36 Томск) и Е. Сарайкину (ТФ НИИ технологии машиностроения), уделивших мне много внимания.

А. ВЕРНИГОРА

ЧТО МОЖЕТ ЭВМ?

Почта, телефон, спутник связи...

Средства связи не напрасно во все времена и во всех странах привлекали внимание революционеров. Но приходит время, когда привычный ряд «почта, телефон, телеграф» следует продолжить словами «... и 30 спутников связи». Именно такого количества низкоорбитальных спутников достаточно, чтобы в поле зрения любого потенциального абонента, находящегося на поверхности Земли, всегда была хотя бы одна приемопередающая антенна.

Каждая космическая антенна позволяет связаться абонентским компьютерам, находящимся на расстоянии до 6 тыс. км друг от друга. Компьютеры могут быть стационарные, переносные и даже встроенные в радиотелефон. В последнем случае на процессор возлагается только задача установления связи, а диалог будет происходить с помощью обычного телефона.

С помощью спутников, облетающих земной шар за полтора часа, сообщение можно послать и на более далекое расстояние по ходу их движения. Правда, скорость доставки вашего оцифрованного послания будет равна скорости спутника, но это тоже, согласитесь, не очень медленно. А если учесть, что прием и передача адресату вашей электронной посылки на спутнике полностью автоматизированы, то единственным неудобством такой системы связи является невозможность передать с ее помощью ничего материального.

Прощай, яблочко?

Фирма Apple («Яблоко»), чьи компьютеры Apple и Macintosh (название одного из сортов яблок) известны во всем мире, приступила к выпуску рабочих станций серии Quadra и портативных компьютеров Powerbook.

Кто там?

Задать этот вопрос, прежде чем открывать дверь, никогда не вредно. Но нет ли более эффективного способа обезопасить себя? Поразмыслив над этим, японская фирма EST Co., Ltd выпустила несколько вариантов электронного швейцара, заменяющего сразу пять глазков (по числу предполагаемых дверей).

Связь между телекамерами, устанавливаемыми над дверьми, и центральным пунктом — беспроволочная, так что перерезать провода врагу не удастся. Управляет охранным радиотелекомплексом скромная БИС, которую вполне можно считать полноценным компьютером, поскольку она оборудована всем, что подобает иметь приличной ЭВМ: памятью, процессором и даже периферийными устройствами в виде дистанционно управляемых засовов и замков.

Теперь, услышав звонок в дверь или сирену сигнализации, вы подходите к маленькому экрану с изображением пришедшего и, если по его внешнему виду непонятна цель прихода, уточняете ее. Хотя иногда именно за такими дорогими приборами и приходят незванные гости.

И. НАРЫКОВА

Компьютерное моделирование в Великобритании

Учитывая интерес специалистов в области информатизации образования к сравнительно новому для нас направлению — моделированию, хотелось бы предложить читателям небольшой обзор работ ученых Великобритании по этому вопросу.

История использования компьютерного моделирования в педагогической практике Великобритании ведет свой отсчет с конца 70-х гг. Этот процесс носил характер постепенного расширения разработок компьютерных моделей (часто в имитационной оболочке) и их использования в учебном процессе, хотя при этом учебные планы не претерпевали сколь-нибудь значительных изменений. Так продолжалось до 1989 г., когда решением на правительственном уровне моделирование было включено в национальные учебные планы по информационной технологии. В них моделирование рассматривалось как некая совокупность знаний и умений, которые учащиеся сначала изучают теоретически, а затем используют практически. Для возрастной группы 14—16 лет была проведена градация знаний и умений по уровням. Не приводя ее здесь целиком, отметим, что на начальных уровнях от учащихся требуется уметь использовать готовые модели для определения и исследования объектов, переменных и соотношений между ними. На последующих уровнях требуется умение построить модель для данной системы и проверить ее на практике. Логика такого подхода к введению метода моделирования в учебные планы — обычная: от простого к сложному, от формирования знаний и их закрепления — к применению. Однако многим английским ученым-педагогам эта концепция кажется неприемлемой, так как общая тенденция современной педагогики — переход от традиционного обучения, рассматриваемого как передача некоторой суммы знаний и умений, к развивающему, широко использующему достижения когнитивной психологии.

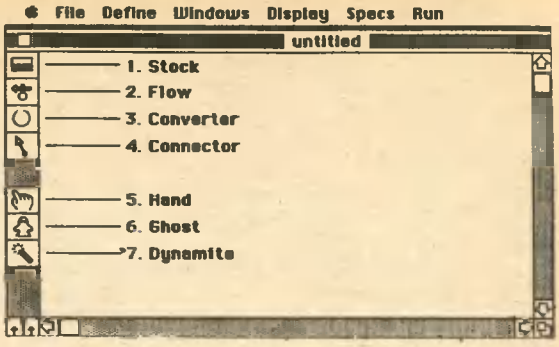
Х. Меллар, специалист в области использования моделирования в обучении, считает, что моделирование в национальных учебных планах по информационной технологии рассматривается как некий технологический алгоритм, последовательность шагов которого должна быть изучена и освоена учащимися до практического использования. Но, как утверждает Х. Меллар, «уже до прихода в школу дети конструируют свои собственные модели, если и не с помощью ЭВМ, то в голове, на бумаге или посредством других материалов. Компьютерное численное моделирование тоже может быть использовано детьми для выражения своих собственных идей, которые таким образом становятся доступными для исследования другими. Однако, вместо того, чтобы начать с тех аспектов моделирования, которые знакомы учащимся, учебные планы настраивают учителей на представление детям готовых компьютерных моделей, созданных взрослыми... После этого учащимся очень трудно провести параллели между своими собственными идеями и готовыми моделями, в результате чего теряются возможности обучения детей выражать свои идеи численно с помощью компьютера» [1]. Таким образом, по крайней мере в младших и средних классах общеобразовательной школы необходимо рассматривать моделирование скорее как способ познания, интуитивно используемый детьми в жизни и обучении, чем как некоторую терра инкогнита, которую еще только предстоит освоить. Задача педагога при этом подходе заключается в том, чтобы придать процессу моделирования большую осознанность и добиться большего КПД в развитии мышления учащихся. Позитивная сторона — возможность достижения более полного и глубокого понимания учащимися себя и окружающего мира, которая обеспечивается уникальными возможностями компьютерного моделирования воплощать идею в некоторую структу-

ру и исследовать собственный результат.

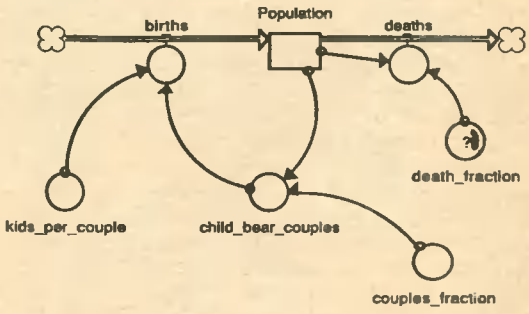
Вполне понятно, что эти возможности обеспечиваются специальными программными средствами, которые развиваются очень быстро [3] и становятся серьезными конкурентами имитационным компьютерным программам (simulation). Последние строятся на основе готовой модели, параметры которой либо недоступны, либо ограниченно доступны для пользователя. В силу того, что работа с такими программами как раз и означает знакомство с чужими моделями, имитационные программы по различным предметам должны были явиться базовым программным обеспечением для обучения моделированию по схеме, предложенной в национальных учебных планах на начальных уровнях. Не ставя под сомнение необходимость и целесообразность создания имитационных программ и их роль в преподавании таких предметов, как физика, химия, биология, география и др., критики правительственного подхода, однако, делают основную ставку на те программы, которые предоставляют пользователю возможность создавать свои собственные модели, или существенно раздвигают рамки имитационных программ [2].

Сейчас для численного моделирования динамических систем все чаще применяют доступные и широко распространенные электронные таблицы, особенно с графическими возможностями. Наконец, созданы и широко используются инструментальные программные средства (ИПС), специально предназначенные для компьютерного моделирования динамических систем. На одном из них, разработанном в Гарвардском университете (США), использующем богатые возможности графического интерфейса Apple Macintosh и названном STELLA, хотелось бы остановиться особо.

STELLA (СТЕЛЛА) — это профессиональный пакет прикладных программ для моделирования динамических систем. Есть версия этого пакета и для образования. Так как конечной целью являются расчеты, созданная модель включает: 1) структурную диаграмму; 2) уравнения, связывающие переменные и константы; 3) другие данные, необходимые для расчетов (начальные условия, значения параметров и др.). Сначала модель составляется в виде структурной диаграммы с использованием четырех рабочих элементов. На рис. 1, воспроизводящем рабочий экран программы, они изображены вверху у левой границы экрана. Ниже, у этой же границы, изображены три основных «инструмента» СТЕЛЛЫ, которые позволяют устанавливать и уничтожать структурные элементы.



Смысл структурных элементов лучше всего разобрать на примере уже готовой модели, например модели популяции. Модель на рис. 2 составляется в одном из четырех окон программы, которое называется «диаграммное окно». Основным структурным элементом модели является «сток» (stock), изображенный прямоугольником. Этот символ применяется для обозначения всего, что способно накапливаться (вода в емкости, деньги на счету и пр.). Для обозначения путей изменения стока применяется структурный элемент «поток» (flow), который имеет «облако» на одном конце, стрелку, указывающую на направление потока, на другом, и механизм, регулирующий величину потока (кружок с насосом). В нашем случае величинами, которые влияют на размер популяции, являются рождения (births) и смерти (deaths). Облака на концах входящего и исходящего потоков символизируют границы модели.



Если на минуту отвлечься от конструирования модели и посмотреть, что происходит за кулисами программы в процессе составления модели (для этого надо сменить диаграммное окно на окно для уравнений), то мы увидим, что сама программа создает инфраструктуру уравнений, необходимых для расчетов по модели. Правило, используемое программой для этого, гласит: «величина стока в данный момент времени равна величине, которая существовала мгновение назад плюс значение произведения длины мгно-

вения на разность значений входного и выходного потоков». В нашем случае программа сама запишет уравнения типа:

популяция=популяция+(рождения — смерти)·dt

рождения=....

смерти=....

Последние два уравнения остаются незаконченными, пока их не заполнит сам пользователь программы.

Займемся дальнейшим описанием модели. Нам необходимо дать полную спецификацию процессов притока (рождения) и оттока (смерти) из стока «популяция», т. е. включить в модель все факторы и переменные, влияющие на эти процессы. Очевидно, что процесс, генерирующий рождения, определяется существованием гетеросексуальных пар детородного возраста, количество которых само зависит от величины популяции. Этот фактор — «число пар» (child bear couples) изображен в модели графически с помощью третьего структурного элемента — «конвертера» (кружок). Связь величины популяции с числом пар — стрелка, называемая «коннектор»,— четвертый структурный элемент. Для простоты условимся, что каждая пара будет давать жизнь определенному числу детей в год, т. е. величина «дети на одну пару» (kids per couple) есть константа нашей модели, которая также изображается конвертером. Введем еще одну константу: предположим, что число пар есть какая-то постоянная часть популяции. Это предположение отображается на модели параметром «доля пар» (couples fraction). Аналогично дается в модели спецификация процессов смерти: вводится параметр «доля смертей» (death fraction) для обозначения зависимости количества смертей от величины популяции. После составления структурной диаграммы модели необходимо дописать все необходимые алгебраические уравнения, значения констант и начальные значения всех стоков в модели. В нашем случае это:

число рождений=«число пар»×«число детей на одну пару»

«число детей на пару» (в год)=0,1 (константа)

«доля пар»=0,35 (константа)

«доля смертей»=0,01 (константа)

«популяция»=100000 (начальное значение)

Большим достоинством программы является возможность задавать переменные и константы, необходимые для расчетов, как сложные нелинейные графические функции. Так, в нашем примере мы могли задать «долю смертей» как переменную величину в зависимости от величины популяции, тогда расчет по модели неизбежно приводил бы к коллапсу популяции при некоторых значениях параметра «доля смертей».

Последним этапом работы является проведение расчетов по модели (ее «запуск» в режиме имитации). Для этого производится предварительное шкалирование всех переменных, которое необходимо для вывода результатов в графической форме. Во время имитации в программе вводится условное время, масштаб которого устанавливается и изменяется пользователем программы. После запуска модели в имитационном динамическом режиме пользователь получает все необходимые ему данные либо в числовой, либо в графической форме. Например, по нашей модели можно рассчитать величину популяции, рождаемость, смертность через определенный промежуток времени.

Программа СТЕЛЛА может служить для моделирования многих других физических, химических, биологических, экономических закономерностей, хотя, возможно, аналогии и метафоричность ее структурных элементов (сток, поток и т. д.) не всегда будут столь явно выражены, как в разобранном примере с популяцией. Например, в динамической модели механической системы стоком будет импульс тела, так как определение импульса как количества движения показывает, что он может накапливаться. В этом случае сила — та переменная, которая регулирует входной поток в сток «импульс тела», ибо изменение импульса тела равно произведению силы, действующей на тело, на время действия.

Таким образом, СТЕЛЛА представляет собой весьма мощное средство моделирования динамических систем, не требующее от пользователя хорошего знания дифференциального исчисления. Наоборот, в процессе работы с программой он может приобрести некоторые навыки мышления такими категориями математики, как интегральная величина, первая производная как скорость ее изменения и т. д. Вообще, фундаментальное значение компьютерного моделирования динамических систем, по мнению Дж. Отборна, состоит в возможности разбивать расчет на элементарные шаги (итерация). Это увеличивает «наглядность и доступность расчета для понимания и способствует ликвидации разрыва между формулированием дифференциального уравнения и использованием в дальнейшем аналитической формы его решения для расчетов, как это обычно делается в курсах физики и химии. Более того, численные модели функционируют так же, как природа: она ничего «не знает» об эллипсах и параболах, но «формирует» их шаг за шагом с течением времени, точно так же, как это делается во время численного моделирования» [2].

Второй тип компьютерных моделей, использующийся в системе образования,— это так называемые клеточные модели, подобные клеточным автоматам фон Неймана. Такая модель может быть представлена как множество клеток (например, изображенных на экране дисплея), каждая из которых имеет ограниченное число состояний (обычно два) и изображает какой-либо объект. Состояние клетки, а следовательно и объекта, в некоторый момент времени определяется собственным состоянием в предыдущий момент и состоянием соседних клеток.

Коренными отличиями клеточных моделей от моделей динамических систем являются следующие. Во-первых, эволюция системы в клеточной модели задается с помощью локального правила (а не дифференциального уравнения), которое связывает событие в клетке с ее предыдущим состоянием и состоянием ее соседей и является справедливым для любых объектов данной системы. Во-вторых, это правило описывает поведение объектов в модели, а не математические зависимости. Компьютерные клеточные модели позволяют обучаемому, наблюдающему эволюцию моделируемой системы в режиме имитации, получать информацию о поведении ансамбля объектов.

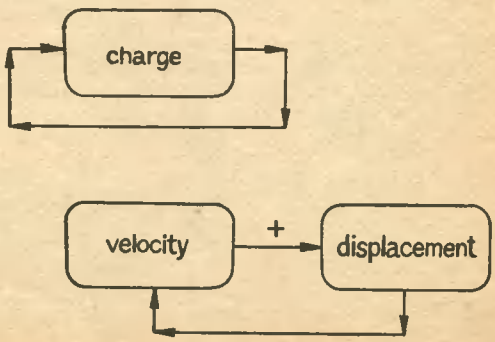
Несмотря на кажущуюся сложность, клеточные модели находят широкое применение в обучении. Классические проблемы, которые могут быть описаны с их помощью,— это различные популяционные задачи типа «хищник-жертва». При моделировании такой системы каждая клетка на экране дисплея будет изображать либо хищника, либо жертву. Программа имеет правила уничтожения ближайшей жертвы хищником и изменения численности обеих популяций за счет процессов рождения и смерти. При запуске такой модели, популяции на экране будут изменяться вследствие отдельных событий, происходящих в клетках.

Примером клеточной модели из другой предметной области является модель процесса диффузии, в которой множество клеток на экране дисплея состоит из клеток пустых и содержащих молекулы. Правило, управляющее работой модели в режиме имитации гласит, что молекула диффундирующего вещества может передвигаться только в соседнюю пустую клетку, а если таковых больше одной, то направление движения выбирается случайным образом. На такой компьютерной модели можно хорошо продемонстрировать разницу процессов на макро- и микроуровнях: систематическое, необратимое изменение на макроуровне может быть вызвано случайными, не систематическими процессами в веществе на микроуровне.

Преимущество клеточных моделей видится специалистам в том, что они обладают большей наглядностью, чем модели динамических систем, поскольку в клеточной модели обучаемый имеет дело с объектами, а не с их описанием на языке математики. Однако с этим связаны значительные трудности при разработке инструментальных программных средств для создания компьютерных клеточных моделей, которые позволяли бы учащемуся самому задавать объекты модели, состояния объектов и правила их изменения.

Третий тип моделей, описанный Дж. Огборном [2],— это так называемые полуколичественные (semi — guantitive) модели, применяемые для моделирования ситуаций, о которых известно значительно меньше, чем необходимо для построения точной количественной модели с помощью дифференциальных уравнений. Для создания таких моделей необходима лишь информация об общем характере процессов, происходящих в моделируемой системе, т. е. информация типа «если величина X будет расти, то Y уменьшится». Очевидно, что получисленное моделирование на компьютере имеет ряд преимуществ, основное из которых — возможность его использования в младших и средних классах школ для реализации учащимися своих собственных идей и моделей в соответствии с общим подходом, описанным в этой статье. Уже создано и используется в учебном процессе программное обеспечение для создания таких моделей в виде диаграмм с указанием причинно-следственных связей между событиями в системе. Для учащегося построение такой модели означает выбор и наименование переменных и обозначение стрелками влияния одних переменных на другие. В моделях обычно используются два основных типа такого влияния: «чем больше, тем больше» (знак +) и «чем больше, тем меньше» (знак —).

Очень важно, что такие качественные модели-диаграммы могут иллюстрировать некоторые важные общие свойства систем. На рис. 3, например, верхняя диаграмма (модель



изменения заряда на конденсаторе при разряде) иллюстрирует систему, всегда стремящуюся к положению равновесия после возмущения. Таковы многие системы с единственной отрицательной обратной связью. Они описываются экспоненциальными функциями. На нижней диаграмме (модель связи между скоростью и смещением маятника) изображена модель системы, которая всегда будет стремиться к осцилляции вследствие наличия двух связей, действующих в противоположном направлении. Такие системы описываются синусоидальными функциями.

Две модели — два принципиально разных типа систем. Данные модели взяты из физики, но и в других предметах можно выделить аналогичные системы. Таким образом, если учащиеся часто будут иметь дело с подобными качественными моделями, это может быть хорошим введением в численное компьютерное моделирование этих же систем на последующих стадиях обучения.

Литература

1. *Mellar H.* Creating alternative Realities: computer, modelling and curriculum change. 1990.
2. Oqborn T. Modelling with the computer: possibilities and prospects.
3. *Невуева Л., Сергеева Т.* О перспективных тенденциях разработки программных средств // Информатика и образование. № 5. 1990.

Уважаемые читатели «ИНФО»!

1 августа начнется подписная компания. Надеемся, Вы останетесь нашими читателями.

Индекс журнала	Цена		
70423	2 мес.	4 мес.	6 мес.
	50 руб.	100 руб.	150 руб.

Редакция ищет возможность организовать подписку по безналичному расчету. Следите за нашими объявлениями.

ЧТО МОЖЕТ ЭВМ?

Плеер для интеллектуалов

Он похож на видеоплеер, во-первых, тем, что проигрывает видеодиски, во-вторых, размерами: 13×10×4 см, вес 800 г. Но это прибор не для развлечения: «электронная книга» Data Diskman фирмы Sony предназначена для чтения в «походных» условиях информации с компьютерных оптических дисков диаметром 3 дюйма, вмещающих по 200 Мбайт данных. Дисплей — жидкокристаллический, с диагональю 3,5 дюйма, подсвечиваемый, отображает 10 30-символьных строк или графическое изображение 256×200 точек. Клавиатура — стандартная, хотя и миниатюрная; программное обеспечение — фирменное, позволяет легко и быстро отыскать на диске нужную информацию.

Карманный принтер

Фирма Citizen предложила покупателям принтер PN 48, вполне соответствующий духу эпохи портативных компьютеров: вес 900 г (без аккумуляторов), размеры 5×9×29 см. Скорость печати не очень высока, 80 символов в секунду, зато качество — 360 точек на дюйм, лучше, чем у среднего лазерного принтера. Единственный недостаток — аккумуляторы: комплекта весом 270 г хватает лишь на 25 страниц.

Видеотоварищество

Как можно стать членом Видеотоварищества

Видеотоварищество набирает коллективы учащихся для совместной работы по созданию видеосюжетов на 1992/93 учебный год. Для того чтобы вступить в Видеотоварищество, необходимо представителю школы заполнить регистрационную форму с подписью ответственного лица и печатью школы.

Адрес и № школы _____

Класс (возраст ребят-участников) _____

Количество участников _____

Руководитель ученического коллектива — участника Видеотоварищества _____
(фамилия, имя, отчество; адрес; телефон)

Предпочитаемые темы (если они есть) _____

Предпочитаемые партнеры (указать город, страну) _____

Заполненную форму необходимо вложить в конверт и отправить по адресу: 119034, Москва, Смоленский бульвар, д. 4, «ИНФО».

В течение 4—6 недель Видеотоварищество ответит о включении Вашей школы в движение, сообщит сумму взноса и адрес, куда ее надо перечислить, а также адрес Вашего партнера.

Видеотоварищество было основано в 1990 г. Ричардом Реем, который предложил американским школьникам организовать своеобразную переписку с ребятами других стран — обмениваться видеосюжетами, снятыми по собственному сценарию. На базе одной из видеостудий учебных программ и началась эта увлекательная работа. Интерес к движению, а именно так можно назвать Видеотоварищество, стремительно рос. Его участниками становились ученики, семьи, школы.

Видеотоварищество позволяет объединить учащихся из различных городов, регионов, стран в работе над общими проектами. Между детьми устанавливаются дружеские контакты, открываются новые возможности для изучения языка, культуры, жизни и быта других народов.

Учителям движение предоставляет возможность организовать совместную работу с коллегами и учащимися не только своей школы. Расширяются рамки школьного урока. Раздвигаются горизонты для обмена опытом, формируется культура общения.

Школы, ставшие членами Видеотоварищества, имеют уникальную возможность выбора партнера, участия в создании совместных видеосюжетов на темы школьной жизни, проведения конкурсов.

Лучшие из видеосюжетов за рубежом показывают в образовательных программах телевидения и рекламируют в газетах.

В нашей стране инициатором создания Видеотоварищества выступила преподаватель русского языка и литературы, научный сотрудник НИИ СО и УК РАПН Е. Н. Ястребцева.

В мае 1990 г. в СШ № 1282 Москвы был снят первый видеосюжет — знакомство, который затем был отправлен в школы г. Когалыма (Ханты-Мансийский округ), г. Кировограда (Украина), г. Ташауза (Туркмения) и г. Сан-Диего и г. Меномони (США). Сюжет демонстрировал жизнь школьников в часы досуга, на спортивных площадках, в музыкальной школе, танцплощадке, на общешкольных праздниках и пр. Из школ США получены ответные видеосюжеты—знакомства.

В 1991 г. учащиеся-партнеры подготовили к обмену видеосюжеты на тему: «Народные праздники и обычаи в сказках и фольклоре». Позже появились фильмы: «Литературная пародия», «Женщины на русском престоле». И ребята, и взрослые работали с большим увлечением, открывая для себя и в себе мир творчества.

Слабая материальная база нашей школы является серьезным препятствием для внедрения многих начинаний, в том числе и движения Видеотоварищество. Тем не менее включаться в подобные проекты необходимо, иначе мы рискуем отстать в понимании современных педагогических тенденций.

Редакцию ИНФО заинтересовал проект Видеотоварищество, например мы думаем, что организация съемки и обмена сюжетами об использовании компьютеров в школе может оказаться увлекательным и полезным делом.

Организацией и координацией работы Видеотоварищества занимается инициативная группа, возглавляемая Е. Н. Ястребцевой, а заявки на участие в проекте предлагаем присылать в адрес редакции с пометкой «Видеотоварищество».

Заседание Координационного совета

При Республиканском институте повышения квалификации работников образования (РИПКРО) организован Координационный совет по проблемам повышения квалификации педагогических кадров в области информатизации образования.

В феврале этого года состоялось первое заседание Координационного совета, решение которого публикуется ниже.

Основные проблемы, связанные с компьютеризацией образования и внедрением новых информационных технологий обучения хорошо известны, а именно:

неопределенность принципов государственного финансирования и поддержки этого направления;

отсутствие унифицированных средств обучения (программно-аппаратно-совместимых);

отсутствие четкого перечня нормативных документов, регламентирующих деятельность административных служб поддержки народного образования в части компьютеризации;

слабая материальная база кафедр, кабинетов, лабораторий в системе повышения квалификации работников образования, как основных звеньев внедрения НИТ в систему образования;

отсутствие надежных систем оперативного обмена информацией и научно-исследовательских разработок в сфере образования.

В связи с этим Координационный совет постановляет:

1. Предложить МО РФ издать перечень нормативных документов, определяющих комплектацию школ и ИПК (ИУУ) учебно-методическими материалами и техническими средствами обучения, необходимыми для успешного внедрения НИТ в образовании и регламентирующих деятельность администрации всех звеньев народного образования для приоритетной поддержки этого направления.

2. Считать целесообразным начинать преподавание курса ОИВТ с VIII класса в тех учебных заведениях, где создана необходимая материальная база и подготовлены педагоги. С этой целью внести в учебные планы на 1992—93 учебный год необходимые изменения.

3. Поручить авторским коллективам учебников по ОИВТ разработать программы преподавания курса ОИВТ в школе с VIII класса, и, как следующий этап, использование компьютеров в начальной школе.

4. С целью более эффективного и широкого использования НИТ в образовании, рассмотреть возможность введения в школы должности заместителя директора по внедрению новых информационных технологий в образование.

5. При аттестации учителей с целью дифференцированной оплаты труда, в качестве одного из критериев учитывать использование ими НИТ в учебном процессе.

6. Рассмотреть вопрос о результатах научных разработок, проводимых системой научно-исследовательских учреждений, в области информатизации и внедрении их через РИПКРО и региональные ИПК (ИУУ).

7. Провести Всероссийскую конференцию о результатах экспериментальной работы по проекту «Пилотные школы» и другим проектам.

8. Создать при РИПКРО экспертную комиссию педагогических программных средств и учебно-методических разработок для образования РФ.

9. На базе ИПК (ИУУ) создать информационную сеть обмена учебно-методическими материалами с целью формирования банка знаний по использованию НИТ в образовании, с центром в РИПКРО.

10. Рекомендовать МО РФ создание телекоммуникационной службы для оперативного обмена информацией между центром и регионами страны с учетом необходимых бюджетных ассигнований.

11. Рекомендовать принять в качестве типового «Положение об учебно-методическом кабинете ИВТ», разработанное Координационным Советом по информатике при РИПКРО.

Разъясняем нормативные документы

В связи с многочисленными обращениями по вопросу установления доплат за неблагоприятные условия труда работникам учреждений и организаций народного образования и науки РФ мы хотим познакомить читателей с приказом Минобразования РФ № 61 от 28.02.92 г. «О нарушениях норм и правил охраны труда и создании безопасных условий жизнеобеспечения в учреждениях Минобразования РФ»*. Приведем выдержки из него, касающиеся работы учителей ОИВТ.

В работе учреждений системы образования Российской Федерации выявлены серьезные недостатки в создании безопасных условий в учебно-воспитательном процессе.

В условиях массовой компьютеризации процессов обучения в ряде учреждений не уделяется должного внимания выполнению санитарно-гигиенических норм, правил, содержания и работ на электронно-вычислительных машинах.

При этом отмечаются случаи невыплат доплат работникам, занятым на работах с вредными условиями труда, в учреждениях и организациях системы Минобразования Российской Федерации (Рязанская, Магаданская обл.).

Приказываю:

1. Руководителям органов управления образованием республик в составе Российской Федерации, краев, областей и округов, городов Москвы и Санкт-Петербурга:

1.6. Обеспечить по каждому учебному заведению:

1.6.1. Назначение приказом ответственных лиц за соблюдение охраны труда, санитарных норм и правил во всех помещениях учебного заведения, включая вспомогательные строения.

1.6.3. Выполнение «Временных санитарно-гигиенических норм и правил устройства, оборудования, содержания и режима работы на персональных электронно-вычислительных машинах и видеодисплейных терминалах в кабинетах вычислительной техники и дисплейных классах всех типов средних учебных заведений» СанПиН № 5146-89.

1.6.4. Установление работникам, занятым на работах с вредными условиями труда, в учреждениях и организациях системы Минобразования Российской Федерации, доплат согласно приказу Государственного комитета СССР по народному образованию от 20.08.90 г. № 579 и рекомендациям ЦК профсоюза работников народного образования и науки РСФСР от 18.06.91 г. № 187/4-14 пк.;

В дополнение к приказу приводим читателям разъяснения* о порядке применения «Положения о порядке установления доплат за неблагоприятные условия труда», утвержденного приказом Госкомитета СССР по народному образованию от 20.08.90 г. № 579 и доведенного до местных органов народного образования письмом Министерства образования РСФСР от 26.10.90 № 1051/13.

Согласно данному Положению, доплаты за неблагоприятные условия труда устанавливаются лицам, непосредственно занятым на работах, предусмотренных Перечнем работ с неблагоприятными условиями труда (приложение № 2 к приказу Гособразования СССР от 20.08.90 г. № 579), по результатам аттестации рабочих мест или оценки условий труда и начисляются по фактически отработанному в неблагоприятных условиях времени.

Аттестация рабочих мест или оценка условий труда проводится экспертной комиссией на основе данных специальных инструментальных замеров уровней факторов производственной среды на рабочих местах. При ограниченной возможности проводить инструментальные замеры допускается в порядке исключения применение метода экспресс-оценки состояния условий труда по критериям, приведенным в приложении 1.

Экспертная комиссия образуется совместным решением органа по управлению образованием района (города) и соответствующего комитета профсоюза. В состав комиссии входят наиболее квалифицированные сотрудники аппарата управления (отдела), представители рай(гор)кома профсоюза, специалисты по охране труда с привлечением представителей СЭС, других служб и организаций. Экспертную комиссию возглавляет заместитель руководи-

* Приказ подписал первый зам. министра образования РФ В. Б. Новичков

* Разъяснения утвердили первый заместитель министра образования РФ В. Б. Новичков и заместитель председателя ЦК профсоюза работников народного образования Г. И. Меркулов.

теля районного, городского управления (отдела) народного образования.

Руководитель учреждения, организации, предприятия народного образования по согласованию с профсоюзным комитетом на основании перечня работ с неблагоприятными условиями труда представляет к началу учебного (календарного) года перечень работ по учреждению с указанием вредных производственных факторов по каждому виду работ (приложение 2).

На основании заключения экспертной комиссии руководитель органа управления народным образованием района (города) по согласованию с соответствующим комитетом профсоюза утверждает перечень работ с неблагоприятными условиями труда.

Перечень работ с неблагоприятными условиями труда должен ежегодно пересматриваться с учетом улучшения условий труда на рабочих местах.

Определение фактического состояния условий труда на рабочих местах с неблагоприятными условиями труда

Фактическое состояние условий труда оценивается на тех рабочих местах, где выполняются работы, предусмотренные Перечнем работ с неблагоприятными условиями труда, на которых устанавливаются доплаты рабочим, специалистам и служащим с тяжелыми и вредными, особо тяжелыми и особо вредными условиями труда. Степень вредности факторов производственной среды, уровня тяжести и напряженности работ определяется в баллах по критериям, примененным в Гигиенической классификации труда, утвержденной Минздравом СССР от 12.08.86 г. № 1437-86.

Баллы, установленные по степени вредности факторов, уровня тяжести или напряженности работ, рассчитываются по формуле:

$$X_{факт} = X_{вред} \cdot T, \text{ где}$$

$X_{факт}$ — фактическое состояние условий труда на рабочих местах;

$X_{вред}$ — степень вредности фактора, тяжести или напряженности работ, установленная по показателям Гигиенической классификации труда;

T — отношение времени действия данного вредного фактора к продолжительности общего рабочего времени. (Если время действия этого фактора свыше 90 % рабочего времени, то $T=1$).

Фактор оценки фактического состояния условий труда приводится в приложении 3.

Администрация учреждений ведет журнал учета рабочего времени работника в неблагоприятных условиях труда (приложение 4). На основании данных журнала руководитель учреждения ежемесячно представляет в отдел (управление) народного образования для оплаты список лиц, занятых в неблагоприятных условиях труда, с размером доплат (приложение 5). Доплата осуществляется за счет выделенного учреждению фонда оплаты труда.

Приложение 1

Критерии для экспресс-оценки состояния условий труда

Факторы	Описание производственной ситуации	Количество баллов
Вредные химические вещества	Воздух на рабочем месте загрязняется веществами 1—2-го класса опасности, имеется вытяжная вентиляция	1
	Воздух на рабочем месте загрязняется веществами 3—4-го класса опасности, вытяжная вентиляция отсутствует	1
	Воздух на рабочем месте загрязняется веществами 1—2-го класса опасности, вытяжная вентиляция отсутствует	2
Пыль	Воздух загрязняется пылью, содержащей SiO_2 (при наличии вытяжной вентиляции)	1
	Воздух загрязняется пылью, содержащей SiO_2 (при отсутствии вытяжной вентиляции)	2

Примечание. Для определения степени вредности $X_{вред}$ (баллов) по шуму, инфракрасному излучению, скорости движения, влажности, температуры воздуха экспресс-оценки условий труда не применяется. Необходимо производить инструментальные замеры.

Приложение 2

Перечень работ с неблагоприятными условиями труда по _____
(учреждение, предприятие, организация)

в _____ **году**

Должность	Место работы (кабинет, № комнаты, помещение)	Условия работы с указанием вредных производственных факторов	$X_{вред}$ (заполняется экспертной комиссией)

Приложение 3

Таблица по определению размеров доплат за неблагоприятные условия труда

Фактическое состояние условий труда $X_{факт}$ (баллов)	Размеры доплат в % к окладу (тарифной ставке)
На работах с опасными, вредными и тяжелыми условиями труда	
до 2,0	4
2,1—4,0	8
4,1—6,0	12
На работах с особо опасными, особо вредными и тяжелыми условиями труда	
0,1—8,0	16
8,1—10,0	20
более 10.0	24

Приложение 4

Журнал учета времени работы сотрудников _____ **в неблагоприятных условиях труда**
(учреждение)

Начат _____ **199**____ **г.**
Окончен _____ **199**_____ **г.**

Дата	Место работы	Вид работы	Фамилия, Имя, Отчество	Начало работ	Окончание работ	Количество часов за день	Подпись сотрудника	Подпись руководителя

Приложение 5

Список лиц, занятых на работах с неблагоприятными условиями труда
в _____ **по** _____
(месяц, год) (учреждение)

Фамилия, Имя, Отчество	Должность	Место работы (кабинет, комната, помещение)	$X_{вред}$ (по данным экспертной комиссии	Т	$X_{факт}$	Размер доплаты в % к окладу тарифик.	Размер доплаты в рублях

Критерии оценки состояния условий труда при работе на ЭВМ

Судя по письмам читателей, на местах возникает много вопросов по критериям оценки состояния условий труда при работе на ЭВМ. Редакция обратилась за разъяснениями в Минобразование РФ. Приводим комментарий ведущего специалиста по вопросам охраны труда и техники безопасности Государственной инспекции при Минобразовании РФ В. И. Алесенкова.

Для реализации программ компьютеризации обучения в средней школе производится выпуск отечественных школьных ЭВМ, а также закупка импортных ПЭВМ. При гигиеническом обследовании 32 типов зарубежных и 7 типов отечественных персональных ЭВМ, предназначенных для общеобразовательных школ, проведенном НИИ общей и коммунальной гигиены им. А. Н. Сысина АМН РФ и НИИ дефектологии АПН РФ, было обнаружено, что только 7 типов зарубежных ЭВМ отвечают современным гигиеническим нормам. Ни один из отечественных видеотерминалов не соответствует существующим отечественным гигиеническим требованиям.

При изучении характера и условий работы на ЭВМ установлено, что по степени развития утомления эта работа стоит на втором месте после работы водителей автобусов.

Во многих случаях у работающих на ЭВМ отмечены жалобы на усталость глаз, боли в глазных яблоках, слезотечение, головную боль. Зарегистрированы и различные функциональные нарушения состояния зрения, а также случаи заболевания конъюнктивитом.

По данным исследований повышенное зрительное напряжение при работе на ЭВМ обусловлено рядом неблагоприятных факторов: постоянной переадаптацией глаз в условиях различной яркости, недостаточной четкостью и контрастностью изображения на экране, строчностью его структуры, яркостными мельканиями. При этом зрительное напряжение усугубляется наличием блестящих пятен в результате отражения светового потока на клавиатуре и экране, неравномерностью освещения рабочих поверхностей, большим перепадом яркости между рабочей и окружающими поверхностями.

Постоянная статическая нагрузка при работе с дисплеями, обусловленная относительно неподвижной рабочей позой, может привести к усталости и болям в мышцах рук, шеи, плеч и спины, к нарушениям опорно-двигательного аппарата.

Напряженность электрического поля между экраном ЭВМ и оператором составляет 5—15 кВ/м, что приводит к загрязнению воздуха в помещении.

С учетом всех изложенных факторов работа на ЭВМ приравнена к неблагоприятным условиям труда, на которых устанавливается доплата до 12 %.

В связи с многочисленными обращениями по вопросу установления доплат преподавателям информатики и вычислительной техники приводим критерии оценки состояния условий труда для работников данной категории.

Шум

Уровень шума дисплейного класса, свободного от учащихся и ВТ, не должен превышать 40 дБА (СНиП 11-12-77 «Защита от шума. Нормы проектирования»), а уровень шума на рабочих местах не должен превышать 50 дБА (ГОСТ 12.1.003-83 «Шум. Общие требования безопасности» и «Санитарные нормы допустимых уровней шума на рабочих местах» № 3223-85, утвержденные Минздравом СССР 12.03.85 г.), а уровни звукового давления в октавных полосах со среднегеометрическими значениями частот: 63, 125, 250, 500, 1000, 2000, 4000, 8000 Гц — соответственно 63, 52, 15, 39, 35, 32, 30, 28 дБ.

Для снижения уровня шума потолок и стены должны облицовываться звукопоглощающим материалом с максимальным коэффициентом звукопоглощения в области частот 63—8000 Гц.

Микроклимат

Оптимальными параметрами температуры при почти неподвижном воздухе являются 19—21 °С, допустимыми 18—22 °С.

Оптимальное влагосодержание составляет 10 г/м^3, допустимое не ниже 6 г/м^3.

Для обеспечения надежного микроклимата и качественного состава воздуха, включая аэрорежим, необходимо систематически перед началом занятий и после каждого академического часа, независимо от погоды, осуществлять проветривание длительностью не менее 10 мин.

Другой путь обеспечения воздухообмена может быть достигнут установлением в оконных проемах автономных кондиционеров типа БК-1500, БК-2500, БК-2000Р.

Расчет необходимого количества кондиционеров должен проводиться по теплоизбыткам.

Состав воздуха

Содержание кислорода в дисплейном классе должно быть в пределах 21—22 об.%. Двуокись углерода не должна превышать 0.1 об.%, озон — 0,1 мг/м^3, аммиак — 0,2 мг/м^3, фенол — 0,01 мг/м^3, хлористый винил — 0,005 мг/м^3, формальдегид — 0,003 мг/м^3.

Количество легких (положительных и отрицательных) ионов должно соответствовать оптимальным значениям: для положительных — 1500—3000, для отрицательных — 3000-5000. Коэффициент полярности (отношение разности числа положительных и отрицательных ионов к их сумме) должен находиться от —0,5 до 0 (см. «Санитарно-гигиенические нормы допустимых уровней ионизации воздуха производственных и общественных помещений» № 2152-80, утв. Минздравом СССР 12.02.80 г.). Для улучшения качественного состава воздуха в дисплейном классе пол должен иметь поливинилхлоридное антистатическое покрытие (ТУ 21-29-108-84). Двери и стенные шкафы могут быть облицованы поливинилхлоридным декоративным антистатическим материалом (ТУ 400-20-38-3-82).

Запрещается применять для отделки интерьера дисплейного класса строительные материалы, содержащие органическое сырье: древесностружечные плиты (ДСП), декоративный бумажный пластик, поливинилхлоридные пленки, моющиеся обои и др.

Освещенность

Уровень искусственной освещенности на рабочих местах (на клавиатуре и поверхности стола) должен находиться в пределах 300-500 лк.

Освещенность на экране дисплеев не должна превышать 200-250 лк.

На уровень освещенности дисплейного класса оказывает влияние цветовая отделка интерьера и оборудования, их отражающая способность. Для окраски стен и панелей рекомендуется применять светлые тона красок (коэффициент отражения 0.5-0.6). Нельзя окрашивать стену, расположенную напротив экрана монитора, более темными тонами красок (0.3-0.4).

Другие факторы

Уровень неиспользованного рентгеновского излучения на расстоянии 5 см от экрана и других поверхностей ЭВМ не должен превышать 100 мкР/ч.

Напряженность электростатического поля на рабочем месте при работе на видеотерминале должна быть не более 15 кВ/м.

Плотность потока ультрафиолетового излучения не должна превышать 10 Вт/м^3.

Характеристики улучшения экрана в рентгеновском, ультрафиолетовом, инфракрасном и радиочастотном диапазонах должны соответствовать гигиеническим нормам для населения (НРБ-76, осп — 7/79, ГОСТ 12.2.003-74, ГОСТ 12.3.002-75. «Санитарные правила работы с источником неиспользуемого рентгеновского излучения», Минздрав СССР, 1982).

Содержание ионов в воздухе кабинетов ВТ должно соответствовать требованиям «Санитарно-гигиенических норм допустимых уровней ионизации воздуха производственных и общественных помещений» № 2/52-80, утвержденных Минздравом СССР 12.02.80 г.

Для устранения накопления зарядов статического электричества и нарушений состава воздуха после каждого занятия и в конце учебного года необходимо влажным способом очищать от пыли экраны дисплеев и другие поверхности интерьера. С этой же целью рекомендуется покрывать полы материалами, обладающими антистатическими свойствами.

Постоянный контроль за санитарным состоянием классов и кабинетов вычислительной техники в школах, режимом труда и отдыха возлагается на медицинский персонал соответствующих учреждений, периодический — на работников СЭС.

Сообщаем нашим читателям

Кроме того, хотим сообщить читателям об изменениях в выплатах доплат. Приводим выдержку из приказа Минобразования РФ № 22-шт от 03.02.92 г.

«В соответствии с п. 2 Указа Президента Российской Федерации от 15.11.91 № 211 организациям и учреждениям, находящимся на бюджетном финансировании, предоставлено право самостоятельно определять размеры надбавок, доплат, премий и других выплат стимулирующего характера.

В связи с этим размеры доплат за выполнение работы, не входящей в круг основных обязанностей, надбавок (в том числе рабочим за профессиональное мастерство, классность и т. д.), премий и других выплат стимулирующего характера определяются учреждением образования за счет и в пределах средств, направляемых на оплату труда.

Размеры доплат, установленные до 1 декабря 1991 г. постановлениями Правительства или по его поручению другими органами, являются минимальными».

Материал подготовила М. КОПЫТИНА.

ТЕХНОКОМ

МЫ УВЕРЕННО ПРЕДЛАГАЕМ ВАМ УЧЕБНЫЙ КОМПЬЮТЕР УКНЦ

И ВСЕ ЧТО ПОМОЖЕТ СДЕЛАТЬ ВАШУ РАБОТУ НА НЕМ БОЛЕЕ ТВОРЧЕСКОЙ, БОЛЕЕ ПРОДУКТИВНОЙ И БОЛЕЕ ПРОФЕССИОНАЛЬНОЙ ВО ВСЕХ ОТНОШЕНИЯХ!

ПОСТАВКА УЧЕБНЫХ КЛАССОВ - ВЫБОР КОНФИГУРАЦИИ В ВАШИХ РУКАХ !

КВАЛИФИЦИРОВАННАЯ ГРУППА НАШИХ СПЕЦИАЛИСТОВ ВСЕГДА ГОТОВА ПРОВЕСТИ НЕОБХОДИМОЕ ТЕХНИЧЕСКОЕ ОБСЛУЖИВАНИЕ И РЕМОНТ ЗАКАЗАННЫХ ВАМИ КЛАССОВ.

ВСПОМОГАТЕЛЬНЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

- СТЕНД ДИАГНОСТИЧЕСКИЙ ДЛЯ РЕМОНТА КОМПЬЮТЕРОВ ТИПА УК И БК
- ЭЛЕКТРОННЫЙ ДИСК ЕМКОСТЬЮ 1 МБАЙТ
- BASIC НА КАССЕТЕ ПЗУ
- РЕДАКТОР ТЕКСТА НА КАССЕТЕ ПЗУ С ВЫВОДОМ НА ПЕЧАТЬ

ПРОГРАММНЫЕ ПРОДУКТЫ,

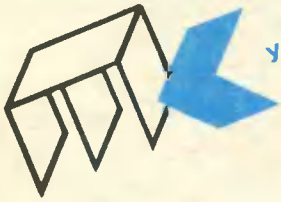
- ГРАФИЧЕСКАЯ ОБУЧАЮЩАЯ СИСТЕМА GUESS
- МАЛАЯ ИЗДАТЕЛЬСКАЯ СИСТЕМА POLYSS
- МНОГОФУНКЦИОНАЛЬНЫЙ СЕТЕВОЙ МОНИТОР NETSVD
- ФАЙЛОВЫЙ МОНИТОР SAM
- ГРАФИЧЕСКИЙ РЕДАКТОР GRUK
- УНИВЕРСАЛЬНЫЙ ДРАЙВЕР ПЕЧАТИ EP
- ПРОЦЕССОР НАСТРОЙКИ SETGEN
- ЭЛЕКТРОННАЯ ТАБЛИЦА UKCALC
- БИБЛИОТЕКА ЭКРАННЫХ ШРИФТОВ И СИМВОЛОВ MENFNT
- ИГРОВЫЕ ПРОГРАММЫ
- ИСХОДНЫЕ ТЕКСТЫ СИСТЕМЫ ПЗУ
- МНОГОКРАТНОЕ УВЕЛИЧЕНИЕ ЛОГИЧЕСКИХ И ГРАФИЧЕСКИХ ВОЗМОЖНОСТЕЙ ВАШИХ КОМПЬЮТЕРОВ!

ПРОГРАММЫ ОБУЧЕНИЯ НА КУРСАХ

- КУРСЫ ПОДГОТОВКИ ПРЕПОДАВАТЕЛЕЙ КОМПЬЮТЕРНЫХ КЛАССОВ
- КУРСЫ ПО РЕМОНТУ И ОБСЛУЖИВАНИЮ УКНЦ

БЫСТРОЕ И ЭФФЕКТИВНОЕ ОБСЛУЖИВАНИЕ ВСЕГДА БУДЕТ ПОД РУКОЙ!

У НАС ДЕЙСТВИТЕЛЬНО ЕСТЬ ВСЁ, ЧТО МЫ ПРЕДЛАГАЕМ !



ЧТОБЫ В ЭТОМ УБЕДИТЬСЯ, ОБРАТИТЕСЬ В НАШ ОФИС ПО АДРЕСУ:

103489 МОСКВА, АБОНЕНТНЫЙ ЯЩИК 51,
МНПП «ТЕХНОКОМ»,
ТЕЛ. 536-50-90, 536-50-51, 536-60-29

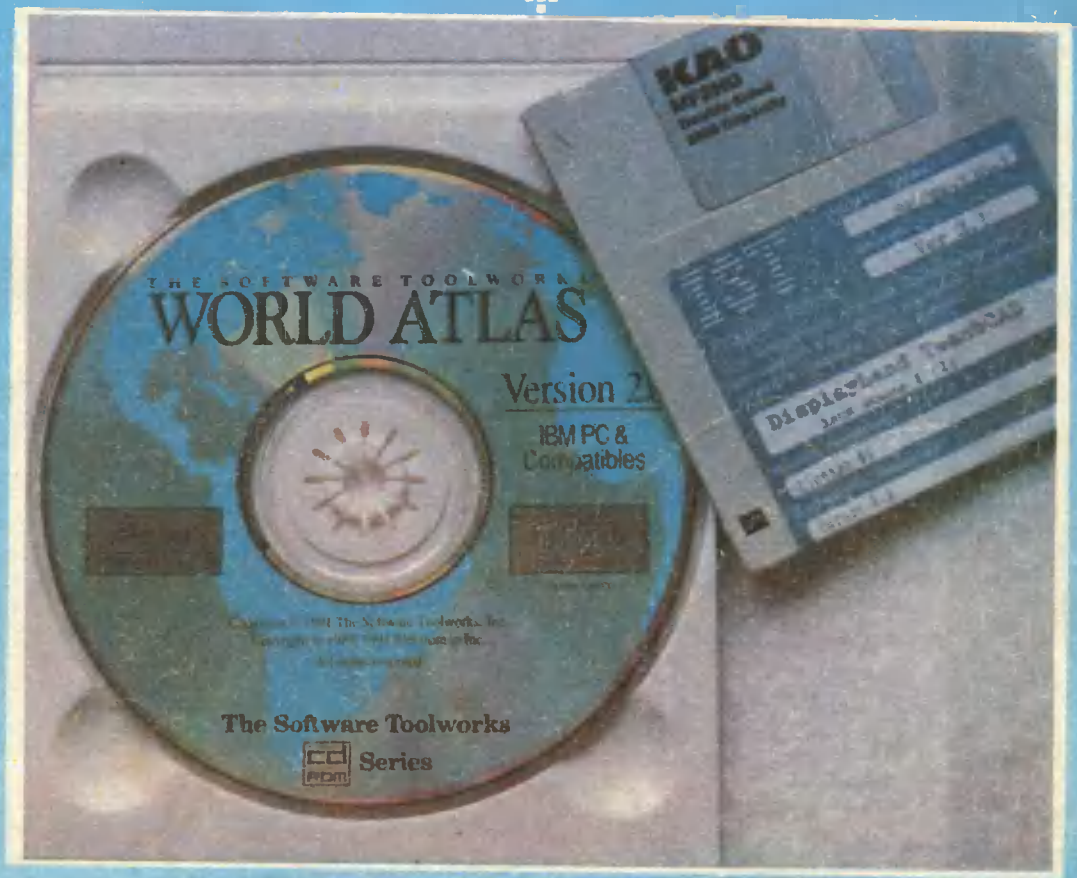
МАЛОЕ НАУЧНО-ПРОИЗВОДСТВЕННОЕ ПРЕДПРИЯТИЕ

ТЕХНОКОМ

ИНФО

3-4'92

ИНФОРМАТИКА И ОБРАЗОВАНИЕ



ISSN 0234—0453. Информатика и образование, 1992, № 3—4, 1—128.