ISSN 0234—0453

ИНФО

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

6 1991

Московский Центр Ассоциации учителей информатики и Курское производственное объединение «Счетмаш»

предлагают

для школьных учебных заведений нового типа комплекс отечественной IBM-совместимой учебной вычислительной техники «Искра-1031» от 1+2 до 1+12 персональных ЭВМ с цветными дисплеями.

Учебные комплексы поставляются с базовым пакетом учебных программ по курсу информатики и обучающими программами по логике, истории и другим школьным учебным дисциплинам.

В состав базового пакета программ входят:

- клавиатурный тренажер;
- учебный редактор текстов;
- учебный графический редактор;
- учебные электронные таблицы;
- учебная база данных;
- учебный интерпретатор Пролога;
- учебный музыкальный редактор;
- антивирусная программа.

Вместе с комплектом вычислительной техники в Центре могут быть заказаны комплекты учебников по информатике В. А. Каймина для средних школ в количестве от 20 до 100 экземпляров.

В Центре могут быть заказаны эти же пакеты программ для других школьных ЭВМ: БК-0010, УКНЦ, «Корвет» и т. п.

Телефон Центра: 197-58-61.

Освоить новую учебную вычислительную технику и пакеты учебных программ можно в учебном центре Ассоциации в Москве по адресу ул. Тухачевского, д. 20.

Слушатели учебных курсов могут получить и заказать новые учебники по информатике В. А. Каймина и пособия по методике преподавания информатики.

Иногородним во время прохождения учебных курсов предоставляются гостиница или общежитие. Оплата за проживание входит в стоимость курсов.

Заявки на прохождение учебы, приобретение литературы, пакетов программ и вычислительной техники высылайте по адресу:

123585, Москва, ул. Тухачевского, д. 20.
Центр Ассоциации учителей информатики.

Научно-методический журнал
Государственного комитета СССР
по народному образованию

Издается с августа 1986 г.
Выходит шесть раз в год
Москва «Педагогика» 1991

ИНФО ИНФОРМАТИКА И ОБРАЗОВАНИЕ

6 '91
ноябрь — декабрь

Содержание

Общие вопросы

Методика обучения

Кабинет ВТ

Клуб БК

Педагогический опыт

Обложка *С. Плющ*
Редактор отдела *А. Кравцова*
Научный редактор *Н. Копытина*
Зав. редакцией *Н. Игнатова*
Художественный редактор *Л. Коновалова*
Корректор *А. Сорнева*

Сдано в набор 23.09.91. Подписано в печать 18.10.91.
Формат 70×100 1/16. Бумага офсетная. Печать офсетная.
Усл. печ. л. 10,40. Усл. кр.-отт. 42,88. Уч.-изд. л. 12,89.
Тираж 55 885 экз. Заказ 1480. Цена 1 руб. 20 коп.

Издательство «Педагогика» Академии педагогических наук СССР и Государственного комитета СССР по печати.

Почту направлять по адресу. 119034, Москва, Смоленский б-р, д. 4. Издательство «Педагогика». Журнал «Информатика и образование».
Адрес редакции: Лефортовский пер., д. 8.
Телефон: 261-11-29

Ордена Трудового Красного Знамени Чеховский полиграфический комбинат Государственной ассоциации предприятий, объединений и организаций полиграфической промышленности «АСПОЛ»
142300, г. Чехов Московской обл.

М. ЛАПЧИК
председатель секции педагогических кадров ОНМС Гособразования СССР,
проректор по научной работе Омского педагогического института

Информатика и технология: компоненты педагогического образования

Интенсивные преобразования в производственных структурах, определяемые в том числе производством, переработкой и использованием информации, необходимой для постоянного повышения эффективности труда в различных сферах деятельности человека, неразрывно связаны с народным образованием. Именно оттуда берут начало процессы воспроизводства квалифицированных кадров для всех отраслей народного хозяйства. Информатизация сферы образования должна опережать информатизацию других направлений общественной деятельности, ибо именно здесь закладываются социальные, психологические, общекультурные, а также — что особенно важно для экономики — профессиональные предпосылки информатизации всего общества.

Государство уделяет проблеме информатизации образования постоянное внимание. Вслед за стартом общесоюзной программы компьютеризации школы в 1985 г. научное обоснование и разработка технического, учебного и программно-методического обеспечения информатизации образования неизменно включаются во многие научные программы различных уровней (упомянем Комплексную программу АПН СССР «Компьютер в школе» — 1986—1990 гг.; отраслевые научно-практические программы, формируемые в республиканских Министерствах народного образования; планы НИР в педвузовском секторе науки и т. п.). На рубеже нового пятилетия ГКНО СССР принял масштабную программу по созданию единой системы информатизации народного образования СССР — ЕСИНО. В рамках этой программы, в частности, предполагается проведение небывалых для образования по характеру, значимости и предполагаемым затратам работ по созданию и внедрению системы связи и управления информатизацией народного образования, получившей название «ИНФОСИСТ».

У истоков формирования этой программы стоял авторитетный организатор науки и руководитель народного образования Ф. И. Перегудов (1931—1990). Тесно связан с перспективами ее развития проект «Пилотные школы», осуществление которого вышло на стадию практической реализации во многих регионах страны.

Проблемы информатизации образования нашли отражение и в других государственных научно-исследовательских проектах, в которых заняты ведущие вузовские научные центры страны и подразделения АН СССР. В их числе программы, нацеленные на создание новых поколений сверхпроизводительных средств вычислительной техники на новых архитектурных и физических принципах, на разработку принципиально новых технологий, конкурентоспособных на внешнем рынке.

Заметим, что проблемы информатизации в той или иной постановке должны находить отражение и в перспективных программах «чисто педагогических» исследований, таких, как «Содержание общего среднего образования», «Сельская школа», «Творческая одаренность», «Социально-психологическая поддержка, обучение и воспитание детей с недо-

статками умственного и физического развития», «Ученая книга», а также в масштабной комплексной программе «Развитие педагогического образования на 1990—1995 годы»*.

После введения в школу предмета ОИВТ минуло шесть лет. Кто же и как участвовал в этом процессе?

«Законодатели» программы обучения информатике в школе — ГКНО (прежде Минпрос СССР) СССР и республиканские министерства народного образования. Не так давно эти законодатели «впали в замешательство» — стали поддерживать и развивать мысли о том, что если в школе нет соответствующих условий (кадры, техника и т. д.) или просто нет на то особого желания организаторов учебно-воспитательного процесса, то курс информатики в школе можно снять. Опрометчивость этого шага была очевидна с самого начала. Понятно, что даже рассуждения на эту тему вредны, а тем более рекомендации на уровне ГКНО СССР. Неужели было неясно, что директору школы, заведующему РОНО, уставшим от многочисленных проблем, привнесенных компьютеризацией, достаточно только сигнала и курс будет снят? А то, что этот шаг приведет к «откату» на рубежи, с которых придется выбираться с еще большими трудностями, об этом не подумали. Решающую роль в принятии соломонова решения сыграла позиция секции школьной информатики ОНМС информатизации образования ГКНО СССР. Правда, о содеянном каялись — в ИНФО было соответствующее разъяснение. Именно в тот неустойчивый период и был принят ряд не вполне обоснованных решений, отрицательная роль которых сказывается и сейчас: курс ОИВТ не попал в союзный компонент учебного плана, в некоторых вариантах учебных планов школ появился весьма неопределенный «тандем» под названием «математика и информатика», были задержаны тенденции «снижения» образовательного курса информатики в среднюю и начальную ступени обучения, кое-где стали назревать тенденции исключения квалификации «Информатика» из учебных планов педагогических вузов.

Представители промышленности — на них возлагалось создание учебной вычислительной техники. Сегодня мы являемся свидетелями того, что государственные программы выпуска персональных компьютеров для сферы образования сорваны — и по срокам, и по качеству. В настоящее время педагоги с нарастающим интересом следят за развертыванием некоторыми отечественными заводами выпуска IBM-совместимых персональных компьютеров; но оправдаются ли их ожидания?

Институты АПН СССР — они должны были обеспечить научно-педагогическую поддержку. Сегодня можно сказать, что, проявив завидную активность на начальной стадии обеспечения школы учебно-методическими материалами и рекомендациями, НИИ АПН СССР потеряли былую инициативу, и не только в подготовке учебников, а в общем и целом по проблеме.

Органы управления народного образования — организаторы на местах. Здесь, к сожалению, не всегда хватало компетентности. Именно поэтому в этой среде нередко возникали опрометчивые инициативы в поддержку попыток ГКНО СССР снять предмет ОИВТ, а с другой стороны, ради «галочки» ставились какие угодно, даже кооперативно-самодеятельные, КУВТ, заведомо не пригодные для школы.

Учительство — с самого начала и по сей день несло и несет на своих плечах весь груз компьютеризации. Есть учителя, которые вывели свои школы вперед, далеко за рамки, очерченные официальными учебными планами и программами. В то же время немало случаев, когда КВТ в школах по причине неподготовленности учителя стоят закрытыми или используются из рук вон плохо.

И все же, невзирая на трудности, школа в области компьютеризации претерпела за эти годы целый ряд позитивных изменений, которые хотя и не отвечают в полной мере требованиям вре-

* Перечень основных проектов опубликован в сборнике: Комплексные научные, научно-технические и образовательные программы и проекты Гособразования СССР на 1989—1994 гг. / Научно-методическое обеспечение перестройки и развития вузовского сектора науки в условиях радикальной экономической реформы. М., 1991.

мени, но говорят о том, что процесс не стоит на месте. Коротко остановимся на вопросах технического обеспечения, учебниках, программных средствах, а также общем положении школы в связи с обсуждаемой проблемой.

Техника. Анализируя оснащение школ компьютерами, можно сказать, что трудностей здесь не убавилось, хотя в «валовом» исчислении в регионах достигнуты определенные, иногда весьма внушительные, показатели. Например, в «обычной» во всех отношениях Омской области практически половина средних школ имеет КВТ. Известно, что в некоторых регионах (например, угольных — там, где предпринимались бартерные операции с зарубежными производителями ПЭВМ) плотность оснащения школ компьютерами более высока. К сожалению, в большинстве случаев техника самая разная: чаще всего — КУВТ-86, УКНЦ, «Корвет», реже — «Ямаха», другие образцы импортной техники, в том числе IBM PC-совместимая (некоторая часть школ при возможности охотно приобретает даже единичные экземпляры IBM PC AT/XT для решения задач управления: расписание, тестирование, профслужба и т. п.). В последнее время начал развиваться вариант оснащения школ классами на базе комплектов IBM PS/2.

На заре компьютеризации многим грезилась идиллическая картина оснащения школ однотипными машинами. Понятно, что это — несбыточная мечта, порождаемая исключительно только теоретическими размышлениями и ни в какой мере не отвечающая реалиям практики. Другое дело, что разноликость и разнокалиберность техники с самого начала сильно пугала, особенно учителей. Кроме чисто психологического фактора это обстоятельство сильно усложняет и делает более дорогим создание учебно-методического обеспечения, не говоря уже о резком сужении возможности для методических контактов и обмена положительным опытом. В настоящее время мы, кажется, стали постепенно привыкать к факту неизбежной разнотипности школьных КУВТ; теперь это обстоятельство стало восприниматься более спокойно — это связано, вероятно, с постепенно наращиваемой обеспеченностью школы хотя бы минимальными комплексами ППС для разных типов машин, а самое главное — с повышением общего уровня образования учителей и выпускников педвузов. При разнотипной технике со временем вычленяются инвариантные, не зависящие от типа КУВТ, методические завоевания и накопления, образующие устойчивый фонд учительского достояния: подходы к структуре и программе курса ОИВТ, к его целям, содержанию и месту в школьном учебном плане, к вопросу организации компьютерного дела в школе в целом и т. п.

Учебники. Это — трудная задача, но и здесь есть известные подвижки. При всей неповоротливости процесса подготовки и издания учебников в последнее пятилетие школа получила их в нескольких вариантах. Учебники, конечно разные, с различными достоинствами и недостатками (сравнительный и местами довольно безжалостный анализ опубликован в «ИНФО», № 1, 1991 г.). Подобные оценки учебных книг, безусловно, полезны для учителя. Правда, следовало бы понять, что надежда на появление так называемого «хорошего и стабильного» учебного пособия не менее несбыточна, чем создание однотипной компьютерной инфраструктуры школы. В современных условиях конкретная схема обучения информатике в школе зависит от столь многих факторов, что пытаться предопределить ее в каком-то конкретном варианте одного учебного пособия — дело безнадежное по определению. Творчески работающий учитель должен опираться на все имеющееся в наличии многообразие учебно-методического и программного обеспечения, а окончательное формирование содержания и методики в рамках заданного целеполагания определять сам.

Общим недостатком имеющихся учебников является то, что ни один из них не сделал принципиально нового шага по сравнению с первым вариантом, который был в предельно сжатые сроки подготовлен авторским коллективом под руководством академика А. П. Ершова. Впрочем, насколько мы продвинемся по пути формирования школьного курса истинно информатики, зависит уже не

от авторов учебников, а от общего уровня информатизации нашего общества. Школьный учебник не может бесконечно долго выдавать желаемое за действительное, в противном случае вера в реальное предназначение знаний из области информатики будет подорвана окончательно. Именно по этой причине уже первому авторскому коллективу школьного учебника приходилось писать не то, что надо бы писать на самом деле, а то, что могло быть осознано, поддержано и хоть как-то востребовано реальным окружением.

Педагогические программные средства. Понятно, что без качественного программного обеспечения, методологически и содержательно увязанного с обучением, не обойтись. И хотя к сегодняшнему дню наработано немало отечественного программного обеспечения, всех проблем это не решает.

Писать ли учителю программы самому? Совсем не обязательно. Однако, если есть стремление (а иногда необходимость), да к тому же соответствующий опыт и творческий потенциал — тогда учитель обязан создавать программно-методическое обеспечение**.

Многие программы сегодня приходится покупать. В рыночных условиях от этого, видимо, не уйти. Надо только с осторожностью относиться к нынешнему довольно пестрому рынку программ для школы — их нередко предлагают люди, далекие от сферы образования. Впрочем, важно, чтобы, приобретая программы, учитель мог (и желал того) сам выбирать, что ему нужно.

В наши дни стало возникать много различных центров, кооперативов и т. п., занимающихся поставкой программных средств и техники в сферу образования. Издаются каталоги, почта развозит по стране тонны бумажной рекламы. Хозрасчетным центрам надо делать свой бизнес, они независимы от ведомственных целей и интересов. А координационная работа, работа по обобщению полезного опыта ведется плохо. Отрасль просвещения не имеет даже сводного каталога ПС, адресованного учителю. Это, конечно, безобразие. Но с другой стороны, надо также понимать, что такого рода «ничейную» научно-практическую работу просто так делать никто не станет. И финансировать такую работу в первую очередь должны министерства народного образования.

Школа. Самое примечательное вытекает из неоднородности современной школы, которая сегодня раскрепощена в главном — она в решающей степени сама может выбирать или даже частично корректировать спускаемый «сверху» учебный план. Это обстоятельство должно было бы позволять при желании поправлять дела с компьютерными компонентами учебных планов, которые пока что очень осторожно «дозируют» информатизацию. Взять, к примеру, учебные планы российской школы (15 вариантов, рекомендованные в 1989 г., и практически без каких-либо изменений в «компьютерной» части действующие в настоящее время). Курс информатики в этих планах как самостоятельный предмет отсутствует — введена интегрированная дисциплина «Математика, информатика и ВТ». В условиях фактического отсутствия рекомендаций (что делать с этой строкой учебного плана?) школы, как правило, придерживаются прежнего положения, при котором на курс ОИВТ отводилось 102 ч распределяемых проверенным способом, восходящим к заре компьютеризации: соответственно по 1 и 2 ч в неделю в X и XI классах. В откровенной форме эти же принципы реализованы в так называемых экспериментальных учебных планах Министерства народного образования РСФСР, рекомендованных письмом от 2 апреля 1990 г. (варианты 1—6 для первой ступени — начальной школы, варианты 7—14 для второй и третьей сту-

** В 1990 г. республиканский Центр новых информационных технологий в образовании, действующий при Омском педагогическом институте, организовал и провел конкурс работ учащихся школ г. Омска и Омской области по созданию ПС. Со всей очевидностью подтвердилось: в школах имеется огромный творческий потенциал, позволяющий создавать глубокие, на профессиональном уровне ПС, в частности для нужд самой же школы. Надо, правда, иметь для этого одну «малость» — хорошего профессионала-учителя, наделенного методическим чутьем и способного вести творчество ребят к результату. По итогам конкурса целый ряд ПС, разработанных школьниками под руководством учителей, были рекомендованы в республиканский Фонд программно-методических средств.

пеней — с V по XI класс). Все структурно-содержательные «достижения» этих планов в области компьютеризации сводятся к следующему:

1. Два из шести вариантов начальной школы включают интегрированную дисциплину «Математика, конструирование и ЭВТ».

2. Курс ОИВТ отсоединен от математики, но ни в одном из 8 вариантов, рекомендованных для средней школы, не включен в союзно-республиканский компонент: как дисциплина блока «Республиканский компонент», курс ОИВТ размещается на «традиционном» месте — X и XI классы; при этом в половине случаев (для физико-математического, естественнонаучного, и трудового направлений), в X классе 1 ч в неделю, в XI — 2 ч; в других случаях (гуманитарные и эстетические направления) — по 1 ч. Как видно, прописанные в учебных планах возможности для курса ОИВТ носят явно ограниченный и консервативный характер и, по сути дела, изначально не отвечают насущным потребностям компьютеризации. Особенно это относится к положению курса ОИВТ — если в обычных, не экспериментальных учебных планах 1989 г. сохранена хотя бы теоретическая возможность «снижения» курса ОИВТ в низшие ступени (в рамках интегрированного курса «Математика, информатика и ВТ»), то его жесткое базирование в выпускном звене школы (X и XI классы) в «экспериментальных» планах противоречит самой сути идеи широкого использования знаний из области информатики при преподавании других школьных дисциплин.

Правда, в предложенных экспериментальных проектах присутствует и школьный компонент, временное пространство которого (от 2 до 6 ч в неделю в зависимости от класса) распределяет совет школы. Как сказано в объяснительной записке, это «открывает широкий простор творческой инициативе каждого учителя и всего коллектива школы». Нет слов, сказано красиво, однако когда замечаешь, что этот «широкий простор» ограничен 2—3 ч в неделю (для V—VII классов), а творческая инициатива, как это водится, связывается с разными интересами, а не только с интересами в области информатики, то становится ясно, что проявить такую инициативу в среднем звене (V—VII классы) непросто. А именно здесь рекомендовано вводить факультативные курсы, а также обобщающие межпредметные занятия. В конечном итоге, как мы видим, все будет зависеть от инициативы и настойчивости конкретного учителя информатики.

И все-таки есть обнадеживающие факторы — в современной обстановке появляются школы, которые резко уходят вперед в сфере компьютеризации. Имеются в виду не только те элитарные школы, которые волею судеб оказываются под могучим покровительством АН СССР или других влиятельных научных организаций. Речь идет о нередких случаях благодатного сочетания наличия хорошей техники (а иногда — обычной отечественной) и энтузиастов-учителей с широким размахом взглядов и творческих потенций.

В связи с этим возникает вопрос о поддержке школьных инициатив в этой сфере. К сожалению, многие школы, вступив на путь поиска, вынуждены «вариться в собственном соку», так как на сегодня ощущается острый недостаток методических рекомендаций по организации обучения информатике и в начальном, и в среднем звене школы.

Можно высказать некоторые соображения по программе непрерывного образования в области информатики в школе (I—XI классы).

Начальное звено — первая ступень (I—IV классы). Систематического курса информатики на этой ступени может и не быть, но школьники при изучении всех учебных дисциплин имеют возможность работать в компьютерном классе; в ткань школьных дисциплин вплетаются пропедевтические элементы знаний из области информатики, направленные на формирование начальных навыков общения с компьютером, а также конструирования и управления объектами с помощью ПЭВМ, чаще всего — в игровой форме; взаимодействие с ПЭВМ и ПС в связи с контролем знаний (а также обучением, тренажом и т. п.) на уроках математики, языка, естествознания, изобразительного искусства, музыки и др. Можно сказать, что это — начальная

стадия пропедевтики систематических знаний школьника из области информатики.

Неполная средняя школа — вторая ступень (V—IX классы). Здесь, на наш взгляд, целесообразно иметь систематический курс информатики, т. е. есть смысл начинать его с этого звена, по меньшей мере — с VII класса; в V—VI классах, наряду с продолжающейся практикой использования ПЭВМ как учебного средства, возможно развитие на доступном материале представлений об информации, информатике, автоматизации в форме кружковой работы, в том числе с использованием программно реализуемых учебных исполнителей — роботов. Первый концентр образовательного курса информатики включает также начальное формирование знаний по использованию инструментальных средств. Наряду с использованием редакторов появляются работы с привлечением учебных баз данных, электронных таблиц, что является предпосылкой к использованию компьютера в старших классах. Элементы программирования — в старших группах ступени (VIII—IX классы). Здесь же — социальные аспекты информатики в обществе: знакомство с государственной службой информации, компьютерные правонарушения и т. п. Эти аспекты информатики получат развитие и в старшем звене.

Параллельно — использование КВТ при изучении школьных дисциплин. При этом надо иметь в виду, что на уроки по предметам не возлагается задач обучения системе знаний из информатики — это задача курса информатики. Уроки же по другим предметам развивают предметные применения системы знаний, получаемых на уроках информатики, а также служат полигоном применения ПЭВМ как средства обучения.

Высшая средняя школа (IX—XI классы). Второй концентр курса информатики (практически — допрофессиональная ориентация — это приложения информатики, формируемые из практических потребностей общества и поддерживаемые мощными программными средствами: применение ЭВМ в экономике, бизнесе и коммерции, в системе государственного информационного обслуживания, в социальной сфере и других областях. Форма: наряду с базовым курсом информатики, который изучается всеми учащимися, составление профильных групп для углубленного изучения дисциплины. Здесь — и углубленное программирование (для тех, кто выбирает соответствующий профиль).

Такого рода проекты и соответствующие программы находятся сегодня в состоянии проработки и поиска и не поддержаны пока еще хоть сколько-нибудь широко известным опытом. Варианты решений могут быть разными — их и надо иметь несколько разных, чтобы любая школа могла выбрать подходящий в соответствии со своими возможностями и интересами. Особенность положения, однако, в том, что конкретная реализация подобных программ зависит не столько от того, насколько замечательна та или иная программа, а более всего от наличия и качества программно-методической поддержки и технического обеспечения.

Все сказанное выше сильно нагружает объем обязательных профессиональных знаний учителя, который в современных условиях, с одной стороны, должен быть грамотным в области информатики, а с другой — в области новых педагогических технологий. Время резко подняло планку его обязательных знаний. Только фундаментальные знания позволят учителю обрести те степени свободы, которые необходимы и для самостоятельно выбора, корректировки учебного плана или программы, а также для адекватного выбора нужной методики и технологии.

Продолжение следует.

А. ГЕЙН, А. СЕНОКОСОВ

Программно-методический комплекс для классов с углубленным изучением ОИВТ

В статье рассматриваются вопросы, связанные с формированием классов с углубленным изучением информатики. Актуальность таких классов очевидна. Во-первых, в современном мире спрос на специалистов по обработке информации существенно возрос. Во-вторых, способности к информатике и программированию проявляются достаточно рано. Поэтому в Свердловске одновременно с разработкой общеобразовательного курса информатики была начата работа по проектированию преподавания углубленного курса.

Один из классов с углубленным изучением информатики был создан в СШ № 35 Свердловска в 1987/88 уч. г. Ежегодно на конкурсной основе в школе формируется специализированный класс. Для преподавания курса информатики была разработана программа, неоднократно уточнявшаяся в результате апробации. Затем постепенно начал создаваться программно-методический комплекс (ПМК) по курсу информатики и основам вычислительной техники для специализированных классов, состоящий из двух компонентов:

1. Методический компонент: программа курса, учебник, книга для учителя.

2. Программный компонент: система разработки программного обеспечения «ПИ», учебный исполнитель алгоритмов «Паркетчик», учебный исполнитель алгоритмов «Вычислитель», учебная виртуальная ЭВМ «Малютка», учебная экспертная система, электронная таблица, учебный пакет прикладных программ, текстовый редактор, графический редактор, информационно-поисковая система, электронная таблица, системы программирования на языках различных типов.

Основной задачей курса является знакомство учащихся с отраслью общественного производства, обеспечивающей разработку, производство и обслуживание средств промышленной эксплуатации информационных ресурсов, т. е. с «промышленностью обработки данных» (ПОД), а также приобретение необходимых трудовых навыков в создании одного из средств производства ПОД — программного обеспечения (ПО).

Курс рассчитан на изучение в VIII—XI классах за счет часов, отводимых на трудовое обучение согласно типовому учебному плану для школ (классов) указанного типа, некоторой части часов ОППТ, а также на уроках информатики в X и XI классах. Следовательно, в VIII и IX классах на этот курс отводится по 136 ч (4 ч в неделю), в X классе — 204 ч (6 ч в неделю), в XI классе — 238 ч (7 ч в неделю).

Включение часов ОППТ в курс оправдано созданием и эксплуатацией учащимися обучающих и контролирующих программ по предметам как естественнонаучного, так и гуманитарного цикла (педагогических программных средств — ППС).

Курс состоит из пяти основных разделов:

1. Информация как промышленный ресурс.

2. Этапы решения задачи с помощью ЭВМ.

3. Программное обеспечение ЭВМ и

методы его разработки.

4. Основные виды алгоритмов и вычислительные аспекты применения ЭВМ.

5. Основные принципы устройства ЭВМ, устройства сбора, накопления и передачи данных.

Программой предусматривается параллельное и концентрическое изучение этих разделов. Это позволяет по мере изучения курса давать все более глубокие знания по каждому из разделов, не теряя при этом целостности изложения всего материала.

Курс рассчитан на широкое применение ЭВМ и предусматривает выделение половины всего времени на практическую работу на ЭВМ. Более того, успешная работа по темам курса возможна лишь при создании в школе Центра программирования, решающего кроме учебных еще и промышленные задачи, в первую очередь задачи, связанные с обеспечением компьютеризации учебного процесса в рамках школы (сопровождение внедренных ППС и создание новых, поддержка АСУ «Школа и т. д.).

Необходимо подчеркнуть, что Центр программирования не имеет ничего общего с кружком или факультативом. Это, скорее, учебно-промышленный производственный участок. Все необходимые теоретические знания для работы в Центре учащиеся получают в основное учебное время.

Программа курса предусматривает наличие в школе средств вычислительной техники (СВТ) — не менее 12 рабочих мест, учебного робота-манипулятора и средств промышленной графики: графопостроителя или цветного графического дисплея с высокой степенью разрешения. И если наличие СВТ является условием необходимым, то в остальном программа, на наш взгляд, легко адаптируется к условиям конкретной школы.

Коротко раскроем содержание вышеупомянутых разделов, а также опишем основные методические особенности учебника, поддерживающего этот курс.

1. Информация как промышленный ресурс

1. 1. Понятие об информации, ее виды и свойства. Историческая необходимость возникновения промышленной обработки данных (ПОД)

Исторические аспекты хранения, преобразования и передачи текстовой информации. Краткое знакомство с другими видами информации, способами их сбора, передачи, хранения и преобразования. Человек как информационный процессор. Особенности восприятия информации человеком. Понятие об информационном потоке, информации как объекте труда (ручного или автоматизированного). Возрастание роли информации в научно-техническом прогрессе.

Информация как неотъемлемый компонент развития цивилизации. Изобретение письменности — первая информационная революция, книгопечатания — вторая. Цикл: знания — наука — общественное производство — знание. Создание предпосылок ускоренного развития технологической цивилизации. Информационный взрыв в XX в.— предпосылка создания ПОД. Появление ПОД — третья информационная революция. Феномен персонального компьютера. Понятие об автоформализации знаний. Особенности четвертой информационной революции.

Текстовый редактор и работа с ним. Диалоговые системы. Работа с диалоговой системой.

Учащиеся должны знать: определение предмета информатики; содержание понятий «информация» и «количество информации»; роль информации в научно-техническом прогрессе и развитии; возможности конкретного текстового редактора и диалоговой системы.

Учащиеся должны уметь: рассчитывать объем текстовой информации; работать с конкретным текстовым редактором и диалоговой системой.

1.2. Системы хранения и поиска данных

Методы хранения и поиска информации. Элементы алгебры логики. Информационно-поисковые системы. Базы данных, типы баз данных, практическая работа с учебной базой данных,

изучение структуры и возможностей реляционной базы данных, понятие о СУБД, языках запросов. Понятие об экспертных системах, банках знаний, искусственном интеллекте. Электронные таблицы как сращивание информационно-поисковых и вычислительных систем. Основные функции электронной таблицы типа SUPERCALK. Ее использование для простых расчетов. Автоматизированные рабочие места.

Учащиеся должны знать: основные принципы построения конкретных файловых систем, принципы работы с ИПС, банками данных, электронными таблицами; типы баз данных; принципы организации диалога в экспертных системах; виды АРМ.

Учащиеся должны уметь: составлять программу поиска, включения, исключения, замены элементов в учебных базах данных или базах данных типа dBASE; пользоваться электронными таблицами для решения задач прогнозирования и планирования; вести диалог с учебной экспертной системой; пользоваться конкретным АРМ.

1.3 Системы автоматизации проектирования как подотрасль ПОД

Понятие о системах автоматизации проектирования, их место в ПОД, роль и значение в народном хозяйстве. Знакомство с графопостроителем или графическими возможностями ПК, пакетом графических программ. Изучение учебной системы машинной графики с элементами расчета, работа с ней, модификация и улучшение, работа на графопостроителе или ПК с графическими возможностями. Знакомство с промышленной системой автоматизированного проектирования.

Учащиеся должны знать: принципы работы графопостроителя или графического ОЗУ ПК, назначение систем автоматизации проектирования, правила техники безопасности при работе на графопостроителе.

Учащиеся должны уметь: пользоваться пакетом графических программ; непосредственно работать на графопостроителе или ПК с графическими возможностями; пользоваться учебной системой машинной графики с элементами расчета.

1.4. Робототехника как подотрасль ПОД

Понятие о робототехнических системах, их специфике и месте в ПОД, роль и значение в промышленном производстве. Знакомство с промышленным вариантом робототехнической системы. Программирование учебного робота-манипулятора.

Учащиеся должны знать: принципы работы промышленных роботов; их типы; системы команд учебного робота-манипулятора.

Учащиеся должны уметь: программировать учебного робота-манипулятора.

1.5. Применение ПОД для управления экономикой

Понятие об управленческом решении. Важность полной и достоверной информации для принятия правильного решения. Структура АСУ «Школа». Работа с ней, улучшение и модернизация. Работа над промышленной задачей экономико-управленческого характера.

Учащиеся должны знать: назначение и принципы построения АСУ; структуру АСУ «Школа».

Учащиеся должны уметь: работать с АСУ «Школа», менять ее структуру, расширять функции.

1.6. Применение СВТ в учебном процессе

Понятие об обучении как одной из отраслей информатики. Основные этапы обучения с точки зрения информатики. Принципиальная возможность применения методов ПОД в обучении. Понятие о педагогическом программном средстве (ППС). Основные требования к ППС, особенности его создания и сопровождения. Разработка собственного учебного ППС с дальнейшей доработкой его до промышленного изделия.

Учащиеся должны знать: требования к ППС, основные этапы его создания и сопровождения.

Учащиеся должны уметь: сопровождать используемые в учебном процессе ППС; разрабатывать простейшие ППС.

2. Этапы решения задачи с помощью ЭВМ

2.1. Компьютерное моделирование

Понятие «плохо» и «хорошо» поставленной задачи. Модель задачи. Понятия компьютерной и математической моделей задачи. Построение модели: выделение предположений, на которых будет основана модель (постановка задачи), определение исходных данных в задаче и результатов, установление соотношений, связывающих исходные данные и результаты.

Проверка адекватности построенной модели. Неоднозначность выбора модели «плохо» поставленной задачи. Понятие о компьютерном эксперименте.

Цикл построения компьютерной модели.

Учащиеся должны знать: этапы решения задачи на ЭВМ; принципы построения модели задачи; цели проведения компьютерного эксперимента.

Учащиеся должны уметь: распознавать «плохо» и «хорошо» поставленные задачи; строить простые компьютерные и математические модели; анализировать соответствие модели и исходной задачи; проводить компьютерный эксперимент для построенных моделей.

2.2. Архитектура промышленного изделия

Понятие об архитектуре промышленного изделия, дизайне. Принципы создания архитектурных проектов различных изделий, специфика архитектуры (спецификация пользователя или внешние спецификации) в программном обеспечении ЭВМ.

Понятие интерфейса «человек — машина». Типы интерфейса. Принципы построения сценариев программного обеспечения.

Учащиеся должны знать: принципы создания архитектурных проектов ПО; особенности организации интерфейса «человек—машина».

Учащиеся должны уметь: разрабатывать архитектуру простейших программных проектов; создавать сценарии работы с несложными программными продуктами.

3. Программное обеспечение ЭВМ и методы его разработки

3. 1. Основы алгоритмизации

Понятия алгоритма и исполнителя алгоритмов. Допустимые действия исполнителя. Понятие о достижимых целях исполнителя. Исполнитель алгоритмов Паркетчик.

Основные алгоритмические конструкции: ветвления, циклы, вспомогательные алгоритмы, определяемые допустимые действия. Ветвления в полной и неполной формах. Ветвление в форме «выбор». Циклы в формах «пока», «пока не» и «для каждого». Требования к оформлению вспомогательных алгоритмов. Формальные и фактические параметры во вспомогательных алгоритмах.

Переменные в алгоритмах. Особенности работы с переменными. Исполнитель алгоритмов Вычислитель. Числовые, логические и символьные переменные.

Проверка алгоритмов. Некоторые приемы доказательства правильности работы алгоритма и его конечности (понятия инварианта цикла, лимитирующей функции и т. п.).

ЭВМ как универсальный исполнитель. Имитация других исполнителей на ЭВМ.

Учащиеся должны знать: способы представления алгоритмов; основные алгоритмические конструкции (ветвления, циклы и т. д.), правила их записи и особенности исполнения; системы допустимых действий учебных исполнителей алгоритмов.

Учащиеся должны уметь: составлять и записывать алгоритмы для учебных исполнителей с использованием соответствующих алгоритмических конструкций; составлять протоколы исполнения алгоритмов; распознавать необходимость применения той или иной алгоритмической конструкции при решении задачи; использовать готовые вспомогательные алгоритмы при создании нового алгоритма; организовывать данные для эффективной алгоритмической обработки.

3.2. Понятие о программном обеспечении как средстве

производства ПОД. Этапы создания ПОД

Соотношение программного обеспечения и средства вычислительной техники в ПОД. Понятие о software и hardware. Виды ПО и их краткая характеристика. Понятие о проектировании и алгоритмизации.

Понятие о машинном языке, трансляторах, кодировке программы, отладке, контрольном варианте. Простейшие приемы отладки.

Учащиеся должны знать: роль программного обеспечения в ПОД, его виды; принципы и методы отладки программ.

Учащиеся должны уметь: осуществлять перевод алгоритма в программу для конкретного исполнителя; осуществлять отладку программы.

3.3. Технология разработки ПО

Метод пошаговой детализации, стандартные программы и функции; функции, определяемые пользователем; способы обращения к подпрограммам. Основные типы и способы организации данных. Понятие о нисходящем проектировании, язык PDL (program desing language) нисходящее PDL-проектирование, структурное программирование. Основные проблемы больших программных проектов: понятие о комплексном программном продукте, проблема межличностных связей исполнителей больших программных проектов. Способы решения проблем: создание бригады Главного программиста, организация системы документирования, система нисходящего комплексного тестирования. Понятие об инструментарии программиста, методы его создания и ведения. Инструментарий при создании больших программных проектов. Понятие о концептуальном единстве системы. Понятие программного интерфейса.

Учащиеся должны знать: основные типы данных и способы их организации; метод пошаговой детализации; основные принципы структурного программирования; основные проблемы создания больших программных проектов и способы их решения; требования к стандартным программам.

Учащиеся должны уметь: организовать данные в соответствии с требованиями задачи; пользоваться методом пошаговой детализации для решения задач; проводить нисходящее PDL-проектирование, структурировать программы; осуществлять разделение труда и вести коллективную работу при реализации простейших программных проектов; документировать свои программы; пользоваться стандартными программами и создавать простые стандартные программы.

3.4. Языки программирования

Язык программирования как одно из средств «общения» с ЭВМ. Соотношение между понятиями «алгоритм» и «программа». Переменные в языках программирования; типы переменных.

Понятие о процедурных языках. Изучение одного из процедурных языков (Си, Паскаль, ПЛ/1 и т. п.).

Понятие о логическом программировании. Изучение языка Пролог.

Понятие об объектно-ориентированных языках. Изучение языка Форт (или объектно-ориентированных диалектов языков Си или Паскаль).

Учащиеся должны знать: основные операторы и запись основных конструкций на изучаемом языке программирования; типы переменных, используемых в данном языке.

Учащиеся должны уметь: программировать на изучаемом языке и вести отладку программ.

3.5. Прикладное и системное программное обеспечение

Типы пакетов прикладных программ: библиотеки программ, проблемно-ориентированные системы. Иерархия языков. Трансляторы: компиляторы и интерпретаторы, принципы построения системы FORTH. Понятие об ОС, БПО, ЯУЗ, организация наборов данных. Простейшие системные работы в конкретной ОС. Системные стандартные программы. Карта памяти ОС.

Учащиеся должны знать: типы пакетов прикладных программ; области применения наиболее употребительных языков программирования; этапы работы трансляторов; функции ОС, взаимодействие ОС и программы пользова-

теля; элементы языка управления заданиями; назначение и состав базового программного обеспечения.

Учащиеся должны уметь: использовать диагностику транслятора для отладки программ; пользоваться конкретным БПО; проводить простейшие системные работы в конкретной ОС (создание, удаление, переименование, копирование наборов данных и т. п.); уметь использовать карту памяти конкретной ОС.

3.6. Цикл «жизни» ПО

Понятие о сопровождении ПО, стоимость различных этапов разработки и внедрения ПО, их специфика. Методы снижения расходов. Система организации сопровождения ПО. Принципы работы с конечным пользователем. Основные части технического задания. Особенности договоров на разработку и сопровождение ПО. Система документации ПО.

Учащиеся должны знать: назначение и принципы сопровождения ПО, этапы его разработки, внедрения и сопровождения; принципы совместной работы с конечным пользователем.

Учащиеся должны уметь: сопровождать программы АСУ «Школа» и ППС.

4. Основные виды алгоритмов и вычислительные аспекты применения ЭВМ

4.1. Основные виды алгоритмов

Понятие об эффективности алгоритмов.

Алгоритмы на линейно упорядоченных множествах: поиск наибольшего элемента; сортировка; двоичный поиск.

Алгоритмы на графах: нахождение связной компоненты; обход графа; поиск кратчайшего пути на графе.

Комбинаторные алгоритмы: генерация перестановок.

Рекурсивные алгоритмы: особенности применения и машинной реализации.

Учащиеся должны знать: указанные алгоритмы и оценки их эффективности.

Учащиеся должны уметь: использовать указанные алгоритмы для решения задач.

4.2. Приближенные вычисления

Особенности приближенных вычислений. Понятие о плохой и хорошей обусловленности задач. Понятие об устойчивости метода к ошибкам округления. Интерполяция.

Метод наименьших квадратов.

Учащиеся должны знать: особенности приближенных вычислений; метод линейной интерполяции; метод наименьших квадратов.

Учащиеся должны уметь: оценивать ошибки вычислений на ЭВМ; пользоваться методами линейной интерполяции и наименьших квадратов.

4.3. Обзор вычислительных методов решения задач

Метод деления пополам. Метод хорд.

Метод прямоугольников для вычисления площадей.

Методы поиска экстремумов (покоординатный спуск, градиентные методы).

Методы линейного программирования.

Метод Монте-Карло. Датчики случайных чисел (ДСЧ); основные требования к ДСЧ.

Статистические методы оценки правдоподобия (критерии Фишера и Стьюдента).

Учащиеся должны знать: указанные методы.

Учащиеся должны уметь: использовать изученные методы для решения задач.

5. Основные принципы устройства ЭВМ, устройств сбора и передачи данных

5.1. Представление информации в ЭВМ

Представление информации в ЭВМ. Понятие о цифровой информации. Обогащение информации. Понятие о центральном процессоре как основном устройстве обогащения информации. Понятие об оперативной памяти, внешних накопителях, устройствах сбора, передачи цифровой информации как о средствах производства ПОД.

Системы счисления. Двоичная и шестнадцатиричная системы. Действия с числами в двоичной системе. Перевод чисел из одной системы счисления в другую.

Учащиеся должны знать: способы представления информации в ЭВМ, принципы записи чисел в позиционной системе счисления, в частности двоичной и шестнадцатиричной; причины использования двоичной системы при работе с ЭВМ; назначение центрального процессора, оперативной памяти, внешних устройств как средств производства ПОД.

Учащиеся должны уметь: переводить числа из двоичной системы в шестнадцатиричную и обратно, из двоичной и шестнадцатиричной в десятичную; подсчитывать объем памяти, необходимой для хранения заданного набора символов.

5.2. Исторический обзор и перспективы развития СВТ. Основы микроэлектронной и микропроцессорной техники

Поколения ЭВМ; принципы фон Неймана; концепция ЭВМ 5-го и 6-го поколений; оптоэлектроника, нейристоры. Понятие об аппаратурном интерфейсе. Контроллер. Принципы работы процессора. Понятие о системе команд процессора, языке ассемблер. Учебная виртуальная ЭВМ «Малютка»: ее устройство и система команд. Ассемблер и Дизассемблер для «Малютки».

Обзор микропроцессорных средств и систем.

Учащиеся должны знать: принципы фон Неймана; разделение информации, хранимой в памяти ЭВМ, на числа и команды, виды представления данных; основные принципы работы процессора и оперативной памяти; организацию памяти и систему команд учебной виртуальной ЭВМ, ее ассемблер и дизассемблер; основные принципы создания и применения микропроцессорной техники.

Учащиеся должны уметь: программировать учебную ЭВМ; писать программы на ее ассемблере; пользоваться дизассемблером.

5.3. Устройства сбора, накопления и передачи информации

Устройства сбора данных. Устройства хранения информации. Устройства передачи информации. Устройства вывода информации.

Учащиеся должны знать: виды и типы устройств сбора, накопления и передачи информации; правила техники безопасности при работе с накопителями на магнитных дисках, лентах, АЦПУ, АЦД, принтером.

Учащиеся должны уметь: обращаться с НМД, АЦПУ, принтером; пользоваться АЦД для ввода информации в ЭВМ; осуществлять обмен информацией между АЦД и НМД.

5.4. Персональные компьютеры

Отличительные особенности персональных компьютеров (ПК). Технический аспект феномена ПК. Специфика эксплуатации ПК.

Учащиеся должны знать: отличительные особенности ПК; правила эксплуатации конкретного ПК и техники безопасности при работе на нем.

Учащиеся должны уметь: работать на ПК конкретного типа.

Заметим, что приведенное конспективное изложение содержания курса не отражает хронологического порядка изложения тем. Тематическое планирование материала (с указанием часов и порядка изучения вопросов) приведено в [7].

Общие методические установки близки к тем, которые изложены в [8]; мы не будем их повторять. Поэтому и методико-дидактическая структура создаваемого нами учебника в своей основе такая же, как и у учебника [3]. Но, разумеется, имеются значительные отличия в отборе материала, в проведении тех или иных «сквозных линий».

Прежде всего отметим, что мы избрали «модульный» принцип построения учебника. Он состоит в том, что для каждого достаточно замкнутого раздела курса фактически написан свой «подучебник». Всего подучебников 7. Это предоставляет пользователю (как учащемуся, так и учителю) целый ряд возможностей.

1. Подучебники (как и модули программного обеспечения ЭВМ) достаточно независимы друг от друга. Если пользователю хочется изучать данную

тему по другой книге, он всегда может это сделать.

2. Все подучебники связаны в единое целое стержневым учебником, в котором каждая тема излагается конспективно. При этом каждому подучебнику соответствует глава в стержневой книге. Стержневой учебник позволяет при необходимости быстро пройти нужную тему, чтобы, скажем, иметь возможность читать какой-то подучебник, не изучая другие подучебники. Вторая цель, которую позволяет достичь эта книга,— организация повторения пройденного.

3. Модульное строение позволяет менять порядок изучения тем в соответствии со вкусами и потребностями пользователя. Мы выстроили курс, согласующийся с нашими собственными представлениями, но отнюдь не навязываем их всем, кто захочет воспользоваться нашим учебником.

Продолжение следует.

М. ГОЛЬЦМАН, Ю. ПЕРВИН, Н. ПЕРВИНА

Элементы музыкальной грамоты в курсе раннего обучения информатике

Урок 4. **Тема урока: Копирование, обмен и сохранение музыкальной информации.**
Цель урока: **Закрепить навыки копирования информации в редакторе и работы в локальной сети.**
План урока: **Обсуждение домашних заданий. Справочник Шарманщика. Копирование информации в Шарманщике.**
Лабораторная работа: **Копирование. Режимы приема и передачи информации. Упражнения в приеме и передаче информации. Сохранение информации, каталог (демонстрация). Вывод музыкальной информации на принтер.**
Домашнее задание: **песня с повторяющимися фрагментами.**

§ 8.12. Копирование в Шарманщике

Композиторы часто используют прием повторения: одинаковая мелодия звучит в нескольких местах произведения. Такой прием подчас делает песню красивой и выразительной. Однако при записи нот такого произведения приходится понапрасну тратить много времени: один раз написанный отрывок необходимо переписывать многократно.

В Шарманщике (как в Микроне и Раскрашке) есть возможность **копировать** информацию. Надо установить курсор (Попугая) над копируемой нотой и нажать клавишу *F*1. Так запоминается нота. Теперь можно отвести Попугая в любое место нотного стана (в частности, за пределы окна) и нажать *SHIFT*+*F*1. Таким образом ноту «вспоминают». Если Шарманщик «вспоминает» скопированную ноту в середине мелодии, то происходит вставка: ноты справа от Попугая отодвигаются, уступая место вспоминаемой ноте.

Каждое нажатие *F*1 перемещает курсор—Попугая на одну позицию вправо, словно правая стрелка. Поэтому, если надо скопировать несколько нот подряд, достаточно столько же раз нажать *F*1. Если теперь стрелками поставить Попугая в то место, где надо повторить запомненную часть мелодии, то выполняется вставка сразу всей копируемой группы нот.

Запомненную информацию можно «вспомнить» несколько раз в разных местах нотного стана.

Вопросы и упражнения:

1. Сравните, как выполняется копирование в трех разных редакторах — Микроне, Раскрашке и Шарманщике.

2. На нотном стане компьютерного экрана написана песня. Как с помощью копирования заставить компьютер сыграть три куплета этой песни?

Лабораторная работа № 5: **Копирование**

1. Посмотрите внимательно ноты песни «Маленькой елочке холодно зимой». Найдите в ней повторяющиеся отрывки.

2. Запишите ноты песни на экране, копируя ранее написанные отрывки.

3. Последние две строки куплета повторяются. Скопируйте этот отрывок и допишите его в конец песни.

Окончание. Начало см.: ИНФО. 1991. № 4, 5

4. С помощью приема копирования напишите ноты для исполнения трех куплетов песни.

§ 8.13. Дискотека Шарманщика

Информацию, хранимую в памяти ЭВМ, можно переписать во внешнее запоминающее устройство — магнитный диск. В Микроне диск был библиотекой текстов. Для Раскрашки диск — архив рисунков. Шарманщик использует магнитный диск как дискотеку:

любую песенку можно записать на диск, для этого достаточно задать ее имя;

любую песенку можно прочитать с диска сколько угодно раз так, что она останется неизменной на том же месте.

Дискотека Шарманщика — это хранилище информации, отличающееся тем, что полученную информацию можно не только увидеть, но и услышать.

Правила пользования дискотекой Шарманщика не новы для того, кто имел дело с редакторами Микрон и Раскрашка: чтобы записать музыкальную информацию на диск, надо сначала передать ее с ученической машины на учительскую (режим «Передача»); для получения мелодии с диска ее должна сначала прочитать учительская машина, а уж затем ученическая машина принимает эту информацию от центральной (режим «Прием»). Шарманщик может пользоваться принтером (режим «Вывод на принтер»), т. е. печатать ноты на листах бумаги. В любой момент можно обратиться к каталогу (режим «Каталог»), чтобы посмотреть имена всех тех мелодий, которые хранит Шарманщик.

О всех этих и других возможностях Шарманщика можно прочитать в его справочнике. Войти в справочник можно, как всегда, нажав клавиши *CTRL*+«?».

Методический комментарий

Этот урок насыщен новыми операциями обработки музыкальной информации. Однако практически все они повторяют общие приемы редактирования, используемые в известных детям программных редакторах. Поэтому понятия копирования информации, режимов общения в сети, именования информации, каталога, по существу, не являются новыми — точно те же понятия использовались ранее в Раскрашке и Микроне [2].

После серии предварительных простых упражнений начинается содержательная работа (использующая копирование) с музыкальным материалом. Она предваряется беседой о распространенности и значимости приема копирования в музыке. Такой прием используется как композиторами, так и (очень часто) в народном творчестве. Типичный пример — белорусская народная песня «Савка и Гришка сделали дуду». На инструменте или голосом (обе эти возможности могут быть совмещены) учитель исполняет песню. После исполнения первой половины песни учитель останавливается, говорит: «Вы услышали только половину этой песни, но даже в прозвучавшей половине отчетливо заметны два повторяющихся отрывка. Из двух повторяющихся частей состоит и вторая половина песни» — и после этого завершает исполнение.

На доске написаны слова песни про Савку и Гришку:

Савка и Гришка сделали дуду,
Савка и Гришка сделали дуду,
Ай-дуду, ай-дуду, сделали дуду,
Ай-дуду, ай-дуду, сделали дуду.

Повторяемость слов подчеркивает повторяемость мелодии. Глядя на слова песни, которые, возможно, детям ранее не были знакомы, класс хором исполняет песню, учитель подпевает им, аккампонируя на музыкальном инструменте или на компьютере.

Далее учитель показывает классу (на экране, на плакате) первую фразу мелодии — первую часть первой половины. Ученики воссоздают ее на своих экранах, прослушивают в наушниках, потом копируют, строя, таким образом, вторую фразу, и снова прослушивают. Затем так же учитель показывает третью фразу, а ученики по этой записи — всю вторую половину песни: третью и четвертую фразы.

После этого предстоит достаточно большая по времени лабораторная работа.

Содержание завершающего этапа урока — правила работы с сетью в Шарманщике. Это скорее напоминание знакомых понятий, чем изложение нового материала: технология обмена та же, что и в Микроне.

В конце урока учитель объявляет, что следующее занятие будет посвящено подготовке к концерту, который будет подготовлен ребятами на компьютерах. Поскольку концертные номера будут

предложены ученикам на следующем уроке, это объявление имеет смысл главным образом в связи с тем, что учащиеся музыкальных школ должны будут исполнить на концерте произведения по собственному выбору. Значит, на следующий урок им предстоит принести ноты выбранных ими песен (песни предпочтительнее других жанров, так как в этом жанре ученик не только заставляет компьютер исполнять написанное произведение, но и сам принимает участие в исполнении).

Надо напомнить учащимся музыкальных школ информацию, резервированную в учебнике для них (§ 8.8—8.10), и бегло показать, как можно набрать бемоли, диезы, точки и паузы. Это необходимо, чтобы, готовясь к концерту, дети не были смущены «сложностью» выбранных ими мелодий.

Домашнее задание: записать в тетради слова песни, в которой предусматривается повторение мелодии. Пример такого повтора — две строки каждого куплета в песне про маленькую елочку. Другой пример (тривиальный) — куплетная структура песни. Задание по учебнику — § 8.12 и 8.13.

Урок 5. **Тема урока: Подготовка к концерту.**
Цель урока: **Закрепление навыков работы с редактором, первое знакомство с правилами участия в концерте.**
План урока: **Обсуждение программы концерта и плана его подготовки. Знакомство с концертными номерами, их распределение. Набор музыкальных произведений на компьютере. Авторские прослушивания и корректировка ошибок. Передача записанных мелодий и сохранение их на диске.**
Домашнее задание: **выучить слова концертных песен.**
Методический комментарий.

Проверка домашнего задания не займет время урока: учитель собирает тетради для проверки.

Все нынешнее занятие посвящено подготовке к предстоящему концерту, который состоится на следующем уроке. Учитель подробно рассказывает, как будет организован этот концерт. Ученикам предстоит по очереди исполнить музыкальные произведения, предварительно записанные ими на компьютерах. Большинство произведений, предлагаемых для концерта,— это песни. Поэтому ученики-исполнители должны будут спеть песни под аккомпанемент компьютера. Для этого на сегодняшем уроке мелодии надо будет записать нотами с помощью Шарманщика и занести на диск учительской машины.

Концерт составлен в основном из народных песен. Учителю надо представить каждый концертный номер, чтобы дети могли выбрать понравившуюся им мелодию. Могут быть предложены несколько способов представления мелодий. По-видимому, наилучший из них — вокальное учительское исполнение. Другой способ — исполнение мелодии учителем на музыкальном инструменте (например, на том же «Соловушке»). Наконец, если педагог смущен собственными музыкальными способностями, он может последовательно воспроизводить с помощью компьютера записанные на диске концертные мелодии. Каждое произведение учитель должен прокомментировать, рассказав либо о его авторе, либо об особенностях исполнения, либо о содержании слов песни.

Итак, в классе звучит очередная мелодия, определяется ее будущий исполнитель, которому учитель выдает листок с отпечатанными на нем нотами и словами песни. Одновременно учитель фиксирует исполнителя в списке, по которому будет формироваться программа концерта. Материал для концерта (ноты и слова песни) приведен в конце комментария к текущему уроку. Можно допустить, чтобы одна мелодия исполнялась несколькими учениками. Так в программе концерта появляется простейший ансамблевый номер — унисон.

Другая коллективная форма работы — песня с припевом. Один ученик готовит мелодию куплета, другой — припева.

Более интересны ансамблевые номера, в которых звучит одновременно несколько разных партий. Правда, такой номер можно поручать только учащимся музыкальной школы. Еще более привлекательный концертный номер для детей, которые учатся в музыкальной школе,— канон. Его можно предложить юным музыкантам, которые уже встречались с такой формой исполнения.

В программу концерта рекомендуется

включить 1—2 танцевальные или плясовые мелодии. Их исполнение можно предложить детям, которым по каким-либо причинам трудно петь.

Учащимся музыкальных школ, не участвующим в ансамблевых номерах, предлагается самим набрать на компьютере их любимую мелодию (выбор мелодии был для них домашним заданием). Они выступят на концерте с произведениями, которые, по-видимому, не звучали в начале урока в учительском исполнении.

Особо надо поощрить тех, кто захочет подготовить к концерту мелодию собственного сочинения.

Одному из учеников учитель поручает миссию ведущего. Лучше для этой цели выбрать школьника с пониженными музыкальными способностями.

Теперь все ученики включают свои машины. Учитель присылает Шарманщика к участникам концерта, а на машину ведущего отправляется текстовый редактор Микрон.

По полученным заданиям дети набирают мелодии. В это время учитель на основании списка, составленного при распределении заданий, редактирует вместе с ведущим программу будущего концерта. Ведущий набирает программу в Микроне, а затем печатает ее на принтере в нескольких экземплярах (для учительской, классной комнаты, кабинета информатики, для гостей концерта).

Дети, набрав мелодию, прослушивают ее в своих наушниках. Им надо рекомендовать при прослушивании пропеть песню про себя (не мешая другим, не вслух!), используя предложенный им листок со словами. Затем дети передают записанные песни на центральную машину для сохранения на диске. Учитель просит детей запомнить дисковые имена своих мелодий.

Торопить детей не следует, поскольку начало следующего урока-концерта можно будет посвятить завершению, проверке и корректировке мелодий. В конце урока все мелодии должны быть переданы на учительскую машину и записаны на диск независимо от степени завершения.

В конце урока надо обязательно предусмотреть время для разговора о культуре поведения на концерте. Эта беседа, естественно, должна научить детей уметь слушать музыку на любом концерте, а не только на предстоящем уроке. В этой же беседе следует сказать и о поведении исполнителей: песню надо исполнять не для себя, а для слушателей, поэтому исполнитель должен стоять так, чтобы его всем было видно, а петь достаточно громко, так, чтобы его всем было слышно.

Домашнее задание: дети должны выучить дома слова песен, которые им предстоит исполнять в концерте, и спеть песенки несколько раз в порядке репетиции. Ведущий получает домой экземпляр программы, чтобы отрепетировать ее выразительное чтение.

Завершая урок, учитель просит детей передать дедушкам и бабушкам, родителям, которые будут свободны в час урока-концерта, а также учителям и вожатым приглашение посетить концерт их класса.

Урок 6. **Тема урока: Концерт (завершающее занятие по теме).**

Цель урока: **Проверить сформированность навыков работы с музыкальной информацией, усвоить элементарные правила поведения на концерте.**

План урока: **Завершение набора мелодий, проверка (авторские прослушивания) и корректировка.**

Рассказ гостям концерта о работе детей с музыкальным редактором Шарманщик. Концерт.

Методический комментарий.

Сначала учитель предлагает учащимся завершить неоконченные произведения. Прочитывая с диска одну за другой ученические записи, он рассылает их на рабочие места. Дети имеют возможность либо завершить набор мелодий, либо проверить правильность набора и подготовиться к концерту.

Возможна корректировка программы концерта. Ведущий, используя текстовый редактор, готовит оперативный вариант программы.

Пока дети записывают ноты, учитель рассказывает гостям сегодняшнего концерта о музыкальном редакторе Шарманщик, его назначении и возможностях, об умении детей писать и читать ноты, распознавать и корректировать ошибки.

Затем начинается концерт. Программа составлена так, что инструментальные номера чередуются с вокальными.

На машинах со встроенными динамиками есть возможность после окончания сольных номеров концерта сделать по крайней мере два ансамблевых номера. Сначала учитель рассылает всем ученикам одну мелодию для исполнения в унисон. Получив на свои экраны ноты, ученики по дирижерскому жесту учителя одновременно запускают компьютеры (точнее, вызывают Шарманщика нажатием клавиши выполнения).

Для следующего ансамблевого номера учитель рассылает два музыкальных текста на две машины (или две группы машин). Это две партии одного оркестрованного произведения. Компьютеры исполнят две одновременно звучащие разные мелодии, создавая впечатление оркестра.

Наконец, заключительным номером концерта можно рекомендовать исполнение канона, однако он требует наличия в классе кроме встроенных динамиков не менее 3—4 учеников музыкальной школы, некоторых элементарных музыкальных способностей учителя и предварительных репетиций участников канона. Все это заставляет называть такой красивый концертный номер факультативным.

По желанию учеников (выраженному при составлении программы концерта) ансамблевые номера могут сопровождаться хором учеников или вокальными группами.

Пример концертной программы:

1. Бульба — белорусский народный танец.
2. «Вечерний звон» — русская народная песня.
3. «Сулико» — грузинская народная песня.
4. «Во саду ли, в огороде» — русская народная песня.
5. «Два веселых гуся» — русская народная песня.
6. «Светит месяц» — русская народная песня.
7. Лезгинка — грузинский народный танец.
8. «Как пошли наши подружки» — русская народная песня (дуэт компьютеров).
9. «Василек» — русская народная песня (двухголосый канон).
10. «Пастушья песенка» — французская народная песня (двухголосый канон).
11. «Солнечный круг» — песня А. Островского (хор с унисоном компьютеров).

После завершения темы полезно провести экскурсию в музыкальную школу, которую предварительно надо внимательно подготовить. Наилучший вариант такой экскурсии — присутствие на концерте, организованном сверстниками, учащимися музыкальной школы.

Литература

1. *Дуванов А., Зайдельман Я., Первин Ю., Гольцман М.* Роботландия — курс информатики для младших школьников // Информатика и образование. 1989. № 5.

2. *Гольцман М., Дуванов А., Зайдельман Я., Первин Ю.* Обработка текстов в Роботландии // Информатика и образование. 1990. № 6.

В. КАЙМИН, Ю. ЗАВАЛЬСКИЙ

Экспериментальная программа по курсу «Основы информатики и вычислительной техники»

Предлагаемая экспериментальная программа рассчитана на изучение основ информатики и вычислительной техники в средних общеобразовательных и математических школах (для VIII—IX и X—XI классов). Она была составлена в 1988 г. и утверждена для экспериментальной проверки в ряде московских школ Министерством народного образования РСФСР и Московским городским комитетом по народному образованию.

Программа рассчитана на машинный вариант обучения и предполагает использование на школьных ЭВМ учебных интерпретаторов одного из новейших языков программирования — языка Пролог.

Для реализации языка Пролог на школьных ЭВМ были развернуты разработки в Марийском университете под руководством С. Г. Григорьева для машин «Ямаха» и БК-0010, а затем в научно-техническом предприятии «Альтернатива» под руководством А. Н. Гриценко. В настоящее время школы могут получить полные пакеты программ для КУВТ «Ямаха», БК, «Корвет», УКНЦ и «Агат» для поддержки экспериментального курса.

Содержательно предлагаемая экспериментальная программа курса «Основы информатики и вычислительной техники» опирается на понимание информатики как целостной науки о накоплении, обработке и передаче информации в ЭВМ в жизни общества и общении людей. Лучшей иллюстрацией идей, принципов и методов этой науки служит составление самых различных данных, программ и фрагментов баз знаний с использованием средств языка Пролог.

Использование этих средств позволяет продемонстрировать учащимся прикладные аспекты информатики как научной дисциплины, такие, как накопление, систематизация, переработка и предоставление информации с помощью ЭВМ. Этим же целям соответствует решение на ЭВМ задач по математике, физике, истории, географии, биологии и другим школьным дисциплинам путем составления различных классификаций и конкретных баз знаний или их фрагментов.

Особенностью предлагаемого курса является необходимость изучения и освоения (через примеры на ЭВМ) элементов математической логики, составляющей базис языка Пролог и образующей, таким образом, вместе с аппаратом дискретной математики фундамент информатики как научной дисциплины.

Структура программы рассчитана на изучение информатики в VIII—IX и X—XI классах с возможными расширениями. Базой для изучения курса является программа для VIII—IX классов общеобразовательных школ. Ее развитием служат материалы для X—XI классов. Программа предоставляет возможность углубленного изучения курса за счет включения элементов теоретической информатики.

Содержание информатики как учебной дисциплины раскрывается в обсуждении с учащимися таких понятий, как информация, компьютеры, базы данных, программы, накопление, обработка и предоставление информации, численные расчеты на ЭВМ, организация вычислений, решение задач, структурное программирование, анализ правильности алгоритмов, вычислительный эксперимент и математическое моделирование на ЭВМ, разработка программ и информационных систем, проектирование применений ЭВМ, автоматизация производства и информатизация общества, перспективы развития индустрии информатики.

Основной общеобразовательной задачей предлагаемой экспериментальной программы прежде всего является приобретение всеми учащимися компьютерной грамотности — умений работать

на персональных ЭВМ и знания основных возможностей вычислительной техники. Не менее важна и вторая общеобразовательная задача — привить учащимся элементы информационной культуры, а именно умений работать с информацией: собирать, систематизировать, переосмысливать и перерабатывать ее с помощью ЭВМ.

Базисной задачей курса, направленного на развитие у учащихся информационной культуры, становится формирование у них элементов логического мышления, выражающегося в умениях рассуждать, доказывать и обосновывать предлагаемые решения. Использование в этих целях языка Пролог как средства для описания данных и языка логического программирования для ЭВМ дает учащимся непосредственный опыт в технике и технологии конструктивного логического мышления.

Цели углубленного изучения информатики, заложенные в предлагаемую экспериментальную программу, состоят в развитии у учащихся во взаимодействии с углубляемым курсом школьной математики элементов базового математического образования. При этом в объединенную программу по преподаванию интегрированного курса информатики и математики включаются новые, современные разделы дискретной математики — элементы математической логики, теории графов, математической лингвистики и прикладной теории алгоритмов.

Построение курса информатики взаимоувязано с действующим курсом школьной математики и может помочь повысить уровень математической культуры учащихся не только в школах с математическим уклоном, но и в общеобразовательных школах. Необходимыми условиями для этого являются наличие в школе комплекта учебной вычислительной техники (КУВТ БК, «Агат», «Корвет» или УКНЦ), наличие пакета учебных программ поддержки учебника информатики под редакцией В. А. Каймина и учителей, получивших базовое математическое образование в пединституте, университете или техническом вузе.

В предлагаемых задачах курса широко используются материалы из таких школьных предметов, как физика, химия, биология, география, история, литература, русский, национальные и иностранные языки, рисование, черчение, музыка и труд. По этой причине с базовой частью курса полезно ознакомить всех преподавателей средних школ при подготовке их к освоению новых технологий обучения, основанных на использовании в учебном процессе ЭВМ.

Раскрытие применений ЭВМ в науке, технике и производстве — логическое завершение предлагаемого экспериментального курса. Знакомство с элементами автоматизации проектирования, автоматизации производства, с использованием ЭВМ в делопроизводстве и управлении может служить помощью в профессиональной ориентации учащихся средних школ. Углубленное изучение в рамках предлагаемой экспериментальной программы ориентировано на предпрофессиональную подготовку специалистов в области информатизации и программирования.

Включение предлагаемого экспериментального курса в программы общеобразовательных школ вполне укладывается в рамки новых интегративных учебных планов и программ, предложенных Академией педагогических наук СССР и утвержденных Госкомитетом СССР по народному образованию в 1989 г. Для школ с математическим уклоном новые учебные планы и программы, включающие предлагаемую программу интегративного курса информатики и математики, можно найти в проекте «Московский математический лицей», разработанном одним из авторов настоящей программы.

Примерное распределение часов по классам для постановки курса информатики по предлагаемой программе приведено в таблице.

Классы	Общеобразовательные	Математические
VIII	34	136
IX	34	136
X	68	202
XI	34	136

Примечание: дополнительные часы в математических классах выделяются за счет трудового обучения и профориентации.

Основной курс

VIII класс (34 ч)

1. Знакомство с компьютером.

Персональные ЭВМ. Состав и основные возможности. Правила и порядок работы на ЭВМ. Техника безопасности (3 ч).

2. Информация и работа с ней.

Виды информации: тексты, рисунки, мелодии, таблицы и т. д.

Редакторы текстов. Операции редактирования текстов. Запись, считывание и печать текстов (5 ч).

Графический редактор. Рисунки, схемы, чертежи. Операции рисования. Запись, считывание и печать изображений (5 ч).

Музыкальный редактор. Нотная запись. Редактирование и исполнение мелодий на ЭВМ (3 ч.).

Углубленный курс

VIII класс (+102 ч)

1. Принципы работы ЭВМ.

Внутренняя организация ЭВМ. Оперативная и внешняя память. Принцип программного управления. Программное обеспечение ЭВМ. Устройства ввода / вывода и накопления информации.

2. Введение в информатику.

Элементы информации: знаки, точки, звуки и т. д.

Правила редактирования текстов: внесение исправлений, компоновка текстов. Понятие стилистики.

Внесение исправлений в изображения. Составление изображений из фрагментов. Понятие композиции.

Исправление нотных записей. Составление партитур. Понятие музыкальной композиции.

Зачет — 1 ч.

3. Работа с базами данных.

Понятие таблиц. Заполнение и редактирование таблиц. Поиск информации в таблицах. Запись и поиск информации в базах данных (5 ч).

4. Введение в Пролог.

Факты и вопросы в языке Пролог. Правила вывода. Понятие базы знаний (3 ч).

Определение понятий в Прологе. Логические связки «и», «или», «не» (3 ч).

Суждения. Рассуждения. Принципы поиска ответа в Прологе (5 ч).

3. Организация баз данных.

Создание новых таблиц. Работа по сохранению и ведению информации в базах данных. Реорганизация баз данных.

4. Введение в логику.

Элементы формальной логики: понятия и высказывания, сложные высказывания, логические связки.

Симпольная логика: переменные высказывания, дизъюнкция, конъюнкция, отрицание и импликация.

Машинная логика: биты и логические операции. Таблицы истинности. Переключательные схемы.

Зачет — 1 ч.

К концу обучения в VIII классе учащиеся должны иметь следующие знания и умения.

Знать: основные возможности ЭВМ; возможности редакторов текстов; возможности графических редакторов; возможности музыкальных редакторов; основные возможности и правила работы с базами данных; основные средства языка Пролог; правила записи фактов, вопросов и правил вывода; основные логические связки «и», «или», «не»; принципы поиска ответов на сложные вопросы в языке Пролог.

Уметь: включать ЭВМ; работать на клавиатуре; набирать и редактировать тексты; выполнять рисунки и чертежи на ЭВМ; набирать и исполнять мелодии на ЭВМ; заполнять и редактировать таблицы; искать необходимую информацию; записывать информацию в базу данных; вести диалог с базой знаний; формулировать факты и вопросы на языке Пролог; подбирать определения понятий и записывать их на языке Пролог.

IX класс (34 ч)

1. Логические основания ЭВМ.

Логические элементы: «и», «или», «не». Логические схемы и их функции (2 ч).

Предикаты над списками. Принадлежность элемента списку. Включение списков (3 ч).

2. Экспертные системы.

Понятия и обобщения. Понятие ведущего признака. Классификация и систематизация (4 ч).

Непротиворечивость данных. Полнота баз знаний (4 ч).

Законы информатики (3 ч).

Зачет (1 ч).

3. Элементы логического программирования.

Графические предикаты Пролога. Композиции графических предикатов. Рекурсивные описания (3 ч).

Арифметические предикаты. Рекурсивные задания числовых функций (2 ч).

4. Использование языка Пролог.

Композиция графических изображений (2 ч).

Понятие и примеры грамматик. Грамматический разбор (3 ч).

Формулы и их преобразования. Примеры формульных преобразований (3 ч).

Примеры решений математических задач. Организация вычислений в Прологе (3 ч).

IX класс (+102 ч)

1. Элементы математической логики.

Исчисление высказываний. Преобразование высказываний. Правила де Моргана.

Исчисление предикатов. Кванторы общности и существования. Отрицание общих суждений и контрпримеры.

2. Элементы теории доказательств.

Основные правила дедуктивного вывода. Принципы индукции.

Понятие об аскиоматических теориях.

Понятие конструктивного доказательства. Математический синтез. Экзамен.

3. Введение в теорию рекурсий.

Конструктивные объекты: списки, деревья, сети. Рекурсивные функции над списками.

Числовые функции и их графики.

4. Введение в рекурсивный анализ.

Элементы аналитической геометрии: координатный метод. Примеры построений сложных композиций.

Примеры порождающих грамматик.

Тождества и равносильные преобразования. Примеры равносильных преобразований.

Смешанные вычисления. Конкретизация правил.

Зачет — 1 ч.

К концу обучения в IX классе учащиеся должны иметь следующие знания и умения.

Знать: понятие логических схем и их функций; понятие списка и предикатов над списками; понятие ведущего признака и обобщения; понятие классификации и систематизации; понятие противоречия; понятие необходимых и достаточных условий; законы информатики; графические возможности Пролога; понятие рекурсивных определений и описаний; рекурсивные определения арифметических функций; принципы композиции и преобразования геометрических фигур на Прологе; понятия грамматики и грамматического разбора; правила записи формул и их преобразований на языке Пролог; принципы организации вычислений в языке Пролог.

Уметь: читать логические схемы и описывать их функции; составлять простейшие предикаты над списками; выделять ведущие признаки понятий; систематизировать факты; предлагать обобщения; выявлять противоречия в данных; выявлять нетождественность понятий; проверять необходимость и достаточность условий; описывать на Прологе простейшие геометрические фигуры; читать и интерпретировать рекурсивные определения; составлять описания геометрических фигур на Прологе; читать и интерпретировать грамматические правила; описывать формулы и их преобразования на языке Пролог; описывать условия математических задач на Прологе.

X класс (68 ч)

1. Вычисления на ЭВМ.

Электронные таблицы на ЭВМ: ввод формул, данных и выполнение вычислений (3 ч).

Понятие алгоритма. Примеры и свойства алгоритма (3 ч).

Примеры постановок и методов решения задач (3 ч).

2. Решение задач на ЭВМ.

Понятие и виды моделей. Постановка задач и выбор моделей (3 ч).

Выбор методов решения. Обоснование их правильности (3 ч).

Постановка решения задач на ЭВМ. Понятие сценария (3 ч).

Алгоритмы и их исполнители. Систематическое конструирование алгоритмов (3 ч).

Проверка алгоритмов и программ. Понятие теста. Испытание программ на ЭВМ (3 ч).

Понятие правильности алгоритмов и программ. Понятие надежности исполнителей и программ (3 ч).

Понятие конечного результата. Анализ и планирование промежуточных результатов (3 ч).

Зачет — 1 ч.

3. Начала программирования.

Переменные и присваивание в алгоритмах и программах (2 ч).

Средства ввода / вывода.

Ветвления (3 ч).

Средства машинной графики (3 ч).

Циклы со счетчиком. Циклы с завершением по условию (4 ч).

Массивы и примеры их использования (6 ч).

Процедуры и функции. Примеры их использования (6 ч).

Языки: Рапира или Паскаль — по выбору (24 ч).

4. Вычислительный эксперимент.

Понятие вычислительного эксперимента. Примеры постановки и проведения вычислительных экспериментов:

а) для физических процессов;

б) в биохимических задачах;

в) для решения экологических проблем;

г) для анализа социодинамических моделей (9 ч).

Зачет — 1 ч.

X класс (134 ч)

1. Организация вычислений.

Точность вычислений. Погрешности. Представления чисел в ЭВМ.

Порядок составления алгоритмов и программ.

Понятие правильности алгоритма.

2. Доказательное программирование.

Анализ результатов выполнения алгоритмов.

Классификация ошибок. Техника выявления ошибок.

Понятие спецификаций программ. Описание функций алгоритмов и программ.

Пошаговая детализация алгоритмов. Модульная организация программ.

Понятие полноты тестов. Правила подбора тестов. Недостаточность тестирования.

Доказательство правильности алгоритмов и программ. Примеры.

Анализ сходимости методов.

Адекватность моделей. Информация как категория. Полнота и достоверность информации.

3. Численные методы.

Вычисления многочленов.

Построение графиков.

Интерполяция функций. Приближенные вычисления функций.

Методы приближенного решения уравнений.

Точные и приближенные методы решения систем уравнений.

Экстремальные задачи. Примеры решения задач линейного программирования.

Аппроксимация функций многочленами.

4. Математическое моделирование.

Понятие математической модели. Примеры моделей в задачах физики, химии, биологии, социологии и экономики.

Разностные схемы.

Интерполяционные модели.

Простейшие дифференциальные уравнения.

Экзамен (по выбору учащегося).

К концу обучения в X классе учащиеся должны иметь следующие знания и умения.

Знать: основные возможности электронных таблиц; понятие и примеры алгоритмов; понятия постановки и метода решения задачи; понятие и виды моделей; понятие правильности метода решения; примеры простейших сценариев; примеры простейших алгоритмов; понятие теста; понятие правильности алгоритмов и программ; понятие надежности алгоритмов и программ; примеры составления планов решения задач; понятие процедур и функций в программах; понятие вычислительного эксперимента; примеры математических моделей из физики, биологии и т. п.

Уметь: проводить вычисления с помощью электронных таблиц; приводить примеры алгоритмов; описывать информационно-логические модели; проводить проверку методов решения на примерах; составлять простейшие сценарии; составлять простейшие алгоритмы; подбирать тесты для проверки алгоритмов; проводить доказательства правильности для простейших алгоритмов; выявлять элементы ненадежности в алгоритмах; составлять планы решения задач; выделять и описывать процедуры и функции в программах; проводить на ЭВМ вычислительный эксперимент по заданной программе; интерпретировать получаемые результаты.

XI класс (34 ч)

1. ЭВМ в производстве:
станки с ЧПУ;
контрольно-измерительные приборы и системы;
управление роботами;
встроенные микропроцессоры (8 ч).

2. ЭВМ в управлении:
делопроизводство и оргтехника;
планирование и контроль поручений;
распределение ресурсов (8 ч).

3. ЭВМ в проектировании:
инженерная графика;
инженерное проектирование;
архитектурное и строительное проектирование (6 ч).

4. ЭВМ в науке:
библиотечное обслуживание;
управление экспериментами;
моделирование на ЭВМ (6 ч).

5. ЭВМ в обществе:
информационные фонды;
информационные сети;
экспертные системы (5 ч).

Зачет — 1 ч.

К концу XI класса учащиеся должны иметь следующие знания и умения.

Знать: примеры использования ЭВМ в производстве; примеры использования ЭВМ в делопроизводстве; примеры использования ЭВМ в проектировании; примеры использования ЭВМ в науке; примеры и перспективы использования ЭВМ в обществе.

XI класс (68 ч)

1. Обработка данных:
организация и обработка файлов;
организация и обработка списков;
синтаксический контроль.

2. Технология программирования:
независимая трансляция модулей;
нисходящее проектирование;
расширение программ.

3. Проектирование программ:
спецификация программ;
оптимизация программ;
унификация и стандартизация программ.

4. Системное программирование:
операционные системы;
распределенные базы данных;
инструментальные средства.

5. Перспективы развития ВТ:
ЭВМ пятого поколения;
оптические компьютеры;
биокомпьютеры.

Литература

1. *Ковальский Р.* Логика в решении задач. М.: Мир, 1990.

2. *Братко И.* Программирование на языке Пролог для искусственного интеллекта. М.: Мир, 1990.

3. *Каймин В. А. Щеголев А. Г.* и др. Основы информатики и вычислительной техники: Пробный учебник для X—XI классов. М.: Просвещение. 1989.

Проект «Культура, наука, образование: США — СССР»

В июне 1990 г. во время встречи Президентов СССР и США М. С. Горбачева и Д. Буша был подписан Меморандум о взаимопонимании по проекту «Культура, наука, образование: США — СССР», направленный на совершенствование системы образования посредством широкого внедрения в средних школах, институтах и других учебных заведениях современных методов обучения с использованием компьютеров, а также на развитие науки и культуры путем компьютеризации и применения новейших информационных технологий. На реализацию данного проекта потребуется свыше 10 лет. В рамках проекта в первую очередь будет осуществляться деятельность в области практического внедрения и использования новых информационных технологий в дошкольных учреждениях, средних школах, школах для детей-инвалидов, профессионально-технических учебных заведениях, школах бизнеса, университета, инженерных вузах, различных специальных учебных центрах в области культуры, библиотечном деле.

Образовательный проект состоит из ряда этапов, по каждому этапу между сторонами-исполнителями заключается отдельное соглашение. Для выполнения первого этапа, предусматривающего поставку в СССР оборудования для более чем 1000 школ, общей стоимостью 20 млн. долларов США и рассчитанного на 1990—1991 гг., между фирмой IBM и Гособразованием СССР был заключен договор.

Прошел год. За это время выполнен большой объем работы по трем направлениям: «Нетрудоспособные дети и инвалиды», «Пилотные школы», «Школы бизнеса».

Проект «Нетрудоспособные дети и инвалиды»

С апреля 1990 г. проводится эксперимент по использованию новых информационных технологий в специальных школах № 1 (для слепых детей) и № 101 (для глухих детей) г. Москвы, создан центр поддержки проекта «Инвалтех» в составе НПО «ИНФОТЕХ», проведена подготовка инженеров и методистов региональных центров, завершается работа по организации 7 центров на базе дефектологических факультетов педагогических институтов в гг. Москве, Ленинграде, Ташкенте, Алма-Ате, Свердловске, Минске, Киеве, проведена конференция дефектологов-практиков по использованию новых информационных технологий в обучении глухих, подготовлены и переданы на тиражирование программно-методические комплексы по Screen Reader и Speech Viewer для использования в 4 школах для слепых детей и 33 школах для глухих.

Проект «Пилотные школы»

С сентября 1990 г. продолжается эксперимент по использованию новых информационных технологий в образовании в школах г. Москвы №№ 1234, 1217. В 1990 г. организован центр поддержки проекта на базе компьютерного учебно-демонстрационного и информационно-издательского сервисного центра («АП КУДИЦ»), проведена подготовка республиканских и региональных менеджеров проекта, завершена подготовка сервис-инженеров и методистов региональных центров, подготовлен и передан на тиражирование первый программно-методический комплекс по курсу информатики для 1000 школ, участвующих в проекте. Кроме того, подготовлены и опробованы непосредственно учителями и детьми в школах программно-методические комплексы для начального образования, для курсов физики, химии, географии, английского языка.

Начали работу 30 региональных центров из запланированных 34, в которых прошли подготовку на данный момент около 650 учителей школ, завезено и установлено оборудование фирмы IBM в 250 школах.

Проект «Школы бизнеса»

В марте 1991 г. создан центр поддержки на базе Государственной академии управления. Определены 15 рес-

публиканских центров проекта, с апреля с. г. в центре поддержки начата подготовка республиканских менеджеров и сотрудников республиканских центров. Ведется работа по созданию методических материалов и программного продукта по курсам прикладного менеджмента и по программе подготовки магистров в области управления.

В рамках проекта Гособразование СССР и фирма IBM провели два международных симпозиума «Новые информационные технологии в образовании».

Для согласованных действий по республиканским проектам подготовлено Межреспубликанское соглашение, создается Координационный совет, в который войдут представители всех республик. До конца года завершится создание инфраструктуры по управлению проектом, пройдут подготовку по использованию новых информационных технологий в образовании около 3000 преподавателей, более чем в 1000 учебных заведений будут установлены компьютеры фирмы IBM с качественным программно-методическим обеспечением.

Пилотные образовательные проекты позволяют реализовывать новую стратегию в области использования технических средств, программного обеспечения, интегрирования этих проектов с современными компьютерными технологиями, вхождения в европейские и мировые информационные сети.

Сейчас в Гособразовании СССР разработаны программы новых пилотных проектов в области инженерного и продолженного инженерного, педагогического, среднего специального, профессионально-технического, экологического образования, а также программа «Пилотный детский сад». Уже в ближайшие месяцы по каждому из этих проектов будут созданы центры поддержки, оборудованы экспериментальные учебные заведения и детские сады. Начнется новый этап большого проекта «Культура, наука, образование: США — СССР», реализуемого Гособразованием СССР и фирмой IBM.

Расширение работ по проекту предусматривает приобретение в 1991 г. еще около 1000 комплектов вычислительной техники (11 000 персональных компьютеров с соответствующей периферией) стоимостью 25 млн. долларов США. Общие затраты по проекту до 1995 г. составят 205 млн. долларов США. Предполагается использовать эту технику с соответствующим программно-методическим обеспечением на всех уровнях образования, начиная с детских садов и кончая высшими учебными заведениями. Полученные в ходе реализации проектов результаты дают возможность говорить об их исключительной важности для сферы народного образования, для последующих шагов ее развития.

LinkWay — мультимедиа для IBM PC

История создания системы

В последние годы в компьютерном мире большую популярность приобрели системы, объединенные общим названием — гипертекстовые. Фирма IBM, немного задержавшись на старте, решила взять под свой контроль эту, весьма перспективную, ветвь программного обеспечения, лицензировав разработку и распространение системы

Создал систему LinkWay Larry Kheriaty. Находясь под впечатлением системы HyperCard (торговая марка Apple Computer Inc.), Kheriaty в конце 1987 г. решил сделать нечто подобное для IBM PC. Кроме того, у него имелся собственный взгляд на возможности компьютеров и их место в современном мире.

В марте 1988 г. автор показал работу в отделении IBM по образованию в Атланте, и фирма, будучи знакома с работами Kheriaty в области создания интерактивных видеосредств, закупила

права на эту систему. Под кодовым названием Marlin фирма IBM инициировала работу над проектом, который в конечном итоге стал продуктом IBM LinkWay. В 1989 г. на рынке появилась первая версия LinkWay, в середине 1990 г. была выпущена вторая, существенно расширенная.

Текст, гипертекст, мультимедиа

Понятия «гипертекст» и «мультимедиа» вошли в современный лексикон благодаря развитию средств вычислительной техники и появлению персональных компьютеров с мощным процессором и развитой периферией.

Пользователи компьютеров, по-видимому, обращали внимание на то, что читать с экрана представленную в линейном виде текстовую информацию не всегда удобно. Если, к примеру, в некоторой системе имеется пространный пояснительный материал («Help»), который можно просматривать, «листая» на экране монитора страницу за страницей, то гораздо удобнее выдать такой материал на печать и изучать бумаги традиционным способом.

В то же время мало кому придет в голову печатать «Help» в системе Turbo C (разработка фирмы Borland), поскольку просматривать с экрана структурно организованную информацию значительно удобнее. Подсказка организована таким образом, что если в тексте имеются некоторые понятия, то можно сразу перейти к тем частям текста, где эти понятия раскрываются. Здесь мы встречаем типичный пример гипертекста.

В литературе имеется немало определений гипертекста. Приведем наиболее лаконичное, и при этом достаточно полное.

«Гипертекст — это форма организации текстового материала, при которой его единицы представлены не в линейной последовательности, а как система явно указанных возможных переходов, связей между ними» [3].

Работу с гипертекстовой системой можно сравнить с чтением энциклопедии. В отличие от романа или статьи, которую читают, как правило, последовательно от страницы к странице, с энциклопедией работают по-другому: встретив в последовательном тексте ссылку на какое-то понятие, читатель либо переходит к странице, где это понятие объясняется, либо продолжает чтение текста дальше. В текст могут быть вкраплены иллюстрации, карты, схемы, иногда требуется уточнить понятие, встретившееся на иллюстрации. После изучения уточняющего материала (в котором могут встретиться понятия, также нуждающиеся в уточнении) читатель возвращается к исходной точке. Но можно представить ситуацию, когда к исходной точке возвращаться не надо, а следует перейти к совершенно иному разделу текста.

Если материал, по форме напоминающий текст энциклопедии, размещен в памяти компьютера и есть программа, позволяющая «бродить» по такому материалу в произвольной последовательности, то можно говорить, что мы имеем дело с гипертекстовой системой.

Система LinkWay является гипертекстовой системой, относящейся к более мощной разновидности — системам мультимедиа (multymedia). Принципиальной особенностью систем мультимедиа является дальнейшее развитие идеи ассоциативно связанной информации, распространяющееся на графическую и звуковую информацию, хранящуюся в ЭВМ в цифровой форме [4].

Основная часть возможностей системы может быть прочувствована ее пользователями только в том случае, если компьютер оснащен дополнительным оборудованием: речевым и музыкальным адаптерами, видеодиском, видеокамерой и т. д.

Назначение системы LinkWay

В работе [1] перечислены следующие области применения системы LinkWay:

разработка демонстрационных «роликов» по разным темам;

построение уроков в гипертекстовой манере;

организация персональной базы данных и настольной канцелярии;

управление внешними устройствами;

построение оболочки операционной

системы или пакетов прикладных программ.

Нас будут интересовать возможности системы LinkWay прежде всего как инструмента для построения уроков.

Материалы к урокам могут включать информацию, представленную в текстовой и графической форме. С помощью системы можно строить уроки, сочетающие в себе как методичное, так и управляемое (направляемое) учеником изложение материала. Возможность воздействовать на процесс обучения позволяет ученику приспосабливаться под удобный для него темп и стиль подачи материала.

Заметим, что типичный урок, поддерживаемый компьютером, включает, как правило, элементы гипертекстовой работы. Действительно, при самостоятельном изучении материала учащемуся может понадобиться вернуться к описанию какого-то понятия, посмотреть материал чуть-чуть вперед, больше времени потратить на изучение очередного понятия. При этом урок развивается уже не в строгих рамках последовательных шагов, предопределенных учителем или автором урока, в определенной степени управляется учеником.

Средства программирования системы (встроенный язык SCRIPT) дают возможность организовывать зачетную часть урока, т. е. проверить глубину усвоения материала. На языке программирования SCRIPT можно написать программы анализа ответов учащегося и выдать некоторые интегральные характеристики относительно темпа и полноты изучения материала.

Мы активно использовали систему LinkWay в своей работе и сейчас можем сформулировать нашу оценку перспективы ее использования в качестве инструмента построения компьютерных уроков.

LinkWay может быть с успехом использована во многих предметах. Главное ее преимущество — быстрое воплощение идей учителя и возможность их апробации в условиях реального учебного процесса.

В качестве характерного примера можно привести урок географии, на котором осуществляются переходы от карты континента к картам стран в более крупном масштабе и далее к отдельным регионам. При этом на экране появляется разного рода справочная информация. LinkWay позволяет проводить уроки по языковым дисциплинам (по русскому, английскому и другим языкам), пока объем материала и его структура не требуют подключения средств специализированных баз данных и словарей.

С точки зрения общей структуры LinkWay представляет собой набор инструментариев (toolkit), решающих те или иные задачи, возникающие в процессе разработки уроков. К числу основных относятся: система поддержки среды создания и редактирования уроков, графический редактор, текстовый редактор, редактор шрифтов, интерпретатор подготовленных уроков, инструмент преобразования файлов с картинками и палитры в формат LinkWay. Вызов инструментариев выполняется либо из среды через меню системы, либо запуском выполняемых программ (файлов с расширением .com и .exe) из среды операционной системы DOS.

Основные понятия системы

Будем считать, что самым крупным понятием, с которым имеют дело ученик и учитель, является урок.

Уроки состоят из ПАПОК (FOLDER). В одной папке учитель собирает одну или несколько СТРАНИЦ (PAGE), которые, как правило, содержат сходную по форме и содержанию информацию. Удачное разделение урока на папки, а папок на страницы облегчает разработку урока, повышает его методическую ценность.

Папкам присваиваются уникальные имена. В каждой папке содержится БАЗОВАЯ СТРАНИЦА (BASE PAGE). На ней размещается информация, общая для всех страниц папки, например, общие заголовки, рамки, картинки и т. д. Остальные страницы папки нумеруются по порядку (1, 2, ...). Кроме того, каждая страница автоматически получает УНИКАЛЬНЫЙ НОМЕР (ID), однозначно идентифицирующий ее в папке. При визуализации страницы складываются изображения

очередной и базовой страниц.

На странице размещаются ОБЪЕКТЫ (OBJECT). Различают следующие типы объектов:

КАРТИНКА (PICTURE), позволяющая показать изображение;

ПОЛЕ (FIELD), содержащее текстовые строки;

КНОПКА (BUTTON), на которую можно «нажимать» с помощью манипулятора типа «мышь», с которой связаны различные действия.

Кроме того, на странице можно нарисовать рамки и линии, улучшающие читабельность текста.

При создании объекта типа «текст» задается количество символов в строке, количество строк в тексте, шрифт и цвет символов.

Для размещения картинки нужно указать место и размеры окна на странице и файл с графическим изображением.

Объекты типа «кнопка» также занимают участок страницы, но в отличие от объектов первых двух типов могут не иметь визуального представления. Это позволяет создавать на странице невидимые кнопки. Кнопки могут накладываться на тексты и картинки. Если кнопки не имеют собственных графических образов, изображение объектов при этом не изменяется.

При наложении объектов разных типов они проявляются или экранируют друг друга. Тексты и кнопки относятся к числу «прозрачных» объектов, через которые проявляются изображения фона. Картинки являются «прозрачными» по отношению к текстам и кнопкам, но экранируют друг друга. Кроме того, картинки закрывают «проявляющиеся» с базовой страницы изображения.

С кнопками связываются различные действия:

переход к новым страницам и новым папкам (кнопки типа LINK и GO),

поиск заданной информации в текстовых полях (кнопки типа FIND),

вывод на экран окна с текстовой информацией (кнопки типа TEXP POP—UP),

вывод на экран картинок, которые исчезают при нажатии клавиши «мышь» (кнопки типа PICTURE POP—UP),

выполнение программы, написанной на языке SCRIPT (кнопки типа SCRIPT),

переход к просмотру или редактированию текстовых документов (кнопки типа DOCUMENT).

Таким образом, с помощью кнопок можно организовать работу с информацией в стиле гипертекстовых систем.

Объекты могут иметь имена. Именовать объекты имеет смысл в том случае, когда планируется обращаться к объектам из программ, связанных с кнопками. Кроме того, для кнопок имена могут стать их графическими образами.

Важная техническая деталь, о которой необходимо знать пользователю,— графический режим. Он определяет качество выводимого на экран изображения. Две характеристики являются существенными для графического режима: разрешающая способность, измеряемая числом точек (пикселей) изображения, размещаемых в строке и столбце экрана, и число цветов, в которые может быть окрашена каждая такая точка.

Система LinkWay поддерживает несколько графических режимов (в скобках указывается разрешение и количество цветов): CGA (320×200, 4 цвета), MCGA mono (640×480, 2 цвета), MCGA 256 (320×200, 256 цветов), EGA (640×350, 16 цветов), VGA (640×480, 16 цветов).

Единственное ограничение, которое накладывает система на использование графических режимов, состоит в том, что каждая папка (все ее страницы) готовится в одном графическом режиме. Однако в урок могут включаться папки, подготовленные в разных режимах.

Язык программирования

Одним из интересных свойств системы LinkWay, которое выгодно отличает ее от других систем подобного рода, является наличие встроенного языка программирования SCRIPT. С помощью языка SCRIPT описываются действия, связанные с кнопками типа SCRIPT.

Язык SCRIPT включает такие понятия, как «оператор», «переменная», «выражение», «процедура». Переменные вво-

дятся выполняемой инструкцией описания переменных и становятся доступными для всех папок, включенных в урок. Переменные, описанные в процедуре, локальны в ней.

В языке нет понятия типов. Переменные можно сравнивать как текстовые строки или как числа.

В языке имеются средства управления объектами, а также папками и страницами папок.

Предусмотрены операторы записи информации в текстовые поля. Кроме того, можно перевести текстовое поле в режим ввода и проанализировать введенную пользователем информацию.

В программах на языке системы можно инициировать кнопки, которые будут выполнять связанные с ними действия таким же образом, как в случае выбора кнопок при помощи «мыши». Имеется возможность организовать ожидание нажатия клавиши «мышь» и распознать имя и тип выбранного «мышью» объекта.

В качестве программной единицы выступает связанная с кнопкой типа SCRIPT последовательность операторов, которую мы будем называть «программа».

Поток управления в программе организуется с помощью условного оператора, оператора перехода на метку, оператора вызова процедуры, а также с помощью передачи управления в другую программу. Для тех, кто знаком с программированием, обратим внимание, что по завершении программы, не являющейся процедурой, возврат в инициирующую программу (обычно реализуемый операторами return или end. subroutine) не происходит.

Передать управление можно в программу, размещенную на текущей странице папки, либо программе, определенной на другой странице, предварительно объявив страницу в качестве доступной. В программе можно вызвать процедуру, находящуюся на текущей странице или на другой странице текущей папки.

При вызове папки из каталога или при запуске системы LinkWay с именем папки в качестве параметра управление автоматически передается процедуре с именем AUTOEXEC, которая должна быть размещена на базовой странице. Эта процедура является своего рода головной программой, которая реализует начальные действия и инициирует другие программы папки.

Программа на языке SCRIPT может вызвать из каталога другую папку, при этом управление передается программе, связанной с кнопкой AUTOEXEC. При вызове папки можно установить новый режим работы монитора (если компьютер предусматривает данный режим).

В системе фаза анализа предложений совмещена с фазой выполнения, поэтому сообщения об ошибках в программе пользователь получает в момент выполнения «неправильного» оператора.

Одним из полезных свойств языка SCRIPT является его открытость для подключения внешних файлов и программ, реализованных в других средах.

Кнопка типа SCRIPT может вызвать программу, написанную на ассемблере и преобразованную в BIN-формат (файл с расширением. BIN). Файл с программой предварительно загружается в переменную и инициируется специальной командой. Данные между программами передаются через память, доступную обеим программам.

Из языка SCRIPT можно выйти в DOS и выполнить любую его команду, в том числе запустить выполняемый файл. Если для вызываемой программы не хватает оперативной памяти, то ее можно вызвать в режиме выгрузки из оперативной памяти системы с последующей загрузкой системы после отработки программы.

Средства анимации

Последняя версия системы LinkWay (версия 2.0) предоставляет средства анимации объектов, обращения к экрану в терминах координат точек и средства управления манипулятором «мышь» низкого уровня.

Возможность получать на экране движущиеся изображения является одним из главных требований, которые разработчики компьютерных уроков предъявляют к инструментальным системам.

В системе LinkWay имеется операция перемещения объектов. Этой операцией можно воспользоваться для получения эффекта движения по экрану объектов типа «текст» и «кнопка».

Для получения движущихся картинок (мультипликации) лучше воспользоваться операциями чтения и записи образа части экрана BITGET и BITPUT. Полученную битовую карту (bitmap) можно хранить в переменных или записывать во внешнюю память.

Низкоуровневые операции управления «мышью» можно использовать для организации циклов, в которых анализируются текущие координаты указателя «мыши». В зависимости от положения указателя на экране можно организовать выполнение тех или иных действий. При помощи таких программ достигается следующий эффект «оживления» картинок: в тот момент, когда ученик укажет на животного, оно начнет, например, двигать хвостом, а когда указатель «мыши» переместится в другое место экрана, движение хвоста прекратится.

LinkWay — инструмент спецификации и быстрого прототипирования

В предыдущих разделах описывались возможности системы LinkWay как инструмента разработки компьютерных уроков. При этом имелись в виду задачи, связанные с этапом реализации уроков.

В действительности разработка компьютерных уроков является длительным и трудоемким процессом, в котором этап реализации уроков средствами инструментальной системы (или среды программирования) играет зачастую далеко не главную роль.

Те, кто сталкивался с аналогичными проблемами разработки программного обеспечения, знакомы с цифрами, приводимыми во многих исследовательских работах по технологии программирования: этап написания программ в среднем занимает 10—20 % общего времени разработки продукта. Остальное время примерно в равной пропорции делится между этапом начального проектирования и этапом тестирования и отладки.

Многие авторы отмечают, что наиболее ответственным этапом разработки программ является начальный этап, включающий разработку спецификаций, структурный анализ системы, разработку интерфейсов и т. д. Тщательно выполненные спецификации и структурный анализ определяют в конечном итоге качество и надежность программного продукта.

В то же время начальные этапы проектирования систем в настоящее время в меньшей степени обеспечены инструментальной и компьютерной поддержкой. По этим же причинам процесс создания программного обеспечения многие все еще склонны относить к области искусства.

Возвращаясь к вопросам разработки компьютерных уроков, выскажем наше видение проблемы, отраженное на рисунке.

Этапы разработки компьютерных уроков

Разработкой компьютерных уроков занимается группа специалистов, в которую входят: методист-предметник, программист, психолог, художник, учитель. Разумеется, участники разработки могут быть специалистами одновременно в нескольких областях.

Работа в группе начинается с подготовки спецификаций — технического задания, в котором описываются общие свойства продукта, логика подачи материала, структура уроков, типовые упражнения, содержание зачетной части уроков и т. д.

Далее имеет смысл подготовить пакет компьютерного урока, в котором будут достаточно полно отражены все компоненты урока, их взаимодействие, возможные переходы и пр. Макет может представлять собой вариант урока, в котором, например, отсутствуют сложные графические объекты или эффекты и реализована только небольшая часть упражнений из числа намеченных. На макете можно проверить качество спроектированного интерфейса с учащимся, эффективность выбранной методики подачи учебного материала.

После подготовки и апробации макета начинается этап реализации пол-

ного варианта компьютерного урока, включающего все компоненты (функции, графику, сложные вычисления), которые в макете были только эскизно намечены.

В практике очень редко удается создать конечный продукт, что называется, с первого захода. Не случайно вскоре после выпуска какой-либо системы разработчики объявляют о появлении улучшенной версии продукта. В применении к компьютерным урокам новая версия с большой вероятностью может появиться после этапа апробации программ в реальном учебном процессе.

Именно использование продукта в школе дает окончательную оценку заложенным в урок методикам, качеству выполненной программной реализации, уровню художественного оформления.

После этапа апробации разработчикам компьютерных уроков зачастую приходится возвращаться к начальным этапам — этапу спецификаций и подготовки макета, для того чтобы отразить все те изменения, необходимость которых была выявлена на занятиях в школе.

Разумеется, приведенная схема создания компьютерных уроков, как, по-видимому, и любая другая схема, в известной степени условна. В некоторых разработках бывает весьма сложно строго выделить и обозначить границы всех перечисленных этапов. Приводя подобную схему, мы не ставим целью навязать некую догму, а хотим прежде всего очертить круг проблем, которые возникают в процессе разработки компьютерных программ.

Следующий тезис, который следует из приведенных выше рассуждений, заключается в том, что решающим моментом в разработке компьютерных уроков является, на наш взгляд, организация работ в группе, взаимодействие участников проекта в процессе создания продукта, их настроенность на длительную работу, в которой возможны возвраты к исходным точкам и пересмотры принятых ранее решений.

Если под этим углом зрения рассмотреть возможности инструментальных систем, то система LinkWay представляется весьма перспективной.

С помощью LinkWay достаточно просто и эффективно можно выполнить спецификацию компьютерного урока: обрисовать общую схему урока со всеми возможными переходами, подобрать подходящую форму и место подсказок, разработать типы меню, содержание полей меню и т. д. Спецификации можно подготовить в форме, понятной не только специалисту в программировании, но и другим участникам проекта.

Реализованные в среде LinkWay папки-спецификации выглядят намного привлекательнее кипы бумаг с согласованиями и резолюциями. Главное достоинство «живых» спецификаций — их приспособленность к внесению изменений, в то время как судьба бумаг с «последними окончательными решениями» печально известна.

Средствами системы LinkWay можно подготовить макет урока с некоторым набором типовых упражнений. Участники группы (методист, психолог, учитель и др.) могут поработать с макетом и высказать свои суждения, не дожидаясь появления полного варианта урока.

LinkWay позволяет быстро выполнить подобные эскизные прорисовки уроков. И этим его качеством имеет смысл пользоваться даже в том случае, когда реализацию уроков планируется вести с помощью других средств, например языков программирования С, Паскаль, Пролог.

LinkWay в проекте «Пилотные школы»

Инструментальная система IBM LinkWay поставляется в рамках проекта «Пилотные школы» в региональные центры. В пилотные школы будут поставляться разработанные в среде компьютерные уроки вместе с версией системы, выполняющей интерпретацию подготовленных уроков (run-time version).

С помощью системы LinkWay уже реализовано несколько учебных программ по информатике, английскому языку, физике, математике. Опыт работы показал, что система LinkWay проста в освоении и доступна широкому кругу специалистов, работающих в области образования.

Литература

1. IBM LinkWay. Hypermedia for the PC // R. Harrington, B. Fancher, P. Black.— John Wiley a Sons Inc., 1990.

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. *Дементьева Л. Б.* Системы гиперзаписи информации // НТИ. Сер. 2 1990. № 2.

5. *Киселев Б. Г., Леонова Н. М.* Организация использования ЭВМ в ходе компьютерного урока. М.: МИФИ, 1987.

6. IBM LinkWay User's Cude.

7. *Гласс Р.* Руководство по надежному программированию. М.: Финансы и статистика, 1982.

Е. БАКМАН, И. ДРЕМОВА
Черниговский педагогический институт

Введение в микро-Пролог

Язык Пролог «родился» во Франции в 1971 г. Автором его был Алэн Колмероэ. Название Пролог получил от сокращенного «ПРОграммирование ЛОГическое», что указывает на объединение в нем идей математической логики и программирования.

По сравнению с процедурными языками, к которым относятся, например, Паскаль, Бейсик или школьный алгоритмический язык, Пролог является языком совершенно иного типа. Вместо подробного описания выполняемых команд Пролог использует знания о предметной области почти в таком же виде, как это делают люди в письменных текстах. Такая форма представления знаний называется декларативной.

Пролог может запоминать сведения и отвечать на вопросы вида «Верно ли ...», «Кто ...», «Где ...» и т. д. Базы данных тоже осуществляют поиск и выдачу ответов на запросы, однако они выполняют только механический поиск в заданном перечне известных фактов. Про людей, которые действуют подобным образом, говорят, что их знания репродуктивные. Пролог же кроме фактов запоминает еще и правила. Правила — это знания о том, как из известных фактов можно вывести новые. Поэтому знания в Прологе имеют более компактный вид. Люди с подобным складом мышления рассуждают примерно так: «Я знаю несколько фактов, и я знаю, как из них можно получить другие факты. Если понадобится, то я их выведу».

Факты и правила из определенной области знаний называют базой знаний. Создание базы знаний требует от учащихся навыков работы с информацией. Поэтому сам процесс разработки базы формирует умения накапливать информацию, анализировать и обрабатывать ее, развивает логическое мышление.

Такие базы знаний могут быть разработаны по любому предмету из школьного курса. Это дает возможность реализовать межпредметные связи информатики с другими дисциплинами, обеспечить индивидуальный подход в изучении ОИВТ, развивать эрудицию и культуру учащихся.

Некоторое время Пролог привлекал внимание только исследователей. Однако в последние годы актуальность проблемы искусственного интеллекта, необходимость разработки и использования всевозможных экспертных систем, сетей связи, позволяющих общение на языке, близком к естественному, заставили специалистов коренным образом изменить свое отношение к Прологу.

Хотя за рубежом Пролог уже стал одним из самых популярных языков искусственного интеллекта, а японский проект вычислительных систем пятого поколения основан на Прологе как на наиболее перспективном языке представления знаний, в нашей стране он еще не получил столь широкого распространения. Сдерживающим фактором является ограниченный выбор учебной литературы по Прологу. Для начального изучения можно предложить только [1—3]. Книги же [4—6] носят специальный характер. Тем более отрадно, что журнал «Информатика и образование» уже поместил семь публикаций [7—13] о языке Пролог.

Цель настоящей статьи — дать более полное описание версии микро-Пролога, реализованной на ПЭВМ «Ямаха MSX-2» Институтом программных систем АН СССР.

Основные понятия

Для работы на микро-Прологе, реализованном на ПЭВМ «Ямаха MSX-2», необходимо наличие двух файлов: P.COM, содержащего исполнительную часть Пролога (систему), и MICROPRO.P, в котором находится основная база знаний. Кроме этих двух файлов пользователь может создавать собственные базы знаний и присоединять их к основной базе.

Запуск микро-Пролога осуществляется командой P. При этом в ОЗУ загружается файл P.COM, который, в свою очередь, загружает основную базу знаний — файл

MICROPRO.P. Когда загрузка заканчивается, на экране появляется приглашение — знак подчеркивания и точка. Знак подчеркивания означает, что предыдущая операция завершена, а точка — приглашение к вводу. Если в процессе работы будет выведена только точка, то это означает, что для завершения операции требуется ввести еще какие-то данные.

Что же представляет собой Пролог? Он напоминает двухлетнего ребенка, который кое-что умеет делать в силу врожденных способностей, но многому его еще нужно научить.

Ребенок может повторить слова взрослого, не вникая в их смысл. Пролог это тоже умеет. В ответ на приглашение введите

```
PP здравствуйте
```

Пролог напечатает слово «здравствуйте». Команда PP соответствует команде PRINT Бейсика или WRITELN Паскаля. Если вместо PP дать команду P, то печать произойдет без перехода маркера на новую строку.

Ребенок может запоминать фразы, сказанные взрослыми. Пролог это тоже умеет. Введите

```
ADDCL((ПОСАДИЛ ДЕД РЕПКУ))
```

Как реагирует Пролог? Почти никак — печатает приглашение. В этой фразе про репку он ничего не понял, как если бы мы сказали: «Кудра будланула бокра». Кто такая кудра, кто такой бокр — неизвестно. «Будланула» может означать «поймала», «укусила», «поцеловала», «увидела» и все, что угодно. Так что же мы узнали? Что существует какая-то кудра и какой-то бокр и между ними было отношение «будлануть». Можно задать вопрос: «Кто будланул бокра?». Ответом будет: кудра.

Точно так же поступает и Пролог. Мы дали ему команду ADDCL, которая расшифровывается как ADD CLause (добавить предложение). Пролог запомнил предложение в базе знаний, расположенной в оперативной памяти, но разговаривать мы его еще не научили. Новорожденному Прологу можно задать вопрос

```
?((ПОСАДИЛ X РЕПКУ))
```

который воспринимается как «верно ли, что существует некий X, который посадил репку?» Пролог роется в своей памяти (базе знаний), обнаруживает, что такой X есть — это дед. Получается ответ «Верно», но Пролог молчит, только печатает приглашение. Именно так работает команда «?».

Попробуйте ввести вопрос

```
?((ПОСАДИЛ X ЖУЧКУ))
```

Пролог не находит подходящего X, печатает вопросительный знак и приглашение. Вопросительный знак нужно понимать так: «Кто его знает? Может быть, кто-то и посадил Жучку, но мне об этом ничего не известно».

Чтобы научить Пролог отвечать на вопросы, введем такие строки:

```
_.ADDCL ((КАКИЕ (X!Y)) (? Y)
                        (PP X) FAIL)
_.ADDCL ((КАКИЕ (X!Y)))
```

Что они обозначают, мы разберем позже. А теперь можно задать вопрос почти как о кудре:

```
_. КАКИЕ (X (ПОСАДИЛ X РЕПКУ))
   ДЕД
_.
```

Запрос КАКИЕ означает: «Какие известны X, которые посадили репку?» На этот запрос получен ответ ДЕД. Можно также спросить: «Что посадил дед?»:

```
_.КАКИЕ (X (ПОСАДИЛ ДЕД X ))
РЕПКУ
_.
```

Если при вводе строки не поставлены закрывающие скобки, то печатается число, равное количеству незакрытых скобок, и точка, которая приглашает продолжить ввод неоконченной строки.

Например:

```
_.КАКИЕ (X (ПОСАДИЛ X ЖУЧКУ)
1
```

Выведенная единица означает, что одна скобка не закрыта.

Редактирование базы знаний

При вводе предложений с помощью команды ADDCL возможны ошибки, но ошибочные предложения тоже запоминаются в базе знаний, а это может привести в дальнейшем к путанице. Требуется какой-то способ, позволяющий исправлять ошибки.

Во-первых, можно посмотреть, какие ваши предложения запомнил Пролог. Для этого служит команда LIST ALL. Введите ее. На экране должно появиться одно предложение про репку и два предложения со словом КАКИЕ.

Если уже введено много предложений, то лучше использовать команду LIST ⟨имя отношения⟩. Имя отношения — это первое слово после команды ADDCL и скобок. У нас было два отношения: ПОСАДИЛ и КАКИЕ. Если ввести команду LIST КАКИЕ, то будут напечатаны только предложения с отношением КАКИЕ.

Теперь можно вычеркнуть из базы знаний ошибочное предложение. Делается это с помощью команды DELCL (DELete

CLause — вычеркнуть предложение).

После команды DELCL должно идти то же предложение, которое стояло после ADDCL при его вводе. Для примера введем не нужное нам предложение:

```
ADDCL ((НЕНУЖНОЕ ПРЕДЛОЖЕНИЕ))
```

С помощью команды LIST НЕНУЖНОЕ убедимся в том, что оно запомнилось, и удалим его командой

```
DELCL((НЕНУЖНОЕ ПРЕДЛОЖЕНИЕ))
```

Проверим, что оно вычеркнулось. Для этого снова наберем LIST НЕНУЖНОЕ. Если все сделано правильно, то в ответ будет напечатано только приглашение.

Другой способ удаления предложений из базы знаний — с помощью команды KILL ⟨имя отношения⟩. При этом будут уничтожены все предложения с указанным отношением.

Есть еще команда KILL ALL (уничтожить все), полностью стирающая все введенные предложения.

Самый простой способ удаления предложения — по его номеру в базе знаний. По команде LIST ⟨имя отношения⟩ предложения с указанным отношением выводятся на экран. Для вычеркивания вводится команда

```
DELCL <имя отношения> <номер>
```

Например (нумерация предложения начинается с нуля!),

```
DELCL ПОСАДИЛ Ø
```

Для записи базы знаний на диск служит команда SAVE ⟨имя файла⟩. Файл на диске будет иметь расширение P. Например, приказ SAVE ПРОБА создаст на диске файл ПРОБА.P (P — латинское).

Команда LOAD ПРОБА выполнит обратную операцию — загрузит базу знаний из файла ПРОБА.P в ОЗУ.

Итак, Пролог способен запоминать факты и выдавать ответы на вопросы, касающиеся этих фактов. Знания представляются в виде отношений, например: *Петр — отец Коли*. В естественном языке отношения выражаются сказуемыми. На латинском языке «сказуемое» — praedicatum, поэтому в математике отношения называют предикатами. Так, отношение *быть отцом* (именное сказуемое) — это двухместный предикат, он выражает отношение между двумя объектами. Если в отношении участвует один объект, то такое отношение называется свойством, например: *светиться* (лампочка светится) или *Петр — мужчина*.

В Прологе имя отношения должно стоять перед именами участвующих в отношении объектов. Так, фраза ПОСАДИЛ ДЕД РЕПКУ построена правильно с точки зрения Пролога. Наращивая базу знаний, можно добиться того, чтобы Пролог понимал естественный язык, но, так как мы работаем с необученным Прологом, имена объектов должны быть в именительном падеже, чтобы при сравнении они совпадали. Поэтому вместо РЕПКУ лучше писать РЕПКА.

Итак, ОТЕЦ ПЕТР КОЛЯ будем понимать как *Петр — отец Коли*, а фразу *Саша — друг Маши* на Прологе запишем в виде ДРУГ САША МАША. Чтобы не было путаницы, договоримся, что подлежащее всегда стоит на втором месте.

Определение новых отношений с помощью правил

Кроме простых фактов можно сообщать Прологу правила. Предположим, что отношение *родитель* и свойства *муж* (мужчина) и *жен* (женщина) известны Прологу в виде фактов, например

```
_. ADDCL((РОДИТЕЛЬ ОЛЬГА ВЕРА))
_. ADDCL((РОДИТЕЛЬ ПЕТР КОЛЯ))
_. ADDCL((ЖЕН ОЛЬГА))
_. ADDCL((ЖЕН ВЕРА))
_. ADDCL((МУЖ КОЛЯ))
_. ADDCL((МУЖ ПЕТР))
```

Для облегчения ввода длинных строк пользуйтесь клавишами со стрелками. Так, нажатие клавиши «стрелка вниз» приводит к появлению предыдущей строки. Нажимая затем клавишу «стрелка влево» (или BS), можно стереть конец строки и затем заменить его на новый. Если вам нужно только начало предыдущей строки, то лучше нажимать клавишу «стрелка вправо», тогда символы появляются по одному.

Из исходных отношений *родитель, муж, жен* можно вывести производные отношения: *мать, отец, сын, дочь*. Сообщается это Прологу следующим образом:

```
_. ADDCL((МАТЬ X Y)(РОДИТЕЛЬ X Y)
                   (ЖЕН X))
```

Это предложение следует понимать так: X является матерью Y, если X является родителем Y и X — женщина.

Вся фраза так и называется — предложение, внутри которого в скобках расположены отдельные предикаты. Каждый предикат описывает отношение, причем первый предикат предложения — (МАТЬ X Y) является заключением, а остальные — условием. Очевидно, условие может состоять из одного, двух и более предикатов. Союз «и», которым соединены предикаты в условии, в Прологе имеет такой же смысл, как и «и» в алгоритмическом языке, AND в Бейсике или Паскале. А именно,

предложение *A и B* (где A, B — предикаты) считают верным тогда и только тогда, когда оба предиката верны. Такое предложение, составленное из двух и более предикатов с помощью союза «*и*», называют конъюнкцией предикатов. В нашем примере условие «X является родителем Y и X — женщина» есть конъюнкция предикатов «X является родителем Y» и «X — женщина». В отличие от указанных языков программирования союз «и» подразумевается при перечислении предикатов в условии, но не пишется.

Таким образом, предложение имеет вид
((заключение) (условие 1) (условие 2)...(условие N))

Смысл предложения можно выразить еще и так:
((заключение) верно, если верно (условие 1), и (условие 2), и (условие N)).

Условия называют также подцелями, а целью является заключение.

Определим теперь отношение ОТЕЦ:

```
_. ADDCL((ОТЕЦ X Y)(РОДИТЕЛЬ X Y)
                    (МУЖ X))
```

А как определить отношение СЫН? X будет сыном Y, если Y родитель X и X мужчина. Следовательно,

```
_. ADDCL((СЫН X Y)(РОДИТЕЛЬ Y X)
                    (МУЖ X))
```

Аналогично определим ДОЧЬ:

```
_. ADDCL((ДОЧЬ X Y)(РОДИТЕЛЬ Y X)
                    (ЖЕН X))
```

Теперь можно задавать Прологу вопросы:

```
_. КАКИЕ (X  (ДОЧЬ X ОЛЬГА))
ВЕРА
_.
```

Вопрос был «Кто является дочерью Ольги?». Ответ: Вера.

А теперь спросим, кто отец Коли.

```
_. КАКИЕ (X (ОТЕЦ X КОЛЯ))
   ПЕТР
_.
```

Мы не сообщали Прологу факт ОТЕЦ ПЕТР КОЛЯ, но мы объяснили, что значит ОТЕЦ, если известно отношение *родитель* и пол. И теперь Пролог использует введенные правила для получения нужного вывода.

Зададим Прологу вопрос, на который он не знает ответа: «Есть ли сын у Ольги?»

```
_. КАКИЕ (X (СЫН X ОЛЬГА))
_.
```

Никакого ответа. Значит, в базе знаний не нашелся такой X.

Продолжение следует.

Литература

1. *Кларк К., Маккейб Ф.* Введение в логическое программирование на микроПрологе. М.: Радио и связь, 1987.

2. *Клоксин У., Меллиш К.* Программирование на языке Пролог. М.: Мир, 1987.

3. *Доорс Дж., Рейблейн А. Р., Вадера С.* Пролог — язык программирования будущего. М.: Финансы и статистика, 1990.

4. *Хоггер К.* Введение в логическое программирование. М.: Мир, 1988.

5. Логическое программирование / Под ред. В. Н. Агафонова. М.: Мир, 1988.

6. *Язык Пролог* в пятом поколении ЭВМ: Сб. статей / Сост. Н. И. Ильинский. М.: Мир, 1988.

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

8. *Каймин В. А., Григорьев С., Угринович Н., Щеголев А.* Элементы логики и начала языка Пролог // Информатика и образование. 1989. № 4.

9. *Каймин В. А., Григорьев С. Г.* Пролог в школьной информатике // Информатика и образование. 1990. № 2.

10. *Григорьев С.* Работа системы Пролог-Д // Информатика и образование. 1990. № 4.

11. *Григорьев С.* Программирование на Прологе-Д // Информатика и образование. 1990. № 5.

12. *Григорьев С.* Графические средства системы Пролог-Д // Информатика и образование. 1990. № 6.

13. *Григорьев С.* Обработка списков на Прологе-Д // Информатика и образование. 1991. № 1.

В. АРТАМОНОВ

КУВТ «Корвет»

Помимо команд, управляющих работой дисковой операционной системы, в состав резидентных утилит входят также команды управления вводом-выводом и фонового выполнения резидентных команд.

УПР+S — приостановить вывод на дисплей. После ее выполнения система ожидает ввода одной из следующих команд: УПР+Q, УПР+C, УПР+P, УПР+F. Любая другая команда игнорируется системой.

УПР+Q — возобновить приостановленный вывод на дисплей. Если команда дана после УПР+S, она возобновит вывод информации той же программы. Если же команде УПР+Q не предшествовала команда останова вывода, то она игнорируется.

УПР+P — переключение режима дублирования на принтере информации, выводимой на дисплей. Команда дает пользователю возможность получения твердой копии. Первый ввод этой команды включает параллельную печать выводимой на дисплей информации, повторный ввод отключает печать. УПР+P обычно используется совместно с транзитной утилитой TYPE, которая выводит на дисплей содержимое заданного текстового файла с ГМД.

Утилита переназначения консоли вывода обеспечивает копирование символов, выводимых на консоль вывода (дисплей), в заданный дисковый файл. Вызывается командой вида

```
КОМАНДА > ДИСКОВОД ИМЯ
```

где параметр ИМЯ — обязательно явное имя файла, параметр КОМАНДА — любая из транзитных или резидентных утилит, параметр ДИСКОВОД — имя дисковода (A: или B:).

Если параметр КОМАНДА отсутствует, то система просто определяет дисковый файл для последующего копирования символов с дисплея. При наличии параметра КОМАНДА выполняется указанная утилита, при этом все символы, выводимые на дисплей, выводятся также и в заданный дисковый файл.

При отсутствии параметра ДИСКОВОД система предполагает, что заданный файл находится в текущей подобласти ГМД текущего дисковода.

Пример.

```
D > FILE.DOC
```

вызовет выполнение резидентной утилиты D показа оглавления текущей области ГМД с одновременной записью копии текста в дисковый файл FILE.DOC.

Файл консоли вывода закрывается системой в случае выполнения «теплого старта» ОС, при обнаружении ошибки обмена с диском или при останове вывода на консоль (утилита УПР+S) с последующим вводом команды закрытия файлов консоли ввода-вывода (утилита УПР+F).

Утилита переназначения консоли ввода обеспечивает ввод данных не с клавиатуры, а из дискового файла, заданного параметрами ДИСКОВОД и ИМЯ. Утилита вызывается командой вида

```
КОМАНДА < ДИСКОВОД ИМЯ
```

параметры которой задаются так же, как и в предыдущем случае.

При наличии параметра КОМАНДА выполняется указанная команда, но ввод символов, необходимых для ее работы, будет осуществляться из заданного дискового файла. Рассмотрим пример

```
COPY < B:F.TXLT
```

где COPY — транзитная утилита копирования файлов, а F.TXT — файл, первая запись которого содержит выражение A:=B:*.* (переписать все файлы с ГМД дисковода B: на ГМД дисковода A:). В этом случае утилита COPY выполнит все необходимые операции по копированию файлов, не требуя от пользователя ввода дополнительной информации.

Ввод из файла, определенного в качестве консоли ввода, прекращается при достижении конца файла или обнаружении ошибки обмена с ГМД или если выполняющаяся программа пользователя выдаст запрос операционной системе о приостановке ее вывода на консоль с последующим вводом команды УПР+F.

Обращение с утилитой переназначения консоли ввода должно быть особенно аккуратным: после ее вызова «ручное» управление ПЭВМ на какое-то время станет невозможным!

После прекращения чтения файла возобновляется обычный ввод команд непосредственно с клавиатуры.

УПР+W — вывести на дисплей содержимое буфера консоли.

Прежде чем говорить об этой утилите, целесообразно рассмотреть понятие командной строки и ее редактирование.

Командная строка должна включать в себя команду, может включать признак и

Окончание. Начало см.: Информатика и образование. 1991. № 5.

параметры. Интерпретатор команд, входящий в состав базовой ДОС, передает утилите (программе) текст командной строки, исключив из него признак и команду. В общем случае состав и синтаксис параметров определяются соответствующей утилитой. Иначе говоря, все команды, которые уже были или еще будут рассмотрены, представляют собой частные случаи командной строки.

Общее описание синтаксиса командной строки достаточно длинно, поэтому поясним его на примере.

```
:HDIR A:*.BAS > B:FILE.TXT
```

Здесь:

:— признак, определяющий режим исполнения команды, в данном случае — условное выполнение команды (команда не выполняется, если предыдущая команда установила неуспешным код исполнения). Этот признак может отсутствовать либо иметь еще два значения:

; — командная строка рассматривается как комментарий;

— — отмена переназначения консоли (используется для программ, у которых символы 〉 и 〈 входят в синтаксис параметров). Признаки : и — можно использовать совместно; при этом сперва ставится двоеточие, затем тире.

HDIR — транзитная утилита показа оглавления ГМД.

A:*.BAS — параметр транзитной утилиты (вывести на экран список имен файлов, имеющих расширение .BAS и находящихся на дисководе A:, с указанием их длины).

На месте данной транзитной утилиты может стоять любая резидентная или транзитная команда с соответствующими параметрами.

〉 — указатель переназначения консоли вывода.

B:FILE.TXT — указание файла, в который происходит копирование символов, выводимых на консоль вывода. Указатель переназначения консоли может отсутствовать (как и связанный с ним файл) либо иметь значение 〈 — указатель переназначения консоли ввода.

Прием команды от оператора завершается при вводе символа возврата каретки (или УПР+M) или перевода строки (УПР+J).

Управляющие символы, вводимые вместе с текстом командной строки, сразу же обрабатываются интерпретатором команд.

До завершения ввода командной строки пользователь может исправить замеченные им ошибки в набранном тексте, используя следующие управляющие символы:

УПР+E — продолжить ввод командной строки с начала следующей строки экрана;

УПР+H — удалить последний введенный символ (клавиша DEL);

УПР+I — дополнить ввод пробелами до количества литер, кратного восьми;

УПР+R — повторный вывод отредактированной командной строки;

УПР+U — удалить набранную командную строку и переместить курсор в начало следующей строки экрана;

УПР+X — удалить текст командной строки и возвратить курсор в ее начало;

УПР+← — удалить последнее введенное слово с экрана.

Обработанная интерпретатором и выполненная командная строка сохраняется в буфере консоли, поэтому в ответ на подсказку системы можно продолжить работу с прежней командной строкой. Для этого и служит команда УПР+W: она выведет на экран текущее содержимое буфера консоли, т. е. последнюю командную строку. После этого вновь вводимые символы будут добавляться к этой командной строке.

УПР+W восстанавливает содержимое буфера консоли, только если вводится сразу за приглашением системы, в любом другом случае МикроДОС воспримет этот приказ как обычную литеру!

УПР+D — утилита фонового выполнения резидентных команд. Если ввести ее в то время, когда транзитная программа ожидает ввода литер, последняя будет приостановлена МикроДОСом. После этого можно задать любую из резидентных команд МикроДОС; она будет выполнена и управление снова вернется к прерванной транзитной программе.

Транзитные команды

Транзитные команды реализуются специальными процедурами (программами), оформленными в виде файлов типа .COM или .SPR, и предназначаются для выполнения типичных операций над файлами (удаление, копирование, переименование и т. п.), а также копирования системных дорожек ГМД и форматирования.

Если программа, заданная для выполнения, является транзитной, то интерпретатор команд производит ее загрузку в специальный раздел ОЗУ — область транзитных программ (ОТП). Если длина программы превышает размер ОТП, на экран выводится сообщение «Не хватает памяти», программа не выполняется и система переходит в ожидание ввода новой командной строки (такое может случиться с «самодельной» транзитной программой).

В книге 22 (базовые программные средства ПК8020) документации к «Корве-

ту» дается достаточно хорошее описание 10 транзитных команд, поставляющихся с КУВТом. Изучив их, попробуйте сделать себе рабочий диск. Для этого следует сначала отформатировать его командой FORMAT, затем скопировать на него операционную систему командой SYSGEN и необходимые утилиты (как резидентные, так и транзитные) командой COPY.

Знакомство с МикроДОС можно считать законченным.

О. ФЕСЕНКО
г. Каркаралинск

Несколько советов обладателям «Корвета»

Два учебных года эксплуатации КУВТ «Корвет» показали, что основные причины выхода компьютеров из строя следующие: повышение напряжения сети, воздействие статического электричества, замыкание некоторых контактов разъема на левой боковой стороне машины, выключение КУВТ общим выключателем на щите питания. Для их устранения мы, во-первых, переключили провода, присоединенные в щите питания к клеммам трансформаторов «42», на клеммы «36» (повышение напряжения сети стало менее опасным), а клеммы «0» вторичных обмоток трансформаторов соединили друг с другом и надежно заземлили (специальный блок заземления вышел из строя через два месяца эксплуатации, и мы его отключили). Теперь компьютеры не боятся даже с головы до ног одетых в синтетику учеников.

Во-вторых, заклеили изолентой разъемы на левой стенке.

Наконец, выключаем машины только их собственными выключателями.

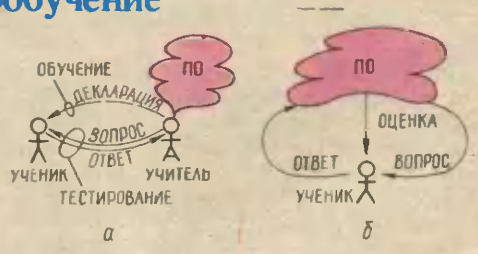
Пришлось столкнуться нам и с самопроизвольным выключением-включением компьютеров. Причиной этого оказалась вибрация двигателя вентилятора. Его нужно получше закрепить, предварительно подложив под него поролон или полиэтилен. Эта операция требует снятия крышки с клавиатурой.

Л. РАСТРИГИН

Компьютерное обучение и самообучение

Обучение и самообучение. Прежде чем говорить о компьютерных аспектах категорий обучения и самообучения, рассмотрим их в исходном (докомпьютерном, или, точнее, внекомпьютерном) понимании. Не будем цитировать классиков, а обопремся на здравый смысл, который и выведет нас на специфику компьютерного обучения и самообучения.

Рассмотрим обе категории в их сравнениях. На рис. 1 схематически изображено взаимодействие в обучении (рис. 1а) и самообучении (рис. 1б). В первом случае одним из непременных агентов процесса является учитель, т. е. лицо, реализующее процесс обучения. В схеме же самообучения его нет, и ученик взаимодействует только с окружающей средой, точнее, с некоторой предметной областью (ПО), т. е. с определенным видом рациональной деятельности.



В схеме обучения взаимодействие ученика и учителя четко регламентируется двумя перемежающимися этапами. На первом осуществляется передача обучающей информации, на втором — выяснение уровня ее усвоения (в режиме вопрос — ответ). При этом учитель, зная специфику предметной области, в которой предстоит действовать ученику, вербализирует ее в курс обучения, состоящий из порций обучающей информации, которые и передает (декларирует)

ученику на первом этапе обучения с учетом знаний ученика и специфики его восприятия, понимания, запоминания и т. д.

В процессе обучения основным является вопрос «Чему учить данного ученика в данный момент времени?». Для его разрешения необходимо ответить на другой: «Что такое данный ученик?». Именно поэтому учитель должен умело перемежать два указанных этапа обучения — декларативный (для передачи обучающей информации) и экзаменационный (для создания представлений об ученике, т. е. его модели).

Самообучение реализуется совсем по-другому. Среда (у нас — предметная область, ПО) ставит перед учеником задачи. Ученик как-то решает их, а среда оценивает правильность этого решения. Здесь оценка решения имеет вид поощрения — наказания, благодаря чему и осуществляется обучение (самообучение).

Для реализации режима самообучения необходимо уметь отвечать на вопрос «Какую задачу следует предложить данному ученику в данный момент?». Очевидно, что для ответа следует иметь представление об этом ученике, т. е. располагать его моделью. При этом модель синтезируется на базе той информации, которая имеется в ответах ученика.

Поэтому вопросы в режиме самообучения несут двойную нагрузку: с одной стороны, поставляют информацию о незнании ученика при его неправильных ответах (эта информация является исходной для синтеза модели ученика), с другой — должны обучать на подсказках. Как видно, незнание ученика является при самообучении той «движущей силой», которая позволяет идентифицировать ученика и обучать его наиболее эффективным образом, т. е. с учетом специфики его знаний, точнее, незнаний.

Отметим еще одну специфику самообучения: среда генерирует задачи по своим собственным законам. И это то, что нужно ученику в его дальнейшей (обученной) деятельности, т. е. при самообучении ученик всегда обучается тому, что требует от него среда (а не представлению учителя о том, что нужно ученику, как в схеме обучения). Это важное обстоятельство выгодно отличает самообучение от обучения.

Однако самообучение имеет и слабую сторону — оно не учит, как решать его задачи, т. е. ограничивается лишь оценкой действий ученика. Но оценка, особенно отрицательная, не указывает, как надо было решать задачу (единственно — не так, как это было сделано). Поэтому описанная схема самообучения применима в случае маловариантности (в идеальном случае — бинарности) решений, когда действие, противоположное наказанному, будет более правильным.

Естественно в компьютерной реализации самообучения устранить этот недостаток путем введения подсказки о правильном решении.

А теперь перейдем к компьютерным аспектам обучения и самообучения.

Компьютерное обучение. С появлением компьютера реализация систем обучения произошла по схеме обучения, а не самообучения, чему способствовали несколько факторов. Во-первых, это традиционная схема организованного обучения в человеческом обществе и ее, естественно, проще перенести на компьютер. Во-вторых, традиция внедрения компьютеров в человеческую деятельность всегда связана с перенесением на компьютер того, что достаточно хорошо отработано в «ручном» исполнении, а традиционный подход в обучении «отрабатывался» более двух тысячелетий. В-третьих, созданием и внедрением компьютерных систем обучения занялись преподаватели, которые, естественно, перенесли в обучающие программы свою методологию, т. е. моделировали с помощью компьютера свое поведение в процессе обучения.

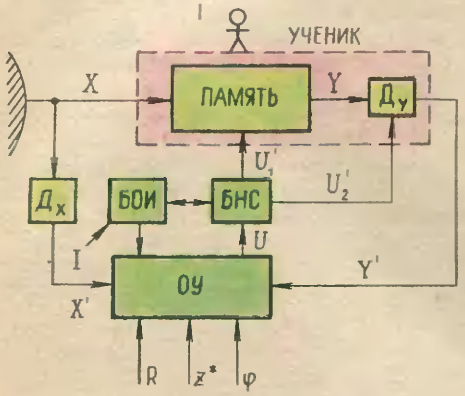
Процесс внедрения компьютеров в обучение шел, как видно, по схеме, аналогичной другим применениям компьютера. Однако он не дал такого эффекта, как в других отраслях. Опыт компьютеризации различных сторон человеческой деятельности показал, что внедрение компьютера дает многократное повышение эффективности (грубо — на порядок, исключая вычисления, где эффективность повышена на много порядков). Образование не получило столь необходимого ускорения. Даже массовое использование компьютера в учебном процессе не сократило время обучения хотя бы вдвое. Обычной является ситуация, когда эффективность внедрения компьютерного обучения надо доказывать с привлечением довольно тонких статистических критериев, т. е. КПД компьютера в учебном процессе попросту мал. И интенсивное его внедрение в обучение опирается пока на энтузиазм и безграничную компьютерную веру [1].

Можно, наверное, придумать много объяснений этому неприятному явлению. Ограничимся одним, методологическим. Известно, что всякое новое средство изменяет не только процесс достижения цели, но и технологию. В традиционном компьютерном обучении сохранена старая докомпьютерная технология, связанная с налаживанием эффективной коммуникации между источником ин-

формации (учителем-компьютером) и ее потребителем (учеником). Именно этому служил институт методик обучения, целью которых по сути дела была и остается «перекачка» знаний от учителя к ученику. С появлением компьютера этот подход не изменился, и преподавательские методики механически переносятся на процесс компьютерного обучения, сохраняя докомпьютерную технологию: подача материала — вопрос — ответ — снова подача и т. д. Это и является первопричиной малой эффективности использования компьютера в образовании.

Что же делать? — Изменить технологию, т. е. рассматривать процесс обучения не как «перекачку» знаний, а как процесс управления состоянием ученика.

Обучение как управление. В основе такого представления лежит постулат о том, что обучение можно рассматривать как целенаправленный процесс изменения состояния памяти ученика путем организации специальных информационных воздействий на него [2]. Управление как целенаправленное изменение состояния объекта — прекрасный инструмент для описания и организации процесса обучения. При таком подходе роль учителя выполняет управляющее устройство, а объектом управления является ученик (рис. 2). Приняв постулат об идентичности обучения и управления, можно воспользоваться для обучения мощным методологическим и формальным аппаратом теории управления, что позволяет по-иному сформулировать и решить задачу обучения.



Теперь схему обучения можно представить как схему управления и применить к ней методологию управления. На этой схеме (см. рис. 2) ученик представлен в виде «преобразователя» F^0 состояния среды X и обучающей информации U_1' в его состояние Y:

$$Y=F^0(X, U_1').$$

Управление $U=(U_1, U_2)$ объектом F^0 вырабатывает обучающее устройство (ОУ), которое генерирует U с помощью алгоритма обучения φ, заданных целей обучения Z^*, выделенного ресурса R и информации X′ и Y′ о состояниях соответственно среды X и объекта Y:

$$U=\varphi(Z^*, R, X', Y').$$

Информацию о состоянии среды, в которой находится ученик, доставляет датчик D_X:

$$X'=D_X(X),$$

а состояние ученика Y проявляется в его ответах Y′ на поставленные вопросы U_2':

$$Y'=D_Y(Y, U_2'),$$

где D_Y — генератор ответов ученика на заданные ОУ вопросы (т. е. преобразователь состояния У и вопроса U_2' в ответ Y′). Таким образом, ученик в этой схеме представлен в виде двойки

$$A=\langle F^0, D_Y\rangle, \qquad (1)$$

где оператор F^0 характеризует его индивидуальные способности по усвоению получаемой информации, а оператор D_Y определяет его возможности отвечать на поставленные вопросы. Большего в системе обучения как управления не требуется.

Реализуются команды обучающего устройства с помощью блока насыщения семантикой БНС, который трансформирует команды U в семантически содержательные порции обучающей информации U_1' и осмысленные вопросы U_2'. При этом используется банк обучающей (и вопрошающей) информации БОИ, содержащий в том числе и задачи. БНС является аналогом исполнительного механизма в системах управления техническим объектом, преобразующего команды управления в реальные физические воздействия на объект F^0.

В предложенной схеме отражаются все основные аспекты всякого обучения, которое определяется целью Z^* и обучающей информацией I, наполняющей банк БОИ, т. е. двойкой

$$B=\langle Z^*, I\rangle \qquad (2)$$

Таким образом, всякая система обучения характеризуется двумя известными факторами: A — кого учить (1) и B — чему учить (2). Начнем с детализации B. Здесь цель обучения Z^* (как цель управления) может быть представлена в виде [2] системы целей трех видов:

$$Z^*: \begin{cases} h_i(X, Y)\geqslant 0, & i=1, ..., k_1, \\ g_j(X, Y)=0, & j=1, ..., k_2, \\ q_l(X, Y)\to\text{extr}, & l=1, ..., k_3, \end{cases} \qquad (3)$$

каждая из которых интерпретируется естественным образом, например, так: h_i — «знание элемента i не должно быть хуже, чем...», g_j — «знание элемента j должно быть строго заданным...» и q_l — «знание элемента

l должно быть наилучшим».

Обучающая информация I в простейшем случае представляет собой, например, набор порций информации и тестов для проверки состояния ученика, его уровня обученности и т. д.

Алгоритм обучения φ (моделирующий методику обучения в стандартной АОС) в этом случае должен привести к быстрейшему достижению цели обучения (3), для чего (в соответствии с методологией управления) необходимо иметь модель объекта обучения

$$Y^m = F^m(X, U_l'), \qquad (4)$$

где F^m — модель ученика. Располагая адекватной моделью ($F^m = F^0$) и подставляя (4) в (3) при $Y = Y^m$, получим систему уравнений относительно того, что нужно учить ученику F^0.

Алгоритм обучения φ сводится, таким образом, к решению оптимизационной многокритериальной задачи вида

$$q_l(X, F^m(X, U_l)) \to \underset{U_l \in \Omega}{\text{extr}}, \quad l = 1, ..., k_3 \qquad (5)$$

в ограничениях

$$\Omega: \begin{cases} h_i(X, F^m(X, U_l)) \geqslant 0, & i = 1, ..., k_1, \\ g_j(X, F^m(X, U_l)) = 0, & j = 1, ..., k_2, \\ U_l \in R \end{cases}$$

где предполагаем $U_l = U_l'$, хотя $U_l' = \psi(U_l)$, где ψ — процесс внесения семантики в команду U_l.

Как видно, алгоритм обучения φ в этом случае является алгоритмом решения многокритериальной задачи (5). Это и есть основной результат применения методологии управления к обучению, позволяющий отказаться от придумывания специальных методик обучения, так как такой алгоритм обучения может быть взят из справочников по оптимизации (см., например, [3]).

Сказанное имеет место для случая, когда известна модель объекта обучения F^m. Очевидно, что такого никогда не бывает. Поэтому возникает типичная для управления задача синтеза адекватной модели F^m ученика F^0. В случае обучения эта задача усложняется тем, что ученик в процессе обучения изменяется, следовательно, во время обучения необходимо с помощью модели отслеживать изменяющиеся свойства ученика. Это означает, что процесс идентификации ученика должен вестись постоянно, т. е. его модель должна быть адаптивной [3]. Проблему можно решить, используя принципы адаптивного управления [3], смысл которых в минимизации разности реакций ученика и его модели путем соответствующего выбора параметров этой модели. Исходной информацией для этого служат ответы ученика на вопросы Y'.

Всякая система обучения работает в двух режимах: собственно обучение ($U = U_1$), когда обучающая информация I передается ученику, и тестирования ($U = U_2$), когда ученик отвечает на вопросы. Для задания режима работы нужно иметь возможность оценивать важность того и другого режима. Для этого вводится два критерия. Первый характеризует близость состояния ученика к заданным целям обучения Z^*:

$$Q_1 = \psi_1(Z^*, F^m),$$

где функционал ψ_1 характеризует степень достижения учеником F^0 (его модель — F^m) цели обучения Z^*. Второй характеризует адекватность модели F^m ученика F^0:

$$Q_2 = \psi_2(F^m, F^0),$$

где функционал ψ_2 определяет близость ответов ученика $Y' = D_Y(Y, U_2')$ и его модели $Y'^m = D_Y(Y^m, U_2)$ на одни и те же вопросы U_2'.

Задачу обучения на каждом (N-м) шаге теперь можно записать в виде решения двукритериальной задачи

$$Q_i \to \min_{U \in R_N}, \quad i = 1, 2,$$

где R_N — ресурс, выделенный на N-й шаг обучения. Естественно оба критерия свернуть линейно в один:

$$Q(U) = \alpha Q_1(U) + (1 - \alpha) Q_2(U), \qquad (6)$$

где $0 < \alpha < 1$ — коэффициент свертки, характеризующий степень относительной важности режимов. Теперь выбор режима определяется величиной приращения показателя (6) при $U = U_1$ или $U = U_2$, т. е.

$$U = \begin{cases} U_1 \text{ при } |\alpha Q_1(U_1)| \geqslant |(1-\alpha) Q_2(U_2)|, \\ U_2 \text{ при } |\alpha Q_1(U_1)| < |(1-\alpha) Q_2(U_2)|, \end{cases}$$

где $U_1, U_2 \in R_N$. Т. е. выбирается тот режим, который более интенсивно снижает значение показателя (6).

Такова формальная модель процесса обучения как управления. А теперь обратимся к самообучению.

Самообучение как управление. В этом случае схема, изображенная на рис. 2, интерпретируется иначе. В режиме самообучения ведущим является вопрос U_2, задаваемый ученику, а U_1 выступает в качестве подсказки, если ответ ученика был неверным. В процессе самообучения нет фазы синтеза модели, т. е. специально организованного тестирования ученика, как в процессе обучения. Здесь модель строится на основе информации об ошибках ученика (при правильных ответах модель, естественно, не корректируется).

БОИ в самообучении образуется из задачи U_2 и подсказок для их правильного решения U_1. В более общем случае БОИ образует (моделирует) среду, в которой предлагается действовать ученику. Правильность

этих действий оценивается ОУ; синтезируется модель ученика, которая позволяет определить близость его состояния к выполнению поставленных целей Z* (3) и смоделировать новую ситуацию, нужную для такого самообучения именно этому ученику, чтобы перевести его в требуемое целью состояние за минимальное время (или число шагов).

При самообучении особую роль играет БОИ, моделирующий ситуации, в которые может попасть ученик. Это не может быть просто набор нужных знаний, умений и т. д. Эти знания должны быть обязательно проранжированы по их важности для ученика в его будущей деятельности.

Пример системы самообучения (АСОЛИЯ). Описанные принципы компьютерного самообучения были реализованы в системе, названной АСОЛИЯ — Автоматизированная Система Обучения Лексике Иностранного Языка. Предметная область, которой обучает АСОЛИЯ, представляет собой лексику иностранного языка, знание которой необходимо при обучении любому языку. Более того, знание лексики в определенной степени компенсирует незнание его грамматики, так как известно [4], что смысл текста понятен, если известно 95 % его лексики. Указанные соображения и явились практическим стимулом создания такой системы самообучения.

АСОЛИЯ моделирует самообучение иностранному языку, создавая лексическую среду для ученика, т. е. обучает его так, как обучается языку ребенок — путем общения (в данном случае с компьютером). При этом задаются вопросы и на неправильные ответы дается подсказка, т. е. ученик поправляется так, как это бывает в реальной жизни.

Обучающей информацией I здесь является набор троек

$$I = \{\langle a_i, b_i, c_i\rangle,\ i=1, ..., n\}, \qquad (7)$$

где a_i — i-я иностранная лексическая единица (слово, грамматическая форма, словосочетание и т. д.), b_i — ее перевод на язык ученика, c_i — важность a_i для необходимого владения ученика языком в его предметной области.

Цель обучения Z* состоит в том, чтобы знание лексики учеником не было бы меньше заданного уровня. Пусть Q — уровень знания лексики учеником (например, доля известных ему лексических единиц в данной предметной области). Тогда цель в АСОЛИЯ:

$$Z^*: \begin{cases} Q \geqslant Q^* \\ T \to \min \end{cases}$$

где Q^* — заданный уровень знания лексики (обычно $Q^*=0{,}95$ [4], но для специальных областей $Q^*=0{,}85$, а для математики $Q^*=0{,}75$), T — время, затраченное на обучение.

В качестве Q удобно задать вероятность того, что ученик знает лексическую единицу (ЛЕ), наугад выбранную из текста предметной области ученика. Для оценки Q необходимо знать частотность i-й ЛЕ (a_i) в тексте (т. е. мерой важности c_i в (7) является частота ее встречаемости в текстах, которые предстоит понимать ученику) и вероятность того, что ученик знает a_i. Если величину c_i ($i=1, ..., n$) легко определить из частотного словаря обучаемой предметной области, то вероятность знания следует оценивать по ответам ученика и его способностям к запоминанию. Эту функцию выполняет модель F^m, которая позволяет определять изменение вероятности знания каждой ЛЕ во времени. Индивидуальные свойства забывания ученика отражаются в этой модели в результате ее адаптации.

Алгоритм обучения φ здесь представляет собой правило выбора для обучения такой ЛЕ, которая наибольшим образом увеличивает Q этого ученика.

Многочисленные эксперименты показали, что АСОЛИЯ обеспечивает наибольшую скорость обучения лексике иностранного языка: примерно одна минута на слово, что, например, в три раза быстрее известной суггестологической системы обучения Лозанова [5].

Литература

1. Обучающие машины, системы и комплексы: Справочник / Под общ. ред. проф. А. Я. Савельева.— Киев, Вища школа, 1986.— 303 с.

2. *Растригин Л. А.*, Эренштейн М. Х. Адаптивное обучение с моделью обучаемого.— Рига: Зинатне, 1988.— 160 с.

3. *Растригин Л. А.* Современные принципы управления сложными объектами.— М.: Советское радио, 1980 .— 230 с.

4. *Кондратьева В. А.* Оптимизация усвоения лексики иностранного языка.— М.: Высшая школа, 1974.— 120 с.

5. *Лозанов Т. К.* Суггестопедия при обучении иностранным языкам // Методы интенсивного обучения иностранным языкам.— М.: 1973. С. 7—16.

Р. САЛИМЖАНОВ

Множества в Паскале

Одной из интереснейших тем в преподавании языка программирования Паскаль является тема «Множества». Она тесно связана с математикой и математической логикой. Данная статья, обобщающая опыт автора по ее изучению с учащимися IX класса (углубленный курс математики), содержит систему задач, позволяющую последовательно раскрывать преимущества теоретико-множественных конструкций, сравнивать и оценивать новые возможности программирования с ранее имевшимися.

Кроме этого, использование почти во всех задачах генератора случайных чисел позволяет активизировать коллективную деятельность учащихся во время выяснения ими причин возникновения различных ответов при различных исполнениях одной и той же программы. При этом учитель получает благоприятную педагогическую ситуацию для разъяснения как тех свойств программы, которые не зависят от генератора случайных чисел, так и тех, которые зависят.

Важно помнить также, что, как бы ни было организовано преподавание той или иной темы, учитель должен раскрывать перед учащимися сущность наиболее важных приемов программирования последовательно, так, чтобы ученик мог сначала понять и получить навыки применения одного приема и лишь затем переходить к освоению других. В данной статье намечен путь реализации именно такого подхода. Для закрепления рассматриваемых приемов учитель легко может сам составить задачи, аналогичные содержащимся в тексте статьи, и использовать их в качестве домашнего задания или предлагать учащимся на самостоятельных, контрольных и зачетных работах.

Упражнения, примеры и задачи, содержащиеся в статье, после небольших преобразований можно реализовать на таких языках, как Рапира или Модула-2.

Множества. Операции над множествами

Как и в математике, одним из фундаментальных понятий в языке Паскаль является множество. Описание переменных типа множество имеет вид

```
var s1,s2,...,sn:set of T;
```

где s1, s2, ..., sn — имена множеств, элементами которых могут быть только величины типа T. Тип T называется базовым. Приведем примеры.

```
var Loto,Cifra:set of 0..49;
var Glas,Bukwa:set of char;
```

Здесь множества Loto и Cifra могут иметь в качестве своих элементов любые целые числа из промежутка от 0 до 49, а множества Glas и Bukwa содержат элементы базового типа char (символ) и не могут содержать объекты иных типов, например целые числа.

В Паскале для элементов множества допускается только простой тип, за исключением вещественного (а для целого типа можно задавать только диапазоны).

Отметим, что если множество в каком-либо месте программы задается в виде перечисления своих элементов, то эти элементы, в отличие от обычной математической нотации, должны быть заключены в квадратные скобки, так как во многих реализациях Паскаля в фигурные скобки заключаются комментарии.

Так, переменным Cifra, Data, Bukwa и Loto можно задать в теле программы при помощи оператора присваивания следующие значения:

```
Cifra:=[0,1,2,3,4,5,6,7,8,9];
Data:=[0,10..20];
Bukwa:=['a'..'d','f'..'h','k'];
Loto:=[23,41,3,45,6,14];
```

При этом переменная и множество, расположенные в левой и правой частях оператора присваивания, должны быть одного и того же типа.

Пустое множество обозначается так: [ ]. Это единственное множество, принадлежащее всем типам множеств. Например, для переменных Cifra и Glas верны операторы

```
Cifra:=[];
Glas:=[];
```

В то же время недопустимы операторы

```
Cifra:=Glas;
Glas:=Cifra;
```

так как переменные Cifra и Glas не одного и того же типа.

Отметим, что если множество задано в виде [e1...e2] и e2, согласно установленному порядку, предшествует e1, то это множество следует считать пустым. Например, [35..—1], ['m'..'b'] — пустые множества.

Имеется также ограничение на количество элементов множества. Оно определяется

в каждой конкретной реализации Паскаля и обычно колеблется между 64 и 256.

Как и в математике, в Паскале на всех множествах определены элементарные операции, приведенные в табл. 1. При их выполнении оба операнда должны принадлежать одному и тому же множественному типу значений.

Таблица 1

Операция	Название	Пример	Результат
*	пересечение	A*B	Множество, элементы которого принадлежат A и B одновременно
+	объединение	A+B	Множество, элементы которого принадлежат A или B или и тому, и другому
—	разность	A—B	Множество, элементы которого принадлежат A, но не принадлежат B

Кроме того, определены пять логических отношений, приложимых к множествам и возвращающие результат типа boolean (табл. 2).

Таблица 2

Отношение	Название	Пример	Результат
=	равно	A=B	TRUE, если множества A и B равны
<>	не равно	A<>B	TRUE, если множества A и B не равны
<=	включение	A<=B	TRUE, если множество A содержится в B
>=	включение	A>=B	TRUE, если множество B содержится в A
in	принадлежность множеству	x in S	TRUE, если элемент x содержится в множестве S

Отношения =, ⟨⟩, ⟨= и ⟩= требуют, чтобы оба операнда сравнения принадлежали к одному и тому же множественному типу. В отношении in первый операнд всегда должен принадлежать базовому типу, а второй — множественному типу значений, построенному на основе этого базового типа.

Чтобы продемонстрировать использование операторов и отношений над множествами, предположим, что переменным A, B и C типа set of shar присвоены следующие значения:

```
A:=['a','b','c','d'];
B:=['b'];
C:=['c','e'];
```

Тогда операции и отношения над множествами дадут результаты, приведенные в табл. 3.

Пересечение и объединение двух множеств часто называют соответственно умножением и сложением множеств. Соответ-

Таблица 3

Выражение или отношение	Значение
A+C	['a'..'e'], т. е. элементы из A либо из C
A—B	['a', 'c', 'd'], т. е. элементы из A, но не из B
A* B	['b'], т. е. элементы из A и B
B<=A	TRUE, так как B включено в A
A>=B	TRUE, так как A включает в себя B
C<=A	FALSE, так как элемент 'e' принадлежит C, но не принадлежит A
'a' in A	TRUE, так как элемент 'a' принадлежит A
'a' in C	FALSE, так как 'a' не принадлежит C

ствующим образом определены приоритеты операций: операция пересечения имеет приоритет перед операциями объединения и разности, а они, в свою очередь, имеют приоритет перед отношением принадлежности.

Примеры использования множеств

Задача 1. Посчитать количество букв, цифр, пробелов и знаков препинания в заданном предложении.

Предположим, что буквами в предложении будут только буквы русского алфавита, которые в реализации Turbo — Pascal для ПЭВМ «Ямаха» образуют отрезок символьного типа 'ю'..'Ю'.

В программе для улучшения ее понятности сходные по назначению строки выделены в группы, состоящие из трех-четырех строк, группы разделены пустыми строками, поэтому назначение отдельных разделов программы очевидно и без комментариев.

```
var KolBukw,KolProbel,KolCifr,
        KolZnakPrepin,ForIndex:integer;
    text:string[255];
    Bukwy,Cifry,ZnakPrepin:set of char;
    litera:char;
begin

  Bukwy:=['ю'..'Ч'];
  Cifry:=['0'..'9'];
  ZnakPrepin:=[';',',','!',',',',','.',':','?'];

  KolBukw:=0; KolProbel:=0; KolCifr:=0;
                        KolZnakPrepin:=0;

  readln(text);

  for ForIndex:=1 to length(text) do
    begin
      litera:=text[ForIndex];

      if litera in Bukwy
        then KolBukw:=KolBukw+1;

      if litera in Cifry
        then KolCifr:=KolCifr+1;

      if litera in ZnakPrepin
        then
          KolZnakPrepin:=KolZnakPrepin+1;

      if litera=' '
```

```
        then KolProbel:=KolProbel+1;

    end;

  writeln('Букв - ',KolBukw);
  writeln('Цифр - ',KolCifr);
  writeln('Знаков препинания - ',
                        KolZnakPrepin);
  writeln('Пробелов - ',KolProbel);
end.
```

Здесь можно было бы обойтись и без множеств. Например, условие

```
litera in Bukwy
```

можно заменить на

```
(litera>='ю') and (litera<='Ю')
```

а

```
litera in ZnakPrepin
```

на

```
(litera=';') or (litera='!') or
        (litera=',') or (litera='.') or
          (litera=':') or (litera='?')
```

Подобная замена, однако, не только увеличивает объем программы, но и снижает ее понятность.

Рассмотрим еще один небольшой пример, подтверждающий преимущество применения множеств в логических операциях для улучшения наглядности программ.

Предположим, что выполнение диалоговой программы останавливается для получения информации с клавиатуры и не должно возобновляться до тех пор, пока пользователь не нажмет одну из двух клавиш: с символами 'у', 'Y', 'ы', 'Ы' или символами 'н', 'Н', 'n', 'N'. При этом не имеет значения, в каком регистре работает клавиатура (русском или латинском, заглавном или прописном).

Задачу можно решить при помощи следующего оператора цикла

```
repeat
  read(kbd,klaw)
until klaw in ['у','Y','ы','Ы','n',
                        'N','н','Н'];
```

Печать элементов множества

В качестве продолжения темы задачи 1 возникает вопрос о том, как распечатать элементы, входящие в некоторое множество.

Задача 2. Задать множество, элементы которого — случайные числа из некоторого диапазона целых чисел; напечатать все элементы этого множества и подсчитать их количество.

```
var ForParametr,Element,Kol:integer;
    SetGruppa:set of 1..49;

begin
  SetGruppa:=[];randomize;
  {*----------------------------*}
  {* ЗАДАНИЕ ИСХОДНОГО МНОЖЕСТВА *}
  {*----------------------------*}
  for ForParametr:=1 to 6 do
    begin
      Element:=random(49)+1;
      SetGruppa:=SetGruppa+[Element]
    end;
  {*----------------------------*}
  {* ПЕЧАТЬ И ПОДСЧЕТ КОЛИЧЕСТВА *}
  {*    ЭЛЕМЕНТОВ В МНОЖЕСТВЕ    *}
  {*----------------------------*}
  Kol:=0;
  for ForParametr:=1 to 49 do
    if ForParametr in SetGruppa
      then begin
             write(ForParametr,' ');
             Kol:=Kol+1;
           end;
    writeln; write('Всего чисел - ',
                                Kol);
end.
```

После исполнения программы на компьютерах несколькими учащимися значения величины Kol у них получаются различными. Это объясняется тем, что, хотя цикл for и исполняется при значениях параметра ForParametr от 1 до 6, это не означает, что будет сформировано множество, состоящее точно из шести чисел; при очередном значении величины ForParametr может быть сделана попытка добавления во множество SetGruppa числа, которое там уже содержится. Если такая ситуация встретится, то мы получим множество, состоящее менее чем из шести элементов.

В Паскале не предусмотрены стандартные процедуры печати элементов множества. Для реализации раздела ПЕЧАТЬ и ПОДСЧЕТ ЭЛЕМЕНТОВ ВО МНОЖЕСТВЕ можно использовать следующий прием, часто применяемый в практике программирования. Так как заранее известен диапазон изменения элементов множества SetGruppa (от 1 до 49), то можно, перебирая все целые числа от 1 до 49, печатать только те, которые принадлежат множеству. После печати очередного элемента оператором Kol:=Kol+1 осуществляется подсчет номера печатаемого числа. Последнее исполнение оператора Kol:=Kol+1 позволит определить количество чисел, содержащихся во множестве

Генерирование множеств с заданным числом элементов

Задача 3. При помощи генератора случайных чисел сформировать множество с заданным числом элементов.

```
const Cardinal=6;
      n=1; k=49;

var Element,ForParametr,nomer:integer;
    SetLoto:set of n..k;

begin
```

Нетрудно заметить, что, решая задачу 3, мы фактически решили задачу о заполнении карточки спортлото 6 из 49.

Отметим, что применение множества SetLoto вместо некоторого массива из шести элементов позволило нам избежать невероятных прогнозов типа 3, 25, 41, 77, 77, 77. Кроме этого, можно не различать прогнозы вида 1, 2, 3, 4, 5, 6 и 6, 5, 4, 3, 2, 1, которые совпадают, хотя и выглядят поразному.

Наибольший и наименьший элементы множества

По аналогии с задачами на нахождение наибольшего и наименьшего элементов массива можно сформулировать и решить следующую задачу.

Задача 5. Найти наименьший (наибольший) элемент во множестве.


```
randomize; SetLoto:=[];
nomer:=0;
while nomer<Cardinal do
  begin
  (*-------------------------------*)
  (* ФОРМИРОВАНИЕ МНОЖЕСТВА SetLoto *)
  (*-------------------------------*)
    Element:=random(k-n+1)+n;
    if not(Element in SetLoto)
      then begin
           SetLoto:=SetLoto+[Element];
           nomer:=nomer+1
            end
 end;
 (*---------------------------------*)
 (*        ПЕЧАТЬ РЕЗУЛЬТАТОВ       *)
 (*---------------------------------*)
  for nomer:=k downto n do
    if nomer in SetLoto
      then write(nomer,' ');
end.
```

```
var ForParametr,Element,Min: integer;
    SetMechok:set of 1..200;
                        (* заменить Min *)
                        (*     на Max   *)
begin

  SetMechok:=[];randomize;
  for ForParametr:=1 to 100 do
     begin
        Element:=random(200)+1;
        SetMechok:=SetMechok+[Element];
     end;

  Min:=0;           (* Max:=201;  *)
  repeat
     Min:=Min+1;   (* Max:=Max-1 *)
  until Min in SetMechok;
  writeln(Min)
end.
```

Отметим, что переменная Min сначала получает значение 0, которое наверняка не принадлежит множеству SetMechok. Затем эта переменная при помощи цикла repeat «наращивается» до минимального элемента множества.

Если две последние строки предыдущей программы, в которых содержится комментарий, заменить на операторы в этих комментариях и выполнить указание, содержащееся в первом комментарии, то легко преобразовать нашу программу в другую программу — для вычисления наибольшего элемента множества.

В. КОВТУН
Белгород

Базы данных для КУВТ УКНЦ

Полезность использования баз данных в компьютеризированном учебном процессе очевидна. В белгородском педучилище № 2 используются на КУВТ УКНЦ сетевая учебная база данных PRSENT и информационно-поисковая система AIST. СУБД PRSENT («Подарок») обеспечивает формирование, редактирование и поиск данных, представленных в табличной форме. В состав пакета, размещенного на одном диске, входят операционная система и программы, поддерживающие работу БД: CREBD (открытие базы данных), DOBTAB (создание новой таблицы и добавление ее в БД), DELTAB (удаление таблицы из БД), ISPPW (изменение имени таблицы и пароля доступа) и REDTAB (редактирование БД — заполнение таблицы, редактирование, поиск данных таблицы в режиме «меню»). Первые четыре программы используются на РМП, пятая — на РМУ.

СУБД позволяет создавать до 10 таблиц, в каждой — до 40 строк. В меню базы данных есть следующие пункты:

0 — выход из редактора;
1 — установка столбцов;
2 — просмотр таблицы;
3 — добавление строк;
4 — добавление столбцов;
5 — очистка столбца;

6 — исправление имени столбца;
7 — поиск.

При выполнении практических работ с этой базой данных у учащихся вырабатываются навыки работы с меню, формирования, редактирования, поиска необходимой информации с помощью ЭВМ.

При подготовке к занятиям необходимо определить тему табличной БД, спланировать экранное поле (столбцы, их названия, длина записи), на РМП открыть базу данных, заполнить наименования столбцов (не заполняя первой строки). Остальную работу по формированию, редактированию и поиску информации оставить на практическое занятие. При этом для каждой практической работы необходимо подготовить материал с описанием БД.

Одним из примеров работы на отделении физической культуры училища может служить БД, позволяющая вести протокол спортивного соревнования (в данном случае по прыжкам). Первоначально на РМП сформирована следующая табличная форма протокола:

	Фамилия	Попытки:	1-я	2-я	3-я
1.	Азаров А.	м.			
2.	Смирнов И.	м.			
.	.....				

Затем на РМУ вводятся данные о попытках, добавляется колонка «Зач. рез.» (зачетный результат), переименовываются колонки, переносятся данные, и в итоге на мониторе формируется окончательная форма протокола из трех колонок:

Фамилия	Результат м.	Место

В этой работе используются почти все действия из меню.

При опробовании пакета СУБД PRSENT (возможно, адаптированного для УКНЦ с другой системы) выявились следующие ограничения: максимальное количество символов в строке таблицы — 45, максимальная длина имени таблицы — 15 символов. В результате не используются возможности экранного поля монитора УКНЦ. После некоторых изменений в программах пакета число символов в строке увеличено до 60, длина названия таблиц — до 30 символов, что позволило резко расширить возможности данной БД. В программе DELTAB необходимо в два раза уменьшить число байтов области резервирования памяти (с учетом объема ОЗУ УКНЦ, иначе программа не запускается), а также переименовать расширение указанных пяти программ на стандартное ASC.

Необходимо отметить, что загрузка БД из пяти столбцов (один из них заполнен в среднем 12 символами) занимает время порядка 1,5 мин на одно РМУ, загрузка полностью сформированной БД — от 2 до 4 мин. Это следует учитывать при подготовке машин к занятиям.

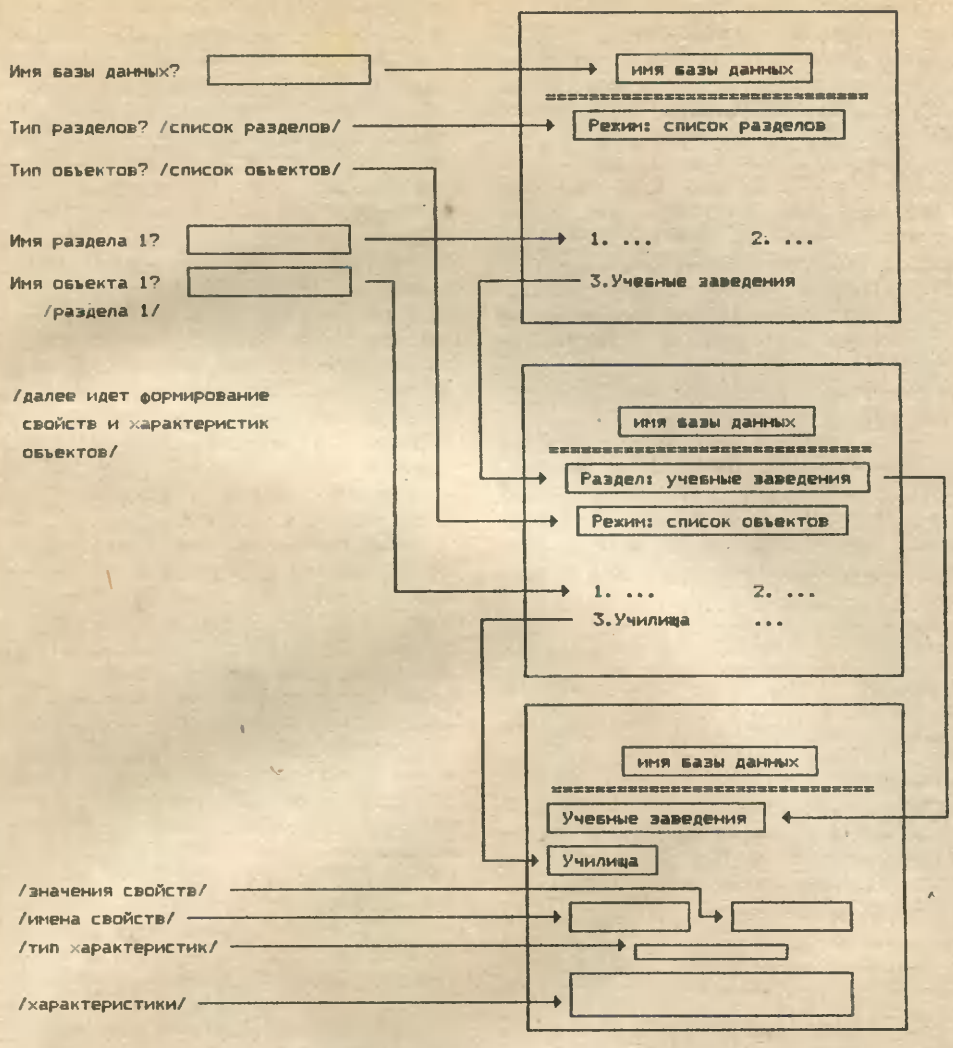
Информационно-поисковая система AIST предназначена для создания информационных структур, состоит из двух пакетов прикладных программ в виде загрузочных модулей, функционирование которых обеспечивается стандартной ОС ФОДОС. СУБД позволяет выполнять открытые базы данных, добавление раздела БД, редактирование содержимого базы, создание и редактирование информационных кадров. Создание и редактирование БД производится на РМП, работа в режиме поиска необходимой информации — на РМУ. База данных СУБД AIST может содержать до 16 разделов, каждый из которых, в свою очередь, до 20 объектов, имеющих до четырех свойств и до 12 характеристик. На рис. 1 хорошо видно, что информационно-поисковая система имеет древовидную структуру. Процесс формирования базы данных изображен на рис. 2.

В качестве примера рассмотрим поиск информации в каталоге «Физкультура и спорт» библиотеки училища.

1

2

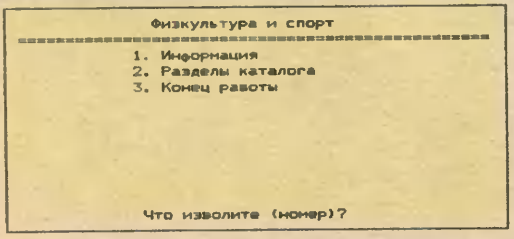
Последовательность формирования базы данных



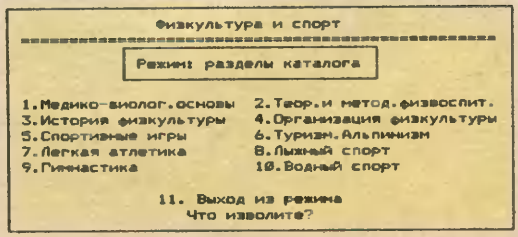
52

Экран монитора после загрузки БД изображен на рис. 3. На нем отражено имя базы данных — «Физкультура и спорт» — и режимы работы с БД: информация (по запросу выдается общая информация о библиотеке, ее фонде и т. п.), разделы каталога (основной режим работы с БД), конец работы. После ввода цифры 2 в качестве ответа на запрос ЭВМ на экране монитора появляется информация первого уровня — разделы каталога (рис. 4). В данном примере 10 разделов каталога. Выбе-

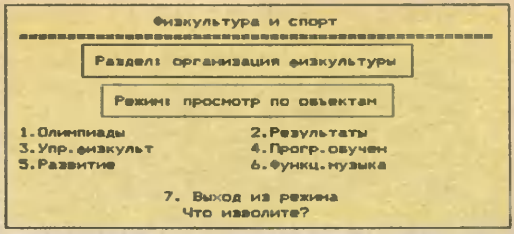
3



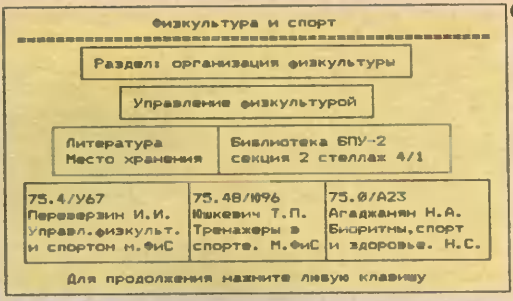
4



5



6



рем четвертый и перейдем на второй уровень поиска, где отражены объекты соответствующего раздела (рис. 5). Для получения информации по интересующему объекту необходимо нажать соответствующую клавишу; нажмем 3 — «Упр. физкультурой». На экране монитора появится информация третьего уровня поиска (рис. 6).

Отметим, что информация первых двух уровней каждой БД после загрузки хранится в ОЗУ ЭВМ, а информация третьего уровня по мере необходимости считывается с внешней памяти. Это обеспечивает оперативность поиска информации.

Объем базы данных, создаваемой с помощью AIST, до 170К байт. AIST предоставляет возможность одновременной работы с базой данных нескольким пользователям в локальной информационной сети.

Е. ЕРЕМИН

Вирусы «Ямахи» MSX-2

В последнее время появилось много публикаций о компьютерных вирусах. Без преувеличения можно сказать, что для пользователей ПЭВМ IBM PC вирусы стали серьезной проблемой. Может показаться, что для работников просвещения тема эта представляет в основном познавательный интерес. Но не следует забывать, что в нашей системе образования (особенно высшего) широко распространены КУВТ на базе ПЭВМ «Ямаха». Эта машина работает под управлением близкой к MS DOS операционной системы, и вирусы для нее уже существуют. Об этом пока нигде не написано*, не считая распространяющегося на дискетах текстового файла П. А. Земцова и Е. В. Налимова, а тем временем вирусы «Ямахи» постепенно «расползаются» по машинам ничего не подозревающих пользователей. Кроме саморазмножения компьютерные вирусы уже сегодня могут, например, делать следующее: поздравив с днем рождения девушку Наташу, нарушить процесс загрузки MSX DOS; организовать периодический «буквопад» на экране; заменяя символы, препятствовать нормальной работе принтера или дисплея; нарушать процесс регенерации памяти, что вызывает потерю информации в ОЗУ; стирать файлы, имя которых начинается с русской буквы, и т. д.

В статье изложены основные принципы устройства компьютерных вирусов и способы их распространения. Поскольку вирусы уже существуют, информация о них должна принести больше пользы, чем вреда, помогая уменьшить вероятность «заражения» компьютера. Описаны меры профилактики. Приводится текст программы-детектора для обнаружения на дисках всех известных автору разновидностей вирусов, а также тех, которые еще будут написаны с применением аналогичной структуры.

Что же такое компьютерный вирус? Так принято называть программу, способную размножаться, переписывая свои копии на текущий диск или на другие диски (в файлы или просто в отдельные сектора), а также, помимо воли пользователя, выполнять различные нежелательные действия на компьютере. Наибольшую опасность представляет не само размножение, а сопутствую-

* Данная публикация готовилась до выхода статьи В. Канивца «Активное средство защиты авторских прав».— *Примеч. ред.*

щие вирусу эффекты, часто приводящие к потере важной информации или просто мешающие нормально работать на компьютере. Многие вирусы даже после завершения работы «зараженной» программы остаются в памяти ЭВМ (так называемые резидентные вирусы). Их можно нейтрализовать только полной перезагрузкой компьютера.

Вся деятельность вирусов обычно тщательно маскируется, поэтому пользователь, работающий на «зараженном» компьютере, часто не замечает ничего подозрительного. Так, например, запись своей копии на диск вирус может производить в тот момент, когда вы пытаетесь вывести на экран директорий диска. Если диск защищен от записи, он, естественно, не пострадает, но никаких сообщений о неудачной попытке записи может не появиться.

Чтобы успешно противостоять натиску компьютерных вирусов, необходимо хорошо знать возможные механизмы их размножения. Прежде всего отметим, что существуют boot-вирусы (boot — загрузочный сектор) и файловые вирусы. Названия отражают место размещения этих двух основных категорий вирусов на диске. Заражение такими вирусами происходит по-разному. Для изучения этого вопроса проследим за работой учительской ПЭВМ «Ямаха» с момента включения питания (наиболее нетерпеливый читатель может пропустить технические подробности и сразу перейти к знакомству с практическими выводами).

Итак, вы включили компьютер. Процессор автоматически начинает функционировать по специальной программе инициализации, записанной в ПЗУ. При этом «в недрах» компьютера протекают следующие важные процессы:

1) определяется объем памяти и ее конфигурация (расположение в адресном пространстве), результаты тестирования заносятся в определенные ячейки ОЗУ;

2) из ПЗУ в рабочую область ОЗУ переписываются необходимые для нормальной работы ЭВМ значения;

3) в специальной области памяти формируется содержимое ловушек (их действие подробно описано в документации по «Ямахе» и здесь не обсуждается);

4) инициализируются устройства ввода-вывода;

5) производится процедура сканирования слота (подпрограмма BIOS со входным адресом 162 — здесь и далее все адреса и коды шестнадцатиричные);

6) управление передается Бейсику.

Рассмотрим теперь подробнее содержание п. 5. При сканировании слота компьютер поочередно подключает с адреса 4000 все возможные страницы памяти и проверяет их первые два байта, сравнивая с кодами 41 и 42. Если фиксируется именно такая последовательность значений, происходит переход по адресу, записанному в байтах 4002 и 4003. Таким образом, каждая «помеченная» страница памяти получает управление и производит те или иные действия. Описанным способом, например, происходит инициализация локальной сети, а также получает управление ПЗУ с дисковым Бейсиком на учительской машине. В последнем случае специальная программа инициализирует дисковую систему и проверяет, вставлен ли диск в дисковод А. При наличии диска производится попытка загрузить с него операционную систему. Загрузка интересующей нас сейчас системы MSX DOS протекает в несколько этапов.

5.1) считывается нулевой сектор (boot), содержащий начальный загрузчик ОС, и управление передается на него;

5.2) загружается файл MSXDOS.SYS, содержащий резидентную часть ОС (обмен с внешними устройствами, дисковые операции, программа перезагрузки и т. п.); при отсутствии файла с системой процесс прерывается и запускается Бейсик (см. п. 6);

5.3) управление передается находящемуся в памяти файлу MSXDOS.SYS, который подготавливает MSX DOS к работе и запускает командный процессор COMMAND.COM. Последний обеспечивает прием команды пользователя и ее исполнение.

Отметим, что если в ОЗУ в адресах 4000 и 4001 записаны указанные коды, управление передается в ОЗУ. Этим способом реализуется перезапуск многих игровых программ, например фирмы «Konami».

Чтобы не утомлять читателя излишними подробностями, в описании опущены некоторые второстепенные детали сканирования, а также вариант загрузки при нажатых служебных клавишах DEL, CTRL и др.

Сделаем некоторые полезные практические выводы из этого подробного описания:

1. До процедуры сканирования слота процессор следует программе, находящейся в ПЗУ, и поэтому «заражение» на этих этапах невозможно.

2. Если не вставлен диск, при включении питания загружается «стерильный» Бейсик. Если же вставлен диск без MSXDOS.SYS, дополнительным условием «стерильности» является отсутствие вируса в boot-секторе

(практический вывод очевиден: при загрузке Бейсика вынимайте диск из дисковода A).

3. «Чистота» ОС MSX DOS зависит от состояния трех компонентов: boot-сектора, файлов MSXDOS.SYS и COMMAND.COM. Рекомендуем использовать для загрузки ОС только дискеты, которым вы доверяете, причем лучше, если эталонная системная дискета будет защищена от записи. Не следует гоняться за «последними версиями» ОС — надежнее использовать фирменную поставку.

4. При копировании дисков «сектор в сектор» возможно заражение диска boot-вирусом, поэтому не злоупотребляйте им.

5. Все ловушки при инициализации обновляются и изменивший их резидентный вирус при перезагрузке «погибает». Исключение составляет случай, когда в ячейках памяти 4000 и 4001 содержатся коды 41 и 42, что теоретически позволяет «выжить» вирусу в памяти.

Итак, если вы загрузились с «чистого» диска, вирус вам пока не угрожает. Но в процессе работы вы запускаете различные файлы в машинных командах, имеющие расширение COM. В этом случае могут активизироваться находящиеся в зараженных файлах файловые вирусы. Вирус может внедряться в файл разными способами. На рис. 1 представлен случай, когда вирус «прикрепился» к концу файла. Чтобы получить управление сразу при запуске, несколько первых байтов программы он сохранил в своем «теле», заменив их на команду перехода. После передачи управления (стрелка 1 на рис. 1) вирус выполняет различные необходимые ему действия, восстанавливает начальные байты программы (стрелка 2) и, не обязательно сразу, передает управление на начало восстановленной программы (стрелка 3).

Несколько иначе выглядит вирус, разместившийся перед файлом (рис. 2). Управление он, естественно, получает первым и проделывает все необходимые ему действия. Но исходный файл, будучи сдвинутым со своих первоначальных адресов, не сможет работать нормально. Поэтому вирусу приходится сдвигать его на место. При этом, чтобы не стереть программу перезаписи, вирус вынужден переписывать ее на свободное место (стрелка 1) и затем передавать управление на нее (стрелка 2; перезапись программы помечена стрелкой 3). Удобно располагать программу перезаписи непосредственно перед восстанавливаемым файлом так, чтобы не требовалось дополнительно организовывать переход на его начало. Отметим, что вирус на рис. 2а переписывает всю программу целиком, а вирус на рис. 2б — только ее начало, остальная часть остается на месте. Нетрудно видеть, что отличие это несущественно и влияет лишь на длину переписываемого массива.

Теперь несколько практических советов по профилактике файловых вирусов (помимо тривиального, но не всегда выполняемого: не копировать все подряд, особенно у малознакомых лиц):

используйте защиту дисков от записи — это простой, но очень надежный способ избежать неожиданных неприятностей;

храните резервные копии всех наиболее ценных программ;

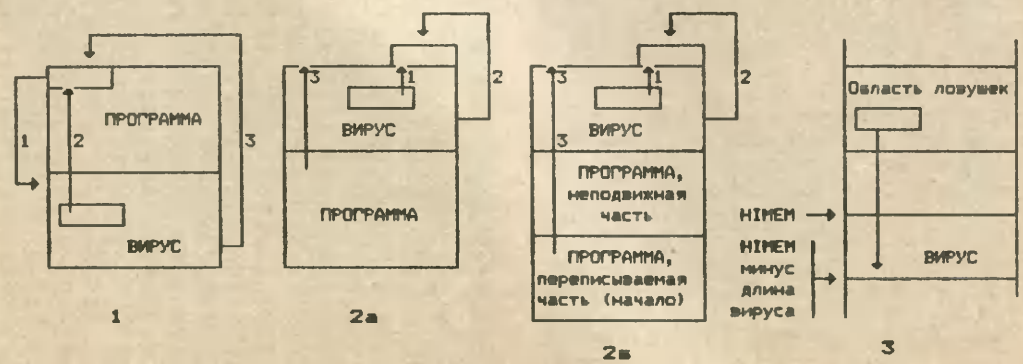
имейте полную распечатку каталога каждого диска, следите за длиной файлов и датами их создания: при «заражении» они почти наверняка изменятся;

после работы с «подозрительными» программами не поленитесь перезагрузить компьютер: это уничтожит возможный резидентный вирус в ОЗУ;

не пренебрегайте специальными антивирусными проверками на своем компьютере, особенно при работе с новым программным обеспечением.

Как видите, профилактические меры очень просты и естественны.

Завершая обсуждение «теоретических ос-



нов вирусологии», опишем устройство резидентных вирусов. Все boot-вирусы должны быть резидентными (иначе им не попасть на другой диск); возможны и резидентные файловые вирусы. В обоих случаях наиболее простой путь получать управление при дальнейшей работе компьютера — перехватывать обращения к ловушкам. Особенно часто используются ловушки дисковых операций, расположенные в области F24F — F2B7, но иногда вирусы меняют и основные ловушки (FD9A—FFD8).

Из сказанного сразу вытекает средство борьбы с резидентными вирусами в памяти ЭВМ. Командой BSAVE запишите в файл-образец содержимое областей ловушек после команд __NETEND и __NETINIT, так как работа с сетью переопределяет некоторые ловушки. Затем не пожалейте времени на несложную Бейсик-программу, которая сравнивает содержимое файла-образца с текущим состоянием ловушек (загружайте файл в свободную область памяти, например, с адреса B000, используя параметр смещения в команде BLOAD — см. с. 315 в [1]). • Такая программа предупредит вас о любом изменении ловушек. Следует, правда, учитывать, что не только вирусы меняют состояние ловушек: некоторые полезные программы также активно используют этот прием.

Резидентный вирус, помимо перехвата управления, должен решить и еще одну важную для своей безопасности задачу — разместить код своей программы в какой-нибудь области памяти, где он не будет стерт при дальнейшей работе компьютера. Все известные файловые вирусы переписывают себя в редко используемые участки служебной области памяти. Boot-вирусы поступают более «квалифицированно». Из величины HIMEM, определяющей верхнюю границу памяти, вирус вычитает свою длину и тем самым отбирает у ОС место для размещения (рис. 3). Таким образом, контроль за HIMEM также является одним из методов своевременного обнаружения в ОЗУ резидентного вируса. Величина HIMEM хранится в двух байтах с адреса FC4A, причем, как обычно, первым расположен младший байт.

Если вы обнаружили одновременное изменение ловушек и HIMEM, это наверняка говорит о вмешательстве в работу ОС некоторой резидентной программы. Правда, не все резидентные программы вредны: например, широко распространенный RAMDISK.COM тоже производит указанные изменения в памяти, не являясь вирусом.

Учитывая данные проведенного анализа, напишем программу-детектор для нахождения вирусов на диске. Она будет выполнять следующие:

проверять состояние дисковых ловушек, так как резидентный вирус меняет их для перехвата дисковых операций;

проверять значение ячейки HIMEM, сравнивая ее содержимое со стандартным;

сравнивать адреса 4000 и 4001 в ОЗУ с кодами 41 и 42, удалять их для предотвращения каких-либо нестандартных действий после нажатия на кнопку «Reset».

проверять загрузочный сектор на совпадение с эталоном, который хранится в ПЗУ;

проверять все файлы с расширением SYS и COM на отсутствие приведенной на рис. 1 и 2 структуры.

Простейший вариант программы VD, реализующей все описанные действия, приводится ниже. Отметим, что это максимально короткий вариант, специально рассчитанный на публикацию в журнале. Он ищет вирусы не по образцам, а по структуре программы, отслеживая связи, обозначенные на рис. 1 и 2 стрелками. Детектор гарантирует обнаружение всех известных автору вирусов и, возможно, аналогичных им новых. Конечно, не следует его идеализировать, так как указанные связи реализуются не единственным способом. Поэтому, хотя VD анализирует несколько возможных вариантов, его можно обмануть.

Распечатка программы VD

```
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
100 21 00 04 3E FF 47 ED B1 BE 20 FB 23 44 4D 05 21
110 00 01 AF 00 00 BE 08 23 0B 78 B1 20 F7 08 BE C0
120 21 9F 02 11 BD 00 01 42 00 ED B0 CD EE 00 56 69
130 72 75 73 20 64 65 74 65 63 74 6F 72 0D 0A 28 43
140 29 20 45 53 43 20 20 56 20 31 2E 31 0D 0A 0D 0A
150 4B 6F 6F 6B 73 24 06 69 3E C9 21 4E F2 23 BE 20
160 07 10 FA CD E4 00 18 05 3E 31 CD D2 00 CD EE 00
170 4B 69 4D 65 6D 20 24 11 93 DF 2A 4A FC 7C BA 20
180 04 7D BB 28 09 CD EE 00 3F 0D 0A 24 18 03 CD E4
190 00 CD EE 00 34 31 2F 34 32 20 24 F3 DB FD F5 3E
1A0 02 D3 FD 1E 2D 2A 00 40 7D FE 41 20 0A 7C FE 42
1B0 20 05 1E 2B 32 00 40 F1 D3 FD 0E 02 CD 05 00 CD
1C0 E4 00 21 00 00 E5 3A 5C 00 77 23 01 3F 0B CD BD
1D0 00 01 00 20 CD BD 00 D1 0E 11 CD 05 00 CD EE 00
1E0 42 6F 6F 74 20 24 3E B7 26 40 CD 24 00 21 05 74
1F0 11 00 06 D5 01 D0 00 ED B0 3E BB 26 40 CD 24 00
200 11 00 00 D5 0E 1A CD 05 00 3A 5C 00 3D FE FF 20
210 05 0E 19 CD 05 00 26 01 6F 11 00 00 0E 2F CD 05
220 00 21 03 00 11 90 00 D5 01 06 00 ED B0 3E 24 12
230 D1 CD FB 00 E1 D1 06 D0 1A BE 20 06 23 13 10 F8
240 18 05 3E 32 32 DF 00 01 30 01 21 D0 08 7E B7 20
250 12 23 0B 78 B1 20 F6 3A DF 00 FE 32 28 07 CD E4
260 00 18 05 3E 33 CD D2 00 21 5D 00 3E 20 BE 20 0E
270 01 3F 08 CD BD 00 11 59 53 72 23 73 23 72 2E 60
280 01 00 50 CD BD 00 11 00 01 0E 1A CD 05 00 21 E1
290 02 11 00 D0 7A 32 FF 00 01 A0 01 D5 ED B0 C9 71
2A0 23 10 FC C9 0B 78 B7 78 20 03 B9 38 01 4F 06 00
2B0 08 ED B1 C9 32 DF 00 CD EE 00 20 45 72 72 20 4E
2C0 20 30 0D 0A 24 C9 CD EE 00 20 4F 6B 0D 0A 24 C9
2D0 D1 62 6B 0E FF 3E 24 ED B1 E5 0E 09 C3 05 00 C3
2E0 A7 0E 11 11 5C 00 CD 05 00 3C 28 57 11 90 00 21
2F0 00 01 01 25 00 D5 ED B0 21 0C 01 36 24 5D 54 13
300 0E 04 ED B8 23 36 2E 54 5C CD FB 00 D1 D5 0E 0F
310 CD 05 00 21 B1 00 06 04 74 23 10 FC 2E 9F 74 20
320 74 34 2A A0 00 D1 18 43 D5 0E 27 CD 05 00 D1 0E
330 10 CD 05 00 CD FD 00 F5 CC D2 00 F1 C4 E4 00 0E
340 12 18 A0 21 65 00 3E 53 BE 20 0A CB A6 23 36 4F
350 23 36 4D 18 BC 3A DF 00 FE 33 30 0E CD EE 00 4E
360 6F 20 76 69 72 75 73 0D 0A 24 C9 ED 4B A2 00 78
370 B1 20 08 3A FF 00 3D 3D BC 30 AD CD EE 00 20 4C
380 6F 6E 67 0D 0A 24 18 B7 3A 00 01 B7 20 0A CD EE
390 00 20 30 24 3E 01 B7 C9 2A A0 00 FE C3 28 0A D6
3A0 40 FE 40 30 70 AF 3E 38 C9 24 ED 5B 01 01 B7 ED
```

```
3B0  52 EB 4B 42 3E 04 B8 30 03 47 0E 00 AF ED B1 C0
3C0  78 B1 3E 01 20 02 B7 C9 BE 20 F1 2B 2B 7E 57 23
3D0  23 FE C3 3E 34 CB 7A FE 21 20 10 3E E9 E5 C5 1E
3E0  10 CD C2 00 C1 E1 3E 35 C8 18 06 E6 EF FE 01 20
3F0  CB E5 C5 7A E6 30 F6 C5 1E 06 CD C2 00 20 04 78
400  B1 20 04 C1 E1 18 B5 3E C9 1E 10 CD C2 00 C1 E1
410  3E 36 C8 18 A7 4D 7C 06 04 B8 30 01 47 21 00 01
420  3E 21 ED B1 C0 78 B1 20 02 3C C9 7E FE ED 20 F0
430  23 7E 2B FE B0 20 E9 E5 C5 3E 22 1E 10 CD C2 00
440  28 04 C1 E1 18 DA 7B B1 28 F8 5E 23 56 2B 3E C3
450  D5 1E 20 CD C2 00 D1 20 E9 7E BB 20 E5 23 7E BA
460  20 E0 C1 E1 3E 37 C9 FF FF 01
```

Как сформировать файл VD.COM и как с ним работать? Для ввода удобно использовать какой-нибудь отладчик для программ в машинных кодах, лучше всего DEBUGGER Леонида Бараза. Возможен вариант ввода из Бейсика: в этом случае коды программы построчно задают в операторах DATA, а затем организуют их циклическое чтение оператором READ и занесение в файл последовательного доступа (см. гл. 5 в [1]), названный VD.COM. При наборе следует обратить особое внимание на правильность первой строки, где расположена процедура контрольного суммирования всей программы. Остальное будет автоматически проверено при запуске.

После формирования файла загрузите MSX DOS и наберите имя VD (пока без параметров). Если на экране ничего не появилось, кроме нового приглашения операционной системы, значит, вы где-то ошиблись в наборе. Не отчаивайтесь — проверьте коды еще раз (лучше вдвоем) и исправьте все ошибки. Когда программа «пойдет», на экране появится текст, подобный следующему:

```
A>VD
Virus detector
(C) ESC V 1.1

Hooks Ok
HiMem Ok
41/42 — Ok
Boot AX-350 Ok
MSXDOS SYS Ok
COMMAND .COM Ok
VD .COM Ok
No Virus
```

С вашим диском все в порядке. Конечно, контрольное суммирование — надежная проверка, но хочется увидеть, что детектор работает. В качестве теста рекомендуем запустить программу RAMDISK и снова вызвать VD. Результат будет другим (для краткости приводятся только изменившиеся строки):

```
Hooks Err N 1
HiMem?
```

Детектор зафиксировал изменение дисковых ловушек (Err N1) и отличие значения HIMEM от стандартного. Теперь наберите VD C:, и детектор обнаружит вирус в несуществующем на диске C boot-секторе (Err N 3).

Диагностические сообщения детектора для минимизации длины программы сделаны предельно краткими. Большинство их вам уже известны. Дополнительно укажем только, что AX-350 — это текстовая метка стандартного boot-сектора. Возможна другая метка — YD-640. В этом случае начальный загрузчик чуть-чуть отличается и детектор выдаст предупреждение Err N 2. У RAM-диска, как уже отмечалось, boot-сектор отсутствует, поэтому вместо метки обычно выводится часть имени первого файла.

Обратите внимание еще на одно сообщение: 41/42 — . Оно означает, что указанные коды в ОЗУ не обнаружены. Если же это не так, вместо минуса будет выведен знак плюс, VD нейтрализует эти коды и напишет Ok, т. е. нормальная перезагрузка по кнопке «Reset» гарантируется.

В заключение приведем расшифровку возможных сообщений VD:

Err N 1 — зафиксировано изменение дисковых ловушек;

N 2 — отличия начального загрузчика в boot-секторе;

N 3 — в области, где у нормального boot-сектора нули, обнаружена какая-то программа;

N 4—6 — три возможных реализации вируса со структурой, изображенной на рис. 1;

N 7 — вирус приведенный на рис. 2;

N 8 — вирус типа MEMORY KILLER, нарушающий регенерацию ОЗУ.

В некоторых случаях после имени файла появляется сообщение «long», которое означает невозможность разместить весь файл в ОЗУ для проверки. Такой файл игнорируется и тестирование продолжается.

Символ 0 после имени файла говорит о том, что этот файл начинается с команды NOP (часто такие файлы возникают при трансляции с макроассемблера). Сообщению можно не придавать значения.

Автор желает всем пользователям спокойной и плодотворной работы на MSX-2. Он будет благодарен за любые сведения о найденных при помощи детектора вирусах, а также о вирусах, на которые VD не реагирует. В последнем случае, если будет прислана хотя бы распечатка вируса, появится возможность в новой версии детектора его учесть. Интерес представляют и теоретически возможные случаи ложного срабатывания детектора. Здесь также кое-что можно исправить, так как в теле детектора содержится ряд эмпирических «подстроечных» констант.

Адрес для переписки: 614600, Пермь, ул. К. Маркса, 24, пединститут, кафедра теоретической физики, Еремину Евгению Александровичу.

Гарантируется возврат всех магнитных носителей, присланных с любой информацией о вирусах.

Литература

Справочное руководство по языку программирования Бейсик для комплектов учебной вычислительной техники на базе персональных компьютеров «Ямаха MSX-2».

Во время подготовки статьи к печати редакция предоставила автору для изучения новый образец вируса. Он замечателен тем, что восстанавливается в памяти после перезагрузки по кнопке Reset, используя описанный прием с кодами 41/42. Если сразу после перезагрузки или при повторном запуске ваш детектор помечает плюсом результат проверки этих кодов, значит, в компьютере действует вирус такого типа. Избавиться от него можно только выключением питания или специально написанной программой.

К сожалению, версия 1.1 детектора не фиксирует наличия этого нового для VD вируса на диске. Несложная «подстройка» позволяет устранить этот недостаток. Требуется только заменить одну из специально предусмотренных констант, а заодно номер версии и контрольную сумму: в адрес 40A следует записать 60, в 14B — 32, а в 469 — 52. Предлагаемое существенное увеличение константы просмотра приведет к росту вероятности ложного срабатывания детектора версии 1.2 по ошибке 6; в дальнейшем усложнением алгоритма анализа надежность работы VD будет повышена.

О. ПАРСАДАНОВ

О пользовательской экспертизе программ

В последнее время существенно изменилась ситуация с оснащением школьных кабинетов ИВТ техникой и программными средствами, и все большее значение приобретает их правильный выбор. Целью данной статьи является попытка выработать общие подходы к проведению начальной пользовательской экспертизы, позволяющей учителю более четко представить возможности интересующих его ППС.

Типы программных средств

Все ППС можно разделить на 2 основных типа, которые условно обозначим как базовые и прикладные. К базовым пользовательским ППС можно отнести:

1. Текстовые редакторы;
2. Графические редакторы;
3. Музыкальные редакторы;
4. Системы Управления Базами Данных:
   4.1. Реляционные,
   4.2. Иерархические (Древовидные);
5. Числовые таблицы (СЧМ);
6. Системы подготовки заданий и обучающих программ.

Прикладные пользовательские ППС включают в себя программы по отдельным предметам и темам школьного курса, различные демонстрационные и развивающие программные средства.

Принципы экспертизы

На каждый класс ППС задается «матрица возможностей», отражающая основные возможности, которые может (должно) иметь данное средство. Экспертиза заключается в определении экспертом соответствия возможностей рассматриваемого ППС заданной матрице.

Для ППС всех классов можно выбрать характеристики, которые надо оценивать в первую очередь, и оценку эту можно сделать достаточно объективно, в виде выбора одного из предлагаемых вариантов. В первом приближении такие характеристики очевидны.

1. Язык реализации	Бейсик
	Бейсик+маш. код
	Машинный код
	Другие (указать)
2. Простота освоения	До 1 урока
	2—3 урока
	Более 3 уроков

3. Пользовательский интерфейс	Слабо Приемлемо Хорошо Отлично
4. Надежность работы	Слабо Приемлемо Хорошо Отлично
5. Хранение информации	Диск, Магнитофон, Диск/Магнитофон
6. Цветовоспроизведение	Да, Нет
7. Возможности распечатки (Для СУБД — активных записей, для СЧМ — выделяемый фрагмент)	Да, Нет
8. Использование экранных окон	Нет До 3 Более 3
9. Экранный справочник-подсказка	Нет Приемлемо Хорошо Отлично

Оценивая надежность программы, эксперт «провоцирует» ее на сбой, нажимая в первую очередь клавиши, работа с которыми не предусмотрена. Очень важно также, чтобы программа не давала сбой при вводе неверных по типу данных и адекватно реагировала на ввод данных, неправильных по смыслу.

При рассмотрении качества пользовательского интерфейса особое внимание уделяется удобству выбора клавиш управления, наличию удобных для пользователя меню и окон, грамотности и полноте диалога с ЭВМ.

Наличие п. 9, на первый взгляд дублирующего пункт о пользовательском интерфейсе, связано с повышенной сложностью освоения некоторых ППС и более развитой системой управления ими по сравнению с другими классами программ. Речь идет в первую очередь о программах типа СУБД или СЧМ, хотя наличие справочника, появляющегося в виде окна при нажатии определенной клавиши, видимо, не повредит любой программе.

В качестве примера рассмотрим возможные критерии для оценки некоторых широко распространенных ППС.

Текстовые редакторы: для этого класса ППС можно выбрать дополнительные характеристики, оценка которых вряд ли будет неоднозначной.

10. Возможности работы с текстом	Сдвиг, раздвиг строк Перенос, копирование строк Контекстные поиск и замена Многооконный перенос Форматирование текста
11. Возможность работы с блоками текста	Да Нет.

Конечно, этот список можно дополнить, но, видимо, для целей первоначальной пользовательской экспертизы он будет достаточен.

Графические редакторы: хотя возможности данного класса ППС очень разнообразны, ограничимся рассмотрением следующих характеристик.

10. Масштабирование всего и/или части изображения	Да Нет
11. Возможности создания библиотек фрагментов изображения и их подгрузки на экран	Нет Да, простые Да, широкие
12. Копирование и/или перенос блоков экрана	Да Нет
13. Получение текста программы изображения	Да Нет

СУБД реляционного типа: для СУБД целесообразно оценить:

10. Количество столбцов таблицы	
11. Количество записей	
12. Возможности редактирования	Через отдельную программу В основной программе
13. Возможности	Просшейшие (только цепочки условий)
командного	Расширение (программируемый запрос)
языка	Разнообразные (многофайловые структуры с отдельным Интерфейсом Пользователя)
пользователя	
14. Количество цепочек условий при поиске	
15. Количество типов данных	
16. Арифметические опеции с данными	Да Нет

Числовые таблицы (СЧМ): для СЧМ ограничимся рассмотрением:

10. Размер таблицы	
11. Количество типов данных	
12. Вставка/удаление строк (столбцов) с соответствующей заменой адресов в формулах	Да Нет
13. Копия/перенос данных с соответствующей заменой адресов в формулах	Только для отдельной клетки Только по столбцу (строке) Для выделяемого фрагмента

14. Копирование	Автоизменение адресов в формулах С запросом на изменение адресов
15. Форматирование данных	Только для отдельной клетки Только по столбцу (строке) Для выделяемого фрагмента
16. Форматы отображения данных (по нарастанию возможностей)	Влево, Вправо Целое число Денежная единица Указание количества цифр после запятой Дополнительные возможности
17. Изменение ширины столбцов	Только для всей таблицы Каждый столбец отдельно
18. Сохранение таблицы	Только вся таблица Любой выделенный фрагмент
19. Скорость загрузки и записи на носитель	Очень медленно Медленно Приемлемо Быстро

Таким образом, в процессе экспертизы ППС надо просто заполнить соответствующую таблицу, выбрав наиболее подходящий вариант ответа. В заключение эксперт кратко, не более одного абзаца на каждый из нижеперечисленных пунктов, характеризует:

рекомендуемую область применения ППС;

сильные и слабые стороны данного ППС;

общее впечатление от уровня реализации ППС.

Последняя часть экспертизы носит наиболее субъективный характер и позволяет эксперту в полной мере проявить свою индивидуальность.

В связи с вышеизложенным подходом явно просматриваются некоторые проблемы, связанные с разработкой отдельных типов ППС. Пока не ясны критерии, которым должны удовлетворять программы, предназначенные для создания текстовых заданий и обучающих программ, а также «обучающие» программы по отдельным предметам и темам школьного курса. Теоретическое осмысление таких критериев, видимо, станет одной из главных задач на ближайшие несколько лет.

Следует также отметить, что этот вопрос не получил пока удовлетворительного решения ни в одной стране мира и проблема создания реально обучающих и развивающих мышление ученика программ в настоящее время весьма активно обсуждается в мировой компьютерной прессе.

В настоящее время учительская экспертная комиссия Ассоциации учителей информатики приступила к проведению широкой экспертизы существующих ППС по описанной выше схеме. Желающие принять участие в этой работе могут обратиться в Центр ассоциации по адресу: 660051, Красноярск, а/я 11533.

А. ГАЗЕЙКИНА, Д. ЕМЕЛЬЯНОВ

Об упорядочении списков

> Feci quod potui, faciant meliora potentes
> (Сделал, что мог, и пусть, кто может, сделает лучше)

С задачей расположения информации, представленной в виде строк символов, в алфавитном порядке, очень часто приходится сталкиваться на практике. Она встает перед программистом всегда, когда возникает необходимость выдать заказчику «в товарном виде» какую-либо информацию из той или иной информационной базы.

Буквы латинского алфавита упорядочены по возрастанию их кодов, поэтому задача упорядочения состоящего из них текста тривиальна. Сложнее обстоит дело с упорядочением списков, состоящих из русскоязычных слов, так как коды букв кириллицы возрастают не в алфавитном порядке. Стандарт на отечественные ПЭВМ, принятый в 1986 г., по которому коды русских букв возрастают в строгом соответствии с алфавитом, к сожалению, не может кардинально решить проблему в связи с непопулярностью отечественной вычислительной техники и ее малочисленностью. Предпочтение отдается западным образцам ВТ, для которой этот вопрос «почему-то» не решен (см., например, код ASCII).

```
5 CLEAR 2000: INPUT"РАЗМЕР СПИСКА:";N
10 DIM C$(N),A(N)
15 S$=" АБВГДЕЖЗИКЛМНОПРСТУФХЦ
                              ЧШЩЬЫЪЭЮЯ"
20 ML=0
```

```
25 FOR I=1 TO N
30 PRINT I;: LINE INPUT"ВВЕДИ
                        СЛОВО:";C$(I)
35 IF LEN(C$(I))>ML THEN ML=LEN(C$(I))
40 NEXT
45 TIME=0
50 FOR I=1 TO N
55 A(I)=0
60 FOR J=1 TO ML
65 IF J<=LEN(C$(I)) THEN
   A(I)=A(I)+INSTR(S$,MID$(C$(I),J,1))
70 A(I)=A(I)*40
75 NEXT J,I
80 FOR I=1 TO N-1
85 FOR J=I+1 TO N
90 IF A(I)>A(J) THEN SWAP A(I),A(J):
                        SWAP C$(J),C$(I)
95 NEXT J,I
100 PRINT"СПИСОК ПО АЛФАВИТУ:":
                  PRINT STRING$(30,"=")
105 FOR I=1 TO N: PRINT C$(I): NEXT
110 PRINT STRING$(30,"="):
        PRINTUSING"ВРЕМЯ УПОРЯДОЧЕНИЯ
                    (СЕК):###";TIME/50
```

Но и эта задача тоже не нова, и тот, кому приходилось с ней сталкиваться, либо разработал свой собственный алгоритм, либо заимствовал его из множества имеющих в соответствующей литературе.

Какому же алгоритму все-таки отдать предпочтение? Естественно, наиболее короткому и быстродействующему. Однако многие из них несовершенны изначально, так как упорядочение проводится только по первой букве строки. Другие требуют задания глубины сортировки, причем с возрастанием ее величины существенно увеличивается время сортировки. Особенно это становится актуальным при обработке достаточно длинных списков символьных строк. Строки обычно бывают различной длины. Какую же глубину сортировки выбрать? Чаще всего ограничиваются глубиной 3—5 символов, но во многих случаях этого оказывается недостаточно.

Предлагаемая программа для «Ямахи» имеет автоматическую регулировку глубины сортировки по самой длинной строке списка, но не более 9 символов, что лимитируется длиной мантиссы вещественной переменной. Мы не считаем ее верхом совершенства (см. эпиграф), но находим красивой и достаточно эффективной.

А. КУЗНЕЦОВ

Грани «Агата»

Постоянно стремясь к новому, более совершенному или более модному, мы порой незаслуженно пренебрегаем немало послужившим старым. А ведь если в начинающем терять свой блеск камне попробовать отыскать какие-то новые грани, то он может снова засиять, и даже невиданным до этого блеском. Попробуем сделать нечто подобное с нашим старым знакомым «Агатом», первым базовым школьным компьютером.

Предшественник «Агата» — американский компьютер «Apple-II», выпущенный в 1977 г., все еще популярен во всем мире. Значит, есть в этой машине нечто такое, чего мы не заметили и не использовали. Действительно, у каждой конструкции (в том числе и у компьютера) есть какие-то специфические преимущества; есть, естественно, и недостатки, необходимо хорошо знать как те, как и другие, чтобы адекватно ориентироваться в имеющемся программном обеспечении и создавать новые программы, максимально используя преимущества и компенсируя недостатки.

Архитектура, схемное решение, система команд процессора и способы адресации «Агата» довольно сложны, а поскольку большинство учителей не имеют специальной подготовки в вопросах схемотехники и программирования в кодах, то можно понять не очень большую популярность «Агата» в нашей школе. Если же добавить к этому 7 книг ребусов, прилагаемых в качестве инструкции...

Однако трудности изучения машины полностью компенсируются ее возможностями. В частности, страничная организация памяти и переключение отображения областей методами схемотехники очень удобны в школьных приложениях, позволяют быстро менять содержимое всего экрана без переноса большой области памяти, как это делается у большинства компьютеров. Количество переключаемых страниц (до 13) позволяет одновременно иметь в памяти до трех вариантов различных фрагментов, поскольку для имитации движения обычно достаточно четырех сменяющих друг друга кадров, особенно привлекает то, что динамика достигается без применения программирования в кодах, средствами только лишь Бейсика.

Программы такого типа достаточно просты. Они состоят из двух частей. Первая

часть является подготовительной: при ее работе текстовые или графические страницы заполняются изображениями, представляющими собой кадры будущей динамической картины. Она работает довольно долго: от нескольких минут (в таком случае можно оставить эту часть в общей программе, которую запускают перед уроком, чтобы прошел подготовительный этап) до нескольких часов. В последнем случае результат ее работы (содержимое заполненных страниц) записывается на диск в виде двоичного файла, который будет загружаться при запуске второй части; первая работает только один раз и в дальнейшем не нужна.

Вторая (или единственная при наличии двоичного файла) часть меняет в циклическом режиме страницы или динамические фрагменты, а также скорость смены страниц в зависимости от нажатия управляющих клавиш.

Рассмотрим в качестве примера две простые программы: одну с встроенной подготовительной частью, другую с двоичным файлом. Первая предназначена для демонстрации внутренних процессов, происходящих в газах при изменении объема и температуры. Можно имитировать изохорный, адиабатный и другие процессы. Программа выполнена в цвете. Подготовительная часть — строки 10—270, исполняющая — со строки 300. После подготовки программа автоматически начинает работать. Можно разделить части введением строки 280 END или 280 GET A$. В таком случае запуск второй части директивой RUN 300 или нажатием любой клавиши. Управление осуществляется клавишами перемещения курсора: ⟨ВВЕРХ⟩ — повышение температуры, ⟨ВНИЗ⟩ — понижение, ⟨ВЛЕВО⟩ — поршень влево, ⟨ВПРАВО⟩ — поршень вправо.

```
10 REM ДЕМОНСТРАЦИОННАЯ ПРОГРАММА
                              ПО ФИЗИКЕ
20 REM "ПРОЦЕССЫ В ГАЗАХ"
100 FOR I=2 TO 8
110 TEXT=I: HOME: M=29: GOSUB 500:
                              RIBBON=5
120 FOR J=5 TO 25 STEP 2
130 FOR K=0 TO 1
140 Y=INT(12*RND(1))
150 HTAB 4+12*K+Y: VTAB J: PRINT "O"
160 NEXT K
170 NEXT J
180 NEXT I
190 FOR I=9 TO 14
200 TEXT=I: HOME: M=17: GOSUB 500:
                              RIBBON=5
210 FOR J=4 TO 26
220 FOR K=0 TO 1
230 Y=INT(7*RND(1))
240 HTAB 4+6*K+Y: VTAB J: PRINT "O"
250 NEXT K
260 NEXT J
270 NEXT I
300 T=4: D=100: N=2: K=8
310 T=T+1: IF T<N THEN T=N
320 IF T>K OR T=K THEN T=N
330 Y=PEEK(-16384)
350 IF Y=153 THEN D=D/2
360 IF Y=154 THEN D=D*2
370 IF Y=149 THEN T=4: N=2: K=8: D=D*2
380 IF Y=136 THEN T=11: N=9: K=14:
                                  D=D/2
382 IF Y=176 THEN TEXT=15: END
385 POKE 8C012,0
390 FOR I=0 TO D
395 NEXT I
400 TEXT=T: GOTO 310
500 RIBBON=2
520 FOR Q=2 TO 31
530 HTAB Q: VTAB 2: PRINT "*"
540 HTAB Q: VTAB 3: PRINT "*"
550 HTAB Q: VTAB 27: PRINT "*"
560 HTAB Q: VTAB 28: PRINT "*"
570 NEXT Q
580 FOR Q=4 TO 26
590 HTAB 2: VTAB Q: PRINT "**"
600 HTAB M: VTAB Q: RIBBON=1:
                    PRINT "**": RIBBON=2
610 NEXT Q
620 RETURN
```

Вторая программа

```
2  REM ПОДГОТОВКА ВОЛНОВОГО ПРОЦЕССА
5  P=-1:C=50
10 INPUT 'X=';X: INPUT 'Y=';Y
20 FOR I=3 TO 10
30 GR=I: P=P+1
40 FOR J=1 TO 64
50 FOR K=1 TO 64
60 S=SQR((X-K)^2+(.8*(Y-J))^2)+P
70 N=INT((S/8-INT(S/8))*8)+1
80 ON N GOTO 90,100,110,100,90,120,
                                130,120,90
90 COLOR=1: GOTO 160
100 COLOR=6: GOTO 160
110 COLOR=3: GOTO 160
120 COLOR=4: GOTO 160
130 COLOR=0
160 PLOT K,J
170 NEXT K: NEXT J: NEXT I
```

```
10 REM ДЕМОНСТРАЦИОННАЯ ПРОГРАММА
20 REM "РАСПРОСТРАНЕНИЕ ВОЛН"
90 HOME
100 HTAB 3: VTAB 10: PRINT
        "  * РАСПРОСТРАНЕНИЕ ВОЛН * "
110 HTAB 8: VTAB 16: PRINT
     " * А.Ф.КУЗНЕЦОВ, ПОС. КУМЕНЫ * "
120 HTAB 7: VTAB 24:
                PRINT "-> - БЫСТРЕЕ":
            PRINT "     <- - МЕДЛЕННЕЕ"
125 HTAB 7: VTAB 28:
                PRINT "<ПРОБЕЛ> - СТОП"
130 GET A$
140 PRINT
150 PRINT CHR$(4);"BLOADWLN"
180 C=20
190 FOR I=3 TO 10
195 L=13-I
200 IF Z<127 THEN 230
205 IF Z-128=32 THEN GET A$
210 IF Z-128=8 THEN C=C+20: GOTO 230
220 IF Z-128=21 THEN C=C-20
225 IF C<0 THEN C=0
```

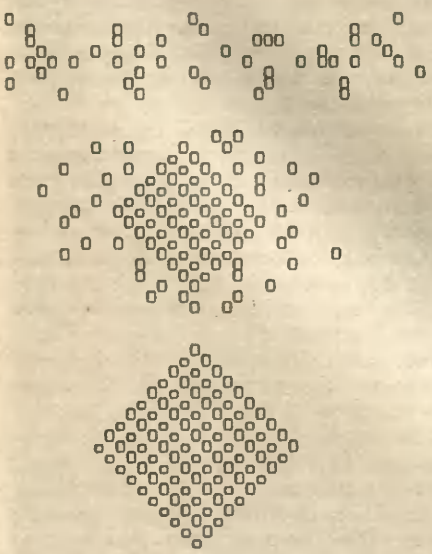
```
230 FOR M=0 TO C
232 Z=PEEK(&C000)
235 NEXT M
237 POKE &C010,00
240 ST=4*L: POKE(&C700+4*L),00
250 NEXT I
260 GOTO 190
```

предназначена для демонстрации волн от одного источника в однородной среде. Вначале вводятся координаты источника (0—63). Подготовительная часть работает около 2. После окончания ее работы на диск, где размещена исполняющая часть программы, записывается двоичный файл директивой

```
BSAVE WLN,A&1800,L&4000<ВК>
```

Запуск исполняющей части производится обычным образом. Управление — клавишами перемещения курсора: ⟨ВЛЕВО⟩ — медленнее, ⟨ВПРАВО⟩ — быстрее. Программа выполнена в цвете, но хорошо смотрится и на черно-белом экране. ⟨ПРОБЕЛ⟩ останавливает картинку. Нажатие любой клавиши продолжит работу.

Рисунки демонстрируют работу программы моделирования процесса кристаллизации, также использующей данный принцип работы. Все подобные программы отличаются сравнительно короткими текстами, большой зрелищностью. Ученики всегда с интересом наблюдают их демонстрации на уроке.



Фрагменты программы "Плавление и кристаллизация"

Возможность отображения любой области памяти в любом из пяти режимов может быть с успехом использована для более подробного изучения работы компьютера. При работе любой программы, в том числе и системной, происходит изменение содержимого ячеек ОЗУ. Подсмотреть, что происходит в памяти машины при ее работе, можно следующим образом. Если мы включим отображение какой-то области путем обращения к ячейкам управления (операторы PEEK или POKE), то это в отличие от директив включения страниц никак не влияет на работу этой области памяти. Правда, набираемые директивы не видны на экране, они заносятся в ту текстовую страницу, которая была включена, и становятся видимыми после нажатия клавиши ⟨СБР⟩, а исполняются обычным образом.

Особенно важно следить за содержанием ячеек нулевой страницы. Там находятся системные ячейки, хранящие сведения о режимах работы (цвет, положение курсора и т. д.), используемые при короткой адресации. Включение ее производится директивой


```
POKE&C782,00
```

Содержимое ячеек отображается на экране символами, код которых равен числу в данной ячейке. Набор директив или текста программ можно контролировать, так как они помещаются в буфер строки с адреса &200 (начало девятой строки, считая сверху). Если нас интересует, например, какие ячейки отвечают за цвет текста, можно набрать короткую программу

```
10 GETA&:RIBBON=VAL(A&):GOTO10
```

Включив нулевую страницу на отображение и запустив программу обычным способом, нажимают цифровые клавиши и наблюдают, какие ячейки меняют свое содержимое. Так можно узнать назначение любой ячейки памяти. Если вас интересует вид программы во внутреннем представлении машины, то введите

```
POKE&C786,00
```

Если теперь набрать строку (на экране она не будет видна) и ввести ее, то на экране она появится во внутреннем представлении. Интересно наблюдать объявление массивов.

При отображении различных участков памяти следует иметь в виду, что картина на экране зависит от режима отображения. Если характер информации, записанной в данной области, не соответствует режиму отображения, то картина представляет бессмысленный набор символов различного цвета и режима воспроизведения (инверсный, мерцающий).

И. ПОЛИВАНЫЙ

Графические возможности диалоговой инструментально-учебной системы программирования

Машинная графика в рамках школьного курса ОИВТ представляет собой большой интерес. С одной стороны, как самостоятельный и очень важный раздел информатики, а с другой, как мощное педагогическое средство, позволяющее повышать наглядность обучения и его эффективность. Многолетний опыт проведения занятий со школьниками свидетельствует о том, что учащиеся, работая с машинной графикой, легче осваивают такие базовые понятия программирования, как циклы, ветвление, подпрограммы и т. п. [1]. Поэтому не случайно во всех современных учебных системах программирования присутствуют различные средства работы с машинной графикой.

Не является исключением в этом отношении и диалоговая инструментально-учебная система программирования (ДИУС) [2, 3, 4, 5]. Первые версии системы ДИУС были реализованы на ПЭВМ «Агат» седьмой модификации и написаны на языке Ассемблер микропроцессора 6502. Выбор данного компьютера обусловлен соображениями, изложенными в [6].

Апробация ДИУС проводилась в течение двух лет (1988—1989) в V—VI классах средней общеобразовательной школы. Контроль за физиологическим состоянием детей осуществлялся по программе, разработанной в НИИ ФДиП АПН СССР.

Входной язык системы, получивший название УМКА (Учебный микро-компьютерный алгоязык), обладает сравнительно небольшим набором команд (10), однако он является достаточно мощным языком программирования, позволяющим производить сложные математические расчеты, в полном объеме работать с символьной информацией, получать нетривиальные графические изображения, эффективно работать с динамической графикой, производить ввод и вывод алфавитно-цифровой информации, а также осуществлять музыкальный вывод.

Прежде чем говорить о графических возможностях системы ДИУС, представляется целесообразным проанализировать сначала, хотя бы вкратце, аналогичные возможности других учебных систем программирования. Учитывая всю сложность проведения такого анализа в пределах одной статьи, мы ограничимся лишь рассмотрением имеющихся подходов к организации графического вывода в указанных системах. При этом постараемся выяснить, с какими проблемами приходится сталкиваться школьнику, работающему в рамках того или иного подхода.

1. Машинная графика в декартовой и «черепашьей» системах координат.

Вообще говоря, графические возможности языка программирования зависят от того:

в какой системе координат приходится оперировать пользователю, выводя на экран дисплея графическую информацию;

каким набором графических примитивов (точка, отрезок, дуга и т. п.) он при этом располагает;

какими дополнительными средствами обладает язык для манипулирования графической информацией (помимо традиционной возможности вывода графических примитивов).

Понятно, что среди перечисленных факторов ключевым является первый, т. е. используемая система координат. Именно она во многом определяет базовый набор примитивов и средства манипулирования графической информацией.

В учебных языках благодаря своей простоте наибольшее распространение получили декартовы и полярные координаты. При этом как в декартовой, так и в полярной системе координат точки могут задаваться либо абсолютными, либо относительными (относительно последней изображенной точки) значениями. Поэтому остановимся на четырех возможных подходах.

Декартовы абсолютные координаты (ДАК) нашли свое воплощение в языках программирования BASIC, Рапира.

ДАК чрезвычайно удобны для построения графиков функций, однако для учебных систем программирования, ориентированных на школьников младших и средних классов, это свойство ДАК не представляется решающим.

Построение простейших рисунков в рамках ДАК осложняется необходимостью ссылаться на фиксированную точку отсчета, которая нередко весьма удалена от самого рисунка. Помимо этого, ситуация часто усугуб-

ляется «перевернутостью» системы координат BASIC, сложностью работы с отдельными графическими примитивами. Например, для изображения дуги в рамках MSX — BASIC ребенок должен указать семь параметров: координаты центра соответствующей окружности (задача нахождения этих координат для произвольной дуги является далеко не простой), начальный и конечный угол дуги «в радианах от направления направо против часовой стрелки» и т. д.

Возможности мультипликационного вывода на основе ДАК ограниченны. С одной стороны, мультипликацию здесь программировать хлопотно из-за необходимости просчитывать на каждом шаге регенерации координаты всех ключевых точек объекта относительно центра, а с другой — расчеты, выполняемые интерпретатором, снижают скорости регенерации. Поэтому не случайно во многих BASIC-системах реализованы дополнительные средства графического вывода для получения качественной мультипликации (*Sprite* в *MSX-BASIC*, *Shape* в *APPLESOFT-BASIC* и т. п.), которые для привязки к ДАК требуют указания координат единственной точки.

Полярные относительные координаты (ПОК) нашли свое воплощение в языке программирования *LOGO*. Здесь ПОК удачно сочетаются с концепцией исполнителя (Черепашки).

Язык прост в изучении. С его помощью без труда можно построить квадрат, прямоугольник, равносторонний треугольник, а в цикле даже красивые многоугольные звездочки. Однако попытка построить в рамках ПОК более «сложный» рисунок, например любой неравносторонний треугольник, неизбежно натолкнется на проблему возврата в исходную точку (без теоремы синусов или косинусов здесь трудно обойтись). Авторы учебников по *LOGO* решают эту проблему либо за счет использования в командах «НАПРАВО» и «НАЛЕВО» ограниченного набора угловых значений: 90°, 60° и 30° ($sin\ 30° = cos\ 60° = {}^1/_2$), либо за счет привлечения средств ДАК, а именно команд: «ДО *xy*» — передвинуть Черепашку до точки с координатами (*x*, *y*); «ДОАВСЦ *x*» и «ДООРД *y*» — передвинуть Черепашку горизонтально (вертикально) до точки с абсциссой *x* (ординатой *y*). Помимо этого, в рамках ПОК весьма непросто построить дугу и окружность. Для осуществления элементарного перемещения графического курсора ребенку, работающему с *LOGO*, необходимо выполнить четыре команды: «ПОДНЯТЬ ПЕРО», «НАПРАВО», «ВПЕРЕД», «ОПУСТИТЬ ПЕРО».

Возможности мультипликационного вывода в рамках ПОК ограничены необходимостью вычислять на каждом шаге регенерации значения тригонометрических функций в каждой узловой точке объекта, а на это, как известно, требуется немало времени работы микропроцессора.

Однако необходимо признать, что педагогическая ценность учебной системы не может определяться одной лишь графикой. Простота и наглядность команд языка *LOGO* во многом окупают его графическую невыразительность.

Полярные абсолютные координаты (ПАК) ни в одном из известных учебных языков не нашли какого-либо логически завершенного воплощения. Во многих системах (*LOGO*, Школьница, ДИУС) графические команды на базе ПАК могут быть смоделированы, однако в качестве основы графического вывода они пока не используются, хотя и здесь вполне возможны интересные решения.

Декартовы относительные координаты (ДОК) легли в основу графики системы ДИУС.

Не секрет, что ДОК и раньше использовались в учебных языках программирования (*BASIC*, РАПИРА) для построения векторов, перемещения курсора, однако только в ДИУС ДОК были реализованы в «чистом» виде, т. е. на их основе строится вся графическая идеология языка УМКА (вывод примитивов, осуществление закраски, мультипликация, работа с фиксированными точками и т. п.).

ДОК обладают рядом очевидных достоинств.

Они просты в освоении, не требуют от ребенка предварительных знаний об углах поворота, градусах, абсолютных координатах. Школьник должен лишь уметь находить величину смещения, переходя от одной точки на плоскости к другой.

ДОК позволяют осуществлять вывод всех графических примитивов в единообразной форме, благодаря чему графика ДИУС базируется практически на одной команде. В связи с этим ребенку одинаково просто нарисовать на экране точку, отрезок, дугу, окружность.

В рамках ДОК удобно строить рисунки произвольной сложности, так как, с одной стороны, отсчет всегда ведется от текущей точки, которая всегда «под рукой», а с другой стороны, с помощью дуг легко аппроксимируются любые кривые.

ДОК позволяют достаточно просто получать качественную мультипликацию. Во-первых, она без труда программируется, не требуя от ребенка использования команды присваивания. Во-вторых, регенерация

изображения осуществляется, практически моментально, так как микропроцессор оперирует исключительно с целыми константами на уровне машинных команд *ADC* (сложить) и *SBC* (вычесть).

Среди недостатков ДОК можно отметить общую для всех относительных координат проблему возврата в исходную точку. Однако в рамках ДИУС эта проблема решается достаточно просто с помощью механизма фиксированных точек, который описывается ниже.

2. Машинная графика на основе одной команды.

'РИС' — единственная команда языка УМКА, обеспечивающая вывод графической информации. Формат команды следующий: РИС код операции, $\triangle X$, $\triangle Y$.

Здесь «код операции» — это символ, указывающий на характер выполняемого действия (например, '/' — построить отрезок, '#' — осуществить закраску и т. п.); '$\triangle X$, $\triangle Y$' — смещение относительно текущей точки. Положительный отсчет координат по оси X ведется слева направо, а по оси Y снизу вверх.

Итак, команда 'РИС' позволяет:

изменять позицию графического курсора, т. е. осуществлять перескок от текущей точки к точке, смещение до которой указано в команде, например: РИС∧,10, 20

изображать отрезок от текущей точки к точке, смещение до которой указано в команде, например: РИС/, 200, 0

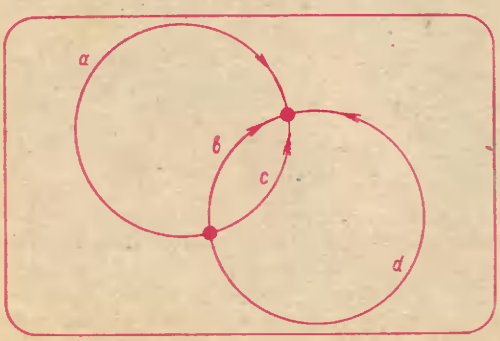
изображать дугу от текущей точки к точке, смещение до которой указано в команде, например:

РИС (,20,30 — малая левая дуга (рис. 1, *b*);

РИС),20,30 — малая правая дуга (рис. 1, *c*);

РИС [,20,30 — большая левая дуга (рис. 1, *a*);

РИС],20,30 — большая правая дуга (рис. 1, *d*).



После запуска системы автоматически устанавливается значение радиуса, равное 30. Для того чтобы задать новое значение радиуса, необходимо его числовое представление записать перед кодом дуги. Например: РИС 10[,20,20

Изображать окружность. Для этого необходимо построить большую дугу с минимальным смещением (0,1), (1,1) и т. п. в зависимости от требуемой ориентации окружности. Например: РИС [,0,1 РИС],1,1 РИС [,1,0

Производить закраску замкнутой области. В данном случае смещение будет определять точку, начиная с которой необходимо осуществлять закрашивание. Например: РИС #,1,1

Изменять цвет выводимой графической информации. При запуске ДИУС автоматически устанавливается белый цвет вывода. Для изменения цвета необходимо в одной из трех команд: 'перескок', 'отрезок', 'закраска' — перед кодом команды записать код цвета. Например: РИС∧0, 0,0 РИС 1#2,2

Фиксировать положение текущей точки в стеке. Данная возможность бывает полезной при изображении сложных рисунков. Как правило, в этом случае возникает необходимость вернуться к точкам, которые уже были использованы в процессе изображения. Поэтому, реализуя настоящую возможность, пользователь может запоминать в стеке положение узловых точек рисунка с тем, чтобы впоследствии иметь возможность вернуться к ним без подсчета соответствующего смещения. Для однократного фиксирования положения текущей точки в стеке используется команда 'РИС' без параметров: РИС. Для многократного фиксирования положения текущей точки в стеке после ключевого слова 'РИС' указывается соответствующее количество запятых. Например, команда РИС , , , осуществляет четырехкратное фиксирование положения текущей точки в стеке;

Осуществлять возврат к ранее зафиксированным в стеке точкам. Возврат может быть реализован в виде перескока, проведения отрезка, дуги и т. п. Для осуществления возврата необходимо в команде 'РИС' опустить смещение. Например: РИС/ провести отрезок от текущей точки до точки, положение которой зафиксировано на вершине стека. После использования зафиксированной на вершине стека точки информация о ней из стека выталкивается и для работы становится доступной точка, которая была зафиксирована в стеке ранее;

Комбинировать все перечисленные выше возможности в рамках одной команды. Например: РИС∧,20,100,/,20,—100,/,20,100 РИС ,/,10, 20,/,10 —40,/

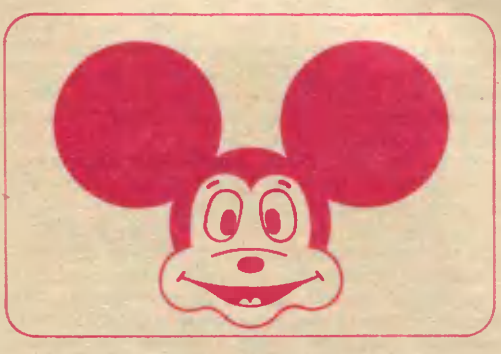
Отменять результат выполнения последней команды 'РИС'. Данная возможность полезна в двух случаях: во-первых, для исправления неверно выполненных действий при работе в диалоговом режиме и, во-вторых, для организации мультипликационных изображений в программном режиме. В первом случае отмена осуществляется без смещения, т.е. с возвратом к предыдущей текущей точке. Команда имеет вид: РИС —

Во втором же случае для обеспечения динамики необходимо изменять положение текущей точки на величину смещения. Команда имеет вид: РИС —,5,2 Пример программы, осуществляющей простейшую мультипликацию:

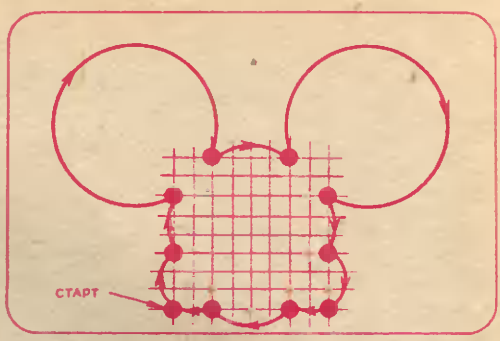
```
ПОВТОР 40
    РИС  /,20,0,/,-5,5,/,-10,0,/,-5,-5
    РИС  -, 3,2
```

Здесь первая команда 'РИС' осуществляет изображение летающей тарелки. Команда 'РИС —, 3,2' осуществляет ликвидацию полученного изображения с последующим смещением графического курсора. Многократный повтор этой группы команд позволяет получить на экране эффект мультипликационного изображения. Варьируя величину смещения в отменяющей команде 'РИС', можно изменять направление движения объекта и его скорость.

Порядок построения простейших рисунков с помощью команды 'РИС' продемонстрируем на следующем примере. Допустим, мы хотим получить на экране дисплея изображение Микки Мауса (рис. 2).



Для этого мы должны в первую очередь выделить на рисунке основные узловые точки, т. е. точки, которые разбивают линии изображения на фрагменты, близкие к дугам и отрезкам (рис. 3).



Затем, учитывая смещения, соединяем узловые точки соответствующими дугами и отрезками:

```
РИС (, 0, 30,     (,    0,  30
РИС [, 20, 20,    (,   40,  -0,   ,  20,—20
РИС (, 0, —30,    (,    0, —30
РИС §, —20, 0,    (, —40,   0, /, —20,   0
```


В результате мы получим «каркас» нашего изображения, обрисованный дугами одного радиуса. Осталось подобрать подходящие значения радиусов для каждой дуги с тем, чтобы прблизить «каркас» к желаемому виду:

```
РИС 16(,   0,  30, 40(,   0,  30
РИС    [,  20,  20,   (,  40,   0,   [,  20, —20
РИС    (,   0,—30, 16(,   0, —30
РИС    /,—20,   0, 25(,—40,   0,   /,—20,   0
```

Помимо простых рисунков система ДИУС позволяет получать и более «серьезные» изображения: графики функций, круговые диаграммы, проекции тел трехмерного пространства и многое другое [4]. Например, следующая программа строит гистограмму на основе значений, генерируемых датчиком случайных чисел (рис. 4).

```
ПОВТОР 10
  РИС
  ПОВТОР INT(100*RND(1))
    РИС /,20,0, ^,-20,1
  РИС ^,^,20,0
```

Если же рамки команды 'РИС' тесны для пользователя, то он всегда может смоделировать в рамках ДИУС необходимую ему графическую среду, которая наиболее полно отвечала бы его потребностям.

3. Создание других графических команд в системе ДИУС.

В отличие от существующих учебных систем программирования ДИУС обладает уникальными инструментальными свойствами, благодаря чему преподаватель имеет возможность приспосабливать учебную среду к своим нуждам, подчинять входной язык собственной методической схеме. В рамках системы могут быть видоизменены команды входного языка, смоделированы операторы и конструкции известных учебных языков программирования, реализованы новые экспериментальные языки и многое другое.

Механизм порождения и использования новых команд весьма прост и не выходит за рамки правил работы с обычными подпрограммами языка УМКА, поэтому даже ребенок может придумывать и создавать свои собственные команды. Как показал опыт, дети с большим интересом воспринимают данную возможность.

Не вдаваясь в подробности, отметим, что инструментальные возможности ДИУС обеспечиваются наличием в системе:

специальной области памяти, предназначенной для хранения подпрограмм (написанных на языке УМКА), благодаря чему преподаватель может «прятать» от ребенка вновь определенные операторы;

инструментального режима работы, позволяющего осуществлять редактирование подпрограмм, хранящихся в указанной области памяти;

самим порядком организации работы с подпрограммами, причем вызов вновь определенных команд синтаксически не отличается от обращения к встроенным командам входного языка.

Итак, на основе десяти встроенных команд входного языка и механизма создания новых команд преподаватель, например, может смоделировать графические команды таких учебных языков, как

MSX — BASIC	*LINE* (10,10)—(10,200),1 *LINE* (10,200)—(150,150),1 *LINE* (150,150)—(10,10),1 *PAINT* (100,100),1
APPLESOFT-BASIC	*COLOR=1* *PLOT* 10,10 *PLOT TO* 10,200 *PLOT TO* 150,150 *PLOT TO* 10,10
LOGO	ЦЕНТР БЕЗСЛЕДА ВЛЕВО 60 ВПЕРЕД 30
РАПИРА:	ТРЕУГОЛЬНИК (10,10,10,200, 150,150) ПРЯМОУГОЛЬНИК (0,0,80,60) ЛИНИЯ (0,0,100,100)

с тем, чтобы работать с новыми командами как со встроенными командами входного языка.

В заключение отметим, что простота освоения системы и небольшой набор команд входного языка позволяют использовать ДИУС для обучения школьников младших, средних и старших классов, а также система может оказаться полезной для любого пользователя, начинающего изучать информатику. Как показала апробация, система способна существенно облегчить процесс формирования у обучаемых основных понятий и навыков программирования. Дети буквально с первого урока начинают практическую работу на компьютере, быстро осваивают команды входного языка, без затруднений переходят к другим языкам и системам программирования.

Для получения дополнительной информации о системе ДИУС необходимо написать по адресу: 637002, Павлодар, ул. Мира, 60, Павлодарский пединститут, кафедра информатики; тел.: 75-29-95.

Литература

1. *Звенигородский Г. А., Салихова А. К., Цикоза В. А.* Машинная графика в математическом обеспечении учебного процесса // Прикладные методы информатики. Новосибирск, 1980.

2. *Беркуцкий В. Я., Поливаный И. В.* Диалоговая инструментально-учебная система программирования // Микропроцессорные средства и системы. 1990. № 1. С. 35.

3. *Поливаный И. В.* Диалоговая инструментально-учебная система программирования: Руководство пользователя. Павлодар, 1991.

4. *Беркуцкий В. Я., Поливаный И. В.* Диалоговая инструментально-учебная система программирования: Пособие для учителя. Павлодар, 1991.

5. *Поливаный И. В.* Диалоговая система программирования начального обучения (ДИУС) // Актуальные вопросы педагогической технологии качества учебного процесса. Труды региональной межвузовской научно-методической конференции. Алма-Ата, 1990.

6. *Беркуцкий В. Я., Поливаный И. В., Щеглов С. А.* Лица «Агата» // Информатика и образование. 1989. № 5. С. 71—72.

С. КОРЕШКОВ

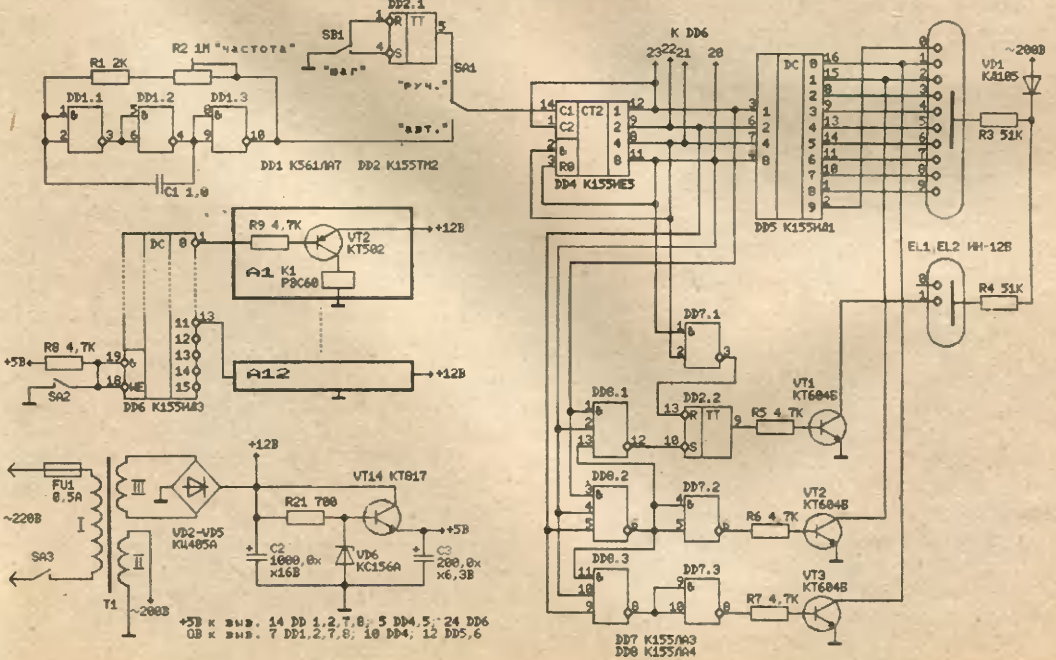
Доработка дисплейного класса

В «ИНФО» № 3 за 1989 г. была опубликована статья А. Караваева «Электронный коммутатор для дисплейного класса». Очень нужный материал. В нашем кабинете установлен КУВТ-86 на базе ДВК-3МШ, и, конечно, при работе возникают те проблемы, о которых пишет А. Караваев. Есть и особенности. В частности, установка переключателя «РМУ—РМП» и отдельного видеовхода в ДВК вызвала возражения со стороны ремонтной организации, поэтому пришлось использовать телевизор, работающий как демонстрационный монитор. При изготовлении коммутатора мы внесли в схему некоторые изменения.

В генераторе применена ИМС типа К561ЛА7, что позволило повысить надежность его работы на низких частотах, использовать С1 малой емкости (1 мкФ), исключить из схемы делитель частоты (DD3 в статье) и триггер, работающий в счетном режиме (DD2.2 в статье). Подстроечный резистор R1 установлен на плате (частота генератора примерно 1 Гц). Переключатель SA1 устанавливает автоматический или ручной режим выбора номера РМУ.

Соединив вывод 2 счетчика DD4 с выводом 8, а 3 — с 11, мы «заставили» его считать не от 0 до 15, а от 0 до 11, чтобы после просмотра «картинки» РМУ12 без задержки происходил возврат к РМУ1. Включив в разрыв проводника, соединяющего выводы 18 и 19 DD6 с общим проводом, переключатель SA2, получили режим, позволяющий выбирать номер РМУ при отключенных транзисторных ключах, что также бывает необходимо в процессе работы. Сами ключи мы упростили, собрав их на одном резисторе, одном транзисторе и миниатюрном реле типа РЭС-60. Питание ключей осуществляется напряжением 12В, которое снимается непосредственно с выпрямителя.

Блок индикации номера РМУ полностью изменен. Индикация осуществляется газоразрядными индикаторами ИН-12Б, высвечивающими номера РМУ от 1 до 12. Блок представляет собой дешифратор двоичного кода для двухразрядного индикатора EL1, EL2. DD5 работает в режиме дешифратора единиц, но индикатор EL1 подключен таким образом, что индикация начинается с цифры 1. По мере поступления импульсов на вход 14 DD4 происходит увеличение показаний EL1. Девятый импульс вызовет свечение 0, элемент DD8.1 переведет триггер DD2.2 в единичное состояние, ключ на транзисторе VT16 откроется —



Выходы счетчика DD4				Цифра на индикаторе	
8	4	2	1	EL1	EL2
0	0	0	0		1
0	0	0	1		2
0	0	1	0		3
0	0	1	1		4
0	1	0	0		5
0	1	0	1		6
0	1	1	0		7
0	1	1	1		8
1	0	0	0		9
1	0	0	1	1	0
1	0	1	0	1	1
1	0	1	1	1	2
1	1	0	0	Сброс DD4 и DD2.2	

индикатор десятков EL2 высветит 1. Так как дешифратор DD5 является двоично-десятичным, то десятый импульс на входе 14 DD4 переведет все выходы DD5 в высокоимпедансное состояние, но элементы DD8.2 и DD7.2 откроют транзистор VT1 — на индикаторе EL2 загорится 2. Двенадцатый импульс переведет счетчик DD4 в нулевое состояние — EL1 высветит 1, элемент DD7.1 переведет триггер DD2.2 в нулевое состояние, EL2 погаснет, и цикл индикации повторится. Более детально зависимость высвечиваемых цифр от сигналов на выходе DD4 представлена в таблице.

Надеюсь, наша информация окажется полезной преподавателям ОИВТ. Адрес для переписки: *215010, Смоленская обл., г. Гагарин, ул. Советская, 5, каб. 7.*

ДОСКА ОБЪЯВЛЕНИЙ

Ищу единомышленников, владельцев компьютера «Вектор-06Ц». Могу предложить большое количество программ, в том числе и абсолютно новые, например, работающие с дисководом; схемы подключения периферийных устройств.

400120, Волгоград, ул. Ростовская, 8, кв. 58. В. И. ЛАПШОВ

Предлагаю владельцам машин «Искра 1080 Тарту», «Искра 1081 (1083, 1084)» объединиться в клуб по переписке.

427760, Удмуртская АССР, г. Можга-7Г, мкрн. 1, д. 15, кв. 13. А. Е. АНИСИМОВ

Обмениваюсь программами для «Микроши» и РК-86. Имею широкий выбор игровых и системных программ, включая серию принципиально новых игровых пакетов.

Москва, тел. 365-66-79. С. В. БОГДАНОВ

Заочный компьютерный клуб приглашает всех владельцев «Микрош» к сотрудничеству.

231400, Гродненская обл., г. Новогрудок, ул. Ленина, 57, кв. 3. И. С. КОТЕНКО

Предлагаю переписываться и обмениваться программами владельцам БК-0010(01). Имею множество программ и большой опыт работы на БК. Высылаю каталог.

194214, Ленинград, Костромской пр., 38, кв. 29. К. И. ГРЕБНЕВ

Хочу обмениваться программами и информацией с владельцами БК-0010.01. Имею около 300 программ. Вышлю каталоги. Интересуюсь программированием на Бейсике и в кодах. Отвечу на все письма.

220101, Минск, ул. Якубова, 30, кв. 23. Тел. 49-64-13. П. В. ФЕДОРЕНКО

Хотел бы переписываться и обмениваться программами с владельцами БК-0010.01. Вышлю каталог.

450055, Уфа, просп. Октября, 136/1, кв. 34. С. Г. ПРОКШИН

Предлагаю обмен программами для ПК «Партнер». Интересуюсь программами для «Спектрума».

623416, Каменск-Уральский, ул. Репина, 58. С. В. ФЕДОТОВ

Прошу откликнуться тех, кто может помочь с программами для КУВТ УКНЦ 01.01 (на базе БК-0011М). Принтер Д-100, НГМД МС 5309 (80 дорожек).

346330, Ростовская обл., г. Жонецк, ул. Кольцевая, 11, кв. 2. С. А. БРЕДИН

Предлагаю обмениваться опытом начинающим программировать на Форте.

658213, Алтайский край, г. Рубцовск, ул. Дзержинского, 15, кв. 74. Е. Г. АГЕЕНКО

Хочу переписываться и обмениваться программами, литературой по IBM PC, «Правец-8Д».

253156, Киев, ул. Матеюка, 5, кв. 21. А. А. ПЛАТКЕВИЧ

Предлагаем переписку, обмен программами владельцам компьютеров МС 1502, «Поиск», «Ассистент».

143500, Московская обл., г. Истра, ул. 9-й Гвардейской дивизии, 60, кв. 32, клуб «ЗТ», Ю. П. ТРОФИМОВУ

Учителям информатики, работающим с УКНЦ и испытывающим недостаток ПО и информации, предлагаем бесплатную помощь.

426075, Ижевск, а/я 1646, УКНЦ

Имею более 100 программ для «Микроши».

103498, Москва, корп. 446, кв. 80. В. Н. КУШНЕРОВ

Хочу переписываться и обмениваться программами с владельцами БК-0010.01.

232007, Вильнюс, ул. Кривю, 35/1, М. Ч. ПЕТКЕВИЧ

Хотел бы переписываться и обмениваться программами с владельцами «Вектора-06Ц».

652210, Кемеровская обл., пос. Тисуль, пер. Почтовый, 11. А. В. ГОРЕЛОВ

Хотел бы переписываться и обмениваться программами с владельцами «Атари» и «Ямахи MSX-2». Имею немало разнообразных программ.

690105, Владивосток, ул. Русская, 59, корп. 1, кв. 23. В. А. БАХТЕЕВ

Хочу обмениваться программами (игровыми, обучающими, системными) с обладателями «Вектора-06Ц». Имею более 200 программ (45 в кодах), работающие квазидиск, дисковод и внешнее ПЗУ.

400066, Волгоград, ул. Краснознаменская, 8, кв. 63. А. Л. ПОНОМАРЕВ

36-я средняя школа г. Томска предлагает сотрудничество работающим на УКНЦ. Ищем информацию по формулам преобразования чисел в АЦЭ, распределению памяти, музыкальным возможностям УКНЦ.

634062, Томск, ул. Беринга, 5/1, кв. 23. А. Н. ВЕРНИГОРА

Интересуюсь созданием программ к компьютеру ПК-01 «Львов». Готов обмениваться программами к нему.

332313, Мелитополь, ул. 4-я Продольная, 17, кв. 3. С. Г. КОНДАКОВ

Хотел бы обмениваться программами и информацией с владельцами БК-0010. Имею большой банк игровых, системных, учебных программ. Высылаю каталог в кратчайшие сроки.

141160, Московская обл., Щелковский р-н, Звездный городок, 48, кв. 29. Д. А. ЖУРАВЛЕВ

Имею более 400 системных, учебных и игровых программ для «Микроши» и «Кристы», 1000 игр для «Спектрума». Высылаю каталоги программ с адресами и краткими комментариями.

663310, Норильск, ул. Талнахская, 1, кв. 128. Тел. 4-63-14. В. В. МОКРЯК

Имею приличный набор игровых программ для ИИК «Ассистент», хотел бы обмениваться и переписываться с обладателями этого компьютера.

215010, Смоленская обл., г. Гагарин, ул. Гагарина, 47, СПТУ-8. Б. А. ОРЛОВ

Книга-почтой для пользователей «Электроника БК-0010».
Адрес: *105187, Москва, Кирпичная ул., 39-41, ПТО «Магистр-2», ИТС*

Малым предприятием «ШИРАКАЦИ» разработано и серийно выпускается

устройство программирования микросхем ППЗУ на базе «Электроника БК-0010.01» — «Программатор SH-PR03».

Программатор предназначен для программирования микросхем следующих типов:

К1801РР1 (без аналога);
К573РФ2, 21, 22 (2716);
К573РФ3, 31, 32, 33, 34 (без аналога);
К573РФ4А, Б (2764-2);
К573РФ41А, Б, 42А, Б (2732);
К573РФ6А, Б (2764);
К573РФ7 (27256);
К573РФ8А, Б (27256);
К573РФ81А, Б, 82А, Б (без аналога);
КС1626РФ1А, Б, 11А, Б, 12А, Б (27С64);
К558РР3 (без аналога).

«Программатор SH-PR03» является программно-аппаратным устройством, работающим совместно с бытовой микро-ЭВМ БК-0010.01, БК-0010. Конструктивно программатор выполнен в виде модуля, подключаемого к порту пользователя. Габаритные размеры: 124×100×25 мм (корпус МСТД). Для работы требуются внешние источники питания: +5,5 В (300 мА) и регулируемый +12...+25 В (100 мА).

Программатор обеспечивает следующие возможности:

выбор типа программируемой микросхемы;
контроль годности ППЗУ;
чтение содержимого ППЗУ в буфер;
запись содержимого буфера в файл на диске или магнитной ленте;
чтение файла в буфер;
запись в ППЗУ содержимого буфера с возможностью указания начального и конечного адресов ППЗУ и буфера;
контроль записанной информации с выдачей сообщения оператору;
вывод содержимого буфера на печатающее устройство;
редактирование информации в буфере.

Программатор предлагается в любой комплектности из следующего набора:

модуль программирования — 199 рублей;
программное обеспечение на дискете или ленте — 98 рублей;
источник питания регулируемый — 96 рублей;
микро-ЭВМ «Электроника БК-0010.01» — 850 рублей;
контроллер дисковода 133 мм — 399 рублей.

Оплата может производиться как по перечислению, так и наличными. Для партий свыше 50 шт.— скидка 10 %.

Наш адрес: 375079, Ереван, МП «Ширакаци».
Тел. 63-24-31, 64-57-18, 63-29-47.

Ю. ЗАЛЬЦМАН

Продолжительность исполнения команд на БК-0010

Основное преимущество ассемблера перед прочими языками — скорость выполнения программ. По техническим условиям для БК-0010 она равна 300 тыс. операций в секунду. Но это не более чем абстрактная цифра, имеющая очень мало общего с фактической скоростью работы ЭВМ. Скорость исполнения программы, состоящей из реальных команд, зависит от многих факторов, из которых главными являются тип команд и способы адресации. С помощью специального устройства (электронного измерителя длительностей) в сочетании со специально написанными для этой цели программами время исполнения отдельных команд может быть непосредственно измерено, а затем путем логических рассуждений и расчетов могут быть найдены правила для вычисления времени исполнения любых команд БК-0010 с любыми методами адресации. Автор проделал такое исследование. Следует заметить, что правила вычисления времени исполнения команд для БК-0010 оказались заметно отличными от, например, аналогичных правил для ЭВМ типа PDP-11. Это естественно, так как внутренняя структура процессоров этих ЭВМ и их микропрограммное управление (несмотря на одинаковую систему машинных команд) заметно различаются. Итак, приступим к изложению полученных на БК-0010 результатов.

Прежде всего необходимо отметить, что тактовая частота процессора, установленная в БК-0010, обеспечивает реальное быстродействие примерно 250 тыс. операций в секунду. Но что это за операции? Речь идет об операциях типа «регистр—регистр», т. е. об адресации с кодом «0», или «RN». При всех прочих способах адресации скорость исполнения команд уменьшается. Время исполнения команд с кодом адресации «0» (или вообще не имеющих операндов) носит название основного времени команды, для БК-0010 оно равно 4,0 мкс. Итак, за 4,0 мкс выполняются операторы NOP, CLC, CLV, CLZ, CLN, CCC, SLC, SLV, SLZ, SLN, SCC, а также все 1- и 2-операндные команды (CLR, INC, ASL, ASR, MTPS, SWAB, MOV, CMP, XOR и т. п.), если способ адресации операндов в них «RN» (код «0»).

Сложнее определить время исполнения команд, если способы адресации операндов иные. Вспомним, как обозначались операнды при описании операторов. Буквой A обозначался операнд-источник, а B — операнд-приемник в двухоперандных командах (например, MOV A,B). Как AB обозначался источник и одновременно приемник в одноопераднных командах (например, COM AB). Как A1, A2 обозначались источник и приемник, если в процессе исполнения команды они не изменялись (например, CMP A1,A2); наконец, операнд — адрес перехода обозначался N. Прибегая к тем же обозначениям, приведем таблицу времени адресации операндов для БК-0010 (табл. 1).

Таблица 1

Адресация		Тип операнда и добавочное время, мкс				
способ	код	A,A1	AB	B	A2,N,	Ns
RN	0	0	0	0	0,	- -
@RN, (RN)+, -(RN), #X, #MET	1, 2, 4	4,0	5,4	6,8	6,8	10,8
@(RN)+, @-(RN), X(RN), MET(RN), @#X, MET	3, 5, 6	6,8	8,0	10,8	9,4	13,4
@X(RN), @MET(RN), @MET	7	9,4	10,8	13,4	12,0	16,0

Как пользоваться таблицей? Основное время для всех команд равно 4,0 мкс. Для вычисления времени исполнения одноопераднной команды к основному времени необходимо прибавить время адресации операнда из таблицы. Приведем примеры (табл. 2) и время исполнения ряда команд.

Таблица 2

Команда	Тип операнда, код адресации		Время исполнения, мкс (основное+добавочное)
TST R2	A1	0	4,0+0,0=4,0
TST @R4	A1	1	4,0+4,0=8,0
COM @R2	AB	1	4,0+5,4=9,4
MTPS #340	AB	2	4,0+5,4=9,4
CLR @#256	AB	3	4,0+8,0=12,0
INC @-(R4)	AB	5	4,0+8,0=12,0
ROR MET	AB	6	4,0+8,0=12,0
NEG @10(R5)	AB	7	4,0+10,8=14,8

С двухоперандными командами дело обстоит значительно сложнее. Для операнда-источника время зависит только от его типа (A или A1). А вот для операнда-приемника тип B принимается лишь тогда, когда источник — RN; в противном случае типом приемника считается AB. Для операторов же CMP и BIT, не меняющих операндов, время адресации второго операнда меньше: если источником является RN, для приемника берется тип A2, при ином же способе адресации источника тип приемника будет A1. У оператора XOR для второго операнда принимается тип A2. Все эти особенности определяются порядком, в котором процессор обрабатывает операнды на микропрограммном уровне. Примеры — в табл. 3.

Таблица 3

Команда	Тип операнда, код адресации		Время исполнения, мкс (основное+добавочное)
MOV R2,R4	A,B	0,0	4,0+0,0+0,0=4,0
MOV R2,@R4	A,B	0,1	4,0+0,0+6,8=10,8
MOV @R2,R4	A,AB	1,0	4,0+4,0+0,0=8,0
MOV @R2,@R4	A,AB	1,1	4,0+4,0+5,4=13,4
SUB @MET,R5	A,AB	7,0	4,0+9,4+0,0=13,4
ADD MET,MET	A,AB	6,6	4,0+6,8+8,0=18,8
CMP #100,R5	A,A1	2,0	4,0+4,0+0,0=8,0
CMP R5,#100	A,A2	0,2	4,0+0,0+6,8=10,8
BIT'A',@#256	A,A1	2,3	4,0+4,0+6,8=14,8
XOR R3,MET	A,A2	0,6	4,0+0,0+9,4=13,4

Отметим, что время исполнения команд на БК-0010 не зависит от того, работает команда с байтом или со словом.

Еще ряд операторов имеет время исполнения, которое должно быть оговорено особо. В табл. 4 приведено общее время исполнения некоторых команд.

Таблица 4

Оператор или группа операторов	Общее время исполнения, мкс
JMP N	Табл. 1, столбец Nj
JSR RN,N (или CALL N)	Табл. 1, столбец Ns
RTS RN (или RET)	10,8
BR MET, BEQ MET и все прочие операторы ветвления независимо от того, выполняется ли переход по условию	5,4
SOB RN,MET независимо от того, выполняется ли ветвление	6,8
EMT, TRAP, BPT, IOT	22,8
HALT	48,0
Прерывания от внешних устройств	Около 20 мкс плюс время завершения исполнения текущей команды
RTI, RTT	13,4
RESET	Около 380 мкс

Приведем также время обработки некоторых часто используемых прерываний EMT БК-0010, это представляет интерес с практической точки зрения (табл. 5).

Таблица 5

Прерывание	Общее время исполнения (ориентировочно)
EMT 4	0,2 мс
EMT 14	240 мс
EMT 16	От 0,4 мс (R0=0) до 1,7мс (в R0 код символа); с погашенным курсором вывод символа - 1,1 мс; в режиме ИНВ.С добавляется 100 мкс, в режиме ПОДЧ добавляется 20 мкс; в режиме "32 символа в строке" все значения возрастают примерно вдвое; сброс экрана - 200 мс; перевод строки - 1,8 мс (со сдвигом экрана - 18 мс)
EMT 20	(1,6xN)+0,5 мс; N - число выводимых символов; в спецрежимах - поправка на каждый символ, как для EMT 16
EMT 22	От 1,1 мс (вывод символа) до 34 мс (сброс строки); в спецрежимах - поправка как для EMT 16
EMT 24	1,8 мс в символьном режиме; 2,4 мс в графическом режиме
EMT 26	0,25 мс в символьном режиме; 0,27 мс в графическом режиме
EMT 30	0,5 мс независимо от кода операции
EMT 32	(0,5xN)+0,5 мс; N - число точек вектора; от кода операции не зависит; время несколько меняется в зависимости от направления вектора

Как работает команда MARK

Опубликованная в «ИНФО» серия статей Ю. Зальцмана по ассемблеру БК-0010 является хорошим пособием для работников нашего образования, которые чаще всего не имеют специальной подготовки в этой области. Одновременно эти материалы являются доступным для рядового учителя справочником по системе инструкций процессора К1801. Поэтому досадно, что в такой нужной и квалифицированной статье содержится ошибочное описание одной из машинных операций: речь идет о команде MARK. Она действительно «применяется крайне редко» и в популярной литературе «нигде не описана достаточно понятно» [1]. Но в то же время она «входит составной частью в стандартное соглашение ЭВМ PDP-11 о возврате из подпрограммы» и заметно «облегчает выполнение очистки стека при выходе из подпрограммы» [2]. Так что рассматривать команду MARK стоит внимательно и не спеша.

```
;ПОДГОТОВИТЕЛЬНЫЕ ОПЕРАЦИИ                    1
                        ; Записать в стек:
     MOV  R5, -(SP)     ;текущее значение
                        ;R5,
     MOV #101, -(SP)    ;параметры
     MOV #102, -(SP)    ;для подпрограммы
     MOV #103, -(SP)    ;SBR,
     MOV #6403,-(SP)    ;код команды MARK 3
     MOV  SP, R5        ; Адрес команды
                        ;MARK занести в R5
                        ;для обеспечения
                        ;перехода к ней
;ВЫЗОВ ПОДПРОГРАММЫ
     JSR PC, SBR
     ;...........       Текст дальнейшей
     ;...........       программы.

;ПОДПРОГРАММА SBR
SBR: ;...........       Текст
     ;...........       подпрограммы.
     RTS R5             ;Выход из SBR на
                        ;команду MARK
```


В [1] при изучении MARK обсуждается стандартный вариант ее применения (почти в том же виде он дается в [2]), поэтому приводимый ассемблерный фрагмент безусловно правилен и будет работать. К сожалению, описание к нему не всегда соответствует действительности, а выполнение рекомендаций статьи по использованию MARK может вызвать непредсказуемые последствия при работе программы.

Как же реализуется эта команда?

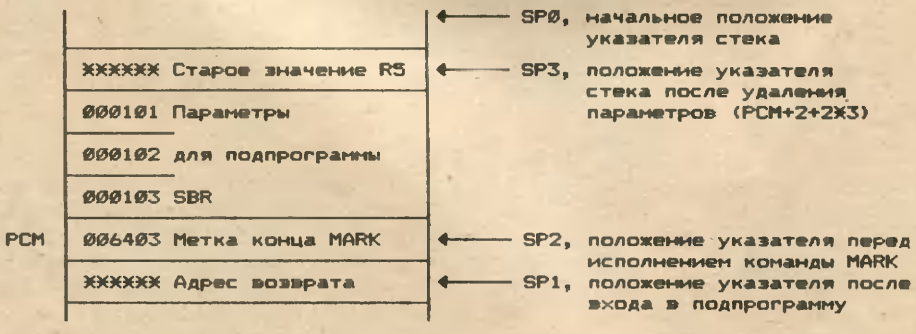
Рассмотрение будем вести на основе приведенного в [1] примера. Для удобства читателей воспроизведем его с незначительными изменениями обозначений и комментариев.

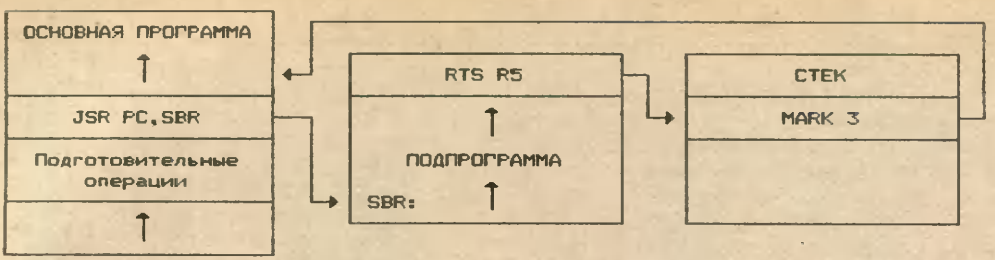
Предположим, что начальное значение указателя стека SP0 (рис. 1) уже определено. Прежде чем использовать команду MARK, необходимо провести некоторые подготовительные операции: запомнить в стеке значение регистра R5 (этот РОН аппаратно использует команда); занести в стек параметры для подпрограммы SBR и метку конца для случая трех параметров MARK 3; текущее положение SP (оно совпадает с адресом команды MARK и на рис. 1 обозначено PCM) скопировать в R5.

Теперь из основной программы следует вызов подпрограммы SBR (рис. 2). При этом текущее содержимое PC, т. е. адрес следующей за JSR PC команды, автоматически попадет в стек. Указатель принимает значение SP1 (см. рис. 1).

Выполнение подпрограммы завершается командой RTS R5 (а не RTS PC!), по которой процессор переписывает R5 в R7, а затем в R5 заносит из стека очередное значение. В рассматриваемом примере в PC попадет

2





подготовленный заранее в R5 адрес команды MARK, а в R5 — адрес возврата из подпрограммы. Новое положение указателя стека на рис. 1 обозначено SP2.

Теперь выполняется команда MARK 3, расположенная в стеке (не забывайте заглядывать на рис. 2). Это происходит следующим образом.

1. Вычисляется новое значение указателя стека SP3 по формуле (PCM+2)+2×N, где PCM+2 — текущее содержимое PC после выборки команды MARK, N — число параметров для подпрограммы. В результате стек освобождается от параметров и его указатель перемещается на исходное значение R5 (см. рис. 1). Обратите внимание на то, что величина SP3 вычисляется не через R6, а через программный счетчик R7. Из этого следует, что команда-метка MARK всегда должна быть расположена именно в стеке, а не внутри программы!

2. Содержимое R5 (в нашем случае адрес возврата) помещается в регистр PC.

3. В R5 выталкивается из стека первоначальное значение этого РОНа, указатель возвращается в исходное положение SP0.

Итак, в PC находится адрес следующей за обращением к подпрограмме команды, SP и R5 восстановлены. Возврат из подпрограммы завершен, стек очищен, и продолжается выполнение основной программы (см. рис. 2).

Обратите внимание на то, что описанные в п. 2 и 3 действия эквиваленты операции RTS R5, поэтому команду MARK можно рассматривать как сочетание своеобразной подстройки указателя стека (п. 1) и RTS R5.

Таким образом, описание команды MARK в [1] содержит две существенные неточности, которые в случае применения рекомендаций статьи приведут к потере управления программой. Во-первых, ошибочно утверждается, что регистр «под буфер адреса возврата» выбирается произвольно, в то время как процессор аппаратно может использовать для этой цели только R5. Если все же попытаться применить для хранения адреса другой РОН, на втором этапе команды MARK произойдет передача управления по не подготовленному для этого содержимому R5. Во-вторых, ни в коем случае нельзя «просто записать в тексте программы MARK X». Последствия этой ошибки будут еще более тяжелыми: помимо перехода по неопределенному адресу из R5 на первом этапе работы команды MARK указатель стека будет установлен на программу (см. формулу для SP3 на рис. 1).

Литература

1. *Зальцман Ю.* Ассемблер БК // Информатика и образование. 1991. № 2. С. 60—61.

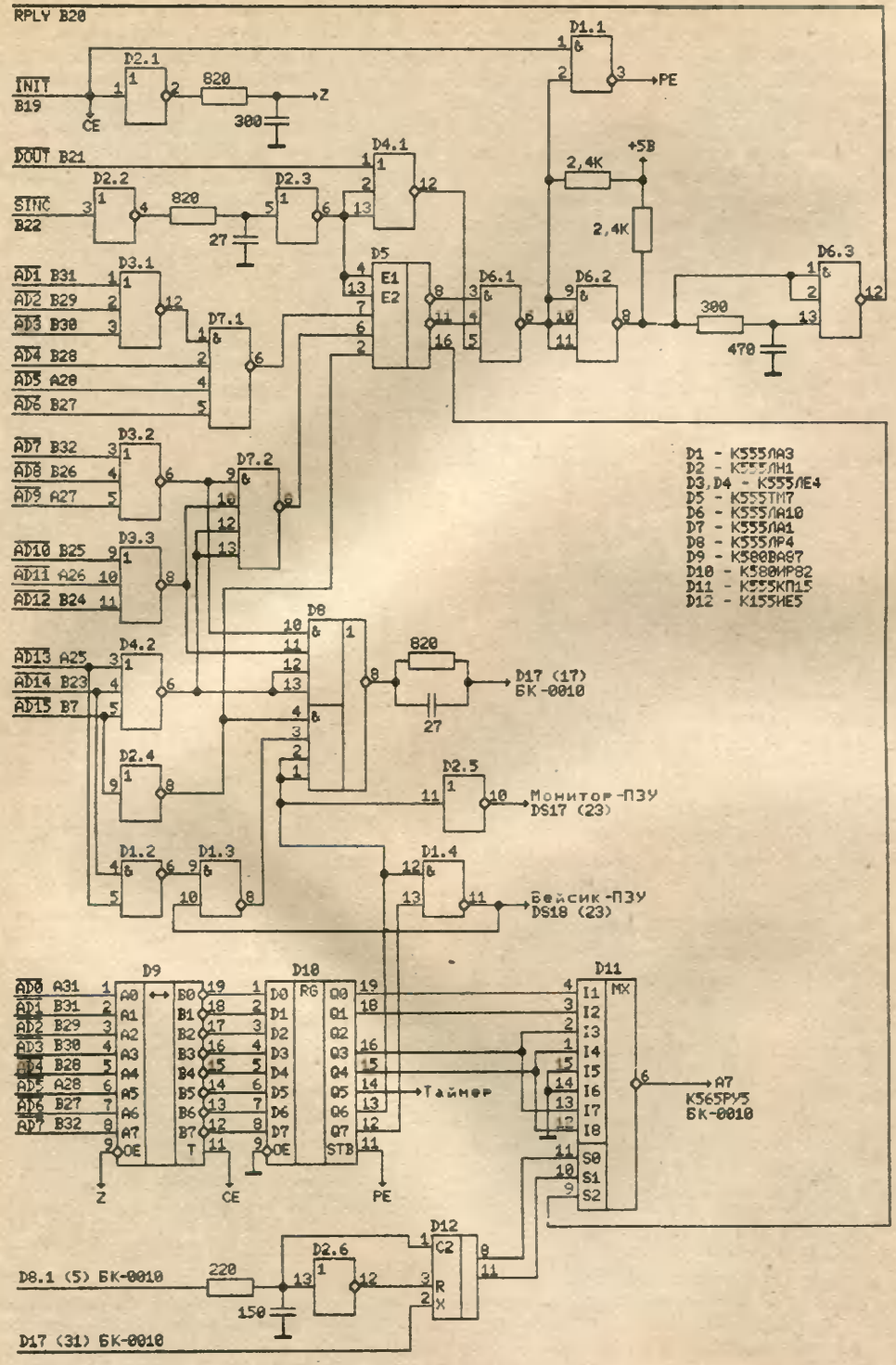
2. *Лин В.* PDP-11 — VAX-11. Архитектура ЭВМ и программирование на языке ассемблера. М.: Радио и связь, 1989.

Е. ЕРЕМИН

Диспетчер ОЗУ и операционная система RAMDOS для БК-128К

В этой статье речь пойдет о БК-0010 без ПЗУ, с дополнительными 96 Кбайт ОЗУ и схемой управления ОЗУ (диспетчером). ОЗУ организовано в четыре страницы по 32 Кбайт, часть этого объема может использоваться в качестве видеопамяти (максимум четыре страницы по 16 Кбайт). В качестве регистра состояния используется адрес 177616.

Диспетчер (рис. 1) представляет собой устройство, поддерживающее работу дополнительного ОЗУ БК-0010(01) со страничной организацией памяти — четыре страницы по 32 Кбайт, подключаемые по адресам 0 — 100000, причем верхние 16 Кбайт каждой страницы могут быть использованы в качестве видео-ОЗУ, обеспечивая многооконн-


RPLY B20
INIT B19
DOUT B21
SINC B22
AD1 B31
AD2 B29
AD3 B30
AD4 B28
AD5 A28
AD6 B27
AD7 B32
AD8 B26
AD9 A27
AD10 B25
AD11 A26
AD12 B24
AD13 A25
AD14 B23
AD15 B7
D1 - К555ЛА3
D2 - К555ЛН1
D3,D4 - К555ЛЕ4
D5 - К555ТМ7
D6 - К555ЛА10
D7 - К555ЛА1
D8 - К555ЛР4
D9 - К580ВА87
D10 - К580ИР82
D11 - К555КП15
D12 - К155ИЕ5
+5В
2,4К
820
300
27
470
150
220
D17 (17) БК-0010
Монитор-ПЗУ DS17 (23)
Бейсик-ПЗУ DS18 (23)
Таймер
A7 К565РУ5 БК-0010
D8.1 (5) БК-0010
D17 (31) БК-0010

ный режим работы. Однако следует иметь в виду, что активной является только текущая страница, т. е. подключенная в нижнюю область ОЗУ по адресам 0 — 100000; информация может быть записана только на нее. На отображение может быть подключена любая из четырех имеющихся страниц, что позволяет, наблюдая одну картинку на экране, готовить в это время другую.

При разработке программного обеспечения о переключении экранов программист должен позаботиться сам. Переключение осуществляется занесением информации в 3—4-й биты регистра состояния (адрес 177616). При включении компьютера пользователю доступны (по умолчанию) нулевая страница ОЗУ и нулевой экран.

Кроме того, дополнительная память может использоваться в качестве электронных дисков.

Регистр состояния доступен только по записи, возможность прочитать его отсутствует.

Назначение битов регистра состояния следующее:

0-й и 1-й — номер активной страницы ОЗУ;

3-й и 4-й — номер активной видеостраницы;

5-й — включение таймера;

6-й — отключение монитора;

7-й — отключение ПЗУ с Бейсиком.

При отключении ПЗУ на его место включается четвертая страница ОЗУ, расширяя тем самым ОЗУ пользователя до 64 Кбайт.

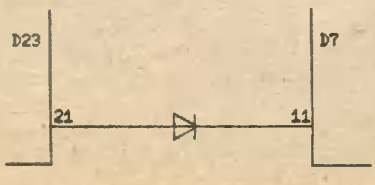
Адреса 177600 — 177776 использованы для системных регистров.

После включения БК распределение памяти стандартное, т. е. 32 Кбайт ОЗУ и 32 Кбайт ПЗУ.

Диспетчер размещается на печатной плате размером 80×95 мм внутри корпуса БК-0010(01). Печатная плата располагается на стойках в левой части БК, рядом с системным разъемом.

Питание на диспетчер подается от компьютера: +5 В — от А12 системного разъема, 0 — от А3 системного разъема.

Для обеспечения при включении питания пуска процессора с адреса 100000 необходимо припаять диод КД522 согласно схеме рис. 2 (номера микросхем приведены согласно принципиальной схеме БК).



Подключение диспетчера к БК производится следующим образом.

Отпаять вывод 17 микросхемы D17 на плате БК и соединить этот вывод с микросхемой К555ЛР4 диспетчера памяти.

Осторожно выпаять из платы 16 микросхем памяти К565РУ6 и на их место установить микросхемы К565РУ5(Б, В), предварительно проверив их на годность. Предварительно у микросхем К565РУ5 отогнуть выводы 9 и объединить их проводом, не впаивая в плату.

При пайке паяльник обязательно должен быть заземлен.

Отпаять вывод 23 микросхемы DS17 БК и в разрыв впаять резистор МЛТ-0,125 1 кОм. Эту же операцию проделать и с другими ПЗУ.

6. Подсоединить контроллер к БК согласно табл. 1.

Таблица 1

Диспетчер	Выводы микросхемы DS19
AD0	16
AD1	15
AD2	14
AD3	13
AD4	4
AD5	5
AD6	6
AD7	7
AD8	8
AD9	9
AD10	10
AD11	11
AD12	12
AD13	18
AD14	19
AD15	20

Таблица 2

Диспетчер	БК
RPLY	Вывод 13 D3.2
INT	В19 системного разъема
DOUT	В21 системного разъема
SYN	В22 системного разъема
A7	Выводы 9 DS1—DS16
MON (монитор)	Вывод 23 DS17
ПЗУ (Бейсик)	Вывод 23 DS19
AD15 ВП37	Вывод 17 D17
+5 В	А12 системного разъема
0	А2 системного разъема

Правильно собранная схема не нуждается в наладке и начинает работать сразу после включения.

Микросхемы серии К555 могут быть заменены на микросхемы серии К155, однако при этом увеличится потребляемый от ис-

точника питания ток и возрастет нагрузка на системный разъем. Микросхема К555КП15 может быть заменена на К155КП7, К580ИР82 — на К555ИР22. Резисторы — МЛТ 0,125 или 0,25, конденсаторы керамические типа КМ-5.

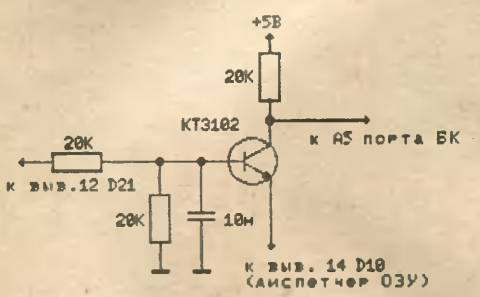
БК имеет вход прерывания таймера. Принцип работы таймера весьма прост: при поступлении на вход процессора ПРТ сигнала низкого уровня тот прекращает выполнение основной программы, запоминает в стеке слово состояния и адрес следующей команды и переходит на обработку прерывания по вектору 100, т. е. по адресу, который записан в ячейку памяти 100. После завершения программы обработки прерывания процессор возобновляет выполнение основной программы.

Плата диспетчера ОЗУ позволяет организовать такой таймер. Им можно управлять программным путем, записывая информацию в 5-й бит регистра состояния, т. е. по адресу 177616.

Нулевое значение соответствует включенному таймеру.

Для организации таймера необходимо на плате диспетчера ОЗУ собрать небольшой узел, схема которого приведена на рис. 3.

3



В качестве временных меток можно использовать сигнал синхронизации видеосигнала, предварительно выделив из него кадровые импульсы, после чего подать их на вход процессора ПРТ, усилив по амплитуде.

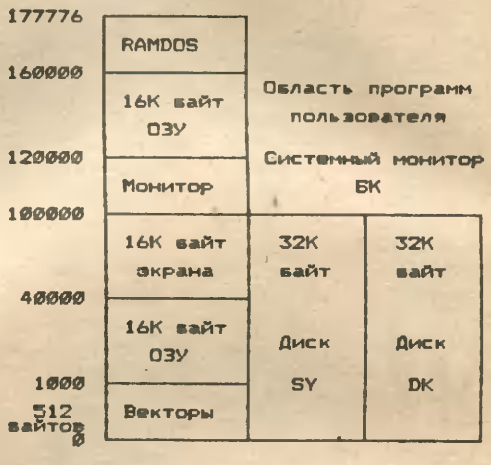
Управление таймером производится по цепи эмиттера, который связан с выводом ТАЙМЕР микросхемы D10 диспетчера ОЗУ.

Для поддержки работы диспетчера памяти разработана ОС RAMDOS. Мнемоника ее команд аналогична операционной системе RT-11.

Среди достоинств RAMDOS прежде всего возможность быстрого обмена информацией между электронными дисками и ОЗУ, что особенно важно при работе с системными программами, например ассемблером, когда программа состоит из модулей и приходится то и дело записывать или читать файлы с магнитофона, или при работе с большими текстами, когда документ имеет многофайловую структуру. Для тех, кто любит играть в многосерийные игры и знает, сколько времени занимает ввод очередной серии, система RAMDOS также будет очень полезна, поскольку считывание программ с электронного диска занимает доли секунды.

RAMDOS маскирует все прерывания, в том числе и таймерные, обрабатывая их соответствующим образом.

Распределение адресного пространства



Изменять информацию в адресном пространстве RAMDOS — начиная с адреса 160000 и выше — категорически запрещается: это может разрушить операционную систему.

Ниже приводится описание операторов.

QUIT — выход из системы в монитор БК с восстановлением таблиц и векторов.

EXIT — выход из системы в монитор БК с возможностью возвращения в систему по СТОП, программы на электронных дисках сохраняются.

RESET — перезапуск RAMDOS, восстановление таблиц системы и векторов прерываний, содержимое электронных дисков сохраняется.

LOAD [DEV:]NAME — чтение программы с магнитофона под именем NAME и занесение ее на электронный диск; DEV: — имя устройства, на которое будет записан файл. По умолчанию файл записывается на устройство с логическим именем DK, при наличии на нем файла с указанным именем запись не производится и выдается диагностическое сообщение. Квадратные скобки здесь и далее обозначают необязательный параметр.

SAVE [DEV:]NAME — запись программы NAME с устройства с логическим име-

нем DEV на магнитофон под тем же именем. При отсутствии файла с указанным именем выдается диагностическое сообщение.

ASSIGN DEV:DEVN: — назначение указанному устройству нового логического имени. По умолчанию в системе зарезервированы имена: системный диск — LD0, рабочий диск — LD1. Примеры:

ASSIGN MX0: SY: — теперь системный электронный диск имеет имя MX0:, ASSIGN DK: SY: — системный диск становится рабочим.

DELETE [DEV:]NAME — удаление файла с именем NAME с указанного устройства; при отсутствии файла выдается диагностическое сообщение. Все операции по удалению требуют подтверждения, поэтому на запрос: "Are you sure?" необходимо ввести Y для подтверждения и нажать ВК. Пример: DELETE SY:FOCAL — удалить с системного диска файл с именем FOCAL.

PRINT [DEV:]NAME — выдать содержимое файла с именем NAME на печатающее устройство. Драйвер печати позволяет работать с принтерами Д100, МС6312 и др. Для нормальной работы принтера необходимо контакт А30 порта БК замкнуть на контакт В11. Распайка кабеля приведена в табл. 3.

Таблица 3

Порт БК	Принтер
В31 (ВВ08)	Готовность
А16 (ВД00)	Данные
А13 (ВД01)	»
В12 (ВД02)	»
В10 (ВД03)	»
В05 (ВД04)	»
В07 (ВД05)	»
В06 (ВД06)	»
А07 (ВД07)	»
А28 (ВД08)	Строб
В11 (Общ.)	Общий
А30 (ВВ15)	»

INIT DEV: — инициализация каталога тома DEV:. Перед инициализацией выдается запрос на подтверждение, при положительном ответе указанный том очищается.

RENAME [DEV:]NAME NAME1 — переименование файла на указанном устройстве. Пример: RENAME RAMDOS DOS — файл RAMDOS, находящийся на диске DK:, будет переименован в DOS.

COPY N,K,A — копирование области памяти с адреса N по K в область, начинающуюся с адреса A. Команда полезна при с выдачей адреса загрузки и длины программы, автозапуск блокируется.

RUN [DEV:]NAME — загрузка и запуск программы под именем NAME с указанного устройства. Программы, имеющие автозапуск, запускаются по установленному в них адресу.

E N[—K] — просмотр на терминале содержимого ячеек памяти в восьмиричном виде. Пример: E 1000—2000 — вывод на терминал значений ячеек памяти с адреса 1000 по 2000.

D A=W[,W...] — запись в память информации, расположенной с адреса A. Попытка записать данные в область RAMDOS блокируется, на терминал выдается предупреждение. Пример: D 1000=12700,101,104016, 137,1000.

Загрузка RAMDOS производится из монитора следующим образом (курсивные символы выводятся компьютером):

```
? M<ВК>
имя ? RAMDOS.WS<ВК>
? S<ВК>
```

После запуска RAMDOS производит самоинициализацию и перестройку монитора, экран очищается и на него выводится заставка системы. При нажатии любой клавиши происходит переход в RAMDOS, о чем свидетельствует надпись в служебной строке и знак «точка» — признак нахождения в редакторе командной строки.

Программа ожидает действий пользователя.

Команда SHOW подсказывает формат команд RAMDOS.

```
.SHOW<ВК>
```

Система распечатает список и формат команд.

Теперь в распоряжении пользователя перемещении программ в другую область памяти перед запуском.

TYPE [DEV:]NAME — выдать содержимое файла под именем NAME на терминал в символьном представлении (например, для просмотра). Приостанов текста для просмотра — клавиши СУ+@, их повторное нажатие — возобновление продвижения по экрану текстовой информации.

DIR [DEV:] — просмотр каталога указанного тома на терминале с выдачей числа файлов, объемов занятой и свободной областей памяти.

START A — запуск программы с указанного адреса.

GET [DEV:]NAME — загрузка программы под именем NAME с указанного устройства в память без запуска на исполнение,

имеется два электронных диска общим объемом 64 Кбайт и 32 Кбайт свободной оперативной памяти при полном графическом экране.

Но диски пока пусты, в этом можно убедиться, «заглянув» в них с помощью команд

```
.DIR<ВК>
.DIR SY:<ВК>
```

Их можно заполнить, загрузив с магнитофона нужные программы. Для этого необходимо подать команду

```
.LOAD SY:ODT3.WS<ВК>
```

Она введет с магнитофона и загрузит на системный диск файл ODT3.WS. Посмотрите теперь каталог,

```
.DIR SY:<ВК>
```

и вы убедитесь, что программа занесена в дисковую память.

Таким же образом загрузите остальные необходимые вам файлы.

Теперь можно приступить к работе. Например, вы хотите редактировать текстовые документы. Для этого необходимо запустить ранее записанный на диск текстовый редактор.

```
.RUN EDASP1Ø<ВК>
```

По этой команде произойдет загрузка и запуск редактора.

Создав текстовый документ, вы можете сохранить его на указанном вами диске, причем практически мгновенно. По умолчанию запись производится на диск DK:.

При повторной записи текста старый файл на диске замещается новым (при условии совпадения имен).

Выходы из программ организованы поразному. Очень часто программы не восстанавливают векторы прерываний, которые были до их запуска, поэтому, если вы при выходе из программы вдруг оказались в мониторе, следует дать команду

```
T<ВК>
```

или (если не был нарушен вектор 4) нажать клавишу **СТОП**.

Для аварийного выхода из программы и перехода в RAMDOS необходимо двукратное нажатие клавиши **КТ** — такая процедура выбрана потому, что некоторые пользовательские программы используют эту клавишу.

При выходе в систему по этой команде содержимое дисков сохраняется.

Редактор командной строки запоминает последнюю набранную команду. Ее повторный вызов производится нажатием клавиши **ВС**.

Программы с автозапуском в рамках RAMDOS работают корректно.

Команды QUIT, EXIT, DELETE, INIT перед выполнением требуют подтверждения — например, при инициализации каталога диска LD0 произойдет следующий диалог:

```
.INIT LDØ:<ВК>
Are you sure? Y<ВК>
```

И только по команде пользователя диск будет очищен.

В. КОРЕНКОВ

Дополнительные сведения о системном таймере

О системном таймере БК-0010 уже сообщалось в «ИНФО» [1, 2]. В данной статье приводятся некоторые дополнительные сведения о нем.

Таймер состоит из следующих узлов:

генератор с периодом 42,9 мкс;

делители на 4 и 16;

реверсивный счетчик (ячейка 177710) с коэффициентом деления до 65536 (загружается из регистра уставки в момент пуска и перехода через 0);

регистр установки (ячейка 177706);

регистр команд и состояний РКС (ячейка 177712).

Управляют работой таймера с помощью РКС путем установки соответствующих битов. РКС представляет собой шестнадцатиразрядный регистр ввода-вывода, в котором доступны разряды 0—7 (разряды 8—15 не используются и всегда установлены в единицу). Функции разрядов РКС следующие:

бит 0 — назначение неизвестно (установка в 1 прекращает счет);

бит 1 — назначение неизвестно (на счет не влияет);

бит 2 — разрешение установки сигнала «конец счета»;

бит 3 — режим одновибратора (по окон-

чании счета сбрасывает бит 4 и прекращает работу счетчика);

бит 4 — запускает счетчик (при чтении индицирует состояние счетчика: 0 — стоит, 1 — считает);

бит 5 — включает делитель на 16;

бит 6 — включает делитель на 4;

бит 7 — указывает окончание счета (готовность) (в зависимости от режима счетчик либо останавливается, либо продолжает считать).

Приведем несколько примеров использования таймера.

Пример 1: работа в режиме одновибратора.

```
;Генератор ноты
;R0 - тон
;R1 - длительность * 42.9 мкс
BEEP:   MOV   R2,-(SP)        ;Сохранить R2
        MOV   R1,@#177706     ;Установка =
                              ;длительность
        MOV   #34,@#177712    ;Запуск
                              ;одновибра-
                              ;тора
LOOP1:  MOV   R0,R2           ;1/2 периода
        BEQ   PAUSE           ;Если тон=0,
                              ;то пауза
        MOV   #320,@#177716   ;Вкл. звук
LOOP2:  SOB   R2,LOOP2        ;Положитель-
                              ;ный полу-
                              ;период
        MOV   R0,R2           ;1/2 периода
        MOV   #220,@#177716   ;Выкл. звук
LOOP3:  SOB   R2,LOOP3        ;Отрицатель-
                              ;ный полу-
                              ;период
RAUSE:  TSTB  @#177712        ;Конец счета
                              ;длительности
        BPL   LOOP1           ;След. период
        MOV   (SP)+,R2        ;Восстановить
                              ;R2
        RTS   PC
```

Пример 2: программа с заданным временем исполнения.

```
        .                     ;Подгото-
        .                     ;вительные
        .                     ;действия
GAME:   MOV   #2331.,@#177706 ;10 раз в
        MOV   #24,@#177712    ;секунду
                              ;запуск
                              ;таймера
        .                     ;Основной
        .                     ;цикл игры,
        .                     ;длитель-
        .                     ;ность
        .                     ;переменная
        .                     ;T<0.1 сек
WAIT:   TSTB  @#177712        ;Конец
                              ;цикла
                              ;0,1 сек?
        BPL   WAIT            ;Ожидать
                              ;конца
        JMP   GAME            ;Следующий
                              ;цикл
```

Пример 3: фиксированная задержка, счетчик продолжает считать.

```
        .
        .
        MOV   #DELAY,@#177706 ;Величина
                              ;задержки
        MOV   #24,@#177712    ;Запуск
WAIT:   TSTB  @#177712        ;Ожидание,
        BPL   WAIT            ;пока
                              ;конец
                              ;счета <>1
        .   ;Содержимое 177710 можно
        .   ;использовать для коррекции
        .   ;времени, бит 3 = 1
```

Пример 4: фиксированная задержка, счетчик останавливается.

```
        .
        .
        MOV   #DELAY,@#177706 ;Величина
                              ;задержки
        MOV   #34,@#177712    ;Запуск
WAIT:   TSTB  @#177712        ;Ожидание,
        BPL   WAIT            ;пока
                              ;конец
                              ;счета <>1
        .                     ;Счетчик=
        .                     ;=DELAY
        .                     ;Бит 3 = 0
        .
```

Из примеров видно, что использование бита 7 гораздо удобнее, чем попытка определения перехода счетчика через ноль. Особенно это заметно в интерпретаторах Бейсика и Фокала.

Если кому-либо известны более подробные сведения о таймере, прошу сообщить их мне или в редакцию «ИНФО». Заранее благодарен.

202020, Эстония, Йыхви, п/я 69.
А. Подгорецкий

Литература

1. *Яковлев В.* Отсчет времени на БК-0010.01 // Информатика и образование. 1990. № 1.

2. *Зальцман Ю.* Архитектура и ассемблер БК-0010 // Информатика и образование. 1990. № 6.

Многоформатный верификатор-каталогизатор

За последние годы ассортимент домашних компьютеров, доступных рядовому пользователю, значительно вырос, однако интерес к БК-0010 не упал ни у покупателей, ни у пользователей, ни у программистов; свидетельством тому является дефицитность этого компьютера и все возрастающее количество разнообразного программного обеспечения для него.

Для систематизации, каталогизации и проверки качества записи программ для БК на магнитной ленте (МЛ) имеется ряд специальных программ, лучшая из которых — VERIFY 2.0 написана А. Тереховским (см.: ИНФО. 1990. № 3). Она имеет достаточно широкий набор функций и удобный интерфейс.

Однако сегодня этот верификатор-каталогизатор уже не может считаться универсальным хотя бы потому, что он не воспринимает записей файлов, сделанных методом модифицированной фазовой модуляции (МФМ). Этот очень перспективный метод записи файлов на МЛ предложен недавно А. Молчановым (Ленинград), причем сразу в двух модификациях — HELP4S и HELP7, и за короткое время получил широкое распространение из-за обеспечиваемой им почти абсолютной надежности считывания записей с МЛ даже сильно изношенными и разъюстированными головками низкокласс-ных магнитофонов.

В последнее время были сделаны попытки объединения штатного метода записи и одной из модификаций метода МФМ в одном копировщике — MAGIA. Нет сомнений, что вскоре появятся и новые капировщики, еще более универсальные. Таким образом, явно наметилась тенденция расширения форматов записей файлов на МЛ, а в связи с этим возникла и необходимость создания нового, многоформатного, верификатора-каталогизатора, усугубленная тем, что новый формат МФМ, помимо уже отмеченных достоинств, идеально подходит для создания очень плотных файловых архивов на МЛ. Между тем хорошо известно, что никакой архив без каталога обойтись не может.

Одновременно необходимо учитывать, что имеется и несколько текстовых редакторов, в формате которых хотелось бы записывать результаты проверки (верификации) файлов. Здесь особо следует отметить редактор, входящий в ассемблер-систему МИКРО.10К С. Кумандина (г. Коломна) и получивший самое широкое распространение среди профессионалов — пользователей БК. Следует упомянуть и конкурирующую с ним систему А. Сомова, ну и, конечно, ставший уже классическим текстовой редактор EDASP. Этими тремя подходами к созданию текстовых файлов вполне можно ограничиться сегодня, но никто не поручится за завтра!..

Именно по этим соображениям автор не рискнул назвать свой верификатор-каталогизатор «универсальным», хотя по универсализму он намного опережает уже упоминавшийся VERIFY2.0.

Верификатор, получивший название VERIFY PLUS (VERIFY+), вобрал в себя весь накопленный опыт в области верификации записей на МЛ, а также ряд новшеств: алгоритм верификации файлов, записанных на МЛ в трех форматах (стандартный, Н4, Н7), алгоритм защиты процесса верификации от систематических помех (вероятность распознавания полезного сигнала доведена до 80 %), алгоритм защиты от сбоев при длине тестируемого файла больше предельно допустимой (для стандартного и Н4 форматов — $40\,000_8$, для Н7 длина не ограничена), процедуру просмотра результатов верификации на экране монитора с возможностью перемещения по списку в обе стороны, просмотр списка сбойных файлов, встроенный редактор текущей строки списка, вывод результатов верификации на печать принтерами разных марок и запись каталожного файла на МЛ в различных форматах.

Программа может быть загружена и запущена с любого четного адреса, причем адрес «теплого» старта всегда на 2 больше адреса загрузки. Буфер программы располагается за ней, если адрес конца программы меньше $37\,000_8$, в противном случае буфер располагается с адреса 1000_8, что позволяет загружать VERIFY+ в область расширения ОЗУ (зашивка в ПЗУ недопустима!). После запуска программы на экран выводится ее обложка, а при нажатии клавиши пробела происходит переход в главное меню. Вызов первого пункта меню «VERIFICATION» запускает процедуру проверки файлов на МЛ. После прочтения каждого файла выводится информация о его имени, адресе, длине, контрольной сумме или наличии ошибки. Если длина файла превышает допустимый предел, то чтение тела файла не производится, а вместо контрольной суммы (КС) выводится признак «long». Все атрибуты файла заносятся в буфер.

При вызове второго пункта меню — «NAMELIST» — чтение тела файла не производится. Этим режимом следует пользоваться для создания каталога кассеты без проверки

качества записи: в нем меньше вероятность потери имен из-за сбоя при чтении. Выход из обоих режимов в главное меню по клавише СТОП, указатель режима меню устанавливается при этом на пункт «FINDINGS» (при ошибках в считанных файлах) или «LIST» (при их отсутствии). При повторном вызове режимов «VERIFICATION» или «NAMELIST» выводится запрос на начало нового списка. При положительном ответе буфер очищается и открывается новый список, в противном случае новые считанные имена будут подстыкованы к уже имеющимся в буфере. Одновременно метка границы (см. ниже) будет установлена на первое имя, считанное после повторного входа в режим.

Режим «NULL READING» предназначен для нахождения нужного участка на ленте для верификации или записи результатов. Имена, считанные в этом режиме, в буфер не заносятся.

Режим «FINDINGS» предназначен для вывода на экран списка файлов, в которых обнаружены ошибки. Если список не умещается на экране, то его вывод приостанавливается до нажатия любой клавиши. При отсутствии сбойных файлов режим блокируется.

В режиме «LIST» удобно просматривать список атрибутов считанных файлов. Выводится информация об имени, адресе, длине и КС. При наличии ошибки вместо КС выводится сообщение ERROR.

Если чтение тела файла не производилось (в режиме «NAMELIST» или при превышении по длине), то вместо КС выводится сообщение UNKNOWN. Если список не умещается на одном экране, то его можно просматривать в обоих направлениях клавишами ↑ и ↓, кроме того, с помощью клавиш СБР и КТ можно перейти к началу и концу списка соответственно. Выход в главное меню — по клавише ВВОД.

Режим «UNLOADING» открывает окно вспомогательного меню и окно параметров. Окно параметров содержит три строки: TYPE, FORMAT, PRINTER. В TYPE можно задать вид файла выходной информации. Первый вариант — STATEMENT — протокол верификации: каждому файлу отведена отдельная строка, выводится полный объем информации. Предполагается следующая структура каталога: ⟨НОМЕР⟩ ⟨ИМЯ⟩ ⟨АДРЕС⟩ ⟨ДЛИНА⟩ ⟨АВТОР⟩ ⟨ГОД НАПИСАНИЯ⟩ ⟨ПРИМ.⟩. Второй вариант — CATALOGUE, каталог файлов: файлы с одинаковыми именами объединяются, длина имени урезается до 14—15 символов (в зависимости от адреса), кроме имени выводятся только адрес и длина. Четыре первые графы такой таблицы автоматически создаются каталогизатором, а три последние можно варьировать.

Третий вариант — TAPE COVER, обложка кассеты. Он предназначен только для вывода на печать с помощью принтеров типа «Электроника МС 6312» и «Электроника МС 6313». Перед печатью необходимо установить метку на первое имя второй стороны кассеты (см. ниже). Если после проверки первой стороны выйти в меню, а затем снова войти в режимы «VERIFICATION» или «NAME LIST», то метка установится автоматически.

Параметр FORMAT задает формат записи файла: EDASP — формат редактора EDASP; MICRO# K — формат редактора МИКРО.10К, а также всех редакторов, использующих «естественный» формат кодирования (кодирование пробелов отсутствует, код перевода строки — 12_8); MICRO# S — формат редактора ассемблер-системы MICRO (Сомов — Бакерин). Последний формат очень интересен для табличных форм представления информации.

PRINTER — выбор типа принтера: MS6312 — «Электроника МС6312»; MS6313 — «Электроника МС 6313», который может быть подключен как с инверторами по цепям STROBE и BUSY, так и без них (дополнительный выбор при этом не требуется); ANOTHER — любой другой принтер, подключенный по стандарту ИРПС в соответствии с паспортом БК (печать обложки кассеты в этом случае не гарантируется).

Меню режима «UNLOADING» состоит из следующих пунктов: SAVE — запись результирующего файла на МЛ в установленном формате, имя запрашивается, записываются две копии; PRINT — печать результирующего файла на принтере в установленном формате на выбранном типе принтера; TYPE, FORMAT, PRINTER — изменение параметров, описанных выше; MAIN MENU — выход в главное меню.

Режим «CORRECTION» основного меню предназначен для редактирования текущей строки каталожного списка. Текущая строка списка инвертируется на экране, ее номер и КС выводятся в служебной строке. Коды, не имеющие графического отображения, выводятся в виде «забоев». Узнать, какой код за ними кроется, можно с помощью клавиши ИНД.СУ. При этом код символа, выделенного курсором, отображается в служебной строке. Редактор позволяет вводить большинство управляющих кодов по регистру СУ, а коды 0 и 32 клавишами ШАГ и СБР соответственно. Коррекция КС осуществляется по клавише БЛОК.РЕД. При этом новое значение КС запрашивается

и после ввода шести символов или нажатия клавиши ВВОД отображается в служебной строке. Метка устанавливается нажатием СУ/М. Отмеченный файл будет печататься первым в списке второй стороны кассеты при печати ее обложки. Удаление строки производится клавишей ВС, при этом делается запрос на подтверждение. Последняя оставшаяся строка списка не удаляется. Выход из редактора по клавише КТ.

Во всех режимах работы верификатора включен режим 32 символа в строке, что позволяет использовать цветной монитор. Выбор цветов осуществляется с помощью пункта COLOR основного меню.

Режим «EXIT» осуществляет выход в монитор БК. При этом запрашивается подтверждение. Возврат в верификатор без разрушения информации в буфере возможен по адресу «теплого» старта, о чем уже говорилось выше.

С. КАРАГИОЗ

О «дребезге» клавиатуры БК—0010

Самым неприятным дефектом БК-0010.01 является выскакивание «чужого» символа при нажатии на какую-либо клавишу. Большинство пользователей винят в этом исключительно дребезг контактов и пытаются как-то исправить их либо увеличивают времязадающие цепи R3-C4 и R4-C3 (здесь и далее все обозначения соответствуют схеме БК). Причина дефекта, однако, глубже: разработчики компьютера механически перенесли типовую схему включения микросхемы К1801ВП1-014 на свою конструкцию, не учтя значительной паразитной емкости строк и столбцов матрицы клавиатуры. Паразитный заряд, не успевший исчезнуть после нажатия предыдущей клавиши, является причиной неверной идентификации следующей. Усугубляет ситуацию и то, что контакты работают в режиме микротоков, не разрешенном в технических условиях на кнопки.

Исправить ситуацию чрезвычайно просто — достаточно ускорить стекание паразитного заряда и увеличить ток через контакты клавиатуры, для чего заменить резисторную матрицу Е3 (7 резисторов по 180 кОм) на аналогичную с сопротивлениями резисторов по 4,7—5,6 кОм.

Поскольку замена матрицы трудоемка и опасна (плата БК чувствительна к статическому электричеству), предлагается доработку произвести несколько другим способом.

Нужно отсоединить разъемы XT1 и XT2 клавиатуры от платы БК, пометив их, чтобы потом не перепутать; расположить клавиатуру клавишами вниз и найти место распайки ленточного провода, идущего к разъему XT2 (на плате клавиатуры — метка XT2). Семь резисторов с номиналом 4,7—5,6 кОм, 0,125—0,25 Вт припаять к точкам 1—7 распайки ленточного провода на плате так, чтобы их выводы располагались вплотную к плате, а корпуса находились за пределами платы (это облегчит последующую сборку). Между выводом резистора 1 и проводником печатной платы проложить кусочек изоленты.

Осталось объединить свободные выводы резисторов с помощью жесткого провода и припаять его второй конец к точке 8 на плате, подсоединить разъемы XT1 и XT2 к плате БК и проверить работу клавиатуры.

Если стопроцентное избавление от «чужих» символов не достигнуто, можно точно так же уменьшить величины резисторов матрицы Е2 (9 резисторов по 22 кОм), припаяв резисторы с номиналом 4,7—10 кОм к точкам 1—4, 6, 7, 9—11 (XT1), а их свободные выводы соединив с точкой 16. Не забудьте подложить изоленту там, где это необходимо.

Борис Фролкин

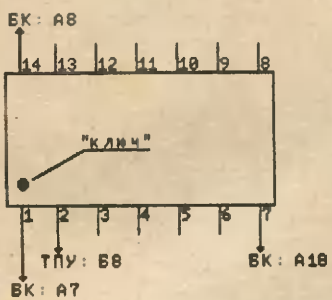
От редакции. Еще не купивших БК-0010.01 спешим обрадовать: теперь завод «Экситон» комплектует эти компьютеры (а также БК-0011) клавиатурой нового типа, более эргономичной и не подверженной «дребезгу». Покупайте БК!

Подключение термопринтера 15ВВП80-002 к БК

ТПУ 15ВВП80-002 принимает 7-битный код и печатает символы согласно таблицам КОИ-7_0 и КОИ-7_1. Первая включает цифры, латинские буквы (строчные и прописные) и специальные знаки, вторая — цифры, русские буквы (строчные и прописные) и специальные знаки. Выбор таблиц осуществляется ТПУ в соответствии с сигналом СИ4 (позиция). Его логический уровень (0 или 1) определяет используемую таблицу. Логический уровень сигнала СИ3 (форма) должен быть равен нулю. Небольшое аппаратное дополнение позволяет использовать 15ВВП80-002 без специального драйвера, непосредственно с вильнюсским Бейсиком или EDASP10.1. Достаточно инвертировать сигнал ВД07 (контакт А7 разъема МПИ БК) и соединить БК и ТПУ в соответствии с таблицей.

Разъем ТПУ		Разъем БК		
Б	Сигнал	А	В	Сигнал
4	ГИ—ИБС	А18	В18	Общий
6	СТР—ИБС	А28		ВД08
7	ЗП—ТПУ		В31	ВВ08
8	СИ4—ИБС	А7		ВД07
9	Д0—ИБС	А16		ВД00
10	Д1—ИБС	А13		ВД01
11	Д2—ИБС		В12	ВД02
12	Д3—ИБС		В10	ВД03
13	Д4—ИБС		В5	ВД04
14	Д5—ИБС		В7	ВД05
15	Д6—ИБС		В6	ВД06
18	$СИ_э$—ИБС	А18	В18	Общий
23	ОВ	А18	В18	Общий
		А8	В8	+5 В

Примечание. Сигнал ВД7 подается на ТПУ не непосредственно, а после инвертирования.



А теперь об инвертировании сигнала ВД07. Оно производится очень просто, с помощью микросхемы 155ЛН1, размещенной в корпусе вилки разъема ТПУ и распаянной, как показано на рисунке.

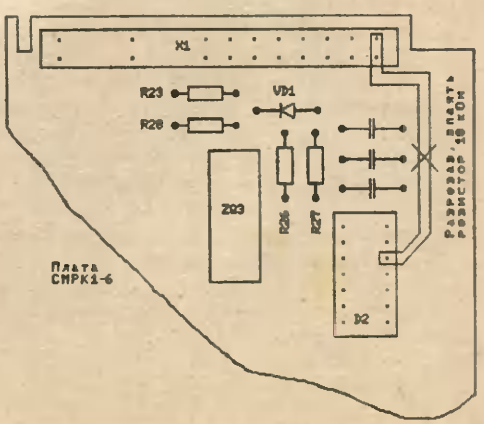
В. КОШЕЛЕВ

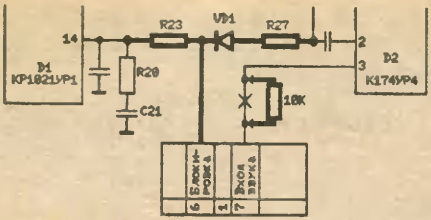
4УСЦТ — монитор

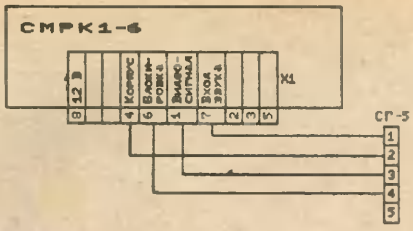
Цветные телевизоры практически невозможно использовать для отображения текста в режиме 64 символа в строке. Однако, если совершить небольшую переделку, телевизор можно будет переключать в режим «монитор», т. е. подавать сигнал с компьютера, непосредственно на субмодуль декодера. А так как с компьютера поступает черно-белый телевизионный сигнал, то телевизор автоматически перейдет в режим черно-белого изображения, как и при приеме черно-белых передач.

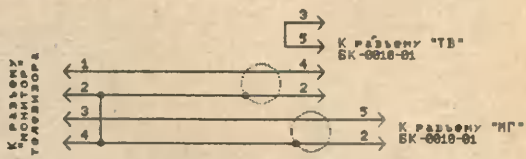
Для переделки в субмодуль радиоканала СМРК1-6 впаивают R23 (33 кОм), R27 (10 кОм), VD1 (Д220); места для них предусмотрены на печатной плате. Необходимо также впаять резистор с сопротивлением 10 кОм между контактом 1 разъема X1 и выводом 3 микросхемы D2. Это можно сделать на печатной плате, предварительно разрезав соответствующую токоведущую дорожку.

На фрагменте принципиальной схемы телевизора новые детали выделены утолщенными линиями, на рисунке печатной платы показано их расположение.









Затем нужно распаять разъем СГ-5 и установить его вместо антенного гнезда ДМВ или XS1.

Последняя задача — изготовление кабеля.

Теперь, соединяя контакт 6 разъема X1 через контакты 2 и 4 гнезда СГ-5 «монитор» с корпусом, можно заблокировать прохождение телевизионного сигнала и сигнала звукового сопровождения на субмодуль декодера.

Используя эту доработку вместе с опубликованной в «ИНФО» (1990. № 6. С. 72) и соединив БК с телевизором двумя кабелями, можно, работая в ассемблере в черно-белом варианте, нажатием кнопки S1 (см.: «ИНФО» 1990. № 6) перейти в режим «цвет» и посмотреть, как функционирует только что написанная программа.

Кстати, к новому разъему «монитор» можно подключить и видеомагнитофон, имеющий отдельные выходы Video и Audio. Необходимо только изготовить соответствующий кабель.

Ю. ЛУЦКЕВИЧ

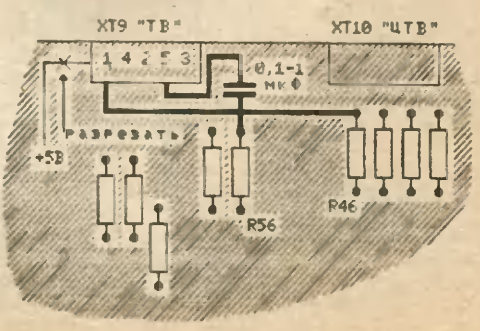
Цветной монитор как черно-белый

Большинство пользователей БК-0010 считают, что хорошее качество текста в режиме 64 символа в строке при использовании цветного монитора получить трудно. Мы же применяем способ, который позволяет достичь «читаемости», практически не уступающей черно-белым мониторам.

В БК есть два выхода телесигнала — для черно-белого (ТВ) и цветного (ЦТВ) мониторов. Если снять видеосигнал с выхода ТВ и подать его на G-линию шнура, подключенного к цветному телевизору, а R- и B-линии отключить от разъема ЦТВ, цветной телевизор превратится в черно-белый (точнее, черно-зеленый). Совсем отключить шнур от разъема ЦТВ нельзя: по нему идет сигнал синхронизации.

Теперь «читаемость» текста будет определяться только масочной сеткой экрана и общим качеством монитора. Правда, из-за недостаточного выходного напряжения черно-белого видеосигнала может снизиться яркость. Не желающие мириться с этим, а также с потерей цвета могут добавить 6-контактный переключатель типа П2К и собрать простейший эмиттерный повторитель на транзисторе типа КТ315, питание которого можно взять с ножки 1 выхода ТВ.

А для тех, кто не боится покопаться во внутренностях БК, можно посоветовать еще один способ подключения, позволяющий переходить из режима ЦТВ в ТВ переключением шнура в соответствующие гнезда. Доработка сводится к трем операциям: отключению напряжения +5В от ножки 1 гнезда ТВ; подаче на нее сигнала синхронизации с резистора R46; подаче сигнала с вывода 10 микросхемы D21 через конденсатор 0,1—1 мкФ на вывод 5 гнезда ТВ. В результате при подключении шнура (непеределанного!)



к выходу ТВ на цветной телевизор подаются ТВ-сигналы (только для зеленого цвета) достаточной амплитуды вместе с сигналом синхронизации и сохраняется возможность подключать к этому же выходу черно-белый телевизор.

На рисунке выделены новые элементы схемы. Способ испытан на мониторах «Электроника 32ВТЦ-201».

Т. СУЛЕЙМАНОВ (преподаватель),
Р. САБИЕВ, И. ЧУМУТКИН, О. ИВАНОВ
(учащиеся)

ЧТО МОЖЕТ ЭВМ?

Критерии выбора

Раньше выбрать ПЭВМ было легче. Объем ОЗУ, марка процессора, число дисководов практически полностью определяли возможности машины. А теперь...

Попробуем поговорить только о памяти, причем только о той, которую можно назвать оперативной. Точнее, ее можно определить как память, адресуемую процессором непосредственно или через диспетчер памяти; именно через нее попадает в процессор информация из остальных запоминающих устройств.

Итак, «ОЗУ». Первый уровень — непосредственно включенные в архитектуру процессора регистры, о которых говорить не будем именно потому, что они принадлежат процессору.

Второй уровень — так называемая кэш-память. Это буферное сверхоперативное запоминающее устройство. Быстродействие его ячеек таково, что даже процессор с тактовой частотой 25, 33 и более мегагерц получает из него информацию в течение следующего за запросом такта. Кэш-память все чаще располагается в той же микросхеме, что и процессор, но не является его частью; если такой «встроенной» памяти недостаточно или нет вообще, ее реализуют на внешних микросхемах. В архитектуре компьютера она практически не отличается от остальной части ОЗУ, но большинство обменов процессора с памятью проходит через нее. Специальная микросхема-диспетчер и специальные алгоритмы «подкачивают» в кэш-память данные из менее оперативной части ОЗУ с расчетом на то, что очередные понадобившиеся процессору данные окажутся именно в ней и ему не придется ждать 3—4 такта, пока отреагирует более медленная память. Заранее угадать, какие данные понадобятся в ближайшем будущем, удается не всегда, но чем больше кэш-память, тем больше в нее можно записать «на всякий случай»; однако сверхбыстродействующие микросхемы весьма дороги, и объем кэш-памяти редко превышает 256 Кбайт, а обычно составляет 32—64 Кбайт.

Здесь пора рассказать о типах микросхем памяти. Они бывают статическими и динамическими. В статических значение бита кодируется состоянием триггера, в динамических — степенью заряда конденсатора. Первые, естественно, значительно сложней (триггер состоит из 6—8 деталей), зато для хранения данных им достаточно напряжения питания. Вторые же требуют периодической регенерации данных, так как конденсаторы постепенно разряжаются; специальные устройства тысячи раз в секунду считывают информацию из динамических микросхем и снова записывают ее в то же место.

Итак, следующий уровень — собственно ОЗУ. Оно в большинстве случаев реализуется на дешевых динамических микросхемах. К его быстродействию не предъявляются особенно высокие требования, если есть страховка в виде кэш-памяти. Если же ее нет, ОЗУ приходится усложнять. Ведь в то время, когда проводится регенерация памяти, она недоступна для процессора, и ему приходится простаивать. Наиболее распространенный способ увеличения быстродействия — так называемое расслоение. Идея его проста: организуются как бы два ОЗУ: одно с четными, другое с нечетными адресами (их называют банками памяти). Пока регенерируются данные по четным адресам, нечетные готовы к считыванию-записи, и наоборот. Поскольку последовательное чтение данных — правило, а переходы по СОТО и им подобные — исключение, процессору при работе с таким ОЗУ гораздо реже приходится сталкиваться с необходимостью ожидать окончания регенерации данных в нужной ему ячейке.

Быстрый процессор может считать данные из четной ячейки, успеть обработать их и обратиться к нечетной раньше, чем в ней закончится регенерация. Чтобы такого не происходило, организуются не два, а три, четыре и больше банков памяти, что, естественно, усложняет и удорожает ОЗУ, но не так сильно, как использование статических микросхем.

Итак, наличие и объем кэш-памяти, быстродействие микросхем ОЗУ, число банков памяти — вот только основные параметры, по которым оценивается оперативная память современного компьютера.

Догнать и перегнать!

Почти 63 % американских семей смотрят кабельное телевидение, около 50 % имеют автоответчики, 6 % — ксероксы. Догоним ли Америку?.. А вот по компьютерам у нас есть шанс даже перегнать «врага». Домашние ЭВМ имеются только в 25 % американских семей! И поскольку «Микроша» намного дешевле самого простенького ксерокса, то...

И. СОКОЛОВСКИЙ

Плохо обусловленные вычислительные задачи в курсах информатики и математики

Вычисления давно стали далеко не единственной областью применения ЭВМ. Это обстоятельство отразилось и в школьном курсе информатики: о вычислениях с помощью ЭВМ рассказывается как об одном из возможных ее применений. В результате у школьников складывается слишком уж упрощенное представление и о проблемах, связанных с вычислениями на ЭВМ, и о ее вычислительных возможностях. Так, например, большинство школьников считают, что задачи, приведенные в учебниках по информатике, такие, как «решить квадратное уравнение», «решить систему линейных уравнений», или «вычислить значение элементарной функции, заданной формулой» и им подобные, не содержат в себе никаких серьезных проблем. Для их решения (по мнению учащихся) достаточно знать метод решения (формулу), правильно составить алгоритм и не допустить синтаксических ошибок при записи программы.

Таким образом, знания и опыт реальных вычислений у учащихся сводятся к одному: умению формально правильно реализовать вычислительный алгоритм.

Однако использование ЭВМ предъявляет особые требования к пониманию сути как самой решаемой вычислительной задачи, так и выбранного метода решения. Дело в том, что при ручных вычислениях есть возможность осмыслить промежуточные результаты, оценить их с точки зрения физического и просто здравого смысла. Если возникли неожиданные трудности или открылись новые, ранее не учтенные обстоятельства, то можно изменить первоначальный план действий. ЭВМ же, работающая по программе, после ввода исходных данных выдает окончательный результат. Конечно, и при вычислениях на ЭВМ можно кроме ответа вывести промежуточные результаты, но, чтобы решить, какие именно выводить, а затем суметь правильно истолковать эти результаты, требуется хорошее понимание сути.

Разнообразие областей применения ЭВМ вовсе не означает, что вычислительные задачи теперь не столь важны и соответствующему направлению в использовании ЭВМ можно уделить меньшее внимание. Напротив, ЭВМ практически вытеснили все другие средства вычисления, стали применяться там, где до сих пор использовался устный или ручной счет. Так что в компьютерную эпоху тем более необходимо вдумчивое, сознательное отношение к процессу вычислений.

Добиться такого результата обучения частными методическими приемами в ходе изучения какой-либо отдельной темы или даже одного предмета, очевидно, нельзя. Это проблема глобальная, и настоящее ее решение возможно только на пути концептуальной перестройки преподавания многих предметов школьного цикла, что, очевидно, не может быть рассмотрено в рамках одной статьи.

Цель настоящей статьи — показать, что плохо обусловленные вычислительные задачи содержат в себе богатые методические возможности. На вполне доступном уровне можно глубже показать природу рационального числа, наполнить физическим содержанием формальные определения абсолютной и относительной погрешности, дать новую целевую установку при выполнении тождественных алгебраических преобразований, представление о природе некоторых ошибок при вычислениях на ЭВМ и средства их преодоления. Эти задачи оказываются содержательными как с математической и физической точек зрения, так и в смысле техники программирования.

Задача 1.

ЭВМ входит в состав комплекса для автоматической обработки результатов эксперимента. Числа a, b и c получаются как результаты измерений. Все измерения выполнены с одинаковой относительной погрешностью. Требуется составить программу вычисления величины

$$P=\frac{a-b}{c}.$$

Эту задачу автор статьи неоднократно предлагал школьникам и студентам педагогического вуза — будущим учителям информатики, уже освоившим соответствующий курс ОИВТ. У всех опрошенных задача вызвала искреннее недоумение: «В чем проблема?» На предложение назвать пределы применимости алгоритма отвечали: «$c \neq 0$». На вопрос: «Как будет реагировать ваша программа на такие исходные данные: $a=2{,}34$; $b=2{,}34$; $c=1\cdot10^{-5}$?» — отвечали «P=0».

Эти ответы подтверждают высказанный выше тезис о формализме знаний учащихся в области реальных вычислений на ЭВМ. Дадим один из возможных вариантов анализа этой задачи, ориентированный именно на таких учащихся.

Прежде всего заметим, что в условии задачи нет лишних слов, как это часто бывает в так называемых сюжетных задачах. Скорее наоборот — некоторые обстоятельства еще потребуют уточнения. Так, например, будем считать, что значение величины P есть окончательный результат, интересующий исследователя. Для определенности будем считать, что ЭВМ выдает 8 десятичных знаков в мантиссе числа, представленного в форме с плавающей запятой. Относительную погрешность измерений величин a, b, c оценим числом верных значащих цифр. Допустим, что таких цифр 3.

Каким должен быть ответ ЭВМ в случае 1) $a=2{,}87$; $b=1{,}53$; $c=3{,}00\cdot10^{-3}$?

Простое вычисление по формуле $P=\dfrac{a-b}{c}$ дает значение $P=446{,}66666$.

Довольно естественно поручить самой ЭВМ определение числа верных значащих цифр в окончательном ответе. По правилам приближенных вычислений ответ должен выглядеть следующим образом: $P=447$.

Таким образом, нам необходимо научить ЭВМ округлять значение P в соответствии с известной относительной погрешностью исходных данных. Допустим, что это сделано. Какой ответ даст ЭВМ в случаях 2) и 3)?

2) $a=2{,}87$; $b=2{,}35$; $c=3{,}00\cdot10^{-3}$.

3) $a=2{,}87$; $b=2{,}83$; $c=3{,}00\cdot10^{-3}$.

В случае 2) машина сначала получит $P=\dfrac{2{,}87-2{,}35}{3{,}00\cdot10^{-3}}=\dfrac{5{,}2\cdot10^{-1}}{3\cdot10^{-3}}=173{,}33333$ и после округления — $P=173$.

В случае 3) $P=\dfrac{2{,}87-2{,}83}{3{,}00\cdot10^{-3}}=13{,}333333$ и окончательно $P=13{,}3$.

Проведем анализ промежуточных результатов.

Для случая 2). После вычитания в числителе получили $5{,}2\cdot10^{-1}$, но этот результат содержит только две верные значащие цифры (при вычитании из-за совпадения уничтожилась одна верная значащая цифра). Следовательно, и ответ должен содержать не более двух значащих цифр. Так что правильный ответ в случае 2): $P=1{,}7\cdot10^{2}$.

В случае 3) уничтожились уже две верные значащие цифры. Значит, в ответе должно быть не более одной верной значащей цифры, т. е. правильный ответ: $P=1\cdot10^{1}$.

Теперь перейдем к рассмотрению случая 4):

4) $a=2{,}34$; $b=2{,}34$; $c=1{,}00\cdot10^{-5}$.

ЭВМ выдаст ответ: $P=0$. Но ведь здесь совпали все верные значащие цифры и, следовательно, в разности $a-b=2{,}34-2{,}34=0$, а значит, и в окончательном ответе нет ни одной верной цифры. Произошла полная потеря точности! Ответ $P=0$ совершенно недостоверен. Это легко понять, если предположить, что параллельно в этом же эксперименте проводятся те же, но более точные измерения. Допустим, до пяти верных значащих цифр, и в случае 4) получены такие, более точные, данные:

$a=2{,}3442$; $b=2{,}3421$; $c=1{,}0000\cdot10^{-5}$.

Ответ для этого случая: $P=\dfrac{2{,}3442-2{,}3421}{10^{-5}}=210$.

В условии не зря сказано, что данные эксперимента обрабатываются автоматически. Это означает, что исследователь не может следить за промежуточными результатами. Если программист заранее не увидит опасности потери точности, то ЭВМ может выдать совершенно недостоверный ответ при том, что математическая модель явления составлена верно и никаких формальных ошибок в алгоритме вычислений нет. Природа ошибки в этом случае — приближенный характер исходных данных и уничтожение верных значащих цифр.

Спрашивается: что должен делать программист, понимающий суть проблемы? Прежде всего следует выяснить, могут ли значения a и b быть близкими числами. Если ответ отрицательный, то проблемы нет. Если a и b в пределах точности измерений могут совпадать, то необходимо выяснить, нельзя ли ее существенно повысить. Если это невозможно, то следует предложить экспериментатору поискать другую постановку эксперимента и попытаться найти значение P из каких-либо других соображений (формул), исключающих подобные неприятные ситуации. Если принятая постановка эксперимента и формула вычисления P единственно допустимые, то необходимо предупредить исследователя о возможности в какой-то момент частичной или полной потери точности.

Таким образом, тривиальная, на первый взгляд, поставленная задача превращается

в достаточно содержательную и совсем не элементарную для начинающего программиста — школьника. Именно: требуется при вычислении значений P выводить результат с указанием числа верных значащих цифр. Можно особым образом отмечать случаи значительной или полной потери точности. Для этого придется проводить (программными средствами) анализ значений исходных данных на степень их близости. Первое, что приходит в голову ученикам, это поразрядное сравнение числовых значений a и b. Но такой алгоритм определения числа совпадающих верных значащих цифр ошибочен по существу дела, так как совпадающие цифры формально могут быть различны. Приведем пример: $a=200{,}1$, $b=199{,}9$. Здесь формально все цифры разные, а по существу следует считать, что совпадают первые три, поскольку разность $a-b=200{,}1-199{,}9=0{,}2$ содержит только одну верную цифру.

Рассмотренная задача интересна еще и тем, что в ней возникает ограничение на область применимости алгоритма по новому для учеников параметру — числу совпадающих верных значащих цифр. Характерно также, что в ней программист не в силах что-либо предпринять, чтобы избежать потери точности, он может только предупредить об этом исследователя.

Ситуацию полной потери точности можно продемонстрировать на обиходном, чисто бытовом примере. Допустим, что вы собираетесь купить шкаф и хотите заранее знать, войдет ли он по ширине в отведенное для него место. Вы предварительно измерили ширину этого места и, допустим, получили результат 156 см (с точностью до 1 см). Измеряя в магазине шкаф, вы неожиданно получаете тот же результат — 156 см. Вычитание дает нуль. Но ведь ясно, что на самом деле нет точного равенства, и вопрос остался нерешенным. Вам придется еще раз более точно провести измерение и либо отказаться от покупки этого шкафа, либо вернуться за ним в магазин.

Рассмотренный бытовой пример потери точности при вычислениях показывает, что если относительная погрешность измерения c не хуже, чем у разности $a-b$, то действие деления на величину c в задаче 1 к существу проблемы отношения не имеет. Это действие введено из чисто методических соображений, чтобы получать более выразительные числовые значения, варьировать физический смысл (размерность) величины P.

В следующей задаче несколько изменяется природа возникновения ошибки, и, в принципе, программными средствами можно эту ошибку предотвратить.

Задача 2.

Требуется составить программу вычисления значений выражения:

$$P=a\cdot x+b-c:y+d.$$

Задача 2 появилась как задача-шутка в ситуации, сознательное подведение под которую может рассматриваться как методический прием, поэтому опишем ее.

В группе начинающих осваивать работу на программируемом МК возник вопрос о форме проведения предстоящего зачета. Договорились о следующем. Будет предложена задача на программирование. Успехи в освоении ПМК будут оцениваться тестированием программы. А именно: правильно составленная программа должна дать в ответе число 5, неверно составленная — число 2. Задача 2 и выступала в качестве образца такого рода задания. Специалисту понятно, что на самом деле такое возможно только в случае, если программа в определенном смысле действительно верно написана и безошибочно введена в машину.

Учащиеся, вдохновленные кажущейся простотой задания, с энтузиазмом и уверенностью принялись за дело. Распределили память ПМК, написали и ввели программу вычислений, прогнали несколько контрольных (самостоятельно составленных) примеров и сказали, что готовы к тестированию. После чего им были предложены следующие исходные данные: $a=1{,}2\cdot10^3$; $b=3$; $c=3{,}78\cdot10^4$; $d=2$; $x=1{,}26\cdot10^5$; $y=2{,}5\cdot10^{-4}$.

Ко всеобщему огорчению, у всех получилось в ответе число 2. После чего было предложено выполнить действия вручную, убедиться, что верный ответ равен 5, и попытаться найти тот момент, когда ПМК делает ошибку, установить ее природу.

Ручные вычисления дают такой результат:

1) $1{,}2\cdot10^3\cdot1{,}26\cdot10^5=1{,}512\cdot10^8$,
2) $1{,}512\cdot10^8+3=151200003$,
3) $3{,}78\cdot10^4:2{,}5\cdot10^{-4}=1{,}512\cdot10^8$,
4) $151200003-151200000=3$,
5) $3+2=5$.

При вычислениях на ПМК результат сложения $1{,}512\cdot10^8+3$ равен первому слагаемому, и в четвертом действии получаем нуль, в чем легко убедиться при пошаговом прохождении программы. Причина ошибки — конечность разрядной сетки ЭВМ и вычитание близких чисел (близкими являются числа $a\cdot x+b$ и $c:y$).

Можно ли составить программу, дающую правильный результат для предложенных исходных данных? Да, для этого нужно поменять местами слагаемые b и $-c:y$. Будет ли такая программа давать верный результат при любых допустимых значениях переменных? Нет, достаточно сделать близкими

числа $c:y—a\cdot x$ и b. Мы подвели учеников к поразительному для них открытию: при вычислениях на ЭВМ результат сложения может зависеть от порядка слагаемых. Таким образом, нарушается математический закон, незыблемость и очевидность которого вошла в поговорку: «От перемены мест слагаемых сумма не меняется». Легко понять, что возможность ошибки при вычислениях многократно возрастает, когда приходится выполнять не 3—5 простейших арифметических действий, а большое число самых разнообразных вычислений, которые совершенно невозможно прокрутить вручную. Этот вывод по-настоящему разочаровал многих учеников. Некоторые пришли к выводу, что ЭВМ нельзя использовать для вычислений (?!). Пришлось доказывать, что ЭВМ — это все же лучший вычислитель из всех существующих. Другое дело, что программист не должен уподобляться машине, формально выполняющей предписанные действия.

Задача 3.

Составить программу решения квадратного уравнения

$$ax^2+bx+c=0.$$

Формулы корней квадратного уравнения достаточно сложны по числу и набору действий:

$$x_1=\frac{-b-\sqrt{b^2-4ac}}{2a};$$

$$x_2=\frac{-b+\sqrt{b^2-4ac}}{2a},$$

поэтому вполне естественно предполагать, что при решении поставленной задачи возможна ситуация, при которой произойдет потеря точности. Мы не будем предпринимать попытки провести исчерпывающий анализ. Ограничимся следующим замечанием. В одной из формул корней в числителе обязательно будет разность чисел. При некоторых условиях эти числа могут оказаться близкими. Пусть для определенности $b>0$, тогда при $|ac|\ll b^2$ числа b и $\sqrt{b^2-4ac}$ близкие и в формуле для x_2 в числителе получается разность близких чисел, что, как известно, может привести к грубой ошибке. В отличие от задачи 1 здесь нам удается полностью устранить причину возможной ошибки за счет простого тождественного преобразования формулы для x_2:

$$x_2=\frac{-b+\sqrt{b^2-4ac}}{2a}=$$

$$=\frac{(-b+\sqrt{b^2-4ac})(b+\sqrt{b^2-4ac})}{2a(b+\sqrt{b^2-4ac})}=$$

$$=\frac{b^2-4ac-b^2}{2a(b+\sqrt{b^2-4ac})}=\frac{-2c}{b+\sqrt{b^2-4ac}}.$$

Итак, мы получили другую формулу для одного из корней квадратного уравнения:

$$x_2=\frac{-2c}{b+\sqrt{b^2-4ac}}.$$

Эта формула уже не содержит вычитания близких чисел, тем самым устранена причина возможной грубой ошибки.

Если $b<0$, то разность близких чисел возможна в формуле для x_1. Соответствующим преобразованием можно и ее привести к виду

$$x_1=\frac{-2c}{b-\sqrt{b^2-4ac}}.$$

Очевидно, что новые формулы для корней квадратного уравнения пригодны и для «нормальных» случаев, когда разность близких чисел не возникает. Таким образом, проводить анализ близости соответствующих чисел нет необходимости. Достаточно, определив знак b, выбрать для вычисления каждого корня ту формулу, в которой разности чисел нет.

Заметим, что при $|ac|\ll b^2$ и $b>0$ для x_2 получаем приближенную формулу:

$$x_2=\frac{-2c}{b+\sqrt{b^2-4ac}}\approx\frac{-2c}{b+\sqrt{b^2}}=-\frac{c}{b}.$$

Если $b<0$, то $x_1\approx-\frac{c}{b}$.

Интересно и поучительно провести следующее исследование решений квадратного уравнения. Запрограммировать вычисление корней квадратного уравнения по всем пяти формулам, включая приближенную, и выполнить серию вычислений при постепенном уменьшении $|ac|$ в сравнении с b^2.

Ниже приведены расчеты, выполненные на школьной ЭВМ «Корвет» по программе:

```
10 INPUT A,B,C
20 D=SQR(B^2-4*A*C)
30 ? "X1=";(-B-D)/2/A
40 ? "X3=";-2*C/(B-D)
50 ? "X2=";(-B+D)/2/A
60 ? "X4=";-2*C/(B+D)
70 ? "X5=";-C/B
```

Здесь x_1 и x_3 — значения корня x_1, вычисленные по различным точным формулам, соответственно x_2 и x_4 — значения корня x_2, x_5 — значение одного из корней, вычисленное по приближенной формуле.

1. «Нормальный случай»: $a=147$; $b=348$; $c=215$.

$x_1=-2{,}87591$; $x_2=0{,}508563$; $x_5=0{,}617816$;

$x_3=-2{,}87592$; $x_4=0{,}508564$;

2) $b=-200$; $c=3$.

a	x_1	x_2	$x_1 \approx$
10	$x_1=0{,}0150131$ $x_3=0{,}0150113$	$x_2=19{,}985$ $x_4=19{,}9825$	$x_5=0{,}015$
1	$x_1=0{,}0149612$ $x_3=0{,}0150011$	$x_2=199{,}985$ $x_4=200{,}518$	$x_5=0{,}015$
0,1	$x_1=0{,}0151825$ $x_3=0{,}0150001$	$x_2=1999{,}99$ $x_4=1975{,}96$	$x_5=0{,}015$
0,01	$x_1=0{,}0137329$ $x_3=0{,}015$	$x_2=20000$ $x_4=21845$	$x_5=0{,}015$
0,005	$x_1=0{,}00762933$ $x_3=0{,}015$	$x_2=40000$ $x_4=78643{,}2$	$x_5=0{,}015$
0,002	$x_1=-0{,}0038147$ $x_3=0{,}015$	$x_2=100000$ $x_4=-393216$	$x_5=0{,}015$
0,001	$x_1=-0{,}0076294$ $x_3=0{,}015$	$x_2=200000$ $x_4=-393216$	$x_5=0{,}015$

Что можно наблюдать в ходе такого исследования?

1) Когда нет вычитания близких чисел, соответствующие точные формулы дают одинаковые ответы. Значение корня, вычисленное по приближенной формуле,— грубо ошибочное.

2) При уменьшении $|ac|$ в сравнении с b^2 значения одного и того же корня, вычисленные по двум соответствующим точным формулам, начинают расходиться. Значение же $x_1 \approx \frac{-c}{b}$ сближается с вычисленным по формуле, в которой нет разности близких чисел.

3) При $|ac| \ll b^2$ значения x_1, вычисленные по точной, но «плохой» формуле, много хуже, чем значения x_1, вычисленные по приближенной формуле.

Таким образом, мы приходим к парадоксальному, на первый взгляд, выводу. В некоторых случаях вычисления по приближенной формуле дают очень хороший результат, тогда как по точной формуле приводят к грубой ошибке.

Следует предупредить желающих повторить подобные исследования о том, что при одних и тех же исходных данных результат вычислений существенно зависит от ряда обстоятельств, например от типа ЭВМ. Такая неустойчивость как раз и характерна для плохо обусловленных задач.

Вопросы, затронутые в настоящей статье, могут показаться весьма частными. На самом деле они тесно связаны с идейной направленностью преподавания математики, информатики, физики и других предметов естественнонаучного цикла. Рассмотренные задачи подтверждают актуальность проблемы оптимального соотношения между алгоритмической и содержательной составляющими в обучении названным предметам.

В. КАЗИЕВ, канд. физ.-мат. наук
Кабардино-Балкарский государственный университет

Дидактические алгоритмические единицы

Одной из актуальный задач преподавания основ информатики и вычислительной техники является воспитание культуры системного мышления, алгоритмической культуры — основ решения задач все возрастающей алгоритмической сложности, перехода к самостоятельности в процессе обучения. Этот процесс обычно осуществляется по схеме: понятие алгоритма, свойства алгоритма (определение, примеры из различных областей, примеры «алгоритмов», не обладающих теми или иными свойствами, например свойством массовости) — базовые алгоритмические структуры: линейная, ветвление, повторение, выбор — алгоритмические единицы, метод пошаговой детализации, основы анализа и синтеза, сборки новых алгоритмов на базе имеющихся алгоритмических единиц или изменения интерфейсов без изменения функционального ядра — решение сложных задач, переход от процесса обучения к процессу научения.

Для реализации этой схемы важно выделить и классифицировать основные *дидактические алгоритмические единицы* — функционально завершенные реализации тех или иных, наиболее часто используемых алгоритмических процедур, на базе которых строится решение большинства задач и которые составляют операционную среду пользователя.

Приведем комплекс таких дидактических алгоритмических единиц. Этот комплекс, разумеется, не является полным, и задача его пополнения является самостоятельной и интересной. Тем не менее его использование, как показывает практика, облегчает и ускоряет реализацию приведенной выше схемы.

1. Работа с числовыми данными

Организация ввода-вывода числовых данных: чисел, последовательностей, матриц (в случае матриц — различными способами:

по строкам, по столбцам, по спирали, по заданному правилу). Вычисление кусочно заданной функции. Вычисление значений заданной функции на заданном отрезке с заданным шагом (равномерным, неравномерным). Нахождение н. о. д. двух чисел. Нахождение н. о. к. двух чисел. Нахождение всех (заданного количества) делителей заданного числа. Вычисление целой части (остатка) от деления одного числа на другое. Округление десятичного числа от заданной позиции после запятой. Вычисление конечной суммы элементов заданной последовательности (матрицы). Вычисление суммы бесконечного ряда с заданной точностью. Вычисление конечного произведения элементов заданной последовательности (матрицы). Вычисление суммы (произведения) элементов, удовлетворяющих некоторому условию (относительно самих элементов, относительно индексов элементов). Вычисление факториала заданного числа. Инвертирование заданной последовательности. Нахождение наибольшего (наименьшего) элемента последовательности (матрицы). Вычисление полинома (схема Горнера). Вычисление по рекуррентной формуле с запоминанием (без запоминания) вычисленных членов. Удаление одинаковых элементов двух заданных последовательностей. Выделение элементов, входящих в две заданные последовательности. Генерирование заданного числа случайных чисел (натуральных, целых, вещественных) из заданного промежутка. Нахождение таблицы значений заданной функции по заданному формату. Рисование эскиза графика или графика с автоматическим масштабированием. Выделение цифр заданного числа. Округление заданного числа до числа, кратного заданному числу. Перевод из системы счисления с основанием $2 \leqslant p \leqslant 16$ в десятичную (и наоборот). Перевод из двоичной в восьмеричную (шестнадцатеричную) систему (и наоборот). Объединение двух матриц в одну. Разъединение матрицы в две матрицы. Замена элементов матрицы по заданному правилу относительно элементов (индексов элементов). Заполнение одномерной (двумерной) матрицы элементами двумерной (одномерной). Вставка заданного элемента в заданную упорядоченную последовательность с сохранением упорядоченности. Включение строки (столбца) в матрицу. Удаление строки (столбца) из матрицы. Перестановка местами двух строк (столбцов) матрицы. Циклический сдвиг элементов последовательности вперед (назад) на заданное число позиций. Ввод-вывод и рассмотрение (вталкивание — выталкивание) заданного числа элементов последовательности (массива) по правилам работы со стеком. Проверка таблицы матрицы на магический квадрат. Построение магического квадрата заданной размерности. Нахождение строки (столбца) с заданным условием относительно элементов строки (столбца). Сортировка элементов по возрастанию (убыванию): обменами, выбором, вставками. Нахождение инверсий в последовательности. Транспонирование матрицы. Нахождение произведения двух матриц. Хранение матриц в компактном виде: симметричной, ленточной, трехдиагональной. Двоичный поиск элемента последовательности. Проверка попадания заданной точки в заданное множество точек (окружность, прямоугольник, криволинейная фигура). Вставка упорядоченной последовательности в другую последовательность с сохранением упорядоченности. Упорядочение одной последовательности по заданной последовательности ключей. Поиск на графе, пути из одной вершины в другую, кратчайшего пути. Представление числа большой разрядности при условии, что задано ограничение на разрядность.

2. Работа с текстовыми данными

Вставка текста в текст с заданной позиции. Удаление заданного подтекста из заданного текста. Поиск в тексте подтекста с заданной позиции. Подсчет числа слов в заданном тексте. Подсчет числа предложений (строк) в заданном тексте. Подсчет числа слов заданной длины. Подсчет числа вхождений заданного символа (подтекста) в текст. Подсчет вхождений заданного множества символов в заданный текст. Подсчет числа слов текста на заданный символ (подтекст). Определение числа несовпадений в двух заданных текстах. Разделение заданного текста на подтексты заданной длины. Изменение разделителя слов (строк) в тексте. Поиск подтекста текста по заданному образцу (свойству). Кодировка буквенной (цифровой) комбинации цифровой (буквенной) с помощью заданной пользователем или генерируемой случайным образом (или по определенному правилу) системы кодов (шифра). Построение полиандрома для заданного текста. Получение полиандрома из заданного текста наименьшим числом удалений (вставок) букв в него. Получение комбинации символов по заданному закону, правилу (пример: игра Жизнь). Выравнивание текста: по правому краю, по левому краю, с переносом, без переноса. Алфавитное упорядочивание заданных слов. Удаление повторений в тексте. Представление «длинного» текста при условии, что задано ограничение на длину текста.

Автор разделяет подход, который не различает алгоритмизацию и программирование.

В то же время при переходе к программированию на конкретном языке программирования этот комплекс может быть пополнен программными единицами работы с экраном (очистка, установка курсора в позицию экрана и др.), с файлами (запись, считывание, сравнение и др.), с графикой (построение прямой, окружности, прямоугольника и др.), со спрайтами. Использование программных единиц позволяет перенести акцент обучения программированию с изучения элементов языка, синтаксиса, конкретного языка программирования (это можно успешно осуществлять с помощью различных обучающих программ и систем) на алгоритмическую сложность, технологию и методы программирования, с процедурного программирования на непроцедурное и логическое программирование.

Разработана автоматизированная система обучения указанным программным единицам (ДВК-3, УКНЦ). Система содержит около 100 основных программных единиц. Использование этой системы позволяет интенсифицировать процесс перехода к решению более сложных задач программирования с использованием программных единиц.

Приведем несколько таких задач.

Задача 1. Задана последовательность действительных чисел x_i, $i=1, 2, ..., n$. Найти два таких равных элемента x_k и x_l, $1 \leqslant l, k \leqslant n$, между которыми заключена убывающая подпоследовательность наибольшей длины.

Задача 2. Задан текстовый массив A из букв a_{ij}, $i, j=1, 2, ..., n$. Соседом элемента a_{pq} называется элемент a_{ij}, $(i, j)=(p-1, q-1)$, $(p-1, q)$, $(p-1, q+1)$, $(p, q-1)$, $(p, q+1)$, $(p+1, q-1)$, $(p+1, q)$, $(p+1, q+1)$. Составить программу, которая определяет, можно ли составить из соседних заданному элементу a_{kl} букв заданное слово S. Каждую букву A использовать только один раз.

Задача 3. Составить программу генерирования случайных шестнадцатеричных n-разрядных чисел.

Задача 4. Составить программу печати заданных n предложений в две колонки (аналогичных газетным) по m символов с выравниванием по левому и правому краю.

Задача 5. Составить программу перемещения всех четных чисел массива X элементов $x_1, x_2, ..., x_n$ в начало массива X, располагая их по возрастанию, а всех нечетных в конец массива — по их убыванию. Дополнительный массив не вводить.

Е. ЯНЧУК, Е. БАЛЫКИНА
Белорусский государственный университет

АУК для самостоятельного изучения философии

Целый ряд аргументов, как правило субъективного характера, выдвигается для обоснования невозможности использовать ЭВМ в преподавании гуманитарных дисциплин, особенно философских. Это и слабая формализуемость философского знания, гибкость философских понятий и связь философии как формы общественного сознания практически со всеми формами и пластами культуры. Это и принципиальная неспособность технических устройств заменить живое слово педагога, живое, эмоциональное общение преподавателя со студентами.

В Белорусском государственном университете в течение ряда лет ведутся работы по компьютеризации преподавания общественных и гуманитарных дисциплин. Практика показала, что в режиме автоматизированного обучения большая часть деятельности студента реализуется в виде самостоятельной работы.

При обучении с помощью ЭВМ самостоятельную работу можно сделать не только оперативно контролируемой, но и управляемой. Тем самым преподавателю предоставляются широкие практические возможности, особенно с учетом современных тенденций к сокращению количества лекций, семинарских занятий, увеличения отданных на самоподготовку тем.

Как всякая система, АОС состоит из подсистем: технической, программной и учебно-методической. Становится все более очевидной необходимость концентрации усилий на совершенствовании учебно-методической подсистемы. Она включает в себя АУК, типовые алгоритмы обучения, методики разработки и отладки обучающих программ, методику освоения работы за терминалом и т. д. Эту часть АОС можно назвать педагогическим обеспечением, понимая под этим общие возможности педагогического воздействия на обучаемых через АОС, возможности их обучения и воспитания при работе в автоматизи-

рованном классе. АОС позволяет в удобное для студента время отработать нужную тему в удобном для себя темпе в отличие от аудиторного занятия, время которого строго фиксировано. По сути, весь учебный процесс можно свести к цепочке форм: лекция — самостоятельная работа — практическое занятие. Они могут быть представлены вместе, поодиночке и в парах: лекция — семинар, лекция — самостоятельная работа, семинар — самостоятельная работа и т. д. В каждом конкретном случае сам преподаватель, исходя из степени сложности темы, ее важности и актуальности, степени подготовки студентов и т. п., может составлять программу учебного процесса. Однако в случае отсутствия либо первого, либо третьего звена (сокращение числа часов на предмет, свободное посещение лекций, пропуск практических занятий) преподаватель будет уверен, что студент все равно приобретет некоторый уровень знаний, если его работой управляет компьютер. Педагогу остается проверить наличие протоколов отработки тем и при недостаточном их количестве отказать в допуске к зачету или экзамену.

Такова в общих чертах наша концепция компьютерной технологии в режиме самостоятельной работы. Но для ее реализации АУК должен иметь определенную структуру.

Нами предложена следующая структура АУК для организации самостоятельной работы студентов при обучении философии.

В нее входят курс-каталог, курс-тест, теоретический, обучающий (или тренирующий) и контролирующий курсы. Придя в автоматизированный класс, студент после регистрации выбирает по каталогу АУК тему для самостоятельного изучения. Здесь же он знакомится с краткой характеристикой выбранного материала. Индивидуализация в наших АУК осуществляется в двух направлениях: по уровню знаний и по уровню психофизиологических характеристик (беспокойство, тревога, усталость, объем памяти и т. п.). Уровень знаний у студентов может быть высокий, средний и низкий. Это выясняется путем прохождения входного курса-теста. Как правило, предлагается ответить на 4 случайно выбранных вопроса различной степени сложности за фиксированное время. В нашем случае это 6 мин, к которым мы пришли экспериментальным путем. Получив «отлично», студент переходит к контролю своих знаний. Ему предлагается ответить на 12 вопросов в течение 18 мин с определением дифференцированной оценки. Причем контроль может повторяться путем получения новых порций вопросов до тех пор, пока итоговая оценка не удовлетворит студента и преподавателя. Далее следует выходной тест: опять 4 вопроса на 6 мин. Такой же путь по программе проходит студент, показавший на входном тесте средний уровень знаний.

Если же входной тест покажет низкий уровень знаний, то студенту предлагается изучить теоретический материал по данной теме (порции учебного материала также подаются с экрана дисплея), а затем приступить к обучению.

Теоретическая часть курса — это своего рода опорные зоны. Это квинтэссенция материала, представленная в наиболее удобном для восприятия и запоминания виде. С этой целью используется система знаков и символов, включающая схематические рисунки и ключевые слова, мнемонические приемы, короткие тезисы, выделение формой и т. д. Современные компьютеры позволяют акцентировать внимание студента миганием определенных цифр и слов, программной мультипликацией, подчеркиванием, негативированием, использованием букв увеличенного в несколько раз размера, когнитивной интерактивной графикой, цветом и музыкой.

Теоретический курс оснащен обратной связью: студенту задается ряд вопросов по узловым моментам темы. Это позволяет оценить степень усвоения материала.

Следующий этап — обучающий курс, включающий набор контрольных вопросов по избранной теме. Вопросы можно разделить на 3 уровня сложности. Во-первых, это задания на выбор правильного ответа из ряда предложенных вариантов. Отношение к таким «программированным» вопросам сейчас отрицательное. Однако практика показывает, что с этими заданиями студенты справляются быстро и ими очень удобно разбавлять материал. Это способствует и психологической разрядке. Такие задания у нас обычно не превышают 8—9 % от общего количества вопросов.

Вторая группа — вопросно-ответные задания, требующие либо предельно краткого ответа, либо развернутого, свободно конструируемого. Именно они лежат в основе АУК. Третью группу (20—30 %) составляют задания на анализ текста, например выявление философской позиции по тому или иному вопросу. Именно их, на наш взгляд, можно охарактеризовать как практические задачи. Они помогут студенту апробировать свои теоретические знания, перевести их из состояния мертвого груза в форму активного знания.

Особенностями обучающего курса являются неограниченное время пользования (единственное ограничение — заказанное время); возможность обратиться к теории, воспользоваться справкой, помощью, подсказкой; обучающий курс предполагает на-

личие пояснений, комментариев, наводящих вопросов в наиболее сложных заданиях; плохо усвоенные вопросы (неправильные ответы, отказ от ответов, превышение лимита возможных попыток для ответа) студент проходит повторно. Тем самым реализуется принцип циклического возврата, своего рода вертушка. Студент не выйдет из программы, пока не проработает все задания правильно. Программа «снисходительна» к студенту: сделав ошибку, он получает текст правильного ответа.

Справочный материал по заданиям составлен из соответствующих теме разделов работ классиков философии. Студент получает перечень справок (своеобразное меню) и выбирает нужную, указав ее номер. Справку можно организовать и по каждому вопросу. Тогда обучаемый, обращаясь за ней, вводит и номер вопроса. В тексте помощи компьютер дает наводящий вопрос, не раскрывающий сам ответ, но подталкивающий к нему. В подсказке дается подробное разъяснение, и студент, хотя бы немного знакомый с изучаемой темой, без труда даст после подсказки верный ответ.

После прохождения всех этапов студент переходит к контролю, о котором было сказано выше (12 вопросов на 18 мин) и к выходному тесту (4 вопроса на 6 мин).

Такова структура АУК для самостоятельного изучения философии.

Практика показала, что если на входном тесте мы получаем в основном удовлетворительные и неудовлетворительные оценки, то в результате прохождения по всем веткам программы после выходного контроля мы имеем оценки хорошие и отличные.

Мы регистрируем все этапы и результаты учебной деятельности студента. Можно фиксировать следующие параметры: прохождение учебного материала по рекомендованным разделам, время, затрачиваемое каждым студентом и группой на изучение материала: количество ошибок и их текст, что полезно для выявления типичных ошибок; регулярность самостоятельной работы. Статистика о ходе самостоятельной работы студента может быть использована преподавателем для совершенствования обучающих программ и корректировки методики других видов учебных занятий.

Л. ВОЛКОВА

Практикум для КУВТ УКНЦ 02.02 (МС 0511)

В статье описывается созданный в Нижегородском радиоэлектротехническом техникуме практикум по предмету «Алгоритмические языки и программирование», цель которого — развитие у учащихся алгоритмического мышления и навыков работы на ЭВМ.

Практикум — это прежде всего хорошо разработанная система заданий различной сложности, обеспечивающая индивидуальный подход к учащимся с различным уровнем подготовки, степенью заинтересованности и творческими способностями.

Практикум предполагает проведение лабораторных работ следующего содержания.

I. Использование операторов Бейсика.

1. Операции и функции с комплексными переменными.

2. Одномерные и двумерные массивы. Операции с матрицами. Обработка результатов, заданных таблично.

3. Символьные величины, работа с текстами, редактирование.

4. Графика и псевдографика.

II. Специальные вопросы программирования.

1. Запоминание результата.

2. Организация цикла с несколькими одновременно меняющимися параметрами.

3. Вычисление суммы бесконечного ряда.

4. Вычисление значений и корней полинома, имеющего действительные корни.

5. Методы сортировки числовых и символьных массивов.

6. Организация диалога с пользователем.

7. Приемы экономии памяти.

8. Перевод программ с одной версии языка Бейсик на другую.

III. Численные методы и их реализация на ПЭВМ.

1. Методы решения нелинейного уравнения с одним неизвестным.

2. Численные методы решения системы линейных уравнений.

3. Численные методы вычисления определенных интегралов.

4. Поиск экстремума функции.

5. Заключительное занятие. Зачет.

Методическое описание каждой лабораторной работы состоит из пяти пунктов:

1) формулировка цели работы;

2) общая теоретическая часть (с примерами);

3) описание последовательности выполнения лабораторной работы;

4) контрольные вопросы и задания;

5) таблица индивидуальных заданий.

В качестве примера предлагаем вашему вниманию п. 3 лабораторной работы «Методы сортировки числовых и символьных массивов».

Сортировка чисел

Изучить программу сортировки методом пузырька.

Переписать программу в тетрадь с комментариями к основным группам операторов.

Ввести программу в компьютер.

Отладить программу и исполнить ее несколько раз для последовательностей из 10, 30, 50 и 100 чисел.

Время, затраченное на сортировку каждой группы чисел, записать в черновик для последующего использования в п. 7.

Выполнить п. 1—5 для программы сортировки методом Шелла.

Результаты сортировки по двум методам оформить в виде таблицы.

Вопросы и задания

Вставьте в программы добавочные строки так, чтобы после окончания сортировки списка, вы получили сообщение о том сколько было сделано сравнений и перестановок.

Для проведения сравнения методов включите (зачем?) в программу генерацию чисел:

```
140 LET N (I) INT (RND (1)×20+1)
150 PRINT N (I)
165 INPUT «Запишите время и нажмите клавишу»; V $
```

Что в строке с номером 140 должно меняться?

Чтобы лучше разобраться в программе сортировки методом Шелла, можно вставить следующие строки:

```
233 PRINT
235 PRINT «F»; F; «G»; G, где F и G — индексы сравниваемых чисел
237 PRINT «Сравнение N («; F;») и N («; G;»)»
245 PRINT «Перестановка местами N («; F;») и N («; G;»)»
275 FOR J 1 TO T
276 PRINT N (J); « »;'вывод на экран текущего
277 NEXT J 'состояния списка
278 PRINT
279 PRINT «Чтобы продолжить, нажмите клавишу»; V$
```

Сортировка слов

Для сравнения букв компьютер использует тот же метод, что и для сравнения чисел. При сравнении символов компьютер сравнивает числовые коды этих символов.

Чтобы сравнить два слова между собой, компьютер сначала сравнивает первые буквы каждого слова, и, если они одинаковы, он сравнивает вторые, третьи, и т. д.

В строковой переменной можно помещать как буквы, так и цифры, поэтому программу сортировки можно использовать для данных, которые содержат слова и числа, например адреса.

Задания.

Изменить программу сортировки методом пузырька для выполнения сортировки слов.

Записать в тетрадь измененные строки.

Ввести программу в компьютер.

Отладить ее и пропустить несколько раз для массива из десяти слов, начинающихся с разных букв, с совпадающими: а) первыми буквами; б) первыми и вторыми буквами; в) первыми, вторыми и третьими буквами.

Время, затраченное на сортировку, записать в черновик.

Выполнить п. 1—5 для программы сортировки методом Шелла.

Результаты занести в таблицу.

Результаты сортировки слов и чисел

Сортировка					
чисел методом Шелла	чисел методом пузырька	слов методом Шелла с совпад. первыми буквами	слов методом пузырька с совпад. первыми буквами	слов методом Шелла с совпад. первыми и вторыми буквами	слов методом пузырька с совпад. первыми и вторыми буквами
...	...	...	...	...	...

Вопросы и задания

Напоминаем!!! Функция ASC () преобразует символ в его числовой код.

Измените программы сортировки таким образом, чтобы при вводе отсортированного списка выводились и коды первых символов слов.

Напишите программу сортировки слов в порядке русского алфавита.

Ко всем лабораторным работам учащимся предлагаются различные типы заданий, вызывающие у них интерес.

Это работа с готовыми программами, компоновка программы учащегося с уже разработанным блоком, составление собственных программ, откладка программ, содержащих ошибки, и т. д. Предлагаемые учащимся программы даются в виде листинга и проверены на отсутствие ошибок.

Программы записаны на диск, прилагаемый к методическому обеспечению практикума.

Сложность работ, включенных в практикум достаточно велика, но каждый преподаватель имеет возможность часть наиболее сложных работ разобрать на теоретических занятиях, выделить на тему большее количество часов, исключить отдельные лабораторные работы, ограничиться заданиями только первого уровня, рассмотреть только один метод из трех предложенных и т. д.

Предлагаемый практикум достаточно гибок для работы, и поэтому, надеемся, понравится широкому кругу преподавателей предметов «Основы информатики и ВТ» и «Алгоритмические языки и программирование».

При отсутствии возможности проводить практикум целиком можно воспользоваться отдельными разработками и программами для проведения разовых (отдельных) лабораторных работ. Практикум прекрасно реализует возможности «Электроники МС 0511», увлекает учащихся разнообразием и обилием заданий все возрастающей сложности и позволяет преподавателю творчески организовать учебный процесс.

Это описание практикума не дает возможности составить о нем полное представление, и автор с удовольствием ответит на все вопросы и будет признателен тем, кто пришлет свои отклики.

Осенью планируется проведение семинаров для преподавателей. Ждем вас в Нижнем Новгороде!

Тех, кто заинтересовался программно-методическим комплексом «Практикум по предмету «Алгоритмические языки и программирование», состоящим из трех частей: методического описания лабораторных работ для учащихся, программного обеспечения и инструкции пользователя для преподавателя, просим обращаться по адресу:

603600, Нижний Новгород, ГСП-284, Студенческая, 6, Нижегородский радиоэлектротехнический техникум, комиссия автоматики и ВТ. Волковой Людмиле Валентиновне. Тел.: 33-46-89, 33-86-05, 34-01-92.

И. ПЕРЕХОД, В. КАСАТКИН

Арифметика системы остаточных классов

Среди недесятичных систем счисления особое место принадлежит системе остаточных классов (СОК). Алгоритм перевода чисел, записанных в СОК, в равные им десятичные числа был известен еще до нашей эры в Китае. Суть алгоритма выражается так называемой Великой китайской теоремой. В оригинальной интерпретации это задача об отыскании натурального числа по известным остаткам его деления на заданные взаимно простые числа.

Особенности системы привлекли внимание конструкторов вычислительной техники; существует немало устройств, работающих в соответствии с арифметикой СОК.

Математиков в СОК привлекают алгоритмы параллельного выполнения арифметических действий, в которых результат операций с цифрами одного разряда не зависит от операций над цифрами других разрядов.

Итак, в СОК осуществляется независимая поразрядная обработка чисел. Покажем, что СОК можно задать с помощью предложенного авторами метода нижних и верхних границ, и применим понятие обобщенного счетчика при разработке алгоритмов работы с числами в СОК.

Задать СОК традиционно — значит указать количество n используемых разрядов и определить для каждого разряда i соответствующий модуль $P(i)$. В роли модулей используются только попарно простые числа.

Пример 1.

Пусть число разрядов n в некоторой СОК равно 5, а модули равны числам $P_1=2$, $P_2=3$, $P_3=5$, $P_4=7$ и $P_5=11$. Известно, что любое натуральное число можно единственным способом записать в конкретной системе остаточных классов.

Число A_{10} конкретной системы остаточных классов представляется в виде набора цифр данной СОК:

$$A_{10}=\overline{a_1a_2...a_n}\,(\text{сок}),$$

где a_i есть остаток от деления числа A_{10} на соответствующий модуль P_i.

Набор цифр меняется от разряда к разряду: $0\leqslant a_i<P_i$, причем для любого i $0\leqslant a_i<P_n$.

Следует иметь в виду, что диапазон представляемых в СОК чисел определяется условием $0\leqslant A_{10}<P_1\cdot P_2\cdot ...\cdot P_{n-1}\cdot P_n$. В приведенном примере $0\leqslant A_{10}<2\cdot 3\cdot 5\cdot 7\cdot 11=2310_{10}$.

Пример 2.

Пусть $A_{10}=39$, $B_{10}=40$, $C_{10}=41$, тогда в системе с модулями $P_1=2$, $P_2=3$, $P_3=5$, $P_4=7$, $P_5=11$ эти числа запишутся так:

$$A_{10}=39_{10}=10446_{\text{сок}};$$
$$B_{10}=40_{10}=01057_{\text{сок}};$$
$$C_{10}=41_{10}=12168_{\text{сок}}.$$

Из сказанного следует, что при заданной разрядности данное число всегда содержит цифры во всех n разрядах. Ведущие нули опускать нельзя: являясь остатками, они несут информацию, по которой вычисляется значение числа.

Программа перевода данного натурального A_{10} в равное ему число в конкретной СОК приводится ниже.

```
10 '----- ДЕШИФРАТОР 10--> СОК ----
   '------ РАЗРЯДНОСТЬ К ------------
20 INPUT "K=";K: DIM P(K)
30 PRINT "МОДУЛИ"
40 FOR I=1 TO K: INPUT P(I): NEXT I
50 INPUT "X=";X: PRINT "Y=";
60 FOR I=1 TO K
70 PRINT X MOD P(I);
80 NEXT I: PRINT: END
```

Арифметические операции над числами СОК определяются особым образом. Не останавливаясь на подробностях арифметики СОК, подчеркнем только главную особенность операций сложения и умножения.

Сложение чисел СОК осуществляется поразрядно и параллельно во всех разрядах, в каждом разряде по модулю P_i.

Пусть, например, СОК задана модулями $P_1=2$, $P_2=3$, $P_3=5$, $P_4=7$, $P_5=11$ и необходимо сложить числа 39_{10} и 41_{10}, а также числа 107_{10} и 41_{10}.

$$+\begin{array}{l} 39_{10}=10446_{\text{сок}} \\ 41_{10}=12168_{\text{сок}} \\ \hline 80_{10}=02033_{\text{сок}} \end{array}$$

$$\begin{array}{l} 107_{10}=12228_{\text{сок}} \\ 41_{10}=12168_{\text{сок}} \\ \hline 148_{10}=01315_{\text{сок}} \end{array}$$

Умножение осуществляется поразрядно без образования промежуточных произведений.

Пусть, например, в данной СОК необходимо умножить 19_{10} на 12_{10} и 49_{10} на 35_{10}.

$$\times\begin{array}{l} 19_{10}=11458_{\text{сок}} \\ 12_{10}=00251_{\text{сок}} \\ \hline 228_{10}=00348_{\text{сок}} \end{array}$$

$$\times\begin{array}{l} 49_{10}=11405_{\text{сок}} \\ 35_{10}=12002_{\text{сок}} \\ \hline 1715_{10}=1200\text{A}_{\text{сок}} \end{array}$$

цифра $A_{\text{сок}}=10_{10}$

Определенные таким способом арифметические операции позволяют кардинальным образом перестроить процесс вычислений. Сложение или умножение цифр может выполняться одновременно во всех разрядах, и результат, полученный в одном разряде, никак не влияет на результаты, получаемые в других разрядах.

Важной является задача построения дешифратора СОК→10. Его алгоритм базируется на формуле

$$A_{10}=(a_1b_1+a_2b_2+\ldots+a_{n-1}b_{n-1}+a_nb_n)\ (\text{mod N}),$$

где $N=P_1P_2\ldots P_n$ — произведение модулей системы, a_i — цифры числа, записанного в СОК, b_i — коэффициенты, которые находят из условия

$$b_i=\frac{m_iN}{P_i},$$

а m_i находят из условия

$$\frac{m_iN}{P_i}=1\ (\text{mod } P_i).$$

```
10 '------- ДЕШИФРАТОР СОК-->10 -------
   ---------РАЗРЯДНОСТЬ К ------------
20 INPUT "K=";K: DIM P(K),R(K)
30 PRINT "МОДУЛИ": N=1
40 FOR I=1 TO K: INPUT P(I): N=N*P(I):
                                NEXT I
50 PRINT "ЦИФРЫ ИСХОДНОГО ЧИСЛА В СОК"
60 FOR I=1 TO K: PRINT "ДЛЯ МОДУЛЯ";
          P(I);: INPUT R(I): NEXT I
70 FOR I=1 TO K: C=N\P(I)
80 FOR M=1 TO P(I)-1: B=M*C
90 IF B MOD P(I)<>1 THEN NEXT M
100 S=S+R(I)*B: NEXT I
110 PRINT "Y=";
        INT((S/N-INT(S/N))*N+.5): END
```

Важной является задача создания последовательного счетчика в СОК.

Пусть, например, система задана набором модулей $P_1, P_2, \ldots, P_n$. Необходимо сконструировать генератор, который будет получать и печатать в возрастающем порядке все натуральные n-разрядные числа СОК.

Возможны два подхода к решению этой задачи.

Первый основывается на том, что генерация следующего числа сводится к сложению с единицей цифры каждого разряда (X_i) по соответствующему этому разряду модулю: $X_i:=X_i+1\ (\text{mod } P_i)$ для всех i. При сложении по модулю единицы переноса не формируются, и задача конструирования n-разрядного счетчика сводится к созданию n независимо работающих одноразрядных счетчиков.

Так как перенос в $(n+1)$-й разряд (разряд переполнения) также не осуществляется, то окончание процесса счета определяется по совпадению очередного показания счетчика с начальным.

Если начальное значение представлено кодом $\overline{0\ldots0}_{\text{сок}}$, то счетчик построит упорядоченную по возрастанию последовательность чисел:

$$\overline{X_{\min}=00\ldots0}_{\text{сок}},\ \ldots,\ X_{\max}=\overline{P_1-1\ P_2-1\ldots P_n-1}_{\text{сок}}.$$

Ниже приводится программа, реализующая первый вариант счетчика СОК, и пример ее работы.

```
10 '---- СЧЕТЧИК В СОК (1) ----
20 INPUT "N=";N: DIM X(N),P(N)
30 LPRINT "МОДУЛИ:";
40 FOR I=1 TO N: INPUT P(I): X(I)=0:
   LPRINT P(I);: NEXT I: LPRINT: LPRINT
45 LPRINT TAB(8)
50 FOR I=1 TO N: LPRINT X(I);: NEXT I:
                                 LPRINT
60 FOR I=1 TO N
70 X(I)=(X(I)+1) MOD P(I)
80 NEXT I
90 FOR I=1 TO N
100 IF X(I)<>0 THEN 45
110 NEXT I: END
```

```
МОДУЛИ: 2  3  5

        0  0  0
        1  1  1
        0  2  2
        1  0  3
        0  1  4
        1  2  0
        0  0  1
        1  1  2
        0  2  3
        1  0  4
        0  1  0
        1  2  1
        0  0  2
        1  1  3
        0  2  4
```

```
1  0  0
0  1  1
1  2  2
0  0  3
1  1  4
0  2  0
1  0  1
0  1  2
1  2  3
0  0  4
1  1  0
0  2  1
1  0  2
0  1  3
1  2  4
```

Второй подход основан на использовании понятия обобщенного счетчика. В этом случае применяется способ, в котором все разряды мыслятся как взаимодействующие, а возможность их независимой обработки игнорируется. Иными словами, в основу этого подхода положен традиционный алгоритм счета. Единица прибавляется к младшему разряду и затем последовательно продвигается в ходе переноса от разряда к разряду. Как и при создании классических счетчиков в других системах счисления, используется обычное (не по модулю) сложение с единицей и применяется единая процедура следования Generalized Succ:


```
Procedure Generalized_Succ;
  Begin
    While i<=n and x[i]=Sup[i] Do
                              i:=i+1;
    c:=i;
    If c<=n Then x[c]:=x[c]+1;
    For i:=c-1 Downto 1 Do
                        x[i]:=Inf[i];
  End;
```

Напомним, что при таком подходе необходимо задать нижнюю INF и верхнюю SUP границы разрядов счетчика.

На первый взгляд, традиционный принцип переноса (по достижении верхней границы) для конструирования счетчиков в СОК неприменим, однако соответственно подобранные INF(i) и SUP(i) позволяют использовать обобщенную процедуру.

Функции эти имеют следующий вид:

$$\mathrm{INF}(i)=(\mathrm{X}(i)+1) \pmod{\mathrm{P}_i};$$
$$\mathrm{SUP}(i)=\mathrm{X}(i).$$

Разъясним принцип действия счетчика с границами, заданными этим способом. На каждом такте работы счетчика значение каждого разряда рассматривается как верхняя граница, и в соответствии с обобщенной процедурой следования верхние границы всех разрядов заменяются нижними. Это отвечает свойствам СОК независимой и одновременной обработки всех разрядов. Суммирование X_i с единицей по модулю P задается в данном варианте счетчика при вычислении функции INF(i).

Текст программы, реализующей второй вариант счетчика СОК, приведем ниже.

```
10 '--- СЧЕТЧИК В СОК (2) ---
20 INPUT "N=";N: DIM X(N),P(N)
30 LPRINT "МОДУЛИ:";
40 FOR I=1 TO N: INPUT P(I): X(I)=0:
   LPRINT P(I);: NEXT I: LPRINT: LPRINT
50 DEF FNSUP(J)=X(J):
       DEF FNINF(J)=(X(J)+1) MOD P(J)
55 LPRINT TAB(8)
60 FOR I=1 TO N: LPRINT X(I);: NEXT I:
                                  LPRINT
70 'ОБОБЩЕННАЯ ПРОЦЕДУРА СЛЕДОВАНИЯ
80 FOR I=1 TO N
90 IF X(I)=FNSUP(I) THEN X(I)=FNINF(I):
                NEXT I ELSE X(I)=X(I)+1
100 FOR I=1 TO N
110 IF X(I)<>0 THEN 55
120 NEXT I: END
```

В заключение остановимся на комбинаторной интерпретации работы последовательного счетчика СОК.

Напомним, что счетчик генерирует n-разрядные числовые наборы. Например, в результате работы трехразрядного счетчика с модулями $P_1=2$, $P_2=3$ и $P_3=5$ осуществляется генерация всех возможных троек, в каждую из которых включается один элемент из множества цифр, используемых в каждом из трех разрядов.

Вообще, счетчик в СОК генерирует комбинаторные объекты в виде наборов из n элементов $\overline{a_{i_1}a_{i_2}\ldots a_{i_n}}$, где

$$a_{i_1} \in \mathrm{A}_1=\{0,\ 1,\ \ldots,\ \mathrm{P}_1-1\}$$
$$a_{i_2} \in \mathrm{A}_2=\{0,\ 1,\ \ldots,\ \mathrm{P}_2-1\}$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$a_{i_n} \in \mathrm{A}_n=\{0,\ 1,\ \ldots,\ \mathrm{P}_n-1\}$$

Такие комбинаторные объекты принято называть размещениями с повторениями и ограничениями. Суть ограничений в том, что элементы разрядов счетчика выбираются из множеств различной мощности, причем мощность A_i равна P_i.

Количество построенных объектов $N=P_1P_2\ldots P_n$, в частном случае, когда $P_1=P_2=\ldots=P_n$, имеем $N=P^n$ обычных размещений с повторениями без ограничений.

Аукцион окончен, работа продолжается

Мы снова возвращаемся к пробеме самопечати. На аукцион, объявленный в «ИНФО» 4—90, пришло множество писем с решениями поставленной задачи. Авторы некоторых из них критикуют нас за расплывчатость, нечеткую формулировку условий конкурса. Отчасти можно согласиться с такого рода критикой. Действительно: не перечислены все запрещенные операторы и приемы, недостаточно оправдан критерий «меньшести». Наверное, логичнее было бы добиваться наименьшего числа операторов или символов, а не строк, так как большинство версий Бейсика позволяют «загонять» в одну строку программу длиной до 255 символов.

Предложенные на аукцион решения можно разделить на несколько групп. В первой — программы, использующие различные алгоритмы с оператором DATA. Новинкой среди них является алгоритм, присланный В. Шляхетским (Москва).

```
FOR I = 1 TO 10
IF I = 6 THEN RESTORE
READ A$
IF I < 6 THEN PRINT A$ ELSE
                    PRINT "DATA "; A$
NEXT
DATA FOR I= 1 TO 10
DATA IF I = 6 THEN RESTORE
DATA READ A$
DATA IF I < 6 THEN PRINT A$ ELSE
                    PRINT "DATA "; A$
DATA NEXT
```

Вторая группа — программы, использующие строковые функции Бейсика (MID, LEFT, RIGHT, CHR и т. д.). Здесь хорошую ставку сделал Д. Звездин из Ленинграда.

```
10 A$="10 A$=20 ? MID$(A$,1,6);
          CHR$(&042);A$;CHR$(&042),,
                    MID$(A$,7,57)"
20 ? MID$(A$,1,6);CHR$(&042);A$;
          CHR$(&042),,MID$(A$,7,57)
```

Наконец, программы, печатающие свои тексты, предварительно сохраненные на каких-либо внешних устройствах. Их прислано немало, но такие алгоритмы не соответствуют условиям конкурса (зависят от конкретной ПЭВМ).

Конечно, необходимость работать на Бейсике сильно сузила рамки аукциона. Этот язык не лучшим образом подходит для решения проблемы самопечати, ограничивает творческую инициативу. Поэтому предлагаем вам еще подумать над самопечатью, но используя уже не Бейсик, а другие языки высокого уровня (например, Си или Паскаль). С их помощью могут быть созданы более красивые и более оригинальные алгоритмы. Для разминки приведем программу, печатающую саму себя, на языке Си (а именно — Turbo C 2.0 для IBM PC).

```
#include"list.h"
main()
{
 list();
}
```


Содержимое файла list.h:

```
/* Этот файл содержит функцию "list",
   определяемую пользователем, которая
   выполняет печать исходного текста
                              программы */
void list(void)
{
 char c='"';
 printf("\n#include%clist.h%c\nmain()
          \n{\n list();\n}\n",c,c);
}
```

В заключение можно добавить, что в наше время, когда созданы мощнейшие средства программирования, проблема самопечати интересна, естественно, не сама по себе, а как тренировка перед решением более серьезных вопросов, например, как сделать так, чтобы роботы могли создавать себе подобных без помощи человека. Не посвятить ли следующий аукцион именно этой задаче?

С. ВОЛКОВ

МИ

МОЛОДЕЖНАЯ ИНИЦИАТИВА

Немного о Лого

Продолжаем знакомить наших читателей с работами молодых победителей выставки «Новые педагогические программные средства для средней школы», проходившей в декабре—январе на ВДНХ СССР, и предлагаем вашему вниманию статью о разработке, завоевавшей на этой выставке серебряную медаль ВДНХ.

Опыт преподавания информатики в школе-лицее № 542 показывает, что, начиная изучать курс, на этапе перехода от абстрактных рассуждений и простейших обучающих программ к конкретным приложениям — языкам программирования, дети сталкиваются с определенными трудностями.

Большинство существующих обучающих программ реализованы на уровне видеоигр, что имеет свои неоспоримые премущества. Игра в этом случае является наиболее простым и естественным средством общения с компьютером, позволяет преодолеть психологический барьер настороженности и в значительной степени раскрепощает сам процесс общения. На этом этапе обычно не возникает проблем, но в дальнейшем, когда получены базовые знания и необходимо их реализовать практически, появляются сложности, которые условно можно разделить на три группы.

Первая — это вопрос среды программирования. Профессиональные трансляторы имеют много возможностей, обычно ненужных новичку, более того, чем сложнее среда, тем труднее в ней разобраться, тем более она отпугивает. Существенно также, что большинство трансляторов имеют англоязычный интерфейс и это затрудняет их использование.

Вторая сложность косвенно связана с первой. Используемые обычно языки, такие, как Бейсик и Паскаль, имеют довольно сложный синтаксис, совершенно излишний при обучении и отвлекающий внимание детей от алгоритма, реализация которого и является их задачей.

И наконец, третья — непосредственно программы и их исполнение. Начиная работать, скажем, с Бейсиком, сразу попадаешь в «сухой язык цифр»: номера строк, двоеточия, переходы. Это делает программы весьма непростыми для восприятия, а попытки использования графики еще более их усложняют. В итоге алгоритм теряется под «завалами» синтаксических конструкций и графических операторов.

Все это разделяет обучение на два этапа: первый — введение базовых понятий, знакомство с простейшими исполнителями; второй — обучение непосредственно языку программирования. Моя разработка была нацелена на обеспечение более плавного перехода между этапами.

В качестве прототипа был выбран язык Лого с его наглядной моделью «Черепашки». Я попытался адаптировать ее к требованиям обучения так, чтобы сохранить простоту и наглядность Лого, для которого накоплен богатый опыт преподавания, и в то же время ориентировать школьников на наиболее популярные языки — типа Бейсика и Паскаля.

Задачи программистского характера разбились на четыре группы: написание

синтаксического разбора алгоритмических конструкций, отображение их графического выполнения, создание простого текстового редактора и интерфейса, обеспечивающего взаимодействие между частями и функциями дискового ввода/вывода. Первоначально программа создавалась на «Ямахе», что накладывало довольно жесткие ограничения на быстродействие и объем занимаемой памяти, с учетом этого языком программирования я выбрал Ассемблер. В результате в течение месяца появилась работающая система, еще столько же времени ушло на интенсивную эксплуатацию с целью выявления мелких, незамеченных «глюков», отлов которых иногда доставляет не меньше проблем, чем само написание программы.

Основу системы составляют базовый набор команд «Черепашьей» графики и простейшие алгоритмические конструкции, но я постарался, насколько это возможно, упростить синтаксис Лого. В результате в идентификаторах не различаются прописные и строчные буквы, разделителем является пробел, группа операторов заключается в квадратные скобки. Упрощена работа с переменными. Они задаются прямым присваиванием и используются только по значению, как и в наиболее распространенных языках. Из операторов я оставил минимально необходимый набор — управление «Черепашкой», операторы выбора, цикла и определения процедур. Единственный тип данных — целые числа. Как показывает опыт, их вполне хватает на данном этапе.

По сравнению со стандартом Лого несколько изменена концепция процедур, вследствие чего они воспринимаются не как вызываемые внешние объекты, а как часть программы. По структуре это ближе к Паскалю и Си.

Важной частью разработки было создание простого и удобного интерфейса. Я решил реализовать его в виде интегрированной среды, включающей в себя текстовый редактор, интерпретатор и систему ввода/вывода. Такая интеграция намного облегчает как сам процесс написания программы, так и ее отладку.

В итоге получилось чуть более 11К байт исполняемого кода, «годного к употреблению». Что же представляет собой получившаяся программа? Пользователь, запустивший ее, первым делом попадает в диалоговый режим. Экран делится на три части. Кроме обычных в Лого окон для ввода команд и непосредственно поля, по которому ползает «Черепашка», я добавил окно, отражающее ее состояние — координаты, направление движения и цвет следа. Другая часть системы — текстовый редактор, в котором набираются исполняемые программы. Учитывая плохое знание новичками клавиатуры, система поддерживает ускоренный набор, позволяющий набрать любую команду нажатием всего двух клавиш. Характерной особенностью системы является поддержка паскалеподобной концепции программы. То есть она рассматривается как единый модуль, и в текстовом редакторе можно набирать и редактировать сразу весь текст, включая процедуры. Сильно помогает и режим пошаговой отладки, при котором интерпретатор печатает каждую выполняемую строку и отображает в окне состояния текущие параметры «Черепашки». В случае ошибки происходит прерывание с выводом соответствующего сообщения. Интерпретатор возвращает вас в текстовый редактор, показывает расположение ошибки и ее краткое описание, по-русски или по-английски. Система получилась довольно удобной и простой в освоении. В то же время она базируется на живом алгоритмическом языке и поддерживает все его основные конструкции, знакомство с которыми упрощает последующий переход к более сложным языкам. Работа в интегрированной среде, с текстовым редактором и отладчиком, формирует практические навыки, необходимые на следующих этапах обучения.

В настоящее время создана версия системы для *IBM PC*, которая в целом соответствует своей предшественнице для «Ямахи», но имеет более удобный интерфейс и использует более богатые возможности нового компьютера.

Хочу поблагодарить преподавателей информатики школы-лицея № 542 Н. М. Леонову, Ю. В. Калмыкова, руководивших разработкой, а также Б. Г. Киселева, М. А. Зайцеву, Л. В. Краеву и других, «пропускавших» программу через учебный процесс и терпеливо относившихся ко всем трудностям, возникавшим при этом. Их рекомендации очень помогли при разработке. Нужно по достоинству оценить и мужество школьников, вдоволь помучившихся с первыми версиями программы.

М. Волконовский,
студент II курса МИФИ

Учебно-игровая программа «Горка»

За два года работы в нашей школе учителем информатики О. Моисеевой было написано немало интересных учебных программ почти по всем предметам. Благодаря чему ребята, начиная с I класса получили возможность работать в компьютерном кабинете. Однако или рука не поднималась, или у юных программистов знаний не хватало, но программ по русскому языку для БК-0010 не было, хотя они очень нужны учителям начальной школы. И вот в результате долгой работы, почти за два месяца: по принципу «с миру по нитке, с каждого по идее» была написана «*UR5.COD*» — игровая, обучающе-контролирующая программа для учеников с I по V класс. (Обкатка программы на ребятах от 11 до 15 лет дала положительные результаты.)

Как же работает программа?

Сев за машину и посмотрев на экран, ученик видит основное меню, из которого может выбрать любую тему: 1) безударные гласные в корне слова; 2) непроизносимые согласные в корне слова; 3) словарные слова.

После нажатия на клавишу (1), (2) или (3) в верхней части экрана пишется выбранная тема и появляется вспомогательное меню:

1. Правило.
2. Контрольная работа.
3. Выбор темы.

Решив повторить правило, ученик нажимает на клавишу (1), и на экран выводится текст доброжелательного содержания, выделенный большими восклицательными знаками по краям. А надпись в нижней части экрана гласит: «Если все понятно, то нажми пробел». Ученику все ясно, он нажимает на пробел и возвращается во вспомогательное меню. Теперь нажав на клавишу (2), он решает проверить свои знания. На экране рисуется стандартная для контрольных картинка.

Цель ученика — благополучно довести человечка до ракеты и улететь на ней. Однако в пути человечка подстерегают неприятности: в случае неправильного ответа облачко станет тучей, закроет солнышко и выльет на человечка дождь, после чего он не сможет пойти дальше. Однако если задание выполнено правильно, то облачко исчезнет, солнышко радуется и переливается, а человечек продолжает путь. Появляется новое задание.

В зависимости от выбранной темы возможно три варианта контрольных работ. Если выполняется контрольная работа по безударным гласным в корне, то в облаке пишется слово с пропущенной буквой. В нижней части экрана появляются все гласные буквы и рисуется человечек с поднятой рукой, которой он показывает на одну из них. Управлять человечком можно клавишами (→), (←) (подведя его к нужной букве, нажать ВВОД).

Если идет проверка знаний по непроизносимым согласным, то ученик при помощи клавиш (↑), (↓) устанавливает мигающую стрелку напротив правильно написанного слова и нажимает ВВОД.

В случае контрольной работы по проверке словаря в облаке пишется слово с про-

пущенными орфограммами, а ученику на приглашение «ВАША ВЕРСИЯ?» необходимо заглавными русскими буквами набрать ответ и ввести его.

В программе во время контрольных работ предусмотрен выход в основное меню, который осуществляется нажатием клавиши (ВС).

По выполнению 20 заданий подводятся итоги. Если все было выполнено правильно, человечек улетает на ракете и появляется надпись «С ТАКИМИ ЗНАНИЯМИ И В КОСМОС МОЖНО». Все это стимулирует ученика не только как-нибудь «проскочить» контрольную, а получить высший балл.

Если ученик не дошел до ракеты, то она снимается со старта, экран очищается и в зависимости от числа правильных ответов дается сообщение: «ДО СТАРТА ОСТАЛОСЬ СОВСЕМ НЕМНОГО» или «В ПУТИ ТЫ ВЕСЬ ПРОМОК» и протокол, имеющий форму: «Пройдено (число правильных ответов) км, до космодрома осталось (число неверных ответов) км.»

Если осталось время до конца урока, то можно поработать и с другой темой, нажав на клавишу (ВС) и вернувшись в начало.

Опыт работы показал, что программа очень удобна для использования на уроках благодаря своей универсальности:

за один присест можно проверить правильность написания 60 слов (по 20 в трех контрольных);

работа словами из словаря выполняет еще роль своеобразного тренажера в освоении клавиатуры. Для самых маленьких это задание хоть и сложновато, но и они выполняют его успешно (ведь на ракете так хочется улететь);

изменив содержимое оператора DATA, можно периодически обновлять словарный запас программы, что позволит поддерживать ее актуальность.

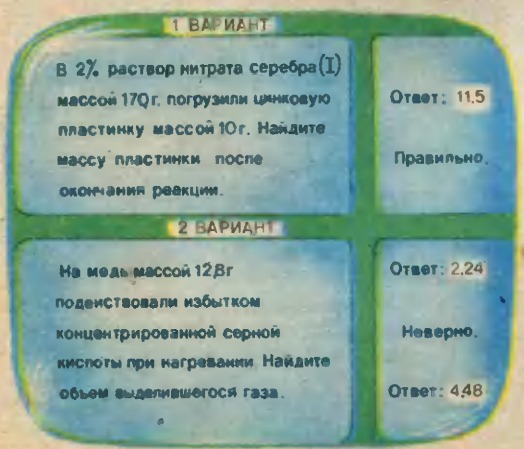
Г. СТИГНЕЕВ,
ученик XI класса СШ № 507 Москвы

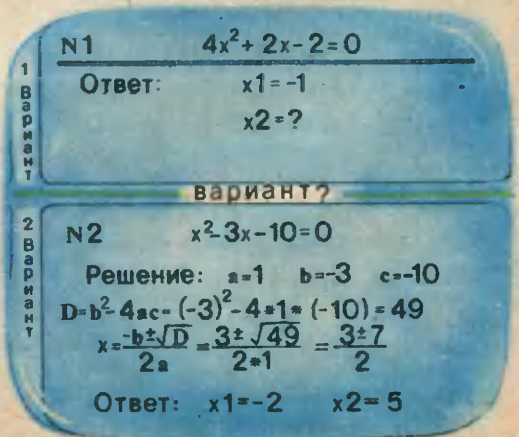
Два ученика — один компьютер

Отсутствие в школах достаточного количества вычислительной техники, а также высокая наполняемость классов вынуждают искать различные способы более рационального и интенсивного использования компьютеров в учебном процессе. Одним из них является программная реализация использования в обучающих и контролирующих программах одного компьютера для независимой работы двух учащихся одновременно. Для этого экран дисплея делится горизонтальной линией на две части и в каждой предъявляются независимые задания для двух учащихся. Выбор первого или второго (верхнего или нижнего) задания осуществляется нажатием клавиши «1» или «2». В данной работе мы опишем одну из программ по химии, реализующих эту идею.

После ввода программы экран дисплея имеет следующий вид:

1 вариант	
Задание1	Ответ?
2 вариант	
Задание 2	Ответ?





Так как задания сразу же высвечиваются на экране, ученики имеют возможность, ознакомившись с ними, выбрать себе вариант для решения, или же его назначает учитель. Подчеркнем, что только после того, как ученик нажмет цифру «1» или «2» — по номеру выбранного варианта,— он может вводить цифровой ответ в соответствующее поле. Сразу после ввода ответа в том же поле высвечивается следующее задание. Если ответ был неверным, то в поле «Ответ» сообщается об этом. В других модификациях этой программы и программ, написанных для других школьных предметов, в частности по математике для учащихся IV—V классов, имеется возможность посмотреть подсказку или снова вернуться к решению той же задачи.

В анализируемой программе для каждого варианта выбирается по три из массива в семь задач. Выбор осуществляется случайным образом. Тексты задач взяты из пособия по химии для XI класса Ф. Г. Фельдмана и Г. Е. Рудзитиса «Основы общей химии» (М.: Просвещение, 1989). Допускается погрешность решения ±0,5. Оценка успешности: все три задачи решены правильно — «5», одна неверно — «4», две неверно — «3», все неверно — «2».

Программа реализована таким образом, что, пока один из учеников решает первую задачу, другой может решить все три и получить свою оценку. Для этого он просто должен каждый раз перед вводом ответа нажимать номер выбранного им варианта, т. е. либо цифру «1», либо цифру «2». Конечно, учащиеся могут вводить ответы и поочередно. После окончания работы высвечиваются оценки, и программу можно запустить снова.

Программа написана для ПЭВМ БК-0010 учеником XI класса московской школы № 462 А. Н. Закарюкиным под руководством преподавателя И. В. Попкова и при участии О. Ю. Ермолаева.

А. Н. ЗАКАРЮКИН,
О. Ю. ЕРМОЛАЕВ

СуперБейсик для УКНЦ

Факт двухпроцессорности УКНЦ ни для кого не секрет, но вот способы «сознательного» использования периферийного процессора (ПП) для большинства — тайна. Такова, увы, особенность документации к компьютеру. Впрочем, возможности ПП там описаны достаточно хорошо, но вот связь ПП и ЦП...

Мы решили упростить сложный механизм связи. Идея такова: создается программа-драйвер, которая помещается в ПП и обеспечивает достаточно простую связь. Эту программу мы присоединили к файлу с Бейсиком. При запуске Бейсика управление передается программе, а уже потом самому Бейсику. В результате ПП становится доступным средствами Бейсика. Полученную комбинацию мы назвали СуперБейсиком.

Немного дополнительных сведений и примеров. Драйвер организует канал связи с адресами (восьмеричными) 450, 452, 454, 456, 176666. Ячейка 450 содержит 1, если ПП готов к приему следующей команды, и 0, если не готов. Ячейка 452 содержит код последней нажатой или отжатой клавиши в формате ПП (см. документацию к УКНЦ). Ячейки 454 и 456 содержат адрес и данные, 176666 служит для передачи команд драйверу.

Список команд приведен ниже.

Перед подачей любой команды, кроме первой, необходимо проверять готовность.

Драйвер занимает около 0,5К байт памяти. С помощью СуперБейсика можно, не зная ассемблера, включать в программы звуковые эффекты, изменять начертания символов и т. п. В нашей школе он быстро завоевал популярность и послужил инструментом для создания многих интересных программ.

```
POKE &0176666,7                 инициализация драйвера

POKE &0454, адрес               занесение слова в ОЗУ ПП
POKE &0456, данные              по указанному адресу
POKE &0176666,1

POKE &0454, адрес               чтение слова из ОЗУ ПП по
POKE &0176666,2                 указанному адресу в ячейку
Z=PEEK(&0456)                   456

POKE &0454, адрес               запуск подпрограммы в машинных
POKE &0176666,3                 кодах с указанного адреса в ОЗУ ПП

POKE &0454, длительность        воспроизведение звука
POKE &0456, частота
POKE &0176666,4

POKE &0454, позиция             печать символа с кодом ASCII от 32
POKE &0456, код                 до 255 в позиции от 0 до 39 в
POKE &0176666,α                 служебных строках (при α=5 - в
                                верхней, при α=6 - в нижней)
```

С. КЛОКОВ, Д. ЛАНИН,
ФМШ № 64, г. Омск

П. ХАТЧЕР

WorldClassroom — многоцелевая международная телекоммуникационная сеть

Питер Хатчер является координатором проекта WorldClassroom («Всемирный класс»), отвечающим за общее управление программной средой сети; преподает специальный курс в школе Webster в Окленде, шт. Калифорния, США; в течение двух лет читал курс по телекоммуникационным проектам для деятелей системы образования в филиале Калифорнийского университета (Беркли). Питер Хатчер является одним из основателей программы обмена учащимися между школами округа Аламида и Советского Союза, которая успешно осуществляется уже четвертый год. В настоящее время он принимает участие в программе профессиональных обменов между региональным отделением (East Bay) ассоциации Computer Using Educators (CUE) и специалистами из СССР, в рамках которой в г. Троицке Московской области проведена совместная конференция в августе этого года.

Хороший педагог способен успешно преподавать практически в любых условиях, тем не менее многие с энтузиазмом используют в работе самые современные средства и доступные ресурсы. Компьютеры и другие технологии в последнее время становятся все более популярными инструментами работников просвещения, их эффективность при наличии высококачественного программного обеспечения оказывается весьма высокой.

В последнее время в США и других странах обсуждаются различные пути внедрения телекоммуникаций в учебный процесс. Проект WorldClassroom, спонсором которого выступает отделение Education Services компании General Telephone and Electronics, дает возможность сделать телекоммуникации существенным компонентом школьной программы. Проект предлагает структурированную образовательную среду, в которой основное внимание уделяется решению поставленной педагогической задачи, а не используемым для ее решения технологиям. Сеть предлагает программы обучения и ресурсы, которые участники в различных точках планеты могут использовать самостоятельно, а потом сравнивать результаты и делиться ими друг с другом. Учителя, стремящиеся расширить собственный кругозор и горизонты своих учеников, получают в распоряжение среду, где целевые уроки высокого качества поддерживаются динамическими массивами данных, помощью экспертов и коллег, причем все это доступно круглые сутки.

Цель проекта — превратить обучение на расстоянии в интерактивное общение в рамках педагогически обоснованной программы. Усилия направлены на воспитание в школьниках интернационального восприятия друг друга через обогащение культурного багажа, через опыт коллективной работы над глобальными проблемами — такими, как мир, голод, окружающая среда.

Проект WorldClassroom имеет свою предысторию. В 1987 г. под названием Foreign Visit («Визит за рубеж») осуществлен его первый этап, в котором приняли участие 20 школ из США, Австрии, Шотландии, Мексики и Японии. Этап продолжался 6 недель в виде ограниченной серии общений. Школьники обменивались письмами, информацией о жизни в своих

регионах, доходах семей, сравнивали цены на общеупотребительные товары и т. д.

В 1988/89 учебном году реализован существенно расширенный вариант «Визита» под названием WorldLink («Всемирный контакт»). Он продолжался 3,5 месяца, охватил около 100 школ из ряда стран, включая СССР. Участниками WorldLink, прежде всего с американской стороны, были высказаны пожелания о большей увязке проекта с учебными планами школ.

На третьем этапе, который проводился под названием SchoolLink («Контакт между школами»), проект был дополнен новыми элементами: программой по общественным наукам, конференциями с участием приглашенных лекторов и др.; продолжен выпуск газеты NewsLink («Обмен новостями»), введенный на втором этапе.

В настоящее время сеть WorldClassroom охватывает школы в США, Канаде, Великобритании, Голландии, Советском Союзе, Германии, Бельгии, Люксембурге, Литве и Перу. В ближайшем будущем этот список пополнится Японией, Испанией, Францией, Аргентиной, Австралией, Саудовской Аравией, Израилем и Индонезией. Предпринимаются активные попытки дальнейшего расширения сети. Проект основан на использовании системы телеконференций «Caucus» и предусматривает доступ по 800 телефонным номерам или через телесеть.

Сеть WorldClassroom действует в рамках пяти секций: знакомство (Welcome), естественные науки (Science), общественные науки (Social Science), искусство владения языком (Language Arts) и специализированные конференции (Special Conferences).

Каждая секция включает несколько конференций, которые обеспечивают возможность взаимодействия участников и проведения дискуссий и содержат подробные планы уроков, данные, полученные пользователем или почерпнутые из других источников, справочную библиотеку по предыдущим проектам.

Пользователи WorldClassroom включаются в сеть через ознакомительные конференции, которые содержат несколько приветственных посланий от ведущих проекта, обзор структуры сети и ее карту, новости о наиболее «свежих» программах и общие сведения о проекте. В этой же секции помещена подробная база данных по всем пользователям сети, раздел для неформальных дискуссий, названный «Учительская» (Teacher's Room), сводка практических советов о том, как лучше пользоваться сетью.

Опыт показал, что новые пользователи любой интерактивной сети нуждаются в персональном контакте и в направляющих подсказках для того, чтобы быстро обрести уверенность и пройти стадию обучения. Новичков приветствует дежурный ведущий, который сопровождает их с помощью ряда сообщений и действий, позволяющих легко и эффективно сориентироваться в системе. Дежурный также помогает новичку понять, какие разделы и кто из участников проекта больше всего отвечают его индивидуальным запросам. В случае необходимости можно воспользоваться телефонной «горячей линией» для получения немедленной помощи. Выдаваемое каждому пользователю справочное руководство содержит всю техническую информацию, описание процедуры быстрого входа в систему и планы уроков.

Конференции проекта в предметных областях дают пользователю богатейший выбор учебного материала. Преследуемая цель не столько в сообщении фактической информации по предмету, сколько в предложении путей проведения исследований непосредственно в каждой школе. Все предлагаемые задания и программы составлены так, чтобы их можно было извлечь из сети, приспособить к индивидуальным требованиям школы, проработать на месте и лишь затем вновь обратиться к сети, чтобы поделиться результатами с другими членами проекта. Наиболее увлекательная часть работы протекает как раз в самих школах, но эту работу поддерживают и обогащают предоставленные сетью возможности.

Во всех конференциях проекта WorldClassroom первый кадр всегда выдержан в одном легко узнаваемом стандарте, он содержит начальные указания о том, как ориентироваться внутри конференции. Кроме того, в каждой конференции обязательно есть указатель и краткий обзор. В обзоре пользователю сообщают, на уровень какого класса рассчитан материал, к какому разделу предмета он относится, каковы преследуемые учебные цели. Хотя эти цели и не подстраивались специально под требования штата Калифорния, все предлагаемые уроки соответствуют рекомендациям штата. Более того, планируется обеспечить вывод на экран информации о соответствии материала учебным планам штата.

Все предметные конференции стимулируют коллективный подход к учебному процессу, многие из них предоставляют возможность междисциплинарного подхода. Мы советуем пользователям максимально индивидуализировать использование ресурсов проекта, приспосабливая их под собственные нужды. Во многих конференциях содержатся предложения о возможностях развития и углубления заданий проекта.

Естественнонаучные конференции пред-

лагают богатый выбор хорошо проработанных и постоянно обновляемых учебных тем. Например, конференция «Землетрясения» представляет собой набор блоков данных, характеризующих картину земных толчков. Сведения ежедневно обновляются на основе информации Национального центра данных по землетрясениям в Денвере (шт. Колорадо). Конференция «Срок жизни» посвящена изучению продолжительности человеческой жизни, причем в качестве исходного материала берутся собранные самими школьниками данные о местной статистике смертности. В конференции «Глобальный климат» ученики анализируют данные температурных измерений по всему миру и исследуют гипотезу о возможной тенденции глобального потепления. В других конференциях изучается погода Соединенных Штатов, вопросы использования энергии, данные о росте и весе людей планеты, наблюдения за бурями. Еженедельный «Трудный вопрос» предоставляет школьникам возможность побороться с нетривиальной научной задачкой, развивающей самостоятельное мышление.

Конференции по общественным наукам имеют выраженную международную направленность. Они ставят целью развить у учащихся планетарное восприятие жизни на Земле. Среди них — «Всемирный альманах», в рамках которого школьники изучают общество, строят гипотезы, предлагают выводы, касающиеся состояния общественного устройства в своей и других странах. В конференции «На рынок» школьники сравнивают и противопоставляют разнообразные экономические структуры и жизненные условия в разных странах. Конференция «Свободно и по справедливости» посвящена изучению функционирования законов в различных политических системах. Постоянную возможность анализа различных общественных явлений и написания сочинений, не ограниченных рамками одной темы, предоставляют «Форум текущих событий» и еженедельные выпуски «Знаменательные высказывания». Все конференции по общественным наукам содержат предложения о мероприятиях с участием родителей, вопросы для предварительных и последующих проверок и написания рефератов.

В конференциях по языку и литературе сами расстояния между участниками используются как важный фактор. Школьникам предлагается рассмотреть в своих сочинениях разнообразие культур и множественность точек зрения. В эту группу конференций входят: «Куда бы прежде всего я повел иностранного гостя», «Мой город», «День из моей жизни». Кроме того, ученики, участвующие в WorldClassroom, регулярно издают международную школьную газету NewsLink, содержащую новости, редакционные комментарии, разнообразные личные мнения, литературное творчество.

Специализированные конференции предлагают участникам широчайший выбор тем. Среди них конференция Techie, посвященная передовым технологиям учебного процесса; «Учительский центр», в котором под руководством специалистов проходят профессиональные дискуссии; «Давайте поговорим» — способ прямого общения участников с профессиональными сотрудниками сети General Telephone and Electronics (GTE). Конференция «Приглашенные докладчики» дает участникам проекта возможность напрямую общаться с интересными людьми со всего мира. В списке гостей второго полугодия 1990 г. были члены Европейского парламента, писатели из Шотландии, исполнительный директор GTE, советский физик, специалист по ветряным мельницам из Голландии, капитан контейнеровоза американской судоходной компании, который ежедневно выходил на связь, пока его судно совершало переход из Калифорнии в Гонконг.

Конечно, же, проект WorldClassroom открыт для новых начинаний. Весной прошлого года была разработана серия мероприятий ко Дню Земли. В WorldClassroom будут обязательно включаться и инициированные самими пользователями проекты. В таком в высшей степени интерактивном окружении, каким являются компьютерные телекоммуникации, просто необходима гибкость среды, немедленно отзывающейся на нужды пользователей. Способность к быстрому отклику ярко проявилась прошлой весной, когда пользователи предложили и тут же реализовали несколько специализированных программ. Среди них были обзор проблем сбора мусора и утилизации отходов, сравнение местных отчетов о деловой активности, краеведческие описания своих регионов.

Сеть WorldClassroom в настоящее время способна передавать файлы в двоичной кодировке, используя один из двух протоколов: Kermit или X-modem. Это позволит нам уже в ближайшем будущем организовать обмен информацией с использованием нескольких языков и организовать конференции, в которых будут использоваться графика, созданные самими пользователями программы, высококачественное шрифтовое и графическое оформление поступающих по сети документов. Возможности проекта неисчерпаемы. Сеть открывает доступ к бога-

тейшим ресурсам, которые к тому же непрерывно совершенствуются для удовлетворения потребностей наших пользователей.

Компьютерные конференции резко увеличивают возможности учителя и учеников — прямо в классе создаются учебные ситуации, взятые непосредственно из реального мира. Цель проекта WorldClassroom — подготовить ученика к XXI в. Из-за экспоненциального роста объема знаний то, что школьники учат сегодня, может устареть уже к моменту окончания ими школы. Поэтому учащиеся должны научиться добывать и обрабатывать постоянно обновляющуюся информацию, чтобы расширять свои знания и принимать решения. Их надо обучать стратегиям решения задач, уделяя особое внимание долгосрочным, глобальным выгодам, а не локальной целесообразности. Может быть, самое важное для наших детей — научиться жить и сотрудничать друг с другом и со всеми соседями по мировому сообществу.

Перевод **А. Ю. СЕБРАНТА**

Урок информатики в безмашинном варианте

Многие учителя, ведущие уроки ОИ и ВТ, сталкиваются с такой проблемой, как использование времени, отведенного для работы на ЭВМ (я имею в виду сельские школы, где нет никакой возможности добраться до вычислительной техники). Предлагаю в таких случаях организовать «эстафету». Суть ее в следующем.

Ученики XI класса готовят вопросы, задания для своих же товарищей. Вопросы могут быть самые разные, например: «Назвать внешние устройства отображения информации» или «Описать принцип работы центрального микропроцессора...» Задача учителя — направить на уроке ученика в нужное русло и проконтролировать по ходу дела его действия. За урок ученик способен составить 2—3 комплекта заданий, в которых 4—5 вопросов по данной теме. Выполненное задание учитель просматривает и передает для ответа учащимся других классов. Ученики, составлявшие задания, проверяют ответы, оценивают их и пишут к ним свою аннотацию. Таким образом, учитель, организуя эту «почту», «убивает сразу нескольких зайцев». Во-первых, занятость на уроке (XI класс тратит на это два, а IX — один урок); во-вторых, неоднократно повторяется материал: в-третьих у учеников развивается чувство ответственности за порученное дело. Эта система очень хорошо работает в сельских малокомплектных школах в классах с малым числом учащихся. Способ организации творческой деятельности учащихся — использование готового раздаточного материала, например комплектов диапозитивов (они всегда имеются в магазинах наглядных учебных пособий).

Готовый комплект учитель разбивает на группы по 2—3 слайда, в зависимости от смысловой информации, и распределяет их среди учащихся для составления к ним сопроводительного текста. После окончания работы слайды вновь собираются в один комплект и проецируются на экран с речевым сопровождением учащихся, подготовивших сообщения. Таким образом получается слайд-фильм. Сделав первый прогон на уроке в том классе, где он готовился, устранив ошибки и недочеты, учитель весь монтаж демонстрирует перед другими классами.

Я уверен, что организация такой «почты» развивает не только чувство ответственности, интерес к предмету, но и одно из самых важных качеств — способность к творчеству. Учителю же отводится роль посредника, организатора творческой деятельности учеников.

О. ФЕДОСЕЕВ,
учитель средней школы с. Отрада
Ульяновской области

О Бейсике и случайных числах

Тесты широко используются на практике для оценки эффективности сложных систем, если возникают трудности с построением и вычислением соответствующих критериев оценки. В основе их использования лежит тот же самый методологический прием, что и при моделировании процесса функционирования системы с целью получения оценочных критериев: конструируется типичная модельная ситуация, в которой систему «заставляют выполнить некоторый объем типичной для нее работы».

С этой точки зрения в заметке «Про Бейсик, быстродействие и кое-что еще» (ИНФО. 1991. № 3. С. 32) в качестве модельной ситуации выступает «прогонка» некоторой программы, а в качестве «объема типичной работы» — 2000-кратное ее повторение.

Очевидно, предположение о том, что тест

содержит достаточно типичные операции, выполняемые повседневно на ПЭВМ, вряд ли близко к действительности. По этой причине показатели из первых двух столбцов приведенной в заметке таблицы допустимо использовать лишь для весьма ориентировочной оценки. Что же касается оценки генератора случайных чисел RND, то она вообще некорректна. Если генератор случайных чисел идеален, т. е. выдает числа, равномерно распределенные в интервале (0,1), то, по центральной предельной теореме теории вероятностей, вероятности того, что при проведении R=2000 испытаний отклонение ΔR будет не больше 5 и не меньше 15, будут $P(\Delta R \leqslant 5) = 0{,}315$, $P(\Delta R \geqslant 15) = 0{,}246$. Таким образом, при однократном испытании предложенного теста даже идеальный генератор случайных чисел с вероятностью 0,246 будет признан плохим и только с вероятностью 0,315 получит отличную оценку.

И еще одно замечание. Часто при оценке программных датчиков случайных чисел отмечают как существенный недостаток то, что порождаемые ими числа не являются «чисто случайными». Однако такая оценка некорректна, если не указывается при этом, для каких целей предполагается использовать получаемые числа. Например, в игровых программах ни закон распределения генерируемых чисел, ни период их повторения не имеют практически никакого значения. В вычислительных задачах важны равномерность распределения чисел в гиперкубах разных размерностей и быстродействие датчика. При этом случайность как таковая не играет никакой роли. Механизм случайности используется только лишь для обеспечения свойства равномерности заданного количества чисел в многомерном пространстве, так как другого метода сделать это не существует.

Ю. СУШКОВ

Возвращаясь к напечатанному

В констатирующей части статьи В. Федотова «Учебные программы в стихии рынка» (ИНФО 1991, № 3) достаточно хорошо описывается сложившаяся ситуация. Надо было бы открыто сказать и про то, что нередко продажа педагогических программно-информационных средств проходит попросту за самые обыкновенные взятки!

Но вот с анализом причин такого положения, и особенно с предложениями по выходу из него согласиться нельзя. Главная причина всех бед школьной информатики — это крайне недостаточная подготовка основной массы учителей. Пока мы кардинально эту проблему не решим, все прочие попытки поправить дело никакого проку не дадут.

Для того, чтобы хоть малые, хоть большие средства тратить разумно, надо и очень хорошо знать сам предмет, и четко представлять его место и роль во всей системе среднего образования. Для проведения «квалифицированной экспертизы» надо иметь хотя бы достаточное число квалифицированных экспертов. А чтобы сама эта экспертиза (даже при высочайшей квалификации экспертов!) не оказалась проведенной впустую, надо чтобы рядовые учителя были готовы освоить те продукты, которые этой экспертизой будут вполне заслуженно одобрены. Ведь не секрет, что иному учителю куда удобнее использовать примитивные поделки, чем серьезные программно-методические разработки.

Нереальными представляются мне и предложения автора по созданию «общесоюзной сети свободного копирования ППС»* и «регулярному проведению опросов пользователей». Причин здесь несколько. Во-первых, функционирование такой сети требует больших трудовых и материальных затрат. На сегодняшний день неясно даже то, как можно было бы обеспечить финансирование этой деятельности, не говоря уже о том, откуда взять необходимое число квалифицированных и беспристрастных исполнителей.

Во-вторых, если бы такая сеть и была создана, вряд ли рядовой учитель смог бы пользоваться ее услугами. Ведь один только перечень имеющихся в этой сети ППИС оказался бы необозримым. Говоря словами автора, мы получили бы здесь громадный склад тех же самых «котов в мешке».

И, наконец, нельзя принять всерьез оценку предлагаемого опроса, как «наиболее объективной и беспристрастной экспертизы». Не говоря уже о том, что сколько-нибудь разумную обработку результатов такого опроса вообще невозможно провести, задумаемся

---

* Уместнее, на наш взгляд, употреблять аббревиатуру ППИС (педагогические программно-информационные средства). Ведь сплошь и рядом не сама учебная программа, а именно используемые ей данные и представляют основную ценность продукта.

над тем, сколь объективны могут быть сами эти отдельные результаты.

Разделяя озабоченность В. Федотова сложившимся ненормальным положением на рынке программных средств, хотелось бы в то же время предостеречь от поверхностных подходов к его исправлению. Мы не вправе тратить время и силы на «борьбу с ветряными мельницами», это будет уводить нас в сторону и тормозить и без того уже крайне запаздывающее решение проблемы.

А. МАТЮШКИН — ГЕРКЕ

ЧТО МОЖЕТ ЭВМ?

Укажи пальцем

Сенсорные экраны, способные «почувствовать», куда пользователь прикоснулся пальцем, используются все чаще. Особенно естественным их использование выглядит при работе с меню или графическим интерфейсом типа Macintosh.

Самыми дешевыми являются резистивно-мембранные экраны, напоминающие мембранные клавиатуры. Две прозрачные пленки, разделенные листом упругого материала, испещренного отверстиями, покрытые напыленными на внутренние стороны прозрачными проводниками — вот и все. Такая простая конструкция позволяет добиться очень высокого разрешения (более 300 точек на дюйм), весьма желательного при работе с графическими системами автоматизированного проектирования и им подобных. Оператор может чертить карандашом на покрытом таким «бутербродом» экране почти как на бумаге. Однако накладная панель не улучшает внешнего вида монитора и ухудшает видимость экрана.

Конструкция, состоящая лишь из одного слоя пленки с электропроводящими покрытиями, нанесенными на внутреннюю и внешнюю ее стороны, заметна меньше. К углам такого экрана подключаются четыре генератора переменного тока четырех высоких частот; при прикосновении пальца к проводящей поверхности емкость тела оператора включается в колебательные контуры этих генераторов и изменяет их частоты, что и позволяет определить точку прикосновения. Это — емкостный экран. Обычно он имеет разрешающую способность 256×256 и значительно более высокую, чем у резистивно-мембранного, долговечность (просто потому, что нажимать на него не нужно, достаточно слегка прикоснуться).

Если вы не хотите заслонять экран дисплея даже очень тонкой и прозрачной пленкой, можете использовать инфракрасный сенсор. Это решетка из инфракрасных лучей, создаваемых светодиодами перед самой поверхностью экрана. Лучи улавливаются фотодиодами, которые, естественно, реагируют на предмет, заслоняющий луч. Разрешение обычно не превышает 3 мм, рама с диодами вокруг экрана выглядит весьма внушительно и стоит довольно дорого.

Самая экзотичная конструкция похожа на предыдущую, но вместо световых пучков в ней используются ультразвуковые. Акустическая волна от пьезогенератора системой отражателей расщепляется на множество пучков, идущих вдоль поверхности экрана. Такой сенсор может реагировать на силу нажатия благодаря тому, что сплющенный об экран палец иначе поглощает акустические волны, чем при легком прикосновении. Однако он непригоден для нерях: небольшие загрязнения поверхности экрана (типа капель воды или жировых частичек) тоже поглощают звук и являются причиной ложных срабатываний.

Телевидение высокой четкости и политика

К дешевым компьютерам (БК-0010, «Микроша» и т. п.) в качестве монитора подключается бытовой телевизор. От бедности. Качество плохонькое, ни цвета, ни контрастности... А вот профессиональный монитор — это да! Скажем, VGA. 640×480!

Недолго осталось существовать такому положению. Еще чуть-чуть (в Европе — лет 7, в Японии — 5), и несчастные обладатели VGA будут жутко завидовать использующим для отображения телевизоры. Естественно, не сегодняшние, а завтрашние — телевизоры высокой четкости. 1250×1440. Каково?

Но ТВЧ — не только зрелище или монитор. Технология теперь вторгается в политику (естественно, через экономику). Например, в США полагают, что ТВЧ сыграет большую роль на президентских выборах 1992 г., поскольку проблемы внедрения нового телевидения упираются в новейшие технологические достижения электроники и имеют непосредственное отношение к конкурентоспособности американской (не только, впрочем, американской) промышленности, а следовательно, взгляды кандидатов в президенты на этот предмет скажутся на отношении к ним публики.

Почему же именно телевидение, а не другие экономически важные технологии? «В конце концов средний избиратель сможет лучше ориентироваться в ТВЧ, чем в динамических запоминающих устройствах с произвольной выборкой» (цитата из американского журнала EUSA).

В. ФРЕЙМАН
Москва

По страницам тематических планов издательств на 1992 год

1. Автоматизация научных исследований и испытание образцов новой техники: В 4 кн.: Практ. пособие / Под ред. Г. К. Круга.— М.: Высш. шк. (вузы и техникумы),— 1992 (II кв.), 20 000 экз.

Книга 1 посвящена вопросам построения, алгоритмизации, исследования математических моделей в АСНИ.

В книге 2 рассмотрены вопросы, связанные с программными средствами в АСНИ, в их числе архитектура и типовые программы, характеристики операционных систем СМ ЭВМ, организация интерфейсов между прикладными программами и базами данных.

Книга 3 содержит сведения о технических средствах АСНИ.

В книге 4 рассмотрены основы проектирования автоматизированных систем научных исследований и комплексных испытаний объектов новой техники.

Для студентов технических вузов, а также специалистов, занимающихся АСНИ.

2. Амосов А. А., Дубинский Ю. А., Копченова Н. В. Вычислительные методы в инженерных расчетах на ЭВМ: Учеб. пособие.— М.: Высш. шк. (вузы и техникумы), 1992 (III кв.).— 15 л.— 1 р. 50 к., 40 000 экз.

Для студентов инженерно-технических специальностей вузов.

3. Белецкий Я. Энциклопедия языка Си: Пер. с пол.— М.: Мир, 1992 (IV кв.).— 36 л., 7 р. 80 к.

Для квалифицированных пользователей ПК и программистов.

4. Беллман Р. Методы вычислений: Избр. главы: Пер. с англ. А. В. Фролова / Под ред. В. Н. Афанасьева, В. Б. Колмановского.— М.: Высш. шк. (вузы и техникумы), 1992 (IV кв.).— 16 л.— 2 р. 50 к., 10 000 экз.

Автор — один из основоположников современной теории управления. Книга посвящена созданию эффективных способов приближенного и численного решения разнообразных задач. В основе развиваемых приближенных и численных алгоритмов лежат методы инвариантного погружения, динамического программирования, теории полугрупп.

Для специалистов по прикладной и вычислительной математике, может служить пособием для студентов соответствующих специальностей.

5. Бессонов В. В. Кружок радиоэлектроники: Пособие для рук. кружков школ и внешкол. учреждений.— М.: Просвещение, 1992 (III кв.).— 13 л.— 1 р. 20 к., 200 000 экз.

Даются рекомендации по организации и методике проведения занятий. Излагаются основы радиотехники, электроники, сведения о микросхемах, элементах цифровой техники, применении микропроцессорных устройств.

6. Браун П. Ч., Мюллен, Норма Д. Английский язык для изучающих вычислительную технику: Курс для обучения чтению: Пер. с англ.— М.: Высш. шк. (вузы и техникумы), 1992 (I кв.).— 15 л.— 2 р. 10 к., 50 000 экз.

В конце пособия приводится базовый словарь, содержащий определение свыше 250 терминов, связанных с вычислительной техникой.

Для студентов технических вузов.

7. Вероятностные методы в вычислительной технике: Учеб. пособие / А. В. Крайников, Б. А. Курдиков, А. Н. Лебедев и др. / Под ред. А. Н. Лебедева, Е. А. Чернявского.— 2-е изд., перераб. и доп.— М.: Высш. шк. (вузы и техникумы), 1992 (IV кв.).— 23 л.— 2 р. 80 к., 20 000 экз.

Для студентов вузов.

8. Вычислительная техника и программирование. Курсовое проектирование: Практ. пособие / П. Н. Шкатов, А. В. Петров, М. А. Титов; Под ред. А. В. Петрова.— М.: Высш. шк. (вузы и техникумы), 1992 (I кв.).— 10 л.— 1 р. 50 к., 100 000 экз.

Изложены методические основы и даны практические рекомендации по выполнению инженерных расчетов в курсовом проектировании. Пособие рассчитано на студентов как пользователей-непрофессионалов в области программирования. Методика работы позволяет осознанно создавать программные разработки и адаптировать готовые ППП для своей предметной области.

9. Геометрия математических моделей.— М.: Наука, 1992 (II кв.).— 10 л.— (Кибернетика: неограниченные возможности и возможные ограничения).— 1 р. 50 к., 30 000 экз.

Для всех интересующихся актуальными проблемами НТП.

10. Гранже М., Менсье Ф. *OS*/2: Принципы построения и установки.— М.: Мир, 1992 (II кв.).— 15 л.— 4 р. 30 к., 50 000 экз.

Для специалистов в области программного обеспечения, а также всех потенциальных пользователей *OS*/2.

11. Давидов П. Д., Марченко А. Л. Бейсик для начинающих.— М.: Наука, 1992 (II кв.).— 12 л.— (Наука и технический прогресс).— 2 р. 40 к., 50 000 экз.

Для широкого круга читателей.

12. Дагене В. А., Григас Г. К., Аугутис К. Ф. Сто задач по программированию: Кн. для уч-ся: Пер. с лит.— М.: Просвещение, 1992 (I кв.).— 15 л.— 1 р. 60 к., 150 000 экз.

Сборник задач с решениями и программами на Паскале. Доступен начинающим.

Для учащихся старших классов и всех интересующихся языком Паскаль.

13. Данкан Р. Профессиональная работа в *MS DOS*: Пер. с англ.— М.: Мир, 1992 (II кв.).— 43 л.— 9 р. 60 к., 30 000 экз.

Для профессиональных программистов, работающих с языками Ассемблер и Си.

14. Делейни У., Ваккари Э. Динамические модели и имитация дискретных событий: Пер. с англ.— М.: Мир, 1992 (IV кв.).— 38 л.— 7 р. 90 к., 5000 экз.

Для студентов и специалистов по теории управления и робототехники.

15. Дуванов А. А., Первин Ю. А. Необычные приключения Пети Кука в Роботландии.— М.: Педагогика, 1992 (IV кв.).— 15 л.— 3 р. 100 000 экз.

Книга вводит в занимательной форме в мир информации. Можно читать ее и без компьютера. В основе книги — программно-методическая система. Книга красочно иллюстрирована.

Для школьников младшего и среднего возраста.

16. Жоль К. К. Логика в лицах и символах.— М.: Педагогика, 1992 (I кв.).— 16 л.— 3 р. 50 к., 100 000 экз.

Книга вводит читателя в мир драматических научных исканий, объясняет связь логики с гуманитарными, естественными, техническими и математическими науками, помогает понять значение логики для программного обеспечения современных компьютеров, вооружает навыками логической грамотности.

Для широкого круга читателей.

17. Журавлев А. П. Звук и смысл: Кн. для уч-ся.— 2-е изд., перераб. и доп.— М.: Просвещение, 1992 (III кв.).— 10 л.— 1 р. 80 к., 100 000 экз.

Книга в доступной форме рассказывает об основанной на экспериментальных данных лингвистической теории содержательности звуковых форм в языке. Ставятся проблемы компьютерной переработки языковой семантики, которая в настоящее время становится центральной в развитии современной кибернетики и информатики.

Во второе издание вошли материалы для самостоятельной и внеклассной работы на компьютере в школе.

Для учащихся старших классов и широкого круга читателей.

18. Защита программного обеспечения: Пер. с англ. / Д. Гроувер, Р. Сатер, Дж. Финкси и др.— М.: Мир, 1992 (III кв.).— 18 л.— 4 р., 20 000 экз.

Предлагаются различные способы защиты ПО: физический, криптографический, интеллектуальный, методы идентификации программ и проблемы юридической защиты авторского права.

Для широкого круга пользователей персональных компьютеров.

19. Инструментальные средства автоматизированного обучения на базе ПК / В. И. Васильев, А. Н. Филиппов, А. Г. Шевченко и др.— М.: Финансы и статистика, 1992 (I кв.).— 12 л.— 2 р., 35 000 экз.

Рассматриваются вопросы создания информационной технологии компьютерного обучения, разработки и реализации инструментальных авторских систем на базе ПК, совместимых с *PC*. Включены вопросы методики проектирования автоматизированных учебных программ по различным дисциплинам.

Описана работа с адаптивными диалоговыми информационными системами РАКУРС и АДОНИС.

Для пользователей-непрофессионалов в области вычислительной техники и программирования, преподавателей и студентов вузов.

20. Инструментальные средства персональных ЭВМ: В 10 кн. / Под ред. Б. Г. Трусова.— М.: Высш. шк., 1992—1993.— 150 00 экз.

В 1992 г. выйдут 4 книги серии.

Первая из них посвящена текстовым редакторам.

Вторая рассматривает вопросы программирования на языке Ассемблер, а также сведения об архитектуре технических средств ПЭВМ, организации памяти, методах адреса-

ции, системе прерываний, адаптерах и контроллерах.

В третьей книге описывается язык программирования Фортран-77.

Четвертая посвящена программированию в среде Турбо Паскаль.

Для широкого круга пользователей ПЭВМ: учащихся техникумов, студентов вузов, аспирантов, инженеров и научных работников.

21. Интеллектуальное программирование: Турбо Пролог и Рефал-5 на ПК / Бабаев И. О., Герасимов М. А., Косовский П. К. и др.— Л.: Изд-во ЛГУ, 1992.— 10 л.— 5 р.,25 000 экз.

Доступно начинающим программистам, в том числе и школьникам.

Обеспечивает быстроту создания и легкость модификации различных систем искусственного интеллекта.

Для специалистов и широкого круга читателей.

22. Искусство разговаривать и получать информацию: Хрестоматия / Сост. Б. Н. Лозовский.— М.: Высш. шк. (вузы и техникумы), 1992 (IV кв.).— 20 л.— 4 р., 50 000 экз.

Для студентов вузов.

23. Каймин В. А. Основы новых информационных технологий: Самоучитель для начинающих работать на ПЭВМ.— М.: Финансы и статистика, 1992 (II кв.).— 12 л.— 5 р., 50 000 экз.

Излагаются основы работы с редактором текстов, базой данных, электронной таблицей, графическим редактором, Бейсиком, Прологом.

Для широкого круга читателей.

24. Клименко Б. И. Универсальный пакет для ПК *Framework*.— М.: Финансы и статистика, 1992 (I кв.).— 14 л.— 4 р. 50 к., 50 000 экз.

Интегрированный пакет с универсальным применением: подготовка и форматирование, распечатка текста, работа с электронной таблицей, построение графиков, создание базы данных, составление программ обработки информации.

Для пользователей ПЭВМ.

25. Когаловский М. Р. Технология баз данных на ПЭВМ.— М.: Финансы и статистика, 1992 (I кв.).— 8 л.— 3 р., 25 000 экз.

Излагаются концепции технологии баз данных — основы разработки информационных систем. Даются краткие сведения о функциональных возможностях ряда наиболее популярных программных продуктов для создания систем баз данных на ПЭВМ, оценка состояния, особенностей и тенденций их развития.

Для разработчиков и пользователей СУБД.

26. Линингмер Ш. (мл.) Вентура Паблишер: советы, хитрости, ловушки: Пер. с англ.— М.: Мир, 1992 (IV кв.).— 31 л.— 6 р. 50 к., 25 000 экз.

Книга — самоучитель по настольным издательским системам. Система *Ventura* используется на ПК типа *IBM PC AT/XT*.

Для издательских работников и пользователей ЭВМ.

27. Компьютерно-игровой комплекс в детском саду: Кн. для работников дошкольных учреждений / Под ред. С. Л. Новоселовой.— М.: Просвещение, 1992 (IV кв.).— 6 л.— 50 к., 600 000 экз.

Даются актуальные примеры использования компьютера в детском саду. Книга знакомит с теоретическими вопросами и результатами экспериментальной работы. Рассматриваются вопросы психологии, санитарии и гигиены, требования к организации занятий, приводятся примерные конспекты игр и занятий с использованием развивающих и обучающих компьютерных программ.

Для работников детских садов и родителей.

28. Компьютерные лицеи — сегодня и завтра.— М.: Наука, 1992 (III кв.).— 10 л.— (Кибернетика: неограниченные возможности и возможные ограничения).— 1 р. 80 к., 30 000 экз.

Авторы сборника — советские и зарубежные специалисты по компьютеризации образования.

Для широкого круга читателей.

29. Кусов И. Ф. Элементарная математика и персональный компьютер.— М.: Наука, 1992 (I кв.).— 10 л. (Кибернетика: неограниченные возможности и возможные ограничения).— 1 р. 80 к., 30 000 экз.

Излагается связь элементарной математики с работой персонального компьютера, понятиями алгоритма и программы. Дается представление о возможностях языков Бейсик, Фортран, Паскаль для решения математических задач: решение уравнений, логические задачи, игры с ЭВМ.

Для широкого круга читателей, в том числе учащихся школы.

30. Либерман Л. В. АРМ бухгалтера на персональной ЭВМ.— М.: Финансы и статистика, 1992 (II кв.).— 6 л.— 1 р., 30 000 экз.

31. Ляхович В. Ф. Руководство к решению задач по основам информатики и вычислительной техники: Практ. пособие.— М.: Высш. шк. (вузы и техникумы), 1992 (III кв.).— 12 л.— 1 р. 50 к., 150 000 экз.

Пособие предназначено для получения и

закрепления навыков практической работы по алгоритмизации, программированию на языке Бейсик и формализации задач, а также общению с ЭВМ в режиме диалога.

Для учащихся техникумов. Могут пользоваться учащиеся средних школ, ПТУ, широкий круг читателей.

32. Макетирование, проектирование и реализация диалоговых информационных систем / Л. И. Гуков, Е. И. Ломаков, А. В. Морозов и др.— М.: Финансы и статистика, 1992 (I кв.).— 25 л.— 5 р. 50 к., 30 000 экз.

Приводится взаимосвязанная последовательность этапов проектирования и реализации информационных систем на основе баз данных. В качестве инструментальных средств приводятся *d Base* IV, *Paradox, CLAPION* и *Du COD*.

Дается сравнительная характеристика языков программирования четвертого поколения и генераторов приложений.

Для специалистов по разработке и эксплуатации программного обеспечения информационных систем.

33. Мак Иннес Дж. Компактный справочник для начинающих пользователей *dBase* III *Plus:* Пер с англ.— М.: Финансы и статистика, 1992 (II кв.).— 5 л.— 3 р. 50 к., 100 000 экз.

Для пользователей ПЭВМ и разработчиков баз данных.

34. Мамиконов А. Г. Теоретические основы автоматизированного управления: Учеб.— М.: Высш. шк. (вузы и техникумы), 1992 (II кв.).— 22 л.— 2 р. 60 к., 30 000 экз.

Рассмотрены вопросы системного анализа и методологии проектирования АСУ. Приведены сведения о методах принятия решений.

Для студентов вузов и практических работников.

35. Марцеллус Д. Проектирование экспертных систем на Турбо Прологе: Пер. с англ.— М.: Финансы и статистика, 1992 (III кв.).— 17 л.— 6 р., 100 000 экз.

Для программистов, создающих интеллектуальные программы и экспертные системы.

36. Массел Б. и ассоциация ФИЛИПП САИЕР. Овладевайте пакетом *Side Kick Plus*: Пер. с англ.— М.: Мир, 1992 (III кв.).— 25 л.— 5 р. 30 к., 25 000 экз.

Мощное средство компьютерной поддержки для всех, кто занимается организационными вопросами.

Для широкого круга пользователей ПЭВМ.

37. Машинный интеллект и инженерия знаний в робототехнике: Пер. с англ.— М.: Наука, 1992 (II кв.).— 10 л. (Кибернетика: неограниченные возможности и возможные ограничения).— 1 р. 80 к., 30 000 экз.

Сборник включает статьи известных специалистов по наиболее перспективным сегодня направлениям робототехники. Это системы технического зрения и обработки информации с использованием методов искусственного интеллекта, способы планирования операций и организаций систем управления роботов.

Для специалистов, преподавателей, студентов.

38. Мизрохин С. В. *Turbo PascaL* и объектно-ориентированное программирование.— М.: Финансы и статистика, 1992 (I кв.).— 7 л.— 3 р. 50 к., 30 000 экз.

Вопросы программирования в новой технологии — объектно-ориентированного программирования. Описана работа с файлами, управление консолью, графика, работа с «мышью» и т. д.

Для программистов, специалистов по ВТ, студентов.

39. Морозов В. К., Маляров А. Н. Моделирование информационных сетей: Учеб.— М.: Высш. шк. (вузы и техникумы), 1992 (I кв.).— 10 л.— 1 р. 05 к., 15 000 экз,

40. Неймарк А. М. Микроэлектроника: Возможности и перспективы использования.— М.: Наука, 1992 (I кв.).— 10 л. (Наука и технический прогресс).— 1 р. 50 к., 30 000 экз.

Для специалистов в области создания и внедрения микроэлектронной техники.

41. Никольский А. Н., Мусин К. А. Автоматизированное рабочее место в среде *Windows* 3.0.— М.: Финансы и статистика, 1992 (I кв.).— 24 л.— 6 р., 50 000 экз.

Излагается технология работы со средой *Windows* 3.0, электронной таблицей *Excel*, текстовым процессором *Word*, базой данных типа *Omnis 5* и др. Описывается обмен данными между программными продуктами и методика их совместного использования.

Для широкого круга пользователей ПЭВМ.

42. Перегудов Ф. И., Тарасенко Ф. П. Основы системного анализа: Учеб.— 2-е изд., доп.— М.: Высш. шк. (вузы и техникумы), 1992 (I кв.).— 25 л.— 2 р. 80 к., 25 000 экз.

Для студентов высших учебных заведений.

43. Проблемы разработки больших программных систем.— М.: Наука, 1992 (III кв.).— 10 л. (Кибернетика: неограниченные возможности и возможные ограничения).— 1 р. 80 к., 30 000 экз.

В сборнике представлены новые данные в областях информатики: информационные

модели, накопление и использование программистских знаний, интегрированные среды разработки программ, ЭВМ нетрадиционной архитектуры.

Для широкого круга читателей.

44. Программирование на языке Паскаль для *MSX*-компьютера: Учеб. пособие для студ. пед. ин-тов / Р. Н. Зарогский, А. Н. Ломакович, Ю. С. Рамский и др.— М.: Просвещение, 1992 (II кв.).— 13 л.— 1 р. 50 к.

Одно из первых пособий по Паскалю для *MSX*-компьютеров «Ямаха» и «Корвет». Даются основы программирования в двух версиях Turbo *Pascal* и *IRTPascal*. Доступная форма, подробные пояснения основных конструкций, отличий двух версий, работа со спрайтами, вывод графической информации. Теоретическое изложение иллюстрировано большим количеством программ.

Для преподавателей, студентов, учащихся и других пользователей КУВТ.

45. Рено Ж. Элементарный Турбо Паскаль для физики: Пер. с фр.— М.: Мир, 1992 (II кв.).— 15 л.— 3 р. 20 к., 7000 экз.

Для начинающих программистов, в основном физиков. Даются структура и синтаксис языка, типовые программы на физическом материале, сложные вычисления.

46. Рыбкин Е. Н., Юдин А. Ю. Программные средства ПЭВМ. Операционная система *MS — DOS*.— М.: Финансы и статистика, 1992 (I кв.).— 8 л.— 2 р., 50 000 экз. Книга сопровождается дискетой с обучающей системой по *MS — DOS* (стоимость дискеты 25 р.).

Для широкого круга читателей.

47. Скляров В. А. Применение ПЭВМ: В 3 кн.: Практ. пособие.— М.: Высш. шк. (вузы и техникумы), 1992 (II кв.).— 100 000 экз.

Первая книга серии посвящена описанию структуры персональной ЭВМ и способов управления ее ресурсами.

Во второй книге рассмотрены широко распространенные операционные системы для ПЭВМ: *MS — DOS* и *OS*/2.

В третьей книге рассмотрены популярные современные пакеты: инструментальные системы программирования, интегрированные базы данных семейства *d Base*, программы для решения различных задач автоматизации проектирования устройств ВТ и автоматизации обучения.

Для студентов вузов и широкого круга читателей.

48. Советов Б. Я. Информационная технология: Учеб.— М.: Высш. шк. (вузы и техникумы), 1992 (II кв.).— 25 л.— 2 р. 90 к., 40 000 экз.

Для студентов вузов и практических работников.

49. Структуры, хаос, вычислительный эксперимент.— М.: Наука, 1992 (III кв.).— 10 л.— (Кибернетика: неограниченные возможности и возможные ограничения).— 1 р. 80 к., 30 000 экз.

Авторы книги наглядно показывают, как тесно связаны порядок и хаос. Рассматриваются парадоксы движения жидкости и загадочные свойства плазмы, удивительные химические реакции и прошлое Земли. Применение компьютеров позволяет увидеть единство в многообразии этих процессов.

Для специалистов, преподавателей, студентов и школьников.

50. Тондо К., Гимпал С. Задачи по языку Си: Пер. с англ.— М.: Финансы и статистика, 1992.

Комментарии и ответы к упражнениям в книге Б. Кернигана и Д. Ритчи «Язык Си: справочное руководство для программиста» (Финансы и статистика, 1991).

Для широкого круга пользователей профессиональных ПК.

51. Трифонов Н. П., Громыко В. И. Стили в программировании.— М.: Наука, 1992 (II кв.).— 10 л.— (Кибернетика: неограниченные возможности и возможные ограничения).— 1 р. 80 к., 30 000 экз.

В книге рассмотрены разные стили программирования в их сравнении: императивный, функциональный, логический, концептуальный, объектно-ориентированный, а также сопутствующие общие вопросы типизации: бестиповость, сильная типизация, абстрактные типы данных.

Для специалистов различных областей науки.

52. Файтс Ф., Джонстон П., Кратс М. Компьютерный вирус: Проблемы и прогноз: Пер. с англ.— М.: Мир, 1992 (II кв.).— 12 л.— 2 р., 50 000 экз.

Для программистов разной квалификации.

53. Фигурнов В. Э. Учимся работать с *IBM PC*.— М.: Финансы и статистика, 1992 (I кв.).— 23 л.— 10 р., 30 000 экз.

О компьютере *IBM PC* и операционной системе *MS — DOS*, оболочке Norton *Commander*, вспомогательных программах, мерах по борьбе с вирусом, о редакторе текстов ЛЕКСИКОН.

Для широкого круга читателей.

54. Фодор Ж., Бонифас Д., Танги Ж. Операционные системы от *PC* до *PS*/2: Пер. с фр.— М.: Мир, 1992 (I кв.).— 17 л.— 3 р. 80 к., 25 000 экз.

Для инженеров, студентов, школьников старших классов, работающих с ПЭВМ.

55. Херш Дж., Херш К. Работа с *Oracle* версии 6.0: Пер. с англ.— М.: Мир, 1992 (III кв.).— 23 л.— 5 р. 50 к., 20 000 экз.

Руководство по проектированию и построению реляционных баз данных, а также управление их работой средствами СУБД версии 6.0. Описаны потенциальные возможности многопользовательских систем.

Для пользователей ПЭВМ.

56. Чоговадзе Г. Г. Персональные компьютеры: 2-е изд., перераб. и доп.— М.: Финансы и статистика, 1992 (II кв.).— 15 л.— 6 р., 30 000 экз.

В системном виде представлена справочная информация по ПК.

Для широкого круга пользователей.

57. Шелест В. Д. Структурный подход к программированию на Фортране, Бейсике и Паскале: Учеб. пособие.— Л.: Изд-во ЛГУ, 1992 (III кв.).— 18 л.— 6 р., 10 000 экз.

Книга учит созданию легкочитаемых, надежных, универсальных, транспортабельных программ. Все примеры решены на ПК.

Для студентов младших курсов и всех, изучающих программирование.

58. Энциклопедический словарь юного информатика / Сост. В. М. Абрамов.— М.: Педагогика, 1992 (III кв.).— 40 л.— 8 р., 500 000 экз.

Для старшеклассников.

59. Юдилевич А. Я. Прогулка с микрокалькулятором: Кн. для учащихся.— М.: Просвещение, 1992 (II кв.).— 5 л.— 35 к., 100 000 экз.

В доступной и увлекательной форме учащимся вспомогательных школ предлагаются игры и занимательные упражнения, которые помогают им быстро освоить клавиатуру простейшего микрокалькулятора и решать задачи.

Для учащихся IV—IX классов вспомогательной школы.

Напечатано в 1991 г.

Общие вопросы

Методика обучения

Кабинет ВТ

Клуб БК

ВЕСЕЛЫЙ УРОК

Непрерывность процесса прогресса постоянно порождает новое — если не изделия, то хотя бы термины. В этом номере мы публикуем дополнение к общедоступному словарю (см.: ИНФО. 1990. № 1).

А. ЖИРОВ

Общедоступный словарь по вычислительной технике

Новые термины

Автоматизированная обучающая система (АОС) — программно-аппаратно-методическое образование, через которое обучаемый и преподаватель обмениваются кодами нажимаемых клавиш. Автоматизирует процесс раздачи и получения образования.

Арифметическо-логическое устройство (АЛУ) — представитель бурно эволюционирующей на Земле ветви устройств, в корне не отличающихся от био-логических.

Выводы микросхемы — металлические проводники, по которым из корпуса микросхемы выводятся наружу логические выводы, сделанные у нее внутри.

Макрокоманда — команда, не помещающаяся в микропроцессоре и собираемая для исполнения из отдельных деталей.

Микрокоманда — миниатюрная команда; десятки их легко умещаются в корпусе микросхемы и используются для формирования любой нужной команды необходимого формата.

Оператор ЭВМ — наиболее оконечное устройство вычислительной системы, подсоединяемое к ее терминалам. Как следствие, имеет наиболее низкую стоимость

часа работы среди других звеньев ВС. Распространение персональных компьютеров делает это устройство устаревшим.

Программист — вспомогательное звено в человеко-машинной вычислительной системе, осуществляющее предварительную подготовку команд и данных для работы процессора. Утрачивает свое значение после окончательного запуска системы в работу.

Устройство отображения информации — периферийное устройство, служащее для быстрого преображения информации, приготовленной процессором, в изображение, поглощаемое жаждущим пользователем, и таким образом предохраняющее информацию от брожения.

В. ТИХОНОВ

Кот, канарейка и программист.

Однажды Программист пошел воды напиться.
И надо же тому случиться, на беду,
Компьютер он оставил на виду
У толстого ленивого Кота.
У рыжего моментом скукота
Вдруг улетучилась,
И, приемов в пять,
Кряхтя, на стол взвалился он,
И ну гулять
По кнопкам ЭВМ
И за изящной мышкою гоняться.*
Компьютер запищал,
Тараща свой дисплей на жирного Кота,
Шкалою заморгал, пытался защищаться.
Но тщетно.
Гибкий диск устал вращаться:
То вбок,
То в бег,
То вскачь,
То вверх.
Процессор верещал, как дикий стерх,
ОЗУ и ПЗУ сплелися проводами,
А принтер раскаленными болтами
Бумагу жег.
Но от когтистых ног
*Покоя все не знал ки-борд***
И может быть, не быть компьютеру в живых,
Да Канарейка, заскучав на жерди,
И [illegible]летки выпорхнув,
Насвистывая Верди,
Воспользовалась тем, что у Кота
Все лапы заняты,
И ну щипать
Конец его хвоста.
Кот заорал и носом — в дисковод.
А там его мотором так промяло,
Что вырвался едва он.
Да...
Немало
Холеной шерсти
На диск системный намотало.
Очухавшись едва, все беды
Кот Канарейке приписал
И уж почти ее достал,
Но тут, попив, поспав и плотно пообедав,
Наш Программист вошел
И их застал.

* * *

Извлечь мораль из басни просто:
Кот должен есть мышей, а Канарейка — просо.
А Программист обязан знать,
Что, уходя попить,
Компьютер надо выключать,
Кота на крышу выпускать,
А Канарейку — запирать.
И уж потом спокойно спать.

* *Мышь — коробочка с кнопками.*
** *Ки-борд — коробочка побольше.*

ЧТО? МОЖЕТ? ЭВМ

Дайте ему слово!

Технологический процесс позволяет сейчас втиснуть персональный компьютер класса PC XT вместе с клавиатурой и экраном в объем спичечного коробка; но удобно ли будет пользоваться такой игрушкой? Если же использовать все возможности при создании обычной настольной конструкции, по вычислительной мощности она превзойдет суперкомпьютер пятилетней давности. Встает вопрос: для чего бы эту мощность использовать?

Улучшение графических возможностей компьютера лимитировано качеством мониторов: их возможности значительно уступают потенции микропроцессоров. «Блокнотный» ввод данных, при котором пользователь просто от руки пишет приказы компьютеру на специальной чувствительной плоскости, как на бумаге, пока еще слишком сложная задача. Значит, что? Значит, голос!

Простейший подход — использование заранее заготовленных файлов, содержащих оцифрованные и сжатые тексты (обычно фразы), надиктованные человеком. Они часто используются в системах автоматической телефонной связи, в автомобильной электронике и т. п. В несколько более сложном методе используется словарь заранее записанных слов, из которых компьютер составляет фразу.

Наиболее сложным и универсальным является синтез речи из фонем, реализованный, кстати, на БК-0010, хотя и не с тем качеством произношения, которое удовлетворило бы International Telephone & Telegraph. Ту же работу выполнила фирма GW Micro и выпустила в продажу три портативных компьютера для слепых со встроенными синтезаторами речи. По сравнению с «немыми» компьютерами такого же класса они стоят вдвое дороже.

Ну а «слух» компьютера? Здесь феноменального успеха добилась японская фирма NEC, предложив систему синхронного перевода с японского на английский и обратно. Система не требует настройки на голос оператора, а переводит речь любого человека, рассчитана на обработку слитной продолжительной речи и производит перевод с той же скоростью, с какой «слышит» речь!!! Любопытно, что фактически осуществляется два перевода: сначала в искусственный промежуточный язык и лишь потом в произносимые фразы.

АВК идут на смену ЭВМ

АВК — это аудиовидеокомпьютеры. На первый взгляд просто персональные ЭВМ с расширенным набором внешних устройств, включающим оптический дисковод (он же проигрыватель видеодисков) и высококачественную систему звуковоспроизведения. А на второй...

Обработка видеоинформации в реальном времени (что, кстати, требует больших вычислительных мощностей и стимулирует развитие компьютерных технологий), возможность оперировать речевыми данными, музыкальная периферия высшего качества не только позволяют создавать на АВК существенно более эффективные в части пользовательского интерфейса программы (представьте, например, как АВК может учить иностранному языку, не только демонстрируя правильные произношения и артикуляцию, но и оценивая произношение учащегося!), но и позволит, наконец, компьютерам проникнуть в каждый дом, «притворившись» бытовой аппаратурой.

Подключайтесь!

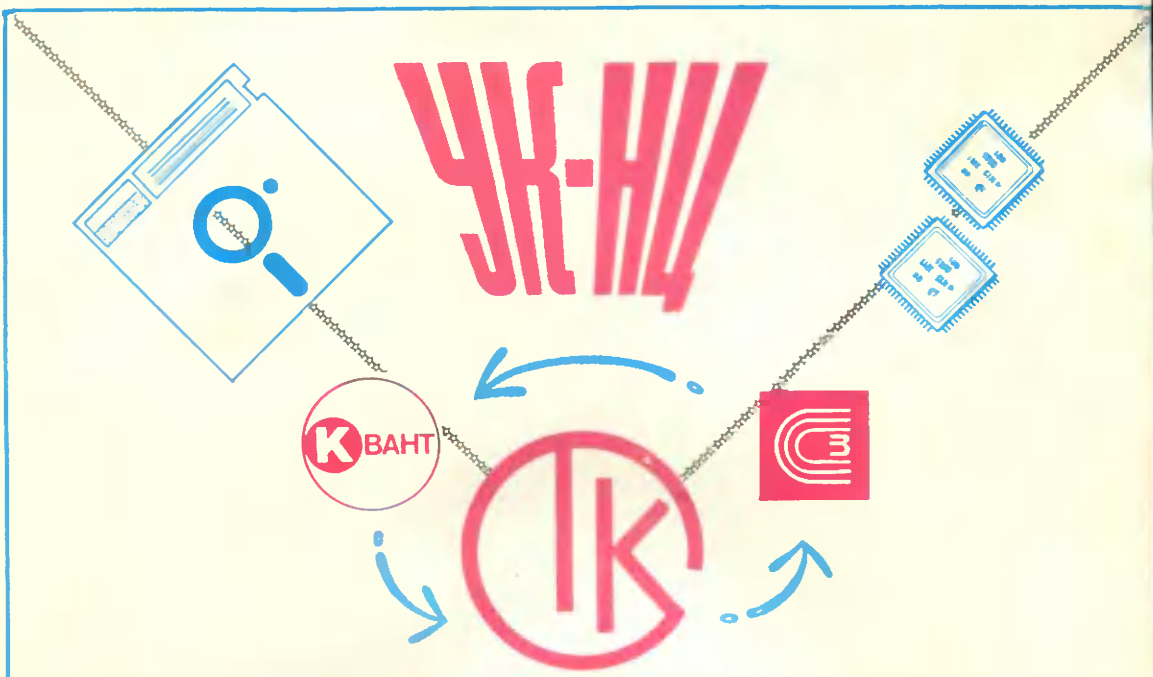
Описания систем телекоммуникаций (наподобие WorldClassroom) воспринимаются как экзотика, вести из будущего. Однако в СССР есть уже около 100 BBS (Bulletin Board System) — любительских центров сбора и распространения компьютерных данных. Это телефоны, к которым подключены компьютеры. Сюда может позвонить каждый желающий (компьютер), оставить и получить свежую информацию. Многие из BBS входят в международные сети, например FIDO — всемирную любительскую сеть.

К сожалению, первый «блин» оказался комом. Когда несколько телефонов BBS были обнародованы, на них обрушился шквал звонков — порой не во время, предназначенное для связи (а ведь это зачастую домашние телефоны!), порой не обращая внимания на протесты... Поэтому мы опубликуем телефоны BBS не здесь, а в одном из следующих номеров, удостоверившись, что эти центры функционируют, и сопроводим «правилами хорошего тона» для желающих использовать электронную почту.

И голос можно сжать

Плата Voice Card нидерландской фирмы Sirex вместе с соответствующей программой позволяет записывать на дискету емкостью 720 Кбайт речь длительностью... Давайте прикинем сами. Самый экономный (из дающих еще терпимый результат) вариант оцифровки речи — 8000 раз в секунду на два уровня, или 1000 байтов в секунду — округленно 1К байт. Следовательно, 720 К хватит на 12 мин.

Не угадали! Методы сжатия оцифрованного сигнала (о них мы как-нибудь расскажем) позволяют Voice Card записать на дискету в 2,5 раза больше — 30 мин речи и с более хорошим качеством.



МАЛОЕ ПРЕДПРИЯТИЕ "ТЕХНОКОМ"

предлагает:

- **СИСТЕМА ПОСТРОЕНИЯ КУРСОВ ОБУЧЕНИЯ И КОНТРОЛЯ ЗНАНИЙ УЧАЩИХСЯ;**
- **СИСТЕМЫ ПОДГОТОВКИ ТЕКСТОВОЙ И ГРАФИЧЕСКОЙ ИНФОРМАЦИИ;**
- **СИСТЕМА СОЗДАНИЯ НАЦИОНАЛЬНЫХ АЛФАВИТОВ С ВОЗМОЖНОСТЬЮ ПЕЧАТИ;**
- **ЭФФЕКТИВНЫЕ СЕТЕВЫЕ ПРОГРАММНЫЕ СРЕДСТВА;**
- **КУРСЫ ПОДГОТОВКИ ПРЕПОДАВАТЕЛЕЙ ПО ЭКСПЛУАТАЦИИ УЧЕБНЫХ КЛАССОВ;**
- **КУРСЫ ПО ОБСЛУЖИВАНИЮ И РЕМОНТУ УК — НЦ;**
- **РЕМОНТ МС 0511;**
- **СТЕНД ДИАГНОСТИЧЕСКИЙ ПО РЕМОНТУ МС 0511;**
- **ПОСТАВКА РАЗЛИЧНЫХ МОДИФИКАЦИЙ УЧЕБНЫХ КЛАССОВ С Ч/Б И ЦВЕТНЫМИ МОНИТОРАМИ;**
- **РАСШИРЕНИЕ ПАМЯТИ — ЭЛЕКТРОННЫЙ ДИСК (1 Мбайт);**
- **КАССЕТА ПЗУ (БЕЙСИК);**
- **АДАПТЕР ТЕЛЕВИЗИОННЫЙ (МОДУЛЯТОР) ДЛЯ ПОДКЛЮЧЕНИЯ УКНЦ К ЦВЕТНОМУ ТЕЛЕВИЗОРУ.**

ЗАЯВКИ НАПРАВЛЯТЬ ПО АДРЕСУ :
103460 г. Москва (з-д "Квант") МНПП "Техноком"

Тел. 536-50-90
536-50-51
536-60-29

Р/с 609706 в Зеленоградском филиале МИБ,
МФО 201478, код банка 31

Куп. 10-11

Цена 1 р. 20 к.

70423

ИНФО

6'91

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

ISSN 0234—0453. Информатика и образование, 1991, № 6, 1—128.