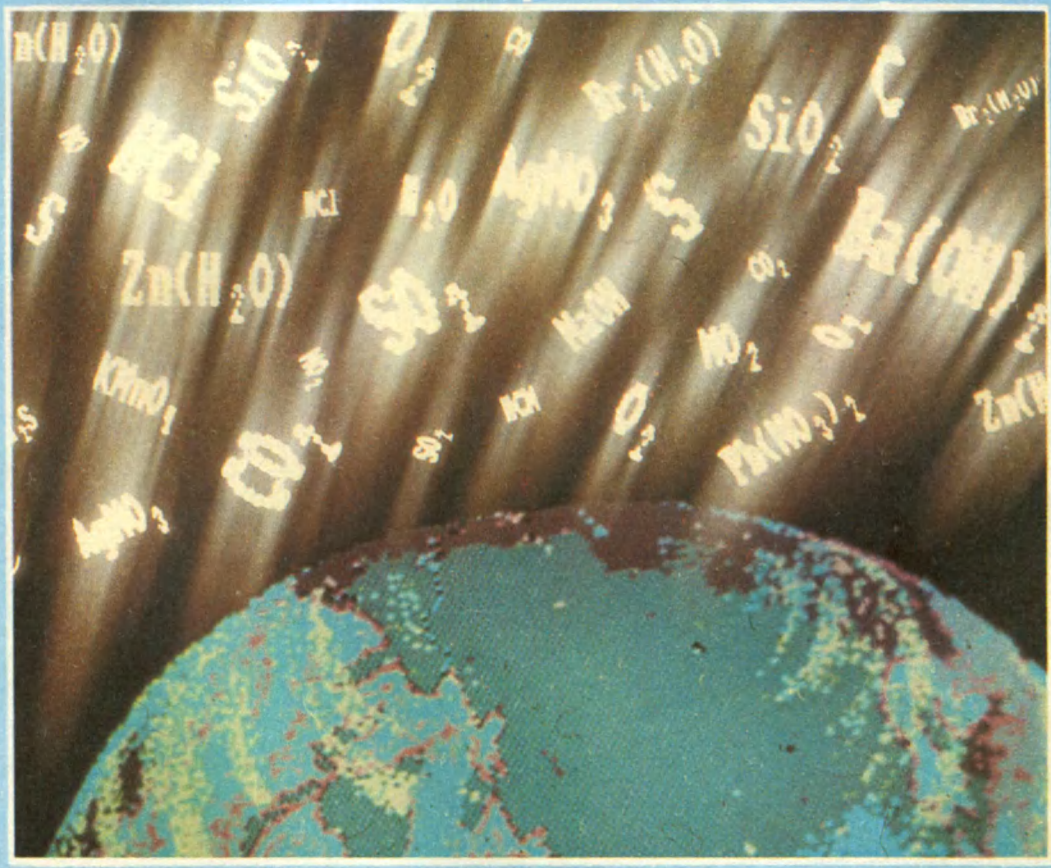
ISSN 0234—0453

ИНФОРМАТИКА И ОБРАЗОВАНИЕ

2 1990



Механико — математический факультет МГУ (ЛВМ, «Аттик»), НСК АН СССР (лаборатория ПО учебного процесса), предприятие «ИнфоМир»

ОБЪЯВЛЯЮТ СВОБОДНЫМИ ДЛЯ КОПИРОВАНИЯ

все свои программные средства для КУВТ «Ямаха» MSX-1, MSX-2, датированные 1989 годом и ранее, в том числе

«Е-практикум» и «МикроМир-88»

Это означает, что если программа для КУВТ «Ямаха» датирована 1989 годом или ранее, а в ее заставке фигурируют **«МГУ, мехмат, ЛВМ»**, или **«Аттик»**, или **«НСК АН СССР»**, или **«ИнфоМир»**, то Вы можете использовать эту программу в любых некоммерческих целях, а также (на некоммерческой основе) копировать и передавать ее любым другим организациям и частным лицам.

Предприятие **«ИнфоМир»** подготовило и распространяет специальную дискету с последними версиями всех свободно копируемых программ.

Дополнительную информацию можно получить по следующим адресам:

117234, Москва В-234, Ленинские горы, МГУ, мехмат, ЛВМ, к. 1310, тел. 939-17-86

109117, Москва, ул. Маленковская, д. 19/17, предприятие «ИнфоМир», тел. 264-30-22

191186, Ленинград, Ломоносова, 11, ЛГИУУ, тел. 314-25-15, Медведь Б. С.

252601, Киев, ул. Героев Революции, д. 5, комн. 8 (пн., чт. 15—17 ч.) предприятие «ИнфоМир», тел. 228-33-86

220809, Минск, Советская 18, Минский педиститут, РУМК, к. 520

226098, Рига, ул. Горького, 10, Латвийский ГИУУ, каб. ИВТ

232034, Вильнюс, ул. Дидлаукио, 82, Литовский ИУУ, каб. ИВТ, тел. 77-29-64

185640, г. Петрозаводск, просп. Ленина, 33 Петрозаводский государственный университет, ИВЦ, тел. 714-61, 714-81

344700, г. Ростов-на-Дону, пл. Стачки, 10, Ростовский областной ИУУ, каб. ИВТ

670000, г. Улан-Уде, ул. Смолина, 24а, Бурятский пединститут, каф. ИВТ; тел. 2-38-04

360004, г. Нальчик, ул. Чернышевского, 173, Кабардино-Балкарский госуниверситет; каб. ИВТ, тел. 2-22-47

Научно-методический журнал
Государственного комитета СССР
по народному образованию

Издается с августа 1986 г.
Выходит шесть раз в год
Москва «Педагогика» 1990

ИНФ
ИНФОРМАТИКА И ОБРАЗОВАНИЕ

2'90
март — апрель

Содержание

НИТ в дошкольном образовании

Молодежная инициатива

Точка зрения

Зарубежный опыт

Репортаж номера

Нам пишут

Информация

Веселый урок

Обложка *Э. Бажилина*

Редактор отдела *А. Кравцова*
Научный редактор *Т. Драгныш*
Зав. редакцией *Н. Игнатова*
Художественный редактор *Л. Розанова*
Корректоры *О. Пурлова, М. Суворова*

Сдано в набор 23.01.90. Подписано в печать 01.03.90. А 06340. Формат 70×100¹/₁₆. Бумага офсетная № 1. Печать офсетная. Усл. печ. л. 10,40. Усл.-кр. отт. 42,88. Уч.-изд. л. 12,86. Тираж 72 225 экз. Заказ 137. Цена 60 коп.

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

Адрес для переписки: 107005, Москва, Лефортовский пер., 8.

Адрес редакции: Студенческая ул., 19, корп. 1, кв. 17.
Телефон редакции: 249-97-77.

Ордена Трудового Красного Знамени Чеховский полиграфический комбинат Государственного комитета СССР по печати.
142300, г. Чехов Московской обл.

П. БРУСИЛОВСКИЙ
ВНТИЦентр, Москва

Языки для обучения основам программирования

На мой взгляд, результатом изучения основ программирования в школе должно стать не зазубривание синтаксиса конкретного языка, а усвоение понятий: алгоритм, исполнитель, программа; знакомство с управляющими структурами языков программирования, методикой разработки программ и т. д. Однако надежное усвоение общих понятий и принципов возможно только в процессе изучения и использования «живого» языка программирования, подкрепленном самостоятельной работой на ЭВМ. Данная статья — попытка ответа на вопрос, какой язык выбрать для начального обучения программированию.

В настоящее время для знакомства с основами программирования в качестве «живого» чаще всего выбирается язык программирования, имеющийся на доступной ЭВМ. В лучшем случае это Паскаль, чаще, к сожалению, Бейсик. Такой выбор ставит перед новичками ряд препятствий.

Во-первых, в «концептуальный базис» языка, наряду с основными понятиями программирования, входит и множество второстепенных, отражающих тонкости данного языка и его реализации. Это затрудняет усвоение материала, мешает организации знаний в правильную структуру, так как детали и тонкости конкретной версии заслоняют общие понятия и принципы.

Во-вторых, основные действия — вычисление и присваивание не являются наглядными. В результате процесс выполнения программы практически скрыт от обучаемого. Отсутствие наглядности затрудняет освоение семантики операторов языка. Специальные исследования показывают, что значительный процент ошибок при написании программ связан именно с неправильным пониманием семантики языковых конструкций [1].

Наконец, реальное использование таких языков, как Бейсик, Паскаль и т. д., т. е. написание первых простых, но содержательных и интересных для обучаемого программ, возможно только после изучения значительного подмножества языка. В результате первые, наиболее трудные шаги в изучении программирования оказываются не подкрепленными самостоятельной работой на ЭВМ.

Опыт показывает, что далеко не все новички оказываются в силах преодолеть эти препятствия, встающие на традиционном пути начального обучения программированию. Но есть ли альтернатива? Существует ли дорога, идя по которой любой новичок может освоить основы программирования?

Специалистами в области информатики неоднократно предпринимались попытки разработки специальных языков

для поддержки первых шагов обучения программированию. Большую роль сыграл известный язык Лого [2], разработанный в начале 70-х гг. В его состав были впервые включены специальные средства, набор которых получил название «черепашья графика» и давал обучаемому возможность управлять отображаемым на экране дисплея исполнителем — Черепашкой. Для этой цели можно было использовать несколько простых команд (поворота, передвижения) и оператора цикла. Результат исполнения команд немедленно отображается на экране — Черепашка «ползла» и «рисовала». «Черепашья графика» сделала язык Лого популярнейшим средством начального обучения программированию.

Черепашка понравилась ученикам. Язык управления Черепашкой очень прост. Он включает всего несколько команд, семантика которых очевидна. Результат выполнения команд немедленно отображается на экране. Уже в начале работы ученики пишут простые программы для интересных рисунков, и программирование из скучной дисциплины превращается в увлекательную игру. Надо ли заставлять ребенка играть?

Черепашка понравилась учителям. Они сразу обратили внимание (а в дальнейшем это подтвердили специальные исследования [3]), что в процессе игры с Черепашкой происходит быстрое и прочное усвоение основных понятий и навыков программирования. И это естественно: в «черепашьей графике» Лого основные понятия (исполнитель, команда, программа, цикл, функция, параметр и т. д.) лежат на поверхности и не замаскированы многочисленными второстепенными деталями.

Всего за десять лет Лого произвел революцию в области обучения. Проблемам языка Лого посвящено огромное количество специальной литературы, а автор проекта Лого Сеймур Пейперт удерживает первенство по частоте цитирования среди специалистов по использованию ЭВМ в обучении [4].

Успех применения «черепашьей графики» стимулировал развитие нового подхода к обучению началам программирования. При этом подходе новички знакомятся с программированием, обучаясь управлять отображающимся на экране исполнителем. Исполнитель (Робот, Черепашка и т. д.) действует в некоторой среде и умеет выполнять определенный набор команд. По этим командам он изменяет свою среду и перемещается в ней. Управлять исполнителем можно с помощью простого мини-языка, который помимо команд исполнителя включает более или менее сложные структурные операторы (цикл, условие и т. д.). Состояние среды и положение в ней исполнителя непрерывно отображаются на экране, т. е. результат исполнения команд сразу виден обучаемому. Составляя несложные программы на мини-языке, можно заставлять исполнителя совершать разнообразные интересные действия и наблюдать за получающимся на экране «мультфильмом».

Быстро осваивая мини-язык, обучаемый с первых шагов начинает писать и отлаживать на нем содержательные программы, управляющие исполнителем. В процессе такой деятельности легко и естественно происходит усвоение многих фундаментальных понятий программирования, приобретаются навыки разработки и отладки программ. Многим обучаемым вполне достаточно опыта работы с таким языком и полученного представления о программировании для достижения в этой области необходимого уровня компьютерной грамотности. Для тех из них, кому требуются более глубокие знания, опыт работы с мини-языком существенно облегчит дальнейшее обучение, освоение настоящего языка.

Накопленный в СССР и за рубежом положительный опыт обучения началам программирования на основе мини-языков позволяет рекомендовать их для широкого использования в школах.

Заинтересовала ли вас первая часть статьи? Хотите ли вы, чтобы ваши ученики легко и быстро освоили основы программирования? Вторая часть статьи поможет вам выбрать наиболее подходящий для вас мини-язык. Если же у вас нет подходящего мини-языка или он вас не устраивает, попытайтесь разработать свой язык. Реализовать простой

мини-язык вам помогут наиболее подготовленные из ваших учеников. В этом случае вторая часть статьи даст вам несколько полезных советов по разработке.

Первый шаг построения мини-языка — определение «концептуального базиса», т. е. набора понятий, которые будут поддерживать этот язык. Следует помнить, что работа с мини-языком и исполнителем является не самоцелью, а способом усвоения определенного набора понятий и навыков. В этот набор могут входить не только чисто программистские понятия, но и некоторые понятия из других изучаемых в школе предметов. В этом случае исполнитель будет полезен не только для обучения информатике. Вид исполнителя и набор управляющих структур языка зависит от возраста и интересов учеников, от цели использования исполнителя. «Черепашью графику» языка Лого, например, удобно использовать для обучения геометрии учеников младших классов. К настоящему времени реализовано достаточно много мини-языков и исполнителей, которые могут помочь при изучении математики, географии, родного языка и даже психологии [5].

Расширяя набор поддерживаемых мини-языков понятий и увеличивая этим его «полезность», следует помнить, что мини-язык не должен терять привлекательности в изучении для той категории школьников, на которую он ориентирован. Преподаватель сам должен определить необходимый компромисс между полезностью и привлекательностью.

Для реализации разработанного мини-языка можно выбрать один из двух сложившихся к настоящему времени подходов. В соответствии с первым подходом они встраиваются в один из «больших» языков программирования. В соответствии со вторым на их основе строится самостоятельный мини-язык для целей обучения. Полученные с помощью этих подходов языки будем условно называть соответственно *подъязыками и проязыками* [6].

Основная цель разработки подъязыков — встроить в используемый язык простое и наглядное подмножество, на основе которого можно было бы начать обучение. Первым разработанным подъязыком явились упомянутые средства «черепашьей графики» Лого. Успех их применения вызвал начиная с конца 70-х гг. ряд попыток встроить «черепашью графику» в такие языки, как Паскаль и ПЛ/I. Известны и другие попытки встроить в «большой» язык наглядный мини-язык, например, с помощью введения в язык Паскаль графических типов данных и операций над ними [7].

В Советском Союзе накоплен оригинальный опыт разработки подъязыков. В новую версию школьного алгоритмического языка [8] встроено сразу несколько исполнителей, основные из которых Робот и Чертежник. С помощью этих исполнителей эффективно объясняются основные понятия информатики и конструкции школьного языка. Формально говоря, каждый встроенный в «большой» язык исполнитель образует вместе со структурами управления этого языка отдельный подъязык. Однако пример с несколькими исполнителями в одном языке четко показывает, что встроенный язык можно называть «языком» лишь условно, рассматривая его в отрыве от включающего языка. Сами авторы школьного языка настаивают: язык одинаков при работе с любыми исполнителями, исполнители лишь реализуют возможность использования дополнительных команд [9].

Наиболее перспективным способом создания подъязыков является технология, при которой для «большого» языка разрабатывается «конструктор исполнителей». С его помощью можно легко реализовать исполнителей любого рода и использовать их как для целей обучения программированию, так и для другой работы. Такая технология была впервые предложена в языке Робик [10]. На механико-математическом факультете МГУ разработана технология конструирования исполнителей для школьного языка. Эта технология используется для построения комплекта учебных миров Кумир. С ее помощью уже реализованы оригинальные исполнители Вездеход, Двуног, Строитель, Редактор.

Если в вашем распоряжении есть подобный конструктор, ваша работа упрощается. Рассмотрим, например, соз-

дание нового исполнителя для школьного языка с помощью «Кумир-технологии». Прежде всего, вам необходимо выполнить творческую часть работы — определить вид исполнителя и среды, в которой он действует, а также набор предписаний исполнителя. В состав набора должны входить команды перемещения исполнителя, изменения среды и, желательно, функции, возвращающие информацию о состоянии среды и исполнителя. Для каждого предписания необходимо определить имя, семантику (как выполняется) и интерфейс — количество и типы параметров, тип возвращаемого значения. Синтаксис (способ записи) предписаний и всего языка, управляющие структуры языка, среду работы вам придумывать не надо. Как и для любого подъязыка, они определяются включающим языком и его средой, в данном случае школьным языком и средой Е-практикума [11].

Для реализации придуманного вами исполнителя необходимо составить набор небольших программ для выполнения каждого из предписаний и задать связь новых предписаний со средой включающего языка — Е-практикумом. В используемой версии «Кумир-технологии» программы должны быть написаны на языке Си, а связь задается с помощью специальной таблицы, в которой указываются имена и интерфейсы предписаний. Далее программы компилируются и объединяются с помощью редактора связей с модулем «чистого» Е-практикума. Полученный результат представляет собой среду Е-практикума со встроенным в нее вашим исполнителем. В настоящее время авторы Е-практикума завершают разработку более простой технологии, в которой исполнителей можно будет реализовывать прямо на школьном языке.

Здесь был кратко рассмотрен лишь один из вариантов реализации «своего» мини-языка с помощью технологии «конструктора». Даже из краткого рассмотрения хорошо видно, что специальные средства существенно упрощают путь от идеи до готового встроенного исполнителя. Запрограммировать необходимо только предписания исполнителя, что доступно даже хорошо подготовленным школьникам. Самое сложное — реализацию управляющих структур и среды языка сделали для вас авторы конструктора. При отсутствии такого конструктора разработка «чисто» встроенного мини-языка доступна только хорошему коллективу программистов.

В отличие от подъязыков разработка проязыков не накладывает никаких ограничений на фантазию автора при работе исполнителя и структуры языка. В зависимости от программистской квалификации авторов всегда можно придумать исполнителя и мини-язык для него такой сложности, чтобы его самостоятельная реализация оказалась посильной. За рубежом к началу 80-х гг. был разработан целый ряд проязыков, которые предназначались для поддержки различных этапов начального обучения программированию и как следствие различались по сложности и богатству. Представление о зарубежных проязыках различной мощности можно получить на примере нескольких языков для управления исполнителем Роботом, передвигающимся по «клетчатому» полю.

Простейшие из этих языков [12, 13] включали лишь несколько команд исполнителя и предназначались для самого первого знакомства с понятием программы, исполнителя, отладки. Эти языки рассчитывались на одно-два занятия. В качестве задач обучаемым предлагалось составлять линейные программы для вывода робота из несложного лабиринта.

Более сложный язык Karel [14] уже содержал все основные структуры управления и предназначался для их освоения. Для этого языка был написан специальный учебник и разработан богатый набор задач. Язык Karel получил широкое распространение. В США, например, он используется в ряде учебных заведений на протяжении первых недель начального обучения программированию для освоения типовых структур управления и подготовки к изучению языка Паскаль.

Разработанный под влиянием языка Karel язык управления роботом Josef [15] еще более полон. В нем есть переменные, ввод, вывод. Мир робота более сложен, но естествен — это участок

города с улицами и площадями, по которым робот может перемещаться. Josef был разработан не для подготовки к изучению одного из промышленных языков, а как альтернатива этим языкам. Язык Josef позволяет поддерживать самостоятельный курс основ программирования. В рамках этого курса обучаемые, не собирающиеся специализироваться в области программирования, могут быстро, не теряя интереса, получить необходимое им представление об этой области, освоить практически все базовые понятия программирования, получить навыки структурной разработки и отладки программ.

В последнее время ряд интересных, не уступающих зарубежным проязыков разработан и реализован в Советском Союзе. Среди них необходимо назвать языки альфа-практикума [16] и другие мини-языки, разработанные на механико-математическом факультете МГУ [17], языки управления исполнителями Таракан, Корректор [18], Черепаха [19]. Все они были ранее описаны на страницах журнала «Информатика и образование», поэтому в данной статье подробно не рассматриваются.

На сегодняшний день можно говорить о мини-языках как об отдельном направлении работы в области обучения программированию. Накопленный в этой области опыт, а также специальные исследования [20, 21] позволяют сформулировать ряд требований, которым должны удовлетворять такие языки, чтобы их использование было эффективным.

1. **Простота языка.** Поскольку изучение языка является не самоцелью, а способом усвоения общих понятий, простота языка является ключевым требованием разработки. В работе [20] сделана попытка раскрыть требование простоты, выделив в нем:

функциональную простоту — простоту элементарных действий исполнителя;

логическую простоту — возможность решения содержательных задач с помощью простых программ;

синтаксическую простоту — простоту и минимальное число инструкций, отсутствие частных случаев и исключений.

2. **Наглядность интерпретации.** Большинство элементарных действий языка должны допускать простую визуализацию (отображение) на экране дисплея.

3. **Привлекательность в изучении.** Процесс изучения и работы с языком должен быть привлекателен для обучаемого. Для этого при реализации языка необходимо использовать все возможности аппаратного обеспечения (графика, цвет, звук). Задачи для этого языка могут носить характер интересных головоломок, а процесс их решения, особенно для детей, напоминает увлекательную игру.

4. **Диалоговый характер языка.** Любая команда исполнителя, введенная обучаемым в режиме диалога, сразу же выполняется. Это, в частности, позволяет получить немедленную обратную связь при усвоении семантики операторов языка. Еще не умея программировать, новичок уже может садиться за ЭВМ и осваивать работу с исполнителем. Примером диалогового промышленного языка является Бейсик.

5. **Модульный характер языка.** Для выполнения отдельных сложных действий можно определять независимые модули-подпрограммы. Каждый из модулей может быть исполнен в режиме диалога или вызван из программы. Такое построение языка дает обучаемому возможность расширить набор элементарных действий исполнителя, а также усваивать стиль разработки программ сверху вниз. Заметим, что наиболее известные зарубежные учебные языки — Лого, Josef [15], Solo [5] — имеют модульно-диалоговый характер.

Перечисленные требования — следствие общих закономерностей взаимодействия начинающих пользователей с ЭВМ, их стоит учесть при разработке любых мини-языков. Помимо этих требований хочется отметить еще два условия эффективного использования мини-языка.

Во-первых, для разработки и отладки программ на проязыке необходима дружественная программная среда, включающая дружественные редактор и интерпретатор. Примером такой среды могут служить среды альфа-практикума [16], Е-практикума [11], Черепахи [19]. Дружественный редактор, «зная» син-

таксис языка, облегчит обучаемому рутинную часть работы по вводу и исправлению программ и позволит сосредоточиться на ее содержательной части. Нажатием одной-двух клавиш начинающий пользователь ЭВМ сможет вводить, удалять и перемещать целые команды и правильные языковые конструкции. Это ускоряет ввод программ и избавляет от большинства языковых ошибок.

Дружественный интерпретатор позволит заглядывать внутрь работающей программы, наблюдать за последовательностью выполнения операторов, отлаживать программу по шагам, экспериментировать над ней. При работе с таким интерпретатором на экране дисплея постоянно отображаются текущее состояние среды, в которой действует исполнитель, и текст программы, в котором выделяется исполняемый оператор. Выполняя очередную команду, интерпретатор сразу же вносит в эту среду изменения (например, двигает Черепаху по полю) и выделяет следующий оператор или команду. Такой метод называется визуализацией исполнения. Визуализация позволяет легко усваивать семантику сложных управляющих структур и быстро отлаживать программы. Дружественная среда, включающая структурный редактор и визуализацию, может эффективно использоваться и при обучении реальным языкам, но для мини-языков разработка такой среды значительно проще, и это еще одно их достоинство.

Во-вторых, вместе с языком должен быть разработан достаточный набор содержательных задач для данного языка. Эти задачи должны быть интересны для обучаемого как с точки зрения получаемой при решении программы результата, так и с точки зрения процесса разработки решения. В набор должны входить задачи разной сложности, они должны охватывать все необходимые понятия. Самостоятельное решение таких задач обучаемым — наиболее эффективный способ освоения языка, а значит, и поддерживаемых им понятий.

Чем сложнее язык, тем больше разнообразных задач необходимо для работы с ним. Опыт показывает, что привлекательный на первый взгляд язык, для которого не удалось придумать достаточного количества интересных задач, быстро надоедает обучаемым. Напротив, множество хорошо подобранных содержательных задач способно долго поддерживать интерес к работе со скромным внешне мини-языком. Мы рекомендуем вам не проводить окончательную реализацию языка до разработки достаточного набора задач. Возможно, в процессе составления задач вам удастся найти небольшое изменение исполнителя или языка, существенно расширяющее спектр осмысленных интересных задач.

Итак, мини-язык, среда для работы с ним и набор задач образуют связную триаду. Главным в ней является, конечно, язык, но, как показывает опыт, отсутствие любого из остальных членов триады значительно снижает возможную эффективность применения мини-языка. При разработке мини-языка и выборе основных его черт следует все время помнить о среде работы с ним и возможных задачах.

В заключение рассмотрим проблему выбора структур управления в проязыке и их синтаксиса (в подъязыке их определяет включающий язык). Как отмечено выше, наиболее успешно идет работа с языками, имеющими модульный и диалоговый характер. В пределах этих требований преподаватель свободен в составлении структур управления и формы их записи. Современные языки программирования предлагают целый ряд разнообразных структур управления. При выборе структур для мини-языка следует учитывать возраст учеников. Для младших достаточно одной-двух простых структур. Такой структурой в Лого является, например, цикл повторить N раз. Чем старше ученики, тем шире может быть набор используемых структур, тем сложнее сами структуры. Если изучение проязыка является подготовкой к изучению реального языка (Фортран, Паскаль, Модула-2), рекомендуется структуры управления заимствовать с минимальной адаптацией из этого языка. Такой выбор существенно облегчает впоследствии освоение семантики структур управления реального языка. Ряд ценных советов по органи-

зации удобных и легко усваиваемых структур можно почерпнуть из специальной литературы [22].

В ближайшее время количество реализованных и успешно применяемых в школах нашей страны мини-языков будет расти.

Литература

1. *Mayer R. E.* The psychology of how novices learn computer programming // Computing Surveys. 1981. V. 13.

2. *Papert S.* Mindstorms: children, computers and powerful ideas. Brighton: Harwester Press. 1980.

3. *Khayrallah M., Van Der Meiraker M.* LOGO programming and the acquisition of cognitive skills // Journal of Computer-Based Instruction. 1987. V. 14. N 4.

4. *Rubincam I.* Frequency cited authors in the literature on computer application to education // Journal of Computer-Based Instruction. 1987. V. 14 N 4.

5. *Eisenstadt M.* A user — friendly software environment for the novice programmer // Communications of the Association for Computer Machinery. 1983. V. 20. N 12.

6. *Брусиловский П. Л.* Языки для начального обучения программированию // Вопросы психологии. 1986. № 6.

7. *Magnenat-Thalman N.* Introducing programming concepts with graphical objects // ACM SIGCSE Bulletin. 1980. V. 12. N1.

8. *Ершов А. П., Кушниренко А. Г., Лебедев Г. В., Семенов А. Л., Шень А. Х.* Основы информатики и вычислительной техники: Пробный учебник для средних учебных заведений. М.: Просвещение, 1988.

9. *Кушниренко А. Г.* О новом учебнике информатики // Информатика и образование. 1989. № 2.

10. *Звенигородский Г. А.* Язык начального обучения Робик в учебной системе программирования // Программное обеспечение задач информатики. Новосибирск, 1982. С. 72—85.

11. *Варсанофьев Д. В., Кушниренко А. Г., Лебедев Г. В.* Е-практикум — программное обеспечение школьного курса информатики // Микропроцессорные средства и системы. 1985. № 3.

12. *Николов Р. В., Бычваров С. Х.* Автоматизированная среда для обучения информатике // Вестник МГУ. Сер. 15. Вычислительная математика и кибернетика. 1984. № 3.

13. *Mawaddat F.* Another experiment with teaching of programming languages // ACM SIGCSE Bulletin. 1981. V. 13 N2.

14. *Pattis R. E.* Karel — the robot, a gentle introduction to the art of programming. London: Wiley. 1981.

15. *Tomek I.* Josef, the robot // Computers and Education. 1982. V. 6 N 3.

16. *Варсанофьев Д. В.* Альфа-практикум // Информатика и образование. 1986. № 2.

17. *Кушниренко А. Г., Лебедев Г. В.* Программирование для математиков. М.: Наука, 1988.

18. *Дуванов А., Зайдельман Я., Первин Ю.* Роботландия // Информатика и образование, 1988. № 1.

19. *Флеров М. Н.* Черепаха учит программировать // Информатика и образование. 1986. № 3.

20. *Du Boulay J. B. H., O'Shea T., Monk J.* The black box inside the glass box. Presenting computing concepts to novices // International Journal of Man-Machine Studies. 1981. V. 14. N 3.

21. *Coombs M. J., Gibson R., Alty J. R.* Learning a first computer language: strategies for making sense // International Journal of Man-Machine Studies. 1982. V. 16. N 4.

22. *Шнейдерман Б.* Психология программирования. М.: Радио и связь, 1984.

И. МОШКОВА, А. СЕМИБРАТОВ

Компьютеризация производственного обучения

Как показывает анализ практики, компьютерным программам, разработанным для уроков производственного обучения, свойственны одни и те же недостатки и стереотипный характер, порожденные прямым переносом традиционной методики обучения на схему организации учебной деятельности учащихся СПТУ с использованием ЭВТ.

Основной чертой подобного подхода является рассмотрение компьютера как нового технического средства обучения, находящегося в одном ряду с телевизором, кодоскопом, диапроектором и

прочими ТСО. Обращение к нему носит не систематический, а фрагментарный характер и связано главным образом с использованием ППС демонстрационного, информационно-справочного или контролирующего типа.

Приходится констатировать, что новая информационная технология производственного обучения фактически рождается на базе старой методической системы, позволяющей сохранить в неизменном виде содержание, методы, средства и организационные формы обучения.

Цель статьи — показать, что новая информационная технология обучения должна строиться на основе новой теории производственного обучения, новой методики, отражающей результаты современных психолого-педагогических исследований по проблемам формирования профессиональной деятельности и нового взгляда на роль и место компьютера в учебном процессе. В ходе изложения мы попытаемся сформулировать основные положения системно-структурного подхода к компьютеризации производственного обучения, взятого нами за основу для определения ее стратегии и тактики.

Практическая значимость работы заключается в определении дидактических основ компьютеризации производственного обучения, которые могли бы выступить общими рекомендациями инженерно-педагогическим работникам системы профтехобразования для разработки ППС по конкретным профессиям.

Современная позиция ведущих психологов и педагогов (О. К. Тихомиров, Е. И. Машбиц, В. В. Рубцов, Б. С. Гершунский и др.), состоит в том, что ПЭВМ рассматриваются не только как универсальные средства обработки данных, но и как средства воздействия на психику человека. Компьютеризация в широком социальном контексте должна сделать труд человека более производительным, повысить его творческое содержание, способствовать всестороннему развитию личности.

Использование ЭВТ в указанных направлениях связывается советскими философами и психологами с разработкой принципа системности. Системный, целостный характер отражения действительности считается основным отличительным признаком нового типа мышления, необходимого современному человеку, вооруженному ЭВТ. Системный стиль мышления вполне допустимо назвать стилем компьютерным (И. Б. Новик, В. П. Кузьмин и др.)

В психологических исследованиях последних лет (Б. Ф. Ломов, В. Д. Шадриков, З. А. Решетова, Е. М. Иванова и др.) показано, что профессиональная деятельность человека имеет сложное системное строение, поскольку регуляция исполнительской деятельности осуществляется функциональной системой психики, имеющей несколько уровней психического отражения одной и той же предметной деятельности. Структурная организация функциональной системы психики образована двумя видами умственных действий: связями субординации и связями координации, за счет которых различные уровни психического отражения (уровни ориентировки) и элементы каждого отдельного уровня компонируются в целостное психологическое образование — системный интеллект, выступающий внутренним механизмом функционирования полноценной целостной деятельности. Следовательно, организация учебного процесса с использованием ЭВТ должна предоставить возможность формирования целостной профессиональной деятельности, представленной в полноте интеллектуальных и двигательных компонентов труда, необходимых рабочему на современном производстве.

Замечено, что расширение профиля профессиональной подготовки осуществляется за счет совмещения отдельных трудовых функций как бы в двух направлениях: по вертикали и по горизонтали. Расширение профиля по вертикали проявляется в способности рабочего не только производить законченный продукт, но и самостоятельно разрабатывать технологический процесс и налаживать оборудование. Оно базируется на интеллектуальных действиях, ответственных за проектирование и планирование исполнительской деятельности.

Расширение профиля по горизонтали обеспечивает человеку готовность к ра-

боте на смежных производственных участках, близких по технологии, видам используемого оборудования или по характеру исполнительских операций. Оно основано на интеллектуальных операциях сопоставления, сравнения различных производственных ситуаций и оценки степени их сходства между собой.

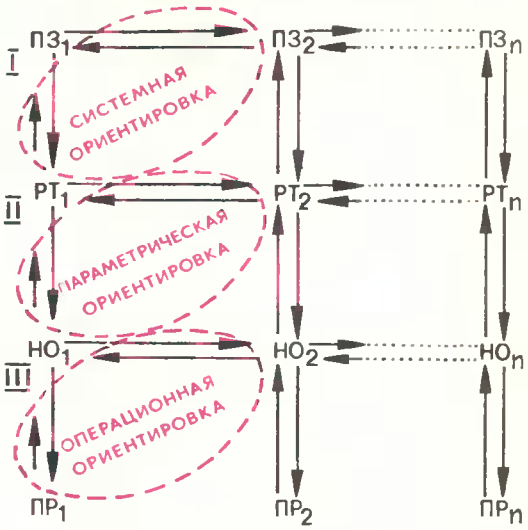
Таким образом, расширение профиля профессиональной подготовки рабочего как по вертикали, так и по горизонтали обусловлено влиянием интеллектуальных компонентов труда, которые не только возрастают в количественном отношении, но и перестраиваются качественным образом, образуя единый механизм психологического регулирования профессиональных реакций. Этот психологический механизм мы и называем системным профессиональным интеллектом рабочего высокой квалификации и широкого профиля, полагая, что он составляет важнейшее звено целостной профессиональной деятельности. Целостная структура деятельности рабочего, включающая системно-организованное профессиональное мышление, показана на рис. 1.

Системная структура интеллекта, представленная нами, образована при помощи взаимообратных связей двух типов: связей субординации, ответственных за передачу управляющих команд с одного уровня деятельности на другой (по вертикали), и связей координации, обеспечивающих переходы от одного производственного задания к другому в пределах каждого уровня (по горизонтали). Она рассматривается нами как инвариантная, базовая форма организации профессионального мышления рабочего высокой квалификации и широкого профиля, независимая от специфики содержания выполняемого им труда.

Содержание и методика производственного обучения в условиях использования ЭВТ должны строиться, исходя из задачи формирования у учащихся СПТУ системного профессионального мышления, вместе с которым целостную структуру будет приобретать и сама профессиональная деятельность.

Базовая системно-структурная организация интеллекта в процессе производственного обучения должна быть

Схема целостной профессиональной деятельности



ПЗ — производственное задание
РТ — разработанная технология
НО — налаженное оборудование
ПР — продукт труда
I — функция разработки технологического процесса
II — функция наладки оборудования
III — функция исполнения

преобразована из внешнего образца во внутреннее психологическое средство деятельности учащихся, превращена в главное условие формирования полноценной профессиональной деятельности и основной фактор профессионального становления их личности.

Согласно нашей гипотезе, компьютер, выполняющий в учебном процессе роль подлинного «квазиинтеллектуального орудия», должен применяться в производственном обучении как средство управления развитием мышления учащихся СПТУ, средство моделирования профессионального системного интеллекта. Применение ПЭВМ именно для этих целей определяет, на наш взгляд, стратегию компьютеризации производственного обучения, осуществляемую на базе нового, системно-структурного подхода.

Разработка стратегических вопросов компьютеризации позволяет выделить два психолого-педагогических требования к организации работы учащихся с ПЭВМ.

1. В связи со сложным строением системного интеллекта его формирование должно осуществляться постепенно, в соответствии с общепринятой логикой декомпозиции систем. В нашем случае это уровни, открывающие связи субординации (по вертикали) и связи между элементами в пределах одного уровня (по горизонтали), в зависимости от которых происходит определение членения учебного процесса на этапы и стадии, тесно взаимосвязанные друг с другом.

2. На всех этапах производственного обучения должна сохраняться целостность учебно-производственной деятельности учащихся СПТУ, связанная с сочетанием познавательных процессов и производственного труда. С одной стороны, работа учащихся в компьютерном классе должна подготавливать их работу в мастерской, способствуя повышению производительности труда, а с другой,— возрастание профессионального мастерства, достигаемое учащимися в мастерских, должно предварять переход на более высокий уровень осуществления познавательных процессов.

Выполнение этих требований приводит к коренным изменениям методики производственного обучения. Их сущность заключена в переходе к уровневому принципу профессиональной подготовки, перестраивающему методику в направлении целенаправленного, планомерного развития мышления учащихся СПТУ, придания ему системно-структурной организации. Использование данного принципа позволяет разложить целостную профессиональную деятельность на составляющие ее трудовые функции: разработки технологического процесса (I), наладки оборудования (II) и исполнения (III) — операторская деятельность. Переход от одного этапа производственного обучения к другому представляется нам как процесс постепенного «наращивания» трудовых функций. Одновременно происходит интенсивное насыщение профессиональной деятельности интеллектуальным содержанием и складываются определенные способы ориентировки учащихся в изучаемой предметной области, которые можно представить как различные иерархические уровни строения системного интеллекта.

«Нижний» уровень ориентировки непосредственно связан с осуществлением исполнительской деятельности, посредством которой налаженное оборудование приводится в рабочее состояние и на выходе получается готовое изделие. Поскольку операторская деятельность рабочего состоит из отдельных исполнительных актов, задача ориентировки на этом этапе состоит в определении состава и последовательности осуществления управляющих воздействий на оборудование. Эти воздействия целиком определяются планом технологических операций, составленным одновременно с наладкой оборудования. Таким образом, состав и последовательность управляющих воздействий аналогичны составу и последовательности технологических операций. С учетом указанной специфики «нижний» уровень ориентировки мы называем «операционным».

«Средний» уровень ориентировки мы называем «параметрическим», поскольку характер трудовой функции на этапе наладки оборудования предполагает установление соответствия между двумя рядами параметров. Это соответствие устанавливается между технологическими показателями процесса получения готовой продукции и техническими режимами работы конкретного оборудования. Основная задача данного этапа работы — обеспечить изоморфное отображение друг на друга технико-технологических характеристик, определяющих получение заданной продукции.

Разработка технологических процессов для широкого класса изделий требует хорошо развитой системы ориентировки рабочего во всей предметной области, в которой он работает как профессионал, способность упорядочивать свои представления об особенностях типов, классов и видов продукции, сравнивать их между собой по принятым технологическим показателям. «Верхний» уровень ориентировки мы называем «системным», так как он обеспечивает гомоморфное, основанное на синтезировании и упорядочивании, отображение в технологии всех видов готовой продукции.

Названные способы ориентировки мо-

гут различаться раскрытием содержательных характеристик базовых образов-регуляторов, управляющих управлением трудовых функций, которые являются «единицами» психического отражения действительности на различных уровнях. Известно, что «операционный» способ ориентировки строится на основе сенсомоторного образца-регулятора, «параметрический» — на основе перцептивного образа, а «системный» способ ориентировки предполагает уже наличие умственного образа-регулятора (А. М. Волков, Ю. В. Микадзе, Г. Н. Солнцева, С. Л. Малов, И. Н. Мошкова). Была обнаружена следующая зависимость: чем выше понятийность образа-регулятора, тем шире перенос профессиональных умений и навыков из одной производственной ситуации в другую. Это позволяет прийти к выводу о том, что феномены расширения профиля профессиональной подготовки рабочего «по вертикали» и «по горизонтали» зависят друг от друга и обусловлены в конечном счете психологической готовностью человека к осуществлению «системного» способа ориентировки.

Признание качественного своеобразия способов ориентировки, свойственных каждой выделенной трудовой функции, ставит вопрос о разработке таких ППС, которые в полной мере позволяли бы реализовать развиваемый нами системно-структурный подход. В рамках данного подхода пофрагментный принцип разработки ППС теряет свое право на существование. Уровневый принцип профессиональной подготовки открывает возможность осуществления иной тактики: отдельные обучающие программы должны рассматриваться как смысловые единицы одного общего целого — пакета проблемно-ориентированных программ, предназначенных для развития мышления учащихся СПТУ. Такое название происходит из современного понимания проблемного обучения как учебного процесса, построенного на основе моделирования мышления учащихся и организации их поисковой познавательной деятельности. В связи с поставленной выше стратегической целью для каждого этапа производственного обучения должны разрабатываться программы моделирующего типа, определяющие содержательные характеристики образов-регуляторов соответствующего уровня. Для формирования различных способов ориентировки требуется применение и различных типов моделирования. Есть основания считать, что для формирования операторной деятельности наибольшую эффективность будут иметь те программы, в которых используется предметное моделирование, для наладки оборудования — аналоговое моделирование, для разработки технологического процесса — моделирование знаково-символической природы. Эти формы моделирования предполагают также и свои специфические виды наглядности, отражающие постепенное возрастание доли понятийности, присутствующей в изображениях на экране дисплея.

Моделирование операторской деятельности целесообразно строить на основе наглядного изображения реального объекта, совпадающего с ним настолько, что работа с компьютером начинает имитировать реальную исполнительскую деятельность, а сам он становится своеобразным тренажером, позволяющим отрабатывать необходимые сенсомоторные координации.

Для моделирования деятельности по наладке оборудования, очевидно, требуются уже совсем иные наглядные средства — условно-пространственные схемы, имеющие вид матриц, таблиц, графиков, диаграмм и т. п., которые позволяют в компактном, лаконичном виде представить характер взаимосвязи технико-технологических параметров. Структурная оформленность параметрических зависимостей, придание им правильных и четких геометрических форм значительно облегчают человеку задачу восприятия, запоминания и воспроизведения необходимой информации, делают формирующиеся перцептивные образы устойчивыми и оперативными.

Моделирование деятельности по разработке технологического процесса, связанной с «системным» способом ориентировки, требует еще более совершенных форм наглядности — понятийных схем, которые являются уже обобщенным и

абстрагированным изображением изучаемой предметной области, подчиненным задаче ее упорядочивания. Проведение такого упорядочивания, как известно, связано с укрупнением единиц анализа предметной области, синтезированием разнообразных видов изделий, объединенных одной типовой технологией изготовления в одну группу, класс или тип продуктов труда. Динамичность, «текучесть» объектов предметной области, их постоянные преобразования друг в друга, отраженные в умственных образах-регуляторах, могут быть описаны на данном уровне только при помощи математических моделей, позволяющих вскрыть объективную логику отношений различных точек единого семантического поля.

Установление специфики средств наглядности, используемых в процессе моделирования операционного, параметрического и системного способов ориентировки, позволяет сделать заключение о характере поисковой познавательной деятельности учащихся СПТУ, которую нужно задать при помощи ППС. Поскольку операторская деятельность строится из отдельных исполнительных актов, то на данном этапе поисковая деятельность учащихся должна иметь комбинаторный характер, связанный с установлением практического результата выполнения технологических операций, различающихся по составу и последовательности входящих в них управляющих воздействий на орудие труда. При обучении функции наладке оборудования поисковая познавательная деятельность учащихся должна носить аналитический характер, позволяющий установить зависимость между значениями различных параметров. Овладение деятельностью по разработке техпроцессов, как уже говорилось выше, связано с упорядочиванием и синтезированием предметной области. Указанные свойства поисковой познавательной деятельности на каждом этапе производственного обучения могут быть заданы при помощи включения в содержание ППС специальных обучающих заданий, направленных на формирование связей субординации и связей координации соответствующего интеллектуального уровня. Учитывая двунаправленный характер этих связей, мы рекомендуем для ППС учебные задачи двух типов, составляющие между собой обратимые пары.

Данный подход открывает возможность радикальной перестройки всей методической системы производственного обучения и ее планомерного перевода на новую информационную технологию.

Выносливый дисплей

Излишне напоминать о том, что стекло — очень хрупкий материал. Это качество практически без изменений передается и изделиям, изготовленным из него. Поэтому дисплеи современных ЭВМ существенно ограничивают области применения компьютеров. Американские специалисты предложили для использования в авиационной промышленности программируемый матричный дисплей VIVISUN-2000. Дисплей предназначен для встраиваемых управляющих систем и представляет собой светодиодную матрицу, способную отобразить любую алфавитно-цифровую и графическую информацию, цвет которой может быть зеленым, красным или янтарным. Дисплей имеет свое ОЗУ, в котором могут храниться до 50 подготовленных к отображению кадров информации.

И хотя такой монитор имеет существенный недостаток — монохромность, в некоторых случаях он предпочтительнее разбитой, но имевшей перед этим сочные цвета вакуумной трубки.

ЧТО МОЖЕТ ЭВМ?

А. АВЕРБУХ, В. ГИСИН, Я. ЗАЙДЕЛЬМАН

Методические рекомендации к учебнику информатики

Примерное распределение материала § 9—12 по урокам (см. табл).

№	Содержание урока
	Тема 5. Условия. Циклы пока
18	Простые условия. Цикл пока. Величины типа лог (§ 9)
19	Решение задач на цикл пока
20	Составные условия. Измеритель (§ 9)
21	Решение задач с Измерителем и составными условиями
22	Обобщение темы
23	Контрольная работа № 2
	Тема 6. Ветвления
24	Команда если (§ 10)
25	Решение задач с командой если
26	Команда выбор. Команды контроля (§ 10)
27	Обобщение темы
	Тема 7. Алгоритмы с результатами и алгоритмы-функции
28	Команда присваивания (§ 11)
29	Решение задач
30	Алгоритмы с результатами. Их вызов (§ 12)
31	Алгоритмы-функции (§ 12)
32	Обобщение темы
33	Контрольная работа № 3
34	Обобщение материала за год

§ 9. Команда повторения пока (6 ч)

В § 9 на примере Робота и комплекса Робот+Измеритель рассматривается важная идея обратной связи — передачи информации от исполнителя к ЭВМ. Понятие **обратной связи** затрагивалось еще в § 3 при рассмотрении схемы взаимодействия человека, ЭВМ и исполнителя.

Различаются два вида команд: **команды-действия и команды обратной связи** (в частности, **команды-вопросы**). *Результат выполнения команды обратной связи — информация, полученная ЭВМ.* Эта информация представлена в виде величины определенного типа. Для команд-вопросов результат имеет тип **лог.**

Проверяя условия в команде пока, ЭВМ анализирует получаемую информацию. Смысл условий и их запись на алгоритмическом языке подробно изучаются в § 10. Во всех примерах (за исключением п. 9.18, 9.19 и некоторых упражнений) используются только простые условия. Тем не менее целесообразно при изучении команды пока рассмотреть и составные условия (п. 10.1—10.3).

Трудность изучения связок и, не, или находится в прямой зависимости от их соответствия языковым эквивалентам. Так, смысл связки и в условии А и Б практически полностью совпадает с обычным употреблением союза «и».

В то же время смысл связки или в условии А или Б заметно отличается от употребительного смысла союза «или» в естественном языке. В алгоритмиче-

Продолжение. Начало см.: Информатика и образование. 1989. № 2, 4, 5, 6.

ском языке **или** имеет соединительный смысл: условие А **или** Б выполняется, если выполняется хотя бы одно из условий А или Б. В житейском же смысле условия А и Б обычно являются взаимоисключающими («Быть или не быть?»).

Нужно обратить внимание на условия вида **не** А. Форма их записи приближена к естественному языку. Например, вместо «**не** на юге свободно» можно писать «на юге **не** свободно». Такая форма записи не должна затушевывать тот факт, что **не** относится к ответу на нерасчленимую команду-вопрос «**лог** на юге свободно».

Использование диалога ЭВМ — Робот позволяет разделить две функции: формирование ответа в зависимости от ситуации и реакция на ответ. Первую реализует исполнитель, в данном случае — Робот, вторую — ЭВМ.

Число повторений в циклах n **раз** и **для** определяется по заголовку и, следовательно, точно известно в момент начала выполнения цикла. Число повторений в цикле **пока** по заголовку не определяется и становится известным только после выполнения цикла.

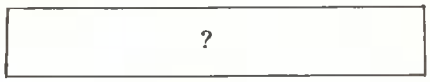
В информатике доказательства, в частности доказательства правильности алгоритмов, играют почти такую же важную роль, как и в математике. Доказательные рассуждения особенно необходимы при построении алгоритмов с командой **пока**, так как эти алгоритмы во многих случаях обладают большим (иногда бесконечным) набором исходных ситуаций, в которых они применимы. Для проведения доказательных рассуждений нужно использовать свойства команды повторения **пока**.

Рассмотрим на уроке простую задачу: Робот стоит в трех шагах западнее стены, нужно подвести его к стене.

Расширим область применимости алгоритма, считая, что расстояние до стены может меняться:

алг до стены (**арг цел** *a*)
дано *a*$\geqslant$0 ! расстояние до восточной стены
надо ! Робот возле стены

Усложним задачу. Пусть расстояние до стены не известно (мы не видим поле Робота).



(ничего не видно)

Чтобы управлять Роботом вслепую, придется перед каждым шагом спрашивать его: «на востоке свободно?» — и, пока он будет отвечать утвердительно, двигать его на восток:

алг на восток до стены
дано ! стена на востоке
надо ! Робот возле стены
нач
. **нц пока** на востоке свободно
. . шаг на восток
. **кц**
кон

При выполнении команды цикла **пока** возможны три случая.

Первый случай. Условие входа в цикл не соблюдается (имеет значение **нет**) при первой же проверке. Тогда серия команд в теле цикла не выполнится ни разу.

Второй случай. Условие входа в цикл сначала соблюдается (имеет значение **да**), а в процессе выполнения серии команд цикла изменяется на обратное.

Третий случай. При входе в цикл и при каждой следующей проверке условие соблюдается (имеет значение **да**). Такой цикл будет повторяться вечно.

Задание

Придумайте задачу, требующую применить цикл **пока** при управлении Чертежником или Черепашкой.

Ответ

Такую задачу придумать нельзя, потому что Чертежник и Черепашка не имеют команд обратной связи.

Запишем в словарь:

нц пока <логическое выражение> . <серия команд> **кц** — *цикл с неизвестным числом повторений*

не, и, или — *логические операции (в порядке убывания приоритета)*

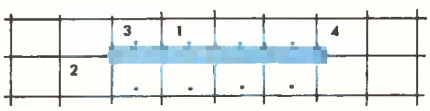
Замечания к упражнениям. На примере упражнения 28. г можно проиллюстрировать некоторые типичные методы оценки эффективности алгоритмов.

Рассмотрим два возможных решения. Первое можно получить, используя алгоритмы из предыдущих пунктов в качестве вспомогательных.

```
алг закрасить прилегающие к стене
                                клетки
    дано ! Робот находится рядом со
         ! стеной, севернее стены; стена
         ! идет с востока на запад
    надо ! закрашены прилегающие к
         ! стене клетки
нач
. закрасить полосу к югу от стены
. шаг на север
. шаг на восток
. закрасить полосу к северу от стены
кон
```

В составленном алгоритме четыре вызова команд. На рис. 2 цифрами отмечены положения Робота: 1 — исходное, 2 — после исполнения алгоритма «закрасить полосу к югу от стены», 3 и 4 — до и после исполнения алгоритма «закрасить полосу к северу от стены».

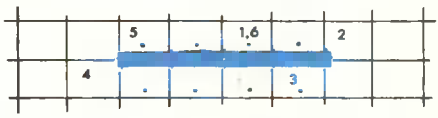
2



Второе решение: Робот движется вдоль стены с севера, закрашивая клетки, обходит стену, движется вдоль стены с юга, снова обходит стену и движется вдоль стены с севера, пока не встретит закрашенную клетку.

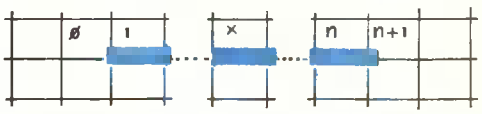
```
нач
. нц пока на юге не свободно
. . закрасить
. . шаг на восток
. кц
. шаг на юг
. шаг на запад
. нц пока на севере не свободно
. . закрасить
. . шаг на запад
. кц
. шаг на север
. шаг на восток
. нц пока клетка не закрашена
. . закрасить
. . шаг на восток
. кц
кон
```

На рис. 3 цифрами отмечены положения Робота: 1 — исходное, 2 — после исполнения первого цикла, 3 и 4 — до и после исполнения второго цикла, 5 и 6 — до и после исполнения третьего цикла.



3

Будем оценивать эффективность алгоритма числом шагов Робота. Пусть n — протяженность стены, а Робот в исходном положении находится в клетке с номером x (рис. 4).



4

При исполнении первого алгоритма Робот сначала переходит из клетки x в клетку 0, делая при этом x шагов, затем проходит вдоль стены в восточном направлении — $n+1$ шагов, делает шаг на юг, проходит вдоль стены в западном направлении — $n+1$ шагов. Всего $p=x+(n+1)+1+(n+1)=2n+3+x$ шагов.

При исполнении второго алгоритма Робот проходит из клетки x в клетку $n+1-(n+1)-x$ шагов, делает шаг на юг, проходит вдоль стены в западном направлении — $n+1$ шагов, делает шаг на север, проходит из клетки 0 в клетку x — x шагов. Всего $q=(n+1)-x+1+(n+1)+1+x=2n+4$ шагов.

Следовательно, при исполнении первого алгоритма Робот делает на $p-q=(2n+3+x)-(2n+4)=x-1$ шагов больше. Если считать, что Робот находится около середины стены, т. е. $x\approx n/2$, то $p-q\approx n/2$. Таким образом, если алгоритм исполняется многократно и начальное расположение Робота носит случайный характер, так что «в среднем» он расположен около середи-

ны стены, то применение второго алгоритма дает «в среднем» выигрыш в $n/2$ шагов при каждом его исполнении.

В задаче 32 требуется провести непростое логическое рассуждение. Прежде всего сформулируем условие, равносильное тому, что Робот находится в северо-западном углу: «на севере **не** свободно **и** на западе **не** свободно». Это условие не соблюдается, когда «на севере свободно **или** на западе свободно». Выполнение последнего условия означает, что Робот не в углу. В этом случае в соответствии с условием задачи он может сделать шаг на север или на запад в зависимости от того, в каком направлении стен нет.

```
алг в северо-западный угол лабиринта
    дано ! Робот находится внутри лабиринта; стены внутри лабиринта имеют вид отрезков и не касаются друг друга и стен прямоугольника
    надо ! Робот находится в северо-западном углу
нач
. нц пока на севере свободно или на западе свободно
. . нц пока на севере свободно
. . . шаг на север
. . кц
. . нц пока на западе свободно
. . . шаг на запад
. . кц
. кц
кон
```

Покажем, что после исполнения алгоритма Робот будет находиться в северо-западном углу.

При исполнении какого бы то ни было алгоритма могут возникнуть всего три ситуации:

1) отказ;
2) зацикливание;
3) нормальное завершение.

Установим сначала, что при исполнении данного алгоритма первые две ситуации невозможны и, значит, алгоритм должен завершиться нормально. Заметим, что команды «шаг на север» или «шаг на запад» выдаются Роботу только в тех случаях, когда в соответствующем направлении свободно, так что отказ в алгоритме исключен.

При исполнении тела большого цикла исполняется хотя бы один малый цикл и, следовательно, Робот делает хотя бы один шаг в северном или западном направлении. Таким образом, если северная стена находится на расстоянии n шагов от исходного положения Робота, а западная на расстоянии m шагов, то тело большого цикла будет исполнено не более $m+n$ раз. Следовательно, исполнение алгоритма не может продолжаться бесконечно долго и, значит, зацикливание невозможно.

При нормальном завершении алгоритма будет нарушено условие в команде **пока** большого цикла (свойство команды **пока**). Это, в свою очередь, означает, что соблюдается условие «на севере **не** свободно **и** на западе **не** свободно», т. е. Робот находится в северо-западном углу.

§ 10. Условия, команды ветвления и контроля (4 ч)

Простые и составные условия развивают введенное в § 6 понятие выражения.

Введение числовых величин (§ 5) потребовало умения научиться строить арифметические выражения. В § 9 вводятся логические величины, представленные как команды-вопросы и отношения между величинами. При этом возникает необходимость строить логические выражения. Таким образом, условие есть не что иное, как **логическое выражение**, т. е. выражение, результатом которого является логическое значение (**да** или **нет**).

Команды ветвления и выбора полностью взаимозаменяемы: всякое ветвление можно представить в форме выбора, и наоборот. Одна команда ветвления может использоваться внутри другой. При первом знакомстве с командами ветвления не стоит останавливаться на подобных сложных случаях.

Использование команды контроля **утв** (наряду с **дано** и **надо**) позволяет четко сформулировать требования к алгорит-

му, круг решаемых им задач и возможные ограничения, выделить промежуточные цели, достигаемые в ходе исполнения алгоритма. Используя утверждения, подобные применяемым в команде контроля, можно доказывать правильность алгоритмов путем строгих математических рассуждений.

При использовании команды **выбор** просмотр вариантов происходит в порядке их написания. Поэтому более строгие условия должны предшествовать менее строгим. На уроке можно разобрать известную задачу. Даны стороны треугольника. Определить его вид и вызвать соответствующий вспомогательный алгоритм.

```
алг определить вид треугольника (арг
    вещ a, в, c)
    дано a+b>c и a+c>b и b+c>a
    надо ! определен вид треугольника
нач
. выбор
. . при a=b и a=c: равносторонний
. . при a=b или a=c или b=c: равнобедренный
. . иначе произвольный
. все
. кон
```

Если в приведенном алгоритме изменить порядок проверки условий, то равносторонний треугольник будет неверно определен как равнобедренный.

Запишем в словарь:

```
если <условие>                  команда ветвления
. то <серия команд 1>           (полная форма)
. иначе <серия команд 2>
все

если <условие>                  команда ветвления
. то <серия команд>             (сокращенная форма)
все

выбор                           команда выбора
    при <условие 1>:            из трех
<серия 1>                       и большего
    при <условие 2>             числа
<серия 2>                       вариантов
...
    при <условие n>
<серия n>
. иначе <серия n+1>
все

утв — команда контроля во время выполнения алгоритма
```

Замечания к упражнениям. Перед выполнением отдельных заданий упражнения 35 полезно рассмотреть общую схему алгоритма.

```
алг закрасить соседние клетки
    дано ! на поле Робота стен нет
    надо ! для 10 клеток восточнее Робо-
         ! та закрашены клетки, при-
         ! мыкающие к закрашенным в
         ! соответствии с требованиями
нач
. нц 10 раз
. . шаг на восток
. . если клетка закрашена
. . . то
. . .
. . .
. . .
. . .
. . .
. . все
. кц
кон
```

Теперь для заданий *а — в* нетрудно выписать содержимое рамки:

а) шаг на юг
закрасить
шаг на север

б) шаг на запад
закрасить
шаг на восток

в) шаг на север
закрасить
шаг на юг

По аналогии с предыдущими заданиями можно записать для задания *г*:

шаг на восток
закрасить
шаг на запад

Однако при этом может произойти неверная закраска. Исполнение такого алгоритма приводит к тому, что оказываются закрашенными все клетки, расположенные восточнее первой закрашенной.

Можно заметить, что задания *в* и *г* превращаются одно в другое, если всюду поменять между собой «запад» и «восток». Получаем такой план: Робот про-

ходит на запад до конца полосы (11 шагов), затем движется на восток и закрашивает клетки восточнее закрашенных.

```
алг закрасить соседние клетки Г
    дано ! на поле Робота стен нет
    надо ! для 10 клеток восточнее Робо-
         ! та закрашены клетки, примы-
         ! кающие к закрашенным с во-
         ! стока
нач
. нц 11 раз
. . шаг на восток
. кц
. нц 10 раз
. . шаг на запад
. . если клетка закрашена
. . . то
. . . шаг на восток
. . . закрасить
. . . шаг на запад
. . все
. кц
кон
```

Алгоритм в задании *д* можно получить, комбинируя алгоритмы из п. *в* и *г*:

```
алг закрасить соседние клетки Д
    дано ! на поле Робота стен нет
    надо ! для 10 клеток восточнее
         ! Робота закрашены клетки,
         ! примыкающие к закрашен-
         ! ным с востока
нач
. нц 10 раз
. . шаг на восток
. . если клетка закрашена
. . . то
. . . шаг на запад
. . . закрасить
. . . шаг на восток
. . все
. кц
. шаг на восток
. нц 10 раз
. . шаг на запад
. . если клетка закрашена
. . . то
. . . шаг на восток
. . . закрасить
. . . шаг на запад
. . все
. кц
кон
```

§ 11. Величины в алгоритмическом языке. Команда присваивания (2 ч)

В § 2 уже говорилось об ЭВМ как об универсальной информационной машине. Информация в ЭВМ представляется в виде **величин.**

Рассмотрим основные характеристики величины: имя, тип, вид и значение.

С точки зрения человека, **имя** величины — это ее обозначение в алгоритме. Именем в алгоритмическом языке может быть любая последовательность букв, цифр и знаков подчеркивания, начинающаяся с буквы.

Для ЭВМ имя означает место в памяти, где хранится значение величины. Память для величин выделяется как бы внутри памяти для алгоритма. Это означает, что *в разных (в том числе вызывающих друг друга) алгоритмах одинаковые имена обозначают разные величины.*

Тип величины задает множество допустимых значений величины и множество применимых к ней операций.

Вид величины характеризует ее использование в алгоритме, роль содержащейся в величине информации. В алгоритмическом языке имеются четыре вида величин: арг, рез, арг рез и промежуточные.

Имя, тип и вид величины указываются в ее **описании.** Это статические характеристики величины, они не могут меняться в процессе работы алгоритма.

Значение — динамическая характеристика величины, оно может многократно меняться в ходе работы алгоритма.

Присваивание — основной способ изменения значения величины. Тип величины в левой части присваивания и тип выражения в правой части должны совпадать. Исключение: допускается присваивание вида вещ:=цел (но не наоборот).

Модель памяти ЭВМ позволяет проиллюстрировать исполнение команды присваивания. Особое внимание нужно

обратить на случай, когда величина из левой части команды присваивания входит в выражение в правой части (см. упр. 37—39).

Важно подчеркнуть, что исполнение команды присваивания меняет значение только одной величины — той, что стоит в левой части.

Рассмотрим некоторые типичные ошибки, возникающие при работе с величинами.

1. Величина не описана.

2. Величина описана дважды.

Примеры:

а) **нач цел** x, **вещ** x;

б) **алг** А (**арг вещ** x)
...
нач вещ x

3. Выполняются операции, не соответствующие типу и значению величины.

Пример:

нач цел m, n, **вещ** a
...
$n:=1$; $a:=5.4$
$m:=\mathrm{mod}(n, a)$

4. Величине присваивается значение, несовместимое с ее типом.

Примеры:

а) **нач цел** n
$n:=1.3$

б) **нач цел** m, n
$n:=3$
$m:=n/2$

в) **алг** А (**арг цел** n)
...
алг В
...
А (1.3)

5. Во время исполнения алгоритма изменяется значение величины-аргумента.

Примеры:

а) **алг** А (**арг вещ** x)
...
$x:=2.3$

б) **алг** А (**арг цел** n)
...
нц для n **от** 1 **до** 5
2...

6. Во время исполнения команды повторения **для** изменяется значение параметра цикла; после исполнения команды повторения **для** используется значение параметра цикла.

Примеры:

нц для i **от** 1 **до** n
. $i:=a+1$
. ...
кц

нц для i **от** 1 **до** n
. ...
кц
$a:=i+1$

Заметим, что перечисленные ошибки имеют разный статус. Ошибки типа 1, 2 — синтаксические и могут быть обнаружены до выполнения алгоритма.

Ошибки типа 3, 4 — тоже синтаксические. Иногда принимаются соглашения, которые позволяют исполнить алгоритм при наличии в нем таких ошибок. Для этого, например, действительное число округляется до целого или берется его целая часть и т. п.

Примеры типа 5, 6 могут формально и не считаться ошибками. При наличии конструкций типа 5, 6 алгоритмы трудны для понимания. В конкретных реализациях алгоритмического языка на ЭВМ конструкции из примеров 5, 6 могут быть разрешены. Тем не менее следует избегать их при составлении алгоритмов.

На уроке можно предложить ученикам серию упражнений на поиск ошибок.

1) **нач цел** d	выражение
. $d:=1$	в левой части
. $d+1:=2$	второй команды
кон	присваивания

2) **нач лог** n, **вещ** x	несоответствие
. $x:=3.1416/4$	типов величины
. $n:=\sin(x)$	и выражения
кон	в правой части

3) **нач вещ** b, y, z	синтаксическая ошибка
. $y:=3$	в правой части
. $z:=1.6$	последней
. $b:=5(y+z)$	команды
кон	

4) **нач вещ** a, b	значение величины a
. $b:=a/2$	в правой части

. a:=0; b:=5
. b:=b+1
кон

первой команды не определено

5) нач лог F, T
. T:=шаг на юг
. F:=не на юге
свободно
кон

в правой части первого присваивания
команда - действие

6) нач цел m
. если m : =1
. то m=1
. все
кон

перепутаны присваивание и сравнение

Запишем в словарь:
:= — *команда присваивания*

Замечания к упражнениям. На примере упражнения 37 разберем характерный прием анализа команд присваивания. Обозначим через a значение величины x до исполнения команд. Тогда получаем:

а) $a+5=3$ $a=-2$
б) $-a=3$ $a=-3$
г) $a+1=3$ $a=2$

§ 12. Результаты алгоритмов и алгоритмы-функции (5 ч)

Указания относятся к § 12 полного текста учебника, опущенному в журнальном конспекте.

Результатом всех ранее рассмотренных алгоритмов были действия исполнителя. В § 12 вводятся алгоритмы, результатом которых является переданная в основной алгоритм информация в форме **величин-результатов.**

Величины-аргументы и результаты называют *параметрами* алгоритма. Параметры обеспечивают явную информационную связь основных и вспомогательных алгоритмов, позволяют писать содержательные алгоритмы, не использующие исполнителей.

При вызове устанавливается связь между *формальными* параметрами, описанными в заголовке вспомогательного алгоритма, и *фактическими*, указанными в команде вызова в основном алгоритме.

Связь устанавливается строго по порядку следования: первый фактический соответствует первому формальному и т. д. Следовательно, количество формальных и фактических параметров должно совпадать, т. е. параметры должны быть **согласованы по числу.**

Кроме того, формальные и фактические параметры должны быть **согласованы по типу.** Фактический параметр должен быть выражением (для аргументов) или именем величины (для результатов) того же типа, что и соответствующий формальный параметр.

Передачу параметров можно представить в виде присваивания «формальный := фактический» для аргументов и «фактический := формальный» — для результатов. Такое представление объясняет необходимость согласования типов и невозможность использования выражений в качестве фактических параметров-результатов.

Величина может быть одновременно аргументом и результатом, в этом случае она описывается в алгоритме как арг рез.

Алгоритмы-функции принципиально ничем не отличаются от обычных алгоритмов с результатами. Разница лишь в оформлении самого алгоритма и его вызова.

Вызов обычного алгоритма — это отдельная команда основного алгоритма. Вызов функции не требует специальной команды, он может использоваться в любом выражении в качестве величины, тип которой определяется типом результата функции.

Особенности оформления функции связаны с особенностями ее использования. Результат функции не включается в список формальных параметров (это нарушило бы согласование по числу). Тип результата указывается перед именем функции, что соответствует использованию функции как величины в выражениях.

Имя результата с точки зрения основного алгоритма совпадает с именем функции. Кажется естественным использование имени функции в качестве имени результата и в самой функции, но при этом становится невозможной рекурсия. Поэтому в качестве имени результата в алгоритме-функции используется специальное служебное слово знач.

Во всех примерах учебника знач —

единственный результат функции. Вообще говоря, функция может иметь и результаты, описанные обычным образом, с помощью слова рез. Такие результаты называются *побочным эффектом* функции. В учебнике и пособии функции с побочным эффектом не используются.

При использовании готовых функций нас по-прежнему может не интересовать внутренний механизм работы функции. В этом смысле функции аналогичны командам обратной связи исполнителей.

Понятие алгоритма-функции можно сопоставить с понятием функции в курсе математики. Сходство достаточно очевидно, а вот вопрос об отличиях является более тонким.

Алгоритм-функция с аргументами может при разных вызовах выдавать разные результаты для одних и тех же значений аргументов. Это объясняется тем, что результаты алгоритмов-функций зависят не только от значений аргументов, но и от состояния среды исполнителя. Чтобы застраховаться от ошибок, обычно придерживаются такого правила: после исполнения алгоритма-функции состояние памяти ЭВМ (исключая величину-результат) и среды исполнителя остаются прежними.

На уроке составим алгоритм с таким заголовком:

алг маркировка теплых клеток
  дано ! Робот перед входом в тупик,
       ! идущий на восток
  надо ! закрашены клетки, темпера-
       ! тура в которых выше средней

Р е ш е н и е:

```
нач вещ T, цел D
. найти среднюю температуру (T, D)
. утв   ! T — средняя температура
.       ! D — число клеток в тупике
. разметить тупик (T, D)
кон
алг найти среднюю температуру (рез
вещ S, цел l)
  дано ! Робот перед входом в тупик,
       ! идущий на восток
  надо ! S=средняя температура
       ! l=длина тупика
нач
.  S := 0; l := 0
. нц пока на востоке, свободно
. .  шаг на восток
. .  l := l+1
. .  S:=S+температура
. кц
. утв l>0 ! S — значение суммарной
  температуры
. S:=S/l
кон
алг разметить тупик (арг вещ t, цел d)
  дано ! t=средняя температура
       ! d=длина тупика
  надо ! закрашены клетки, темпера-
       тура
       ! в которых выше средней
нач
. нц d раз
. . если температура>t
. . . то закрасить
. . все
. . шаг на запад
. кц
кон
```

Как работает каждый из вспомогательных алгоритмов в отдельности, понятно. Важно разобраться в их взаимодействии, в том, как первый из них *передает* результат второму. Процесс *передачи параметров* в данном случае происходит по такой схеме:

```
алг найти среднюю температуру (рез
                вещ S, цел l)
          основной алгоритм: T   D
алг разметить тупик (арг вещ t, цел d)
```

На уроке можно «оживить» анализ алгоритма, вызвав к доске учеников. В рассматриваемом примере потребуются пятеро. Пусть для определенности алгоритм «маркировка теплых клеток» выполняет ученик Авербух, алгоритм «найти среднюю температуру» — ученик Гисин, алгоритм «разметить тупик» — ученик Зайдельман, наиболее ответственные роли Робота и Измерителя поручим отличницам Володиной и Яшиной.

Рассмотрим ход выполнения алгоритмов для тупика, изображенного на рис. 5. В каждой клетке указано значение температуры.

5

Авербух рисует на доске область па-

мяти для своего алгоритма, выделяет ячейки для величин:

6

Затем *Авербух* вызывает к доске *Гисина*.

Гисин выделяет на доске часть памяти для своего алгоритма. Затем выполняет первую строчку (команды присваивания):

7

Гисин. На востоке свободно?

Володина. Да.

Гисин. Шаг на восток!

Володина перемещает Робота на один шаг на восток.

Гисин увеличивает значение I на единицу, после чего обращается к *Яшиной:* — Температура?

Яшина. Пять.

Гисин заносит полученное число в ячейку S...

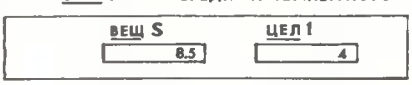
Комментарий

В этом месте мы пропустим часть диалога и обратимся сразу к моменту окончания выполнения первого вспомогательного алгоритма.

Конец комментария

Гисин выполняет последнее присваивание (значение S в этот момент равно 34):

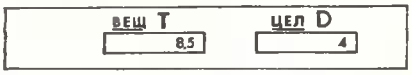
8



Гисин сообщает значения величин *Авербуху*: «Первая величина равна восьми с половиной, вторая — четырем». После этого *стирает с доски свою часть памяти* и садится на место.

Авербух заносит полученные числа (в порядке их поступления) в соответствующие ячейки:

9



Авербух вызывает к доске *Зайдельмана*.

Зайдельман выделяет область памяти с ячейками для своего алгоритма, спрашивает у *Авербуха* значения величин.

Авербух. Первая величина равна восьми с половиной, вторая — четырем.

Зайдельман заносит значения величин в ячейки:

10

Зайдельман. Температура?

Яшина. Восемь.

Зайдельман. Шаг на запад!

Володина перемещает Робота на запад.

Зайдельман. Температура?

Яшина. Двенадцать.

Зайдельман. Закрасить!

Володина закрашивает клетку.

Зайдельман. Шаг на запад!.. И так далее.

Закончив выполнение своего алгоритма, *Зайдельман* стирает с доски свою область памяти и (вместе с *Володиной* и *Яшиной*) возвращается на место. *Авербух* уходит на свое место, не стирая свои записи. На доске остаются область памяти основного алгоритма и поле Робота, на котором вторая и третья клетки тупика закрашены.

Запишем в словарь:

рез — *величина-результат;*

арг рез — *величина, одновременно аргумент и результат;*

знач — *обозначение результата в теле алгоритма-функции.*

Окончание следует.

В. КАЙМИН, С. ГРИГОРЬЕВ

Пролог в школьной информатике

Не стихают жаркие споры о предмете информатики как научной и учебной дисциплины. Установка на программирование как вторую грамотность не выдержала проверки практикой. Не дало значимых результатов и объявление алгоритмизации ядром школьной информатики.

Вспомним, однако, данное А. П. Ершовым определение информатики как научной дисциплины, изучающей средства и методы накопления, обработки и передачи информации в ЭВМ. А ведь такие же процессы протекают не только в ЭВМ, но и в общении людей, в жизни общества, в жизнедеятельности организмов, в функционировании автоматических устройств и автоматизированных систем.

Действительно, накоплением, передачей и обработкой информации человечество занимается в течение своего существования и без активного обмена информацией существовать не может. Этим целям служило развитие языков, появление письменности, бумаги, почты, телеграфа, телефона, радио, телевидения и т. п.

Обработка информации и принятие решений всегда оставались прерогативой людей, распределенной к тому же весьма неравномерно в зависимости от их положения в обществе. Положение существенно изменилось в связи с появлением автоматических устройств и вычислительных машин в качестве средств передачи, обработки и накопления информации.

Особые надежды в этом плане возлагаются на ЭВМ пятого поколения, которые будут общаться с людьми, воспроизводить логические рассуждения и использовать накапливаемые в их памяти знания для решения различного рода практических задач и поиска ответов на широкий круг вопросов.

Применение ЭВМ с такими возможностями может в корне изменить сложившиеся в человеческом обществе технологии производства и даже сферу взаимоотношений людей путем широкого использования баз данных, экспертных систем и сетей связи. Создание и использование такого рода автоматизированных систем на базе сетей ЭВМ, на наш взгляд, и составляет ядро информатики как научной дисциплины, а изучение логических принципов работы ЭВМ, организации баз знаний и общения с ЭВМ должно составлять содержание информатики как учебной дисциплины в высшей и средней школе.

Технической базой для обучения информатике с таким толкованием содержания могут быть современные персональные ЭВМ, имеющие развитую внешнюю память и интерпретаторы языков искусственного интеллекта типа Пролог, который является, как известно, одним из прототипов языков программирования для вычислительных машин пятого поколения.

Интерпретаторы языка Пролог в настоящее время созданы практически для всех зарубежных и отечественных ЭВМ и, в частности, для наших школьных ЭВМ — БК-0010, «Корвет», УКНЦ, «Ямаха». Отличительной особенностью этих интерпретаторов является не только диалоговый режим работы с ЭВМ, но и возможность общаться с ЭВМ на основе лексики родного языка.

Особенностью языка Пролог является то, что он является не столько языком для программирования, сколько языком для описания данных и логики их обработки. Типичным примером может служить описание на нем данных о рождении и хобби своих друзей и знакомых:

```
рождение(Иванова, Лена, 22, июнь, 1971);
рождение(Петров, Сергей, 25, октябрь, 1973);
рождение(Сидорова, Оля, 1, декабрь, 1974);
любит(Иванова, Лена, книги);
любит(Иванова, Лена, танцы);
любит(Петров, Сергей, видео);
любит(Сидорова, Оля, кино);
```

Приведенные строки есть примеры записи фактов на языке Пролог. Ввод и накопление в ЭВМ такого рода информации позволяет позже ставить разнообразные вопросы:

```
?рождение(x,y,_,_,1971);      - кто родился
                                в 1971 году?
?рождение(x,y,_,октябрь,_);   - кто родился
                                в октябре?
?любит(x,y,книги);            - кто любит
                                книги?
?любит(x,y,книги),            - кто любит
 любит(x,y,танцы);              одновременно
                                и книги, и
                                танцы?
```

Для сложных вопросов характерно использование сложносоставных понятий, определение которых в Прологе задается в форме правил:

```
книголюб(x,y)<-любит(x,y,книги);
взрослый(x,y)<-рождение(x,y,_,_,год'),
                МЕНЬШЕ(год',1972);
```

Примеры вопросов с использованием этих правил:

```
?книголюб(x)
?взрослый(x)
?взрослый(x),любит(x,танцы)
```


Уже из этих примеров можно заключить, что Пролог приспособлен для описания фактографической, понятийной и логической информации и может служить средством формализации описания самых разнообразных сведений и представлений. Эти свойства и используются при создании на его основе баз знаний и экспертных систем по самым различным приложениям и предметным областям.

В таких применениях описания на Прологе есть не что иное, как определенного рода модели. Эти модели, которые вполне можно назвать информационно-логическими, позволяют описывать характеристики, свойства и взаимосвязи самых различных вещей и явлений. Главным вопросом становится подбор и описание модели для тех или иных объектов и процессов.

Здесь важно учитывать, что различаются четыре основных вида моделей — графические представления, словесные описания, информационно-логические модели и математические (количественные) модели. Графические представления — зримые образы вещей и явлений (фотографии, рисунки, чертежи, схемы), представляющие их с той или иной степенью детальности с той или иной точки зрения.

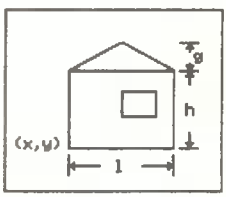
Словесные описания — выражение сведений о вещах и явлениях на некотором разговорном языке, представляющие их опять же с определенной точки зрения, с определенной степенью детальности. Базой для таких описаний служит система понятий, в которых выражаются интересующие нас сведения.

Информационно-логические модели — формализация словесных описаний, выражаемая средствами математической логики, а математические модели — точные количественные модели, используемые для описания физических объектов и явлений. В действительности и то и другое — математические модели, поскольку и то и другое предполагает математическую точность определений.

Определяющими факторами в выборе степени детальности и учете характеристик для модельных представлений являются, во-первых, круг подлежащих решению задач, а во-вторых, адекватность — степень соответствия действительности тех предсказаний и выводов, которые получаются с помощью этих моделей.

Взаимосвязи между графическими представлениями, словесными описаниями и информационно-логическими моделями достаточно выпукло могут быть проиллюстрированы на примерах описания изображений объектов средствами языка Пролог. Одним из простейших примеров может служить описание изображения домика.

Рисунок



Словесное описание

Домик состоит из крыши, стены и окна. Крыша треугольная. Стена и окно прямоугольные.

Информационно-логическая модель:

```
домик(x,y,l,h,g,c) <- стена(x,y,l,h,c),
                      крыша(x,y+h,l,g,c);
крыша(x,y,l,h,c) <- треугольник(x,y,x+l,y,
                                 x+l/2,y+h,c);
стена(x,y,l,h,c)<-рамка(x,y,l,h,c),
                  рамка(x+l/2,y+h/2,l/3,h/3,c);
треугольник(x1,y1,x2,y2,x3,y3) <- линия(x1,y1,
                  x2,y2,c),линия(x2,y2,x3,y3),
                                  линия(x3,y3,x1,y1,c);
рамка(x,y,l,h)<- линия(x,y,x+l,y,c),
                 линия(x+l,y,x+l,y+h,c),
                 линия(x+l,y+h,x,y+h,c),
                 линия(x,y+h,x,y,c);
```

Здесь x, y — координаты нижнего левого угла домика, l — его длина, h — высота стены, g — высота крыши. Каждая линия, треугольник и рамка, а также крыша и стена могут иметь свой цвет c. Система всех этих понятий с их характеристиками и определениями задана в описании этой модели.

Описание изображения на Прологе получается из этой информационно-логической модели замещением всех формул соответствующими базовыми предикатами СЛОЖЕНИЕ и УМНОЖЕНИЕ (а также ДЕЛЕНИЕ и ВЫЧИТАНИЕ, которые в Прологе определяются из этих базовых предикатов).

```
домик(x,y,l,h,g,c) <- стена(x,y,l,h,c),
                      СЛОЖЕНИЕ(y,h,y1'),
                      крыша(x,y1',l,g,c);
крыша(x,y,l,h,c) <- СЛОЖЕНИЕ(x,l,x1'),
                    ДЕЛЕНИЕ(l,2,l1'),
                    СЛОЖЕНИЕ(x,l1',x2),
                    СЛОЖЕНИЕ(y,h,y2'),
                    треугольник(x,y,x1',y,
                                x2',y2',c);
стена(x,y,l,h,c) <- рамка(x,y,l,h,c),
                    ДЕЛЕНИЕ(l,2,l1'),
                    СЛОЖЕНИЕ(x,l1',xo'),
                    ДЕЛЕНИЕ(h,2,h1'),
                    СЛОЖЕНИЕ(y,h1',yo'),
                    ДЕЛЕНИЕ(l,3,lo')
                    ДЕЛЕНИЕ(h,3,ho'),
                    рамка(xo',yo',lo',ho',c);
треугольник(x1,y1,x2,y2,x3,y3,c) <- ЛИНИЯ(x1',
        y1',x2',y2',c),ЛИНИЯ(x2',y2',x3',y3',c),
                      ЛИНИЯ(x3',y3',x1',y1',c);
рамка(x,y,l,h,c) <- СЛОЖЕНИЕ(x,l,x1'),
                    ЛИНИЯ(x,y,x1',y,c),
                    СЛОЖЕНИЕ(y,h,y1'),
                    ЛИНИЯ(x1',y,x1',y1',c),
                    ЛИНИЯ(x1',y1',x,y1',c),
                    ЛИНИЯ(x,y1',x,y,c);
ДЕЛЕНИЕ(c,a,b) <- УМНОЖЕНИЕ(a,b,c);
```

Представленного описания домика, стены и крыши достаточно для решения целого класса задач построения разнообразных домиков, поселков, многоэтажных домов и городков. Отдельный дом, представленный на исходном рисунке, будет построен по запросу

```
? домик(100,120,40,40,20,2);
```

Изображение пары домиков тех же размеров, но разных цветов будет получено по запросу

```
? домик(60,120,40,40,20,2),домик(120,120,
                                 40,40,20,3);
```

Для изображения поселка из таких домиков, расставленных на одинаковых расстояниях друг от друга, используются рекурсивные описания:

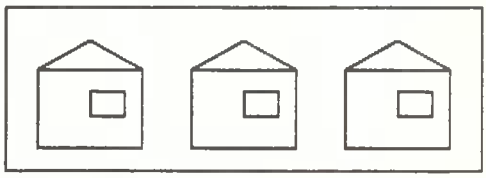
```
дома(x,y,l,h,g,c,r,1) <- домик(x,y,l,h,g,c);
дома(x,y,l,h,g,c,r,n) <- домик(x,y,l,h,g,c),
                         дома(x+r,y,l,h,g,
                                  c,r,n-1);
```

Данное описание интерпретируется так: если указан один дом, то рисуется единственный домик, если же указываются n домов, то рисуется один домик, а затем n—1 на расстоянии r от первого. Соответствующее описание на Прологе:

```
дома(x,y,l,h,g,c,r,1) <- домик(x,y,l,h,g,c);
дома(x,y,l,h,g,c,r,n) <- домик(x,y,l,h,g,c),
          СЛОЖЕНИЕ(x,r,x1'),ВЫЧИТАНИЕ(n,1,n1'),
                         дома(x,y,l,h,g,c,r,n1');
ВЫЧИТАНИЕ(c,a,b) <- СЛОЖЕНИЕ(a,b,c);
```

Изображение поселка из трех домиков, используя эти описания, можно получить по запросу

```
? дома(60,120,40,40,20,2,60,3);
```



Аналогично можно получить с помощью рекурсивных описаний изображения многоэтажных домов и т. п.

Рекурсивные определения в Прологе играют важнейшую роль, поскольку они являются единственным и в то же время универсальным средством описания правил обработки данных и организации вычислений. Этим Пролог существенно отличается от традиционных языков программирования, в которых основным средством организации вычислений являются циклы.

В качестве примера приведем рекурсивные определения двух функций — возведение в степень $y=x\backslash n\backslash$ и факториал $m=n!$

Примеры вопросов и ответов:

```
?степень(2,5,y);
 y = 32
?факториал(5,m);
 m = 120
```

Следующее существенное отличие Пролога от традиционных языков программирования в том, что для представ-

Пролог	Логика
степень(x,Ø,1); степень(x,n,y) <- ВЫЧИТАНИЕ(n,1,n1'), степень(x,n1',y1'), УМНОЖЕНИЕ(y1',x,y); факториал(Ø,1); факториал(n,m) <- ВЫЧИТАНИЕ(n,1,n1'), факториал(n1',m1'), УМНОЖЕНИЕ(m1',n,m);	{ x\Ø\ = 1 } { n1 = n-1 } & { x\n-1\ = y1 } & { y = y1-x } { Ø! = 1 } { n1 = n-1 } & { m1 = (n-1)! } & { m = m1-n }

ления данных в нем используются не массивы, а списки. Списки в Прологе — это перечни конкретных данных, заключаемые в квадратные скобки.

Например, записи о днях рождения, используя списки, можно было бы записывать в следующей форме:

```
рождение([Иванова,Лена],[22,июнь,1971]);
рождение([Петров,Сергей],[25,октябрь,1973]);
рождение([Сидорова,Оля],[1,декабрь,1974]);
```


Тогда на вопрос «Когда родилась Иванова Лена?»

```
? рождение([Иванова,Лена],дата');
```

ЭВМ даст ответ:

```
дата =[22,июнь,1971]
```

А на вопрос «Кто родился в октябре?»

```
? рождение(кто',[_,октябрь,_]);
```

будет ответ:

```
кто = [Петров,Сергей]
```

Обработка данных, представляемых списками, в Прологе, естественно, проводится с помощью рекурсивно определяемых правил. Примером может служить анализ данных об успеваемости, которые можно было записывать в следующей форме:

```
оценки([Петров,Сергей],физика,[4,3,2,5]);
оценки([Петров,Сергей],лирика,[3,2,4]);
оценки([Иванова,Лена],физика,[5,5,5]);
оценки([Иванова,Лена],математика,[5,5,4,5]);
оценки([Сидорова,Оля],математика,[4,5,4,5]);
```

Информацию о том, какие оценки имеет данный ученик по физике, можно получить с помощью вопроса:

```
? оценки([Петров,Сергей],физика,x);
  x = [4,3,2,5]
```

Для выделения элементов списка в Прологе предусмотрено специальное средство — образец списка $[x|y]$, который выделяет первый элемент списка и остаток списка после удаления первого элемента. Пример применения образца:

```
? оценки([Петров,Сергей],физика,[x|y]);
  x = 4
  y = [3,2,5]
```

Наличие конкретных значений в списке можно определить в форме рекурсивного предиката.

Пролог	Логика
есть(x,[x|Z]); есть(x,[y|Z]) <- есть(x,Z);	x ∈ {x,.. } x ∈ {z,..} => x ∈ {y,z,..}

Теперь для выяснения, у каких учеников есть пятерки по физике, можно задать вопрос:

```
? оценки(x,физ,5),есть(5,5);
```

Ответом будет список

```
x = [Петров,Сергей]
x = [Иванова,Лена]
x = [Иванова,Лена]
x = [Иванова,Лена]
```

Троекратный ответ с одной и той же ученицей вызван тем, что в списке ее оценок по физике имелось три пятерки. Это особенность языка Пролог, в котором в общем случае предполагается поиск в списках всех вариантов вхождения. Впрочем, есть и возможность прекратить поиск после нахождения первого вхождения.

Теперь мы достаточно вооружены, чтобы демонстрировать примеры обработки данных средствами Пролога. Легко, например, ввести определения неуспевающих учеников:

```
неуспевает(x,p) <- оценки(x,p,5),есть(2,5);
неуспевающий(x) <- неуспевает(x,_);
```

Для анализа успеваемости обычно используются средние баллы. Средние значения в списках числовых данных можно определить с помощью следующих предикатов.

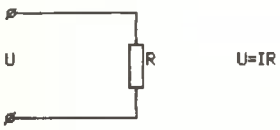
```
СУММА([],0);
СУММА([x|y],S)<-СУММА(y,S1'),СЛОЖЕНИЕ(S1',x,S),
ДЛИНА([],0);
ДЛИНА([x|y],n)<-ДЛИНА(y,n1'),СЛОЖЕНИЕ(n1',1,n),
СРЕДНЕЕ([],0);
СРЕДНЕЕ(x,s)<-СУММА(x,S),ДЛИНА(x,n),
                              ДЕЛЕНИЕ(S,n,s);
```

Этих предикатов достаточно для определения учеников, лучших по тому или иному предмету *p*:

```
отлично(x,p) <- оценки(x,p,S),СРЕДНЕЕ(S,5);
физик(x) <- отлично(x,физ);
лирик(x) <- отлично(x,лит);
```

В целом Пролог оказывается удобным средством формализации и постановки задач для самых различных предметных областей. Наиболее просто это осуществляется в тех областях, где известны математические модели. В качестве примера опишем на Прологе математические модели электрических цепей.

Сначала опишем модель простейшей электрической цепи, содержащей единственное сопротивление:



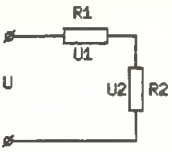
В Прологе связь между напряжением, сопротивлением и силой тока можно описать предикатом

```
ом(U,R,I) <- УМНОЖЕНИЕ(R,I,U);
```

Здесь УМНОЖЕНИЕ (*a*, *b*, *c*) — базовый предикат Пролога, выражающий связь $a \cdot b = c$. Особенность этого предиката в том, что неизвестным может быть любой из трех его аргументов *a*, *b*, *c*. По этой причине для указанной цепи можно получить ответы, задавая вопросы с любым параметром в качестве неизвестного.

```
? ом( U, 10, 4);
 U = 40
? ом( 100, 25, I);
 I = 4
? ом( 100, R, 2);
 R = 50
```

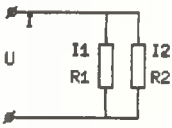
В моделях более сложных цепей, содержащих несколько сопротивлений, необходимо учитывать законы Кирхгофа, описания которых приведены ниже.



Первый закон Кирхгофа - закон суммы напряжений:

U=U1+U2

```
напряжение(U,U1',U2') <- СЛОЖЕНИЕ(U1',
                                  U2',U);
```



Второй закон Кирхгофа - закон суммы токов:

I=I1+I2

```
токи(I,I1',I2') <-  СЛОЖЕНИЕ(I1',I2',I);
```

Описание модели цепи с последовательным соединением сопротивлений может быть представлено предикатом

```
схема1(U,R1',R2',I) <- ом(U1',R1',I),
                       ом(U2',R2',I),
                       напряжение(U,U1',U2');
```

а описание модели цепи с параллельным соединением сопротивлений — предикатом

```
схема2(U,R1',R2',I) <- ом(U,R1',I1'),
                       ом(U,R2',I2'),
                       токи(I,I1',I2');
```

Для первой схемы по этому описанию мы можем узнать величину напряжения при заданных сопротивлениях и известной силе тока:

```
? схема1(U,20,20,4);
 U = 160
```

А для второй цепи — узнать силу тока при известном напряжении и заданных значениях сопротивлений:

```
? схема2(100,20,20,I);
 I = 10
```

Однако при попытке по этим описаниям моделей определить для первой цепи силу тока, а для второй — входное напряжение ЭВМ даст отказ. Причина в том, что в используемых определениях в первом же предикате требуется найти сразу две неизвестные величины (напряжение и силу тока), а известно только сопротивление.

Данный пример показывает, что

описания на Прологе одних только моделей и постановок задач недостаточно для получения от ЭВМ их решений; необходимо указать еще и способы решения.

Для рассматриваемых задач с электрической цепью способ решения (как известно из физики) состоит в определении общего сопротивления цепи и нахождении на его основе связи между напряжением и силой тока.

Для цепи с последовательным соединением общее сопротивление

$$R_{общ}=R1+R2.$$

Соответствующее описание на Прологе способа определения напряжения U и силы тока I в цепи выглядит так:

```
схема1(U,R1',R2',I)   <- последовательно(R1',
                                  R2',Rобщ'),
                         ом(U,Rобщ',I);
последовательно(R1',R2',Rобщ) <- СЛОЖЕНИЕ(R1',
                                  R2',Rобщ');
```


Для цепи с параллельным соединением общее сопротивление

$$R_{общ}=R1\cdot R2/(R1+R2).$$

Соответствующее описание на Прологе:

```
схема2(U,R1',R2',I)   <- параллельно(R1',
                                  R2',Rобщ'),
                         ом(U,Rобщ',I);
параллельно(R1',R2',Rобщ') <- СЛОЖЕНИЕ(R1',
                                  R2',R3'),
                        УМНОЖЕНИЕ(R1',R2',R12'),
                        ДЕЛЕНИЕ(R12',R3',Rобщ');
```

Теперь ЭВМ может давать ответы на вопросы и о величине напряжения, и о силе тока в обеих цепях.

```
? схема1(200,20,20,I);
 I = 5
? схема2(U,20,20,4);
 U = 40
```

Знание и формальное описание способов решения — основа основ в решении задач при использовании языка Пролог и любого другого языка программирования. Однако если для традиционных языков программирования способы решения выражаются в алгоритмической форме (последовательность планируемых действий), то в Прологе способы решения задач, как правило, выражаются через решение более простых подзадач.

В качестве иллюстрации и сравнения Пролога с алгоритмическими языками рассмотрим простой алгоритм диалога и соответствующую программу на Прологе.

```
алг"распорядок дня"
нач
  вывод("распорядок")
  запрос("час=",n)

  если 0≤n<7 то
   вывод("сон")
  инес 7≤n<8 то
   вывод("завтрак")
  инес 8≤n<13 то
   вывод("школа")
  инес 13≤n<14 то
   вывод("обед")
  инес 14≤n<19 то
   вывод("свобода")
  инес 19≤n<20 то
   вывод("ужин")
  инес 20≤n<23 то
   вывод("отдых")
  инес 23≤n<24 то
   вывод("сон")
  кесли
кон
```

```
программа<-ВЫВОД("распо
                  рядок"),ПС,
          ЗАПРОС("час=",n),
          ПС,ОТВЕТ(n);
ЗАПРОС(t,n)<-ВЫВОД(t),
             ВВОДЦЕЛ(n);
ОТВЕТ(n)<-МЕЖДУ(0,n,7),
          ВЫВОД("сон");
ОТВЕТ(n)<-МЕЖДУ(7,n,8),
          ВЫВОД("завтрак");
ОТВЕТ(n)<-МЕЖДУ(8,n,13),
          ВЫВОД("школа");
ОТВЕТ(n)<-МЕЖДУ(13,n,14),
          ВЫВОД(("обед");
ОТВЕТ(n)<-МЕЖДУ(14,n,19),
          ВЫВОД("свобода");
ОТВЕТ(n)<-МЕЖДУ(19,n,20),
          ВЫВОД("ужин");
ОТВЕТ(n)<-МЕЖДУ(20,n,23),
          ВЫВОД("отдых");
ОТВЕТ(n)<-МЕЖДУ(23,n,24),
          ВЫВОД("сон");
МЕЖДУ(a,n,b)<-МЕНЬШЕ(n,b),
             НЕ(БОЛЬШЕ(n,a);
```

Запуск на выполнение этой программы на Прологе проводится командой-вопросом

```
?программа
```

В ответ на этот вопрос ЭВМ выведет на экран

```
распорядок
час = ?
```

В ответ на ввод любого целого числа от 0 до 24 ЭВМ сообщит запланированный вид занятий.

Приведенное алгоритмическое решение можно намного упростить, представив информацию в ОТВЕТах в форме фактов:

```
занятие(0,7,сон);
занятие(7,8,завтрак);
занятие(8,13,школа);
занятие(13,14,обед);
занятие(14,19,свобода);
занятие(19,20,ужин);
занятие(20,23,отдых);
занятие(23,24,сон);
```

При таком представлении информации программа на Прологе принимает следующий вид:

```
программа  <- ВЫВОД("распорядок"),ПС,
              ЗАПРОС("час=",n),ПС,
              ОТВЕТ(n);
ЗАПРОС(t,n)<- ВЫВОД(t),ВВОДЦЕЛ(n);
ОТВЕТ(n)   <- занятие(a,b,дело'),
              МЕЖДУ(a,n,b),
              ВЫВОД(дело');
МЕЖДУ(a,n,b)<-МЕНЬШЕ(n,b).НЕ(БОЛЬШЕ(n,a));
```

Эта запись совсем не похожа на описание алгоритма. Из анализа этого примера можно сделать вывод о том, что

программы на Прологе — это наборы правил, применение которых позволяет получать ответы на некоторый круг вопросов или решения некоторого круга задач.

Развивая эту тему, нужно сказать, что программы на Прологе могут представлять некоторые базы знаний об определенных предметах и предметных областях и о способах решения связанных с ними задач. Основной вопрос здесь — достаточны ли полнота этих знаний и полнота способов и средств решения возникающих задач и проблем для того, чтобы можно было говорить о программе как о некоторой базе знаний.

Такого рода базы могут создаваться и наполняться самыми различными знаниями — по истории, литературе, произведениям искусства, памятникам архитектуры, геологии, географии, химии, биологии и другим гуманитарным, естественнонаучным и техническим дисциплинам как в среднем, так и в высшем образовании.

Выполнение такого рода работ в явном виде требует (и формирует) от составителей определенной информационной культуры — умений собирать, систематизировать и анализировать информацию, накапливать ее в ЭВМ, проводить на ЭВМ эксперименты, выявлять логические ошибки и находить правильные решения.

В то же время работа с конкретной содержательной информацией, с литературными источниками по самым различным темам развивает общечеловеческую культуру, необходимую каждому, и особенно молодежи, которая будет жить в новом, информационном обществе, насыщенном компьютерами, сетями ЭВМ, базами данных и экспертными системами.

Разбор учебных примеров с программами на языках типа Пролог при работе на ЭВМ и анализе получаемых результатов требует от учащихся воспроизведения логических рассуждений и вычислений, производимых компьютерами. Такие упражнения начинаются с решения на ЭВМ простейших задач со словами, словарями, таблицами, далее — с графическими иллюстрациями и простейшими геометрическими построениями.

Такого рода занятия, с одной стороны, не требуют особой математической подготовки и доступны всем учащимся, а с другой — незаметно погружают их в технику логических рассуждений и умений манипулировать переменными и формулами, т. е. подготавливают их к активному освоению аппарата математики в упражнениях на ЭВМ. Эта часть курса информатики может основой стать для интеграции информатики и математики в системе среднего и высшего образования.

Литература

1. *Григорьев С. Г., Морозов В. Н.* Знакомьтесь — язык Пролог // Информатика и образование. 1987. № 4.

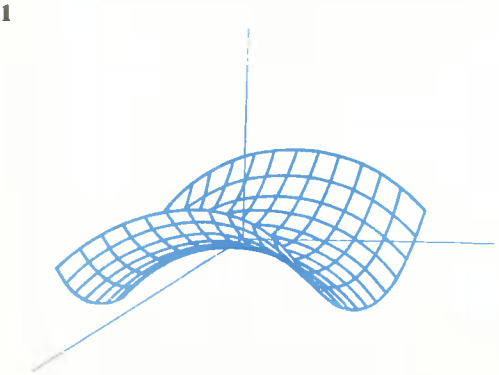
2. Основы информатики и вычислительной техники: Пробный учебник для учащихся X — XI классов средних школ. М.: Просвещение, 1989.

Л. БЕЛОВА, Ю. БЕЛОВ
Ярославский государственный университет

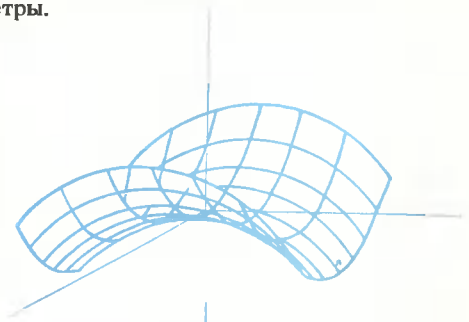
Изображение поверхностей

С необходимостью изображения поверхностей приходится сталкиваться в аналитической геометрии при изучении поверхностей второго порядка и в анализе при изучении функций двух переменных. В школьных факультативах по информатике тоже не раз приходится встречаться с задачей построения на экране или бумаге красивого купола, или гамака, или волн моря. Мы обратились к литературе по машинной графике [1, 2] и на основе имеющихся алгоритмов разработали, используя Бейсик, простые программы построения поверхностей, точнее — графиков функций двух переменных. При этом удалось даже улучшить контроль видимости путем простейшей модификации алгоритма, описанного в [1] и [2]. Мы предлагаем читателям краткое описание основных процедур программы построения графика функции двух переменных. Программа реализована на БК-0010.01 и ДВК-3; графические возможности этих машин не позволяют создавать полутоновые изображения, поэтому поверхность приходится изображать каркасным (сеточным) методом с помощью семейств кривых, являющихся сечениями поверхности графика функции плоскостями $x=\text{const}$ или $y=\text{const}$. На рис. 1—4 приведены результаты работы нашей программы для различных семейств сечений и функции FNS(X, Y)=A*X*X+ +B*Y*Y, где A, B — запрашиваемые параметры.

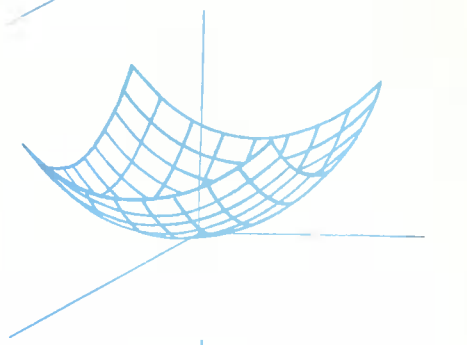
1



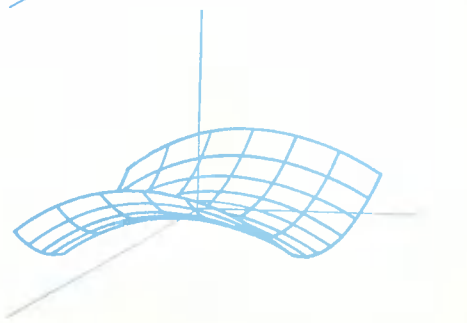
2



3



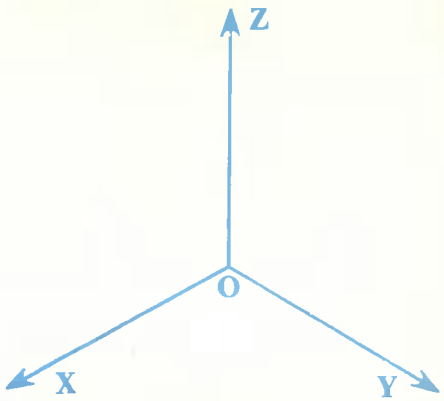
4



Итак, нам надо как-то спроектировать на экран график функции $Z=f(X, Y)$, т. е. множество точек трехмерного пространства, координаты которых имеют вид $(X, Y, f(X, Y))$. Считаем при этом, что $|X| \leqslant K$ и $|Y| \leqslant K$, т. е. точка (X, Y) пробегает на плоскости X0Y квадрат размером 2*K×2*K. Как отмечалось выше, при сеточном методе на экран отображаются не все точки поверхности, а лишь те, которые образуют семейство сечений с некоторым равномерным шагом. При этом возникает два основных вопроса: как спроектировать на экран точку трехмерного пространства, имеющую известные мировые координаты (X, Y, Z), и как обеспечить контроль видимости, т. е. уничтожение тех частей линий, которые закрыты другими фрагментами поверхности? Для решения первого вопроса надо прежде всего условиться, как располагать изображения трех пространственных координатных осей на экране. На уроках черчения в школе употребляется изометрическая проекция (рис. 5), а при изучении стереометрии оси располагаются чаще так, как на рис. 6 (диаметрическая проекция или близкая к ней).

В программе реализован второй вариант; при этом начало координат располагается в центре экрана, ось 0Y идет вправо, 0Z — вверх, 0X — к наблюдателю, а на экране изображается идущей влево и вниз. Предполагается, что наблюдатель расположен выше плоскости X0Y. Напомним, что начало двухмерных дисплейных координат в БК-0010 находится в левом верхнем углу экрана, ось XD горизонтальна, YD направлена сверху вниз. Экранные единицы длины очень малы, поэтому в программе используется масштаб M, определяющий, сколько экранных единиц содержится в одном шаге по оси 0Y или 0Z. Так как XD изменяется от 0 до 255, а YD — от 0 до 240, то точка (0, 0, 0) будет иметь экранные координаты (128, 120). Изображение точки (0, 1, 0) с учетом масштаба будет иметь экранные координаты XD=128+M*1, YD=120. Изображение точки (0, 0, Z) будет иметь координаты XD=128, YD=120—M*Z, так как оси 0Z и YD направлены противоположно. Экранные координаты изображения точки (X, 0, 0) будут, очевидно, таковы: XD=128—M*k*X, YD=120+M*k*X, где $0<k<1$ можно подбирать, добиваясь лучшей реалистичности изображений. Можно, например, положить $k=0.5$. В общем случае получим, что образ точки (X, Y, Z) на экране будет иметь координаты

```
XD=128+M*Y-0.5*M*X      (1)
YD=120-M*Z+0.5*M*X
```

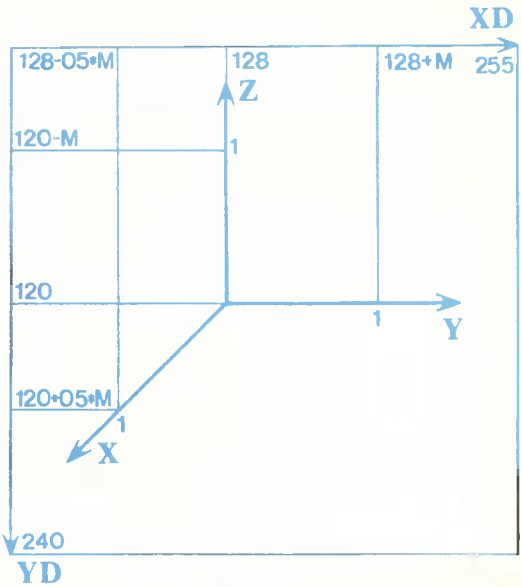


Конечно, имеется общая математическая теория проектирования пространства на плоскость, но наша задача — разобраться только в одном конкретном виде проектирования. Отметим, что для ДВК-3 формулы (1) немного изменятся в связи с тем, что там пределы XD и YD — 400×280 и начало экранных координат в левом нижнем углу. Для этой ПЭВМ верны следующие формулы:

33

```
XD=200+M*Y-0.5*M*X      (2)
YD=140+M*Z-0.5*M*X
```

Теперь, считая, что $f(X, Y)$ определена на квадрате от —K до +K по X и от —K до +K по Y, можно организовать построение семейств сечений. Сечение строится поточечно, каждая точка проектируется на экран с помощью формул (1) (или (2)). Вот соответствующий фрагмент программы.



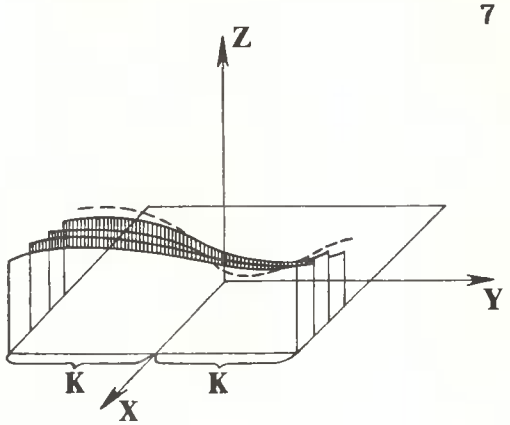
```
280 REM ПОСТРОЕНИЕ СЕЧЕНИЙ,
          ПАРАЛЛЕЛЬНЫХ OY С.280-360
290 FOR X=K TO -K STEP -2ЖК/LX
300 FOR Y=-K TO K STEP 1/M
310 Z=FNS(X,Y) 'ВЫЧИСЛЕНИЕ
                        ЗНАЧЕНИЯ F(X,Y)
320 GOSUB 510 'ПОЛУЧЕНИЕ
            ЭКРАННЫХ КООРДИНАТ (XD,YD)
330 GOSUB 610 'ВЫЧИСЛЕНИЕ
                   ПРИЗНАКА ВИДИМОСТИ V
340 IF V=1 THEN PSET(XD,YD) 'ПОСТРОЕНИЕ
      ТЕКУЩЕЙ ТОЧКИ, ЕСЛИ ОНА ВИДИМА
350 NEXT Y
360 NEXT X
```

Здесь LX — количество сечений, параллельных оси 0Y; во внутреннем цикле шаг 1/M соответствует единичному шагу по экрану. Переход на 510 соответствует вычислениям по формулам (1) для текущих значений (X, Y, Z), вычисление признака видимости в точке 610 будет разобрано ниже. Отметим, что совершенно аналогично проводится построение семейства сечений, параллельных 0X, только снаружи будет цикл по Y, а внутри — по X.

Попробуем теперь разобрать вопрос о контроле видимости, который решается независимо для семейств сечений, параллельных 0X и параллельных 0Y. Отметим, что сечения, параллельные 0Y, строятся по убыванию X, т. е. по удалению от наблюдателя («вглубь» экрана). Поэтому очередное строящееся сечение может быть заслонено (экранировано) только той частью поверхности, которая расположена ближе к наблюдателю и уже простроена. Аналогичное соображение имеется и для сечений, параллельных 0X. Это позволяет вести контроль видимости, пользуясь принципом «текущего экранирования» («текущего экрана»), т. е. очередную точку (XD, YD) текущего сечения отображать на экран только тогда, когда она не попадает в промежуток между уже построенными сечениями. Для отслеживания этого факта заводятся два массива MAX (255) и MIN (255), в которых хранятся верхняя и нижняя границы «текущего экрана». Перед рисованием сечений в массив MAX заносятся нули, а в массив MIN — единицы, что означает пустоту «текущего экрана». В тех точках XD, где рисунок уже появился, имеем, конечно, неравенство MAX(XD)$\geqslant$MIN(XD), а разность MAX(XD)—MIN(XD) равна толщине текущего рисунка вдоль вертикали XD. После проверки очередной точки «текущий экран» может, если надо, расшириться, включая ее в себя (рис. 7). Вот соответствующий фрагмент программы.

```
610 REM ВЫЧИСЛЕНИЕ ПРИЗНАКА ВИДИМОСТИ
                        ТЕКУЩЕЙ ТОЧКИ (XD,YD)
620 V=0
630 IF MIN(XD)<=YD AND YD<=MAX(XD)
                                THEN 670
640 V=1
650 IF YD>MAX(XD) THEN MAX(XD)=YD
660 IF YD<MIN(XD) THEN MIN(XD)=YD
670 RETURN
```

7



Это основные моменты программы. Перед началом построения графика программа запрашивает K, M, параметры A, B и требуемое количество сечений LX и LY, параллельных 0Y и 0X. Затем экран дисплея очищается, инициируется массивы MAX и MIN и запускается цикл построения семейства сечений, параллельных 0Y, если LX>0. Перед построением второго семейства снова инициируются MAX и MIN, если LY>0, строится второе семейство.

Теперь об упомянутом ранее замечании по улучшению алгоритма контроля видимости. Отметим, что описанный выше алгоритм контроля «лишь приближенно отслеживает реальный контур спроектированного к данному моменту участка поверхности» (1), поэтому возможен неполный контроль видимости на краях рисунка. Для уменьшения дефектов в [1] рекомендуется увеличивать качество сечений. Однако даже для указанной довольно простой функции FNS одно сечение на БК-0010.01 строится около 7 с. Для тригонометрических функций расход времени еще возрастает. Легко понять, что время построения графика фактически пропорционально LX+LY, так что вряд ли имеет смысл задавать LX и LY больше, скажем, 20. Невелика и разрешающая способность дисплея.

Поэтому для улучшения контроля видимости при построении сечений, параллельных 0X (в этом семействе искажения больше), можно MAX и MIN после обычной инициализации «подправить» — задать им значения, равные граничному сечению графика функций, параллельному оси 0Y, т. е. выполнить следующие команды:

```
381 X=K
382 FOR Y=-K TO K STEP 1/M
383 Z=FNS(X,Y)
384 GOSUB 510
385 MAX(XD)=YD
386 MIN(XD)=YD
387 NEXT Y
```

Аналогично можно улучшить контроль в семействе сечений, параллельных 0Y, присвоив MAX и MIN значения соответствующего граничного сечения, параллельного 0X. С дополнительным контролем время выполнения будет пропорционально LX+LY+2. Один и тот же график с дополнительным контролем и без него приведен на рис. 1, 2.

Описанная программа реализована на Бейсике БК-0010.01, на «зеленоградском» варианте Бейсика для ДВК-3, на Паскале в системе РАФОС с графическим пакетом MED для ДВК-3.

Кроме того, программа на Паскале для ДВК-3 компонуется с разработанным одним из авторов интерпретатором арифметических выражений, что позволяет в режиме диалога строить графики любых комбинаций элементарных функций, допускаемых, например, синтаксисом Паскаля, даже с некоторыми расширениями.

Она может использоваться для изучения поверхностей второго порядка в курсе аналитической геометрии и линейной алгебры, так как фактически реализует классический метод изучения вида поверхностей — метод координатных сечений. Программа полезна и в практических занятиях по математическому анализу при изучении темы «Функции нескольких переменных», так как позволяет наглядно показать различные типы особенностей их графиков, зависимость вида функции от выбираемых параметров и другие детали поведения функции.

На занятиях можно требовать при заданных параметрах и выражении функции «угадать» заранее (до включения программы) график или, наоборот, по предъявленному на экране графику определить функцию.

Адрес для справок: 150000, Ярославль, ул. Кирова, 8/10, факультет ИВТ Ярославского государственного университета, кафедра системного программирования.

Литература

1. *Баяковский Ю. М., Галактионов В. А., Михайлова Т. Н.* Графор. Графическое расширение Фортрана. М.: Наука, 1985.

2. *Котов Ю. В.* Как рисует машина. М.: Наука, 1988.

В. ПРОХОРОВ, С. КОНДИН
Москва

«Микроша» в компьютерном обучении

Бытовая ПЭВМ «Микроша», выпускаемая Лианозовским электромеханическим заводом, вполне успешно используется для создания первичной сети компьютерного обучения. У нас имеется опыт ее применения в трех областях:

для обучения слушателей курсу «Основ информатики и вычислительной техники»;

для оптимизации занятий по общественным наукам;

для безбумажного сбора и первичной обработки научно-исследовательской информации.

Отсюда логично вытекает курс на создание программ:

тренажеров для изучения клавиатуры и выработки навыков работы на ПЭВМ;

прикладных программ для обеспечения научно-исследовательской работы слушателей;

обучающих, обучающе-контролирующих и контролирующих программ.

Все вместе они существенно расширяют информационные горизонты будущих специалистов, сокращают рутинные нетворческие процессы в деятельности обучающих и обучаемых, за счет реализации возможностей моделирования и выполнения прогностических функций помогают глубже проникать в сущность изучаемых явлений, процессов и их закономерностей и, наконец, позволяют усилить практическую направленность и наглядность процесса обучения.

Среди программ-тренажеров на первых занятиях мы используем программы двух типов по изучению и освоению клавиатуры ПЭВМ «Микроша»: первый позволяет в удобном для начинающего режиме разобраться с функциональным назначением каждой клавиши, второй используется как средство закрепления выработанных навыков и их совершенствования. Обе программы на Бейсике. Сразу надо оговориться, что мы пользуемся интерпретатором Бейсик-Микрон, который гораздо более удобен, чем версия, предложенная заводом-изготовителем (имеются средства редактирования и отладки про-

грамм (EDIT), стыковки файлов (MERGE) и т. д.).

У нас разработан ряд программ, связанных с решением слушателями различных научно-исследовательских и учебных задач. Это программы по статистическому моделированию социальных процессов, сбору и обработке информации в ходе экспертного опроса и т. д.

Важное место занимают программы, используемые для оптимизации занятий по общественным дисциплинам. Мы исходили из положения, что такие программы должны быть унифицированными, доступными всем (в том числе и неподготовленным) пользователям, иметь возможность адаптации к уровню подготовки слушателя.

На одной из таких программ хотелось бы остановиться подробнее. Речь идет о системе подготовки педагогических программ (СППП) «Собеседник».

СППП «Собеседник» — инструментальное средство, предназначенное для создания и эксплуатации педагогических программ, позволяющих проводить автоматизированные консультации, тренировки, практические занятия, контрольный опрос и тестирование.

Система позволяет создавать информационно-методические пособия, информационно-справочные программы, информационно-обучающие программы широкого назначения и состоит из трех самостоятельных программ, написанных на Ассемблере: «Собеседник-исполнение», «Собеседник-заполнение», текстовый редактор. Авторами созданы первые две программы и использована программа текстового редактора ED.CPA-80 с учетом возможности их сопряжения.

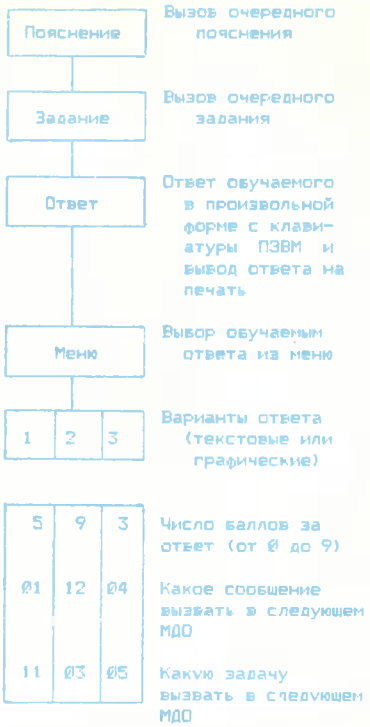
Назначение программ системы:

«Собеседник-исполнение» предназначена для работы обучаемого с дидактическим материалом, используется на занятии;

«Собеседник-заполнение» предназначена для проверки и отладки работы педагогических программ, используется преподавателем в период подготовки к занятию;

текстовый редактор предназначен для заполнения, исправления и распечатки дидактического материала, используется преподавателем в ходе подготовки к занятиям.

При помощи системы строятся педагогические программы модульного типа. Упрощенный алгоритм работы одного модуля показан на рисунке. Соединение в определенной последовательности модулей представления дидактического материала позволяет строить педагогические программы и с линейным (последовательным) представлением пояснений и учебных заданий, и с ветвлением (по уровням сложности).



В зависимости от алгоритма представления дидактического материала «Собеседник» позволяет использовать режимы работы:

информационный — представление информационного дидактического материала последовательно или с поиском необходимой информации;

обучающе-контролирующий — контроль усвоения дидактического материала следует после его текстового или схематического представления (разъяснения) обучаемому;

контролирующе-обучающий — пояснение дидактического материала следует в случае ошибочного ответа на задание;

контролирующий — проверка изученного ранее дидактического материала без пояснений;

проведение анкетирования или тестирования с оценкой по количеству набранных баллов за выбранные ответы.

Общий объем дидактического материала, используемого без перегрузки ОЗУ ПЭВМ, может достигать 25К (около 15 машинописных страниц текста).

Критерии оценки знаний определяются преподавателем при составлении содержания дидактического материала и могут оперативно изменяться непосредственно перед применением обучающих программ на занятии. Общая оценка определяется в зависимости от количества решенных обучаемым задач и суммы полученных за каждый ответ бал-

лов. По итогам работы обучаемого ему может предоставляться динамическая призовая игра и исполняется одна из четырех мелодий в зависимости от полученной оценки.

Система позволяет вести электронный журнал успеваемости обучаемых на занятии и протокол работы каждого обучаемого, получить распечатку этих документов на печатающем устройстве.

СППП «Собеседник» позволяет преподавателю, не имеющему специальной подготовки в области программирования, создавать и использовать на занятиях педагогические программы с любым дидактическим материалом по любой теме общественных наук.

На ее основе созданы автоматизированный курс практических занятий по формальной логике, консультационные программы по работам классиков марксистско-ленинской теории, программа автоматизированного тестирования «Простые и сложные аналогии», логико-комбинаторные тесты, ряд специализированных контролирующих программ.

В настоящее время проводится работа по адаптации СППП «Собеседник» для ПЭВМ серии «Электроника» (БК-0010, БК-0011, ДВК и др.).

Дополнительную информацию можно получить по тел. 409-16-61 у С. Н. Кондина.

Будем рады обменяться программами с другими владельцами «Микрош». У нас в кабинете информатики, оснащенном «Микрошами», собрано 115 системных и игровых программ для этой ПЭВМ.

Наш адрес: 424000, Йошкар-Ола, ул. Анциферова, 29, СШ 20, кабинет информатики.

Е. ЕРЕМИН
канд. физ.-мат. наук, г. Пермь

Русскоязычная диагностика ошибок на «Ямахе»

Всем преподавателям хорошо знакомы трудности, с которыми сталкиваются начинающие пользователи при расшифровке всевозможных сообщений об ошибках на английском языке. Увидев на экране дисплея несколько непонятных фраз на ломаном английском, многие теряют веру в собственные силы и приобретают стойкую иллюзию о невозможности общения с машиной без посторонней помощи. Не лишена такого недостатка и широко распространенная японская ПЭВМ «Ямаха». Этот предназначенный для учебных целей компьютер хотя и произведен специально для нашей страны, но русских диагностических сообщений не имеет. На первый взгляд может показаться, что здесь ничего нельзя изменить, так как Бейсик-интерпретатор хранится в ПЗУ. Между тем избавиться от необходимости перевода и сделать процесс общения с компьютером более дружественным довольно просто. Для этого достаточно ввести небольшую программу в машинных кодах, разработанную автором. Предлагаемое усовершенствование доступно любому пользователю, знающему алгоритмический язык Бейсик, и позволяет буквально за несколько часов «научить» ПЭВМ выдавать вместо английских диагностических сообщений русские.

В основу работы программы русскоязычной диагностики (в дальнейшем для краткости будем называть ее ПРД) положен принцип обработки ловушек. Подробное описание этого способа модификации ПЗУ с интерпретатором дано в прилагаемом к машине руководстве по Бейсику на с. 253—254 и 312—314. Программа записывается на диск в виде двоичного файла, основную часть которого занимают собственные тексты сообщений об ошибках. Рассылка и запуск ПРД может производиться как стандартными средствами Бейсика, так и монитором BASMON, поставляющимся с последними комплектами КУВТ MSX-2 (во втором случае загрузка удобнее и быстрее).

Программа ПРД рассчитана на ученическую ПЭВМ «Ямаха MSX-2» без дисковода. Она выводит на русском языке все сообщения об обнаруженных ошибках, в том числе и при выполнении оператора INPUT, а также заменяет текст «Break» на «Останов». ПРД работает как в непосредственном режиме, так и при выполнении Бейсик-программы, причем в последнем случае возможен один

из двух описываемых ниже вариантов вывода информации. Вариант А аналогичен принятому на «Ямахе»: дополнительно к сообщению о характере ошибки выводится номер строки, к которой относится это сообщение. Вариант В вместо указания на строку ПРД автоматически генерирует команду LIST, которая выводит строку с ошибкой на редактирование. Переход от одного варианта к другому осуществляется изменением содержимого ячейки F22CH (это адрес в памяти после запуска ПРД; если же вы захотите внести исправление в файл, то адрес будет BE1FH). Число 0 в этой ячейке соответствует варианту А; в прилагаемом тексте программы рассматриваемый байт ненулевой, т. е. реализован вариант В.

Итак, что же нужно сделать, чтобы ваш компьютер «заговорил по-русски?» Прежде всего сформировать файл ПРД в соответствии с приводимой таблицей шестнадцатеричных кодов. Работа эта хотя и трудоемкая, но несложная, причем имеющихся в Бейсике средств вполне достаточно. Те, кто будет испытывать затруднения на этом этапе, могут обратиться за помощью к главе 16 уже упоминавшегося руководства по Бейсику. Отметим, что для проверки правильности набора программы в ПРД предусмотрено контрольное суммирование.


```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F 0123456789ABCDEF
B900 F5 C5 D5 E5 3A 2D 00 FE 01 28 13 21 6C BC 11 69 <==
B910 BB D5 CD A3 B9 CD 23 73 E1 CD A3 B9 18 3F 21 72
B920 BD 3A FE FE FE 87 28 E6 21 00 B9 11 74 BF AF F5    Загрузчик:
B930 F1 8E F5 23 7C BA 20 F8 7D BB 20 F4 F1 2E 6F AE
B940 21 BD BB 11 E4 BD 20 C9 21 62 B9 22 9C F3 11 AC     часть 1
B950 B9 01 1A 00 21 F0 FB 22 FA F3 CD B4 BE E1 D1 C1                =
B960 F1 C9 F5 C5 D5 E5 21 F3 B9 11 00 EE 01 80 05 ED =><==
B970 B0 21 C6 B9 56 23 5E 23 7E 23 FE 24 28 07 B7 28
B980 F3 12 13 18 F3 3E 0C CD A2 00 21 F2 7E CD A3 B9     часть 2
B990 E5 21 E4 7E CD A3 B9 E1 CD A3 B9 21 4A F3 CD A3
B9A0 B9 18 BA 7E 23 B7 C8 CD A2 00 18 F7 63 6C 65 61          ==>clea
B9B0 72 32 30 30 2C 26 68 45 44 46 46 3A 61 3D 75 73 r200,&hEDFF:a=us
B9C0 72 31 28 61 29 0D FE FD C3 2F F2 00 FF 02 C3 62 r1(a) <==
B9D0 F2 00 FF 61 C3 CC F2 00 FD E0 AF 32 2E F2 00 F3  Информация для
B9E0 7C 45 00 FF 5C C3 DC F2 00 F3 9C 5A 47 00 F3 7D     ловушек
B9F0 53 43 24 00 4E 45 58 54 20 C2 C5 DA 20 46 4F 52 ==>¦NEXT без FOR
BA00 00 F3 C9 CE D4 C1 CB D3 C9 DE C5 D3 CB C1 D1 20 ¦Синтаксическая
BA10 CF DB C9 C2 CB C1 00 52 45 54 55 52 4E 20 C2 C5 ошибка¦RETURN бе
BA20 DA 20 47 4F 53 55 42 00 E9 D3 DE C5 D2 D0 C1 CE з GOSUB¦Исчерпан
BA30 D9 20 C4 C1 CE CE D9 C5 20 D7 20 44 41 54 41 00 ы данные в DATA¦
BA40 EE C5 D0 D2 C1 D7 C9 CC D8 CE D9 CA 20 D7 D9 DA Неправильный выз
BA50 CF D7 20 C6 D5 CE CB C3 C9 C9 00 F0 C5 D2 C5 D0 ов функции¦Перепо
BA60 CF CC CE C5 CE C9 C5 00 F0 C1 CD D1 D4 D8 20 C9 лнение¦Память и
BA70 D3 DE C5 D2 D0 C1 CE C1 00 EE C5 20 CF D0 D2 C5 счерпана¦Не опре
BA80 C4 C5 CC C5 CE 20 CE CF CD C5 D2 20 D3 D4 D2 CF делен номер стро
BA90 CB C9 00 F7 D9 C8 CF C4 20 DA C1 20 D0 D2 C5 C4 ки¦Выход за пред
BAA0 C5 CC D9 20 CD C1 D3 D3 C9 D7 C1 00 F0 CF D7 D4 елы массива¦Повт
BAB0 CF D2 CE CF C5 20 CF D0 C9 D3 C1 CE C9 C5 20 CD орное описание м
BAC0 C1 D3 D3 C9 D7 C1 00 E4 C5 CC C5 CE C9 C5 20 CE ассива¦Деление н
BAD0 C1 20 CE CF CC D8 00 EE C5 C4 CF D0 D5 D3 D4 C9 а ноль¦Недопусти
BAE0 CD CF 20 D7 20 CB CF CD C1 CE C4 CE CF CD 20 D2 мо в командном р
BAF0 C5 D6 C9 CD C5 00 EE C5 D3 CF CF D4 D7 C5 D4 D3 ежиме¦Несоответс
BB00 D4 D7 C9 C5 20 D4 C9 D0 CF D7 20 C4 C1 CE CE D9 твие типов данны
BB10 C8 00 EE C5 D4 20 CD C5 D3 D4 C1 20 C4 CC D1 20 х¦Нет места для
BB20 D3 D4 D2 CF CB 2E 20 D0 C5 D2 C5 CD C5 CE CE D9 строк. переменны
BB30 C8 00 F3 CC C9 DB CB CF CD 20 C4 CC C9 CE CE C1 х¦Слишком длинна
BB40 D1 20 D3 D4 D2 CF CB C1 00 F3 D4 D2 CF CB 2E 20 я строка¦Строк.
BB50 D7 D9 D2 C1 D6 C5 CE C9 C5 20 D3 CC C9 DB CB CF выражение слишко
BB60 CD 20 D3 CC CF D6 CE CF 00 F0 D2 CF C4 CF CC D6 м сложно¦Продолж
BB70 C5 CE C9 C5 20 CE C5 D7 CF DA CD CF D6 CE CF 00 ение невозможно¦
BB80 EE C5 CF D0 D2 C5 C4 2E 20 C6 D5 CE CB C3 C9 D1 Неопред. функция
BB90 20 D0 CF CC D8 DA CF D7 C1 D4 C5 CC D1 00 EF DB  пользователя¦Ош
BBA0 C9 C2 CB C1 20 D5 D3 D4 D2 CF CA D3 D4 D7 C1 20 ибка устройства
BBB0 D7 D7 CF C4 C1 2D D7 D9 D7 CF C4 C1 00 EF DB C9 ввода-вывода¦Ошi
BBC0 C2 CB C1 20 D0 D2 C9 20 D0 D2 CF D7 C5 D2 CB C5 вка при проверке
BBD0 20 D0 D2 CF C7 D2 C1 CD CD D9 00 EF D4 D3 D5 D4  программы¦Отсут
BBE0 D3 D4 D7 D5 C5 D4 20 CF D0 C5 D2 C1 D4 CF D2 20 ствует оператор
BBF0 52 45 53 55 4D 45 00 EF D0 C5 D2 C1 D4 CF D2 20 RESUME¦Оператор
BC00 52 45 53 55 4D 45 20 C2 C5 DA 20 CF DB C9 C2 CB RESUME без ошибк
BC10 C9 00 EF DB C9 C2 CB C1 20 C2 C5 DA 20 D7 D9 D7 и¦Ошибка без выв
BC20 CF C4 C1 20 D3 CF CF C2 DD C5 CE C9 D1 00 F0 D2 ода сообщения¦Пр
BC30 CF D0 D5 DD C5 CE 20 CF D0 C5 D2 C1 CE C4 00 F0 опущен операнд¦П
```

```
      Ø  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F Ø123456789ABCDEF
BC4Ø C5 D2 C5 DØ CF CC CE C5 CE C9 C5 2Ø C2 D5 C6 C5 ереполнение буфе
BC5Ø D2 C1 2Ø D3 D4 D2 CF CB C9 ØØ FØ C5 D2 C5 DØ CF ра строки!Перепо
BC6Ø CC CE C5 CE C9 C5 2Ø DØ CF CC D1 ØØ F7 CE D5 D4 лнение поля!Внут
BC7Ø D2 C5 CE CE D1 D1 2Ø CF DB C9 C2 CB C1 ØØ EF DB ренняя ошибка!Ош
BC8Ø C9 C2 CF DE CE D9 CA 2Ø CE CF CD C5 D2 2Ø C6 C1 ибочный номер фа
BC9Ø CA CC C1 ØØ E6 C1 CA CC 2Ø CE C5 2Ø CE C1 CA C4 йла!Файл не найд
BCAØ C5 CE ØØ E6 C1 CA CC 2Ø D5 D6 C5 2Ø CF D4 CB D2 ен!Файл уже откр
BCBØ D9 D4 ØØ F7 D7 CF C4 2Ø DØ CF D3 CC C5 2Ø CB CF ыт!Ввод после ко
BCCØ CE C3 C1 2Ø C6 C1 CA CC C1 ØØ EF DB C9 C2 CF DE нца файла!Ошибоч
BCDØ CE CF C5 2Ø C9 CD D1 2Ø C6 C1 CA CC C1 ØØ EB CF ное имя файла!Ко
BCEØ CD C1 CE C4 CE D9 CA 2Ø CF DØ C5 D2 C1 D4 CF D2 мандный оператор
BCFØ 2Ø D7 2Ø C6 C1 CA CC C5 ØØ F4 CF CC D8 CB CF 2Ø  в файле!Только
BDØØ DØ CF D3 CC C5 C4 CF D7 C1 D4 2E 2Ø D7 D7 CF C4 последоват. ввод
BD1Ø 2D D7 D9 D7 CF C4 ØØ E6 C1 CA CC 2Ø CE C5 2Ø CF -вывод!Файл не о
BD2Ø D4 CB D2 D9 D4 2Ø CB CF CD C1 CE C4 CF CA 2Ø 4F ткрыт командой O
BD3Ø 5Ø 45 4E ØØ 2Ø D7 2Ø D3 D4 D2 CF CB 65 2Ø ØØ EF PEN! в строке !O
BD4Ø D3 D4 C1 CE CF D7 ØØ 3C 69 F1 3D 7E F1 4Ø 92 F1 станов!<==  Ин-
BD5Ø 41 A2 F1 42 B6 F1 43 C8 F1 46 DD F1 ØØ EF DB C9 формация ==>!Оши
BD6Ø C2 CB C1 2Ø D7 2Ø 46 41 54 2D D4 C1 C2 CC C9 C3 бка в FAT-таблиц
BD7Ø C5 ØØ EE C5 DØ D2 C1 D7 2E 2Ø D7 D9 DA CF D7 2Ø е!Неправ. вызов
BD8Ø C6 C1 CA CC C1 ØØ E6 C1 CA CC 2Ø C5 DD C5 2Ø CF файла!Файл еще о
BD9Ø D4 CB D2 D9 D4 ØØ E6 C1 CA CC 2Ø D5 D6 C5 2Ø D3 ткрыт!Файл уже с
BDAØ D5 DD C5 D3 D4 D7 D5 C5 D4 ØØ 52 41 4D 2D C4 C9 уществует!RAM-ди
BDBØ D3 CB 2Ø DA C1 DØ CF CC CE C5 CE ØØ F3 CC C9 DB ск заполнен!Слиш
BDCØ CB CF CD 2Ø CD CE CF C7 CF 2Ø C6 C1 CA CC CF D7 ком много файлов
BDDØ ØØ 52 41 4D 2D C4 C9 D3 CB 2Ø CE C5 2Ø C7 CF D4 !RAM-диск не гот
BDEØ CF D7 ØØ 3F FØ CF D7 D4 CF D2 C9 D4 C5 2Ø D7 D7 ов!?Повторите вв
BDFØ CF C4 21 ØA ØD ØØ ØA ØD F3 CD 2E 2Ø D3 D4 D2 CF од!  !  См. стро
BE ØØ CB D5 2Ø ØØ EC C9 DB CE C9 C5 2Ø C4 C1 CE CE D9 ку !Лишние данны
BE1Ø C5 2Ø C9 C7 CE CF D2 C9 D2 D5 CØ D4 D3 D1 21 Ø1 е игнорируются!В
BE2Ø ØØ Ø1 3E 3D BC 2Ø Ø8 3E 75 BD 2Ø Ø3 21 ØØ EE CD   <==Замена     ▲
BE3Ø 4Ø F2 C9 FB Ø1 52 F1 Ø3 Ø3 ØA Ø3 B7 C8 93 2Ø F7 сообщений:      |
BE4Ø 69 6Ø 5E 23 56 21 5E F5 E5 77 13 23 1A 77 B7 2Ø ч.1 л.FEFD    Ва-
BE5Ø F9 E1 1E Ø1 C9 E3 E1 3E 3F BC 2Ø ØB 3E DC BD 2Ø   ==><==       ри-
BE6Ø Ø6 21 4C F1 32 2D F2 E5 CD D2 ØØ E1 7E FE 3F 2Ø                  ант
BE7Ø Ø3 21 1F FØ CD 78 66 3E Ø7 DF E1 7C A5 3C C4 95
BE8Ø F2 AF 32 2D F2 C3 22 41 E5 21 2C F2 AF BE 2Ø ØB     часть 2
BE9Ø 21 Ø3 F2 CD 78 66 E1 CD 12 34 C9 23 BE 28 Ø5 21  ловушка  FF02
BEAØ 41 F1 18 EF E1 11 FC 13 Ø1 Ø9 ØØ 21 FØ FB 22 FA
BEBØ F3 36 1E 23 36 15 EB 13 ED BØ ED 53 F8 F3 C9 E3                ==><
BECØ E1 3A A6 F6 B7 C2 4F 4Ø C1 21 FØ F1 C3 5B 4B E5 ==ч.3 л.FF61==><
BEDØ 21 2E F2 7E 36 Ø1 E1 B7 CØ E5 D5 C5 CD FØ F2 C1 ==
BEEØ D1 E1 C9 11 2Ø ØØ 21 ØØ 18 3A AF FC FE Ø1 28 1Ø     часть 4
BEFØ B7 CØ 26 ØØ 11 28 ØØ 3A BØ F3 3D BB 38 Ø2 1E 5Ø  ловушка  FF5C
BFØØ 3A DC F3 3D 3D B7 28 Ø3 19 18 F9 43 CD 4A ØØ 23
BF1Ø FE 3F 28 Ø3 1Ø F6 C9 CD 4A ØØ 4F 3A 7C F3 B9 CØ (* ловушка FDEØ
BF2Ø E5 Ø6 ØC 11 31 4C 23 CD 4A ØØ 4F 1A B9 2Ø B2 13 обрабат. в раб.
BF3Ø 1Ø F4 D1 21 11 F2 Ø1 1B ØØ CD 5C ØØ C9 ØA ØD F2 обл. *)   ==>  Р
BF4Ø D5 D3 D3 CB C1 D1 2Ø C4 C9 C1 C7 CE CF D3 D4 C9 усская диагности
BF5Ø CB C1 2Ø CF DB C9 C2 CF CB ØA ØD F7 C5 D2 D3 C9 ка ошибок  Верси
BF6Ø D1 2Ø 31 2E 32 2C 2Ø 38 2E 31 39 38 39 2Ø 2Ø 45 я 1.2, 8.1989
BF7Ø 53 43 ØØ FD ØØ
```


Предположим, вы ввели все требуемые коды в память ЭВМ и записали их в файл командой.

BSAVE «ПРД», &hB900, &hBF74, &hB900

Теперь необходимо переслать файл на ученическую машину и запустить. Если вы работаете со стандартной сетью, то используйте для пересылки команду CALL BSEND. После завершения передачи программа запускается на ученической машине вводом строки

DEFUSR=&hB900:A=USR(A)

(эти команды можно также переслать по сети с учительской машины). При использовании монитора BASMON достаточно выполнить команду

T, N, ПРД, r (N — номер машины), после выполнения которой переданный файл запустится автоматически.

Автор надеется, что после запуска ПРД на экране вашего дисплея не появилось сообщение «Ошибка при проверке программы. Повторите ввод!», что возможно из-за несовпадения контрольной суммы с эталоном. В случае удачного старта остается только окончательно убедиться в работоспособности программы. При проверке полезным оказывается оператор ERROR, позволяющий имитировать ошибку с заданным номером. Советуем также испытать реакцию ЭВМ на неправильный ввод в операторе INPUT, так как она определяется особой частью ПРД.

В заключение — несколько слов об устройстве программы. ПРД можно разделить на три блока: загрузчик из двух частей, тексты сообщений об ошибках и блок замены сообщений. Файл ПРД вводится с адреса B900H, который является началом первой части загрузчика. В ее функции входит: убедиться, что компьютер серии «MSX-2»; проверить, что в машине отсутствует дисковый Бейсик, меняющий распределение памяти; вычислить контрольную сумму; если все вышеуказанные проверки не вызовут прекращения работы программы, подготовить автоматический запуск второй части загрузчика и инициировать команду CLEAR для резервирования адресов памяти EE00H—F37FH. Затем стартует вторая часть загрузчика, которая выполняет следующее: переписывает тексты сообщений и блок замены сообщений в указанную область памяти; изменяет ловушки; очищает экран и выводит заголовок. После этого управление возвращается Бейсику. Отметим, что в результате загрузки ПРД свободная оперативная память уменьшается на 1,4K байт.

В результате работы загрузчика ловушки переопределяются, и блок замены сообщений ПРД получает возможность в нужный момент вмешиваться в работу Бейсик-интерпретатора. В этом блоке четыре независимые части — каждая для действий по своей ловушке. Первая заменяет адрес основного массива сообщений и задает новое начало для текстов об ошибках диска ОЗУ. Вторая служит для указания адреса текста об останове, а также выводит информацию о номере строки с ошибкой в соответствии с выбранным вариантом A или B. Небольшая третья часть «переводит» малопонятный для начинающего пользователя текст «Redo from start». И наконец, четвертая, достаточно сложная часть в соответствии с режимом SCREEN и шириной строки находит и заменяет в видеопамяти уже помещенный туда текст «Extra ignored». К сожалению, для перехвата управления до вывода текста ловушки найти не удалось.

Как следует из приведенного выше описания, ПРД нигде не использует аппаратные особенности «MSX-2». Учитывая также, что ПЗУ Бейсик-интерпретатора в основном не изменилось, автор убежден в возможности переноса ПРД на компьютер «MSX-1». Работа в этом направлении будет продолжена.

Адрес для справок: 614600, Пермь, ГСП, ул. К. Маркса, 24, Пермский государственный педагогический институт, кафедра теоретической физики, Е. А. Еремину.

В. НАСТАСЕНКО
г. Кишинев

Игровой пульт

Игровые пульты (джойстики) — мечта каждого заведующего кабинетом ВТ. Тишина в классе, клавиатура как новенькая — разве этого мало? К тому же наличие джойстиков позволяет создавать качественно новые динамические игры. Само же конструирование активизирует детское техническое творчество и помогает выработать бережное отношение к технике.

Каждый компьютер КУВТ «Ямаха» имеет два разъема для подключения игровых пультов. От каждого игрового пульта может быть принято шесть различных сигналов. В Бейсике их обработка производится функцией STICK ( ) (по позициям координатной ручки) и прерываниями по ON STRIG GOSUB (по нажатию на пробел и на кнопки A и Б триггеров игровых пультов).

Приведем данные по контактам.

№	Сигнал	Направление
1	ВВЕРХ	вход
2	ВНИЗ	вход
3	ВЛЕВО	вход
4	ВПРАВО	вход
5	+5 В	—
6	ТРИГГЕР А	вход-выход
7	ТРИГГЕР В	вход-выход
8	ВЫХОД	выход
9	земля	

Схема подключения джойстика приведена на рис. 1.

Первый сконструированный нами джойстик был предельно прост, почему и оказался, наверное, самым «живучим».

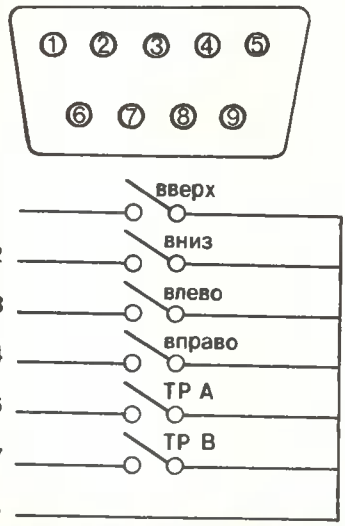
Для его изготовления понадобится пластмассовая коробка размером ориентировочно 120×70×25 мм, с высотой внутреннего пространства около 20 мм, стальной шар с ручкой (например, нижняя часть телескопической антенны радиоприемника или шарик от подшипника, насаженный на винт), резиновая пробка.

Крышка коробки будет служить основа-

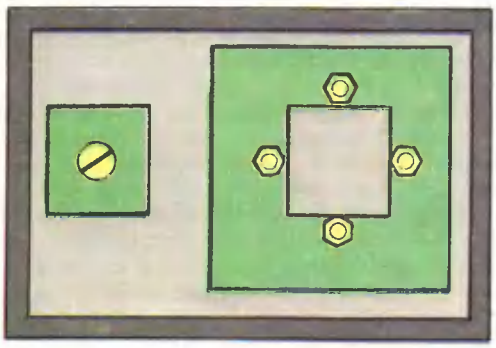
нием джойстика. На ней расположены четыре контакта для координатной ручки и контакт для триггера А (рис. 2).

Расстояние между контактами подбирается индивидуально, в зависимости от радиуса стального шара, так, чтобы, при перемещений координатной ручки, например, на себя замыкался верхний контакт, а при перемещении по диагонали (под углом 45°) — два

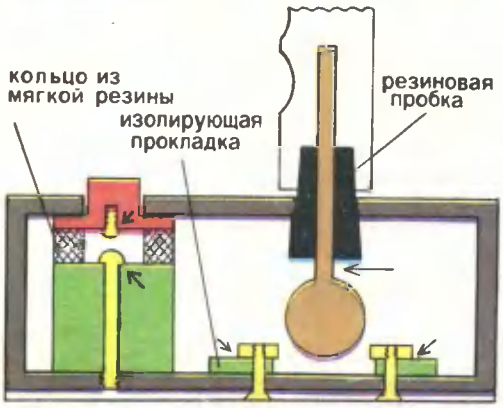
1



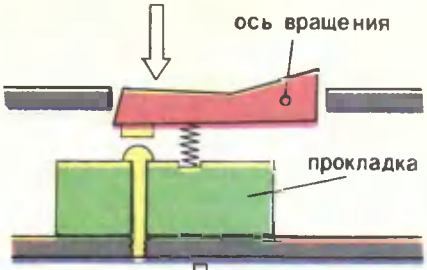
2



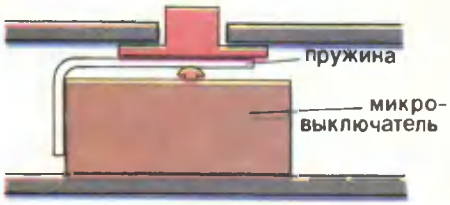
3



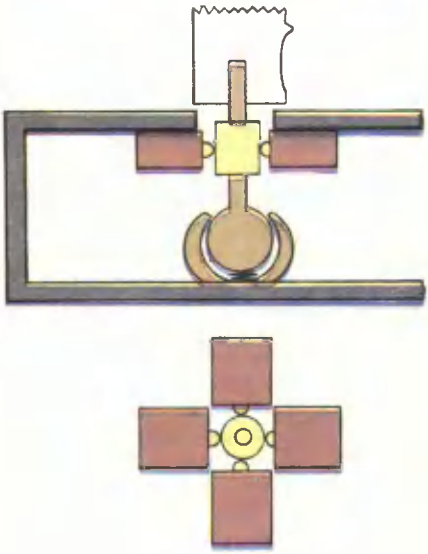
4



5



6



контакта. Для улучшения взаиморасположения с координатной ручкой контакты можно поднять на непроводящей пластине.

В дне коробки сверлим такое отверстие, чтобы резиновая пробка туго вошла в него. В пробку вставляем координатную ручку (рис. 3). Стрелками показаны места подсоединения проводов.

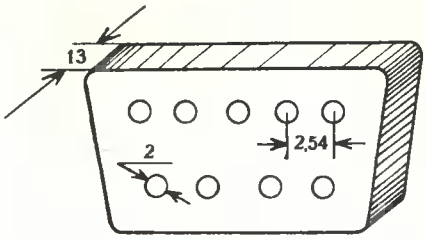
Удобную координатную ручку можно сделать, вырезав из дерева или пластмассы ее верхнюю часть и насадив на трубку, соединяющуюся со стальным шаром.

Если трубка полая, то в верхнюю часть координатной ручки можно вставить микровыключатель, контакты от которого пойдут на триггер В, и закрыть его гибкой пластиной.

Кнопки триггеров можно выполнить разными способами (рис. 3—5). С помощью четырех микровыключателей и телескопического шарнира можно изменить и конструкцию координатной ручки (рис. 6). В последнем варианте микровыключатели можно заменить простыми контактами, а цилиндр сделать из проводящего материала.

Для устойчивости игрового пульта на днище полезно прикрепить резиновые присоски.

Вилку для разъема можно изготовить из куска эбонита или оргстекла, просверлив отверстия диаметром 2 мм, как показано на рис. 7. Металлические контакты цилиндрической формы можно взять из обычных розеток для магнитофона, закрепив их эпоксидным клеем.



В заключение сообщу, что изготовили эти пульты бывшие ученики, ныне студенты Кишиневского политехнического института Олег Зияев и Олег Васенев.

КЛУБ БК

Особенности транслятора с языка Бейсик для БК-0010.01

Большинство трансляторов с Бейсика являются интерпретаторами. Но на БК-0010.01 реализован компилятор Бейсика: исходная программа после ее запуска командой RUN преобразуется в памяти ЭВМ в прямой шитый код*. Он представляет собой последовательность адресов подпрограмм, реализующих соответствующие операторы Бейсика, и исходных данных: констант, адресов переменных и т. д. Подпрограммы «зашиты» в ПЗУ и являются частью компилятора, Каждый оператор программы на Бейсике реализуется с помощью одной или нескольких таких подпрограмм. Зная адреса этих подпрограмм и способ передачи им параметров, можно существенно облегчить написание программ в машинных кодах или в таких системах программирования, как МИКРО-10, МИКРО-11, так как при этом открывается сравнительно простой доступ ко многим средствам языка высокого уровня.

Остановимся на методике нахождения адресов подпрограмм для версии Бейсика «Вильнюс 1986.07.24», поставляемой заводом-изготовителем вместе с БК-0010.01.

В слове по адресу 2002 хранится адрес последнего байта текста введенной программы на Бейсике. Соответствующий ей шитый код транслятор начинает размещать со следующего слова ОЗУ (ближайший больший четный адрес). В этом легко убедиться, если ввести программу, запустить ее командой RUN, а затем, остановив программу, подключить к микро-ЭВМ блок МСТД, войти в тестовый монитор и просмотреть содержимое памяти, руководствуясь числом, хранящимся в ячейке 2002. Но такой способ, конечно, не слишком удобен, если требуется определить шитый код, соответствующий различным командам, операторам и операциям, так как в программе большого объема разобраться достаточно сложно, а повторять такую процедуру для каждого оператора слишком утомительно. Удобнее для этих целей написать сервисную программу, позволяющую просматривать содержимое памяти как в числовом, так и в символьном виде, а также выполнять ряд вспомогательных функций. Представление операторов и команд языка в шитом коде было определено с помощью следующей программы:

```
10 REM ПРОГРАММА ЗАПУСКАЕТСЯ
20 REM КОМАНДОЙ "RUN 100"
30 REM ЗДЕСЬ ДОЛЖНЫ НАХОДИТЬСЯ
40 REM ВАШИ ОПЕРАТОРЫ
50 ...
100 CLS
110 N%=PEEK(&O2002)
120 IF N% MOD 2%=1% THEN N%=N%+1%
130 IF PEEK(N%)=&O156776 THEN STOP
```

* Детальное определение шитого кода см., например: Баранов С. Н., Ноздрунов Н. Р. Язык Форт и его реализации. М.: Машиностроение, 1988. С. 43.

```
140 ? OCT$(PEEK(N%))
150 N%=N%+2%
160 GOTO 130
170 END
```

В ее первых строках размещаются операторы, шитый код которых требуется определить. После запуска программа выводит на экран в столбик в восьмиричном виде соответствующий им шитый код. Вывод кодов производится до первого встреченного оператора CLS (адрес подпрограммы в ПЗУ равен 156776).

В приводимой ниже таблице каждому оператору языка, записанному в общей форме, сопоставлен соответствующий ему шитый код — содержимое последовательно расположенных в ОЗУ микро-ЭВМ 16-разрядных слов. В шитом коде кроме адресов подпрограмм могут располагаться константы, адреса переменных и другие данные.

Типы параметров, приводимых в таблице, соответствуют стандарту языка Бейсик-MSX и обозначаются: % — целое число; ! — дробное число одинарной точности; # — дробное число двойной точности. Параметры, помещенные в правой части таблицы в скобках, означают, что их значения необходимо переслать в стек в порядке их перечисления. На вершину стека указывает регистр R6 процессора микро-ЭВМ. Переслать параметры в стек можно при помощи вспомогательных подпрограмм Бейсика, также перечисленных в таблице. Восьмеричные числа в правой части таблицы — это адреса подпрограмм, вызов которых приводит к выполнению данного оператора. Некоторые неудобства при таком подходе к составлению программ доставляет способ вызова подпрограммы: для этого требуется записать в регистр R4 адрес слова в ОЗУ, где расположен адрес подпрограммы, а затем выполнить команду JMP @(R4)+. Такова особенность компилятора — передача управления от одной подпрограммы к другой осуществляется через регистр R4 по команде JMP @(R4)+, содержащейся в конце каждой подпрограммы. В связи с этим желательно весь список адресов подпрограмм записывать в ОЗУ подряд, тогда автоматически после выполнения одной подпрограммы будет вызываться на исполнение следующая.

В качестве параметров, передаваемых подпрограммам, могут использоваться как константы, так и содержимое некоторых переменных, значение которых можно переслать в стек с помощью соответствующих вспомогательных подпрограмм.

В конце таблицы приведены адреса некоторых системных ячеек Бейсика, знание которых может оказаться полезным. Известны также и адреса подпрограмм, реализующих команды Бейсика, однако механизм передачи им параметров авторами пока еще не установлен.

Если теперь в памяти ЭВМ последовательно разместить адреса необходимых подпрограмм и исходные данные для нескольких операторов Бейсика, то, в принципе, можно получить некую исполняемую программу. По сути дела, этим проделывается работа, аналогичная той, которую выполняет компилятор Бейсика после команды RUN.

Иллюстрацией может служить программа, очищающая экран, рисующая в его левом верхнем углу квадрат и закрашивающая его текущим цветом. Она написана на ассемблере МИКРО-10 и выглядит так:


```
        MOV #PRG, R4
        JMP @(R4)+
PRG:    .#156776.#156160.#0.#156160.#0
        .#156160.#62.#156160.#62
        .#125454.#156160.#24.#156160
        .#24.#125376.#125404.#125642
        .#155560.#PRG
      END
```

Запуск этой программы на исполнение производится в режиме 32 символа в строке. Соответствующая ей программа на Бейсике выглядит так:

```
10 CLS
20 LINE(0%,0%)-(&062,&062), B
30 PAINT(&024,&024)
40 GOTO 10
```

Необходимо помнить, что параметр целого типа, передаваемый подпрограммам Бейсика, занимает в ОЗУ 2 байта, дробный параметр одинарной точности — 4 байта, двойной точности — 8 байт. Ввод строки с клавиатуры оператором INPUT производится в буфер длиной 256 байт, расположенный с адреса 2422_8.

Операторы

KEY A%,B$	(A%, адрес B$, длина B$) 124570
PRINT A%	(A%) 157070, 156770
PRINT A!	(A!) 157014, 156770
PRINT A#	(A#) 157002, 156770
PRINT A$	(адрес A$, длина A$) 156460, 156770
INPUT X1,...,Xn	160634, 157150, адрес следующей команды, список адресов X1,...,Xn

OPEN A$ FOR OUTPUT	(адрес A$, длина A$) 162336
OPEN A$ FOR INPUT	(адрес A$, длина A$) 163360
CLOSE	162656
POKE A%,B%	(A%, B%) 160536
OUT A%,B%,C%	(A%, B%, C%) 160542
CLS	156776
COLOR A%,B%	(A%, B%) 125172, 125242
LOCATE A%,B%,C%	(A%, B%) 124512, (C%), 124324
PSET(A%,B%),C%	(A%, B%, C%) 125242, 125410, 125220
LINE(A%,B%)-(C%,D%),E%	(A%, B%, C%, D%, E%) 125242, 125426, 125220
LINE(A%,B%)-(C%,D%),E%,B	(A%, B%, C%, D%, E%) 125242, 125454, 125220
CIRCLE(A%,B%),C%,D%	(A%, B%, C%, D%) 125334, 126724, 126742, 127122, 127324
PAINT(A%,B%),C%,D%	(A%, B%, C%) 125334, (D%), 125334, 125642
DRAW A$	(адрес A$, длина A$) 131412
BEEP	156762
TRON	154626
TROFF	154674
ON A% GOTO m1,...,mn	(A%) 155564, n, адреса m1,...,mn
STOP	154754
END	154722
GOTO N	155560, адрес N
GOSUB N	155526, адрес N
RETURN	155722
FOR I%=A% TO B% STEP C%	(A%) 156350, адрес I%, (B%, C%), 156426, адрес следующего оператора, адрес I%
FOR I#=A# TO B% STEP C#	(A#) 156334, адрес I#, (B#, C#), 156404, адрес следующего оператора, адрес I#
NEXT I%	156466, адрес I%, 156572
NEXT I#	156466, адрес I#, 156636, 155176, 155466
IF...THEN...ELSE	сравнение, 155404, адрес ветви ELSE, ветвь THEN, 155560, адрес оператора, следующего за ELSE, ветвь ELSE

Отношения и логические операции

A%<B%	(A%, B%) 155172, 155452
A#<B#	(A#, B#) 177034, 155452
A%>B%	(A%, B%) 155172, 155456
A#>B#	(A#, B#) 177034, 155456
A%>=B%	(A%, B%) 155172, 155446
A#>=B#	(A#, B#) 177034, 155446
A%<=B%	(A%, B%) 155172, 155442
A#<=B#	(A#, B#) 177034, 155442
A%<>B%	(A%, B%) 155172, 155436
A#<>B#	(A#, B#) 177034, 155436
NOT A%	(A%) 162306
A% AND B%	(A%, B%) 162316
A% OR B%	(A%, B%) 162312
A% XOR B%	(A%, B%) 162324
A% EQV B%	(A%, B%) 162340
A% IMP B%	(A%, B%) 162344
A%=B%	(A%, B%) 155172, 155430
A#=B#	(A#, B#) 177034, 155430

Функции

SQR(X#)	(X#) 171350	CHR$(A%)	(A%) 160510

```
SIN(X#)   (X#) 173614
COS(X#)   (X#) 173566
TAN(X#)   (X#) 174306
ATN(X#)   (X#) 174434
EXP(X#)   (X#) 171762
PI       167102
LOG(X#)   (X#) 173052
ABS(X#)   (X#) 166566
ABS(X%)   (X%) 166556
FIX(X#)   (X#) 176212
INT(X#)   (X#) 176340
SGN(X%)   (X%) 166572
SGN(X!)   (X!) 166624
SGN(X#)   (X#) 166614
RND(X#)   (X#) 175176
FRE(X%)   (X%) 160600
FRE(A$)   (адрес A$,
            длина A$)
              160620
CINT(A#)  (A#) 160760
CSNG(A#)  (A#) 162020
CDBL(A!)  (A!) 166742
CDBL(A%)  (A%) 166646
PEEK(A%)  (A%) 160556
INP(X%,Y%)   (X%, Y%)
              160564
ASC(A$)    (адрес A$,
            длина A$)
              161534

LEN(A$)      (адрес A$,
               длина A$)
                 161530
MID$(A$,B%,C%)   (адрес A$,
                   длина A$,
                     B%, C%)
                      161340
STRING(X%,A$)    (X%) 160634,
                  (адрес A$,
                  длина A$),
                      161404
VAL(A$)      (адрес A$,
               длина A$)
                 161550

INKEY $      124132
STR$(A%)     (A%) 161610
OCT$(A%)     (A%) 161624
HEX$(A%)     (A%) 161734
BIN$(A%)     (A%) 161672
CSRLIN(A%)   (A%) 124560
POS(A%)      (A%) 124544
LPOS(A%)     (A%) 157054
EOF          163040
AT(A%,B%)    (A%, B%) 124512
TAB(A%)      (A%) 124420
POINT(A%,B%)      (A%, B%)
                     125602
```


Адреса вспомогательных подпрограмм

```
156160,A% - запись слова A% в стек
156156,A! - запись двух слов в стек
156152,A# - запись четырех слов в стек
156232,A% - пересылка слова по адресу A% в стек
156214,A% - пересылка двух слов по адресу A% в стек
156170,A% - пересылка четырех слов по адресу A%
                                          в стек
156350,A% - запись слова из стека по адресу A%
156330,A% - запись двух слов из стека по адресу A%
156334,A% - запись четырех слов из стека по адресу A%
124164    - вызов вспомогательного ПЗУ (CALL)
156770    - вывод кода ВК
124324    - включение курсора
166562    - изменение знака целого числа в стеке
156750    - изменение знака вещественного числа
                                          в стеке
125242    - установка цвета (значение цвета
                              находится в стеке)
```

Некоторые системные адреса Бейсика

```
2006      - количество зарезервированных байтов
                                          (CLEAR)
2002      - содержит адрес конца программы
2050      - флаг трассировки (1 - включена,
                                  0 - выключена)
37400     - буфер ввода-вывода с магнитофона
120300    - адрес аварийного входа в Бейсик
121052    - вызов строкового редактора текстов
                                        Бейсика
146404    - адрес подпрограммы обработки прерывания
                                по прерыванию TRAP
120234    - адрес подпрограммы обработки прерывания
                                по клавише СТОП
120540    - адрес подпрограммы обработки прерывания
                                  по вектору 10
```

Команды Бейсика

```
136650  CLOAD      136104  LOAD       124302  MONIT
136420  BLOAD      137256  SAVE       134626  LLIST
```

```
136306  FIND        134600  NEW         141034  RENUM
137124  CSAVE       140376  AUTO        140514  DELETE
136562  BSAVE       134634  LIST        136066  CONT
                                        135242  RUN
```

Арифметические операции

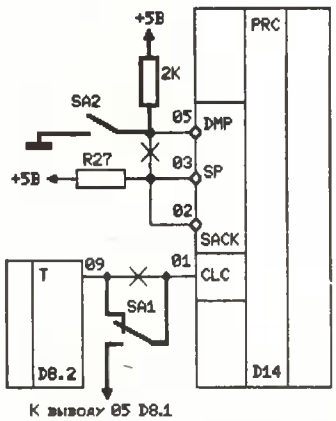
```
A%+B%   (A%, B%) 162102     A#/B#   (A#, B#) 170776
A#+B#   (A#, B#) 167144     A%\B%   (A%, B%) 162052
A%-B%   (A%, B%) 162112     A%*B%   (A%, B%) 162116
A#-B#   (A#, B#) 167124     A#*B#   (A#, B#) 170210
```

В. АВСЕЕВ, А. АВСЕЕВ

Увеличение тактовой частоты

Худшим из языков программирования для БК считается Фокал (в силу своей уникальной медлительности). Тем не менее существует возможность повысить быстродействие компьютера при работе с этим и некоторыми другими языками. Для этого необходимо увеличить тактовую частоту микропроцессора. Как показал опыт, удвоение тактовой частоты совершенно не сказывается на надежности работы БК, а программы выполняются почти в два раза быстрее.

Фрагмент схемы доработанного компьютера приведен на рисунке. Соединения, которые необходимо разорвать, перечеркнуты, а новые элементы изображены жирной линией. Микросхемы показаны в упрощенном виде. Переключатели SA1 и SA2 — микротумблеры; их можно установить на заднюю панель БК. Переключателем SA1 изменяют частоту, а SA2 переводит процессор в состояние прямого доступа к памяти (ПДП).



Последовательность действий при переходе с одной скорости на другую такова: сначала приостанавливаем выполнение текущей программы, переведя переключатель SA2 в нижнее по схеме положение, затем с помощью тумблера SA1 изменяем тактовую частоту и возвращаем переключатель SA2 в исходное положение. Переключать скорости при активно работающем микропроцессоре нельзя — произойдет сбой. Кстати, переключатель SA2 реализует дополнительное удобство: возможность временного приостанова выполнения любой программы (аналогично действию клавиши ШАГ в Фокале).

Следует учесть, что успешно работать с магнитофоном БК может только при нормальной тактовой частоте (3 МГц).

Д. АНТОНОВ

Спрайты для БК

Для программного вывода на экран небольших изображений в стандарте Бейсика MSX имеется оператор SPRITE$, который позволяет задать до 256 картинок (спрайтов) размером 8×8 или 64 спрайта 16×16. С его помощью можно создавать и плавно перемещать спрайты по экрану. Безусловно, такой оператор очень полезен, но в БК его нет. Что делать?

Ниже приведено несколько программ для БК, позволяющих получить на экране спрайты размером 8×10 или 16×20 точек. Координаты вывода и размеры спрайтов не меняются при переключении размеров курсора (32/64).

При использовании программ надо будет в начале основной программы поместить оператор PRINT CHR$(140); CHR$(140), да и в самой программе нельзя «сдвигать» экран вверх; самой верхней точке экрана с коорди-

натами (0, 0) должен постоянно соответствовать адрес &O42000.

Можно использовать программы, написанные на Бейсике,— их легче ввести в основную программу. Можно использовать программы в кодах — они значительно быстрее выводят спрайт на экран.

Недостаток моих программ — крупный шаг координат спрайтов по горизонтали (он равен широкому курсору). Желающие могут доработать их — сделать шаг координат равным одной точке.

Вариант на Бейсике. Приведенные ниже программы выводят на экран спрайт в соответствии со следующими параметрами:
N — номер спрайта (0—...);
X — координата X (0—31);
Y — координата Y (0—229).

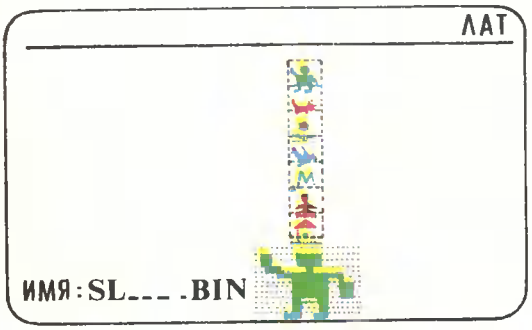
Программы используют файл, содержащий коды изображений спрайтов. Он записывается в память с адреса:

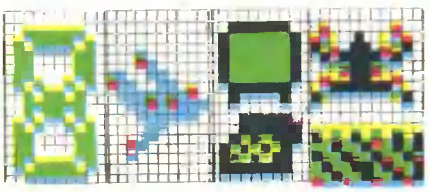
&O27774 — если перед кодами изображений спрайтов идут 4 байта, содержащие данные о ширине и высоте столбца спрайтов (результат работы редактора GRED0);

&O30000 — если коды начинаются сразу.

```
10 'Программа SPRITE 8*10
20 NN=N*&024
30 XX=X+X
40 YY=Y*&0100
50 XY=XX+YY+&042000
60 FOR I=&030000+NN TO &030023+NN
                                  STEP &02
70    POKE XY,PEEK(I)
80    XY=XY+&0100
90 NEXT I

10 'Программа SPRITE 16*20
20 NN=N*&0120
```





```
30 XX=X+X
40 YY=Y*&0100
50 XY=XX+YY+&042000
60 FOR I=&030000+NN TO &030116+NN
                                  STEP &04
70    POKE XY,PEEK(I)
80    POKE XY+2,PEEK(I+2)
90    XY=XY+&0100
100 NEXT I
```

Вариант в кодах. Следующие программы выводят на экран спрайт, параметры которого заданы в следующих ячейках:

&O37770 содержит в себе номер спрайта (0—...);

&O37772 содержит в себе координату X (0—31);

&O37774 содержит в себе координату Y (0—229).

Файл спрайтов записывается с адреса:

&O30174 — если коды изображения идут не сразу;

&O30200 — если коды изображения идут сразу.


```
10 'Программа SPRITE 8*10
20 DATA 5572,16376,4865,5568,20,2563,
                                  2562,3009
30 DATA 777,1539,2817,26050,1,24579,
                                  32322,3010
40 DATA 769,2819,4293,5572,16380,
                                  4865,5568,64
50 DATA 2563,2562,3009,777,1539,
                                  2817,26050,1
60 DATA 24579,32322,3010,769,2819,
                                  5568,16378,4612
70 DATA 3268,24772,26052,17408,
                                  5571,12416,24899,4290
80 DATA 26050,19,5324,26052,64,8386,
                                  2043,2560
90 DATA 2561,2562,2563,2564,2565,
                                  135,0,0
100 DATA -1,0,-1,0
120 FOR A=&030000 TO &030200 STEP &02
130    READ K
140    POKE A,K

150 NEXT A
160 END

10 'программа SPRITE 16*20
20 DATA 5572,16376,4865,5568,80,2563,
                                  2562,3009
30 DATA 777,1539,2817,26050,1,24579,
                                  32322,3010
40 DATA 769,2819,4293,5572,16380,
                                  4865,5568,64
50 DATA 2563,2562,3009,777,1539,
                                  2817,26050,1
60 DATA 24579,32322,3010,769,2819,
                                  5568,16378,4612
70 DATA 3268,24772,26052,17408,5571,
                                  12416,24899,4290
80 DATA 26050,78,5332,5324,26052,62,
                                  8386,2042
90 DATA 2561,2562,2563,2564,2565,
                                  135,0,0
100 DATA -1,0,-1,0
120 FOR A=&030000 TO &030200 STEP &02
130    READ K
140    POKE A,K
```

```
150 NEXT A
160 END
```

Будьте осторожны — использовать сразу оба типа программ нельзя, так как возникнет путаница из-за использования одинаковых адресов для записи файла спрайтов.

```
10 '  Пример использования
20 '       подпрограммы
25 '       SPRITE 16x20
30 '------------------------
40 DEF USR=&O30000
50 INPUT "Номер";N
60 PRINT CHR$(140);CHR$(140)
70 POKE &O37770,N
80 FOR Y=0 TO 200 STEP 20
90    FOR X=0 TO 30 STEP 2
100      POKE &O37772,X
110      POKE &O37774,Y
120      A=USR(A)
130   NEXT X
140 NEXT Y
150 GOTO 50
```

Подготовка спрайтов. Изображение спрайтов подготавливается с помощью графического редактора типа GRED0. Рисовать спрайты надо один под другим — столбиком, тогда при записи на магнитофон будут идти последовательно сперва 4 байта, несущие информацию о высоте и ширине изображения (столбца), потом байты первого спрайта, затем байты второго спрайта и т. д.

При записи имени файла спрайтов можно использовать только 6 символов, после чего надо добавить расширение «.BIN». Это дает возможность считать файл через команду BLOAD.

Пример:

запись: ИМЯ.BIN

чтение: BLOAD «ИМЯ», &O30174

Общий вид загрузки готовой программы:

загрузка основной программы: CLOAD «ИМЯ 1»

загрузка подпрограммы вывода: BLOAD «ИМЯ 2», &O30000

загрузка файла спрайтов: BLOAD «ИМЯ3», &O30174

запуск: RUN

Е. УМНИКОВ

Строки вместо матриц

Про цвет на экране БК подробно написано в «ИНФО» № 2 за 1989 г. Задавать матрицы псевдоцветов, как предлагает С. Зильберштейн, заманчиво и просто; сколько цветов — столько и матриц... Но такой способ не очень экономичен, особенно если вспомнить про куцее ОЗУ. Лучше задавать не всю матрицу, а одну строку. Чередуя строки, получим различные комбинации. Для иллюстрации этой идеи привожу программу КАЛЕЙДОСКОП. Она рисует многоцветные калейдоскопические узоры, но, в отличие от известной программы В. Куманина ЦМУ, цветов на экране в несколько раз больше.

```
10  'Программа КАЛЕЙДОСКОП
20 DEF FNA(A,B)=INT(RND(1)*A)+B
30 ? CHR$(148)CHR$(158)CHR$(140)
                       CHR$(140)CHR$(146)
40 DATA &H1111,&H2222,&H3333,&H5555,
        &H6666,&H7777,&HAAAA,&HBBBB,
        &HFFFF,0 'маски псевдоцветов
50 CR=FNA(10,1)
60 RESTORE
70 FOR I=1 TO CR          'выбор цвета
80 READ C%
90 NEXT
100 NY=64*FNA(200,1)      'выбор
110 NX=2*FNA(28,1)        'размера
120 LY=64*FNA(53,2)       'прямоугольника
130 LX=2*FNA(5,1)
140 AN=16384+NY+NX        'вычисление
150 AK=32704-NY+NX        'адресов
160 AM=AN-2*NX+62         'ОЗУ
170 AL=AK-2*NX+62           экрана
180 FOR I=0 TO LY STEP 128
190 FOR J=0 TO LX STEP 2
200 POKE AN+I+J,C%        'запись
210 POKE AM+I-J,C%        'маски
220 POKE AK-I+J,C%        'в экранную
230 POKE AL-I-J,C%        'область
240 NEXT J,I              'через строку
250 GOTO 50
```

Как видно из текста программы, десять масок в случайной последовательности записываются в ОЗУ экрана (адреса от &O40000 до &O77776). Каждая маска, заданная в виде шестнадцатеричной константы, заполняет прямоугольную область со случайно выбранными размерами. Запись делается через строку. Кроме того, строится еще три одинаковых прямоугольника, чтобы получить центральную симметрию, т. е. калейдоскопический узор. Постепенное наложение прямоугольников даст различные сочетания масок (вспомните пропущенные строки и посчитайте, сколько разных комбинаций возможно при десяти масках). Визуально многие псевдоцвета почти не отличаются, поэтому реально на экране отобразится «всего лишь» несколько десятков оттенков. Небольшая модификация программы позволит рисовать не только прямоугольники, но и треугольники, диагональные полосы и другие фигуры.

Что же такое маска? И почему масок только десять? Как они выбраны?

Действительно, масок может быть намного больше, и задавать их можно по-разному. Давайте вспомним, как в БК организовано видео-ОЗУ. В цветном режиме экран представляет собой матрицу 256×256 точек. Каждой точке соответствуют два бита ОЗУ. Байт «отвечает» за четыре точки, слово — за восемь. В два бита можно поместить только четыре двоичных числа. Отсюда и четыре цвета. Их маски: 00 — черный, 01 — синий, 10 — зеленый, 11 — красный. Маской, точнее — трафаретом, называется образец, по которому заполняются биты ОЗУ. Маски задают для битов, байтов или слов. Процессор БК имеет 16-разрядную шину, и оператор POKE обращается к машинному слову, т. е. только к четным адресам. Теперь ясно, что запись двоичного числа (маски) &B1111111111111111 по адресу &O40000 отобразится в виде красной горизонтальной полоски длиной в восемь точек в левом верхнем углу экрана. В строке размещается 32 слова или 64 байта, поэтому адреса позиций в соседних по вертикали телевизионных строках отличаются на 64 (в операторах программы 100, 120 использован именно этот множитель). Двоичные маски для слова наглядны, но слишком длинны. Можно перевести их в десятичные числа; станет коротко, но непонятно. Удобнее всего использовать шестнадцатиричные константы. Что означает, к примеру, маска &H7777? Каждая цифра соответствует четырем битам, или двум точкам. Вспомнив двоичное представление семерки (7=0111), легко видеть, что эта маска означает чередование синих и красных точек. Все остальные маски выбраны либо как одинаковые точки (4 цвета), либо как последовательное чередование двух цветов (например, синий — зеленый, черный — красный и т. д.). Всего сочетаний по два из четырех возможно шесть, отсюда — десять масок. Остается объяснить, что делает оператор в строке 30. Он очищает служебную строку и экран, а также устанавливает видео-ОЗУ в начальное положение (т. е. сбрасывает рулонный сдвиг). Как получается симметричный узор, вы, очевидно, уже поняли (см. операторы 140—170).

С. КОМАРОВ

Хитрости Бейсика БК

Недостатки компилятора Бейсика для БК-0010.01 хорошо известны. Наибольшее их число связано с операциями с символьными переменными и символьными функциями. Вот характерный пример.

```
10 A$="ДА"
20 B$="МА"
30 C$=A$+B$
40 PRINT C$
50 GOTO 10
```

Компьютер при такой программе должен выдавать слово ДАМА, пока не нажата клавиша STOP, но, выведя в первый раз слово ДАМА, БК будет затем упорно выдавать слово МАМА.

Чтобы «обойти» эту особенность интерпретатора, можно определить операцию сложения символьных переменных через функцию пользователя:

```
8 DEF FNA$(A$,B$)=A$+B$
...
40 C$=FNA$(A$,B$)
```

Теперь программа будет работать нормально. Подобный прием помогает почти всегда, хотя в практике автора был случай, когда пришлось определить через функцию пользователя даже операцию присвоения значения символьной переменной:

```
10 DEF FNE$(A$)=A$
...
100 C$=FNE$(B$)
```

Обратимся теперь к функции STR$, которая ошибочно добавляет к соответствующему символьному представлению числа по пробелу слева и справа, вследствие чего перестают верно работать функции LEN, VAL. Рецепт здесь такой же — определить функцию пользователя

```
10 DEF FNSR$(N)=MID$(STR$(N),2,
                     LEN(STR$(N))-2)
```

которую и использовать в дальнейшем вместо функции STR$. Возможно, это не лучший выход, но работать можно.

Несколько слов о процедуре ввода-вывода числовых данных по каналу работы с магнитофоном. Если магнитофон не снабжен дистанционным управлением или же этот канал используется для обмена данными между двумя БК, то возникает неприятная проблема: если файл, открываемый оператором OPEN «имя» FOR INPUT/OUTPUT, состоит

из нескольких блоков (один блок равен 256 байтам), то типичной является ситуация, когда временной интервал между блоками при выводе оказывается меньше, чем это необходимо при вводе. В результате при попытке считать файл данных с магнитофона компьютер, приняв первый блок, «не успевает» к началу второго и зависает. В подобной ситуации для повторного запуска после команды STOP обязательно надо выполнить команду SAVE и только потом RUN. Для предотвращения же такой ситуации следует увеличить интервал между блоками при выводе. Сделать это можно следующим образом (выводится массив X(N)):

```
100 OPEN "DATA" FOR OUTPUT
110 FOR I=0 TO N
120 A=1.1^10%'ЗАДЕРЖКА
130 PRINT# X(I)
140 NEXT
150 CLOSE
```


Оператор в строке 120 обеспечивает необходимое увеличение интервала между блоками, конкретное же значение показателя степени в нем зависит от ряда факторов, в частности при выводе целых чисе оно обычно может быть уменьшено до 2—4. При отладке программы всегда можно подобрать значение показателя степени таким, чтобы интервалы между блоками при выводе и вводе были примерно равны. Однако если последний блок файла при выводе оказывается существенно меньше 256 байтов, то временной интервал между предпоследним и последним блоками заметно сокращается, и при вводе компьютер опять может зависнуть. Выход — добавить при выводе еще один, лишний блок. Для этого в приведенный выше фрагмент программы между строками 140 и 150 следует включить, например, следующие строки:

```
142 A%=1%
144 FOR I=0 TO 62
146 PRINT# A%
148 NEXT
```

Разумеется, при вводе все дополнительно введенные операторы не нужны, поэтому соответствующий фрагмент программы должен выглядеть так:

```
100 OPEN "DATA" FOR INPUT
110 FOR I=0 TO N
120 INPUT# X(I)
130 NEXT
140 CLOSE
```

Использование этого приема позволяет обеспечить надежный обмен данными с магнитофоном или с другим компьютером БК.

А. БОЧАРОВ

Вмешательство в работу программы

Часто, особенно в играх, нужно, чтобы программа получала информацию с клавиатуры, не останавливаясь. Например, в игре типа «Клад» необходимо управлять действиями человека, в то же время должны происходить полет пули, зарастание стенки или бег хранителей кладов.

Мне известно три способа реализации этого на БК-0010.

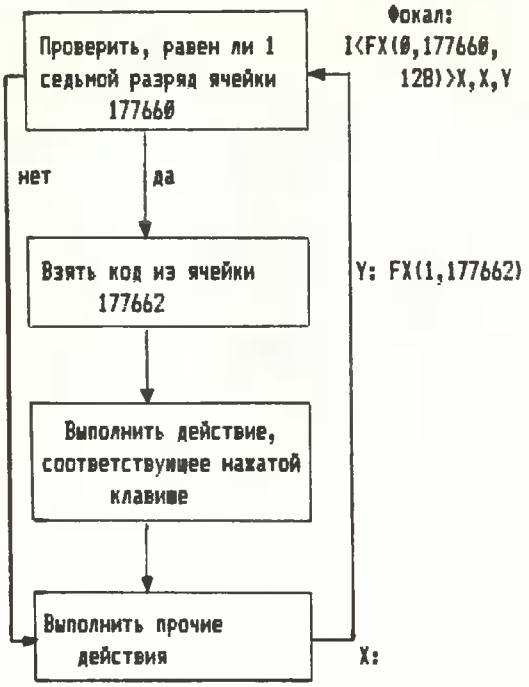
Первый способ. Позволяет после нажатия клавиши выполнять кодируемое ею действие, пока не будет нажата другая клавиша (рис. 1).

Если этот способ используется в игре типа «Клад», то «выполнить действие, кодируемое нажатой клавишей»,— передвинуть, например, человечка вверх по лестнице, если была нажата клавиша «↑»; «выполнить прочие действия» — нарисовать следующую стадию полета пути, передвинуть хранителей кладов, зарастить стену и т. п. Пример программы, использующей такой способ,— «LODE RUNNER» Маркова.

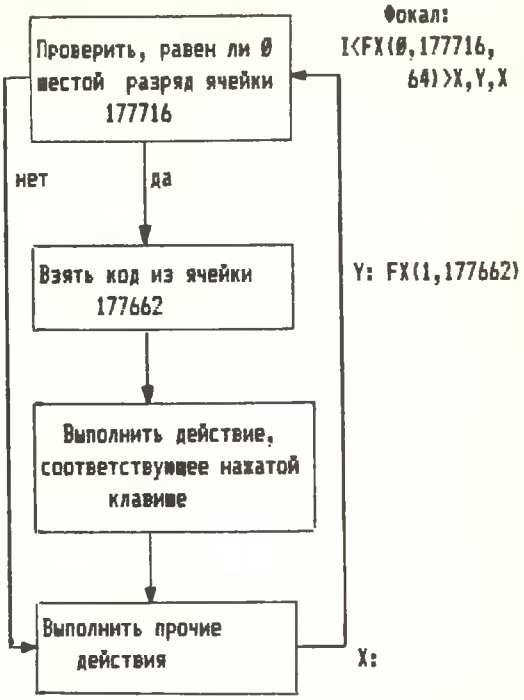
1

Второй способ. Позволяет делать действие, кодируемое клавишей, только один раз после ее нажатия. Для повторного исполнения этого действия необходимо снова нажать эту клавишу (рис. 2).

2



3



Если этот способ используется в игре типа «Клад», то, в отличие от первого способа, при нажатии, например, клавиши «←» человек не будет бежать влево, пока не будет нажата другая клавиша, а сдвинется влево лишь на одну позицию. Чтобы заставить его двигаться влево непрерывно, надо все время нажимать—отпускать данную клавишу. В играх типа «Клад» это, конечно, не лучший вариант, но в других случаях может оказаться удобным.

Такой способ используется в программе «Диверсант» Кумандина.

Третий способ. Позволяет выполнять действие, пока нажата кодирующая его клавиша (рис. 3).

Он используется во многих динамических играх: «Клад», «LAND» и др. По такому же принципу построен механизм действия клавиши «повтор».

В. БУЛИТКО
VIII класс
г. Одесса

Советы и наблюдения

Запятая либо точка с запятой необходимы при выводе вещественных чисел, не содержащих признака #. Не обойтись без точки с запятой и в самом конце списка выводимых на печать выражений, когда нужно заблокировать перевод строки, а без запятой — когда выдача должна происходить зонами. В других же случаях разделителями могут служить признаки вида чисел #, !, %, ¤, префиксы-признаки системы счисления &B, &O, &H, пары скобок в функциях печати AT(...), ТАБ(...) и символьных функций CHR¤(...), MID¤(...), пары кавычек с пробелами или без между ними.

Все это позволяет в соответствующих ситуациях опускать точку с запятой в качестве разделителя и так экономить место в ОЗУ.

Пример.

```
10 D=111
20 E!=222!
30 C%=333%
40 T$="444"
50 ? D;E!C%T$CHR$(76%)&B101
60 ? D#TAB(3%) MID$(T$,2%) " "
                          HEX$(255%)
```

На монитор будет выдано:

```
111  222  333  444L  5
111 44 FF
```

В. ЯКОВЛЕВ

Хочу предложить исправления к программе А. Чистякова «Голос-2» (ИНФО. 1988. № 4).

1. По адресу 1124 необходима не команда ROL R4 (код 6104), а команда ROR R4 (код 6004).

2. Дополнительно вместо 40 по адресу 1134 предлагаю поставить 140, чтобы при воспроизведении работал динамик компьютера.

После этого голос БК неузнаваемо изменился: серьезные искажения исчезли, стало просто различать все слова не только при скоростях 1—6. В качестве указателя скорости можно нажимать даже буквы и специальные знаки. В этом случае длительность записываемой речи может достигать минуты и более.

Для устранения шума между словами можно обнулить эти участки памяти хотя бы с помощью директив отладки.

А. ЖАРИКОВ

При перенумерации программ в тексте могут оставаться пробелы; например, GOTO 1000, если 1000 превратится в 1, будет выглядеть так: GOTO 1. Если памяти «в обрез», пробелы нужно убрать.

Я заметил в интерпретаторе ошибку. Проиллюстрировать ее можно следующим примером (далее ответы интерпретатора выделены крупным шрифтом).

```
A$="1"
Ok
A$=A$+"2"
Ok
? A$
12
Ok
A$=A$+"3"
Ok
? A$
223
Ok
```

А правильный ответ — 123.

Консультат редакции И. Г. Панченков советует: чтобы избавиться от этого эффекта, следует каждое присваивание начинать с пустой строки: A$=«1»+A$+«2»

Если в ключ записать слово, начинающееся с русских букв, то в верхней строке экрана

(в подсказке) отобразится первая латинская буква (или цифра), встретившаяся в тексте ключа.

Кстати, чтобы обновить строку с первыми буквами ключей, надо одновременно нажать АР2+СУ+Ч, затем клавишу ввода. Если эта клавиша не будет нажата, то, когда вы дойдете до середины строки, набирая текст, БК зависнет.

И. КОНИВЕЦ

Длительное прижатие магнитной ленты к тонвалу при выключенном двигателе магнитофона может вызвать ее порчу в месте прижатия и даже отслоение магнитного слоя от основы (особенно склонны к этому импортные МЛ).

Ю. А. ЗАЛЬЦМАН

Часть владельцев БК испытывают затруднения при вводе-выводе на МЛ программ, написанных в кодах БК. Приводим подробные описания этих операций. Обычным шрифтом набраны команды пользователя, жирным — ответы БК, мелким — комментарии для читателей.

Считывание, первый способ.

```
<ЛАТ>Р М<ВВОД>
  ?
М<ВВОД>
  ИМЯ?
<название программы><ВВОД>
  /названная программа
     загружается/
  ?
 /загрузка уже произошла/
S<ВВОД> /или S1000<ВВОД>/
```

Считывание, второй способ.

```
<ЛАТ>Р Т<ВВОД>
  +
```

```
<РУС><ТС><ВВОД>
  ¤
МЧ<ВВОД>
  Нажмите клавишу магнитофона
  "ПУСК". Адрес=
0<ВВОД>
  Имя=
<название программы><ВВОД>
 /названная программа
    загружается/
  ¤
 /загрузка уже произошла/
1000<ЛАТ>Б<ВВОД>
 /БК выполняет заданную
     программу/
```

Для правильного проведения записи нужно знать начальный адрес загрузки программы и ее длину. Эти параметры записаны в ячейках 346 и 350, узнать их можно командой

```
¤<РУС>346А4Л<ВВОД>
```

Процедура записи.

```
<ЛАТ>Р Т<ВВОД>
  +
<РУС><ТС><ВВОД>
  ¤
МЗ<ВВОД>
  Нажмите клавиши магнитофона
  "ПУСК" и "ЗАПИСЬ"
  Адрес=
<адрес><ВВОД>
  Длина=
<длина><ВВОД>
  Имя=
<название программы><ВВОД>
```

Хотел бы переписываться с поклонниками Бейсика-MSX, XFOCALa и Мини-Фонда. Имею небольшую библиотечку системных и игровых программ.

А. МУМИНОВ.
703007, Самарканд, ул. Октябрьская, 53, кв. 27.

Поправка

В предыдущем номере «ИНФО» в статье «Отсчет времени на БК-0010.01» допущена опечатка. На с. 68, правая колонка, в программе ЧАСЫ пропущена строка 45:
45 IF T%>0 % TH 30 EL T=TO—T%/365.47+.5

ЧТО МОЖЕТ ЭВМ?

Когда глаз начинает «нервничать»

Глаукома является второй по частоте причиной слепоты в США, несмотря на то что при своевременном начале лечения эта болезнь не так страшна. Обнаружить болезнь на самой ранней стадии позволяет новая компьютерная диагностическая система, разработанная в одной из высших медицинских школ США. В ее основе лежит свойство поврежденных глазных нервов изменять внутреннее давление глаза.

С помощью специальной видеокамеры изображение глаза пациента передается в компьютер, в котором поступившая информация перерабатывается таким образом, чтобы врач-офтальмолог мог увидеть структурные изменения глазного дна и состояние нервных окончаний глаза, восстановленное по распределению глазного давления.

Помимо предупреждения глаукомы система позволяет диагностировать и некоторые другие заболевания. Профессор офтальмологии доктор Джозеф Каприоли считает, что компьютерная диагностика позволит избавить многих его потенциальных пациентов от необходимости прибегнуть к его помощи.

Сколько нас?

Среди американских ученых наибольшую долю (23 %) составляют специалисты по электронно-вычислительной технике. Их 445 тысяч. На втором месте — биологи (20,2 %), на третьем — представители социальных наук (18,8 %).

Временные санитарно-гигиенические нормы и правила

устройства, оборудования, содержания и режима работы на персональных электронно-вычислительных машинах и видеодисплейных терминалах в кабинетах вычислительной техники и дисплейных классах всех типов средних учебных заведений (№ 5146—89*)

1. Общие положения

1.1. Требования настоящих «Временных санитарно-гигиенических норм и правил устройства, оборудования, содержания и режима работы на персональных электронно-вычислительных машинах и видеодисплейных терминалах в кабинетах вычислительной техники и дисплейных классах всех типов средних учебных заведений»** распространяются на все виды персональных электронно-вычислительных машин (ПЭВМ) и видеодисплейных терминалов (ВДТ) на электронно-лучевых трубках, предназначенных для использования в системе Госкомитета СССР по народному образованию, а также других министерств и ведомств, где ПЭВМ и ВДТ предназначены для работы и обучения учащихся.

Требования Временных СанПин распространяются также на проектируемые, реконструируемые и существующие здания и помещения всех типов средних учебных заведений, предназначенных для обучения, практики и работы с ПЭВМ и ВДТ.

1.2. Требования Временных СанПин являются обязательными для выполнения руководителями предприятий и организаций, проектирующих, изготовляющих и эксплуатирующих средства вычислительной техники на электронно-лучевых трубках, а также здания и помещения, предназначенные для обучения, практики и работы при использовании ПЭВМ и ВДТ учащимися средних и специальных учебных заведений независимо от их ведомственной принадлежности.

1.3. Действие Временных СанПин устанавливаются с 01.01.90 года до утверждения и вступления в действие «Единых санитарных норм и правил для пользователей средств информационно-вычислительной техники».

1.4. С вводом в действие настоящих Временных СанПин отменяются:

«Временные санитарно-гигиенические нормы и правила к устройству видеотерминалов школьных ЭВМ», часть 1-«Дисплеи», утвержденные Минздравом СССР и Минпросом СССР 01.12.86;

«Режимы обучения учащихся за видеотерминалами ЭВМ (Временные рекомендации)», утвержденные Минпросом СССР 08.02.87 и согласованные с Минздравом СССР 29.01.87;

временные методические рекомендации «Гигиенические условия организации учебных занятий с применением компьютеров в средней общеобразовательной школе» № 4217—86, утвержденные Минздравом СССР 11.11.86.

1.5. Контроль за выполнением Временных СанПин возлагается на органы и учреждения санитарно-эпидемиологической службы.

2. Требования к помещениям кабинетов вычислительной техники, дисплейным классам, размещению рабочих мест и оборудованию

2.1. Объемно-планировочные решения зданий и помещений, предназначенных для работы со средствами отображения информации на электронно-лучевых трубках (ЭЛТ), используемых в ПЭВМ и ВДТ, должны соответствовать требованиям настоящих Временных СанПин.

2.2. Кабинет вычислительной техники (КВТ) и дисплейные классы (ДК) должны иметь естественное освещение. Ориентация окон должна быть на север, северо-восток.

2.3. КВТ и ДК не должны располагаться в подвалах и цокольных этажах.

2.4. КВТ и ДК должны быть изолированы от помещений, имеющих повышенные уровни шума (машинные залы, гимнастический зал, мастерские и др.)

2.5. Площадь КВТ и ДК должна приниматься из расчета не менее 6 кв. м на одно рабочее место, высота помещения не менее 4 м.

2.5.1. При КВТ и ДК, в которых наличие ПЭВМ и ВДТ не менее 10, должна быть лаборантская площадью не менее 18 кв. м; при организации лаборантской следует предусмо-

---

*Утверждены Министерством здравоохранения СССР и Государственным комитетом СССР по народному образованию.

** В дальнейшем по тексту — Временные СанПин.

треть наличие радиомонтажного стола с местным нижним или боковым отсосом воздуха, рабочего стола, шкафов и стеллажей для хранения инструментов, оборудования и комплектующих изделий (Приложение 1, а).

2.6. Поверхность пола должна быть ровная, без выбоин, нескользкая и удобная для очистки, обладающая антистатическими свойствами.

2.7. Запрещается применять для отделки интерьера КВТ и ДК строительные материалы, выделяющие из своего состава вредные химические вещества и соединения в воздух.

2.8. К столам с ПЭВМ и ВДТ подводится электропитание и кабель локальной сети, с обеспечением электробезопасности.

2.9. Стол преподавателя с ПЭВМ или ВДТ с тумбами-приставками для графопроектора и принтера устанавливается на подиуме.

2.10. Длина (слева направо) одноместного рабочего стола* для учащегося должна быть не менее 70 см, ширина должна обеспечивать место перед клавиатурой 30 см для расположения тетради и опоры предплечий рук для снятия статического напряжения с мышц плечевого пояса. Поверхность стола для установки монитора должна быть горизонтальной, а поверхность, на которой находится клавиатура,— наклонной (угол наклона 12—15°).

Высота края стола, обращенного к работающему за видеомонитором, и стула над полом должна приниматься в соответствии с ростом учащегося (2.10)

Таблица 2.10

Рост учащегося (см)	Высота над полом в мм		
	стол не менее	Пространство для ног не менее	стул не менее
145—160	640	530	380
161—175	700	590	420
>175	760	650	460

Ширина пространства для ног под столом должна быть не менее 500 мм, глубина — не менее 450 мм.

При наличии высокого стола, не соответствующего росту учащегося, необходимо пользоваться регулируемой по высоте подставкой для ног.

Конструкция подставки: ширина 300 мм, длина — 400 мм, угол наклона опорной поверхности до 20°. Подставка должна иметь регулировку по высоте в пределах до 150 мм. Поверхность подставки должна быть рифленой, по переднему краю — бортик высотой 10 мм.

2.11. КВТ и ДК должны быть снабжены стульями с меняющимися по высоте полумягким сиденьем и профилированной спинкой, а также позволяющими осуществлять поворот сиденья и спинки стула в пределах ±180°.

2.12. Расстановка рабочих мест с ПЭВМ и ВДТ может быть трех вариантов: периметральная, рядами (1—3-рядная), центральная.

Схемы планировки КВТ или ДК с экспликацией представлены в Приложении 1.

Оптимальным вариантом расстановки рабочих мест с ПЭВМ и ВДТ является периметральная (Приложение 1, а).

2.12.1. При периметральной расстановке рабочих мест устанавливаются следующие расстояния:

а) по ширине КВТ и ДК:

расстояние между стеной с оконными проемами и столами должно быть не менее 0,8 м;

расстояние между стеной, противоположной оконным проемам, и столами с ПЭВМ и ВДТ должно быть не менее 0,1 м, в ряде случаев в зависимости от используемых видеотерминалов столы могут быть установлены непосредственно у стены;

б) по длине КВТ и ДК:

столы двухместные с ПЭВМ и ВДТ должны быть расставлены без разрыва (Приложение 1, а). Одноместные столы должны быть расставлены с разрывом, при этом расстояние между столами определяется площадью КВТ или ДК и количеством рабочих мест учащихся с видеотерминалами.

При небольшом количестве рабочих мест с ПЭВМ или ВДТ предпочтение следует отдавать расстановке их у стены, противоположной оконным проемам.

При периметральной расстановке столов с ПЭВМ или ВДТ КВТ или ДК должен быть дополнительно оборудован ученическими столами (ГОСТ 11015—86) со стульями (ГОСТ 11016—86) для работы без ПЭВМ или ВДТ из расчета количества занимающихся. В зависимости от количества рабочих мест с ПЭВМ или ВДТ ученические столы могут расставляться в один или два ряда. Соединять двухместные ученические столы не разрешается (расстояние между рядами не менее 0,5 м).

2.12.2. При расположении столов с ПЭВМ или ВДТ рядами каждый стол должен иметь защитный экран со стороны, к которой обращена тыльная часть видеомонитора.

При одно-, двух- и трехрядной расстановке одноместных столов с ПЭВМ или ВДТ

*Допускается КВТ и ДК оборудовать двухместными столами (Приложение 1).

необходимо соблюдать следующие расстояния:

а) по длине КВТ и ДК:

расстояние в каждом ряду между рабочими столами должно быть не менее 1,0 м;

расстояние между последними рабочими столами и стеной — не менее 0,8 м;

при расположении двери в КВТ или ДК у последних столов расстояние между ними и торцевой стеной должно быть не менее 1,2 м;

б) по ширине КВТ и ДК:

расстояние между стеной с оконными проемами и первым рядом столов с ПЭВМ или ВДТ должно быть от 0,8 до 1,1 м;

расстояние между третьим (от оконных проемов) рядом столов и стеной — 0,8—0,9 м.

При двухрядной расстановке рабочих столов с ПЭВМ или ВДТ проход между рядами соответственно увеличивается.

2.12.3. При центральной расстановке столов с ПЭВМ или ВДТ, когда два ряда столов стоят без разрыва в центре КВТ или ДК и экраны видеомониторов обращены в противоположные стороны. ПЭВМ или ВДТ могут располагаться в шахматном порядке или напротив друг друга тыльными сторонами мониторов.

2.12.4. При входе в КВТ и ДК необходимо предусмотреть шкафы для портфелей и сумок учащихся.

3. Требования к правильной посадке учащихся при работе на ПЭВМ или ВДТ

3.1. При работе на ПЭВМ или ВДТ учащийся должен соблюдать правильную посадку. В соответствии с ростом должен быть подобран стол и стул. Спина в области нижних углов лопаток должна иметь опору. Край сиденья стула должен заходить на 3—5 см за край стола, обращенного к учащемуся. Голова слегка наклонена вперед (5—7°). Предплечья должны опираться на поверхность стола (перед клавиатурой) для снятия статического мышечного напряжения с плечевого пояса и рук. Угол, образуемый предплечьем и плечом, а также голенью и бедром, должен быть не менее 90°.

3.2. Уровень глаз при вертикальном расположении экрана должен приходиться на центр экрана или 2/3 его высоты. Линия взора должна быть перпендикулярна центру экрана, и оптимальное ее отклонение в вертикальной плоскости должно находиться в пределах $\pm 5°$, допустимо — $\pm 10°$. Оптимальный обзор в горизонтальной плоскости от центральной оси экрана должен быть в пределах $\pm 15°$, допустимый — $\pm 30°$. При рассматривании информации, находящейся в крайних положениях экрана монитора, угол рассматривания (угол, ограниченный линией взора и поверхностью экрана) должен быть не менее 45°. Угол рассматривания следует соблюдать при необходимости работы за ПЭВМ и ВДТ вдвоем. Чем больше угол рассматривания, тем легче воспринимать информацию с экрана видеомонитора и меньше будут уставать глаза.

3.3. Угол рассматривания символов на экране монитора должен быть не менее 20 угловых минут*.

3.4. Оптимальное расстояние глаз учащихся до экрана ПЭВМ или ВДТ должно быть в пределах 0,6—0,7 м, допустимое — не менее 0,5 м.

4. Требования к условиям работы на ПЭВМ или ВДТ

4.1. Требования к микроклимату.

4.1.1. КВТ и ДК оборудуются системой отопления, приточно-вытяжной вентиляцией или кондиционированием воздуха в соответствии с требованиями главы СНиП «Отопление, вентиляция и кондиционирование воздуха. Нормы проектирования» и требованиями настоящих Временных СанПин.

Подача воздуха должна проводиться в верхнюю зону малыми скоростями из расчета создания подвижности воздуха на рабочем месте учащихся до 0,1 м/с (лучше через подшивной перфорированный потолок).

4.1.2. При организации КВТ и ДК в приспособленных помещениях и отсутствии приточно-вытяжной вентиляции необходимо организовывать кондиционирование воздуха. Рекомендуются бытовые кондиционеры типа БК-1500, БК-2500, БК-2000Р. Расчет необходимого количества кондиционеров должен проводиться по теплоизбыткам от машин, людей, солнечной радиации и от источников искусственного освещения.

Режим работы кондиционеров должен обеспечивать температурно-влажностный режим в соответствии с п. 4.1.4.

4.1.3. В КВТ и ДК перед началом уроков и после каждого академического часа занятий следует осуществлять сквозное проветривание для улучшения качественного состава воздуха, включая аэроионный режим.

*Угол рассматривания можно вычислить по формуле:

$$\operatorname{tg}(\alpha/2) = S/(2L),$$

где S — высота буквы и цифры (ММ);

L — расстояние глаз до объекта информации на экране, (мм);

α — угол рассматривания (угловые мин).

Если позволяют погодные условия, занятия должны проводиться при открытых окнах (фрамуги, створки окон).

4.1.4. Оптимальными параметрами температуры при почти неподвижном воздухе являются 19—21 °С, допустимыми 18—22 °С, соответственно относительная влажность воздуха 62—55 и 39—31 %. Скорость движения воздуха должна быть не более 0,1 м/с для всех климатических зон (табл. 4.1).

Таблица 4.1

Оптимальные и допустимые параметры температуры и относительной влажности воздуха

Температура воздуха (градусы С)	Влажность воздуха			
	абсолютная (г/м³)		относительная (%)	
	оптимальная	допустимая	оптимальная	допустимая
18	10	6	65	39
19	10	6	62	37
20	10	6	58	35
21	10	6	55	33
22	10	6	52	31

4.1.5. В КВТ или ДК должна ежедневно проводиться влажная уборка. Для повышения влажности воздуха следует использовать увлажнители.

4.2. Требования к освещению.

4.2.1. Помещение КВТ или ДК должны иметь естественное и искусственное освещение, отвечающее требованиям главы СНиП 11—4—79 «Естественное и искусственное освещение. Нормы проектирования» санитарного законодательства и настоящих Временных СанПин.

4.2.2. Ориентация КВТ на С или С — В (см. п. 2.2.). При ориентации этих помещений на другие стороны горизонта должны быть предусмотрены регулярные солнцезащитные устройства типа жалюзи, расположенные снаружи или в межстекольном пространстве.

Основной поток естественного света должен быть слева. Не допускается направление основного светового потока естественного света справа, сзади и спереди работающего на ЭВМ. Солнечные лучи и блики не должны попадать в поле зрения учащихся и на экраны видеомониторов.

4.2.3. Коэффициент естественного освещения (КЕО) в климатических зонах с устойчивым снежным покровом должен быть не ниже 1,2 % и не менее 1,5 % — на остальной территории страны.

4.2.4. Соотношение яркостей поверхностей, находящихся в поле зрения учащихся, должно быть в рабочей зоне в пределах 3:1; между поверхностью экран — стол и дальним окружением: стены, классная доска, пол, потолок — 10:1.

Коэффициент отражения рабочих поверхностей и цветовой отделки интерьера КВТ и ДК должен быть следующим: потолка — 0,7—0,8; стен — 0,5—0,6; пола — 0,3—0,4; стола — 0,45—0,5; клавиатуры — 0,4—0,6.

Следует окрашивать стены в КВТ или ДК до потолка или на высоту панелей холодными тонами красок: светло-голубым, светло-зеленым, светло-серым цветами. Допускается окраска стен светло-желтым, светло-бежевым цветом или цветом слоновой кости.

Нельзя окрашивать стены, расположенные напротив экрана монитора, темными тонами красок (коэффициент отражения 0,3—0,4).

Поверхности рабочих столов с ПЭВМ и ВДТ, ученических столов должны быть цвета натуральной древесины, голубого, светло-зеленого, светло-серого цвета.

Поверхности ограждающих конструкций дисплейного класса, классной доски, рабочих столов должны быть матовыми.

Оконные переплеты рам, подоконники следует окрашивать белой краской.

На окнах монтируются занавеси, по цвету гармонирующие с окраской стен. Занавеси не должны пропускать естественный свет и полностью закрывать оконные проемы. Ширина занавеси должна быть в два раза больше ширины окна.

Запрещается применять для окон черные занавеси!

4.2.5. В осветительных установках (ОУ) КВТ и ДК следует применять систему общего освещения, выполненную потолочными или подвесными люминесцентными светильниками, равномерно размещенными по потолку рядами, параллельно светопроемам. Экран монитора должен располагаться в зоне защитного угла светильника, и его проекция должна быть вне экрана монитора.

Светильники не должны отражаться на экране ПЭВМ или ВДТ.

Выбор светильников должен проводиться с учетом ограничения прямой и отраженной блескости.

Люминесцентные светильники должны включаться рядами.

Светильники должны освобождаться от пыли не менее двух раз в течение учебного года: перед началом учебного года и во время зимних каникул, а также по мере загрязнения.

4.2.6. При проектировании ОУ с люмине-

сцентными светильниками в КВТ или ДК следует выбирать коэффициент запаса, равный 1,4, согласно постановлению Госстроя СССР «Об изменении главы СНиП II—4—79 «Естественное и искусственное освещение. Нормы проектирования» № 205 от 04.12.85 г.

4.2.7. Допустимая величина дискомфорта, одного из основных качественных параметров ОУ, регламентируемого для ограничения прямой блескости, не должна превышать 15. При проектировании ОУ следует пользоваться инженерным методом оценки слепящего действия ОУ по дискомфорту (Пособие по проектированию естественного, искусственного и совмещенного освещения к СНиП II—4—79. М., 1985 г.).

4.2.8. Коэффициент пульсации не должен превышать 10 % (для чего следует применять многоламповые светильники с компенсирующими ПРА, осуществлять расфазировку светильников при электромонтаже ОУ).

Для освещения КВТ или ДК рекомендуется применять светильники серий ЛП013, ЛП031, ЛП033, исполнение 001 и 006, ЛС002, ЛС004 с металлической экранирующей решеткой и непрозрачными боковинами. Для освещения классной доски — светильники серии ЛПО-125.

4.2.9. В качестве источников света рекомендуется использовать люминесцентные лампы мощностью 40 Вт или энергоэкономичные мощностью 36 Вт типа ЛБ, ЛХБ как наиболее эффективные и приемлемые с точки зрения спектрального состава, цветовая температура (Тцв.) излучения которых находится в диапазоне 3500—4200 °К.

4.2.10. Не разрешается в КВТ и ДК применять люминесцентные лампы типа ЛД и ЛДЦ (Тцв.=6000—65000 °К)!

4.2.11. Перегоревшие лампы должны своевременно заменяться новыми, по мере выхода из строя.

4.2.12. В КВТ и ДК освещенность на рабочих поверхностях должна приниматься по табл. 4.2.

4.2.13. Работа на видеотерминалах во время учебных занятий и при производственной практике должна осуществляться при следующих видах освещения:

общем люминесцентном освещении и зашторенных окнах, когда мониторы располагаются по периметру помещения или при центральном расположении рабочих мест в два ряда по длине дисплейного класса с экранами, обращенными в противоположные стороны, а также при одно-трехрядной расстановке рабочих мест с ПЭВМ или ВДТ, когда на улице темно;

совмещенном освещении (естественное+ +искусственное) только при одно-, двух- и трехрядном расположении рабочих мест (экран и поверхность рабочего стола находятся перпендикулярно светонесущей стене);

естественном освещении при расположении рабочих мест с ПЭВМ или ВДТ в один ряд по длине КВТ и ДК на расстоянии 0,8—1,0 м от стены с оконными проемами и экраны видеотерминалов находятся перпендикулярно этой стене.

Таблица 4.2

Нормируемые уровни освещенности КВТ и ДК

Характер работы	Рабочая поверхность	Плоскость	Освещенность (лк, не менее)
Работа за ПЭВМ или ВДТ (учебные занятия, практика)	Экран	В	200—300
	Клавиатура	Г	400
	Стол	Г	400
	Классная доска	В	500

Примечание: В — вертикальная плоскость, Г — горизонтальная плоскость.

4.3. Требования к защите от электромагнитных излучений и электростатических полей.

4.3.1. ПЭВМ и ВДТ на электронно-лучевых трубках являются источниками широкополосных электромагнитных излучений:

мягкого рентгеновского;

ультрафиолетового (УФИ) — 200—400 мм;

видимого — 400—700 мм;

ближнего инфракрасного — 700—1050 мм;

радиочастотного диапазона — 3 кГц — 30 МГц;

сверх- и инфранизкочастотного — 0—3 кГц;

электростатических полей (ЭСП).

4.3.2. Мощность дозы рентгеновского излучения в любой точке пространства на расстоянии 5 см от поверхности экрана ПЭВМ или ВДТ не должна превышать $7{,}74\times10^{-13}$ А/кг (0,003 мкР/с, что соответствует 10,8 мк/Р/час) согласно «Санитарным правилам работы с источниками неиспользуемого рентгеновского излучения» № 1960—79, утвержденным Минздравом СССР 19.01. 1979 г.

4.3.3. Ультрафиолетовое излучение в диапазоне 200—315 мм не должно превышать 10 мкВт/м^2, а в диапазоне 315—400 мм и видимом диапазоне излучений от 400 до 700 мм — 0,1 Вт/м^2; в ближнем инфракрасном излучении (700—1050 мм) — 0,05 Вт/см^2, а в дальнем инфракрасном излучении (1050 мм — 1 мм) — 4 Вт/м^2.

4.3.4. Уровни напряженности электростатических полей (ЭСП) не должны превышать 15 кВ/м.

При превышении ЭСП следует руководствоваться ГОСТ. ССВТ. 12.4.124—83 «Средства защиты от статического электричества. Общие технические требования», мероприятиями по повышению влажности воздуха в настоящих Временных СанПин (п. 4.1.3; 4.1.5), а также каждый день освобождать от пыли экран и поверхность монитора.

4.3.5. Содержание легких аэроионов должно соответствовать «Санитарно-гигиеническим нормам допустимых уровней ионизации воздуха производственных и общественных помещений» № 2152—80 (табл. 4.3.).

Таблица 4.3

Допустимые уровни ионизации воздуха

Уровни	Число ионов в 1 $см^3$ воздуха	
	$n+$	$n-$
Минимально необходимые	400	600
Оптимальные	1500—3000	3000—5000
Максимально допустимые	50 000	50 000

4.4. Требования к защите от шума и вибраций.

4.4.1. КВТ и ДК не должны граничить с помещениями, имеющими повышенные уровни шума (п. 2.4).

Уровень шума в КВТ и ДК при неработающей вычислительной технике не должен превышать 40 дБА, а при работе систем воздушного отопления, вентиляции и кондиционирования воздуха 35 дБА согласно «Санитарным нормам допустимого шума в помещениях жилых и общественных зданий и на территории жилой застройки» № 3077 от 03.08.84 г.

Шумящее оборудование, такое, как АЦПУ, печатающие устройства, создающие высокие уровни шума, превышающие допустимые уровни, должны находиться вне помещения КВТ и ДК (лаборантская, специально отведенное помещение).

4.4.2. Звукоизоляция ограждающих конструкций КВТ или ДК должна отвечать требованиям главы СНиП 11—12—77 «Защита от шума. Нормы проектирования» с дополнениями и изменениями.

Для снижения уровня шума потолок или стены выше панелей, а иногда и стены и потолок должны облицовываться звукопоглощающим материалом с максимальным коэффициентом звукопоглощения в области частот 63—8000 Гц. (Необходим расчет.)

Дополнительным звукопоглощением в КВТ или ДК служат занавеси на окнах, подвешенные в складку на расстоянии 15—20 см от ограждения, выполненные из плотной тяжелой ткани (см. п. 4.2.4.).

4.4.3. Время реверберации в КВТ или ДК должно быть не более 1 с. Частотная характеристика времени реверберации в диапазоне частот 250—4000 Гц должна быть ровной, а на частоте 125 Гц спад времени реверберации должен составлять не более 15 %.

4.4.4. Уровень шума на рабочих местах во время работы на ПЭВМ или ВДТ в КВТ и ДК не должен превышать 50 дБА (табл. 4.4.)

4.5. Требования к химическому составу воздуха в КВТ и ДК.

Вредные химические вещества не должны превышать предельно допустимых концентраций в соответствии с «Предельно допустимыми концентрациями (ПДК) загрязняющих веществ в атмосферном воздухе населенных мест» № 3086—84 от 27.08.84 г. и утвержденными дополнениями.

Примерный перечень вредных веществ:

аммиака	0,2 мг/$м^3$;
диоксида углерода .	0,1 об. %;
озона	0,03 мг/$м^3$;
фенола	0,003 мг/$м^3$;
формальдегида . .	0,01 мг/$м^3$;
хлористого винила	0,005 мг/$м^3$.

Таблица 4.4.

Уровни звука, эквивалентные уровни звука и уровни звукового давления в октавных полосах частот в КВТ или ДК при работе на ПЭВМ и ВДТ

На рабочем месте с ПЭВМ или ВДТ	Уровни звукового давления (дБ)									Уровни звука LA; эквивалентные уровни $L_{экв.}$ А (дБА)
	Среднегеометрические частоты октавных полос (Гц)									
	31,5	63	125	250	500	1000	2000	4000	8000	
При работе на ПЭВМ или ВДТ	86	71	61	54	49	45	42	40	38	50

Таблица 4.4.[a]

Нормируемые уровни вибрации в дБ в октавных полосах частот*

Уровни	Среднегеометрические частоты октавных полос, Гц					
	2	4	8	16	31,5	63
Виброскорости	79	73	67	67	67	67
Виброускорения	25	25	25	31	37	43
Вибросмещения	133	121	109	103	97	91

* «Санитарные нормы допустимых вибраций в жилых домах», № 1304—75 от 13.06.75.

5. Гигиенические требования к ПЭВМ и ВДТ на базе ЭЛТ

5.1. Конструкция видеомонитора должна предусматривать меры, обеспечивающие хорошую разборчивость изображения, независимую от внешней освещенности.

В зависимости от назначения и области применения видеотерминалы могут быть разделены на следующие группы.

Группа А — цветные мониторы только для демонстрационных целей в учебном процессе.

Группа Б — цветные мониторы для персональной работы пользователей в учебном процессе.

Группа В — монохромные мониторы для комплексов учебной вычислительной техники.

5.2. Величина контраста определяется отношением максимальной яркости экрана к минимальной:

$$K = L_{max}/L_{min},$$

где L — яркость.

При негативном контрасте L_{max} соответствует яркости знака.

При позитивном контрасте L_{max} соответствует яркости фона, L_{min} — яркости знака.

5.3. Параметры мониторов в зависимости от группы приведены в таблице 5.

5.4. Комплектование ПЭВМ и ВДТ, не отвечающих требованиям Временных СанПин (раздел 5), запрещается.

5.5. Мониторы должны иметь антибликовое покрытие (кроме группы А) с коэффициентом отражения не более 0,5. Покрытие должно также обеспечивать снятие электростатического заряда с поверхности экрана, исключать искрение и накопление пыли.

5.6. На лицевой стороне панели монитора должны находиться ручки управления регуляторами яркости, контрастности, звука, а также кнопка выключения питания с индикатором. На боковой панели должны под крышкой располагаться регуляторы: частоты кадров, частоты строк, линейность и размеры по вертикали, центровка по горизонтали, для цветных — переключение на режим монохромных мониторов темно-зеленым цветом экрана.

5.7. Монитор должен иметь регулировку угла наклона по горизонтальной оси в пределах $\pm 20°$ и по вертикальной оси — $\pm 30°$.

Корпус монитора должен обеспечивать защиту от ионизирующих и неионизирующих излучений.

Таблица 5

Параметры	Группы		
	А	Б	В
Размер экрана по диагонали (см)	61—67 и более	32—42	31—41
Минимальная яркость знака (кд/м^2)	35	35	35
Регулирование яркости (кд/м^2) в диапазоне	0—120	0—120	0—120
Неравномерность яркости частей знака (%) не более	30	30	30
Величина детального контраста (см. п. 5.2.) (отношение максимальной яркости на экране к минимальной), не менее	5:1	5:1	5:1
Размер точки (мм) при яркости 50 кд/м^2 не более	0,6—0,8	0,4	0,4
Четкость изображения — перепад яркости на границе линии фона при контрасте 5:1, не менее (кд/м^2 на мм)	70	70	70
Частота кадровой развертки (Гц) не менее	50	70	70
Нестабильность изображения (мм, не более)	0,1	0,1	0,1
Геометрические искажения изображения (%, не более)	3	3	3
Нелинейность изображения (%, не более)	3	3	3
Остаточное несведение лучей (мм) в центре	0,5	0,4	—
по углам	0,6	0,5	—
Максимальное количество цветов на экране не менее	16	16	2
Число градаций серого цвета не менее	—	—	8

5.8. Характеристика знаков:

угловой размер знаков не менее 20 угловых минут, оптимальный размер знака составляет 24—35 угловых минут;

высота знака при расстоянии глаз до экрана 50—80 см должна быть не менее 3—4 мм;

оптимальная ширина знака — 0,7 его высоты;

расстояние между знаками должно быть 15—20 % от их высоты;

расстояние между строками по вертикали не менее 50 % от высоты знаков;

толщина штриха не более 1/5—1/6 от его длины, но не менее 0,4 мм;

расстояние между словами в строке не менее 70 % от высоты знака;

емкость экрана не менее 64 знаков в строке и не менее 24 строк;

расстояние между светящимися точками, из которых составлен знак, должно быть меньше диаметра точки.

Для отображения букв и цифр формат матрицы должен быть 7×9 точек.

5.9. Характеристики цветового изображения.

При частоте кадровой (развертки регенерации изображения) не менее 70 Гц необходимо пользоваться позитивным изображением (темные знаки на светлом фоне экрана), при частоте кадровой развертки (регерации изображения) 50 Гц — негативным изображением (светлые знаки на темном фоне).

Яркостный контраст цветных знаков и фона должен быть не менее 60 %.

(Для светящихся изображений прямого и обратного контраста наилучшее цветовое сочетание дают черный и оранжевый цвета. При обратном контрасте на цветных ПЭВМ и ВДТ хорошее сочетание, вызывающее наименьшее утомление зрения, дает темно-зеленый фон и белые знаки.)

При использовании монохромного монитора оптимальным цветом отображения светящихся знаков сложной конфигурации является желто-зеленый участок спектра с длиной волны от 500 до 570 мм, характеризуемый малой насыщенностью (т. е. незначительно отличающийся от белого) и имеющий максимальную видимость.

5.9.1. Красный, фиолетовый и синий (особенно голубой) цвета не рекомендуется для отображения знаков сложной конфигурации (использование неудобных сочетаний цветов фона и текста ухудшают читаемость и различимость, приводят к более выраженному утомлению глаз).

При разработке программного обеспечения можно предъявлять не более 7 цветов одновременно.

5.10. Клавиатура:

клавиатура не должна быть жестко связана с монитором;

по цвету клавиатура должна приближаться к цвету корпуса монитора (коэффициент отражения — 0,4—0,6);

наклон клавиатуры должен находиться в пределах 12—15°;

клавиатура должна иметь опорное приспособление;

толщина клавиатуры на уровне среднего ряда должна быть не более 30 мм;

длина клавиатуры должна позволять пользоваться двумя руками.

В клавиатуре должна быть предусмотрена возможность звуковой обратной связи от включения клавиш с возможностью регулировки.

Клавиши, корпус клавиатуры должны иметь матовую поверхность. Предпочтителен светло-серый цвет клавиш с черным шрифтом на них.

Часто используемые поля клавиш и клавиатуры должны располагаться внизу справа, редко — вверху слева.

Функциональные группы клавиш должны выделяться размером, цветом и формой, а также местом расположения.

Минимальный размер клавиш — 13 мм, рекомендуемый — 15 мм с углублением (с вдавленной поверхностью в центре клавиши). Шаг клавиши 19 мм±1 мм, расположение клавиш по ГОСТ 14289—88.

Сопротивление клавиш минимальное — 0,25 Н, максимальное — 1,5 Н.

Должен обеспечиваться свободный одинаковый ход для всех кнопок и клавиш.

Расстояние между краями вогнутых поверхностей клавиши должно быть не менее 3 мм.

6. Режимы учебных занятий с использованием ПЭВМ и ВДТ

6.1. Рациональный режим занятий учащихся предусматривает соблюдение регламентированной длительности непрерывной работы с видеотерминалом и перерывов, а также соблюдение профилактических мероприятий, направленных на охрану здоровья учащихся.

6.2. Длительность работы на ПЭВМ и ВДТ во время учебных занятий в средних учебных заведениях определяется характеристиками видеотерминалов, возрастом учащихся, временем начала работы, длительностью перемен, предшествующих работе за видеотерминалом при соблюдении гигиенических требований к условиям, организации рабочего места и соблюдения правильной посадки.

Длительность работы за видеотерминалами (размер светящейся точки в центре экрана

не более 0,4 мм) не должна превышать:

для учащихся V классов — 15 мин;

для учащихся VI—VII классов — 20 мин;

для учащихся VIII—IX классов — 25 мин;

для учащихся X—XI классов при двух уроках подряд на первом из них — 30 мин, на втором — 20 мин.

Длительность работ на ПЭВМ и ВДТ со светящейся точкой в центре экрана до 0,5 мм уменьшается до 30 %:

для учащихся V классов — 10 мин;

для учащихся VI—VII классов — 15 мин;

для учащихся VIII—IX классов — 18 мин;

для учащихся X—XI классов — 20 минут (на первом сдвоенном уроке) и на втором — 15 мин.

6.3. Число занятий с использованием ПЭВМ и ВДТ должно быть не более двух в день для учащихся X—XI классов и не более одного для учащихся V—IX классов.

6.3.1. Перемена между уроками, на которых используется ПЭВМ или ВДТ, должна быть не менее 10 мин (до 20 мин)

Для учащихся X—XI классов перед пятым уроком, а для учащихся VIII—IX классов — перед четвертым уроком целесообразно устраивать перемену длительностью 50—60 мин для обеда и отдыха учащихся.

6.4. Во время производственной практики ежедневная длительность работы за видеотерминалами, светящаяся точка которых до 0,4 мм,— 3 ч, а за видеотерминалами со светящейся точкой до 0,5 мм — 2 ч.

6.5. Факультативная и кружковая работы с использованием ПЭВМ и ВДТ для учащихся старших классов должна быть не более двух академических часов в неделю.

6.6. Для предупреждения развития переутомления зрительного анализатора необходимо проводить комплекс упражнений для глаз (Приложение 2).

При работе за видеомониторами учащимся необходимо выполнять комплекс упражнений для глаз через каждые 20—25 минут, в зависимости от типа видеотерминала (см. п. 6.2.).

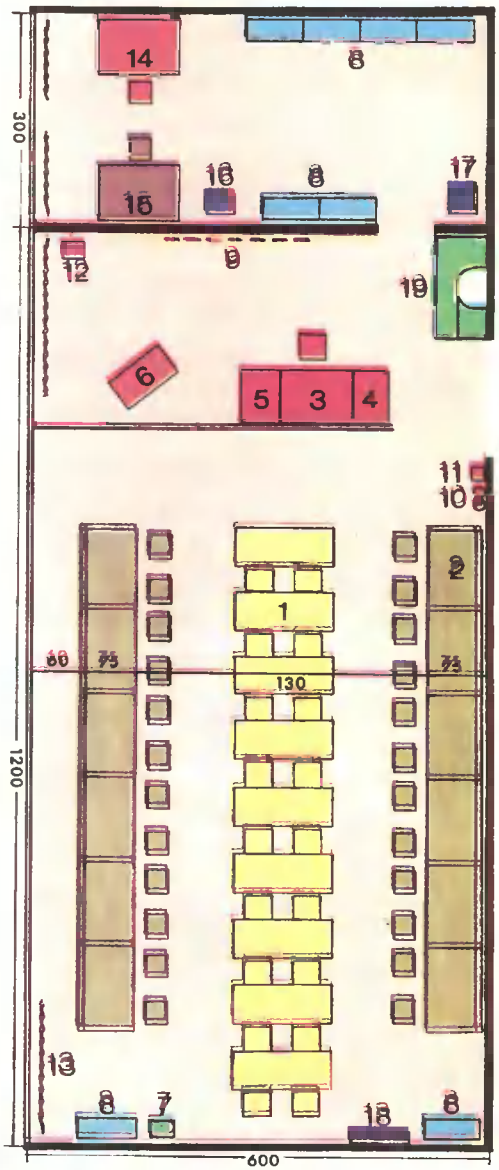
6.7. Для снятия общего утомления учащихся X—XI классов между уроками информатики и вычислительной техники необходимо проводить физкультпаузу, включающую упражнения общего воздействия, улучшающую функциональное состояние нервной, сердечно-сосудистой, дыхательной систем, снимающую утомление с мышц плечевого пояса, рук, туловища и ног (Приложение 3).

Комплексы упражнений для глаз и физкультпаузы рекомендуется менять не реже одного раза в месяц.

При проведении предупредительного и текущего санитарного надзора за организацией, условиями и режимом обучения учащихся всех типов средних учебных заведений при работе на ПЭВМ и ВДТ следует руководствоваться перечнем официальных документов (Приложение 4).

Приложение 1 а

Кабинет вычислительной техники

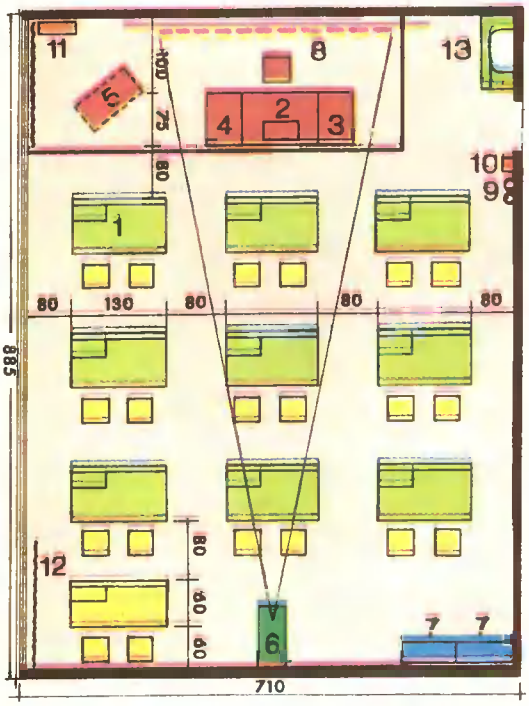


1. Стол двухместный ученический 130 (120)×50.
2. Стол двухместный с ПЭВМ
3. Стол преподавателя с ПЭВМ 130 (120)×75.
4. Стол-подставка под графопроектор.
5. Стол-подставка под принтер.
6. Телевизор.
7. Подставка под проектор.
8. Шкаф пристенный 90×45.
9. Доска меловая.
10. Огнетушители.

11. Сигнализация.
12. Электрощитовая.
13. Шторы.
14. Стол рабочий 130 (120)×75.
15. Стол радиомонтажный с местным отсосом.
16. Тумба для инструментов.
17. Сейф-шкаф.
18. Настенный стенд.
19. Умывальник.

Приложение 1 б

Кабинет вычислительной техники



1. Стол двухместный с ПЭВМ 130 (120)×75.
2. Стол преподавателя с ПЭВМ 130 (120)×75.
3. Стол-подставка под графопроектор.
4. Стол-подставка под принтер.
5. Телевизор.
6. Подставка под проектор.
7. Шкаф пристенный 90×45.
8. Доска меловая.
9. Огнетушители.
10. Сигнализация.
11. Электрощитовая.
12. Шторы.
13. Умывальник.

Приложение 2

Комплексы упражнений для глаз

Упражнения выполняются сидя или стоя, отвернувшись от экрана, при ритмичном дыхании, с максимальной амплитудой движения глаз.

Вариант 1.

1. Закрыть глаза, сильно напрягая глазные мышцы, на счет 1—4, затем раскрыть глаза, расслабив мышцы глаз, посмотрев вдаль, на счет 1—6.

Повторить 4—5 раз.

2. Посмотреть на переносицу и задержать взор на счет 1—4. До усталости глаза доводить нельзя. Затем открыть глаза, посмотреть вдаль на счет 1—6.

Повторить 4—5 раз.

3. Не поворачивая головы, посмотреть направо и зафиксировать взгляд на счет 1—4, затем посмотреть вдаль прямо на счет 1—6. Аналогичным образом проводятся упражнения, но с фиксацией взгляда влево, вверх, вниз.

Перевести взгляд быстро по диагонали: направо вверх — налево вниз, потом прямо вдаль на счет 1—6; затем налево — вверх — направо — вниз и посмотреть вдаль на счет 1—6.

Повторить 3—4 раза.

Вариант 2.

1. Закрыть глаза, не напрягая глазные мышцы, на счет 1—4, широко раскрыть глаза и посмотреть вдаль на счет 1—6.

Повторить 4—5 раз.

2. Посмотреть на кончик носа на счет 1—4, а потом перевести взгляд вдаль на счет 1—6.

Повторить 4—5 раз.

3. Не поворачивая головы (голова прямо), делать медленно круговые движения глазами вверх — вправо — вниз — влево и в обратную сторону: вверх — влево — вниз — вправо. Затем посмотреть вдаль на счет 1—6.

Повторить 4—5 раз.

4. При неподвижной голове перевести взор с фиксацией его на счет 1—4 вверх — прямо — вниз — прямо — вправо — прямо — влево — прямо. Аналогичным образом проделать движения по диагонали в одну и другую сторону.

Повторить 4—5 раз.

Вариант 3.

1. Поморгать, не напрягая глазные мышцы, на счет 10—15.

2. Не поворачивая головы (голова прямо) с закрытыми глазами посмотреть направо на счет 1—4, затем налево на счет 1—4 и прямо на счет 1—6. Поднять глаза вверх на счет 1—4, опустить вниз на счет 1—4 и перевести взгляд прямо на счет 1—6.

Повторить 4—5 раз.

3. Посмотреть на указательный палец, удаленный от глаз на расстояние 25—30 см, на счет 1—4, затем перевести взор вдаль на счет 1—6.

Повторить 4—5 раз.

4. В среднем темпе проделать 3—4 круговых движения в правую сторону, столько же в левую сторону и, расслабив глазные мышцы, посмотреть вдаль на счет 1—6.

Повторить 1—2 раза.

Приложение 3*

Комплексы физкультурных пауз

Физкультурная пауза повышает двигательную активность, стимулирует деятельность нервной, сердечно-сосудистой, дыхательной и мышечной

* (Извлечение из кн.: Н. И. Косилиной, А. П. Колтановского «Производственная гимнастика для работников умственного труда». М.: Физкультура и спорт, 1983.)

систем, снимает общее утомление, повышает умственную работоспособность.

Физкультурная пауза 1.

Ходьба на месте — 20—30 с. Темп средний.

1. Исходное положение (и. п.) — основная стойка (о. с.). 1 — руки вперед, ладони книзу. 2 — руки в стороны, ладони кверху. 3 — стать на носки, руки вверх, прогнуться. 4 — и. п.

Повторить 4—6 раз. Темп медленный.

2. И. п.— ноги врозь, немного шире плеч. 1—3 — наклон назад, руки за спину. 3—4 — и. п.

Повторить 6—8 раз. Темп средний.

3. И. п.— ноги на ширине плеч. 1 — руки за голову, поворот туловища направо. 2 — туловище в и. п., руки в стороны, наклон вперед, голову назад. 3 — выпрямиться, руки за голову, поворот туловища налево. 4 — и. п. 5—8 — то же в другую сторону.

Повторить 6 раз. Темп средний.

4. И. п.— руки к плечам. 1 — выпад вправо, руки в стороны. 2 — и. п. 3 — присесть, руки вверх. 4 — и. п. 5—8 — то же в другую сторону.

Повторить 6 раз. Темп средний.

5. И. п.— ноги врозь, руки на пояс. 1—4 — круговые движения туловищем вправо. 5—8 — круговые движения туловищем влево.

Повторить по 4 раза. Темп средний.

6. И. п.— о. с. 1 — мах правой ногой назад, руки в стороны. 2 — и. п. 3—4 — то же левой ногой.

Повторить 6—8 раз. Темп средний.

7. И. п.— ноги врозь. Руки на пояс. 1 — голову наклонить вправо. 2 — не выпрямляя головы, наклонить ее назад. 3 — голову наклонить вперед. 4 — и. п. 5—8 — то же в другую сторону.

Повторить 4—6 раз. Темп средний.

Физкультурная пауза 2.

Ходьба на месте — 20—30 с. Темп средний.

1. И. п.— о. с., руки за голову. 1—2 — встать на носки, прогнуться, отвести локти назад. 3—4 — опуститься на ступни, слегка наклониться, вперед, локти вперед.

Повторить 6—8 раз. Темп медленный.

2. И. п.— о. с. 1 — шаг вправо, руки в стороны. 2 — повернуть кисти ладонями вверх. 3 — приставить левую ногу, руки вверх. 4 — руки дугами в стороны и вниз, свободным махом скрестить и перед грудью, 5—8 — то же влево.

Повторить 6—8 раз. Темп средний.

3. И. п.— стойка ноги врозь, руки в стороны, 1 — наклон вперед к правой ноге, хлопок в ладони. 2 — и. п. 3—4 — то же в другую сторону.

Повторить 6—8 раз. Темп средний.

4. И. п.— стойка ноги врозь, левая впереди, руки в стороны или на поясе. 1—3 — три пружинистых полуприсяда на левой ноге. 4 — переменить положение ног. 5—7 — то же, но правая нога впереди левой.

Повторить 4—6 раз. Перейти на ходьбу — 20—25 с. Темп средний.

5. И. п.— стойка ноги врозь пошире. 1 — с поворотом туловища налево, наклон назад, руки назад. 2—3 — сохраняя положение туловища в повороте, пружинистый наклон вперед, руки вперед. 4 — и. п. 5—8 — то же, но поворот туловища вправо.

Повторить по 4—6 раз в каждую сторону.

Темп медленный.

6. И. п.— придерживаясь за опору, согнуть правую ногу, захватив рукой за голень. 1 — вставая на левый носок, мах правой ногой назад, правую руку в сторону — назад. 2 — и. п. 3—4 — то же, но согнуть левую ногу.

Повторить 6—8 раз. Темп средний.

7. И. п.— о. с. 1 — руки назад в стороны, ладони наружу, голову наклонить назад. 2 — руки вниз, голову наклонить вперед.

Повторить 6—8 раз. Темп медленный.

Физкультурная пауза 3.

Ходьба на месте — 20—30 с. Темп средний.

1. И. п.— о. с. 1 — правой рукой дугой внутрь. 2 — то же левой и руки вверх, встать на носки. 3—4 — руки дугами в стороны. И. п.

Повторить 4—6 раз. Темп медленный.

2. И. п.— о. с. 1 — с шагом вправо руки в стороны, ладони кверху. 2 — с поворотом туловища направо дугой кверху левую руку вправо с хлопком в ладони. 3 — выпрямиться. 4 — и. п. 5—8 — то же в другую сторону.

Повторить 6—8 раз. Темп средний.

3. И. п.— стойка ноги врозь. 1—3 — руки в стороны, наклон вперед и три размашистых поворота туловища в стороны. 4 — и. п.

Повторить 6—8 раз. Темп средний.

4. И. п.— о. с. 1—2 — присед, колени врозь, руки вперед. 3—4 — встать, правую руку вверх, левую за голову. 5—8 — то же, но правую за голову.

Повторить 6—10 раз. Темп медленный.

5. И. п.— о. с. 1 — выпад влево, руки в стороны. 2—3 — руки вверх, два пружинистых наклона вправо. 4 — и. п. 5—8 — то же в другую сторону.

Повторить 4—6 раз. Темп средний.

6. И. п.— правую руку на пояс, левой поддерживаться за опору. 1 — мах правой ногой вперед. 2 — мах правой ногой назад, захлестывая голень.

Повторить по 6—8 раз каждой ногой.

Темп средний.

7. И. п.— о. с. 1—2 — правую ногу назад на носок, руки слегка назад с поворотом ладоней наружу, голову наклонить назад. 3—4 — ногу приставить, руки расслабленно опустить, голову наклонить вперед. 5—8 — то же, отставляя другую ногу назад.

Повторить 6—8 раз. Темп медленный.

Приложение 4

Перечень официальных документов

1. «Санитарные правила работы с источниками неиспользуемого рентгеновского излучения», № 1960—79, утв. Минздравом СССР 19.01.79. М., 1982.

2. «Нормы радиационной безопасности». НРБ-76/87 и «Основные санитарные правила работы с радиоактивными веществами и другими источниками ионизирующего излучения». М.: Энергоатомиздат, 1988.

3. «Санитарно-гигиенические нормы допустимых уровней ионизации воздуха производственных и общественных помещений», № 2152—80, утв. Минздравом СССР 12.02.80.

4. «Санитарные нормы допустимого шума в помещениях жилых и общественных зданий и на территории жилой застройки», № 3077—84, утв. Минздравом СССР 03.08.84.

5. «Санитарные нормы допустимых уровней шума на рабочих местах», № 3223—85, утв. Минздравом СССР 12.03.85.

6. СНиП 11—4—79 «Естественное и искусственное освещение. Нормы проектирования». М., 1980.

7. Постановление Госстроя СССР «Об изменении главы СНиП II—4—79 «Естественное и искусственное освещение. Нормы проектирования», № 205 от 04.12.85.

8. СНиП II—33—75 «Отопление, вентиляция и кондиционирование воздуха. Нормы проектирования». М., 1985.

9. СНиП 11—12—77 «Защита от шума. Нормы проектирования».

10. ГОСТ ССБТ 12.1.003 — 83 «Шум. Общие требования безопасности (СТ. СЭВ 1930—79)».

11. ГОСТ ССБТ 12.1.006—84 «Электромагнитные поля радиочастот. Допустимые уровни на рабочих местах и требования к проведению контроля».

Изменение № 1 к ГОСТ.

12. ГОСТ ССБТ 12.2.032—78 «Рабочее место при выполнении работ сидя. Общие эргономические требования».

13. ГОСТ 11015—86 «Столы ученические».

14. ГОСТ 11016—86 «Стулья ученические».

15. ГОСТ ССБТ 12.4.124—83 «Средства защиты от статического электричества. Общие технические требования».

16. ГОСТ 14254—80 (СТ. СЭВ 778—77) «Изделия электротехнические. Оболочки. Стенки защиты. Обозначения. Методы испытаний».

17. ГОСТ 14289—88 Средства вычислительной техники. Клавиатуры. Расположение клавиш и символов, функции управляющих клавиш.

18. ГОСТ 29.05.006—85 «Трубки электронно-лучевые приемные. Общие эргономические требования».

19. ГОСТ 15962—84 «Трубки электронно-лучевые приемные.

20. ГОСТ 19785—88 «Трубки электронно-лучевые приемные. Методы измерения и контроля параметров».

21. ГОСТ 21130—75 «Изделия электротехнические. Зажимы заземляющие и знаки заземления. Конструкция и размеры».

22. ГОСТ СЧМ 21829—76 «Кодирование зрительной информации. Общие эргономические требования».

23. ГОСТ СЧМ 23144—78 «Трубки ЭЛ для отображения визуальной информации».

24. ГОСТ 258688—83 «Устройства ввода, вывода и подготовки данных вычислительных машин. Термины и определения».

25. ГОСТ 27016—86 «Дисплеи на электронно-лучевых трубках. Общие технические условия.»

26. ГОСТ 27201—87 «Машины вычислительные электронные персональные. Типы, основные параметры, общие технические требования».

27. ГОСТ 27954—88 «Видеомониторы персональных электронных вычислительных машин. Типы, основные параметры, общие технические требования».

28. Положение о кабинете вычислительной техники всех типов средних учебных заведений, утвержденное Государственным комитетом СССР по народному образованию 09.12.88 г., согласованное с ЦК профсоюза работников просвещения, высшей школы и научных учреждений 31.10.88 г. М., 1989. 45 с.

Изящное решение

Существует достаточно много проблем, для решения которых в реальном масштабе времени трудно рассчитывать на помощь компьютера. Например, изучение взаимодействия кварков. Конечно, исследователи пытались обойти технические ограничения, объединяя большое количество дешевых однокристальных ЭВМ, каждая из которых обрабатывает свой поток информации. Со временем для подобных многопроцессорных однородных комплексов была даже найдена наиболее удобная конструктивная форма — трехмерный «гиперкуб». Но рост скорости вычислений в такой объемной ЭВМ был ограничен ее программным обеспечением, управляющим распределением информационных потоков, поскольку работа ПО должна происходить последовательными шагами.

Ученые сандийской Национальной лаборатории (США) изменили программное обеспечение таким образом, чтобы информационный обмен осуществлялся, как правило, между смежными чипами. Такая переделка привела к резкому увеличению производительности и довела ее почти до абсолютной величины: компьютер, состоящий из 1024 чипов, исполняет программу более чем в 1000 раз быстрее, чем однопроцессорная ЭВМ.

Е. МАРГУЛИС, Ю. КОСОВ, Ю. МЕЛЕЖИК

Компьютерные игры в обучении

Появление в школе компьютера привело к тому, что в жизнь учеников вошли компьютерные игры. С течением времени отношение к ним существенно изменилось. Если на первых порах игра была в центре внимания, рассматривалась как пропедевтика компьютеризации, а подчас и выполняла социально-имитационную функцию (создавая видимость применения компьютеров в учебном процессе), то ныне и практики, и теоретики относятся к игре прохладнее. Первые — потому что она все чаще кажется им малополезным и даже вредным занятием, расслабляющим учащихся и отбирающим драгоценное учебное (а тем более компьютерное!) время. Вторые — поскольку злоупотребление играми ведет к ряду нежелательных эффектов: снижаются волевые качества, возникает иллюзия вседозволенности (вследствие иллюзорности игровых санкций — «авария», «взрыв» и пр.— на экране), причем учащийся иногда сознательно избирает проигрышные варианты, чтобы еще и еще раз посмотреть «интересное» наказание. Нельзя, в соответствии с такой точкой зрения, готовить к учебе и труду, пользуясь преимущественно «несерьезной» игровой мотивацией. Кроме того, в решении игровых задач имеются, как правило, отклонения от нормативного решения задач учебных. Поэтому их трудно применить для систематического изучения учебного материала; преимущественная сфера приложения — контроль и оценка знаний и умений, постановка задач, введение в проблемную ситуацию и т. д.

Подобные соображения не являются абсолютной истиной, с ними можно было бы и поспорить. Но они весьма распространены и существенно влияют на статус компьютерных игр и их место в учебном процессе. Когда чаще всего применяется компьютерная игра? Либо она предваряет изложение нового материала, помогая «озадачить» учащихся и подвести их к нужной проблеме, либо завершает изучение темы, выступая средством контроля и оценки, а иногда и «призом» за хорошие результаты в учении. Сколько времени отводится на игру? Здесь диапазон очень широк, а конкретные цифры зависят от наличия техники, ее загруженности и отношения заведующего кабинетом ВТ к подобному занятию. Как правило, удел игры — внеурочное время, и это затрудняет ее привязку к конкретному предмету. Куда включены компьютерные игры? Либо в отдельные пакеты игровых программ, либо являются «изюминкой» дискеты с обучающими программами. Как выбирает их учитель? Нередко — случайным образом, а иногда вообще не контролирует этот процесс, давая ученикам игровую дискету для «свободного выбора». Здесь, очевидно, нивелируется и проблема оптимизации соотношения между игрой и не игрой.

Как мы предлагаем решать эту проблему? Путем разработки и применения в учебном процессе игровых сериалов — системы компьютерных игр, объединенных одной сюжетной линией и позволяющих многократно возвращаться к одной и той же игре, но различным, поставленным в ее рамках, игровым задачам. Такой путь, действенный и для традиционных длительных педагогических игр, становится особенно продуктивным для игр компьютерных: появляется возможность построения разветвленного пакета игровых учебных программ, применяемых на различных уроках и позволяющего учителю и учащимся свободно выбирать интересую-

щий их фрагмент игры. В пользу такого решения говорят и психологические исследования, показывающие, что для учащихся разного возраста уже само «называние» учебной задачи игрой повышает мотивацию к деятельности. По сути, речь идет о длительном пребывании учащихся в игровой среде, причем даже решение в ней неигровых задач приобретает характерную для игры эмоциональную окраску. Наконец, целесообразность такого пути подтверждают известные примеры популярных мультипликационных сериалов «Ну, погоди!», «Приключения кота Леопольда» и пр., ряда зарубежных учебно-игровых сериалов (например, «Маленький профессор» содержит тысячи задач и учебных игр по математике).

Разумеется, разработка и практическая апробация подобного игрового сериала — весьма трудоемкая задача, однако именно с ее решением связан, по нашему мнению, наиболее реальный путь к признанию учебной эффективности компьютерной игры в практике массового обучения. В настоящее время, насколько позволяют об этом судить имеющиеся публикации, такая работа нигде систематически не проводится. Поэтому попытка проектирования игрового сериала приобретает очевидную практическую значимость.

При разработке сериала мы исходили из того, что он должен удовлетворять следующим требованиям:

а) расширяемость — возможность дополнять имеющийся пакет программ другими играми, естественно сочетающимися с основным сюжетом и направленными на достижение иных актуальных учебных целей;

б) релевантность — указанное расширение не должно являться самоцелью: новые учебные игры и задачи проектируются и модифицируются в соответствии с динамикой актуальных учебных целей; из этого, в частности, следует, что для достижения конкретных целей игра может применяться в «урезанном», фрагментарном виде, что должно предусматриваться в ее программной реализации;

в) должен быть обеспечен свободный доступ учащегося к любому фрагменту игры и его выход из игры в любой момент по собственному желанию или по указанию учителя;

г) должны быть предусмотрены средства индивидуализации деятельности; в частности, учащийся должен иметь возможность выбора различных уровней сложности как для всей игры в целом, так и для каждого ее фрагмента (например, если в соответствующей предметной области его подготовленность выше, чем в других областях).

Учитывая итеративный характер проектирования игр и возможность многоэтапной их модификации, мы ограничились для создания первого варианта реализацией перечисленных требований. Одна из перспектив их расширения — обеспечение индивидуализированного управления учебной деятельностью на основе динамической модели личности учащегося.

В статье описывается фрагмент компьютерного игрового сериала «Стажер», который может быть рассмотрен как самостоятельная учебная игра.

Краткое описание сюжета игры

Учащемуся предлагается принять участие в компьютерной игре, которая потребует от него не только знаний по различным учебным предметам, но и умений применять их в сложной обстановке и в условиях дефицита времени, поможет оценить его способности к решению проблем в условиях неполной информации. Во вводной части игроку сообщается, что он зачислен стажером в отряд космонавтов, осваивающих неизвестные планеты; его перевод из стажеров в члены отряда будет зависеть от проявленных им в пробных испытаниях качеств. В центре управления полетами стажеру предлагается пройти психологическое обследование. Этим достигается возможность в игровой форме провести психодиагностику («встроенность» психодиагностических методик в сюжет игры — один из принципов ее проектирования). Конкретные методики и способ обработки результатов, а также построение на их основе обучающих воздействий здесь не обсуждаются, поскольку их подбор зависит от представлений о динамической модели учащегося; эта задача решается на следующем этапе проектирования, который предполагает использование в игровом сериале элементов интеллектуальной обучающей системы.

Далее стажер «летит» на избранную им планету, где, попадая в различные экстремальные ситуации, он должен решить соответствующие актуальным учебным целям задачи.

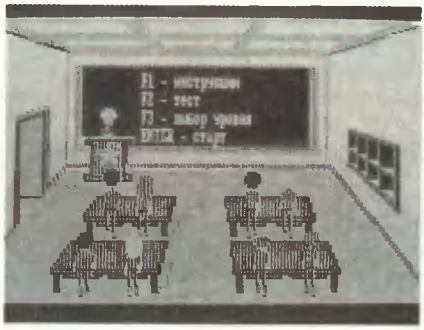
Учебные цели

Описываемый фрагмент игрового сериала «Стажер» посвящен изучению и обобщающему повторению ряда вопросов из курса химии. Ближайшие учебные цели связаны с формированием способов действий по определению состава атмосферы: освоение таблицы растворимости, составление уравнений химических реакций, выявление их качественных особенностей. Более отдаленные учебные цели — выработка умений принимать решения в экстремальных условиях,

выработка системных представлений у школьников, развитие логического мышления и т. д.

Рассматриваемый фрагмент сериала «Стажер» называется «Пилот». В структуре программы можно выделить четыре основные части. Первая часть включает заголовок, вводный текст общего характера (драматизацию), в котором учащемуся предъявляется игровая ситуация; «процедуру старта», выполненную в виде мультипликации «ракета летит по звездному небу»; при этом возможно

1

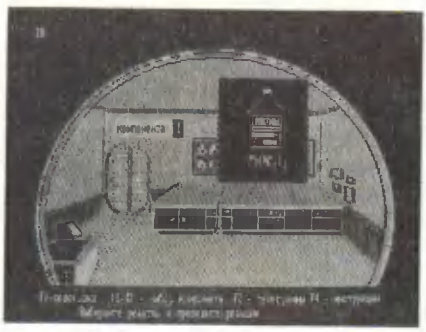


расширение игры путем включения задач по астрономии и физике с соответствующей драматизацией. Сообщения и рекомендации дает учащимся-стажерам профессор Космического центра (рис. 1), который далее предлагает меню для выбора следующих действий: установление уровня сложности и уровня оказываемой в процессе игры помощи в зависимости от того, на какую роль претендует играющий — капитана, борт-инженера или юнги; тестирование для определения уровня подготовленности, получения инструкций по игре, продолжения игры. Здесь же, как и в любом месте программы, предусматривается возможность выхода из игры по желанию учащегося.

При выборе уровня сложности учащемуся предлагается выбрать одну из девяти планет, на которые он может отправиться.

Вторая часть — программа «Атмосфера» — представляет собой основной модуль системы. В инструкции указывается, что 1-й

2



3

уровень — самый легкий (планета с самым простым составом атмосферы), а 9-й — самый сложный. На самом деле тенденция к возрастанию сложности не соблюдается, однако анализ выборов позволяет выявить уровень притязаний учащегося.

Каждый вариант атмосферы содержит четыре газа, которые учащийся должен идентифицировать. После некоторой паузы, которая может быть сокращена по желанию пользователя, учащийся автоматически попадает в химическую лабораторию на борту космического корабля (рис. 2), где ему и предлагается провести анализ компонент атмосферы. Анализ производится путем выбора одной из четырех компонент (клавиши 1—4). Реактив, воздействующий на нее, выбирается путем перемещения игрового персонажа вдоль стеллажа. Результат каждого выбора — воздействия реактива на компоненту — отображается на экране компьютера-анализатора (рис. 3).

Из лаборатории имеется возможность выхода в библиотеку (блок «Помощь»), а также передачи полученных результатов на Землю или получения с Земли правильного ответа в случае отказа от игры (рис. 4).

В библиотеке учащемуся в виде меню предъявляется список дополнительной информации, которую он по своему усмотрению может затребовать (рис. 5). Это:

1) список 16 газов, из которых и составлены 9 возможных вариантов «атмосфер планет»: H_2; N_2; O_2; Cl_2; NH_3; H_2O; CO_2; CO; CH_4; HCl; SO_2; SO_3; H_2S; NO; NO_2; HCN;

2) таблица растворимости и результат реакции, например:

Компонента 1+pH=7.
Компонента 1+O_2.
Реакция не происходит.
Компонента 3+H_2.
Реакция не происходит.
Компонента 2+H_2.
Горит;

3) распечатка протокола анализа атмосферы, т. е. последовательность ходов, где конспективно демонстрируются все действия играющего;

4

4) возможность проверки реакций (пользователь может проверить реакцию любого из 16 газов с любым из имеющегося набора реактивов), например:

pH <7

Это могут быть:

HCl; Cl_2; SO_2; SO_3; NO_2

Реакция с $AgNO_3$:

выпадает белый осадок

Это могут быть:

HCl; Cl_2; SO_3

Реакция с Br_2 (H_2O):

раствор обесцвечивается

Вывод: Это Cl_2;

5) возврат в лабораторию.

При входе в блок «Ответ» на экран выводится список всех возможных газов. Учащемуся предоставлена возможность выбрать любые четыре газа, имеющиеся, по его мнению, в исследуемой атмосфере. Эти газы (их формулы) помещаются в правый верхний угол экрана.

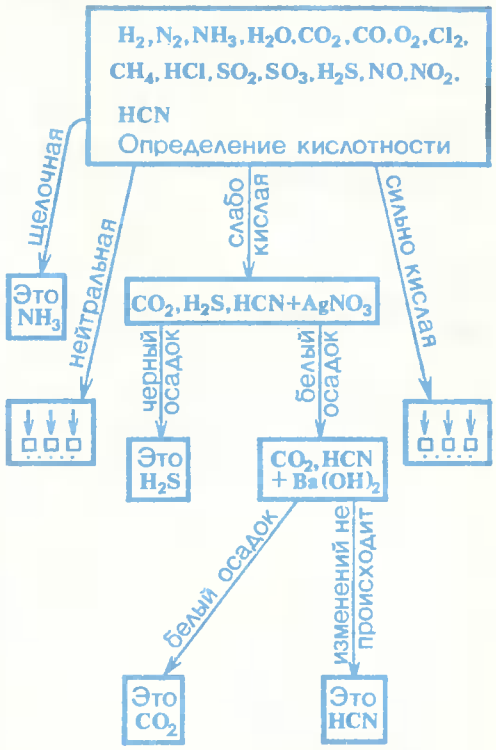
Все действия учащегося заносятся в память для дальнейшего использования при анализе стратегии игры.

Пользователь может дать ответ три раза. По окончании первой попытки в верхней части экрана в юмористической форме оценивается правильность ответа. После второй попытки неправильно выбранные компоненты высвечиваются в инверсном изображении. После третьей попытки автоматически происходит переход к демонстрации эталонного анализа ситуации. В сюжет вплетены элементы, ассоциирующиеся с известными научно-фантастическими и художественными произведениями, используются графические образы из популярных компьютерных игр.

5

Вторая часть программы завершается в соответствии с одним из критериев: окончанием времени анализа, исчерпанием лимита времени (600 с), использованием трех попыток ответа, отказом от дальнейшего анализа или же вводом правильного ответа. В этом случае учащийся должен ответить на последний вопрос второй части: «Пригодна ли атмосфера для дыхания?». Далее автомати-

6



чески вызывается третья часть программы, называемая «Экзамен».

Она является по своей сути диагностикой уровня усвоения материала. Учащемуся ставится задача, которая по условию и логике решения связана с анализом в программе «Атмосфера», требует овладения способами решения соответствующих задач.

В этом блоке предусматривается решение учащимся пяти критериальных задач. Для решения каждой из них отводится 60 с. Если за это время решение не найдено, считается, что ответ на вопрос не поступил.

Ответы учащегося сохраняются для дальнейшего использования в четвертой части программы, называемой условно «Для учителя».

Фактически — это протокол всех действий учащегося во время игры. Он может выда-

ваться на устройство печати или на экран. На рис. 6 приведен фрагмент возможного правильного пути решения игровой задачи, который может служить ориентиром для учителя, оценивающего учебную игру (как процесс, так и результат). Отклонения от представленных на рисунке действий могут свидетельствовать о недостаточной сформированности у учащегося соответствующих способов действий по решению химических задач, позволяют выявить ошибки, установить путем анализа протокола их причины и наметить пути устранения. Следует отметить, что на этом рисунке показан фрагмент лишь одного из возможных принципов анализа атмосферы — принципа кислотности, который выбран исходя из следующих соображений:

а) этот алгоритм приводит к однозначному определению вещества;

б) этот путь требует умения сопоставить результаты анализа и, следовательно, способствует развитию логического мышления учащихся;

в) данный алгоритм — по сравнению с другими — ведет к цели за наименьшее число шагов;

г) знания, необходимые для построения данного алгоритма, связаны преимущественно с изучением свойств оснований и кислот и поэтому учебная игра может успешно использоваться при изучении соответствующих тем школьного курса.

На рис. 7 приведен пример полного протоколирования действий учащегося (протокол для учителя), на основании которого производится анализ процесса и результата игры. Как видно из протокола, на 1-м шаге учащийся определил, какова кислотность первой компоненты. Оказалось, что среда нейтральная. На 2-м шаге была проверена возможность данной компоненты вступать в реакцию с кислородом. Следовательно, можно однозначно выделить O_2, H_2O, N_2. Реакцией с водородом («не реагирует») был «отсечен» кислород. На 4-м шаге, проведя реакцию с водой и получив на экране ответ «Весь газ поглощается раствором», учащийся определил, что это вода (видно из его ответа).

Далее испытуемый перешел к анализу второй компоненты (шаг 6-й), определив слабокислую среду. На 7-м шаге однозначно был определен сероводород, поскольку выпал черный осадок. 8-й шаг, сделанный учащимся, оказался поэтому излишним.

Шаг 9-й — анализ третьей компоненты. Среда нейтральная. На следующем шаге учащийся, по-видимому, определил CH_4 или H_2. Однако он не учел, что это может быть CO. На 11-м шаге была сделана попытка «угадать» третью и четвертую компоненты. Шаг 12-й также оказывается совершенно ненужным.

ВАРИАНТ N 3 (H_2O, H_2S, H_2, NO)
Протокол анализа
Время анализа 387 секунд

	Номер компонента		Реагент
1	1		Определение кислотности - среда нейтральная
2	1	+	O_2
3	1	+	H_2
4	1	+	H_2O
5	Ответ N 1. (H_2O, Cl_2, N_2, HCl)		
6	2		Определение кислотности - среда слабо кислая
7	2	+	$AgNO_3$
8	2	+	$KMgO_4$
9	3	+	Определение кислотности - среда нейтральная
10	3	+	O_2
11	Ответ N 2 (H_2O, H_2S, H_2, CH_4)		
12	3	+	$Ba(OH)_2$
13	3		Определение кислотности - среда нейтральная
14	4	+	O_2
15	4	+	NaOH
16	4	+	H_2
17	Просмотр протокола анализа		
18	Проверка реакций		
19	Ответ N 3 (H_2O, H_2S, H_2, NO)		

Критериальная задача

Правильный ответ	Ответ учащегося
Cl_2	Cl_2
CH_4	H_2
CO_2	CO_2
NH_3	NH_3
CO	???

Шаг 13-й посвящен анализу pH третьей компоненты (повторение). Здесь, видимо, лучше было бы обратиться к блоку «Помощь» — «протокол анализа». На 14-м шаге был определен четвертый компонент смеси.

Шаги 15-й и 16-й не нужны. Видимо, проанализировав свои действия и два полученных ответа, а также соответствующие комментарии, учащийся сделал правильные выводы и дал нужный ответ (угадывание ответа не исключено, хотя и маловероятно).

Таким образом, протокол анализа позволяет установить, что до 8-го шага правильно были определены первая и вторая компоненты. Дальнейшие шаги (компоненты 3 и 4) сочетали в себе попытку угадать правильный ответ с достаточно логичными попытками решить задачу. Автоматизация процесса анализа ответа / протокола — одна из линий доработки игровой программы, которая проводится в настоящее время.

Апробация «Пилота» была проведена в химическом кружке Киевского городского Дворца пионеров и школьников им. Н. Островского, в ряде школ г. Киева. Получены благоприятные отзывы учителей и методистов.

Дополнительная информация о разработке игровых сериалов для различных типов компьютеров может быть получена по адресу: 252033, Киев, ул. Степана Халтурина, 2. НИИ психологии УССР. Маргулису Евгению Дмитриевичу.

А. СПИВАКОВСКИЙ
г. Херсон

Педагогические программные средства: объектно-ориентированный подход

На нынешнем этапе развития программного и методического обеспечения компьютерного обучения особо актуальным является поиск новых подходов — как на уровне идей, так и на уровне написания программ. Ведь большинство существующих и создающихся педагогических программных средств (ППС) реализуют в основе своей идеи программированного обучения.

Здесь будет изложен новый подход (обозначенный в заголовке) к созданию обучающих программ и описанию адекватной программной реализации в конкретной предметной среде.

На наш взгляд, основу объектно-ориентированного подхода к созданию ППС должна составить следующая система требований.

1. Пользователь должен работать с реальными объектами предметной области (уравнениями, многочленами, матрицами, системами линейных уравнений и т. п.), а не с текстами и вопросами, как это реализуется в программированном обучении.

2. Пользователь должен работать в реальной операционной среде, однозначно определяемой предметной областью (например, для матриц: сложи две строки, домножь строку на число, переставь две строки местами и т. п.).

3. Интерфейс пользователя практически не должен отличаться от традиционного (лист бумаги заменяется экраном, последовательность листов бумаги — последовательностью кадров на экране, которые можно передвигать как вперед, так и назад, карандаш — курсором, резинка — забоем). При этом пользователь в рамках дидактических целей может использовать вычислительные ресурсы, предоставляемые ППС.

4. ППС должно предоставлять пользователю свободу, ограниченную только рамками предметной области (например, с матрицей можно делать любые элементарные преобразования в любой последовательности, главное — найти ее ранг), т. е. пользователь не должен находиться под прессом алгоритма решения, определенного на стадии написания ППС. Причем пользователь должен иметь возможность «передвигаться» по своим действиям, вставляя между ними новые. Более того, пользователь должен иметь возможность вообще отказаться от операционной среды ППС и строить произвольно новый объект, а дело программы — оценить правильность его действий.

5. Пользователь должен всегда иметь выход из тупиковых ситуаций, для чего операционную среду необходимо дополнить, например, приказом «Очередной шаг решения сделай, машина, сама».

6. История работы пользователя должна представляться в виде последовательности его действий с соответствующей диагностикой, что резко облегчит работу проверяю-

щего, а также позволит пользователю иметь оперативную информацию о правильности своих действий.

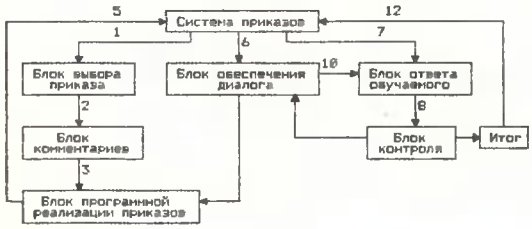
7. ППС должно иметь два режима:

а) «пассивное наблюдение» за работой пользователя с выдачей информации о правильности его действий;

б) «параллельная работа» (обучаемый сделал, затем программа показала, как бы сделала она).

Очевидно, ППС, удовлетворяющее вышеперечисленным свойствам, интегрирует в себе функции демонстратора, тренажера и контролера в естественной для обучаемого форме, позволяя при этом преподавателю проследить не только диагностику, но и логику действий обучаемого.

Диаграмма схематично отражает структуру программной реализации объектно-ориентированного подхода.



Опишем вкратце технологию производства ППС, в основе которой лежит рассматриваемый подход, на примере созданного программного средства «Системы линейных уравнений» для ПЭВМ «Ямаха», написанного на Паскале.

1. Определяем систему объектов, с которыми будет работать обучаемый (в данном случае — системы линейных уравнений над полем рациональных чисел, без ограничения на совместность и количество решений).

2. Определяем операционную среду пользователя (см. табл.).

3. Определяем интерфейс пользователя. В данном случае он представлен четырьмя окнами.

Первое окно, расположенное в строках 0—2, столбцах 0—80 экрана, предназначено для организации управления учебным процессом. Именно здесь обучаемый имеет возможность вызвать исполнение приказов 1—17.

Второе окно, расположенное в строках 3—12, столбцах 0—40, организует работу с последовательностью действий обучаемого.

Третье окно, расположенное в строках 3—12, столбцах 40—80, позволяет видеть пользователю параллельную работу ППС над матрицей.

№ п/п	Приказы	Действие приказа
1	+	Производит сложение строк, указанных пользователем
2	—	Производит вычитание строк, указанных пользователем
3	*	Производит домножение строки системы на множитель
4	/	Производит сокращение строки системы на множитель
5	p	Производит перестановку двух строк системы
6	r	Производит перестановку двух столбцов системы
7	b	После приведения системы к диагональному виду организует определение ранга основной и расширенной матриц системы, совместности системы, числа решений, частного и фундаментальных решений системы линейных уравнений
8	o	Организует работу с коэффициентом основной матрицы
9	c/o	Возвращает систему в форме, предшествующей приказу «o»
10	del	Организует удаление нулевой строки системы
11	x	Предоставляет пользователю перейти к произвольной системе. ППС оценит эквивалентность введенной и исходной систем. Данный приказ предоставляет пользователю полную свободу в рамках предметной области
12	←	Позволяет пользователю возвратиться к любой системе, полученной им ранее, и, если надо, продолжить работу с этого места (откатка назад)
13	→	То же, что и откатка назад, но вперед
14	⌴	В случае затруднения очередное преобразование выполнит компьютер с кратким пояснением
15	cntr/f	Вывод на экран полного решения системы машиной или распечатка на принтере всех выполненных действий
16	cntr/q	Настоящее ППС предоставляет возможность моделировать две схемы взаимодействия обучаемого и педагога: а) обучаемый выполнил действие — педагог прокомментировал его правильность; б) обучаемый выполнил действие — педагог прокомментировал его и предложил свой шаг в решении задачи Настоящий приказ позволяет переключать эти режимы.
17	h	Вызов окна помощи, в котором непосредственно можно выбрать один из приказов 1—16

Четвертое окно организует работу по приказу h.

Интерфейс может в каждом конкретном случае иметь свою структуру, однако должен настраиваться (в разумных пределах) самим обучаемым.

4. Определяем алгоритм решения задачи, которая будет задаваться обучаемому, таким образом, чтобы соответствующая процедура «сумела оценить» состояние изменяемого объекта и найти из системы приказов нужный (см. приказ 14).

5. Создаем процедуры, реализующие приказы 1—13 и 15—17.

6. На основе диаграммы строим управляющую программу, отражающую структуру дерева обучения.

В заключение необходимо сказать, что технология конкретного наполнения пунктов 4—6 является предметом отдельного исследования, имеющего значительный объем, и потому в рамках настоящей статьи не описывается.

Б. КИСЕЛЕВ, Н. ЛЕОНОВА

Педагогика сотрудничества и компьютеризация учебного процесса

Физико-математическая школа № 542 при Московском инженерно-физическом институте организована в 1982 г. и является одной из первых школ в стране, созданных в рамках комплексных мер по совершенствованию системы народного образования. Отличительная особенность школы — профессорско-преподавательский состав института непосредственно участвует в учебно-воспитательном процессе школы. Ряд учебных дисциплин курируется определенными кафедрами, которые отвечают за организацию обучения, методическую подготовку преподавателей, материальное оснащение лабораторий, обеспечение занятий соответствующей учебно-методической литературой. В школе по профилирующим дисциплинам введена вузовская система проведения занятий: лекции — практические занятия (семинары) — лабораторные работы.

С момента создания школы большое внимание уделяется изучению учащимися основ вычислительной техники и внедрению ЭВМ в учебный процесс. Так курс, получивший название «Основы вычислительной техники», был введен в учебный план школы в 1982 г., т. е. за три года до официального включения ОИВТ в программу общеобразовательной школы.

Нами ставилась задача согласованного достижения нескольких целей:

совершенствования учебного процесса по всем предметам;

работы с учащимися, проявившими интерес и способности к

проблемам информатики и программирования;

решения воспитательных проблем.

Под совершенствованием учебного процесса подразумевается возможность передать в руки преподавателю компьютер как мощное средство, с помощью которого он мог бы быстро, полно и точно передавать свои знания и умения ученикам.

Учебный процесс в этом случае строится как единый цикл компьютерных уроков, каждый из которых предполагает ту или иную форму использования вычислительной техники при подаче учебного материала, контроле и в самостоятельной работе учащихся.

Основные проблемы, с которыми приходится сталкиваться при организации компьютерного урока, состоят в выборе или разработке технического, программного и информационного, а также методического обеспечения.

В данной работе мы останавливаемся на двух последних аспектах разработки компьютерного урока, которые, по нашему мнению, тесно взаимосвязаны.

Хорошо известно: для того чтобы учитель мог эффективно использовать компьютер, ему необходимо дать в руки соответствующее программное обеспечение. Опыт показывает, что типовой программный продукт, распространяемый по линии просвещения, далеко не всегда может быть адаптирован в соответствии с требованиями учебного процесса в конкретной школе в рамках конкретного курса. Так, у нас в школе доля компьютерных уроков по курсу информатики, основанных на использовании типового программного продукта составляет 9 % учебного времени. Учитель, который желает более широко использовать компьютерную технику, должен искать другие пути разработки программного обеспечения. Как правило, учитель не может самостоятельно создать программный продукт в необходимом количестве и необходимого качества, а заказывать его

ввиду высокой стоимости разработки не позволяют финансовые возможности школы.

На базе вычислительного центра школы № 542 в 1985 г. создан компьютерный клуб с целью привлечения студентов вуза к разработке программного обеспечения для школы. Вскоре в состав членов клуба вошли и школьники, которые на равных со студентами стали принимать участие в его работах.

Сейчас в состав компьютерного клуба входят преподаватели, выпускники вуза, студенты, школьники и «дошкольники». Под «дошкольниками» мы подразумеваем учеников VII—VIII классов из других школ, участвующих в работе нашего клуба.

Преподаватели обеспечивают постановку общей задачи, контроль за выполнением, апробацию в условиях учебного процесса.

Выпускники вуза выполняют функцию руководителей групп, ведущих разработку конкретных программных продуктов.

Студенты — это системные и прикладные программисты.

Школьники — в основном прикладные, реже системные программисты.

Основная задача «дошкольников» — набираться опыта и учиться.

Хотя при создании клуба ставилась достаточно ограниченная цель поддержки разработанного программного обеспечения, жизнь показала, что его возможности гораздо больше.

Компьютерный клуб является очень эффективной формой учебной работы. В табл. 1

Таблица 1

Классы	Средний балл			
	Общий	Мальчики	Девочки	Члены клуба
IX	4,5	4,5	4,4	5,0
X	4,3	4,08	4,4	5,0

приведены данные об успеваемости учеников школы по информатике.

Видно, что успеваемость членов клуба по информатике выше средней по школе. О более глубоких знаниях членов клуба по информатике и программированию свидетельствует и тематика работ, доложенных на школьной научной конференции, и итоги олимпиад по программированию, в которых принимали участие члены клуба. Их успехи во многом объясняются той атмосферой радостного общения, которая характерна для работы клуба и является, по нашему мнению, воплощением педагогики сотрудничества. Его члены и по окончании школы и даже института стремятся остаться в клубе. Престиж члена клуба определяется не возрастом, а знаниями, опытом и результатами. В клубе при решении задач на равных сотрудничают все его члены.

Таблица 2

Типы программных продуктов	Процент использования в учебном процессе
Обучающие программы	13,3
Тесты	26,6
Тренажеры	20,0
Моделирующие программы	13,3
Системы для решения задач	20,0
Игровые программы	6,6

С точки зрения производственной клуб оказался очень эффективным. В настоящее время 74 % компьютерных уроков проводятся с использованием программных продуктов, разработанных в нем: обучающих программ, тестов, тренажеров, моделирующих программ, систем для решения задач, игровых программ. По нашему мнению, именно разнообразие является одними из ключевых факторов успеха применения компьютерной технологии обучения (в табл. 2 приведено процентное соотношение использования различных типов программных продуктов, разработанных членами компьютерного клуба, в учебном процессе).

Члены клуба составляют ядро студенческих научно-производственных отрядов, которые специализируются на создании программного обеспечения для учебного процесса вуза, в частности они принимали участие в разработке программного обеспечения для курса общей физики.

Тем, кого заинтересовал наш опыт, сообщаем номер телефона ВЦ школы: 324-97-69.

С. ФАДЕЕВ

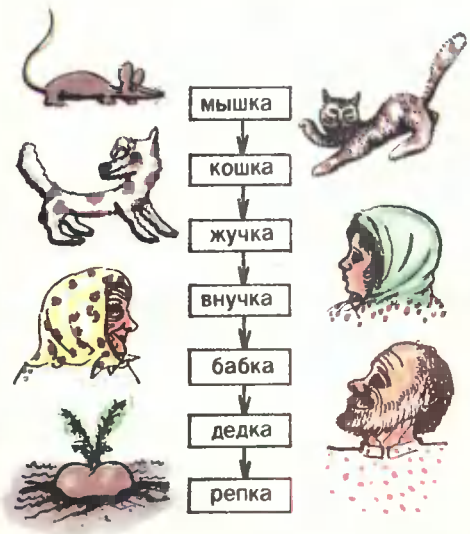
Сказка ложь, да в ней намек

Нет, пожалуй, человека, который бы не любил слушать в детстве сказки или же не рассказывал позднее их своим детям и внукам. Народные сказки живут веками, передаваясь от поколения к поколению. И мы даже не задумываемся, в чем же столь притягательная сила сказок. Раз услышав, мы помним их на протяжении всей жизни. Только ли в красоте волшебного сюжета, который, преломившись в детском сознании, оставляет в нем неизгладимый след? Думается, что дело здесь не только в этом. Ведь есть множество простых, «неказистых», на первый взгляд, сказок («Репка», «Колобок», «Курочка Ряба»), но любой из нас может воспроизвести их практически дословно.

Одним из факторов, способствующих лучшему запоминанию любого материала, является, как известно, его четкая, алгоритмическая структура. Если взглянуть с этой точки зрения на любую народную сказку, то можно без труда обнаружить, что в ее основе зачастую лежат линейные и циклические алгоритмы, часто включается в сказочное повествование и условие.

Указанные особенности народных сказок позволяют, на наш взгляд, как нельзя лучше использовать их при раскрытии темы «Алгоритмы и их свойства» практически в любой аудитории: взрослых, школьников и даже дошкольников!

Понятие линейного алгоритма вводится на основе цепочки, характерной для народной сказки «Репка» (рис. 1), после чего учащиеся



моментально (проверено на практике) вспоминают множество подобных цепочек. При таком подходе новый материал накладывается на уже хорошо известные учащимся примеры и благодаря этому прочно усваивается.

Не составляет большого труда для учащихся обнаружить в сказке и циклический алгоритм. Как правило, это центральная часть сказки. Например, попытки персонажей «Репки» вытянуть репку; разговоры девочки с печкой, молочной рекой с кисельными берегами, с яблоней («Гуси-лебеди»); встречи Колобка со зверями («Колобок») и т. п.

Несколько труднее обнаружить в сказке логическое условие, которое, как известно, лежит в основе разветвляющегося алгоритма. Но и с этой задачей учащиеся справляются после рассмотрения сказки «Гуси-лебеди» (рис. 2).

Большинство учащихся с удовольствием расписывают сказки в виде блок-схем, многие не ограничиваются простыми сказками, а стараются найти сложные сюжеты.

Для школьников важно также и то, что уже на уроках информатики они получают понятие о композиции литературного произведения. Некоторых эта тема настолько увлекает, что они сами начинают сочинять небольшие сказки на основе типовых блок-схем или же создают так называемые альтернативные сказки, т. е. при анализе логических условий в известных народных сказках выбирают ответ (решение), противоположный сказочному, развивая тем самым сюжет по своему усмотрению.

Таким образом, зерно новых знаний — понятие об алгоритмах — попав на подготовленную почку — сказки — дает довольно быстрые и бурные всходы. И потому тысячу раз прав был древний сказитель: «Сказка ложь, да в ней намек, добрым молодцам урок».

В. ОЧКОВ

Рассказ о трех Э (Eureka, этика, экономика)

«—...Теперь по арифметике... Берите доску. Какая следующая задача?

Петя плюет на доску и стирает рукавом. Учитель берет задачник и диктует:

— «Купец купил 138 арш. черного и синего сукна за 540 руб. Спрашивается: сколько аршин купил он того и другого, если синее стоило 5 руб. за аршин, а черное 3 руб.?» Повторите задачу.

Петя повторяет задачу и тотчас же, ни слова не говоря, начинает делить 540 на 138.

— Для чего же вы это делите? Постойте! Впрочем, так... продолжайте. Остаток получается? Здесь не может быть остатка. Дайте-ка я разделю!

Зиберов делит, получается 3 с остатком и быстро стирает.

«Странно...— думает он, ероша волосы и краснея.— Как же она решается? Гм!.. Это задача на неопределенные уравнения, а вовсе не арифметическая...»

Учитель глядит в ответы и видит 75 и 63.

«Гм!.. странно... Сложить 5 и 3, а потом делить 540 на 8? Так, что ли? Нет, не то».

— Решайте же!— говорит он Пете.

— Ну, чего думаешь? Задача-то ведь пустяковая!— говорит Удодов Пете.— Экий ты дурак, братец! Решите уж вы ему, Егор Алексеич.

Егор Алексеич берет в руки грифель и начинает решать. Он заикается, краснеет, бледнеет.

— Эта задача, собственно говоря, алгебраическая,— говорит он.— Ее с иксом и игреком решить можно. Впрочем, можно и так решить. Я, вот, разделил... понимаете? Теперь, вот, надо вычесть... понимаете? Или, вот что... Решите мне эту задачу сами к завтраму... Подумайте...

Петя ехидно улыбается. Удодов тоже улыбается. Оба они понимают замешательство учителя. Ученик VII класса еще пуще конфузится, встает и начинает ходит из угла в угол.

— И без алгебры решить можно,— говорит Удодов, протягивая руку к счетам и вздыхая.— Вот, извольте видеть...

Он щелкает на счетах, и у него получается 75 и 63, что и нужно было.

— Вот-с... по-нашему, по-неученому.»

Фрагментом из миниатюры А. П. Чехова «Репетитор» мы предваряем рассказ о среде программирования Eureka по трем причинам:

1. Описанная ситуация очень часто встречается и сейчас. Только люди стали протягивать руку не к счетам, а к персональному компьютеру.

2. Процесс решения задачи на ЭВМ в среде Eureka во многом похож на процесс решения задачи о купце и сукне Удодовым-отцом или на попытки решения этой задачи Зиберовым. Машине достаточно сообщить только условия задачи, а как там компьютер «прощелкает ее на счетах», будет делить или вычитать — это его личное дело. Главное, чтобы ответ сходился. А для его проверки Eureka имеет специальные, достаточно мощные средства.

3. И современная, и прошлого века (взгляните еще раз на рассказ А. П. Чехова) методика преподавания школьной (и не только школьной) математики, физики, химии и других дисциплин ориентирована, как правило, не на постановку задачи, а на решение уже кем-то сформулированных проблем. Это сразу ограничивает кругозор обучаемого некими шорами, мешающими посмотреть на проблему иначе, чем это предусмотрено методикой решения. Компьютеризация школы, базирующаяся на преподавании языков про-

граммирования (Бейсик, Паскаль, Рапира и др.) в качестве основ информатики, фактически ничего не меняет в этом не совсем нормальном положении вещей. Сре́ды общения с ЭВМ на естественном (почти) для человека языке, к которым относится Eureka, позволяют больше внимания уделять постановке проблемы, математическому моделированию реальных ситуаций.

Помня о том, что лучше один раз увидеть, чем сто раз услышать, мы помещаем на рис. 1 копию экрана дисплея с решением в среде Eureka вышеописанной задачи о купце и сукне. Заранее просим простить за то, что дальнейший текст будет густо пересыпан английскими терминами. Делать нечего: хотите быть «на ты» с вычислительной техникой — учите английский язык или по крайней мере запомните несколько десятков терминов, давно ставших международными. Когда-то кто-то у нас в азарте борьбы с космополитами обозвал кибернетику публичной девкой империализма. Вот теперь и мстит она нам за это, не желая разговаривать по-русски.

При работе в среде Eureka экран разбивается на 5 окон:

окно главного меню (Eureka: The Solver) с названиями восьми «блюд»: работа с файлами (File), ввод с клавиатуры и редактирование задачи (Edit), решение задачи (Solve), специальные команды (Commands), создание отчетов о решении задачи (Report), построение графиков (Graph), дополнительные возможности (Options) и работа с окнами дисплея (Window);

окно ввода с клавиатуры и редактирования (Edit). В нем в данный момент записаны условия задачи Удодова. Точка с запятой означает начало комментария (вспомним слово REM или апостроф в Бейсике), который может содержать и буквы кириллицы; в тексте же задачи допустимы только латинские и греческие буквы. Сама задача проста — это система линейных алгебраических (Зиберов был прав) уравнений с двумя неизвестными: black (метраж, вернее, аршинаж черного сукна) и blue (синего);

окно с решением задачи (Solution), куда машина записывает переменные (Variables) и их найденные значения (Values), удовлетворяющие условиям поставленной задачи. По умолчанию Eureka присваивает переменным статус вещественных с восьмью знаками в мантиссе, но пользователь может сюда внести изменения, оговорив, например, целочисленность или комплексность переменных. Работает Eureka и с индексными переменными, объединенными в массивы;

окно проверки (верификации — Verify) результатов решения задачи. По команде человека машина подставляет в формулы окна Edit значения переменных из окна Solution и отмечает разницу (difference) между левыми и правыми частями выражений, заодно распечатывая и исходные данные для их дополнительной проверки;

окно с отчетом (Report) о решении задачи, куда машина опять же по команде человека может записать дату и время решения задачи, имя и место хранения на диске файла с ее условием, само условие, решение задачи и его проверку (верификацию); отчет, кроме того, можно отпечатать на принтере или записать на диске. В нижней части экрана (см. рис. 1) отмечены «нагрузки» функциональных клавиш клавиатуры. Так, например, нажав F1, пользователь в появившемся на экране дисплея окне получит дополнительную информацию (помощь — Help) о работе в среде Eureka. После ее прочтения нужно нажать клавишу ESC (to escape — убегать), чтобы восстановить рабочие окна в их первоначальном виде.

А теперь подошло время пояснить термин «среда», стоящий перед фирменным названием Eureka и, возможно, несколько режущий слух нашим читателям. Условие задачи Удодова (см. рис. 1) находится в окружении команд программного продукта Eureka — см. верхний и нижний ряды экрана дисплея. Это же условие можно окружить другими командами, «погрузив» его в другую программную среду: в текстовый процессор (Лексикон, MultiEdit и др.), в инструментальную систему PCTOOLS, в более новую версию среды Eureka и т. д.

Просматривая файлы на диске, можно заметить, что они, как правило, имеют не только имя, но и «фамилию» типа «.exe», «.txt» и т. д.; она отмечает программную среду, в которой эти файлы можно использовать или в которой они были созданы. Файлы с задачами среды Eureka имеют фамилию-хвостик «.eka».

Если просмотреть пакет Eureka, то в нем можно увидеть много файлов с окончанием «.eka». Фирма-разработчик записала их на диск, преследуя две цели. Во-первых, просмотрев эти файлы, можно довольно быстро вникнуть в методику решения задач в среде Eureka (такой метод обучения используется и в нашей статье), а во-вторых, при решении конкретной задачи часто бывает достаточно вызвать с диска ее аналог (решение систем уравнений, одномерная и многомерная оптимизация и т. д.) и, отредактировав его, «прогнать» на машине.

В окнах Report и Verify на рис. 1 умести-

1


```
*****************************************************
        Eureka: The Solver, Version 1.0
        Name of input file : C:\OKRUG.EKA
*****************************************************
;  Пример          Задача о поиске координат окружности
;(cx,cy и R), проходящей через три точки на плоскости.
x1=26 : y1=4;                  Координаты первой точки.
(x1-cx)^2+(y1-cy)^2=R^2; Вытекает из теоремы Пифагора.
x2=9  : y2=21;                 Координаты второй точки.
(x2-cx)^2+(y2-cy)^2=R^2;     Второе уравнение системы.
x3=17 : y3=17;                 Координаты третьей точки.
(x3-cx)^2+(y3-cy)^2=R^2;     Третье уравнение системы.
;       Ход решения системы нелинейных алгебраических
;.;.;   уравнений сильно зависит от начальных заданных
;         значений неизвестных.  Пусть поэтому будет
cx:=(x1+x2+x3)/3;              ордината центра окружности,
cy:=(y1+y2+y3)/3;                          ее абсцисса и
R:=abs(x1+x2)/2 ;    радиус в первом приближении. Все!
*****************************************************
Solution (Решение):
      Variables     Values
      cx          =      2.0000000
      cy          =     -3.0000000
      R           =      25.000000
      x1          =      26.000000
... (Далее повтор исходных данных)
     Maximum error is 2.2737368e-13
Warning: root or log of negative number. (Предупреждение: брался корень или логарифм от отрицательн. числа.)
Evaluation of formulas (Проверка формул):
      Formulas            Values (Формулы    Значения)
... (x1, y1)
      (x1-cx)^2+(y... =      625.00000
      R^2             =      625.00000
        difference (error) = 2.2737368e-13
... (x2, y2)
      (x2-cx)^2+(y... =      625.00000
      R^2             =      625.00000
        difference =      .00000000
... (x3, y3)
      (x3-cx)^2+(y... =      625.00000
      R^2             =      625.00000
        difference =      .00000000
      cx              =      2.0000000
      cy              =     -3.0000000
      R               =      25.000000
     Maximum error is 2.2737368e-13
*****************************************************
```

3

2


```
*****************************************************
        Eureka: The Solver, Version 1.0
     Name of input file: C:\EKA\TEMPO\VEDRO.EKA
*****************************************************
;  Пример          Задача о проектировании конусообразного
;                        пожарного ведра с максимальным объемом.
D = 1;Диаметр круглой заготовки, из которой вырез. сектор.
0 < a < 360;                Диапазон изменения угла вырезки.
3.14*d = 3.14*D*(360-a)/360;  Длина окружности основания
;ведра-конуса (d) равно длине дуги заготовки с вырезкой.
(D/2)^2 = h^2 + (d/2)^2;       Радиус заготовки, высота и
;              радиус основания ведра-конуса - это стороны
;                              прямоугольного треугольника.
V(a) = 1/3*3.14*d^2/4*h;                  Объем ведра-конуса.
V = V(a);                          Анализируемая функция.
$max(V);  Установка на поиск максимального объема ведра.
tan(3.14*A/180) = d/2/h;  Подсчет угла раскрытия конуса.
*****************************************************
Solution (Решение):
Variables     Values (Переменные   Значения)
 a     =     66.061930 (Угол вырезки в заготовке, градусы)
 A     =     54.763180 (Угол раскрытия конуса, градусы)
 D     =     1.0000000 (Диаметр заготовки)
 d     =     .81649464 (Диаметр основания ведра-конуса)
 h     =     .28867651 (Высота ведра-конуса)
 V     =     .050357773 (Объем ведра-конуса)
Confidence level =    99.4% (Доверительный уровень)
All constraints satisfied. (Все ограничения выполнены)
*****************************************************
```

4

лась только часть информации о решении задачи Удодова. Возможность полного ее просмотра также предусмотрена (активизацией через главное меню нужного окна с последующим его раскрытием (F5 — Zoom)). В данный момент активизировано (подготовлено к раскрытию, к редактированию) окно Report, о чем свидетельствует его двойная рамка. Текст отчета при этом можно подправить, сделать, например, более компактными, а уже потом отпечатать на бумаге. Так, кстати, и было сделано с отчетами (рис. 3—6) о решении четырех задач, которыми будут проиллюстрированы некоторые особенности работы в среде Eureka.

Каждый из этих отчетов разбит рядами звездочек на три зоны: зона заголовка, зона условия задачи и зона ее решения.

Мы не будем расшифровывать термины «активизировать», «войти в окно дисплея» (или в программную среду), так как цель статьи — на описание техники общения с ПЭВМ (она, как правило, одна при работе и с языком программирования, и с текстовым процессором, и с электронными таблицами, и с базой данных), а пояснение некоторых особенностей решения задач в среде Eureka.

Технология решения новой задачи такова. Включается компьютер, запускается программа с именем «eureka.exe» в пакете EKA. После этого на экране дисплея появляются окна среды Eureka (рис. 2). Нижние четыре окна будут пустые, а первое «блюдо» меню (File) выделено фоном. Этот фон с помощью курсорных клавиш можно передвигать вдоль меню. Вход в нужный режим осуществляется нажатием клавиши Enter (ввод). При этом может появиться новое меню (подменю), как это показано на рис. 2 для режима работы с файлами.

Среда Eureka предоставляет пользователю следующие услуги через подменю режима File:

загрузка (Load) с диска в окно Edit старой задачи, записанной на диске;

очистка окон перед решением новой (New) задачи;

сохранение (Save) задачи на диске; так специалист-прикладник может расширять библиотеку своих программ, создавая, как принято говорить, автоматизированное рабочее место (АРМ) на базе персонального компьютера;

запись (Write to) задачи с новым именем на диск. К этому режиму прибегают, когда хотят сохранить новую модификацию задачи либо для записи безымянной совсем новой (NONAME — см. рис. 1);

просмотр оглавлений (Directory) дисков;

смена (Change dir) текущего оглавления диска;

переименование (Rename) файла на диске;

временный выход из среды Eureka в операционную систему (OS shell) с обратным возвращением по команде Exit;

окончательный выход (Quit) из среды Eureka.

Подобные подменю есть у всех режимов главного меню, кроме Edit и Solve. Эти подменю в свою очередь могут иметь свои собственные подменю следующего уровня и т. д. Сколько вопросов может задать Eureka человеку, никто не знает: разработчик уже забыл, а пользователь еще не запомнил. Да такого запоминания и не требуется. Сидя за дисплеем компьютера и действуя клавишами или «мышью», можно быстро найти то, что в данный момент требуется для решения поставленной задачи.

Выбор режима Edit (перемещение пользователем фона на это слово в главном меню и нажатие клавиши Enter) приводит не к появлению нового подменю, а к активизации окна Edit, где после этого можно записать условие задачи. После того как задача будет сформулирована, нажимается клавиша Esc для входа в главное меню, где дается команда Solve, заполняющая решением задачи окно Solution. Подобным образом можно заполнить окна Verify и Report и получить картинку, показанную на рис. 1. Выбор нужного режима из меню можно осуществить и другим способом, нажимая одновременно управляющую клавишу (Ctrl) и клавишу с заглавной буквой названия режима. Если у компьютера есть «мышь», то это еще упрощает работу: для выбора нужного режима следует поелозить «мышью» по столу (установить курсор «мыши» на название режима) и нажать левую кнопку «мыши».

Пример 1. Найти координаты центра и радиус окружности, проходящий через три точки на плоскости с координатами (x;y): 26;4, 9;21 и 17;17.

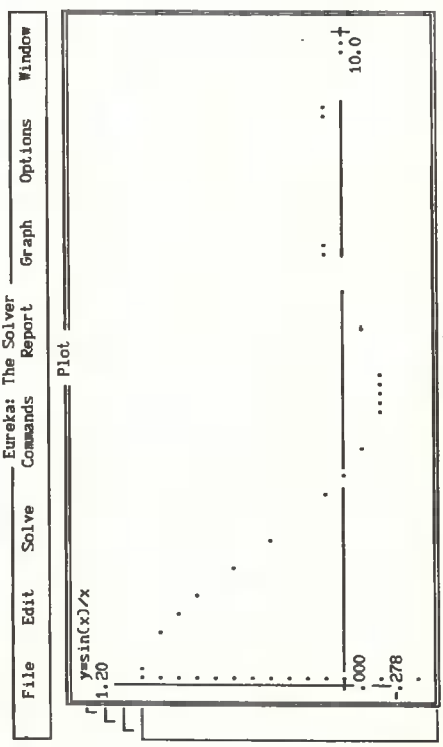
На рис. 3 представлен отчет о решении этой задачи. Она сводится к поиску корня системы трех нелинейных алгебраических уравнений, полученных приложением теоремы Пифагора (сумма квадратов катетов прямоугольного треугольника равна квадрату гипотенузы) к координатам исходных точек и искомого центра окружности. Эта система имеет два решения, где значения радиусов равны по модулю, но отличаются по знаку. Поэтому очень важно в условии задачи записать первое приближение к решению, опираясь на которое машина и быстрее, и точнее найдет ответ. В против-

```
*************************************************
      Eureka:  The Solver, Version 1.0
    Name of input file: C:\EKA\TRANS.EKA
*************************************************
;  Пример                    Транспортная задача.
;     В городе есть два склада муки и две пекарни.
;Ежедневно с 1-го склада нужно вывезти 50 т муки,
M1 + M2 = 50
;                                  а со 2-го - 70 т.
M3 + M4 = 70
;                 При этом 1-я пекарня получает 40 т,
M1 + M3 = 40
;                                  а 2-я - 80 т.
M2 + M4 = 80
;                  С пекарен на склады муку не везут:
M1 > 0 : M2 > 0 : M3 > 0 : M4 >0
;    Стоимость перевозки муки (руб/т)
;                 Пекарня 1   Пекарня 2
;    Склад 1        1.2          1.6
;    Склад 2        0.8          1.0
;     Целевую функцию (затраты на перевозку муки)
zatraty = 1.2* M1 + 1.6* M2 + 0.8* M3 + 1* M4
$min(zatraty);                  нужно минимизировать.
*************************************************
 Solution (Решение):
 Variables     Values (Переменная  Значение)
 M1 = 40.000000 (С 1-го склада на 1-ю пекарню)
 M2 = 10.000000 (С 1-го склада на 2-ю пекарню)
 M3 = .00000000 (Со 2-го склада на 1-ю пекарню)
 M4 = 70.000000 (Со 2-го склада на 2-ю пекарню)
 zatraty = 134.00000 (рубля в сутки)
 Confidence level =   99.4%
 All constraints satisfied.
*************************************************
```

```
*************************************************
      Eureka: The Solver, Version 1.0
    Name of input file: B:\LINPROG.EKA
*************************************************
;  Пример          Задача планирования производства.
;На фабрике делают (больше нуля) стулья двух типов
    stul1 > 0          :          stul2 > 0.
;     Далее таблица с линейными неравенствами:
;                      Идет на стул 1 Стул 2   Ресурс
;       Доски (м2)                 2      4        440
2 * stul1 +   4 * stul2 <= 440
;       Обивочная ткань (м2)     .5    .25         65
.5 * stul1 + .25 * stul2 <=  65
;       Трудозатраты (чел.часы) 2      2.5        320
2 * stul1 + 2.5 * stul2 <= 320
;       Стоимость (руб)          8     12         ---
8 * stul1  + 12 * stul2 = val
;    Фабрика не частная, а государственная. Поэтому
;      в условиях командно-административной системы
;       (диктат производителя, фондирование средств
;     производства - см. вышеописанные неравенства)
;  они, как правило, старается гнать вал в рублях.
$max(val);       Так это записывается в среде Eureka
*************************************************
Solution (Решение):
 Variables     Values (Переменные  Значения)
val         =      1440.000 (Стоимость стульев,руб.)
stul1       =      59.99999 (Выпуск стульев 1 типа)
stul2       =      80.00000 (Выпуск стульев 2 типа)
Confidence level =   95.1%
All constraints satisfied.
Evaluation of formulas (Проверка формул):
Formulas          Values (Формула  Значение)
2*stul1+4*st... =      440.00000 (Реальный расход)
440             =      440.00000 (Фонды)
difference = -2.5466801e-07 (Опилки и стружка)
.5*stul1+.25... =      50.000000 (Реальный расход)
65              =      65.000000 (Фонды)
difference = -15.000000 (Остаток - ресурс фабрики)
2*stul1+2.5*... =      320.00000
320             =      320.00000
difference = -7.321270e-07(Людей заняли полностью)
  ...           Далее все тривиально и неинтересно.
*************************************************
```



```
Edit: C:\EKA\help.eka          * Line 19    Col 42    1,756 Free   32
     Функции EUREKA (HELP FILE.)
   abs(x) Absolute value          atan2(y,x) Arc Tangent
   cos(x) Cosine               cosh(x) Hyperbolic Cosine
   deriv(f(x),x) Derivative       exp(x) Exponential
   fact(n) Factorial of constant      floor(x) Floor
   fval(rate,time,pres_value,payment)  Future Value
   im(x) Imaginary part
   integ(f(x),x,lower_bound,upper_bound) Integral
   ln(x) Natural Logarithm  log10(x)  Logarithm (base 10)
   msqrt(x) Minus Square Root
   ncum(x) Normal Cumulative Distribution
   payat(rate,time,pres_value,future_value) Payment
   polar(x,y) Polar Coordinate Conversion
   poly(x,...) Polynomial          pos(x) Positive Part
   pval(rate,time,payment,future_value)  Present Value
   re(x) Real Part              sgn(x) Sign     sin(x) Sine
   sinh(x) Hyperbolic Sine          sqrt(x) Square Root
   sum(f(x),x,first,last) Sum          tan(x) Tangent
   tanh(x) Hyperbolic Tangent
1Help  2Save  3      4      5      6    , 7Search 8      9      10Quit
```

ном случае ЭВМ может «метаться» между возможными решениями, не склоняясь ни к одному из них (принцип буриданова осла). Правые и левые части уравнений в условии задачи разделяются либо знаком =, либо знаком :=. Решая поставленную задачу, машина в первую очередь старается удовлетворить выражения со знаком =, а потом уже по возможности (принцип остаточности) и те, какие разделены знаком :=.

Прогонку (Solve) задачи с двумя и более решениями, из которых удовлетворяет только одно, всегда следует завершать проверкой полученного решения. Это и было сделано при решении задачи об окружности на плоскости: отчет на рис. 3 заканчивается подсчетом отклонений левых и правых частей уравнений.

Пример можно расширить, заставив машину работать не с тремя, а с четырьмя и более точками на плоскости. ЭВМ при этом найдет (или по крайней мере постарается найти) координаты окружности, наилучшим образом вписывающейся в заданные точки. Такая особенность среды Eureka — и достоинство, и недостаток. С одной стороны, она позволяет решать не совсем корректно поставленные задачи, по которым нужно найти какое-нибудь более-менее удовлетворительное решение (например, сглаживание табличной зависимости аналитической функцией). С другой стороны, эта особенность может обернуться тем, что машина выдаст совсем не то, что от нее ожидают.

Можно отметить, что систему трех нелинейных уравнений, описывающих задачу об окружности, можно свести к системе линейных уравнений за счет несложных преобразований, отняв, например, первое нелинейное уравнение от второго и т. д. Но Eureka тем и хороша, что она позволяет решать задачи в их, так сказать, первозданном математическом описании без упрощений, которые очень часто затрудняют физическое понимание сути задачи или даже выхолащивают его неизбежными в этом случае допущениями.

Пример 2. Требуется из круглой жестянки изготовить пожарное ведро максимального объема. Технология изготовления ведра такая: из заготовки вырезается и выкидывается сектор с углом a в вершине (неизвестное), затем полученная таким образом выкройка сворачивается в конус, а шов проваривается (клеится, паяется). Наполненное пожарное ведро нельзя поставить на пол. Оно и делается в виде конуса, чтобы исключить использование не по прямому назначению.

Поставленную оптимизационную задачу можно, конечно, решить аналитически и без компьютера. Для этого необходимо вывести зависимость, связывающую объем ведра-конуса (функция) и угол сектора-вырезки (аргумент). Затем нужно взять первую производную от этого выражения и найти ее корень. Не обойтись тут и без второй производной, если нужно удостовериться, что найденная точка (угол вырезки) отмечает максимум, а не перегиб. Жестянщик, которому поручат сделать пожарное ведро, скорее всего, не знаком с теорией математического анализа функции одной действительной переменной. Но в среде Eureka поставленная задача ему окажется по плечу — см. рис. 4.

Пусть диаметр заготовки равен одному метру. Этого в условии задачи можно было бы и не оговаривать, так как, во-первых, значение оптимального угла вырезки не зависит от диаметра заготовки, а во-вторых, переменной D по умолчанию машина сама бы присвоила единичное значение. Это одна из особенностей среды Eureka, отличающая ее от языка Бейсик, где переменные по умолчанию обнуляются.

Далее исходя из здравого смысла устанавливается допустимый диапазон изменения угла вырезки, чтобы машина не «залезла» в отрицательную область или в область больших углов (вспомним синусоиду с бесчисленным множеством корней, максимумов и минимумов). Два последующих выражения вытекают из несложной геометрии круга и конуса: дуга выкройки становится окружностью основания конуса, а высота конуса, радиус его основания и радиус заготовки связаны теоремой Пифагора. Число π (3,14) в первом выражении можно, конечно, сократить. Но тогда геометрический смысл этого выражения не проступал бы так ясно. Общение с компьютером вырабатывает у человека так называемый KISS-принцип: Keep it simple, stupid — делай это проще, дурачок!

Если определены параметры конуса (диаметр основания из первого выражения, а высота — из второго), то можно рассчитать его объем $V=V(a)$ и поручить машине максимизировать (max) эту величину, варьируя значением угла сектора-вырезки.

Из последнего выражения в условиях задачи (оценка угла в днище ведра через тангенс) видно, во-первых, что Eureka оперирует углами в радианах, а во-вторых, что эта программная среда, конечно, пропускает смысловые ошибки пользователя: угол раскрытия конуса на самом деле будет в два раза больше, не 54.76, а 109.52 градуса. Максимизация пожарного ведра превращает его в тазик.

Два последующих примера тоже из об-

ласти оптимизации, но связаны с более серьезными проблемами, решение которых в какой-то мере поможет владельцу окупить сумасшедшие затраты (50—100 тыс. рублей) на приобретение персонального компьютера. Программное обеспечение в этих затратах не фигурирует, так как его у нас, как правило, не покупают, а друг у друга переписывают без разрешения разработчика. Причин на то много. Во-первых (финансовая сторона проблемы), западные фирмы свои даже «мягкие» продукты (software) продают только за твердую валюту (деньги), а не за рубли. А во-вторых (моральная сторона проблемы), воровство матобеспечения на фоне откровенной спекуляции персональными компьютерами в государственно-кооперативном масштабе выглядит просто детской шалостью.

Пример 3. Транспортная задача, связанная с оптимизацией перевозки грузов. В городе N имеются два продовольственных склада и две пекарни. Необходимо ежедневно с первого склада вывозить 50, а со второго 70 тонн муки. Первая пекарня при этом получает 40, а вторая — 80 тонн. Спрашивается: как нужно спланировать работу транспорта, если известны затраты на перевозку тонны муки со складов на пекарни и хочется на этом деле сэкономить деньги?

На рис. 5 представлен отчет о решении этой задачи в среде Eureka: минимум затрат на перевозку муки составляет 134 рубля в сутки. Как ни странно, но на первую пекарню со второго склада мука не завозится (M3=0), хотя такие перевозки самые дешевые — 80 копеек за тонну. Для спасения горящего месячного, квартального или годового плана этим парадоксом может воспользоваться плановик автокомбината, осуществляющего перевозку муки. Для этого необходимо слово min в условиях транспортной задачи заменить на слово max. В этом случае затраты на перевозку увеличатся до 142 рублей в сутки, хотя внеше все будет выглядеть вполне прилично: M1=0, M2=50, M3=40, M4=30 — на первую пекарню мука завозится только со второго склада. В условиях самоедской экономики с ее девизом «План любой ценой!» плановик автокомбината, оправдывая и свое название, и свое существование, может «похимичить» еще более серьезно, снимая табу типа M1>0 и благословляя тем самым встречные перевозки. Среда Eureka этого не запрещает: деньги не пахнут, а вычислительные машины не краснеют.

Еще одна задача из области так называемой плановой экономики.

Пример 4. Мебельной фабрике вышестоящие планирующие органы поручили выпускать стулья двух типов — ценою по 8 и 12 рублей. Под этот госзаказ выделены материальные и людские ресурсы: 440 погонных метров досок, 65 кв. м обивочной ткани, 320 человеко-часов. Известно, сколько досок, ткани и трудов идет на изготовление каждого стула. Спрашивается: как нужно спланировать производство стульев, чтобы... А что «чтобы»?! Что является целью производства?! Фабрике, конечно, выгодно сделать стульев ценою поболее. Это и отображено в отчете о решении задачи о стульях (рис. 6). Из него видно, что фабрика, заботясь о вале в рублях, должна сделать 60 стульев по цене 8 рублей и 80 по 12 рублей. При этом все доски израсходуются и полностью будут заняты люди, но останется 15 кв. м. обивочной ткани, которые можно будет пустить, например, на лозунг «Экономика должна быть экономной».

Среда Eureka позволяет, как и в предыдущем примере с перевозками муки, немного «поиграть» с задачей. Исправив, к примеру, целевую функцию,

stul1+stul2=val,

можно скорректировать производство так, чтобы хоть немного снизить остроту мебельного дефицита: те же самые ресурсы, оказывается, позволяют сфабриковать не 140, а 150 стульев: 110 дешевых и 40 дорогих. Но тут-то и выявится в чистом виде противоречие нашего способа производства: выпуск стульев в большем количестве невыгоден фабрике. Во-первых, вал упадет с 1440 до 1360 рублей, а во-вторых, люди на фабрике будут не полностью заняты — останется резерв в 65 человеко-часов. Вот так-то.

Примеры 3 и 4 относятся к области линейного программирования. Это, конечно, совпадение, но факт остается фактом: основы этой дисциплины закладывались Л. В. Канторовичем в тот трагический период нашей истории (30-е гг. XX столетия), когда рыночный механизм экономики (НЭП) силовыми приемами заменялся на командно-административный. Почти во всех примерах из учебников по линейному программированию, связанных с планированием производства, принимается за аксиому, что потребитель «проглотит» все, что ему предложит промышленность. Все целевые функции при этом, как правило, учитывают интересы только производителя, будь то план в рублях, в штуках или в тоннах, себестоимость продукции или экономия ресурсов.

Но вернемся к основной теме нашего рассказа.

На рис. 7 представлены функции среды

Eureka, «выуженные» из файла help.eka с помощью редактора инструментальной системы The Norton Commander (еще один западный программный продукт, который, как и Eureka, распространяется у нас пиратским способом по вышеописанным причинам). Кроме обычных функций (синусы, косинусы, логарифмы и др.), на рис. 7 можно увидеть и экзотические: взятие производной (deriv) и интеграла (integ), подсчет нормального распределения (ncum), роста вкладов в банке (fval, paymt, pval) и суммы ряда (sum).

Такой набор встроенных функций позволяет решать в среде Eureka разнообразные задачи. Но при этом не следует обольщаться: более-менее серьезная проблема, выливающаяся в десяток-другой алгебраических, дифференциальных или интегральных уравнений, часто оказывается среде Eureka «не по зубам». В таких ситуациях она расходует лимит времени, отведенный на решение задачи, и, как бы извиняясь, выдает какое-то приближенное решение. После его проверки человек должен сам решить, что с этим вариантом решения делать дальше.

Успех решения задачи очень сильно зависит от начального задания условий (опорная точка — взглянем еще раз на пример 1), которые может знать только специалист по решаемой проблеме. Подсказками он будет как бы подталкивать машину к правильному решению. Если традиционные языки программирования создавались программистами для собственного, так сказать, употребления, то среда Eureka — это в первую очередь инструмент прикладника-непрограммиста.

Еще одна «экологическая ниша» для среды Eureka и других ей подобных — сфера обучения (вспомним о чеховском начале нашего рассказа). Здесь очень пригодятся графические возможности. На рис. 8 показано еще одно окно (Plot), предназначенное для построения различных графических объектов с использованием встроенных функций или выражений из окна Edit. Так можно намного лучше:

а) понять сущность решаемой задачи (обучение!);

б) подобрать приемлемые начальные условия (решение прикладных задач!);

в) красиво оформить отчет о проделанной расчетной работе и т. д. и т. п.

А. СВЕРДЛОВ

Моделирование многомерных процессов

Любое современное производство не может обойтись без строгого контроля за ходом составляющих его технологических процессов. Нужда в подобном контроле возникает всякий раз, когда необходимо добиться совместного слаженного действия многих устройств, например при организации работы ЭВМ в многопрограммном режиме, при обеспечении бесперебойного функционирования автоматических линий и т. д. При проектировании таких систем весьма полезно предварительно смоделировать их на ЭВМ, для чего созданы специальные алгоритмические языки имитационного моделирования (наиболее известные из них — Симула и JPSS). Однако, чтобы воспользоваться этими языками, нужно, во-первых, знать их, а во-вторых, иметь соответствующий транслятор. Попытки же программировать задачи такого типа в «лоб» на универсальных алгоритмических языках наталкиваются на значительные трудности из-за чрезвычайной громоздкости описания и в силу отсутствия в этих языках стандартных средств организации взаимодействия переменных, связывающего их в единое целое.

Оказалось, однако, что эти трудности преодолимы. Ниже предлагается простой алгоритм, применяя который читатель сможет программировать разнообразные динамические задачи, используя доступные универсальные алгоритмические языки типа Фортрана или Паскаля.

Идею алгоритма продемонстрируем на примере так называемой задачи n станков.

Пусть в цехе, куда поступают на обработку детали, установлено n типов станков. На них можно обрабатывать k различных видов деталей. Деталь каждого вида должна пройти определенный технологический цикл обработки, который в дальнейшем будет называться ее маршрутом. Маршрут детали каждого вида задается в виде последовательности номеров станков, упорядоченных

в порядке обработки на них детали. Ноль в этом списке означает окончание обработки. Например, последовательность 4, 5, 10, 1, 0 показывает, что полный технологический цикл обработки детали до получения из нее готового изделия включает в себя ее последовательную обработку на четвертом, пятом, десятом, первом станках.

Для простоты примем, что время обработки детали на станке не зависит от вида детали и полностью определяется типом станка.

Производственный процесс теперь можно описать следующим образом. В течение дня в некоторые случайные моменты времени в цех поступают детали на обработку в соответствии со своими маршрутами, которые определяются видом детали. Если очередной станок, указанный в маршрутном списке детали, свободен, то деталь начинает на нем обрабатываться, в противном случае она поступает в очередь на обработку к этому станку.

А теперь построим математическую модель процесса функционирования цеха, которая позволит определить его важнейшие характеристики.

Будем считать, что маршруты всех деталей объединены в маршрутную матрицу $M=(m_{ij})$, $i=1,2,...,k$, $j=1,2,...,l$; здесь k — число видов деталей, l — длина самого длинного (по числу станков) маршрута. Примем также, что задан вектор $T=(t_j)$, $j=1,...,n$, где t_j — время обработки детали на j-м станке. Время будем считать изменяющимся с шагом h.

Для построения алгоритма используются меняющиеся со временем вспомогательные матрица $A(t)$ и вектор $B(t)$.

Элементы матрицы $A(t)=(a_{ij}(t))$, $i=1,2,...,N$, $j=1,2,...,n$, где N — общее количество поступающих в цех в течение дня деталей всех видов, определяются следующим образом:

$$a_{ij}(t)=\begin{cases} 0, \text{ если } i\text{-я деталь не обрабатывается и не находится в очереди на обработку к станку } j \text{ в момент времени } t; \\ p, \text{ если } i\text{-я деталь в момент времени } t \text{ стоит } p\text{-й в очереди на обработку к } j\text{-му станку; при } p=1 \text{ деталь обрабатывается на станке.} \end{cases}$$

Здесь следует сделать одно небольшое пояснение. В течение всего маршрута обработки за каждой деталью закреплен номер, равный порядковому номеру ее поступления в цех. Этот номер как раз и совпадает с номером строки матрицы A. Не путайте его с числом, идентифицирующим вид детали! Последнее «сопровождает» деталь, благодаря чему ЭВМ в любой момент может установить ее вид.

Элементы $b_j(t)$, $j=1,...,N$, вектора $B(t)$ равны

$$b_j(t)=\begin{cases} T_{ok}, \text{ если в столбце } j \text{ матрицы A есть элемент } a_{i_0 j}(t)=1; \\ 0, \text{ если в столбце } j \text{ матрицы A такого элемента нет.} \end{cases}$$

Здесь T_{ok} — момент окончания обработки детали с номером i_0 на j-м станке.

Иначе говоря, j-й столбец матрицы A изображает текущее состояние очереди деталей на обработку к j-му станку. Для детали, которая в данный момент времени уже обрабатывается (первая в очереди), значение элемента $b_j(t)$ равно моменту окончания обработки. Наглядно это изображено на рисунке.

При помощи матрицы $A(t)$ устанавливается взаимосвязь между переменными, относящимися к станкам, и переменными, относящимися к деталям.

Из сказанного ясно, что текущее состояние цеха в момент времени $t=t_0$ полностью определяется множеством $\{A(t_0), B(t_0)\}$.

Предположим, что это множество для момента времени $t=t_0$ нами уже построено. Укажем алгоритм перехода от него к множеству $\{A(t_1), B(t_1)\}$, описывающему состояние цеха в следующий момент времени $t_1=t_0+h$.

А. Все столбцы матрицы $A(t_0)$ просматриваются последовательно, элементы каждого столбца преобразуются по следующим правилам.

1. $j=1$ (начало просмотра).

2. Если $b_j(t_0)\neq t_0$, то перейти к пункту 8 (станок к моменту времени t_0 еще не освободился).

3. Запомнить индекс i_0 элемента $a_{i_0 j}(t_0)$, равного 1 (обработка детали на станке j за-

кончена к моменту времени t_0 и деталь в дальнейшем должна поступить на обработку к следующему станку своего маршрута или покинуть цех).

4. Положить $a_{i_0 j}(t_0+h)=0$, $b_j(t_0+h)=0$. (Деталь уже обработана на станке, и мы вычеркиваем ее из очереди.)

5. Найти в j-м столбце элемент $a_{i_1 j}(t_0)=2$; если такого элемента в столбце нет, то перейти к пункту 8 (в очереди к j-му станку ищется деталь, которая в момент времени t_0+h должна поступить на обработку).

6. Положить $b_j(t_0+h)=\mathrm{T}_{ok}$, где $\mathrm{T}_{ok}=t_0+t_j$ (время окончания обработки детали T_{ok} отмечается в момент поступления ее на станок).

7. Для всех $a_{ij}(t_0)>0$ положить $a_{ij}(t_0+h)=a_{ij}(t_0)-1$ (после того как одна деталь покинула станок, вся очередь сдвигается на единицу).

8. Положить j равным $j+1$ и, если $j\leqslant n$, перейти к пункту 2 (если не все станки просмотрены, переходим к следующему столбцу матрицы А).

9. Определить, должна ли очередная деталь к моменту времени t_0+h поступить в цех на обработку; если да, то присвоить ей порядковый номер и задать ее вид (периодичность поступления деталей в цех определяется спецификой производственного процесса и может быть промоделирована с помощью генератора случайных чисел).

Б. Таким образом сформировано множество номеров деталей $i_д=\{i_k\}$, $k=1,2,\ldots,r$ (см. п. 3 ч. А алгоритма), обработка которых завершена на станках в момент времени t_0. Эти детали, включая деталь, поступившую в цех, если такая есть, должны пройти обработку на следующем станке своего маршрута или покинуть цех. Пусть $j_д=\{j_k\}$ — множество номеров станков, на которых они должны обрабатываться. Передвижение деталей по маршруту и установка их в очередь производятся следующим образом:

1. $k=1$.

2. Если $j_k=0$, то технологический цикл обработки детали завершен. Деталь должна покинуть цех. Перейти к пункту 5.

3. Положить $a_{i_k j_k}(t_0+h)=\max\limits_i\{a_{ij_k}\times(t_0+h)\}+1$ (деталь ставится в конец очереди к станку j_k).

4. Если $a_{i_k j_k}(t_0+h)=1$, то положить $b_{j_k}(t_0+h)=\mathrm{T}_{ok}$, где $\mathrm{T}_{ok}=t_0+t_{j_k}$ (в случае когда станок свободен и деталь сразу начинает на нем обрабатываться, необходимо отметить момент окончания обработки).

5. Положить k равным $k+1$ и, если $k\leqslant r$, перейти к пункту 2 (проверка — все ли детали передвинуты по маршруту).

6. Если рабочий период функционирования цеха не закончен, то перейти к п. 1 ч. А.

7. Конец.

Суть алгоритма наглядно можно проиллюстрировать следующим образом. Совокупность значений элементов матрицы $\mathrm{A}(t_0)$ и вектора $\mathrm{B}(t_0)$ как бы задает «фотографию» цеха в момент времени t_0. Пользуясь правилами преобразования переменных, мы получаем «фотографию» цеха в следующий момент времени t_0+h и т. д. Таким образом, начав с начального состояния, целиком состоящего из нулей, и задав исходные данные, структуру входного потока и т. д., мы можем промоделировать работу цеха на любом отрезке времени.

Покажем теперь, как можно найти некоторые характеристики модели. Предположим, что мы хотим выяснить, сколько всего времени некоторая деталь находилась в очередях к станкам своего маршрута, ожидая обработки. Для определения этой величины обозначим через t_{ij}^1 момент времени, в который i-я деталь заняла очередь к станку j, а через t_{ij}^2 — момент начала ее обработки. Тогда очевидно, что искомое время Т ожидания детали в очередях $\mathrm{T}_i=\sum\limits_j (t_{ij}^2-t_{ij}^1)$.

Суммирование проводится по всем станкам технологического маршрута данной детали.

В заключение отметим, что задача n станков является лишь простейшим примером приложения описанного алгоритма. Гораздо более многообещающей выглядит возможность ее применения для управления реальными системами. Но тут пусть эстафету подхватит читатель.

И. СОКОЛ, С. ШИП
г. Одесса

Компьютер — помощник в музыкальном обучении

Вопрос о применении компьютеров в музыкальном образовании вызывает, как мы убедились, скорее недоумение и недовольство, чем понимание и одобрение. Чем это объяснить? Музыканты-педагоги неприязненно относятся к этому вопросу, вероятно, потому, что его практическое решение требует от них дополнительных затрат энергии и времени на приобретение новых знаний и педагогических умений. Кроме того, представители музыкальной педагогики ощущают себя «лириками», чей «долг» — сопротивляться влияниям точных наук и техники на процессы обучения и воспитания.

Что же касается программистов и педагогов-информатиков, то они тоже без особого энтузиазма встречают попытки обсуждения этого вопроса. Им представляется очевидным, что компьютеры должны быть как можно скорее внедрены во все сферы педагогической практики, в том числе в музыкальную педагогику. И они ждут от специалистов в разных учебных отраслях конкретных предложений по разработке программ для ЭВМ.

На наш взгляд, сегодня одинаково необходимы как теоретико-методологические исследования возможностей применения ЭВМ в музыкальном образовании, так и попытки получения начального практического опыта в данном новом деле. Первое без второго неполно и мертво, а второе без первого бессмысленно. Очевидно, что рано или поздно учащиеся и их педагоги стихийно подойдут к программированию музыкальных и околомузыкальных информационных процессов. Лучше, если это произойдет не поздно и не стихийно, и если музыкальная педагогика будет готова к эффективному использованию нового обучающего средства.

Итак, ЭВМ — устройство, способное производить громоздкие вычисления, решать логические задачи, если есть соответствующая программа, распознавать пространственные и звуковые формы, контролировать некоторые технологические процессы и управлять ими. Как это можно использовать в «околомузыкальных» сферах?

Музыкантов-теоретиков компьютер заинтересовал прежде всего как моделирующее и информационное поисковое устройство. Некоторые композиторы, экспериментирующие с техникой письма, обратились к счетно-решающим способностям ЭВМ. Наибольший резонанс тогда вызвали «компьютерные композиции», которые, строго говоря, не являются произведениями художественного творчества в полном и точном смысле этого понятия. Их следует рассматривать как квазимузыкальные феномены.

Достойные внимания образцы такого рода описаны в книге Р. Х. Зарипова «Музыка и кибернетика». Среди них, например, знаменитая сюита «Иллиак», созданная композитором Л. Хиллером и американскими инженерами с помощью ЭВМ. Каждая из частей этой сюиты по замыслу авторов должна восприниматься как модель определенного стиля и соответствующего способа композиции: старинного полифонического стиля, серийного атонализма и др.

Примерно в те же годы был апробирован метод сочинения музыки на основе формализованной, превращенной в алгоритмически решаемую задачу техники композиции. Компьютер здесь осуществляет сложный расчет структуры на основе принципов математической вероятности, формальной логики, теории групп. Такой способ сочинения разработал французский композитор Янис Ксенакис.

Большую помощь исследователям компьютер принес как средство хранения и оперативной обработки музыкальной информации. Так, машинная каталогизация фольклора позволяет не только надежно его сохранять, но и получать информацию по всем формализуемым свойствам и параметрам музыкально-поэтического текста песен. Такие каталоги созданы в западных университетских центрах изучения музыкального фольклора, имеющих крупные собрания напевов. Каталог есть и в нашей стране, в Ереване. Он создан известным советским фольклористом Л. Гошовским.

Достаточно определенно обнаруживает себя направление в теоретическом музыкознании, характеризуемое применением компьютера в качестве инструмента анализа формальных сторон и свойств музыкального произведения. С помощью ЭВМ ведутся, к примеру, исследования стилей, жанров, типов композиции. Решаются задачи определения подлинности анонимных произведений, их возможного авторства и т. д. В этой области накоплен значительный опыт, есть очевидные положительные результаты.

Гораздо скромнее выглядят учебно-музыкальные приложения ЭВМ. Насколько нам

известно, в нашей стране еще не предпринимались педагогически последовательные и результативные попытки использовать компьютер на занятиях по музыкальным предметам ни в специальных учебных заведениях, ни на уроках музыки в школе — и это несмотря на то, что большинство выпускаемых сейчас персональных ЭВМ предлагают пользователю определенные возможности музыкального программирования. Наличие звукогенерирующего устройства в системе ПЭВМ либо возможность соединения последней с электронным синтезатором позволяют не только решить какую-то формально-логическую задачу построения музыкального текста, но и реализовать результат в живом звучании с помощью отдельной программы формирования звука. Обычно ПЭВМ выпускаются с уже готовыми программами, воспроизводящими звучание различных популярных мелодий, дающими возможность исполнять музыку с помощью алфавитно-цифровых клавиш, сохраняющими в памяти подобранный фрагмент музыкального текста, воспроизводящими на экране дисплея (или на бумаге) нотную запись фрагмента. Конечно, это программное обеспечение носит демонстрационный характер, служит приятным дополнением к играм и является элементом рекламы данной марки ПЭВМ. Но музыкальные возможности машины можно использовать и на благо музыкальной педагогики.

Прежде всего отметим, что в качестве компонентов компьютерной музыкальной обучающей системы (КМОС) мы рассматриваем учащегося, компьютер и предмет «информационного отношения» между ними, т. е. саму музыку. Следует помнить, что в системе объединены весьма разные самостоятельные области человеческой практики; мы имеем в виду музыкальное искусство, кибернетику (информатику) и учение как тип деятельности человека. Сложность компонентов определяет разнообразие вариантов системы, богатство ее свойств.

И все-таки на первых порах можно установить наиболее очевидные характеристики КМОС.

Первая их группа характеризует интерфейс (способ контактирования человека с ПЭВМ). Среди вариантов компьютерного «входа» можно выделить стандартно-клавиатурный, музыкально-клавиатурный и микрофонный. «Выход» информации может быть незвуковым (через принтер, дисплей), звуковым (через собственный динамик компьютера или через звукосинтезатор), комбинированным.

Теперь представим КМОС с точки зрения ее музыкального содержания. Здесь уместно еще раз подчеркнуть, что ни один из возможных результатов машинной обработки информации о музыкальной форме не может быть назван собственно музыкой. И это совершенно не зависит от того, кто и как этот результат планирует, воспринимает и осознает. Строго теоретически результаты компьютерной обработки данных не обладают целым рядом атрибутивных свойств музыки, т. е. именно тех свойств, которые позволяют определять некое акустическое явление как музыку (доказательство этого тезиса здесь невозможно, оно требует детальной аргументации). Разумнее всего трактовать эти результаты как модели музыки, отражающие определенные ее свойства, стороны, закономерности.

Исходя из этого, рассматриваемые системы можно различать по степени совершенства и полноты реализации в них моделей музыкальной формы. Очевидно, самыми простыми и обобщенными моделями такого рода, способными фигурировать в качестве элементов компьютерной информации, являются отдельные тоны разной высоты и длительности, интервалы, гаммообразные последовательности. Эти модели соответствуют звукоморфологическому уровню строения музыкальной формы. В сочетании друг с другом подобные элементы образуют структуры, которые модельно соответствуют более высокому — лексико-синтаксическому этажу организации музыкального произведения. Это ладово, тонально упорядоченные ряды звуков, модели одноголосной или многоголосной музыкальной речи.

Самый высокий уровень компьютерного моделирования может быть назван квазимузыкой. Это высокоорганизованные многоуровневые модели, учитывающие и воспроизводящие все основные закономерные черты композиционно упорядоченной речи, целостной формы музыкального произведения. Подобные модели обычно несут и некую художественную информацию, «закодированную» в звуковых музыкально-речевых формах. Таковы, например, некоторые эпизоды уже упоминавшейся сюиты «Иллиак» или песенные мелодии, «сочиненные» ЭВМ по методике Р. Х. Зарипова.

Рассмотрим, наконец, компьютерную обучающую систему с точки зрения ее практической функции. Каков ее возможный педагогический смысл? В этом плане мы предлагаем различать:

1) информирующие компьютерные системы («компьютер-лектор»), где ПЭВМ выступает только в роли источника информации и эффективно «загружает» память учащегося; 2) формирующие системы («компьютер-тренер», «репетитор»), где ПЭВМ нацелена

на формирование автоматических физических и умственных действий учащегося, навыков распознавания образов. Если говорить о музыкальном содержании подобных систем, то они могут формировать слуховые «сенсорные эталоны», т. е. устойчивые представления о высотных, ритмических и прочих соотношениях между звуками, а также навыки специфических мыслительных действий со звуковыми представлениями и понятиями;

3) развивающие системы («компьютер-учитель») имеют педагогической целью совершенствование способностей обучающегося. Таких, например, как способность к произвольному запоминанию информации, комбинированию образов, творческому мышлению. К этому типу следует отнести и самые сложные (в плане формализации учебных задач) программы стимулирования художественного творчества.

Приведенная рабочая типология КМОС ни в коей мере не претендует на полноту и строгость. Это лишь попытка очертить область теоретических проблем машинно-информационного подхода к музыкальному обучению и, самое главное, увидеть определенные границы и перспективы практических разработок в этом направлении.

Условия для последних уже сложились. В настоящее время во многих школах, техникумах и вузах действуют классы ПЭВМ. В Киевской государственной консерватории им. П. И. Чайковского оборудован информационный центр, оснащенный современной электронно-вычислительной аппаратурой, в Новосибирской государственной консерватории им. М. И. Глинки недавно создана кафедра «Использование ЭВМ в музыкальном обучении». Повсеместно организуются курсы программирования для представителей различных специальностей. Так что практическая работа по созданию КМОС — дело уже сегодняшнего дня.

Познакомим читателя с простой программой, одной из разработанных совместными усилиями кафедры теории и истории музыки и кафедры информатики и вычислительной техники Одесского государственного педагогического института им. К. Д. Ушинского.

Эта программа представляет КМОС, которая, с точки зрения нашей типологии, имеет следующие характеристики:

1) ее педагогическое назначение состоит в формировании навыка распознавания типичных отношений тонов по звуковой высоте (т. е. интервалов);

2) содержание данной КМОС определяют модели звуковысотных соотношений в музыкальной форме;

3) с точки зрения интерфейса эта система имеет стандартно-клавиатурный вход, звуковой и экранно-дисплейный выход.

Действие данной КМОС напоминает работу педагога-репетитора с учеником по предмету сольфеджио. Программа так и называется: «Урок сольфеджио. Слуховой анализ». В точном соответствии с традиционной формой занятия машина предлагает учащемуся в «живом» звучании различные однотипные элементы музыкальной формы: интервалы, трезвучия, септ- и нонаккорды, тетрахорды, лады народной музыки и т. п. Эти элементы могут прозвучать мелодически (последовательно) либо гармонически (одновременно). Учащийся должен сравнить звуковую форму, предложенную машиной, с теми слуховыми эталонами, которые хранятся в его памяти (или же складываются постепенно в процессе этого и других упражнений). «Отыскав» соответствующий образ-эталон, ученик должен дать машине ответ: ввести название этой эталонной формы с клавиатуры. Далее машина дает оценку и продолжает педагогический диалог с учащимся.

Написана программа на Бейсике. Опишем кратко ее работу.

Первый этап собственно учебного действия — общее ознакомление учащегося с содержанием. На экране дисплея появляются слова «Урок сольфеджио. Слуховой анализ» и меню программы: 1) интервалы; 2) трезвучия; 3) тетрахорды; 4) лады народной музыки; 5) септаккорды; 6) нонаккорды. В нижней части экрана высвечиваются слова «Нажмите нужную цифру». Нажатие, например, клавиши с цифрой 1 приводит к выдаче информации о первом подразделе программы. На экране появляются пронумерованные названия интервалов от примы до квинтдецимы (две октавы).

Программа предполагает возможность ограничения объема материала (количества интервалов), который будет затем включен в работу. На экране под списком интервалов высвечивается слово «предел», рядом с которым нужно вписать числовой идентификатор наибольшего интервала, которым будет ограничен ряд предъявляемых затем звуковых форм. Например, ввод цифры 12 — номера интервала октавы — означает, что будут предлагаться звучания различных интервалов в диапазоне октавы. Как только будет нажата клавиша «ввод», на экране появятся следующие определители задания: «1 — вверх» и «2 — вниз». Так машина «спрашивает»: исполнять ли ей далее интервалы вверх или вниз от звука? Для ответа нужно нажать клавишу с цифрой 1 или 2 соответственно намерениям. После очередного нажатия клавиши «ввод» машина воспроизводит через динамик первый случайно выбранные

ею интервал. Это и есть вопрос, в ответ на который нужно ввести тот цифровой индекс, под которым в оглавлении значится данный интервал. Если ответ ошибочен, то ПЭВМ вновь воспроизведет тот же музыкальный интервал — и так до тех пор, пока он не будет определен правильно. Заметим, что искать в оглавлении название интервала и соответствующий ему номер не обязательно. Ведь номер интервала отражает количество входящих в него полутонов, что является традиционной музыкально-теоретической характеристикой интервала. Таким образом, одновременно с освоением качественных различий интервалов закрепляется представление об их количественных характеристиках.

Так работают и все другие подразделы данной КМОС. Система предлагает вместе с тем разные варианты постановки учебной задачи и способы обработки ответов учащихся. Например, такие:

наличие в серии заданий одного неизменного звука, служащего для построения разных элементов вверх и вниз от него;

ограничения высотного регистра (нижний, средний, верхний), в котором будут звучать задаваемые элементы;

варианты последовательности появления звуков в трезвучиях, септаккордах, тетрахордах;

ограничение времени, отпущенного для ответа;

учет и анализ верных и неверных ответов обучаемого и автоматическое построение нового корпуса знаний, учитывающих допущенные ошибки в ответах;

выбор и воспроизведение заданий из разных подразделов системы в рамках одного теста (вариант «экзамен»);

проверка сугубо теоретических знаний (например: определить абсолютные названия звуков какого-либо элемента, заданного в конкретной тональности).

Подобная программа, по нашему мнению, может найти применение на всех уровнях системы музыкального образования. Особенно в тех случаях, когда необходимо интенсивное восстановление навыков после длительного перерыва в учении либо при необходимости быстрого и прочного формирования специальных музыкальных навыков. В детских музыкальных школах подобные программы целесообразно облекать в форму компьютерной игры. Что же касается средней школы, то здесь необходимы не просто иные формы, но программы качественно иного рода. Их задачи должны коррелировать с универсальными целями эстетического, художественного, музыкального развития детей. Пришла пора обдумывать возможности ПЭВМ в решении и этих сложных задач.

ЧТО МОЖЕТ ЭВМ?

Компьютер управляет сердцем

Когда речь заходит о компьютерах, люди старшего поколения, как правило, сразу же пытаются ретироваться. А вот один американец в возрасте 71 года из штата Теннесси носит компьютер в сердце.

Программируемый кардиостимулятор, разработанный в Алабамском госпитале,— первый из имплантируемых приборов подобного типа, в котором применено компьютерное управление режимами работы стимулятора. Новый аппарат не только обеспечивает работу сердца в заданном режиме, но и подстраивается под его ритм, обеспечивая тем самым наиболее комфортабельные условия жизни пациенту. Кроме того, стимулятор постоянно собирает информацию о работе сердца, которая позднее, на приеме у лечащего врача, может быть считана и обработана с помощью более мощного стационарного компьютера.

Проанализировав работу сердца пациента за последний период, врач с помощью своей кабинетной ЭВМ может подстроить программу компьютера вживленного стимулятора посылкой радиокоманды, задав либо более мягкий, адаптивный режим работы стимулятора, либо более жесткий и не зависимый от ритма сердца режим работы в заданном темпе.

Начиная с этого номера, мы открываем новую рубрику, посвященную проблемам использования компьютеров детьми дошкольного возраста. Со вступительным словом к читателям обращаются:

С. НОВОСЕЛОВА, канд. психол. наук, НИИ дошкольного воспитания АПН СССР, председатель секции дошкольного воспитания ОНМСИО

— Компьютер как средство человеческой деятельности войдет в жизнь ребенка через игру — наиболее близкую ему форму размышления о мире. НИТ обогатят интеллектуальную основу психического развития личности дошкольника при условии соблюдения заповеди: не навреди!

Н. РЕУЦКАЯ, Управление дошкольного воспитания Гособразования СССР, ученый секретарь секции дошкольного воспитания ОНМСИО

— Введение НИТ в систему дошкольного уровня образования находится и будет еще какое-то время находиться на этапе экспериментирования. Главная задача — как можно точнее, проще и надежнее отладить сложный педагогический механизм. Результатом станет качественно новая среда для всестороннего развития личности.

Ю. ГОРВИЦ, ЦНИИ «Электроника», член секции дошкольного воспитания ОНМСИО

— На основании личного опыта использования компьютеров в детских садах и дома я считаю, что эффективное использование компьютеров для развития детей в дошкольных учреждениях различного профиля не только возможно, но и необходимо, при условии комплексного подхода и при участии компетентных и заинтересованных людей.

Приглашаем к сотрудничеству

Рубрика создается не на пустом месте — пусть небольшой, но опыт применения компьютерной техники в дошкольных учреждениях уже есть. С каждым днем содержание и география экспериментальных и практических работ в этой области расширяются. К моменту выхода в свет этого номера журнала число дошкольных учреждений, оснащенных компьютерами, будет составлять уже не десятки, а сотни. Это все равно капля в море по сравнению с общим числом детских учреждений — одних только детских садов в стране насчитывается около 140 тыс., а ведь есть еще детские больницы, детские дома, пионерские лагеря, станции юных техников, центры детского творчества, дома пионеров, в которых можно встретить детей дошкольного возраста, приобщающихся к использованию компьютера как к средству своей деятельности (практическое экспериментирование, конструирование, художественная деятельность, игра и др.). Процесс информатизации образования необратим, поэтому мы считаем жизненно необходимым появление в журнале раздела, в котором будут регулярно освещаться вопросы применения новых информационных технологий (НИТ) в начальном (базовом) звене системы образования — дошкольном.

В этой вступительной статье мы приглашаем к сотрудничеству на страницах журнала всех, кому близки проблемы нового раздела, в первую очередь ученых — педагогов и психологов, гигиенистов, физиологов, эргономистов, специалистов в области информатики, программирования, социологов. Их участие необходимо для реализации концепции информатизации дошкольного уровня образования, обеспечения научной основы стратегии развития дошкольной практики на базе использования НИТ. Представляется важным освещение теоретических и практических вопросов программно-методического обеспечения дошкольных учреждений, оснащенных компьютерно-игровыми комплексами (КИК). КИК — это не только форма организации внедрения компьютерной техники в жизнь детского сада, но и эффективный метод воспитания, развития личности дошкольника. Одновременно это возможность поднять качественно методическую службу детского сада, обеспечить своевременный и полный анализ состояния здоровья детей, разгрузить педагогический персонал от рутинных элементов управленческой деятельности в пользу высвобождения времени и сил для индивидуального подхода к каждому из воспитанников. Опыту использования КИК в детских садах, другим подходам в области использования НИТ в практической работе с дошкольниками будет уделено большое внимание.

Особая тема — создание правильных условий использования НИТ в детских учреждениях. Вопросы сертификации детских садов до приобретения и установки ими компьютерной техники, соответствующие рекомендации, предложения по проектированию детских садов, предусматривающих наличие компьютерно-игрового комплекса, найдут, надеемся, место на страницах нашего раздела.

Одной из центральных проблем является обучение кадров, владеющих НИТ, по всем уровням образования, готовящих специалистов в области дошкольного воспитания, подготовка воспитателей в педагогических училищах, педагогов и методистов высокой квалификации на дошкольных факультетах педвузов страны, формирование корпуса психологов, владеющих компьютерными методами исследования, диагностики и коррекции психического развития дошкольников и младших школьников. Подготовка специалистов указанных профилей в системе институтов усовершенствования учителей, ИПК, методических объединений, в информационных учебных центрах не может остаться вне внимания авторов и читателей нашего раздела. В этой связи представляет интерес не только внутрисоюзный обмен опытом научных исследований, практики использования НИТ в дошкольном воспитании, но и систематическая информация о положении дел за рубежом, о международных конференциях, симпозиумах, о работе ученых других стран в этой области.

В наших публикациях мы постараемся ответить на ряд вопросов, главным из которых является: «Зачем нужен компьютер маленьким детям и какие последствия (психологические, социальные и др.) для человеческой личности повлечет за собой использование НИТ уже в дошкольном возрасте?» Эти вопросы в СССР были впервые поставлены в 1986 г. при определении содержания комплексной программы АПН СССР «ЭВМ в школе», которая включила направление исследований «Психолого-педагогические основы использования компьютерных развивающих игровых программ и игрушек в системе дидактики детского сада».

Головной организацией по этому направлению стал НИИ дошкольного воспитания АПН СССР (директор — чл.-кор. АПН СССР Н. Н. Поддьяков). Летом 1986 г. работа этого направления получила мощную поддержку со стороны ЦНИИ «Электроника» МЭП СССР (директор — д-р техн. наук Ю. Б. Митюшин). Совместными усилиями двух коллективов был разработан первый в стране компьютерно-игровой комплекс, установленный в яслях-саду № 1623 Гагаринского района Москвы (директор — Г. М. Лаштабега). Центр реализации дошкольного раздела научно-исследовательской программы Гособразования СССР «Информатизация народного образования» (головная организация — НИИ дошкольного воспитания АПН СССР) функционирует на широкой междисциплинарной основе, учитывает достижения ученых различных институтов и ряда педвузов страны (Шауляйский ГПИ, Новосибирский ГПИ, Горьковский ГПИ, Бельцкий ГПИ и др.), практических работников детских садов, преподавателей педучилищ, специалистов органов управления дошкольным уровнем образования и др.

Программа основывается на положении о том, что НИТ является мощным средством амплификации (обогащения) когнитивной, в том числе мотивационной, интеллектуальной, операциональной, эстетической основы деятельности ребенка, его сознания, личности. Человеческий фактор имеет в этой программе ключевое значение, мощно ориентирует все направления научных разработок в этой области.

К проведению работ по программе привлечены высококвалифицированные специалисты в области дошкольной педагогики, общей и возрастной психологии, физиологии и медицины, эргономики, программирования.

По мере завершения исследований предполагается широкое обсуждение разработанных рекомендаций, программ и методик с членами секции дошкольного воспитания Общесоюзного научно-методического совета информатизации образования Гособразования СССР (ОНМСИО), которая является общественным органом, определяющим основные направления работ по применению информационных технологий в сфере воспитания и обучения детей дошкольного возраста.

С. НОВОСЕЛОВА, Н. РЕУЦКАЯ, Ю. ГОРВИЦ

С. НОВОСЕЛОВА

Проблемы информатизации дошкольного образования

Психологическая готовность к жизни в информатизированном обществе, начальная компьютерная грамотность, культура использования персонального компьютера как средства решения задач деятельности становятся сейчас необходимыми каждому человеку независимо от профессии. Все это предъявляет качественно новые требования и к дошкольному воспитания — первому звену непрерывного образования, одна из задач которого — заложить потенциал обогащенного развития личности ребенка.

Проблема обогащения (амплификации) психического развития ребенка, его личности в целом была поставлена А. В. Запорожцем как главнейшая в числе проблем воспитания. Детство, писал он, является самоценным периодом в жизни человека. Именно в этот период ребенок овладевает общественным опытом — человеческими родовыми способами и средствами осуществления своей деятельности — важной основой полноценного развития его личности.

Опыт человеческой деятельности всегда общественен и обобщен. Он становится достоянием личности, пройдя сквозь призму активного процесса его присвоения в собственной деятельности каждого человека, что происходит непрерывно в течение всей жизни, но особенно ярко и полно — в детстве, в периоды младенчества, раннего и дошкольного детства.

Материалы исследований о развитии предметной, предметно-игровой и игровой деятельности детей показали, что способы решения задач деятельности в условиях ее формирования приобретают все большую обобщенность, вбирая в снятой форме предыдущий социальный опыт. Таким образом, обобщение ребенком опыта своей деятельности в способе ее осуществления, выступает в качестве внутренней основы развития его мышления, которое, в свою очередь, является «механизмом», обеспечивающим содержательно-предметный характер деятельности. Мышление, согласно последующим разработкам концепции амплификации, является интеллектуальной основой развития деятель-

ности ребенка, а сам процесс овладения обобщенными способами решения задач деятельности ведет к ее осуществлению на все более высоком уровне, что в полной мере касается и овладения способами деятельности, опосредованной компьютерными средствами.

Поступательное повышение уровня реализации деятельности не означает однобокости развития личности. Чем выше интеллектуальный уровень осуществления деятельности, тем полнее в ней происходит обогащение всех сторон личности.

Достижению высоких целей всестороннего творческого развития личности следует дать фундамент раннего амплифицированного развития ребенка. Этот процесс нуждается не только в научном и методическом обосновании, но и в системе материальных средств своевременного интеллектуального развития. Концепция такого подхода, связанного с программированием процесса усвоения ребенком всей иерархии обобщенных способов действия в своей самостоятельной деятельности, должна быть воплощена в игрушках, предметах и средообразующих игровых объектах для младенцев, детей раннего возраста и дошкольников. НИТ должны занять свое место в системе средств обогащенного развития ребенка.

Введение компьютера в систему дидактических средств детского сада может стать мощным фактором обогащения интеллектуальной основы умственного, эстетического, нравственного и физического развития ребенка. Компьютер перспективен для повышения общего уровня воспитательно-образовательной работы с детьми в детском учреждении. Идея воспитания поколения, имеющего с ранних лет сформированную психологическую (личностно-мотивационную, интеллектуальную, операционную) готовность к использованию персональных и профессиональных компьютеров, компьютерных программ различного содержания и уровня, является перспективной для развития промышленной, гуманитарной, научной, экологической, бытовой сфер общества, всей его культуры.

Процесс овладения компьютерной грамотностью требует ответственного учета человеческого фактора. Особенно остро это требование звучит по отношению к детям на дошкольной ступени образования.

Предпринимать что-то новое в отношении детей дошкольного возраста всегда следует с разумной осторожностью.

В обществе обсуждаются вопросы бытования человекоемких и наукоемких технологий. К сожалению, и педагогический метод может оказаться детоемким, т. е. лишающим детство самоценности, ломающим возможности развития личности, ведущим не к творчеству, а к репродуктивности целей, исполнительству без инициативы.

Успех компьютеризации во многом зависит от метода ее введения на тех или иных уровнях образования и, разумеется, от качества техники и методики ее применения, соответствия человеческим возможностям.

В целом процесс компьютеризации образования обнаруживает тенденцию постепенной адаптации содержания информатики по нисходящей от вуза к школе и от старших классов к младшим: осуществляется изучение языков программирования, использование программных средств для обучения, тренинга, диагностики и т. д.

Мини-информатика и компьютерный тренинг представляются многим доступными и для дошкольников, но в так называемых детских вариантах. Этот традиционный для системы образования ход от вершины науки к ее простой модели, по-видимому, может быть доведен до абсурда именно на ступени младшего школьного и дошкольного возрастов: наука умирает в мини-варианте, а ребенок-дошкольник не приобретает нового средства реализации своей деятельности.

Совершенно очевидно, что необходимо на ступени дошкольного, а возможно, и младшего школьного возраста идти не от информатики к деятельности, а наоборот: от деятельности к информатике. Иными словами, компьютер должен стать развивающим и обогащающим средством самостоятельной деятельности ребенка. Информатика войдет в жизнь ребенка через его игру, конструирование, художественную деятельность и др. Компьютерные игровые и обучающие программы должны носить прежде всего развивающий характер, быть близки интересам ребенка, реализовать его стремление к экспериментированию, творчеству.

Достижение этих целей, как показывает отечественный и мировой опыт, практически невозможно в ранних возрастах без учета специфики деятельности ребенка, его игровых интересов. Необходимы компьютерные программы, строение которых отвечало бы интеллектуальной структуре деятельности ребенка. Лабиринтная модель, так же как модель выбора или вопросно-ответная форма построения компьютерной программы, здесь недостаточны.

Для проведения с дошкольниками игр и занятий с использованием компьютерных программ необходимо создание компьютерно-игрового комплекса, в котором игра и ее новое средство — компьютер, компьютерная программа выступят в единстве и в

новом инициативно-проблемном качестве.

Здесь уместен вопрос: почему именно компьютерно-игровой комплекс, а не «класс информатики» или, на худой конец, «компьютерный зал», куда можно приводить дошкольников, младших школьников с тем, чтобы они под руководством воспитателей или учителей играли бы в привычные всему миру видеоигры или упражнялись бы в началах математики, грамматики, постигали бы тайны классификации и сериации предметов по форме, величине, цвету, смыслу и т. д.? Почему бы им после этого, или до, не порезвиться, играя в обычные игры? Какая необходимость в компьютерно-игровом комплексе, в новом типе компьютерных игровых программ?

Дело в том, что процесс психического развития ребенка, развитие его мышления — это непрерывный процесс, происходящий в его деятельности. И с этим нужно считаться, разрабатывая стратегию и средства повышения интеллектуального потенциала личности в условиях информатизации всех сфер жизни общества.

Компьютерно-игровой комплекс — это система условий, позволяющих сочетать игры и занятия на компьютерах с разнообразными развивающими играми и занятиями, а также с физически подвижными играми, занятиями в физкультурном зале, в бассейне. Компьютерно-игровой комплекс — это метод, требующий установления содержательных связей между системой знаний, получаемых ребенком в повседневной жизни, в процессе обучения, игрой как ведущей деятельностью, в которой складываются прогрессивные психические новообразования его личности и его деятельностью, осуществляемой с помощью компьютерных средств.

Использование компьютерных игр и игрушек должно стать звеном дидактической системы детского сада. Компьютерные игры не заменяют обычные игры, а дополняют их, обогащая педагогический процесс новыми возможностями. Для реализации этих возможностей необходимо четкое понимание того, что способности ребенка, обеспечивающие его игры с использованием компьютера, формируются до того, как он сел за ЭВМ. Немаловажное значение для успеха освоения компьютера имеет своевременное приобщение детей ко всем формам ознакомления с окружающей действительностью, природной и социальной, как источнику чувства, мысли и творчества. Для полноценного использования компьютера как средства деятельности, средства познавательного анализа информации о действительности от ребенка требуется умение оперировать символами (знаками), обобщенными образами, т. е. ему необходимо достаточно развитое мышление, творческое воображение, определенный уровень произвольности действий. Все это формируется у ребенка в его многообразной практической и игровой деятельности. При этом особое значение для формирования потребности целенаправленного управления компьютером, развития компьютерных игр имеет ведущая деятельность ребенка-дошкольника — игра.

Игра — форма размышления ребенка об окружающем мире. В игре ребенок оперирует своими знаниями, опытом, впечатлениями, отображенными в обобщенной форме игровых способов действия, игровых знаков, приобретающих значение в смысловом поле игры. Способность детей замещать в игре реальный предмет игровым с переносом на него реального значения, реальное действие — игровым, замещающим его действием лежит в основе способности осмысленно оперировать символами на экране компьютера. Именно поэтому компьютерные игры должны быть неразрывно связаны с обычными играми. Если в обычной игре ребенок отображает действительность с помощью «обозначающих», но вполне реальных предметов, то при игре с помощью компьютера он может, пользуясь специальной развивающей игровой программой, создавать функционирующий «видеоряд», опираясь на образ своего воображения. Мыслимое ребенком переносится им на экран. Эти новые условия игры должны предоставить инициативу постановки игровых проблем на экране самому ребенку. Постановка игровых проблем, их «видео» и речевая интерпретация, включение в контекст игрового и реального общения с другими детьми и воспитателем — сложнейший процесс, обеспечить который должны принципиально новые компьютерные программы. Их строение, как уже говорилось выше, должно соответствовать интеллектуальной структуре деятельности ребенка, отвечать ее мотиву. Только такие развивающие игровые программы могут успешно служить средством деятельности ребенка, направленной на освоение и творческое преобразование многообразных содержаний-информаций. Здесь недопустимо навязывание готовых содержаний, тренинг ради тренинга, т. е. сугубо нормативного усвоения материала в вопросно-ответной форме, персонифицирующей компьютер. Процесс освоения ребенком компьютера как средства осуществления человеческой деятельности оборачивается не прямой связью «ребенок — компьютер», а опосредованной «ребенок — компьютер — цель». Причем под целью в зависимости от контекста задачи

может пониматься сколь угодно широкий ряд целей, возникающих в игровой, художественной, учебной, трудовой, общении и других видах деятельности, осуществляемых с включением компьютера. Компьютер — средство деятельности ребенка, а не его собеседник, как это часто декларируется многими. Собеседником, объектом общения при взаимодействии с компьютером для ребенка выступает не «железо», не конкретная программа, а внутренний мир самого играющего ребенка или его товарища по игре. Ребенок, впервые садящийся за компьютер, вступает в общение не с ним, а с информатизированным обществом, создавшим информационную технологию с целью повышения уровня интеллектуальной основы разумного управления происходящими в нем процессами развития. В этой связи традиционная для классической доинформационной эргономики система «человек — машина» здесь должна быть заменена системой «человек — машина — человек», поскольку человеческая деятельность (ее цель) должна быть обращена к человеку, иначе она теряет свой гуманистический смысл. Именно поэтому освоение ребенком компьютера (в широком и более узком смысле) должно рассматриваться как процесс формирования у него отвечающих этим новым условиям способов и средств осуществления своей деятельности, т. е. решения задач деятельности.

Вместе с тем становится очевидно, что компьютер как средство деятельности не должен в своей технической и программной сути быть враждебен ребенку, а компьютерно-игровой комплекс в целом должен отвечать необходимым физиолого-гигиеническим, эргономическим (включая дизайн) и психолого-педагогическим требованиям, быть безопасным в эксплуатации.

Очерченный выше круг проблем требует основательной научной и опытно-педагогической базы для своей дальнейшей разработки. Начат эксперимент по внедрению компьютерно-игрового комплекса (КИК) в практику работы дошкольных учреждений. Первые шаги в этом направлении поставили ряд острых вопросов. Среди них основной — необходимость соответствия условий эксплуатации КИК сертификационным требованиям, выдвинутым в результате проведенных физиолого-гигиенических и эргономических исследований.

В настоящее время планируется создание дошкольного уровневого центра информатизации образования. Положено начало фундаментальным исследованиям на широкой междисциплинарной основе.

Журнал в журнале
для школьников, студентов,
учащихся СПТУ и техникумов

МОЛОДЕЖНАЯ ИНИЦИАТИВА

В НОМЕРЕ:

Объединенная библиотека языка Си для машин типа MSX

В данной статье кратко описывается новая работа, выполненная в МКБ при ИПИАН СССР и представляющая, с нашей точки зрения, интерес как для учащихся, так и для преподавателей. Это объединенная библиотека языка Си (UCLIB — United C LIBrary) для компилятора ASCII-C на ПЭВМ серии MSX. С ее использованием уже написаны прикладные программы, которые имеют вполне самостоятельное значение. Кроме того, процесс написания этих программ явился прекрасным «индикатором» уровня библиотеки как необходимой и достаточной базы для обеспечения полноценной работы на языке Си, отличающемся от многих языков высокого уровня компактным и быстрым объектным кодом, что является решающим критерием при выборе средств программирования на ПЭВМ малой мощности.

При написании библиотеки преследовалось несколько целей:

скорость;

компактность;

удобство;

совместимость с реализациями языка Си на других машинах на уровне стандарта (прежде всего с MS-C на IBM-совместимых машинах);

реализация большого количества «красивых» и интересных идей, накопившихся за годы работы в различных направлениях.

Скорость и компактность достигнуты прежде всего потому, что все без исключения функции библиотеки написаны на Ассемблере (Assembler-Z80) тогда как стандартная библиотека ASCII — C (CLIB — CLIBrary), написана на Си. Кроме того, структура файла, содержащего функции UCLIB, такова, что при компоновке программ, написанных на языке Си, из библиотеки выбираются только задействованные в программе пользователя функции — это обеспечивает дополнительную значительную экономию в Си-программах.

Удобство — это прежде всего:

смысловые имена функций, отражающие действия, ими производимые, что упрощает их запоминание и делает программу удобочитаемой;

продуманная система передачи параметров функциям и возврата результатов, позволяющая во многих случаях использовать имя функции (ее вызов) в качестве

формальных параметров других функций и тем самым значительно сократить количество промежуточных переменных в программе;

максимально возможная диагностика ошибок пользователя, возникающих при некорректном вызове функции или из-за сбоев, возникающих при работе с периферийными устройствами (внешними носителями, печатающими устройствами и др.).

Совместимость с другими реализациями языка Си обеспечивает практически «безболезненный» перенос программ на языке Си даже между машинами разных классов. Это позволит полноценно использовать наиболее широко распространенные в сфере народного образования ПЭВМ серии MSX как инструментальное средство для реализации идей и отработки алгоритмов решения очень многих задач. Причем возможности библиотеки позволяют варьировать тематику решаемых задач от чисто системных (вплоть до драйверов и трансляторов с других языков) до чисто прикладных, связанных с графикой или (и) сложными математическими расчетами.

Библиотека называется «объединенной», поскольку кроме полного стандартного набора функций языка Си она содержит очень много (порядка 180) дополнительных функций (для реализации графики и плавающей арифметики), разработка которых могла вестись гораздо раньше создания библиотеки. Таким образом, создание библиотеки явилось своеобразной чертой, подводящей итог почти четырехлетней работы коллектива МКБ.

Рассмотрим более подробно структуру UCLIB.

Все функции библиотеки можно сгруппировать в несколько смысловых законченных блоков, связанных единой идеологией:

полный набор стандартных функций языка Си;

базовые графические функции из набора GCS;

функции работы с художественными объектами прямоугольной формы (образами);

функции работы с небольшими обособленными объектами (спрайтами);

функции взаимодействия процессора (CPU) и видеопроцессора (VDP);

функции взаимодействия оперативной памяти (RAM) и видеопамяти (VRAM);

функции работы с группой экранов (игровым полем);

функции плавающей арифметики;

функции работы с файлами как с логически неделимой единицей данных;

функции быстрого доступа к информации во внешней памяти непосредственно через драйвер устройства;

функции работы с прерываниями;

функции обслуживания периферийных устройств (джойстики, мышь, принтер и т. д.).

Ниже приводится выборочный обзор каждого из разделов с очень кратким комментарием.

Стандартные функции языка Си

Функции:

вызова внутренних функций BIOS'a и BDOS'a: bdos(), bios(), bdosh(), biosh();

вызова машинных подпрограмм пользователя: call(), calla();

работы с портами: inp(), outp();

анализа и преобразования символов: isupper(), islower() isdigit(), isspace(), isalpha(), iscntrl(), toupper(), tolower();

работы со строками и подстроками: strcmp(), strncmp(), strcat(), strncat(), strcpy(), strncpy(), strchr(), strnchr(), strlen();

ввода/вывода: getch(), getchr(), getche(), getchar(), gets(), putchar(), puts();

преобразования строка ⟷ число: itoa(), itou(), atoi();

работы с памятью: movmem(), fillmem(), swapmem();

распределения ресурсов (управления памятью): alloc() halloc(), sbrk(), alloca(), free();

работы с файлами (низкого и высокого уровня): open(), creat(), read(), write(), close(), fopen(), putc(), fgets(), fputs(), fclose(), filelength(), fsize();

управления файлами прямого доступа: lseek(), fseek().

Принципиальное изменение файловой структуры позволило снять ограничение на количество файлов, одновременно используемых программой.

Базовые графические функции

Их имена (и параметры) преднамеренно совпадают с соответствующими функциями языка Бейсик для их простоты их описания и использования: pset(), preset(), line(), lineb(), linebf(), paint(), circle().

Работа с художественными образами

Функция:

get() — взять образ с экрана в память;

put() — поместить образ из памяти на экран;

getp() — взять образ, описанный посредством таблицы ссылок, с экрана в память;

putp() — поместить образ, описанный посредством таблицы ссылок, из памяти на экран;

putl() — поместить образ из памяти на экран, с применением логических операций из набора 9 базовых функций алгебра логики и дополнительной SWAP: NOP — отсутствие операции;

NOT — логическое побитное отрицание;
OR — дизъюнкция (логическое побитное сложение);
AND — конъюнкция (логическое побитное умножение);
XOR — исключающее «или» (сложение по модулю 2);
PIRS — стрелка Пирса;
SHEF — штрих Шеффера;
EQU — эквивалентность;
IMP — импликация;
SWAP — обменять содержимое области экрана с фрагментом.

Кроме того, существует возможность имитации работы с экраном в памяти. В этом случае ко всем именам этих функций добавляется буква «m»: getm(), putm(), getpm(), putpm(), putlm().

Работа со спрайтами

Обособленность этих объектов заключается в способности их присутствия и движения на экране независимо от остального изображения.

Функция:

setspr() — задать образ спрайта;
setcol() — задать цвет спрайта (в режиме цветных спрайтов);
putspr() — спозиционировать спрайт в определенном месте экрана;
killspr() — убрать спрайт с экрана;
initspr() — инициализировать все спрайты;
getatr() — получить данные о спрайте;
turnspr() — повернуть спрайт;
mlitturn() — повернуть группу спрайтов по нескольким направлениям с различной кратностью;
driver() — функция управления спрайтовой мультипликацией.

Функция driver() является примером красивой реализации целой идеологии примитивной мультипликации, разработка которой велась около двух лет. За одно обращение она обрабатывает некоторую очередь объектов, задаваемую элементарной формальной таблицей данных.

Взаимодействие процессора и видеопроцессора

Функция:

setwrite() — установка VDP в режим записи по адресу с последующей автоинкрементацией адреса;
setread() — установка VDP в режим чтения по адресу с последующей автоинкрементацией адреса;
display() — «выключатель» дисплея;
screen() — установка видеопроцессора в один из 9 режимов работы с инициализацией видеопамяти;
waitvdp() — ожидание горизонтальной развертки;
statvdp() — чтение из регистров статуса VDP;
workvdp() — запись в рабочие регистры VDP.

Взаимодействие оперативной памяти и видеопамяти

Функция:

scrtomem() — перебросить блок из VRAM в RAM;
memtoscr() — перебросить блок из RAM во VRAM;
swapscr() — обменять местами два блока VRAM;
movscr() — передвинуть блок во VRAM;
vpeek() — считать байт из VRAM по адресу;
vpoke() — записать байт во VRAM по адресу;

Работа с группой экранов

В стандартном формате кодировки изображения в RAM ПЭВМ серии MSX нельзя одновременно работать более чем с четырьмя экранами. При определенном изменении этого формата (с небольшими ограничениями на изображение) мы получаем возможность работы одновременно максимум с 54 экранами, что позволяет ввести понятия «поле экранов» (или «игровое поле») и «окна» — видимая на экране часть поля.

Функция:

fildsize() — установка размеров поля;
windsize() — установка размеров окна;
windbase() — позиционирование окна на экране;
fildmove() — движение «поля под окном»;
windmove() — движение «окна над полем»;
unpack() — изменение сжатого формата изображения на стандартный (распаковка);
fragment() — обработка части поля в окне как образа (понятие образа см. выше) с возможностью объявления части образа прозрачной.

Плавающая арифметика

Очень большим недостатком реализаций языка Си на MSX является отсутствие плавающей арифметики, что делает невозможным разработку сколько-нибудь серьезных алгоритмов, связанных с вычислениями. В UCLIB эта возможность реализована на уровне функций, потому что написание препроцессора — задача не из простых.

Функция:

atof() — преобразование строки к плавающему числу;
ftoa() — наоборот;
itof() — преобразование целого числа к плавающему;
ftoi() — наоборот;
cmpf() — сравнение двух плавающих чисел;
movf() — операция присваивания;
negf() — унарный минус;
sgnf() — возвращает знак плавающего чила;
rnd() — генератор псевдослучайных чисел;
subf(), addf(), mulf(), divf() — вычитание, сложение, умножение и деление плавающих чисел соответственно;
cos(), sin(), tan(), atn(), log(), sgr(), exp() — стандартные

элементарные математические функции.

Файлы

Функция:
load(), bload() — загрузка файла в RAM;
loads(), bloads() — загрузка файла во VRAM;
save(), bsave() — запись файла из RAM;
saves(), bsaves() — запись файла из VRAM.

Функции с буковой «b» в начале работают со стандартными файлами системы MSX.

Быстрый доступ к диску

Функция:
absread() — чтение блока секторов диска;
abswrite() — запись блока секторов на диск.

Работа с прерываниями

Функция:
di() — запрещает клавиатурные прерывания;
ei() — разрешает клавиатурные прерывания;
di_38() — запрещает немаскируемые прерывания;
ei_38() — разрешает немаскируемые прерывания;
setint() — установка вектора прерываний («повесить функцию на прерывания»);
resint — сброс вектора прерываний
remint() — восстановление вектора прерываний.

Работа с периферийными устройствами

Функция:
stick() — опрос клавиш управления курсором, джойстиков или мышь;
strig() — опрос клавиши «пробел» клавиатуры или триггера джойстика;
printc() — вывод символа на печатающее устройство;
prints() — вывод строки на печатающее устройство.


```
/*************************************
 *                                   *
 * Программа перевода числа из одной *
 *    системы счисления в другую.    *
 *                                   *
 *************************************/

#include <uclib.h>  /*****************
                     *               *
char buf[17];        *   Вариант I   *
main()               *               *
{                    *****************/
int i,o,n;

i=atoi(gets(buf,2),10);
       /* Исходная система счисления */
o=atoi(gets(buf,2),10);
         /* Новая система счисления */
n=atoi(gets(buf,16),i));
                       /* Ввод числа */
puts(itoa(n,o,buf));
                     /* Вывод числа */
}


#include <uclib.h>  /*****************
                     *               *
char buf[17];        *   Вариант II  *
main()               *               *
{                    *****************/

puts(itoa(atoi(gets(buf,16),
                atoi(gets(buf,2),10))),
            atoi(gets(buf,2),10),buf));
}
```

```
/*************************************
 *                                   *
 *        Программа построния        *
 *  разноцветных линий по генератору *
 *       псевдослучайных чисел       *
 *    (оформление фона программы     *
 *           "600 секунд")           *
 *                                   *
 *************************************/

#include <uclib.h>

real x1,y1,x2,y2,c,dx,dy,dc;

main()
{

atof(&dx,"255.5");
atof(&dy,"191.5");
atof(&dc,"15.0");
screen(2,0);

while('strig(0)) {
   mulf(rnd(&x1,&x1),&dx,&x1);

   mulf(rnd(&y1,&y1),&dy,&y1);
   mulf(rnd(&x2,&x2),&dx,&x2);
   mulf(rnd(&y2,&y2),&dy,&y2);
   mulf(rnd(&c,&c),&dc,&c);
   line(ftoi(&x1),ftoi(&y1),ftoi(&x2),
                    ftoi(&y2),ftoi(&c));
   }
exit();
}
```

Заключение

Разработка объединенной библиотеки языка Си выполнена К. Чурюкановым. Проверка функционирования библиотеки выполнена М. Булановым, К. Кеворковым и А. Локуциевским. Функция управления спрайтовой мультипликацией разработана Д. Черемушкиным. Руководитель работ Д. Непочатых.

Разработчик библиотеки будет благодарен читателям за отзывы и предложения.

Тел. 132-19-33

К. ЧУРЮКАНОВ,
Д. НЕПОЧАТЫХ

Знакомьтесь: компьютер

В свое время братья Стругацкие выпустили книжку про программиста Сашу Привалова, которая сделала их знаменитыми, а сама быстро исчезла не только с прилавков магазинов, но и из многих библиотек. Странное название «Понедельник начинается в субботу» сопровождалось простым подзаголовком — «Сказка для младших научных сотрудников».

Не имея возможности «перевести с русского на русский» все сказки братьев Стругацких, издательство «Мир» стало искать товар за рубежом. И нашло! В Александрии (штат Вирджиния, США) в издательстве «Тайм-лайф букс» имеется целая серия подобных сказок, да еще с картинками, и какими!

Взяв для пробы несколько штук («Основы компьютера» и «Системы ввода/вывода»), издательство подготовило их для наших младших...

У истоков компьютерной революции. Все было очень просто: взяли костяшки, посадили на спицы, получили счеты. А затем повалили счетные машины как из рога изобилия. И суммирующая машина Паскаля, и палочки Непера, и калькулятор Лейбница, и машины Чарльза Бэббиджа, и табулятор Холлерита. Правда, ушло на это полторы тысячи лет. Но это не сказка, а только присказка.

Микропроцессор. И пошли авторы нашей сказки разбирать в поисках Джина века автоматики все современные устройства: и цифровые кварцевые часы — по деталькам, и фотоаппарат — полупрозрачный, и целую кучу медицинских приборов — со снятыми крышками, и автомобиль — под рентгеном, и даже дом — в разрезе. Оказалось, что там везде «специалисты». А вот дома на столе универсал — персональный компьютер, который не появился бы на свет, если бы не было двоичного кода.

Возможности двоичного кода. Начав с семейной хроники Лейбница и Буля, поиграв в кубики с кодом ASCII, авторы рассказали нам еще про нескольких «виновных» в рождении цифровых схем. А дальше все эти элементы логических схем, соединения логических вентилей, каскады сумматоров и даже музыка в числах были раскрашены, чтобы получить веселые картинки.

Разработки военных лет. Война — невеселое время. Поэтому, разбросав на страницах книги десятка три историй про Томаса Дж. Уотсона, Говарда Эйкена, Конрада Цузе, Хельмута Шрайера, Джона фон Неймана и его «Предварительный доклад», авторы сопроводили их фотографиями конкретных лиц и разработанных ими компьютеров. Этих фотографий и в Политехническом музее не увидишь.

Эволюция интегральных схем. Все началось с переключателей. И если у людей бывают интересные биографии, то у переключателей они не менее занимательные. История транзистора настолько увлекательна, что биографии ее создателей уже являются «прилагательными» к ней. А фотографии? А рукописный автограф первой в мире электрической схемы на транзисторе? Разве это не фантастика?

Вспомнив Левшу, авторы специально для нас стали под мелкоскопом показывать микросхемы. Для микросхемы Джека Килби подойдет и простой. А для крупной микросхемы пришлось даже карту рисовать, где на ней какие узлы ЭВМ расположены. Еще несколько фантастических снимков предшествуют небольшому мультфильму о том, как делают транзистор.

Золотой век предпринимателей. Это уже ближе к нашим временам. По крайней мере, сегодняшние специалисты из области вычислительной техники родились вместе с компьютерами. И им интересны фотографии и первых ЭВМ, и их создателей, которые только и знали, что соревновались между собой. Кто выпустит более быстродействующую ЭВМ наименьших размеров? И чтоб в карман влезла, и чтобы по карману была?

Биография персональных компьютеров завершается историей возникновения фирмы «Эппл» и ее эксцентричных основателей — Возняка и Джобса.

Гибкие системы в мире информации. Но компьютер был бы бесполезным ящиком, если бы не выдавал «на-гора» массу информации. А для этого нужны устройства ввода — вывода. Их разработка — это практическая область, тесно связанная с вычислительной техникой.

Рассказав про перфораторы и принтеры, нам опять предлагают посмотреть мультфильм о том, как все это взаимодействует между собой и насколько

Знаномьтесь: компьютер / Пер. с англ. К. Г. Батаева; Под ред. В. М. Курочкина. М.: Мир, 1989.

важны преобразования аналоговых сигналов в цифровые и обратно.

Не обойдены вниманием варианты самых быстрых клавиатур и обработка японских иероглифов, история фирмы «Эпсон», струйные и лазерные принтеры.

Авторы забрались в клавиатуру под клавиши и рассказали, что там происходит. Посмотрели в микроскоп на экран дисплея и показали, как строится изображение вишни. Протиснулись между печатающей головкой и бумагой и разобрали каждую буковку на точечки.

На пути к совместимости компьютеров. Свободное творчество закончилось 7 апреля 1964 г., когда фирма «Ай би эм» объявила о результатах проекта стоимостью 5 млрд. долларов, что вдвое превышало затраты на Манхэттенский атомный проект. Анонсирование System/360 явилось ударом для других фирм — изготовителей ЭВМ. Попытки прямого копирования этой серии ЭВМ такими гигантами, как «Сименс» (ФРГ) и «Хитачи» (Япония), кончились плачевно. А фирма «Эр си эй» (США) потеряла на этом 500 млн. долларов.

Длинная цепочка событий, к которым привело появление единой серии ЭВМ, заканчивается подробностями выхода компьютера к массовому пользователю. Работы фирм «Эппл» и «Ксерокс» — система окон на экране, мышь и интерфейс, основанный на символических изображениях, стали неотъемлемой частью персональных компьютеров.

Машины в медицине и другие чудеса. Пока компьютер был внутри предприятий и лабораторий, он существовал как бы сам по себе. Но, выйдя к людям уже с развитыми средствами общения, он стал для непосвященных каким-то чудом. Станок «сам» обрабатывает стальную болванку, изображение ваших органов в разрезе «само» появляется на экране телевизора, самолеты и космические корабли управляются «сами». Как это происходит, и рассказывается в этой части книги.

За пределами возможного. Если вы прочитали книгу до этого места, для вас не будет удивительным то, что пока считается за пределами возможного. Голосовая связь с компьютером, компьютерная музыка, синтезирование звука и изображения, компьютерные мультфильмы — все это трудно, но уже возможно.

А то ли еще будет!

А. АРХАНГЕЛЬСКИЙ

Программные средства для использования в учебном процессе

Корректное использование новых информационных технологий для целей обучения дает устойчивый положительный эффект. В первую очередь это проявляется в индивидуализации обучения и интенсификации учебного процесса. Применение средств вычислительной техники дает возможность преподавателю больше внимания уделять общению с обучающимися как с низким, так и с высоким уровнем подготовки и способностей.

В Гродненском госуниверситете разработаны и эксплуатируются следующие пакеты программ для обеспечения учебного процесса в системе народного образования и производства:

1. «ЭКСПЕРТ» — диалоговый «электронный» экзаменатор, предназначенный для проведения автоматизированного опроса. Может использоваться при проведении тестирования, зачетов и экзаменов.

«Эксперт» ведет диалог с пользователем (т. е. экзаменующимся) путем предъявления ему текстов заранее составленных вопросов и приема ответов. На появляющийся на экране дисплея вопрос может быть предусмотрен ответ одного из трех типов: «да»/«нет», выбора из списка альтернатив (меню) или свободно конструируемого ответа. В последнем случае ответ считается правильным, если он текстуально совпадает с одним из вариантов, предусмотренных составителем вопросов. Варианты ответов в случае меню размещаются на экране дисплея случайным образом, т. е. их порядок не воспроизводится при многократных запусках программы. Минимальное количество вариантов ответов на меню — 2, максимальное ограничивается только размером экрана диспея (21 строка).

При работе может вестись хронометрирование диалога и игнорирование ответа пользователя после истечения некоторого интервала времени с момента

предъявления текста вопроса. Этот интервал определяется для каждого вопроса отдельно. После введения ответа пользователю сообщается оценка и в случае неправильного ответа — правильный. Оценка за ответ на каждый вопрос может быть установлена от 1 до 9 баллов.

Пользователю может быть предложена определенная часть вопросов, случайным образом выбранных из всего массива либо из отдельных групп вопросов. Оценки за все ответы (как реально полученные, так и максимально возможные) суммируются, и в конце работы подводится итог, который можно занести в системный журнал. Журнал может быть просмотрен на экране или выведен на печать.

2. «ИДОС» — интерпретатор языка диалоговых обучающих сценариев. Язык «Идос» предназначен для создания обучающих, тестирующих или контролирующих программ, сценарий которых может быть представлен в виде ориентированного графа. Средства языка позволяют создавать сценарий практически неограниченной сложности, предоставляя автору большой набор сервисных программ для отладки и контроля работы обучающихся с системой. Макрокоманды языка позволяют обеспечить:

постраничную подачу материала,
выдачу заданий из специализированных банков,
прием численных и символьных ответов,
контроль их правильности,
сохранение (при необходимости прервать на время обучение) положения обучающегося в системе,
последующее восстановление в том же положении,
проведение хронометрирования диалога,
накопление информации о ходе обучения,
запуск обслуживающих и вспомогательных программ и т. д.

Пакет программ «Идос» предоставляет самые широкие возможности (включая все функции системы «Эксперт») для использования ваших средств вычислительной техники в учебном процессе.

Оценочные расчеты показывают, что затраты времени на разработку методических материалов по одной теме (1—1,5 ч дисплейного времени) составляют 100—150 ч и оправдывают себя в течение года, если с данной обучающей системой работали не менее 40—50 человек. Число лиц, которые могут одновременно работать с программами, ограничивается лишь количеством дисплеев, входящих в конфигурацию вашей ЭВМ, или числом персональных компьютеров, имеющихся в вашем распоряжении.

Все программные продукты имеют защиту от несанкционированного доступа обучающегося к служебной информации систем.

В настоящее время у разработчиков имеются программы для обучения и контроля знаний по разделам курсов «Механика», «Молекулярная физика», «Теоретическая механика».

Программы работают при использовании вычислительного комплекса СМ-1600 с операционной системой ОС РВ. Может быть проведена адаптация указанных программных средств для других машин серии СМ, персональных ЭВМ, совместимых с IBM/PC, а также средств вычислительной техники типа ДВК. Принимаются заказы на разработку автоматизированных обучающих и контролирующих программ по конкретным предметам и темам с использованием пакетов «Эксперт» и «Идос».

Разработчики обеспечивают фирменное сопровождение программных продуктов, включающее в себя их адаптацию на средствах вычислительной техники заказчика, гарантийное обслуживание программ и обучение работе с ними персонала ЭВМ и преподавателей.

Срок поставки пакетов «Эксперт» и «Идос» с адаптацией на СВТ заказчика — 2 месяца, разработка обучающих и контролирующих систем по конкретным темам и предметам — до 6 месяцев.

Дополнительную информацию можно получить по адресу: 230023, БССР, г. Гродно, ул. Ожешко, 22, университет, НИС.

Телекс: 194203, «Квант».

Телефон: 47-14-58, 45-40-17.

В. ПРОХОРОВ
канд. физ.-мат. наук

Уроки олимпиады

Когда к сотруднику Института математики и механики Уральского отделения АН СССР академику Н. Н. Красовскому обратились с просьбой стать председателем Программного комитета II Всесоюзной олимпиады по информатике (на предыдущей он возглавлял работу жюри), он образовал в институте инициативную группу научных работников для оказания помощи олимпиаде в подготовке задач. Я участвовал в этой деятельности, а позднее — и в работе жюри и хочу поделиться впечатлениями.

Наверное, найдется немало причин для критики олимпиады — от традиции последнего времени быть недовольным всем до влияния чрезвычайного разнообразия взглядов (подчас весьма аграссивно выражаемых) как на вопросы школьной информатики, так и на понятие «олимпиадная задача». К сожалению, опыт всесоюзных олимпиад по информатике крайне невелик. Так что каким бы ни был набор задач, все равно найдутся считающие его никуда не годным.

Однако, вне сомнения, имелся и ряд существенных недостатков в процедуре подготовки комплекта задач.

Откуда же берутся задачи? Кто их придумывает? Каков их путь на олимпиаду?

Официально записано, что подготовка задач — функция создаваемого Гособразованием СССР Программного комитета (в основном — из представителей различных связанных с преподаванием информатики ведомств Москвы).

Замысел, видимо, таков. После предварительного обсуждения возможных концепций члены ПК независимо готовят фонд задач-кандидатов. Для этого, вероятно, каждый из них должен «у себя на местах» изрядно потрудиться, организовав творческие группы специалистов. Затем, собравшись вновь, комитет сформирует из многообразия собранного фонда наиболее приемлемые комплекты задач для туров олимпиады. И на этом завершит свою работу.

Просто? Оказалось, не очень... Достаточно вспомнить, что за день до практического тура не только не существовало задания для этого тура, но и «фонд» оказался пуст. В чем же причины такого бедствия? С одной стороны, в том, что далеко не все члены Программного комитета подготовили свои коллекции задач. С другой — в «издержках демократии» в сочетании с отчаянным (и непримиримым) плюрализмом мнений, когда достичь согласия даже в мелочах оказывается порой невозможно.

Что касается придумывания задач для фонда Программного комитета, то наш институт деятельно в нем участвовал. Вообще, работа со школьниками в области информатики и программирования имеет у нас давние традиции. Первую городскую олимпиаду по программированию мы провели около десяти лет назад (руководителем ее оргкомитета, Программного комитета и жюри был С. С. Титов); с тех пор она проходит ежегодно. А в 1988 г. нам довелось участвовать в проведении I Всесоюзной олимпиады по информатике, которая проходила в Свердловске.

Сразу после создания в институте упоминавшейся уже «группы содействия» началась интенсивная работа. На специальных семинарах обсуждались концепция, проекты задач. В конечном счете было отобрано и подготовлено около полутора десятков задач — кандидатов на олимпиаду, наш «свердловский пакет». Конечно, в него были включены только такие задачи, по которым

была уверенность, что они будут вполне соответствовать требованиям одного из туров олимпиады.

К сожалению, не все члены Программного комитета подготовили подобные пакеты. А ведь более серьезное отношение к созданию фонда сделало бы работу существенно менее нервозной и хаотичной.

В нашем «пакете» отражались определенные взгляды на то, какой должна быть «идеальная» олимпиадная задача, определенная концепция (во избежание недоразумений отметим, что она не является «свердловской» — школьной информатикой в городе занимаемся, разумеется, не только мы и мнения могут различаться). Какова же наша концепция?

Прежде всего, безусловно, задача должна быть новой, неизвестной школьникам, придуманной специально для олимпиады. Об этом можно было бы и не говорить, но на предыдущей олимпиаде по крайней мере две из четырех задач теоретического тура были взяты из известных книг (знаменитого трехтомника Кнута и популярной книжки «Комбинаторика для программистов»).

Более того, олимпиадные задачи должны, на наш взгляд, быть не только формально новыми по тексту, но и новыми для школьника по существу, по содержательной области, по подходам к решению, по применяемым методам и средствам. При решении олимпиадной задачи сложностью должны быть не технические проблемы «расписывания» известного метода для данной задачи, а творческие муки изобретения самого метода.

И еще более того: совсем замечательно, если не только сотворение метода решения, но и сотворение самой задачи в традиционно школьном понимании требуется от школьника; иначе говоря, когда условие задачи содержит ее неформальное содержательное описание, когда основную проблему составляют внесение необходимых уточнений в исходную размытую постановку, дополнение ее разумными гипотезами, построение четкой содержательной, а затем и математической моделей.

По нашему убеждению, не следует ограничиваться в задачах лишь алгоритмическими (процедурными) информационными моделями. Существуют ведь и другие! (К сожалению, из авторов имеющихся сейчас школьных учебников по информатике лишь один уделил внимание логическим моделям.)

Большим плюсом задачи мы считаем требование в ее условии доказательства того, что основанный на эвристических соображениях ответ действительно является решением, а не просто убедительной гипотезой.

Не следует забывать, что олимпиадные задачи не только «работают» во время олимпиады, когда их решают участники, но живут и после нее, являясь в определенных отношениях «маяком» для широких слоев как школьников, так и учителей. Так что вопрос о всех «новизнах» задачи важен и с точки зрения ее поучительности. Именно здесь уместно сказать и о нашем положительном отношении к постановкам задач, содержащим не один, а несколько различающихся по сложности вопросов.

То, что школьники показывают слабость в решении определенных классов задач (например, всего, что касается построения даже самых простейших моделей), мы воспринимаем не как аргумент для устранения их из заданий, а, наоборот, как стимул к тому, чтобы именно таким задачам (или подзадачам) уделять на олимпиадах повышенное внимание.

Наконец, положительным, с нашей точки зрения, качеством задачи является ее увлекательная постановка в форме, близкой к той, которая обычно характерна для исходных постановок заказчиков в реальной деятельности специалистов (а не в форме «тепличных» школьных «условий»). Увлекательность здесь имеется в виду в том числе и с литературной точки зрения.

Разумеется, сказанное выше не является четким изложением нашей позиции, но общее представление о ней дает.

К сожалению, вынесенные на обсуждение Программного комитета его председателем эта концепция и, соответственно, «взнос» — задачи «свердловского пакета» не нашли поддержки и были отвергнуты на самых ранних этапах обсуждения почти по всем изложенным пунктам.

Что же вызвало возражения? Прежде всего, доминировало убеждение, что необходимость разработать при решении задачи ее четкую содержательную и математическую постановки, как и любая недоговоренность в условии, не достоинство, а крупный порок задачи, свидетельство ее недоработанности автором. Задача должна иметь однозначное решение («верный ответ»), поскольку в противном случае возникают сложности с ее проверкой. Из-за сложностей с определением баллов не допускалось включать в условие несколько вопросов.

Отсюда, из стремления формализовать проверку, в идеале — свести ее к сличению ответов, вытекает и желание четкой математической постановки исходной задачи, и требование достаточной очевидности единственного «верного» метода решения.

К примеру, в самом начале была отвергну-

та задача, связанная с неалгоритмическим языком функционального программирования. Аргументы — длинное условие и сомнения в способности жюри квалифицированно провести ее проверку. Хотя мне, как ее автору, кажется, что она украсила бы олимпиаду.

Ну да ладно, каждый имеет право на собственные взгляды. Пусть даже не всем они кажутся правильными. Призывы превратить всесоюзную олимпиаду во всесоюзную контрольную нас всегда будут огорчать. Пусть. Но ведь тем более надо, вероятно, дать возможность попадания в программу задачам, отражающим разные идеологии. Тем более когда непримиримость приводит к опустошению «фонда».

В конечном счете на олимпиаде были представлены разные подходы, но ведь задания туров «пеклись» в последний момент, а до этого последнего момента все предлагавшееся в «фонд» отвергалось столь решительно, что за считанные дни до олимпиады он оказался пуст...

При воспоминаниях о предварительной работе по подготовке задач на ум приходят крыловские Лебедь, Рак и Щука. Взглядов на школьную информатику и олимпиады много, и кто-то тянет в облака, в высь эстетики теоретической информатики, к изысканной информационной культуре, кто-то — в мутную воду головоломных приемов программирования, когда высшим пилотажем считаются эмуляция рекурсивного описания процедуры на Бейсике или постижение премудростей из сферы БК-логики (скажем, экспериментальное выяснение, на что влияет содержимое ячейки 123 БК-0010, записью туда различных значений), а кто-то пятится назад, видя основные задачи информатики в кропотливом расписывании какого-либо численного метода для конкретной математической задачи и «ручном» исполнении его на калькуляторе.

Конечно, в каждом из этих (и других) подходов можно увидеть как резоны, так и свои недостатки. И бог с ними, хоть «раком», да надо же куда-то двигаться! Ведь какими бы горячими ни были обсуждения, они должны заканчиваться принятием комплекта задач! Не за день, а за пару месяцев до олимпиады!

Так что речь не о том, что в итоге в комплекте олимпиады оказались будто бы никуда не годные задачи. Хотя и здесь могут быть разные точки зрения.

Еще раз можно вспомнить ситуацию с задачами за день до практического тура. Почему «фонд» был пуст? В своих «не на жизнь, а на смерть» обсуждениях Программный комитет забраковал на этапе подготовки все предлагавшиеся задачи. И вот приходится — ситуация безвыходная — в спешке в гостиничном номере сочинять условия «хоть каких-нибудь» задач (теперь уж не до жиру). На ходу автор переделывает под практический тур разрабатывавшуюся им для теоретического (и отвергнутую Программным комитетом в самом начале) задачу «Город Глуповъ». Получается задача о катании кубика с перенумерованными гранями, которая плохо годится для практического тура — разрабатывать за два часа теоретическую часть и довести решение до работающей программы, выдающей верные ответы (требование практического тура),— дело весьма непростое. Но делать нечего... И вот утром, пока школьники завтракают перед началом практического тура, Программный комитет наконец-то отбирает из трех задач-кандидатов (каждая из которых весьма сложна и могла бы одна занять все четыре часа) две, которые срочно уносятся для раздачи школьникам... Нужна ли такая штурмовщина?

Что же делать? Предлагаю Гособразованию СССР рассмотреть предложения по изменению «устава» Программного комитета; принятие одного из них, представляется, способствовало бы более успешному преодолению описанных проблем.

Первый вариант — «демократический». В Программный комитет включаются лишь те, кто берет на себя обязательство обеспечить определенное количество новых задач-кандидатов в фонд предстоящей олимпиады. Программный комитет коллегиально проводит отбор представленных задач и составляет окончательное олимпиадное задание. Члены комитета, не представившие задач, на заседания не допускаются.

Второй вариант — «конкурсный». Каждый из членов Программного комитета представляет свой полный («неделимый») комплект заданий для туров олимпиады. Комитет проводит конкурс этих комплектов; победивший и будет предложен школьникам. Возможно проведение конкурсов по заданиям теоретического и практического туров отдельно.

Третий — «монархический». При этом Гособразование СССР назначает лишь председателя Программного комитета, который один несет ответственность за подготовку олимпиадных заданий и берет на себя обязательство к определенной дате представить оргкомитету запечатанные конверты с заданиями туров. Пакет вскрывается впервые лишь в день соответствующего тура олимпиады при раздаче заданий школьникам. Как готовятся эти задачи — проблема председателя. Единственное ограничение — обеспечение «скрытности работ» и, разумеется, высокий уровень заданий. Если будет признано воз-

можным провести эксперимент по такой схеме, то коллектив, подготовивший «свердловский пакет» к прошедшей олимпиаде, готов взять на себя работу по разработке комплекта задач и предложить кандидатуру председателя.

Еще вариант — «коллегиально-монархический», когда на каждую задачу либо на каждый из туров назначается свой зам. председателя. Каждый из них готовит (как — его дело) свой раздел, из которых механически комплектуется олимпиадный пакет. Координация действий заместителей председателя подразумевается.

Небесспорной представляется и «формула» олимпиады: четыре задачи на теоретическом туре и две — на практическом, каждый из туров — четыре часа. Может быть, имеет смысл обсудить и какие-либо другие формы? Например, две задачи — теоретический тур и одна — практический, или одна задача и на теоретический, и на практический туры. В последнем варианте имеется в виду предлагать участникам «комплексную» задачу, требующую доведения после достаточно сложной теоретической проработки до уровня некоторого программного продукта, либо задачу, требующую проведения как теоретических изысканий, так и некоторых машинных экспериментов. При этом теоретический и практический туры могут проводиться друг за другом (или как один теоретико-практический тур) в течение, скажем, шести (или более) часов.

Нужно ли бояться экспериментов?

И наконец, заключение. При закрытии II Всесоюзной олимпиады по информатике всем ее участникам были в качестве «Заочной олимпиады ИНФО-УРАЛ» выданы условия нескольких задач из упоминавшегося «свердловского пакета». Их, как отражающие некоторые наши взгляды, мы приводим ниже.

1. Фирма осуществляет производство коробок, используя в качестве сырья квадратные стальные листы со стороной a. Для изготовления коробки из листа по его углам вырезают одинаковые квадраты, из полученной крестовидной заготовки сгибанием и свариванием получают готовое изделие. Оставшиеся куски металла можно затем либо продать, либо использовать на новом этапе для изготовления коробок поменьше и т. д.

Доход от продажи листа металла равен qS, где S — площадь листа; доход от продажи одной коробки равен $nv - lx - my$, где v, x, y — объем, сторона основания и высота коробки; q, n, l, m — заданные неотрицательные коэффициенты.

Как фирме следует распорядиться исходным листом, чтобы суммарный доход от продажи коробок и металла был наибольшим? Каков будет доход?

Варианты:

$a=40$, $n=1000$, $q=1$, $m=l=0{,}1$;
$a=100$, $n=q=10$, $m=l=0{,}1$;
$a=40$, $n=1$, $q=10$, $m=l=0{,}1$;
$a=100$, $n=1$, $q=m=l=0$.

(Н. Н. Красовский)

2. В одной из книг по фотографии приведен рецепт удивительного проявителя Р-6726, повышающего чувствительность пленки в 200 раз. К сожалению, авторы, подробно описав свойства Р-6726, не опубликовали способ его приготовления, указав лишь, что проявитель — водный раствор перечисленных фотореактивов, и дав расход каждого из них на литр проявителя.

Попытка растворения компонентов Р-6726 в воде в выбранной наугад последовательности показала, что все не так просто: в жидкости образовался бурый осадок и началось обильное газовыделение.

Поиски в специальной литературе привели к некоторому успеху. В одном из химических справочников в разделе «Водные растворы» для некоторых из компонентов Р-6726 давался список веществ, которые непременно должны быть в растворе при растворении данного компонента (эти вещества оказались реактивами из рецепта Р-6726). Кроме того, для некоторых компонентов приводился перечень веществ (ряд из них были компонентами Р-6726), присутствие которых в растворе при растворении данного компонента недопустимо. Вероятно, нарушение каких-либо из этих условий (отсутствующие при фирменном способе изготовления Р-6726) и привело к неудаче эксперимента. Заметим, что для каждого процесса растворения приводился рекомендуемый диапазон температур раствора, однако все эти диапазоны включали температуру 60 °С, при которой проводился эксперимент.

А. Разработайте математическую модель задачи определения способа приготовления Р-6726, удовлетворяющего приведенным в литературе требованиям.

Б. Разработайте алгоритм определения допускаемой справочником последовательности растворения реактивов для приготовления Р-6726.

В. Что означает ответ алгоритма пункта Б о невозможности выполнения всех требований справочника? Ведь авторы как-то изготовляют Р-6726!

(В. В. Прохоров, И. Ф. Сивергина)

3. Имеется следующее описание языка PROLAN/F, предназначенного для работ со

строками символов и логическими величинами.

Программа на PROLAN/F — последовательность разделенных запятыми определений функций и запросов значений этих функций для конкретных значений фактических параметров (строковых или логических).

В языке имеется несколько «встроенных» функций:

Обращение	Аргументы	Тип значения функции	Чему равно значение функции
пусто? (*x*)	*x* — строка	логический	ДА, если *x* — пустая строка
симрав? (*x*, *y*)	*x*, *y* — символы	логический	ДА, если *x* и *y* равны
консим (*x*)	*x* — непустая строка	односимвольная строка (символ)	Последнему символу *x*
начстр (*x*)	*x* — непустая строка	строка	Строке, получающейся отбрасыванием у *x* последнего символа
припконсим (*x*, *y*)	*x* — строка *y* — символ	строка	Строке, получающейся приписыванием к строке *x* символа *y*

Определения новых функций имеют вид:

⟨Имя определяемой функции⟩ (⟨Список формальных параметров⟩)= =⟨(Условное значение⟩.

Здесь ⟨Список формальных параметров⟩ — последовательность разделенных запятыми идентификаторов формальных параметров. ⟨Условное значение⟩ имеет вид:

(⟨Условие⟩ — ⟨Значение⟩,
⟨Условие⟩ — ⟨Значение⟩,
. . .
ИНАЧЕ — ⟨Значение⟩).

⟨Условное значение⟩ полагается семантически эквивалентным первому из ⟨Значений⟩, для которых истинно записанное перед ним ⟨Условие⟩, если же все ⟨Условия⟩ ложны — то ⟨Значению⟩, записанному после ИНАЧЕ. ⟨Условия⟩ и ⟨Значения⟩ в ⟨Условном значении⟩ могут быть константами (логическими для ⟨Условий⟩ и логическими или строковыми для ⟨Значений⟩), обращениями к функциям (в том числе рекурсивным) или опять же некоторыми ⟨Условными значениями⟩. Такими же могут быть и фактические параметры упоминаемых в ⟨Условном значении⟩ функций.

Заметим, что в языке PROLAN/F нет таких понятий, как оператор присваивания, цикл и т. п., что делает программирование на нем не очень скучным.

В описании приведен пример программы на PROLAN/F.

И(*x*, *y*)=(*x*—*y*, ИНАЧЕ — НЕТ),
ИЛИ(*x*, *y*)=(*x* — ДА, *y* — ДА, ИНАЧЕ — НЕТ),
НЕ(*x*)=(*x* — НЕТ, ИНАЧЕ — ДА),
НЕ(И(ДА, ИЛИ(ДА, НЕТ))),
И (НЕТ, НЕ(НЕТ))

и ее результат: «НЕТ, НЕТ».

Требуется дать на PROLAN/F определения функций (с доказательствами):

а) стррав (*x*, *y*), равной ДА, если аргументы (строки *x* и *y*) равны;

б) подстрока (*x*, *y*), равной ДА, если строка *x* является подстрокой строки *y*;

в)* перевор (*x*) равной строке, получаемой перестановкой символов строки *x* в обратном порядке;

г)* подставить_лев (*x*, *y*, *z*), равной строке, получающейся из строки *z* заменой самого левого вхождения в нее подстроки *x* (если таковые в *z* есть) на строку *y*;

д)* подставить_все (*x*, *y*, *z*), равной строке, получающейся из *z* заменой каждого из вхождений в нее подстроки *x* (если таковые имеются) на строку *y*.

е) какой-либо функции, определение которой на этом языке кажется вам поучительным.

(В. В. Прохоров)

4. Разработайте программу-интерпретатор программ, написанных на PROLAN/F (транслятор с PROLAN/F).

5. Придумайте интересную задачу для олимпиады по информатике.

Примечание. Задачи 4 и 5 сформулированы для заочной олимпиады и на всесоюзную не предлагались: пункты, помеченные звездочкой, в заочной олимпиаде были опущены.

В оговоренный срок (около месяца со дня получения условий) наилучшие решения прислал Константин Ткачук из Ташкента (VIII класс). Особое внимание привлек сделанный им транслятор с языка PROLAN/F (работающий!). Далее отметим работу Ахто Труу из Тарту, в особенности

решение второй задачи. Неплохие работы прислали Николай Коровин из Горького и Дмитрий Копелевич из Краснодара.

Подробнее с задачами «свердловского пакета» для II Всесоюзной олимпиады (и их, надеемся, поучительными авторскими решениями) можно будет ознакомиться, прочитав книжку «Школьник и компьютер — учимся друг у друга», которая должна выйти из печати в 1990 г.

Н. ВИЛЕНКИН, М. РАТИНСКИЙ, А. Ин

Информатика или программатика?

Идет пятый учебный год существования в программе средней школы предмета, который имеет совершенно необоснованное название — основы информатики и вычислительной техники. Средняя школа вообще имеет цель ознакомить учащихся с основами наиболее важных дисциплин и ни в коей мере не предполагает достижения фундаментальных знаний в какой-либо из областей. Все изучаемые в школе предметы носят краткие, но вполне исчерпывающие наименования: метематика, физика, биология, химия и т. д.

Введение в программу школьного обучения предмета со столь необоснованным названием позволило сместить акценты в обучении от информатики, которая, по нашему мнению, должна являться одной из базовых дисциплин в средней школе, к программированию, т. е. технике программирования на конкретном языке программирования.

Кроме того, на правильном становлении школьного курса информатики сказалось отсутствие достаточного материального обеспечения (технической базы, утвержденной программы и учебника по предмету).

При введении в школу столь важного предмета не была проведена тщательная и глубокая проработка собственно проблем информатики, а также возникающих в связи с этим психолого-педагогических проблем.

Простые в обращении современные компьютеры могут применяться для решения самого широкого круга информационных задач во всех сферах человеческой деятельности, включая и область образования. С этой целью необходимо предусмотреть психолого-педагогически оправданную систему разработки и создания баз данных и других сервисных средств для работы с ними непрофессиональных пользователей.

Базы данных необходимо создавать с учетом потребностей всех ступеней управления учебной деятельностью, начиная от системы «учитель — ученик» и кончая системой «министерство — отделы народного образования». Это позволит освободить большую часть сотрудников народного образования от рутинного умственного труда и направить их интеллектуальный потенциал на решение основной проблемы обучения, а именно — научить учащегося думать, рассуждать, творить.

В действующем курсе школьной информатики, а также в программах, рекомендованных для подготовки различных категорий работников просвещения, с информационной культурой и ее основной составляющей — структурой данных дело обстоит значительно хуже, чем с вопросами, относящимися непосредственно к алгоритмизации и программированию. Практически не дается никаких сведений о поисковых, справочных, экспертных и других системах. А ведь это именно то, с чем чаще всего сталкиваются в повседневной жизни работники сферы образования и будут встречаться сегодняшние ученики в будущем. В обязательном порядке необходимо учить тому, как с помощью различных типов данных можно представить объекты реального мира. Ведь все задачи информатики, связанные с процессами сбора, хранения и преобразования информации осуществляются лишь посредством работы со структурами данных.

В качестве основных проблем обучения ставились способ описания алгоритмов и выбор языка программирования. Сколько-нибудь полноценное решение этих вопросов окажется возможным при решении более глубоких проблем, а именно:

какова цель обучения информатике в массовой школе;

какой уровень владения приемами алгоритмизации и программирования в массовой школе следует считать обязательным, а какой факультативным;

в каком объеме и где будут использовать знания по информатике выпускники массовой школы в практической деятельности;

какими должны быть межпредметные связи курсов информатики с другими предметами школьной программы;

какими разделами информатики и в каком объеме должны владеть различные катего-

рии работников народного образования.

В общеобразовательной школе следует обучать информатике в более широком аспекте, чем это трактуется рекомендуемыми на сегодняшний день программой и пробными учебными пособиями. По нашему мнению, не совсем верно трактовать компьютерную грамотность лишь с точки зрения общения с компьютером на том или ином уровне. Основной задачей школьного курса информатики, по нашему мнению, является развитие у учащихся структурно-логического мышления. Поэтому весьма существенной проблемой, требующей глубокой проработки и изучения, является анализ взаимосвязей курса информатики с другими предметами школьной программы, т. е. определение сфер прикладного применения информатики.

Необходимо разработать принципиально новые программы практически по всем предметам школьного обучения с учетом возможностей современной информационной технологии и перспектив развития. При этом следует в обязательном порядке предусмотреть первоначальное ознакомление с некоторыми вопросами информатики в ряде предметов, изучаемых в V—VI классах (в первую очередь в математике), что позволит подготовить учащихся к изучению информатики. Знания, получаемые при изучении информатики, должны, в свою очередь, использоваться для более углубленного изучения других предметов.

Должен быть изменен подход и к самому содержанию школьных программ. В них должны быть определены уровни знаний, умений и навыков, а также и перечень основных разделов и тем. Но установление конкретного количества часов, отводимого на изучение того или иного раздела (темы), должно быть прерогативой учителя.

Обучение необходимо основывать на умении работать с разнообразными базами данных, что позволит отказаться от запоминания обширных, но впоследствии ненужных сведений.

Ученик к окончанию школы должен быть ориентирован в вопросе, где и как ему лучше использовать свой интеллектуальный потенциал на благо общества.

Ключевыми фигурами в решении столь важного вопроса являются в первую очередь учителя, а также работники народного образования, организующие и контролирующие учебно-воспитательный процесс. Поэтому очень остро стоит вопрос о своевременной и целенаправленной подготовке различных категорий работников сферы просвещения к практической деятельности в условиях применения новой информационной технологии.

Подготовку целесообразно планировать и проводить не по вопросам обучения некоторым основам программирования или алгоритмизации, которые являются лишь составными частями информационной культуры, а по вопросам, связанным прежде всего с управлением учебной деятельностью в условиях применения новой информационной технологии.

Учитель должен обладать широкими познаниями в области готовых программных средств и методов обработки различных видов информации, уметь спроектировать и составить на основе готовых пакетов или систем проектирования несложные программные системы для решения своей задачи.

Современные компьютеры обладают рядом преимуществ перед традиционными средствами обучения, что делает их более привлекательными для использования в учебном процессе. При этом надо отметить, что не только предметы естественно-математического цикла допускают создание учебных программ для ЭВМ, они могут быть созданы и по гуманитарным предметам. Усвоение большого объема цифрового и графического материала в форме активного диалога с ЭВМ, несомненно, улучшает качество обучения и делает учебу для ученика более интересной. Однако разрабатываемые и рекомендованные для использования в учебном процессе программы, как правило, типа «тренажер» и «имитатор» не имеют должного психолого-педагогического обоснования и рассчитаны на существующие методики.

При всех преимуществах компьютера перед другими средствами обучения он должен использоваться только в том случае, если его применение оправдано как методически, так и экономически. Компьютер в школе, на наш взгляд, необходимо рассматривать в основном в качестве средства обучения, а не предмета изучения.

Последующие уровни компьютерной грамотности фактически имеют отношение к профессиональной подготовке, тесно связанной со сложной программной продукцией и техникой будущего, поэтому в данной статье опускаем их содержание.

С целью реального внедрения новой информационной технологии обучения в практику массовой школы необходимо решить задачу правильной организации и проведения дифференцированного повышения квалификации по информатике сотрудников всех уровней системы народного образования.

Учитывая возможности современной информационной технологии, основное внимание при повышении квалификации должно быть уделено использованию при управлении учебной деятельностью имеющихся программных средств по обработке данных.

А. УВАРОВ

ЭВМ в американской школе сегодня

Компьютеры и интерактивные технологии

Четверть века назад один из пионеров использования ЭВМ в образовании американский исследователь П. Суппес обещал, что пройдет не так уж много лет, и каждый школьник будет пользоваться компьютером для обучения [1]. Об этом прогнозе стоит вспомнить сегодня, чтобы правильно оценить современное состояние и ближайшие перспективы внедрения новой информационной технологии в американской школе. За прошедшие четверть века на теоретические и экспериментальные исследования в этой области были израсходованы миллиарды долларов. Вместе с поколениями ЭВМ менялись взгляды на способы использования вычислительной техники в образовании, на педагогические задачи, которые она позволяет решать. Сегодня обычно говорят не о компьютеризации школы, а о распространении «интерактивных технологий» [2].

Термин «интерактивные технологии» используют для всех информационных технологий, обеспечивающих быстрый адекватный ответ на воздействие человека. Взаимодействие может происходит как между человеком и машиной (например, при использовании обучающих программ), так и между людьми, использующими новые средства телекоммуникации (электронная почта, телеконференции, обучение на расстоянии с помощью телемостов и т. п.). Интерактивные технологии включают в себя множество различных средств, однако их основу составляет обработка данных с помощью ЭВМ. Тем самым ЭВМ оказывают определяющее воздействие на развитие всех информационных технологий и их использование в образовании.

Об оснащенности американских школ компьютерами

В США, как и в других странах, средства вычислительной техники распределены по школам весьма неравномерно. Есть немало школ, где ЭВМ практически включены в учебный процесс и помогают ребятам осваивать счет, чтение и письмо (разработанный фирмой IBM для начальной школы курс «Чтение через письмо» пользуется растущей популярностью). В некоторых отдаленных районах повседневным учебным средством становятся телемосты, позволяющие одаренным учащимся из американской глубинки изучать дисциплины по выбору под руководством высококвалифицированных городских педагогов. Все больше школьников проводят лабораторные работы с применением компьютерных лабораторий (автоматизированных экспериментальных установок) при изучении естественнонаучных дисциплин, используют машинные модели в курсах истории и современной политики. В некоторых школах уже начинают применять видеодиски и системы интерактивного видео. Есть немало школ, где созданы компьютерные лаборатории на 20—30 рабочих мест, и каждый учащийся работает с ЭВМ от 20 мин до нескольких часов в неделю. Существуют экспериментальные школы, в которых каждый ученик имеет практически неограниченные возможности использовать ЭВМ в школе и дома. И вместе с тем немало школ, где ученики практически не имеют доступа к ЭВМ.

Массовое оснащение американских школ компьютерами началось около десяти лет назад. В 1981 г., когда в США впервые начали регулярно собирать данные об устанавливаемых в школах ЭВМ, компьютеры

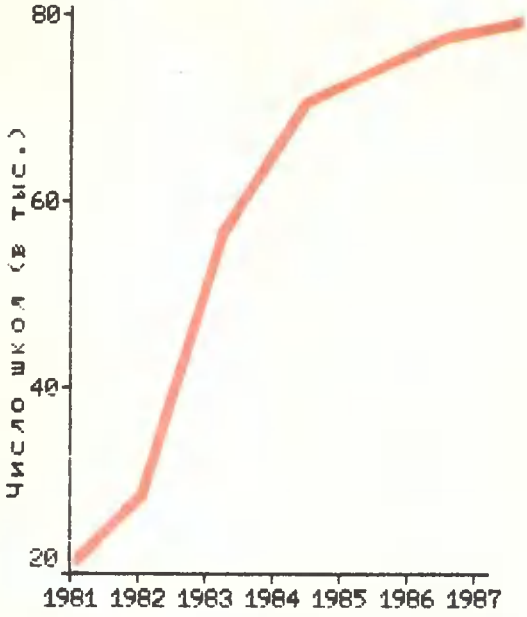


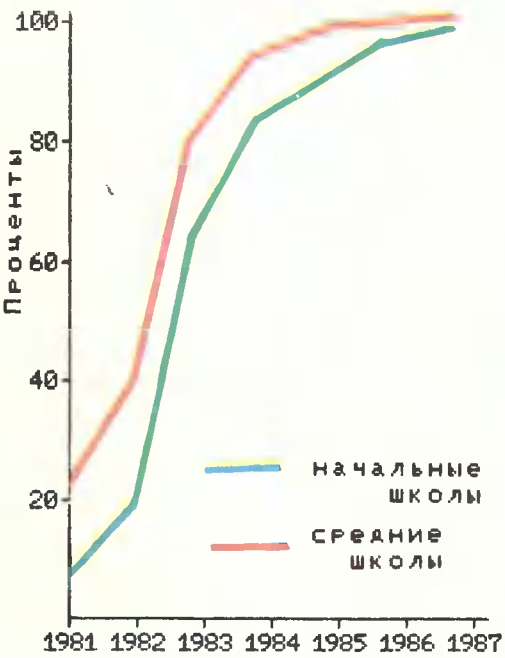
Рис. 1. Общеобразовательные школы США, оснащенные компьютерами

имелись лишь в 15 тыс. школ (далее без ссылок используются данные из [2]). Сегодня более 95 % из 81 тыс. общеобразовательных школ США имеют хотя бы одну ЭВМ (см. рис. 1). Менее чем за десять лет ЭВМ стали в школе повсеместно распространенным инструментом. Как следует из результатов обследования, выполненного в 1985 г., не более 5 % всех установленных в школах компьютеров оставались неиспользованными [3].

Примечательно, что начальные школы, где до начала 80-х гг. практически не было вычислительной техники, начали широко приобретать и устанавливать ЭВМ для учебных целей (см. рис. 2).

Массовая установка учебных ЭВМ в американских школах происходит на фоне продолжающегося улучшения материальной базы учреждений образования. Соотношение числа школ, оснащенных ЭВМ и кассетными видеомагнитофонами, приведено на рис. 3.

Рис. 2. Доля общеобразовательных школ США, имеющих по крайней мере один компьютер



Доступность ЭВМ для учащихся

Сегодня в американской школе один компьютер в среднем приходится на 30 учащихся, что теоретически позволяет каждому школьнику работать с ЭВМ в среднем около

Рис. 3. Доля общеобразовательных школ США, оснащенных компьютерами и видеомагнитофонами

Таблица 1

Распределение школ по числу установленных ЭВМ в 1987/88 учебном году

Виды школ	Количество компьютеров на школу				
	1	2—5	8—10	11—20	более 21
Начальные школы	5388	13 164	13 059	12 672	3900
Средние школы	694	2984	4208	7322	11 912

1 ч в неделю. Однако на практике все выглядит иначе из-за неравномерного распределения компьютеров по школам (см. табл. 1).

Несмотря на массовые закупки вычислительной техники, большинство школ в США пока не могут организовать работу школьников с ЭВМ в составе класса (см. табл. 2).

Таблица 2

Распределение школ по числу учащихся, приходящихся на один компьютер в 1987/88 учебном году

Количество учащихся на один компьютер	Школы (в процентах)		
	Начальные	Неполные средние	Средние
1—29	34,5	43,4	53,1
30—59	33,3	34,3	31,4
60—89	14,4	10,9	8,2
90—119	7,1	4,8	3,4
Более 120	10,7	6,6	3,9

Доступность компьютеров для учащихся зависит и от размера школы. Маленькие школы имеют обычно меньше установленных ЭВМ, чем большие, однако в маленьких школах учащиеся имеют больше возможностей работать с ЭВМ (см. рис. 4).

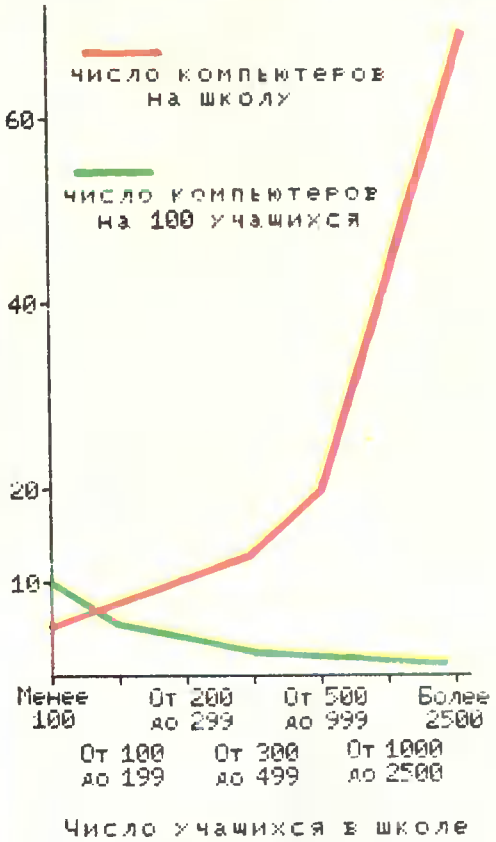


Рис. 4. Связь между величиной школы и доступностью ЭВМ для учащихся

Педагогические программные средства

Использование ЭВМ в школах определяется не только наличием технических средств, но и доступными педагогическими программными средствами (см. табл. 3). Количество предлагаемых на рынке учебных программ быстро возрастает.

В 1985—1987 гг. было создано больше педагогических программных средств, чем за все предыдущие годы (см. рис. 5). Распределение программных средств по способам их использования в школах приведено в табл. 4.

Некоторые педагоги скептически относятся к играм, в то время как другие уверены в их потенциальной педагогической ценности. Наибольшую популярность в школах начинают приобретать программные средства обще-

Таблица 3

Распределение педагогических программных средств по учебным предметам*

Область применения педагогических программных средств	Количество программ
Общего применения	427
Информатика	331
Английский язык	894
Иностранные языки	356
Математика	1971
Обучение чтению	869
Естественные науки	1148
Общественные науки	565
Прочие области применения	1329

* Многие программы относятся к нескольким учебным предметам, и в списке они отнесены к ближайшей области применения. В категорию «прочие» отнесены программы по 13 учебным предметам: автодело, слесарное дело, токарное дело, домоводство, делопроизводство и т. п.

Таблица 4

Распределение педагогических программных средств по способам их использования*

Способы использования педагогических программных средств	Количество программ
Упражнения на запоминание	1107
Практическая обработка навыков	3708
Обучение	2447
Демонстрация понятий	216
Разработка понятий	270
Проверка гипотез	91
Педагогические игры	1425
Моделирование	669
Инструментальные программы	807

* Многие программы включают в себя несколько способов использования.

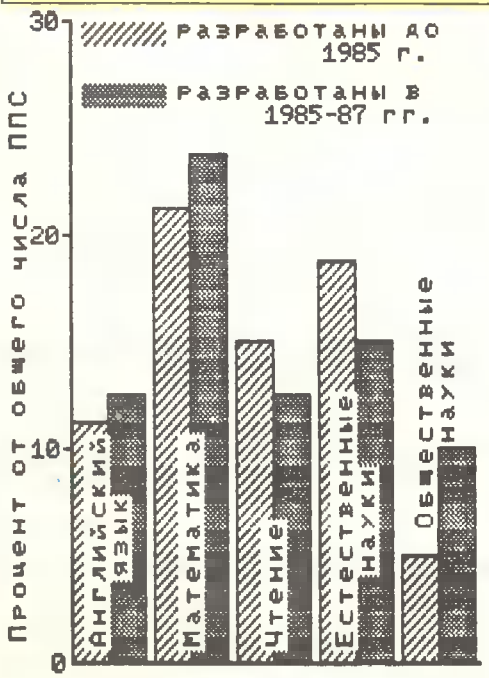


Рис. 5. Производство педагогических программных средств по основным предметам учебного курса

го применения. Так, редактор текстов становится сегодня повседневным инструментом школьника, который используется на уроках, при подготовке домашних заданий, во внеклассной работе. Сегодня школы приобретают прежде всего интегрированные системы, включающие текстовый процессор, систему управления базой данных и электронные таблицы.

Качество программных средств, их педагогическая эффективность сегодня находятся в центре внимания американских педагогов. В результате предпринимаемых усилий качество программ быстро повышается (см. рис. 6). Этому, в частности, способствует увеличение ассигнований, выделяемых централизованно на исследования в области педагогической технологии и использования ЭВМ. Только Министерство обороны США ежегодно затрачивает более 200 млн. долларов на исследования в области использования интерактивных технологий в образовании. Около 30 млн. долларов в год тратят федеральные гражданские агентства в США на исследования, связанные с использованием информационных технологий в общеобразовательной школе. Например, Национальный научный фонд США на исследования в области педагогической технологии, разработку учебных материалов, переподготовку и повышение квалификации учителей израсходовал в 1987 финансовом году 11,8 млн. долларов. (В 1989 г. на эти цели предполагалось выделить 19,1 млн. долларов.)

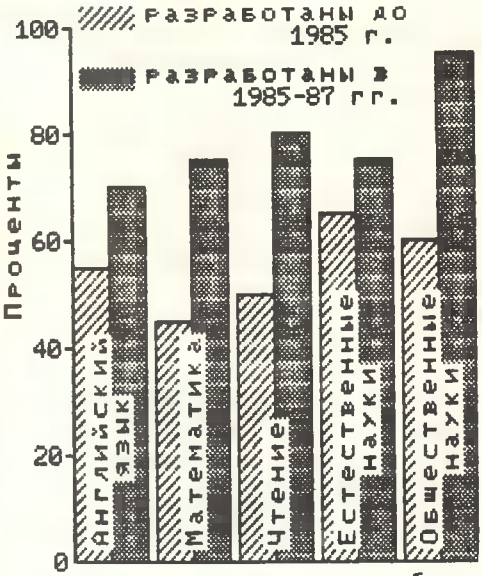


Рис. 6. Изменение качества разрабатываемых педагогических программных средств

Литература

1. *Суппес П.* Образование и вычислительные машины. Информация. М.: Мир, 1968.

2. U. S. Congress. Office of Technology Assesment. Power On Neu Tools for Teaching and Learning. OTA-SET-379. Washington: DC: U. S. Government Printing Office. September. 1988.

3. Instructional Uses of Shool Computers: Reports from the 1985 National Survey. Issue No 1. June 1986.

Обучающая система DIL

Министерство образования провинции Квебек (Канада) организовало исследование возможностей применения компьютерной обучающей системы DIL в детских садах. Система DIL предназначена для детей раннего возраста (2,5—7,5 лет) и детей с различными видами отклонений в развитии [1, 2].

При постановке эксперимента исследователи исходили из теоретических положений А. Мишле, Президента международного совета по детской игре, который считает, что для развития мышления ребенка необходимо учитывать опасность разрыва между его логическим мышлением и сенсорным опытом, строить обучение в направлении «конкретное — абстрактное — конкретное». По мнению А. Мишле, это лучший путь развития способностей к решению задач.

Исследователи ставили перед собой следующие цели:

изучение эффективности применения компьютеров в качестве средства развития способностей по решению задач;

изучение возможностей создания в учебной игровой деятельности ситуаций, требующих решения задач;

выделение видов задач, которые могли бы быть поставлены самими детьми.

Исследование проводилось в детском саду с детьми 5—6 лет. Компьютерная установка располагалась в комфортабельной части классной комнаты.

Одна из педагогических целей — сделать компьютер частью обычной классной жизни, предусмотреть такие докомпьютерные виды деятельности ребенка, которые должны подготовить его к работе с компьютером, создать условия для практического применения приобретаемых умений, а также для развития его творческих способностей.

Для включения компьютера в учебную деятельность возможны два подхода. Первый предусматривает структурирование преподавателем всех деятельностей ребенка (докомпьютерной, компьютерной и посткомпьютерной) вокруг какой-либо определенной темы, например: «Зоопарк», «Взморье», «Компьютер». Второй — вокруг определенного понятия или умения, например: «Распознавание образов», «Сопоставление».

Важным моментом является то, что начинать работать с компьютером ребенок должен на доступном ему уровне, который позволяет, с одной стороны, осуществлять взаимодействие с компьютером, а с другой — продвигаться в этой деятельности за счет экспериментирования с конкретными трехмерными объектами; с символьными представлениями (рисунками и картинками) путем упражнения и выполнения действий с карандашом и бумагой.

Обучающая система DIL включает специализированные аппаратные и программные средства. В состав аппаратных средств входит сенсорный планшет, позволяющий пользователю с помощью специальных дидактических карт управлять изображением на экране компьютера или изображением, спроецированным на настенный экран. Планшет совместим с ПЭВМ типа IBM PC и Apple. В состав комплекса входит печатающее устройство, цветное или черно-белое.

Дидактические карты представляют собой прямоугольные пластины. На одной стороне пластины размещен дидактический материал в виде красочных картинок. На полях расположены пиктограммы, предназначенные для управления, например, принтером. На обратной стороне вместо дидактического материала может находиться схема алфавитно-цифровой клавиатуры. Каждой дидактической задаче соответствует определенная дидактическая карта.

Психолого-педагогическое обеспечение системы DIL направлено на формирование у детей элементов компьютерной грамотности, знаний о предметных связях, на развитие основных интеллектуальных операций и некоторых навыков социализации. Каждая дидактическая программа соотносится с определенными аспектами психолого-педагогического обеспечения. Например, программа «Мой сад» предполагает формирование понятий из области естественных наук и математических навыков, развитие словарного запаса ребенка. Кроме того, эта программа направлена на формирование стратегии «проб и ошибок», развитие ассоциативного мышления, умения решать задачи, а также развитие сенсорных способностей и таких интеллектуальных операций, как сериация, классификация. Программа способствует формированию координации «глаз — рука», приемов запоминания, играет большую роль в формировании умения работать индивидуально.

Приобщение к компьютеру через игру вызывало позитивное отношение к нему детей, удовлетворенность успехами, достигаемыми на глазах партнеров. Аспекты забавности, игровой динамики поддерживали внимание ребенка, обеспечивали его сосредоточенность и любознательность, необходимые для получения эффективных результатов компьютеризированных деятельностей. Бо-

лее того, испытуемые демонстрировали больше терпения и настойчивости, чем это свойственно детям этого возраста. Ни один из них не бросил работу с компьютером до завершения игры. Все стремились к повторению игры столько раз, сколько потребуется для безошибочного результата. Дети работали с компьютером строго в соответствии с их желанием, играли в те компьютерные игры, которые сами выбирали. Каждая игра рассчитана примерно на 3 мин. Общий сеанс игры с компьютером — не более 6—8 мин в день.

Система позволяет работать с так называемой электронной книгой. В этом случае знакомство с сюжетом сопровождается мультипликацией и музыкой. Дети могут активно участвовать в сюжетном действии, вмешиваться в него и получать твердые копии картинок, иллюстрирующих сюжет, а также созданных разработчиками или самими детьми программ.

Особый интерес представляет роль взрослого. Его основная цель — обеспечить максимальную самостоятельность ребенка как в постановке задачи, так и в поиске путей ее решения. Взрослый должен помогать вербализовывать и выражать взаимосвязи между объектами, которые воспринимает или которыми манипулирует ребенок. Учитель предоставляет ребенку возможность реализовывать свои индивидуальные проекты и подходы и анализировать свои решения. В этом контексте, по мнению М. Лабелль, руководителя работ по созданию системы DIL, у ребенка развиваются способности к анализу собственных рассуждений и проверке их посылок. Важную роль играют поощрения со стороны учителя и вербализация вновь приобретенных понятий и навыков.

Как показали результаты исследования, компьютер можно рассматривать в качестве инструмента поддержки тех дидактических материалов и стратегий научения, которые в настоящее время используются в детских садах.

По мнению исследователей, ввод компьютера в систему обучения в детском саду — необратимый феномен в том смысле, что компьютеры становятся обычным явлением в жизни современного общества и уже стали частью обычного окружения детей.

В ходе эксперимента, который продолжался в течение года, наблюдалось положительное отношение к применению компьютера не только у воспитателя, но и у родителей испытуемых.

На основе анализа полученных в исследовании данных сформулированы следующие выводы:

с помощью компьютеров можно создавать учебную среду, характерными чертами которой являются исследование и открытие;

чем больше компьютер стимулирует изобретательность, тем яснее ребенку взаимосвязи в окружающем мире;

использование компьютера позволило детям обнаруживать связи между окружающей средой, знаниями и информацией;

обучение детей раннего возраста с помощью системы DIL создает благоприятные условия для использования компьютеров на более поздних стадиях обучения.

Материал подготовлен
В. БОНДАРОВСКОЙ, И. НИКОНОВОЙ
г. Киев, НПО «Горсистемотехника»

Литература

1. *M. L. Esperance-Labelle.* Educational Computer Activities and Problem Solving at the Kindergarten Level // Proc. of Third International Conf. «Children in the Information Age». Prepr. 1. Sofia. 20—23 May, 1989. P. 386—399.

2. The DIL Learning System: The WonderWorker / PowerPad and DIL's Wide Assortment of Electronic Books and Activities. Saint-Foy, Quebec. Canada, 1989.

Ярмарка-2000

Полное название этого состоявшегося в ноябре на ВДНХ СССР мероприятия звучит так: 1-я Международная выставка-ярмарка «Электроника-2000». Первая потому, что ее организаторы (советско-польское СП «Интермост» и Культурно-информационный центр газеты «Московские новости») намерены устраивать выставки-ярмарки под таким названием ежегодно. После двухтысячного года придется, конечно, придумывать что-нибудь новое, но до той поры далеко. Впрочем, название не соответствует содержанию уже сейчас: большинство выставленных экспонатов не модели будущего, а электронный ширпотреб (последнее слово употреблено отнюдь не в ругательном смысле), и перспективой следующего десятилетия они являются только для нас.

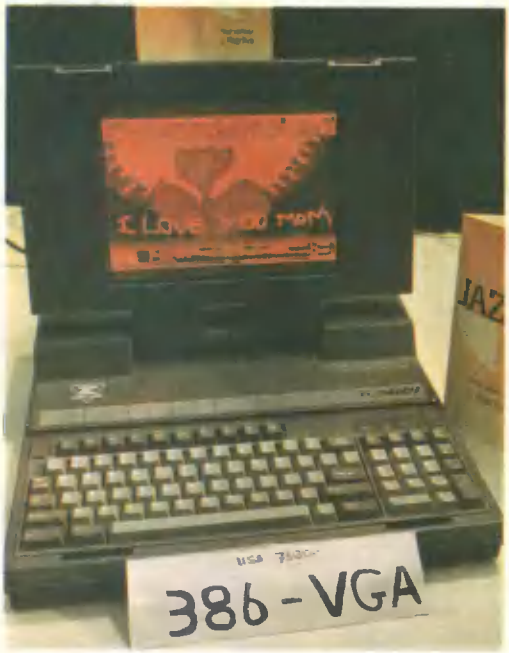
Основное впечатление: Польша — новое окно в Европу, а заодно и в Азию. Подавляющее большинство заграничных участников — польские предприятия, фирмы, кооперативы, акционерные общества и чуть ли не индивидуалы. Подавляющее большинство товаров — западного и восточноазиатского производства (Rank Xerox, Canon, Tochiba, Mitsubichi, Thomson, Centronics, Seagate, Sanyo, Olivetti, Hitachi, Seikosha, Minolta, IBM, Star, Hewlett-Packard, Gold Star, Epson, Sony, Samsung...). Похоже, польское правительство (в отличие от нашего) недооценивает опасности торгово-закупочной деятельности. Впрочем, может быть, дело просто в том, что польские торговцы получают от своих операций некую выгоду, которой и делятся со всем обществом в отличие от наших многострадальных кооператоров, получающих преимущественно синяки и шишки (не потому ли многие из них не склонны делиться?).

Кстати, о торгово-закупочной деятельности. В редакцию журнала начали приходить письма с вопросами о ценах на внешнем рынке. Так вот: цены эти на один и тот же товар могут различаться в 3—4 раза. Самую низкую цену назначает изготовитель, но купить у него обычно трудно: он сидит где-нибудь в Калифорнии, представительства в Москве не имеет и служащих со знанием русского языка не держит — ни к чему. А главное, продает компьютеры только партиями: 1000 штук — цена побольше, 10 000 — поменьше... Покупают у него товар, как правило, оптовые торговцы, имеющие сеть складов по всему миру, налаженные контакты с транспортниками и т. п. Цены у них уже повыше. Следующие в цепочке рады и отдельному покупателю, однако заплатить им надо уже минимум в два раза больше, чем получил за тот же компьютер производитель. Так что не удивляйтесь тому, что вроде бы одинаковые ПЭВМ очень по-разному стоят у разных продавцов. Осложняет ситуацию и то, что вроде бы одинаковые ПЭВМ могут различаться комплектацией (нет монитора — есть — CGA — EGA — VGA...), малозаметными параметрами (винчестер с временем доступа 40 или 28 мс), надежностью, авторитетом производителя... Для примера приводим цены на PC-совместимые компьютеры с процессором Intel 8088 (обычные параметры: тактовая частота 4,77 МГц, ОЗУ 512—640K байт, дисковод 360K байт, винчестер 20M байт).

«Альфа» (совместное предприятие): без винчестера — 750 долларов или 13 000 рублей;

«Мегамекс» (Индия): с винчестером 40M байт и монитором CGA — 800 долларов;

«Комтех» (Польша): тактовая частота



Плазменный дисплей «Jazzy»

10 МГц, без винчестера — 780 долларов или 15 600 рублей;

«Кеми Интер» (Польша): 4,77/10 МГц, с двумя дисководами, монохромным монитором, без винчестера — 19100 рублей, с винчестером 20М байт — 27 000 рублей;

«ЛХК Электроникс Сервис» (Польша): 4,77/12 МГц, без монитора и винчестера — 580 долларов, винчестер 40М байт — 400 долларов;

«Витабест» (Польша): 4,77/12 МГц, с двумя дисководами и монитором CGA — 31 110 рублей;

«ИДМ» (Польша): 4,77/10 МГц, с монохромным монитором и принтером 10 дюймов — 23 500 рублей;

«Текома» (Польша): 4,77/12 МГц, с двумя дисководами, монохромным монитором, без винчестера — 19 670 рублей, с монитором CGA — 35 840 рублей, винчестер 40М байт — 11 750 рублей;

«СиЭс трейдинг компани» (Австрия): 10 МГц, с двумя дисководами, без монитора и винчестера — 690 долларов, монитор EGA — 530 долларов, винчестер 40М байт — 600 долларов;

«Джази» (Тайвань): без монитора и винчестера — 910 долларов, с винчестером 20М байт — 1390 долларов, монитор монохромный — 188, EGA — 600, VGA — 735 долларов.

Не обманывайтесь ценами в рублях — рублей много. Обычно под «рублями» подразумеваются либо СЭВовские рубли, либо бартерные,— иначе говоря, советские товары на соответствующую сумму. Обыкновенный советский рубль может служить платежным средством крайне редко.

Еще одна любопытная для нас особенность цен — их склонность меняться со временем, причем, несмотря на хорошо нам известную западную инфляцию, в сторону уменьшения.

«ABC Computersystems» (Западный Берлин) продает, в частности, принтеры «Star». Сравнивая ценники разных месяцев, можно получить впечатление о динамике цен на западном рынке. В таблице перечислены все модели, продающиеся в ноябре.

«Компан» (Польша) — фирма-производитель. Здесь делают дискеты, сами же их и продают. Разговор о цене ничего не дал — «Всё зависит от контракта!» Большой заказ (а большой для фирмы миллион дискет) — малая цена, маленький заказ — большая (за 1000 штук, например, 10 000 рублей). За доллары не продают — невыгодно, резко повышаются налоги. За рубли тоже — не разрешает польское правительство. Основные платежные средства — переводные рубли и товары. От вида оплаты цена также зависит довольно сильно: «Хороший бартер — хорошие цены!» А хороший бартер — это печатная бумага, хлопок и х/б ткани, сухое и столовое вино, электрокабели... Одним словом, дефицит. «Конечно, можно взять и товар попроще, но только чтобы много. У вас очень тяжело торговать. Оформить один контракт сколько надо сил! Для маленького контракта и стараться не стоит. А вот если бы покупатели собрались вместе, объединились... С ними гораздо проще стало бы иметь дело. И выгода была бы и нам, и им». Есть о чем подумать.

Дочерняя фирма «Меримпекс» записывает на компановские компакт-кассеты курсы иностранных языков («Можем разработать

Модель	Цены 1989 г. (в марках ФРГ)		
	Февраль	Октябрь	Ноябрь
LC-10	450	350	350
LC-15	—	750	750
LC-10, цветной	550	450	450
LC24-10	770	770	690
LC24-15	—	1050	990
PR-10	—	900	900
PR-15	—	1100	1100
XB24-10	—	1100	1100
XB24-15	—	1350	1350
LS-08, лазерный	3900	3900	3900
DP-8340	550	972	972

курс любого языка по вашему заказу!»). Хотя готовые изделия и выгоднее экспортировать, чем сырье, но еще выгоднее «нагружать» их продуктами интеллектуального производства. Впрочем, все это мы знаем — в теории...

Чрезвычайной популярностью пользовался стенд польской фирмы «Про-Инфо», он же — книжный киоск. Руководства по AMIGA DOS, AUTO CAD, dBASE III, Турбо-Паскалям множества разновидностей, принтерам LC-10, NX-15, SG-15, NL-10, системе FRAMEWORK, MS DOS, OS/2, XENIX, Турбо-Си и Турбо-Прологам... И всё за рубли. Наличные. Подходи и покупай! Конечно, подходили очень многие. Покупали, впрочем, не все. Ксерокопии фирменных руководств (а как, кстати, насчет охраны авторского права?) на английском и польском языках стоят по 50—100 рублей. С закрытием выставки торговля не прекратилась — у «Про-Инфо» есть компаньоны в Москве и Киеве.

Но вот и советские экспонаты — и сразу учебная компьютерная техника. «Агат»! Четыре зала, машины «заряжены» игровыми программами. Однако народ на выставке, видимо, серьезный — вскоре два зала пришлось закрыть, да и оставшиеся пустуют.

НПО «Квазар» демонстрирует учебный телевизионно-дисплейный класс (УТДК) «Квазар», разработчик — Хмельницкое специальное конструкторское бюро «Таймер». УТДК предназначен для автоматизированного обучения в вузах, техникумах, средних общеобразовательных школах, ПТУ. Представляет собой распределенную вычислительную систему из центральной ПЭВМ типа IBM PC и 12—24 терминалов, состоящих из дисплея и клавиатуры. Одновременно центральная машина может обслуживать 4—5 пользователей в многопользовательском режиме, остальным учащимся приходится лишь наблюдать за работой товарищей или педагога. Стоимость класса — 40 000 рублей без центральной ПЭВМ, которая стоит еще около 80 000 рублей, программное обеспечение — стандартное для IBM PC.

Книжный киоск «Про-Инфо»

Среди предлагаемых услуг — продажа технической документации, предоставление услуг по внедрению, разработка вариантов УТДК «Квазар» (в зависимости от типа центральной ЭВМ и программного обеспечения) с учетом требований заказчиков.

Нечто похожее предлагает польская фирма «Микроб» (интересно, что они хотят сказать своим названием?). Там тоже считают, что PC XT/AT/AT-386 в виде персональной ЭВМ, рассчитанной на выполнение одновременно только одной задачи, неэффективны в конторских и учебных применениях, для которых гораздо более приемлемы сети ПЭВМ. Однако сеть стоит немалых денег, не говоря уже о входящих в нее ПЭВМ. Обойти это обстоятельство позволяет использование конфигурации, в которой к центральной ЭВМ класса AT или AT-386 с помощью оптических волокон подключены более простые терминалы. Приобретенные таким образом рабочие места в среднем вдвое дешевле рабочих мест в обычной локальной сети.

Терминал «PC-Граф» этой фирмы дает возможность работать с программами на центральной машине, к которой можно подключить до 16 терминалов; главная проблема их совместной работы — время ожидания. Центральная машина не совсем обычна: ее плата точно такая, как у IBM PC AT, но на месте процессора стоит не Intel 80 286, а Intel 80386SX с тактовой частотой 16 МГц; его внутренняя 32-битовая архитектура совпадает с архитектурой процессора 80 386, а «снаружи» он выглядит как 80286. В результате бывшая AT с новым процессором получает возможность охранять общую память от несанкционированных изменений удаленными пользователями. Для профессионала время ожидания ответа центральной ПЭВМ приемлемо при одновременной работе четырех терминалов «PC-Граф». В Польше такая система используется для обучения в двух университетах. При использовании в качестве конторской (или, говоря высоким слогом, АСУшной) системы в бухгалтериях, на складах, в небольших предприятиях вполне комфортный режим работы обеспечивается при подключении восьми и даже более терминалов.

Для терминала фирма закупает на Западе процессоры и еще несколько микросхем, остальная начинка польская; центральная машина целиком западная, польская только

сборка. Можно было бы поставить какие-то компоненты и польского производства или, например, тайваньского, но фирма считает, что качество важнее цены.

Стоимость системы (центральная ПЭВМ и четыре «РС-Граф» с процессорами Z-80 — такими же, как стоящие в наших школьных «Ямахах») — 2987 долларов; на рубли фирма продавала бы ее по цене 56600, если бы не правительственный запрет.

В числе услуг — бесплатная консультация, комплектование оборудования, программное обеспечение, монтаж и обучение персонала.

Страховое общество «Прогресс» (Москва) предлагает новые формы страховой защиты, значительно расширяющие действующую в стране государственную систему страхования. Оно страхует не только от стихийных бедствий и несчастных случаев, но и от действий третьих лиц и неблагоприятных хозяйственных обстоятельств, а также компенсирует затраты, возникающие вследствие ответственности по закону организаций (предприятий) и кооперативов за причиненный ими ущерб здоровью или имуществу третьих лиц. «Компьютерные» аспекты его деятельности — страхование от краж не только ПЭВМ (что обходится в 2 % стоимости компьютера в год), но и программ. Застрахованные программы будут защищены специалистами «Прогресса» от несанкционированного копирования; если незаконное распространение все же начнется, в действие вступят судебные пути защиты.

А недавно «Прогресс» выразил готовность страховать граждан от полтергейста. Похоже, о его деятельности нам еще придется рассказывать.

Восстановление взаимоотношений США с Китаем когда-то началось с настольного тенниса — поездки спортсменов стали предвестниками вояжей дипломатов. Не сыграют ли компьютеры такую же роль во взаимоотношениях СССР с Тайванем? Последний был представлен на «Электронике-2000» нидерландским филиалом (начинать надо осторожно!) фирмы «Jazzy Industrial Co., Ltd». Ее продукция — IBM-совместимые машины класса PC и PS/2, в том числе портативные. Последние производят очень приятное впечатление благодаря высокому качеству изображения плазменных (640×400 точек, 4 градации серого) и жидкокристаллических (640×480 точек, 32 градации серого) экранов. Оба дисплея вполне эргономичны, мало уступают электронно-лучевым.

Учебный робот «Торий»

Вот комплекс, привлекающий внимание необычной ценой. «Софтлан» (Польша) специализируется на изготовлении компьютерного телетайпного интерфейса, поэтому ПЭВМ продает только вместе с ним; однако «нагрузка» не страшна — это акционерное общество почему-то высоко ценит рубли, и PC XT (10 МГц, 640K ОЗУ, два дисковода по 360K, винчестер 20M) вместе с телетайпом продает за 7000 долларов или 25 000 рублей.

А вот нечто несуразно дорогое. Уфимское научно-производственное объединение им. С. М. Кирова демонстрирует ПЭВМ «Башкирия-2М»: 8-разрядный микропроцессор, 64K ОЗУ, 16K ПЗУ, псевдографика 128×72 точки. Цена без монитора — 1200 рублей; хорошо хоть не инвалютных!

Неисповедимы пути договорных цен. В магазине «Электроника» ДВК-3 в минимальной конфигурации стоит более 16 300 рублей, а учебный робот производства НПО «Торий» вместе с управляющим им ДВК-3 продается за 15 000 рублей, без ПЭВМ — за 2000. Объем выпуска, к сожалению, только 400 штук в год. Имеет набор инструментов (сверло, фреза, карандаш), может использоваться в качестве графопостроителя, имитировать металлообрабатывающий станок.

Уставшие от цен с множеством нулей, обратите внимание: кооператив «Медитекс» демонстрирует ПЭВМ «Камекс». Благодаря отбору наиболее качественных экземпляров микропроцессора КР580 ВМ80 гарантируется тактовая частота 2 МГц; ОЗУ — 32—48K, ПЗУ — 2—12K, цена — 400—450 рублей.

Конечно, не один лишь ширпотреб был на выставке «Электроника-2000». Будущее рождается сегодня, точнее, оно родилось вчера и не наступает только потому, что мы к нему не готовы.

Что вы скажете о словосочетании «врач-алгоритмист»? Профессия следующего века? А ведь еще лет 20 назад врач Л. Б. Наумов создал свой первый медицинский алгоритм — формализованную систему диагностики сердечных заболеваний (естественно, рассчитана она была на «ручное» использование, без ЭВМ). К сожалению, этой весточке из

будущего так и не удалось проникнуть в нашу повседневность. На практике она применялась лишь отдельными энтузиастами. Непривычно!

Это было вчера. А как сегодня? Увы, и теперь, будучи переведенной на ПЭВМ, она в невыгодном положении — в моде экспертные системы, наумовские алгоритмы кажутся старомодными... Однако это впечатление немедленно исчезает после знакомства с программной системой «Эскулап». Она может работать даже на БК, обеспечивает быструю и, главное, точную диагностику при минимальном (что тоже немаловажно) числе диагностических манипуляций (анализов, осмотров и т. п.). Экспертные же системы требуют значительных вычислительных мощностей, больших объемов входных данных для анализа, зачастую плохо справляются с задачей... «Старомодный» «Эскулап» выигрывает по всем параметрам, и здесь нет парадокса. Экспертная система способна самостоятельно находить закономерности, делать выводы, выносить заключения. Но не везде это нужно! Во многих областях деятельности накоплены исчерпывающие массивы знаний, которые не нужно повторно искать, а достаточно систематизировать и вовремя напоминать человеку. Классический пример — диагностика неисправностей автомобиля. Проведение ее экспертной системой — стрельба из пушки по воробьям. Гораздо эффективнее формализовать имеющийся у людей исчерпывающий опыт и заложить в ЭВМ его выжимку — справочник простой древовидной структуры типа «если — то — иначе».

Доктор медицинских наук, профессор Л. Б. Наумов проделал такую операцию во многих областях медицины, оптимизировал свои алгоритмы по скорости спуска к конечным вершинам, экспериментально доказал их эффективность (а она, например, в области кардиологии заметно превышает способности специализированных кардиологических бригад скорой помощи), и, несмотря на это, использование его системы не стало повсеместным, хотя и особых препятствий вроде нет, и противников не наблюдается... Что ж, подождем, когда она приобретет популярность за рубежом.

К. ШЕХОВЦЕВ

ЧТО МОЖЕТ ЭВМ?

В тысячу раз тоньше волоса

Если у вас есть пластинка кремния и вакуумная камера для напыления на нее различных материалов, то вам ничего не стоит изготовить однокристалльную ЭВМ. Для этого достаточно напылить на кремниевую подложку очередной слой, нанести на него фоторезист и засветить его через фотошаблон. Затем надо смыть незасвеченные участки фоторезиста, вытравить не защищенные ими участки металла и смыть его остатки. Проделав указанные операции с разными фотошаблонами необходимое число раз над разными слоями проводников, полупроводников и диэлектрика, вы получите необходимый вам персональный однокристалльный компьютер. Правда, если для засветки фоторезиста вы использовали обычный дневной свет, он будет не очень высокого качества.

Вы сможете повысить его быстродействие и значительно уменьшить размеры, если для засветки фоторезиста будете использовать не обычный, пусть даже самый коротковолновый ультрафиолетовый свет, а электронный луч диаметром всего в 0,25 микрон. Это позволит сократить ширину проводящих дорожек и величину других элементов схемы до 0,5 микрон, а это — и уменьшение размеров интегральной схемы, и сокращение длины проводников и паразитных емкостей, и, следовательно, повышение тактовой частоты микропроцессора.

Если же вы освоите рентгеновскую литографию, то вашим успехам позавидует даже Министерство обороны США, которое еще только планирует с помощью X-лучей освоить 0,10-микронную технологию для разработки сверхскоростных микропроцессоров с тактовой частотой свыше 100 Мгц.

Как изучать тему

Хотим поделиться идеей изложения тем, изучаемых в X классе: «Алгоритмы. Алгоритмический язык» и «Алгоритмы работы с величинами».

В курсе ОИВТ узловым моментом при изучении алгоритмического языка являются составные команды. Учителя области, используя в работе планирование материала, изложенное в журнале «Информатика и образование» (1987, № 5), сталкиваются с трудностями, связанными с разрывом в изучении составных команд и величин, исполнением алгоритма.

Этот вопрос обсуждался творческой группой учителей основ информатики и вычислительной техники. Было определено наиболее оптимальное, на наш взгляд, планирование материала, объединяющее темы «Алгоритмы. Алгоритмический язык» и «Алгоритмы работы с величинами»:

Понятие алгоритма. Формальное исполнение алгоритма. Общие правила алгоритмического языка	1 ч
Алгоритмы работы с графической информацией	1 ч
Величины. Заголовок алгоритма	1 ч
Промежуточные величины. Присваивание значений	1 ч
Исполнение алгоритма	1 ч
Команда ветвления. Отношения между величинами в качестве условий	2 ч
Команда повторения	2 ч
Решение задач. Самостоятельная работа	4 ч
Контрольная работа № 1	1 ч
	14 ч

По приведенному плану в 1988/89 учебном году были проведены экспериментальные занятия. Опыт показал, что такая последовательность занятий представляется более целесообразной.

Наша творческая группа хотела бы узнать мнения других учителей по вопросам преподавания информатики в школе. Может, и в других институтах усовершенствования учителей есть творческие группы. Мы были бы рады установить с ними контакты.

А. ЯЛТУХОВСКИЙ,
Т. ТРЕГУБ,
Запорожский областной институт усовершенствования учителей

Извините, Елена Константиновна!

Начну с того, что я — директор Программно-методического центра «Учебные компьютеры» Московского городского комитета по народному образованию (МГКНО), а Елена Константиновна — учитель информатики из Протвы. К нам приезжают многие, но она запомнилась более других, так как пришла не одна, а вместе со своими учениками (хорошие ребята!) Извиняюсь же я потому, что их школа получила новый КУВТ.

Они получили КВУ УКНЦ. Сейчас не время говорить о том, *что* это такое с точки зрения программиста, об аппаратных особенностях, о проблемах, связанных с эксплуатацией. Речь пойдет о другом. Представьте ситуацию: вы должны учить детей как программированию, так и грамотному использованию готовых программных продуктов, имея *только* интерпретатор Бейсика (олимпиадная задача: как устроено распределение памяти интерпретатором, если после удаления из работающей программы одной строки программе перестает хватать памяти?), работающий *только* в сети и не желающий работать на рабочем месте преподавателя (РМП). Все!

Вообще-то многие знали, что в школы будет поставляться КВУ УКНЦ, и прозревали нужду в программных продуктах. В частности, МГКНО совместно с Научным центром программных средств обучения (НЦПСО) и предприятием «ИнфоМир» силами профессиональных программистов создали пакет для поддержки курса информатики: игра-клавиатурный тренажер, исполнители, практикум по изучению архитектуры компьютера, комплект учебных миров (полная поддержка курса информатики по учебнику А. Кушниренко), МикроМир (организация файловой структуры и редактор текстов), сетевой монитор ВАМ (позволяет выполнять на РМУ практически любую программу, которую можно выполнить на РМП), интерпреватор LOGO (язык для обучения младших школьников, по мощности не уступающий LISPу), «Writer» (профессиональная система подготовки текстов).

Пакет включает также учебники и руководства пользователя в виде книг и брошюр и дополнен (на дискетах) большим количеством примером, шпаргалок и разного рода компонентов поддержки работы операционной системы и организации сети.

Почему же я не переписал Елене Константиновне все с дискет, не дал ей все книги? Книги стоят денег — это понятно. К сожалению, не все (особенно среди администраторов) понимают, что денег стоит и программный продукт. Существует также авторское право — в данном случае оно принадлежит НЦПСО и «ИнфоМиру», которые, вложив средства в разработку, хотели бы их оправдать и получить некую прибыль. Так что я не мог помочь Елене Константиновне, однако обнадежил, сообщив о планирующейся поставке пакета в школы РСФСР.

Дальнейшие события я изложу в виде обмена репликами (преимущественно между Минпросом РСФСР и НЦПСО), вполне передающими смысл происходившего, добавляя некоторые комментарии.

НЦПСО. Мы имеем в своем распоряжении вот такой пакет программных средств. Нужен?

МП РСФСР. Нужен!

НЦПСО. Договорная цена устраивает?

МП РСФСР. Устраивает! Подписывайте договор с нашей базой!

НЦПСО базе. Давайте подпишем договор.

База. Никаких договорных цен! Делайте калькуляцию!

НЦПСО. Сделали! Теперь хорошо?

База. Нет! Плохо! Дайте нам теперь письмо от МП РСФСР, что им это надо!

Параллельный диалог с учителем из N-ска (город не назван, так как приезжали из десятков городов — от Владивостока до Кишинева).

— Дайте пакет! Слухи ходят — у вас есть!

— Пакет денег стоит.

— Я куплю! Очень надо!

— Подожди. Зачем платить дважды? МП РСФСР централизованно закупит и всем раздаст.

— Не может быть!

— Да вроде уже договорились...

НЦПСО — МП РСФСР. Письмо дадите?

МП РСФСР. А как же! Нате!

НЦПСО базе. Теперь хорошо?

База. Нет! Мы, пожалуй, у МП РСФСР сами спросим.

База — МП РСФСР. Брать или не брать?

МП РСФСР. Вообще-то взять надо, но... У НЦПСО заключение экспертной комиссии есть?

НЦПСО. Московской есть!

МП РСФСР. А нашей, которая в НИИ школ?

Экспертная комиссия НИИ школ установила полную методическую целесообразность распространения продукта, но выразила неудовольствие тем, что расчет стоимости был произведен по цене разработки. НЦПСО произвел расчет по готовому продукту; цена осталась той же.

НЦПСО. Экспертизу и перерасчет произвели! Необходимый вам тираж изготовлен, штамп ОТК на продукте, хорошая полиграфия и т. д!

МП РСФСР. Молодцы! Но брать не будем. Дорого!

Речь шла о 3500 рублях за пакет. В патовой ситуации прикинули цену, не учитывающую накладные расходы и сводящую плановую прибыль к нулю. Получилось 2500 рублей.

МП РСФСР. Хорошо! Но брать все равно не будем. Программа не должна стоить больше 100 рублей. До свидания.

А Елена Константиновна уехала. И борется с самым плохим BASIC-интерпретатором в мире. И, вероятно, ждет того момента, когда из местного управления народного образования ей принесут вышеописанные программные средства. Но, похоже, не дождется. А если приедет ко мне, то получит отказ (право тиражирования принадлежит НЦПСО).

Зря я Вам (как и многим другим ходокам за программами) пообещал скорое облегчение жизни. Извините, Елена Константиновна!

С уважением
М. ПЕТРОВ

Совещание на заводе «Экситон»: перспективы БК

Потребности нашей страны в персональных компьютерах всех типов — от 20 до 60 млн. штук. Среди немногих предприятий, которые вносят реальный вклад в решение проблемы компьютеризации,— завод «Экситон» (г. Павловский Посад Московской обл.). Его продукция — «Электроника БК-0010» — хорошо знакома школьникам и учителям, студентам и научным работникам. Очереди в магазинах «Электроника» растут, спрос на БК-0010 и КУВТ-86 весьма велик. Каковы же перспективы производства? Этот вопрос обсуждался на техническом совещании пользователей отечественных комплексов вычислительной техники на базе ПЭВМ типа БК в ноябре 1989 г.

Перед началом работы участникам совещания были показаны новинки: БК-0011, переносные комплексы контроля и диагностики неисправностей, новые КУВТы. Здесь можно было увидеть промышленные, а не любительские мышь и джойстик к БК, а также... послушать музыку Баха — шестиголосный синтезатор на основе БК-0010 демонстрировали представители Музыкально-педагогического государственного института им. Гнесиных.

Затем состоялся разговор о заводских проблемах. На смену БК-0010 идет более мощный во всех отношениях БК-0011. Главное его отличие — 128К байт ОЗУ и контроллеры всевозможных внешних устройств. В новом КУВТе все рабочие места — и учеников, и преподавателя — будут выполнены на базе БК-0011, что существенно повысит надежность комплекса. Изменится и организация обмена информацией по сети, будет использован моноканал, сеть сможет обслуживать до 32 рабочих мест. Комплекс будет называться УКНЦ.01 (Учебный Комплекс — «Научный центр»), цена, как и у КУВТ-86, около 28 тыс. рублей за 12 РМУ и одно РМП. Большая часть программного обеспечения КУВТ-86 будет работать и на КУВТ УКНЦ.01.

Пока это, однако, лишь перспектива. На производство большого количества БК-0011 заводу просто не хватает ряда микросхем, поэтому для поддержания сегодняшнего уровня производства продолжится выпуск БК-0010. Казалось бы, можно продолжить и выпуск КУВТ-86 но завод перестал получать ДВК-3Ш, поэтому в качестве переходного варианта предлагается заменить ДВК на БК-0011 и выпускать под названием КУВТ-86М. Все БК с января 1990 г. будут выпускаться с новой клавиатурой — без «дребезга»; повысить эффективность компьютеров завод предлагает также за счет расширения поставок программного обеспечения.

Гости завода в выступлениях отметили возросшее качество БК, улучшение технологии производства, хотя клавиатура вызывает всеобщие нарекания. Говорили участники совещания и об опыте использования компьютеров завода «Экситон» во всех сферах образования: детских садах, общеобразовательных и музыкальных школах, институтах и военных училищах, с использованием как русского языка, так и других языков народов СССР. Имеется опыт применения БК на телевидении для создания компьютерных картинок и заставок, на производстве для управления и контроля работы устройств.

Отмечались положительные стороны массового использования однотипных компьютеров типа БК как в целях регионах (Латвийская ССР, Владимирская обл.), так и в от-

дельных учебных заведениях (Серпуховское высшее военное командно-инженерное училище ракетных войск им. Ленинского комсомола). Во многих выступлениях звучала мысль о том, что позиция Гособразования СССР должна быть более прагматичной, «приземленной» на нашу сегодняшнюю действительность — ведь тень фирмы IBM часто проскальзывает в выступлениях его руководителей. Все мы понимаем, что IBM PC лучше, чем БК-0011, но какой из этих компьютеров может производиться в достаточных для школы количествах?

Говорили счастливые владельцы и об опыте работы с КУВТ УКНЦ.01 — это, безусловно, шаг вперед по сравнению с КУВТ-86. Однако производство нового комплекса пока не санкционировано Гособразованием СССР: его представители потребовали включения в протокол совещания пункта о дополнительной опытной эксплуатации КУВТ УКНЦ.01 и техническом заключении своих специалистов.

Все-таки участники совещания настроены оптимистично: выпуск БК продолжается, идет работа над БК-100, прорабатывается совместимость БК-0011 и IBM PC, но это уже более далекие перспективы. Хорошо бы за всеми этими проблемами не забыть о рядовом пользователе и продолжать выпуск БК-0010, ведь новые компьютеры будут намного дороже.

О. КОЗЛОВ

Программное обеспечение для КУВТ «Корвет»

Программно-методический комплект (ПМК) «Корвет-89» по основам информатики и вычислительной техники для работников народного образования предназначен для учителей и учащихся средних школ, работников народного образования. Курс знакомит слушателей с основами информатики и вычислительной техники в объеме 36 ч и служит для обеспечения формирования положительного отношения к вычислительной технике и навыков работы на персональных электронно-вычислительных машинах (ПЭВМ).

ПМК включает:

1) методические рекомендации для пользователя, содержащие тематический план, лекционный материал по ряду тем, описание программ и методические рекомендации по использованию каждой программы;

2) пакет прикладных программ, состоящий из 37 игровых, демонстрационных и обучающих программ, размещенных на одной дискете. Язык реализации программ — Бейсик.

Тематический план курса включает семь тем (см.: Информатика и образование. 1986. № 3). По каждой теме имеется набор программ, которые в методических рекомендациях используются для раскрытия каждой темы курса.

В пакете имеются программы, демонстрирующие использование ПЭВМ в учебном процессе. Эти программы можно использовать на уроках по общеобразовательным дисциплинам: истории, литературе, русскому и английскому языкам, физике, математике, химии. Методические материалы по программированию на языке Бейсик позволяют пользователям в отведенное курсом время освоить этот язык программирования.

Основная форма обучения — практические занятия, на которые отводится 30 ч.

Заказ на ПМК можно сделать по адресу: 630098, г. Новосибирск, ул. Приморская, 22, НИИ ИВТ АПН СССР, лаборатория № 14.

Стоимость ПМК для школ — 40 руб., для других организаций — 100 руб.

В. ТИХОНОВ

Теория ошибок

Ошибки так же неисчерпаемы, как и атом.

Аксиома. В любой программе есть ошибки.

Закон пропорциональности. Чем более программа необходима, тем больше в ней ошибок.

Следствие. Ошибок не содержит лишь совершенно ненужная программа.

Фундаментальный закон теории ошибок. На ошибках учатся.

Следствие 1. Программист, написавший программу, становится ученым.

Следствие 2. Чем больше программист делает ошибок, тем быстрее он становится ученым.

Следствие 3. Крупный ученый-программист никогда не пишет правильные программы.

Замечание. На то он и ученый.

Указание начинающему программисту. Если вы с первого раза сумели написать программу, в которой транслятор не обнаружил ни одной ошибки, сообщите об этом системному программисту. Он исправит ошибки в трансляторе.

Закон находимости ошибок. Программист может обнаружить ошибку только в чужой программе.

Следствие. Ошибке не все равно, кто ее обнаружит.

Совет начинающему программисту. Никогда не исправляйте найденные ошибки, ибо это повлечет за собой появление неизвестного числа ненайденных. Лучше опишите их в сопроводительной документации как особенность программы.

Определение. Будем называть языком ошибок правила, в обход которых пишутся программы.

Язык ошибок

Ошибки могут следовать друг за другом.

От перестановки двух эквивалентных ошибок результат не меняется (коммутативность эквивалентных ошибок).

Две последовательные ошибки можно объединить в одну, более сильную.

Одинаковые ошибки необязательно делать каждый раз, достаточно сделать одну, а затем обращаться к ней по мере необходимости из любого места программы.

Ошибки могут образовывать циклы. Наиболее устойчивый из них — бесконечный.

Ошибки могут вызывать друг друга и сами себя (рекурсивность ошибок).

Ошибки допускают многократное вложение друг в друга. Две одинаковые вложенные ошибки называются четной ошибкой и ошибкой не являются.

Свойство четности ошибок. Если написанная программа сработала правильно, то это значит, что во время ее работы выполнилось четное число ошибок или программист не понял задание.

Формулировка вышеприведенного свойства, предназначенная для политиков. Ошибка, повторенная дважды, перестает быть ошибкой.

Взаимодействие ошибок с базовой операционной системой

Во время исполнения ошибки имеют наивысший приоритет. Прервать исполнение ошибки может только другая, более активная ошибка.

Запросы операционной системы к ошибкам ошибками могут игнорироваться.

Запросы ошибок к операционной системе игнорироваться не могут.

При работе с файлами ошибки могут пользоваться файловой системой базовой ОС и ее ошибками.

На ЭВМ с параллельной архитектурой может выполняться несколько ошибок одновременно.

Системные программы

Системные программы облегчают процесс написания прикладных программ и их ошибок.

Определение. Тестирование — это процесс нахождения ошибок в тесте.

Хороший тест должен содержать ошибки, компенсирующие их нехватку в тестируемой программе.

Языковый редактор, призванный уберечь программиста от синтаксических ошибок, позволяет вносить в программу весьма хитроумные ошибки, которые не удается обнаружить ни транслятором, ни отладчиком. Обычный текстовый редактор таких возможностей не предоставляет.

Программа-транслятор, предназначенная для перевода программ с языка высокого уровня на машинный язык, при переводе порождает ошибки. Ошибки, которые содержались в исходном описании, переводятся безошибочно.

Заключительный совет тем, кто до него добрался. До начала работы над проектом следует тщательно продумать все необходимые ошибки и связи между ними. Это значительно упростит работу над ошибками в самом проекте.

Электронная матрешка

Энергичные конструкторские изыскания в области компьютерных архитектур привели к появлению мощных однокристальных ЭВМ — транспьютеров. Компактность, высокая производительность и надежность — все это стало возможным благодаря появлению новых субмикронных технологий.

32-разрядный транспьютер T800, например, за одну секунду может выполнить 10 млн. команд. Кроме того, T800 выполняет операции над числами с плавающей точкой со скоростью 1,5 млн. операций в секунду. Столь высокая производительность позволяет ему успешно соперничать с такими процессорными связками, как 80386+80387 фирмы «Интел» или 68020+68881 фирмы «Моторола».

Но, наверное самым замечательным свойством транспьютеров является то, что их легко можно объединять в комплексы. Причем увеличение числа транспьютерных узлов в такой системе приводит к

практически пропорциональному увеличению ее суммарной мощности. Так, всего семь T800 позволяют получить вычислительную мощность суперкомпьютера «Крэй 1С». А используя стандартную плату с 10 транспьютерами T800, можно довести производительность настольной персоналки IBM PC до 150 операций над числами с плавающей точкой в секунду (Мфлопс), что эквивалентно мощности 150 ЭВМ VAX 11/780.

По утверждению разработчиков, единственное, что может ограничить суммарную мощность транспьютерных комплексов,— это финансовые возможности их потребителей.

Видеопринтер

У любого приличного персонального компьютера есть клавиша, нажав на которую, можно получить твердую копию изображения с экрана дисплея на бумаге. А как быть тем, у кого такой клавиши нет или компьютер не персональный? Выход предлагают разработчики термовидеопринтера TPG-4300. Это устройство позволяет получать полноцветные копии форматов A (11) и B (12) с любого электронно-лучевого монитора с разрешением 1280×1024 точки. На экран ЭЛТ монитора накладывается прозрачный детектор, а собранная им информация поступает в принтер, который распечатывает ее с разрешением 300 точек на дюйм.

Принтер во время печати может селектировать любой цвет, инвертировать изображение, выровнять его тон, увеличить изображение оригинала до 16 раз с шагом 0,1, повернуть изображение на 90 градусов, отцентрировать его или расположить в любом месте поля, ну и, конечно, сделать отличную копию.

Сервисный центр при ГДИВЦ ВДНХ СССР
проводит техническое обслуживание оборудования фирмы

HEWLETT-PACKARD

на всей территории Советского Союза за рубли.

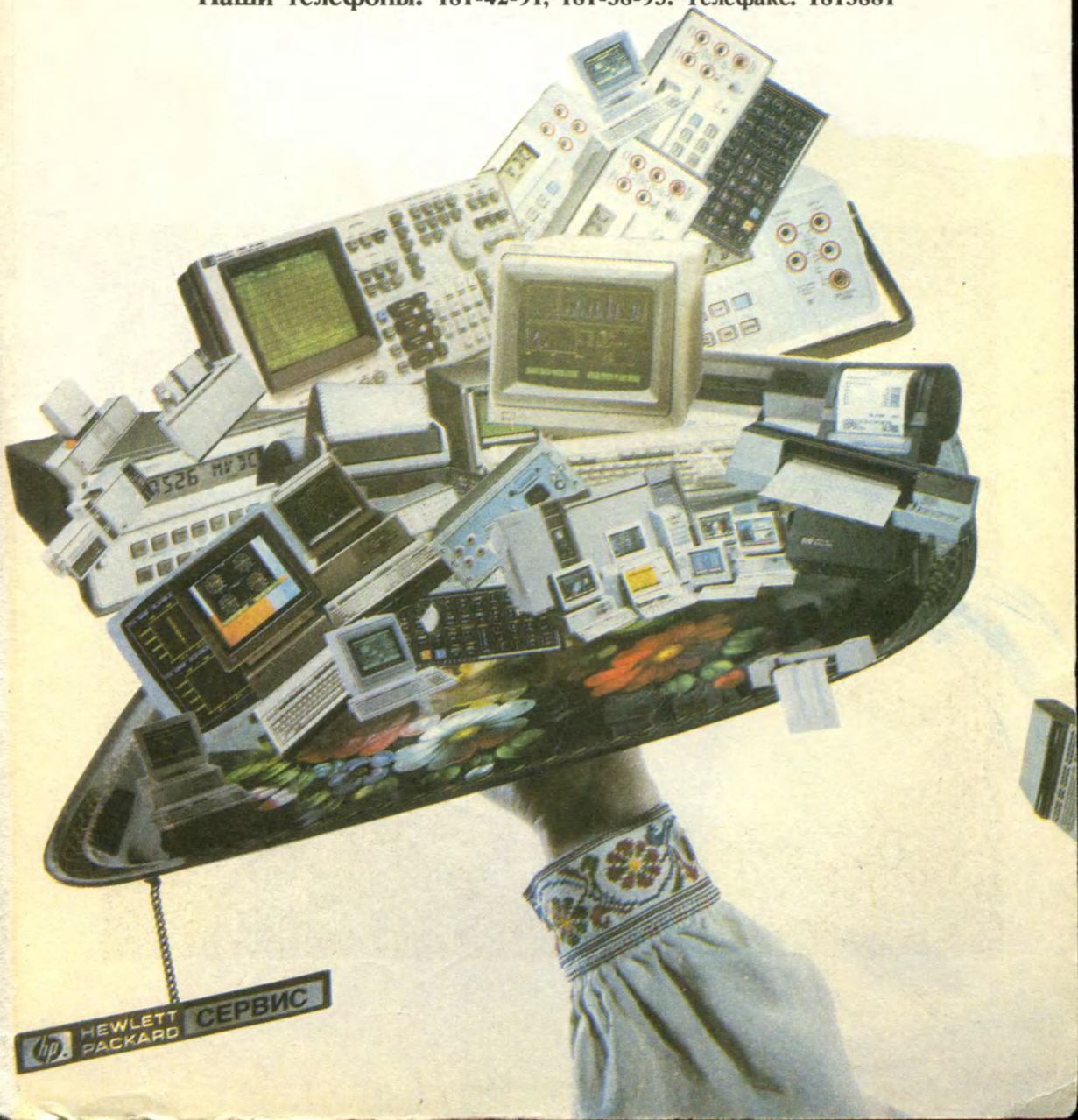
ТРИ ГОДА

вы не будете знать проблем
с запасными частями — фирма поставляет их

БЕСПЛАТНО

Если вам необходимы консультации
или первоначальный тренинг — мы также готовы помочь.

Наш адрес: 129223 Москва, проспект Мира, ВДНХ СССР, ГДИВЦ
Наши телефоны: 181-42-91, 181-38-95. Телефакс: 1813881

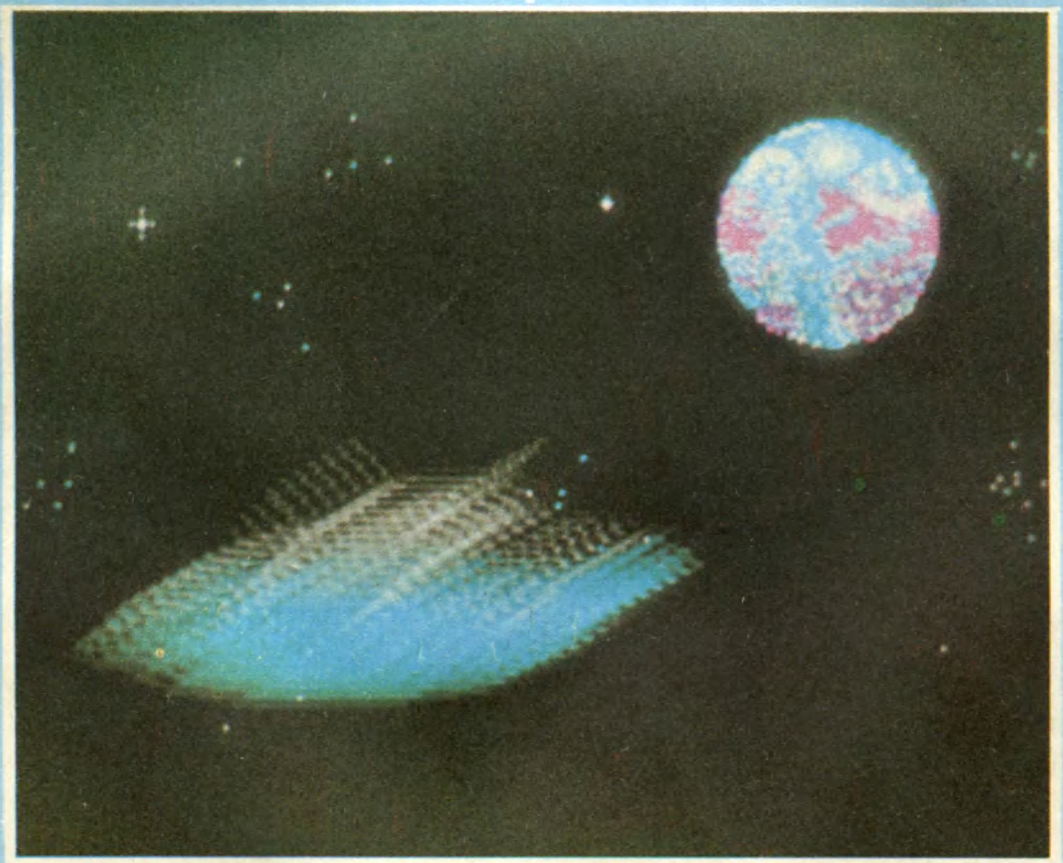


Цена 60 коп.
70423

ИНФО

2 '90

ИНФОРМАТИКА
И ОБРАЗОВАНИЕ



ISSN 0234—0453. Информатика и образование, 1990, № 2, 1—128.